I

Principes Relationnels et Concepts Oracle

Objectifs

A la fin de ce chapitre, vous saurez :

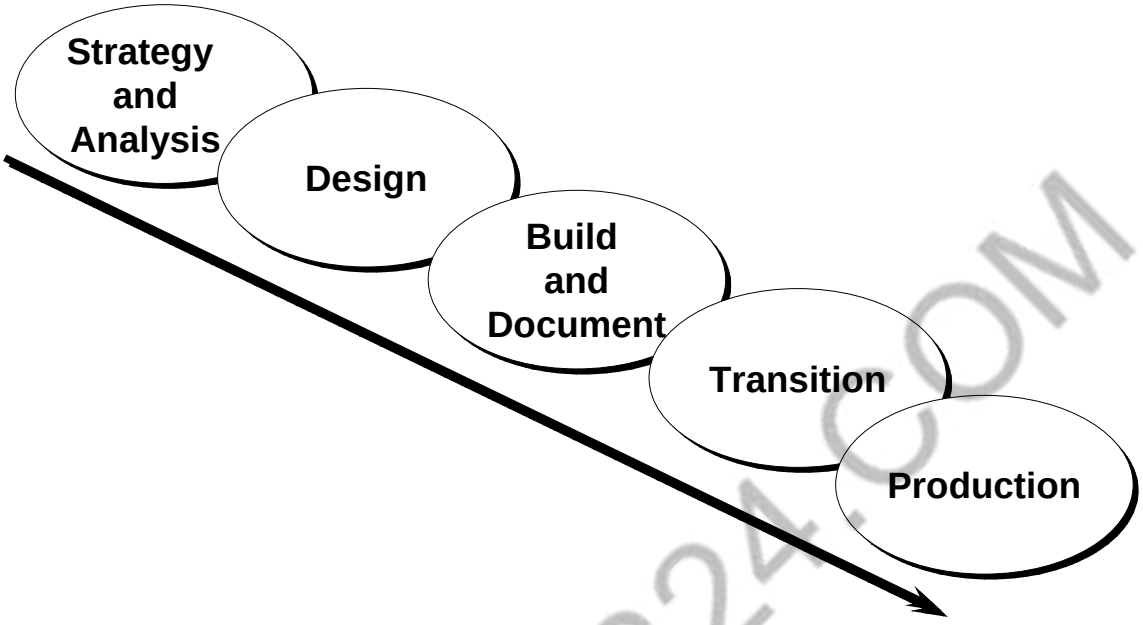
- **Décrire les phases du cycle de vie d'un système**
- **Décrire les aspects théoriques d'une base de données relationnelle**
- **Décrire l'implémentation Oracle des SGBDR et SGBDRO**
- **Décrire l'utilisation de SQL dans les produits Oracle**

I-2

Objectifs

Au cours de ce chapitre, vous allez étudier les principes de base des systèmes de gestion de bases de données relationnelles (SGBDR) et des systèmes de gestion de bases de données relationnelles objet (SGBDRO).

Cycle de Vie d’un Système



Cycle de Vie d’un Système

Il est possible d'utiliser la méthode du cycle de vie d'un système pour développer une nouvelle base de donnée.Ce cycle est constitué de différentes étapes allant de la stratégie à la production. Cette approche systématique permet de traduire des besoins d'informations de gestion en une base de données opérationnelle.

Strategy and Analysis

- Etude et analyse des besoins. Consultation des utilisateurs et responsables en vue d'identifier les besoins en informations. Intégration des divers rapports d'entreprise et comptes-rendus de mission, ainsi que des éventuelles spécifications futures du système.
- Création de plusieurs modèles du système. Transformation des spécifications écrites en une représentation graphique des besoins et règles d'information de gestion. Révision et confirmation de certains éléments du modèle avec les analystes et les experts.

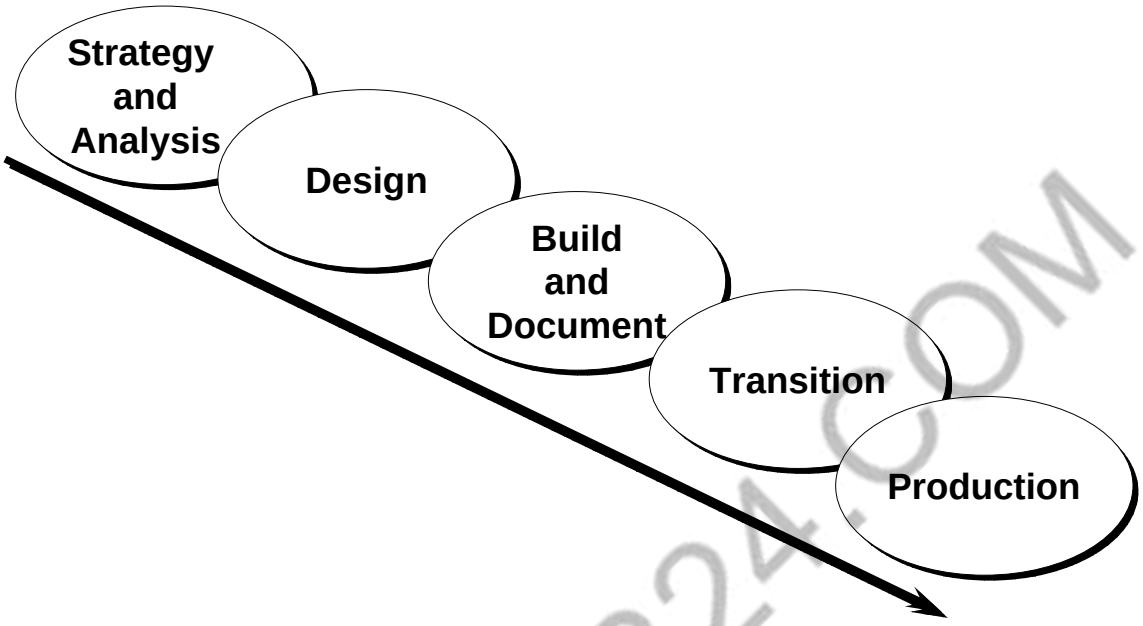
Design

Définition de la base de données à partir du schéma conceptuel des données développé au cours de la phase préédente.

Build and Document

- Génération des commandes de création des tables et des objets de la base de données.
- Développement de la documentation utilisateur, des textes d'aide et des manuels destinés à faciliter l’utilisation et le maniement du système.

Cycle de Vie d'un Système



Transition

Passage en production impliquant les tests d'acceptation utilisateur, la conversion des données existantes et les opérations parallèles. Ajout des modifications nécessaires.

Formation des utilisateurs finaux.

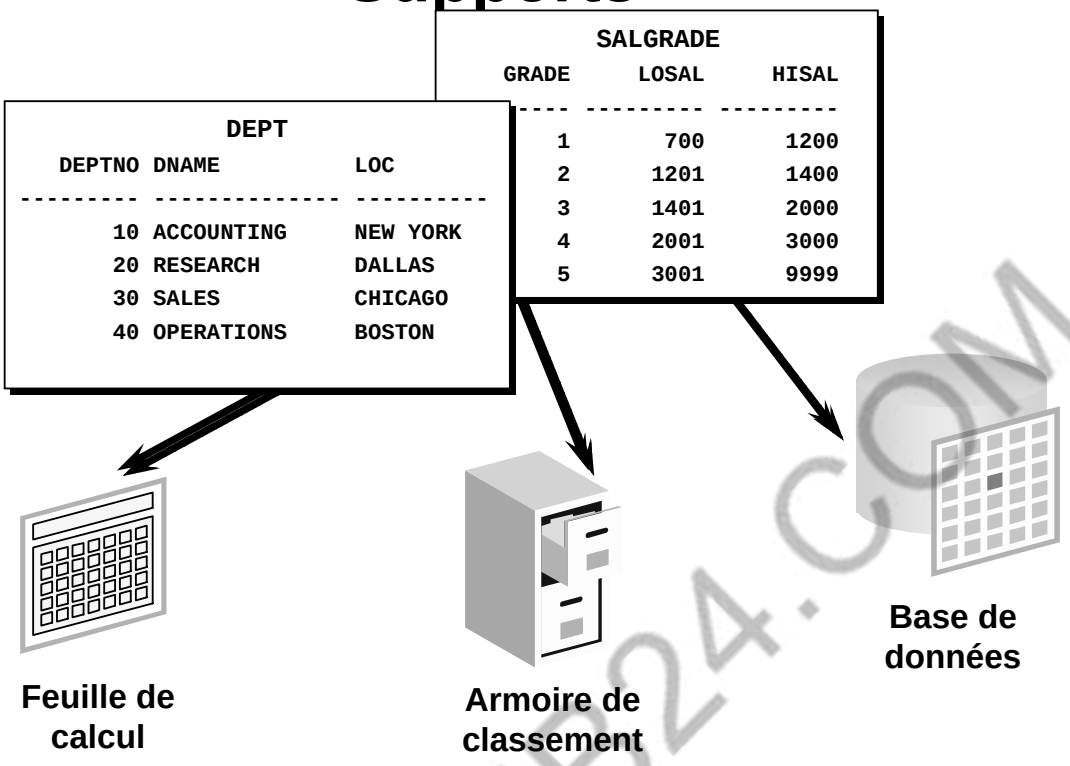
Production

Livraison du système aux utilisateurs et mise en oeuvre. Contrôle des performances et amélioration du système.

Maintenance du système.

Remarque : le cycle de vie d'un système est itératif. Ce cours se concentre sur la phase 'Build and Document'.

Stockage de Données sur Différents Supports



I-5

Stockage de l'Information

Chaque organisation a des besoins particuliers en matière d'information. Par exemple, une bibliothèque tient un registre de ses adhérents, de ses livres, des dates de prêt et des amendes. Une entreprise doit conserver des informations sur ses départements, ses employés et leur salaire. Tous ces éléments d'information sont appelés *données*.

Il est possible de stocker ces données sur différents supports et dans différents formats, par exemple sur un document papier, dans une armoire ou dans une feuille de calcul ou bien encore dans une base de données.

Une *base de données* est un ensemble organisé d'informations.

Pour gérer une base de données, il faut un système de gestion de bases de données (SGBD). Un SGBD est un programme qui permet de stocker, d'extraire et de modifier des données à la demande dans la base de données. Il existe cinq types principaux de bases de données : *hiérarchique*, *réseau*, *relationnel*, *relationnel objet* et *objet*.

Remarque : Oracle7 est un système de gestion de bases de données relationnelles et Oracle8 un système de gestion de bases de données relationnelles objet.

Principe d'une Base de Données Relationnelle

- **En 1970, Dr E. F. Codd propose le modèle relationnel pour les systèmes de bases de données.**
- **Est à la base des systèmes de gestion de bases de données relationnelles (SGBDR).**
- **Composants du modèle relationnel :**
  - **Collection d'objets appelés encore relations**
  - **Ensemble d'opérateurs pour agir sur les relations**
  - **Règles d'intégrité pour garantir exactitude et cohérence des données**

I-6

Le Modèle Relationnel

C'est le Dr. E. F. Codd qui, en juin 1970, a présenté pour la première fois les principes du modèle relationnel pour les bases de données dans un article intitulé "A Relational Model of Data for Large Shared Data Banks".

A cette époque, les modèles les plus utilisés étaient les modèles hiérarchique et réseau, voire de simples structures séquentielles. Les systèmes de gestion de bases de données relationnelles (SGBDR) furent rapidement adoptés en raison, notamment, de leur simplicité d'utilisation et de la flexibilité de leur structure. En outre, plusieurs fournisseurs novateurs, tels que Oracle, ont complété les SGBDR en leur adjoignant une série de puissants outils de développement d'applications et produits utilisateur, offrant ainsi une solution complète.

Composants du Modèle Relationnel

- Collections d'objets appelés encore relations pour stocker les données
- Ensemble d'opérateurs agissant sur les relations afin de produire d'autres relations
- Des règles d'intégrité pour garantir l'exactitude et la cohérence des données

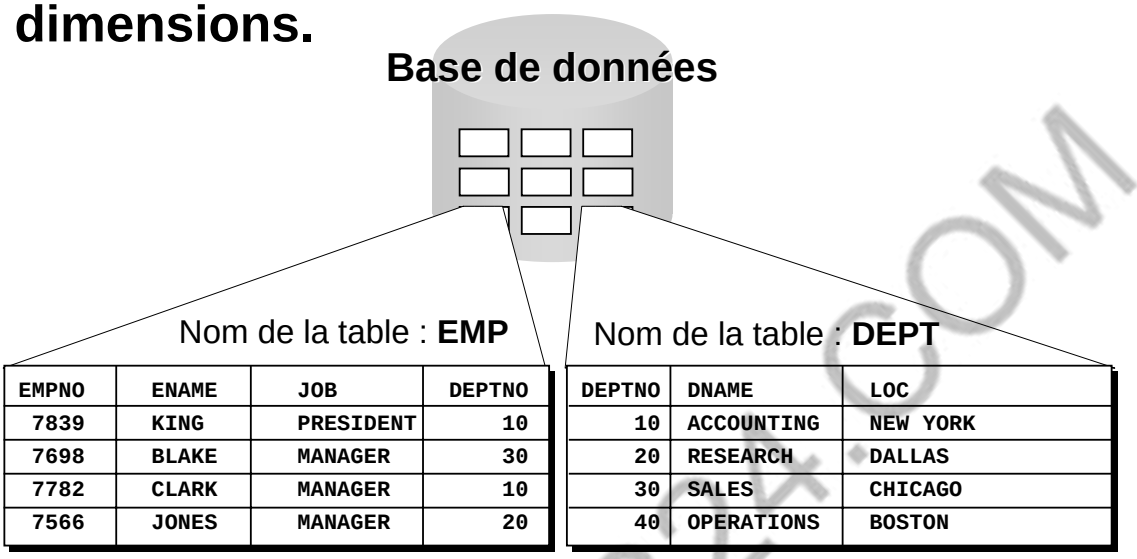
Pour plus d'informations, reportez-vous à

E. F. Codd*, The Relational Model for Database Management Version 2* (Reading, Mass. :

Addison-Wesley, 1990).

Définition

Une base de données relationnelle est un ensemble de relations ou tables à deux dimensions.



EMPNO	ENAME	JOB	DEPTNO
7839	KING	PRESIDENT	10
7698	BLAKE	MANAGER	30
7782	CLARK	MANAGER	10
7566	JONES	MANAGER	20

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

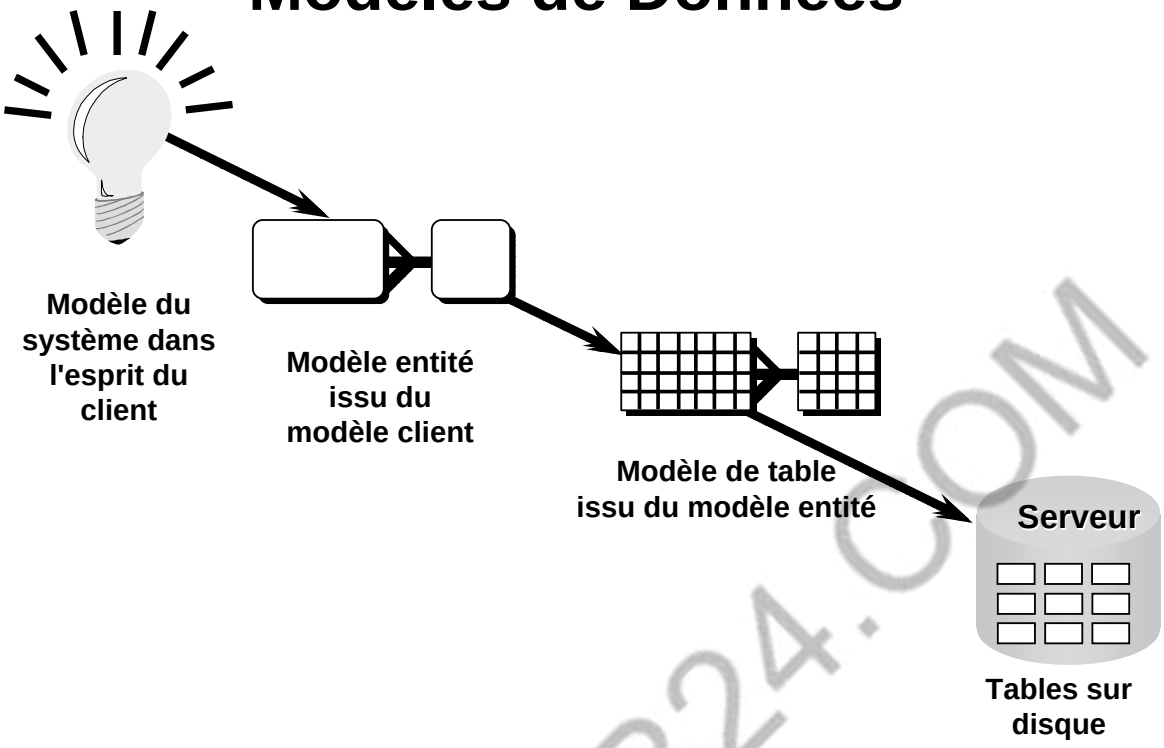
I-7

Base de Données Relationnelle

Une base de données relationnelle stocke l'information au moyen de relations ou tables à deux dimensions.

Supposons, par exemple, que vous souhaitiez stocker toutes les informations concernant les employés de votre société dans une base de données relationnelle. Vous pourriez créer plusieurs tables pour stocker différents éléments d'information sur vos employés, telles que la table des employés, la table des départements, et la table des salaires.

Modèles de Données



I-8

Modèles de Données

Toute conception repose sur un modèle. Ainsi, les ingénieurs automobile commencent par concevoir un modèle de voiture, afin d'en vérifier tous les détails avant de lancer la production. De la même manière, les concepteurs de systèmes développent des modèles pour tester les nouvelles idées et améliorer la conception des bases de données.

Rôle des Modèles

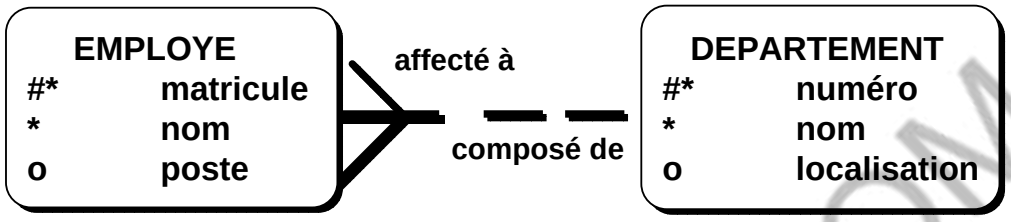
Les modèles facilitent l'explication et la compréhension des concepts. On peut les utiliser pour :

- Communiquer
- Classifier
- Décrire
- Spécifier
- Rechercher
- Evoluer
- Analyser
- Imiter

L'objectif est de produire un modèle adapté au maximum de ces usages, compréhensible par un utilisateur final et suffisamment précis pour permettre au développeur de créer un système de base de données.

Modèle Entité-Relation

- **Création d'un schéma entité-relation à partir de règles de gestion ou de comptes-rendus d'interviews.**



Scénario

- **". . . Affecter un ou plusieurs employés à un département . . ."**
- **". . . Certains départements n'ont pas encore d'employés qui leur soient affectés. . ."**

I-9

Modélisation ER (Entité-Relation)

Dans un système réel, les données sont divisées en catégories discrètes ou entités. Un modèle *ER* est la représentation des diverses entités qui existent au sein d'une société et de leurs interrelations. Un modèle ER est issu de règles de gestion ou de compte-rendus d'interview et est conçu au cours de la phase d'analyse du cycle de vie du système. Dans les modèles ER, l'information nécessaire à une société est séparée des activités de cette société. En effet, même si l'activité change, le type d'information, lui, a tendance à ne pas varier. Ainsi les structures de données tendent elles aussi à être constantes.

Avantages de la Modélisation ER

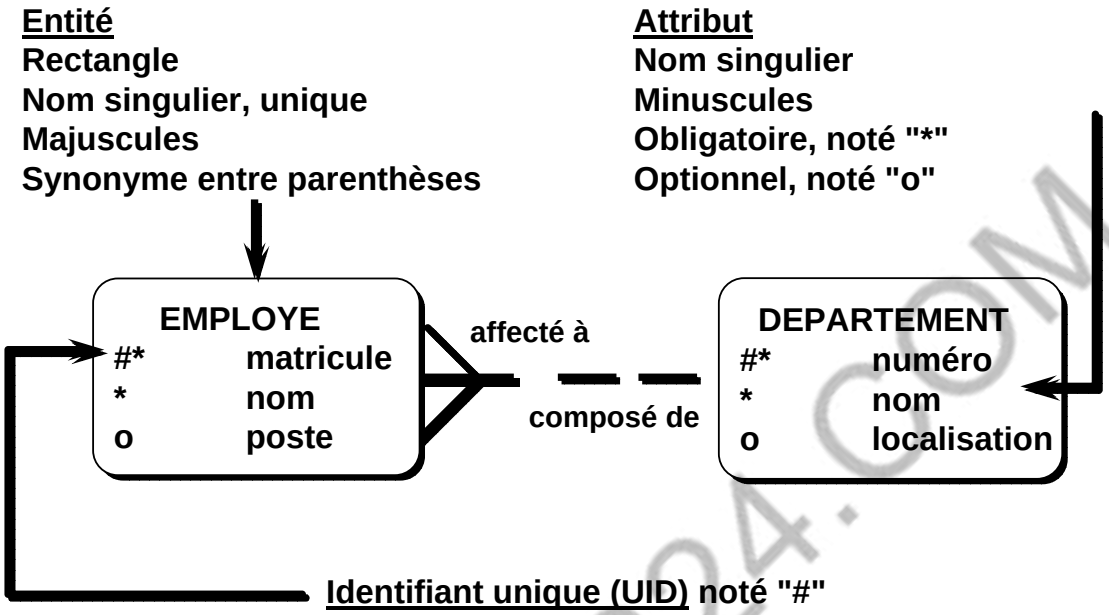
- Documente l'information dans un format clair et précis.
- Offre une idée précise de l'étendue des besoins d'information.
- Fournit une représentation graphique simple pour la conception de la base de données.
- Offre un cadre efficace pour l'intégration de plusieurs applications.

Les Composants Clés

- Entité : Elément d'une organisation, concret ou abstrait, identifiable, porteur d'informations et important pour l'activité considérée. Exemple : les départements, les employés, les commandes d'une société.
- Attribut : élément qui décrit ou qualifie une entité. Par exemple, pour l'entité employé, les attributs peuvent être le matricule, le nom, l'intitulé du poste, la date d'embauche et le numéro de département, etc., de l'employé. Chaque attribut est optionnel ou obligatoire.
- Lien : association entre entités ayant un nom et caractérisée par son caractère obligatoire ou optionnel et son degré. Exemple : employés et départements, commandes et articles.

M. Hanoune

Conventions de Modélisation selon le Modèle Entité-Relation



I-10

Entités

Pour représenter une entité dans un modèle ER, on utilise les conventions suivantes :

- Rectangle de dimensions quelconques
- Nom unique et au singulier
- Nom en majuscules
- Synonymes optionnels, en majuscules et entre parenthèses : ( )

Attributs

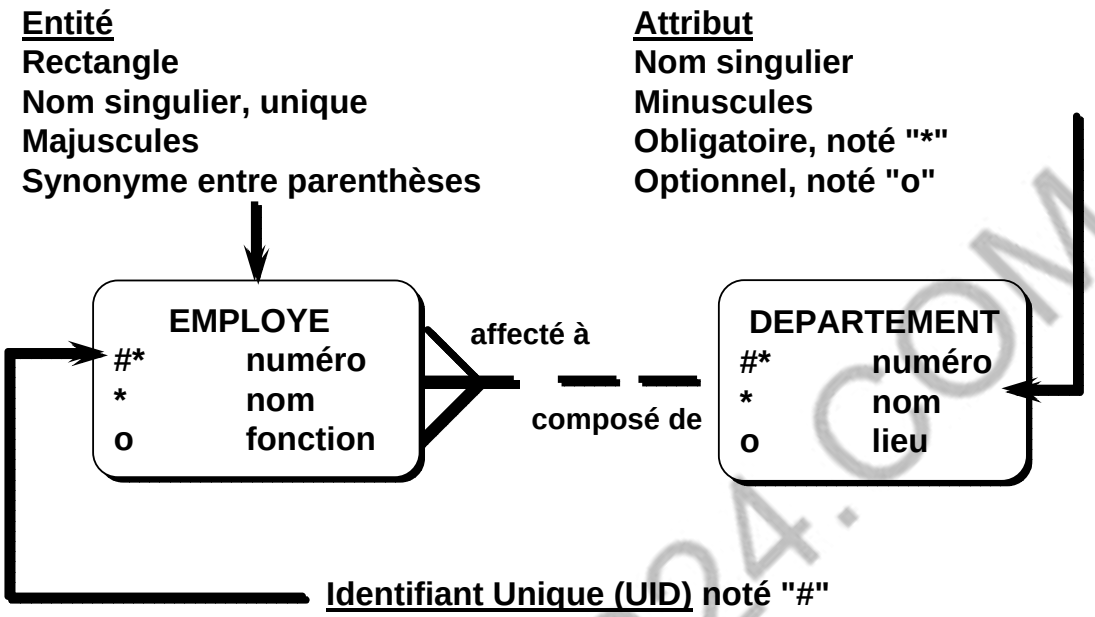
Symbole	Description
Ligne discontinue	Elément optionnel signifiant "peut être"
Ligne continue	Elément obligatoire signifiant "doit être"
Patte d'oie	Elément de degré signifiant "un ou plus"
Trait	Elément de degré signifiant "un et seulement un"

Pour représenter un attribut dans un modèle, on utilise les conventions suivantes :

- Noms au singulier et en minuscules
- Astérisque pour baliser les attributs obligatoires ou les valeurs indispensables :*
- Lettre o pour baliser les attributs optionnels ou les valeurs non indispensables

M. Hanoune

Conventions de Modélisation selon le Modèle Entité-/Relation



I-11

Relations

Dans chaque sens, un lien comporte :

- Une désignation, par exemple, *enseigné par* ou *affecté à*
- Un caractère obligatoire ou facultatif : *doit être* ou *peut être*
- Un degré : *un(e) et un(e) seul(e)* ou *un(e) ou plus*

Remarque : le terme cardinalité est synonyme de degré.

Chaque entité source {peut être | doit être} désignation du lien {un(e) et un(e) seul(e) | un(e) ou plus} entité cible.

Remarque : la convention se lit dans le sens des aiguilles d'une montre.

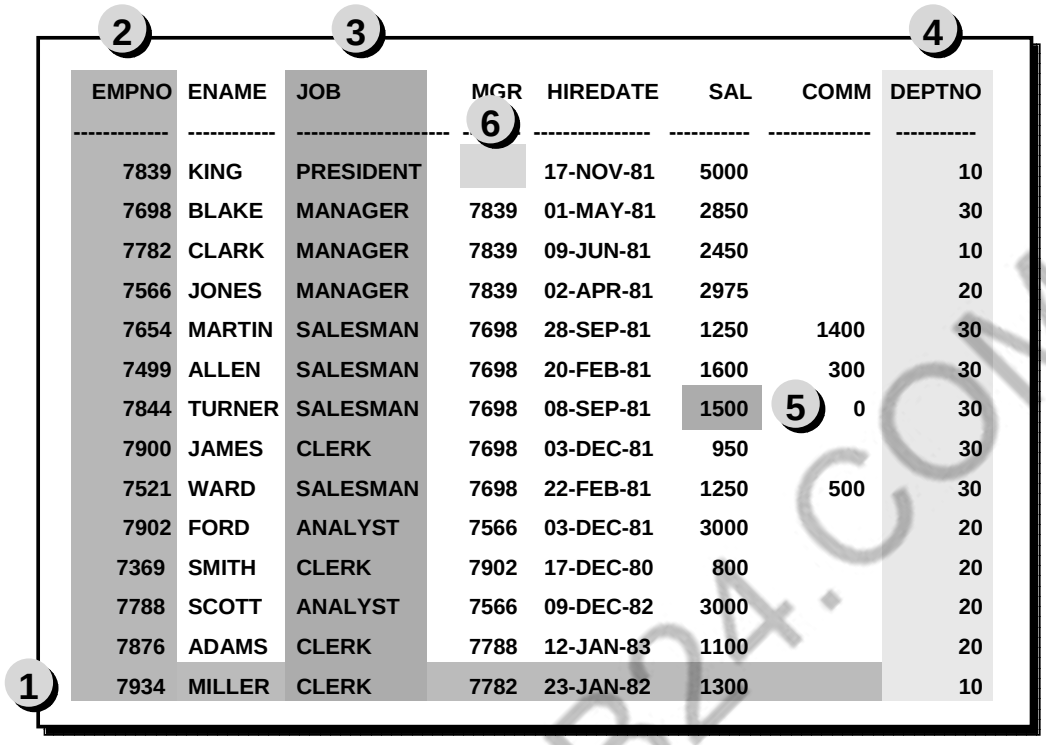
Identifiant Unique

Un identifiant unique (UID) est une combinaison d'attributs ou de liens ou des deux, permettant de distinguer les différentes occurrences d'une entité. Chaque occurrence d'une même entité doit être identifiée de façon unique.

- Balisage de chacun des attributs de l'UID avec le symbole numérique : #

Remarque : si vous souhaitez obtenir plus d'informations sur la modélisation ER, vous pouvez assister aux cours de modélisation.

Terminologie des Bases de Données Relationnelles



EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		17-NOV-81	5000		10
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7902	FORD	ANALYST	7566	03-DEC-81	3000		20
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7788	SCOTT	ANALYST	7566	09-DEC-82	3000		20
7876	ADAMS	CLERK	7788	12-JAN-83	1100		20
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

I-12

Terminologie des Bases de Données Relationnelles

Une base de données relationnelle peut contenir une ou plusieurs tables. La *table* est la structure de stockage élémentaire d'un SGBDR. Elle contient toutes les données nécessaires relatives à des éléments du monde réel, par exemple, des employés, des factures, des clients.

La diapositive ci-dessus montre le contenu de la *table* ou *relation* EMP. Les numéros désignent les éléments suivants :

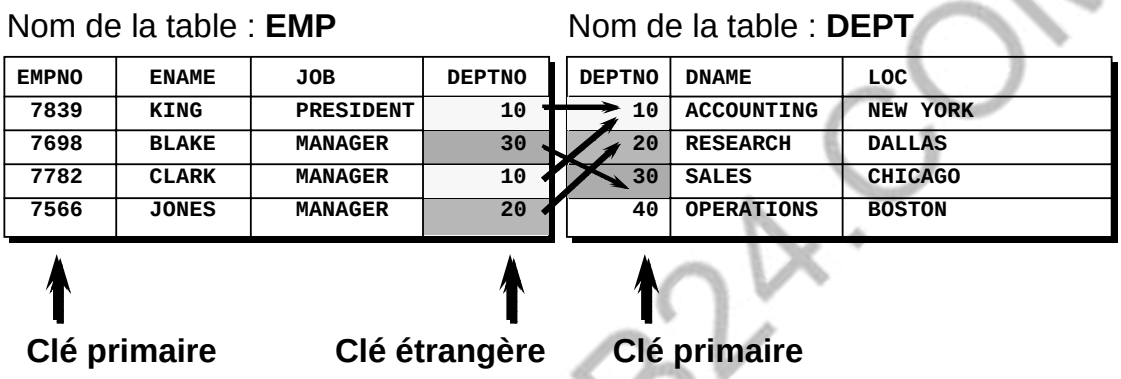
1. Une *ligne unique* ou *tuple* indiquant toutes les données concernant un employé particulier. Chaque ligne de la table doit être identifiée par une clé primaire, ce qui permet d'éviter les doublons. L'ordre des lignes n'a pas d'importance ; il peut être spécifié lors de l'extraction des données.
2. Cette *colonne* ou *attribut* contenant le matricule des employés, est la clé primaire. Le matricule identifie un employé *unique* dans la table EMP. La colonne clé primaire doit obligatoirement être renseignée.
3. Cette colonne ne contient pas de valeur clé. Dans une table, une colonne représente un type de données ; ici, il s'agit des intitulés de postes de tous les employés. L'ordre des colonnes n'a pas d'importance pour le stockage des données ; il peut être spécifié lors de l'extraction des données.
4. Cette colonne contenant le numéro du département fait également office de *clé étrangère*. Une clé étrangère est une colonne qui définit la manière dont les tables sont liées entre elles. Elle fait référence à une clé primaire ou à une clé unique d'une autre table. Dans l'exemple, DEPTNO identifie de façon *unique* un département de la table DEPT.
5. Un *champ* se situe à l'intersection d'une ligne et d'une colonne. Il ne peut contenir qu'une seule valeur.
6. Un champ peut ne contenir aucune valeur. On parle alors de *valeur NULL*. Dans la table EMP, seules les lignes des employés ayant le statut de vendeur contiennent une valeur dans le champ COMM (commission).

Remarque : les valeurs NULL sont traitées plus en détails dans les chapitres suivants.

M. Hanoune

Relier Plusieurs Tables

- **Chaque ligne de données d'une table est identifiée de manière unique par une clé primaire (PK).**
- **Les données de plusieurs tables peuvent être liées logiquement à l'aide de clés étrangères (FK).**



I-13

Liaison de Plusieurs Tables

Chaque table contient des données qui décrivent une et une seule entité. La table EMP, par exemple, contient des données sur les employés. Un format de table permet immédiatement de visualiser, comprendre et utiliser l'information.

Les données concernant chaque entité distincte étant stockées dans différentes tables, il peut être nécessaire de combiner deux ou plusieurs tables afin de répondre à une question particulière. Supposons, par exemple, que vous vouliez savoir où se situe le département d'un employé. Vous avez besoin pour cela des données de la table EMP (contenant des informations sur les employés) et de la table DEPT (contenant des informations sur les départements). Un SGBDR permet de lier les données de deux tables au moyen de la clé étrangère. La clé étrangère est une colonne ou un ensemble de colonnes faisant référence à une clé primaire de la même table ou d'une autre table.

La possibilité de relier les données d'une table à celles d'une autre table permet d'organiser l'information en unités séparées et faciles à gérer. Ainsi, vous pouvez séparer de façon logique les données concernant les employés de celles concernant les départements, en les stockant dans des tables différentes.

Règles concernant les Clés Primaires et Etrangères

- La clé primaire ne doit comporter aucun doublon.
- La clé primaire ne peut généralement pas être modifiée.
- Les clés étrangères sont basées sur des valeurs des données. Ce sont des pointeurs purement logiques et non physiques.
- Une valeur de clé étrangère doit renvoyer à une valeur de clé primaire ou unique existante, ou bien à une valeur NULL.

Propriétés des Bases de Données Relationnelles

Une base de données relationnelle :

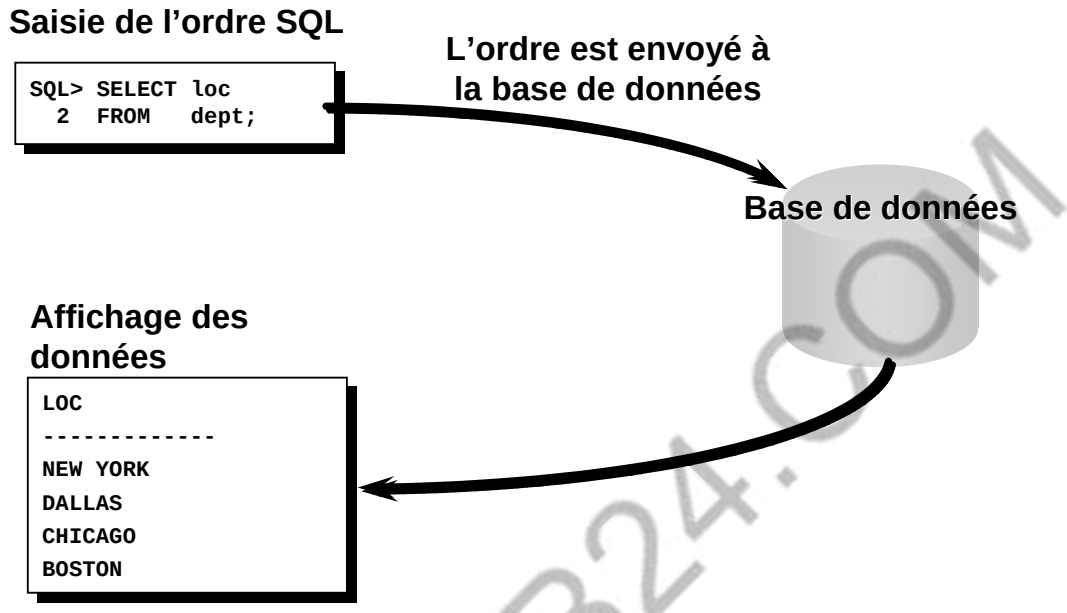
- **Peut être consultée et modifiée à l'aide d'ordres SQL (Structured Query Language)**
- **Contient une collection de tables sans pointeurs physiques**
- **Utilise un ensemble d'opérateurs**

Propriétés des Bases de Données Relationnelles

Dans une base de données relationnelle, il est inutile d'indiquer le chemin d'accès aux tables ni de connaître l'agencement physique des données.

Pour accéder à la base de données, vous exécutez un ordre du langage SQL (Structured Query Language) qui est le langage standard ANSI des bases de données relationnelles. Il comprend un grand nombre d'opérateurs qui permettent de partitionner et combiner les relations. La base de données peut être modifiée au moyen d'ordres SQL.

Communiquer avec un SGBDR au Moyen de SQL



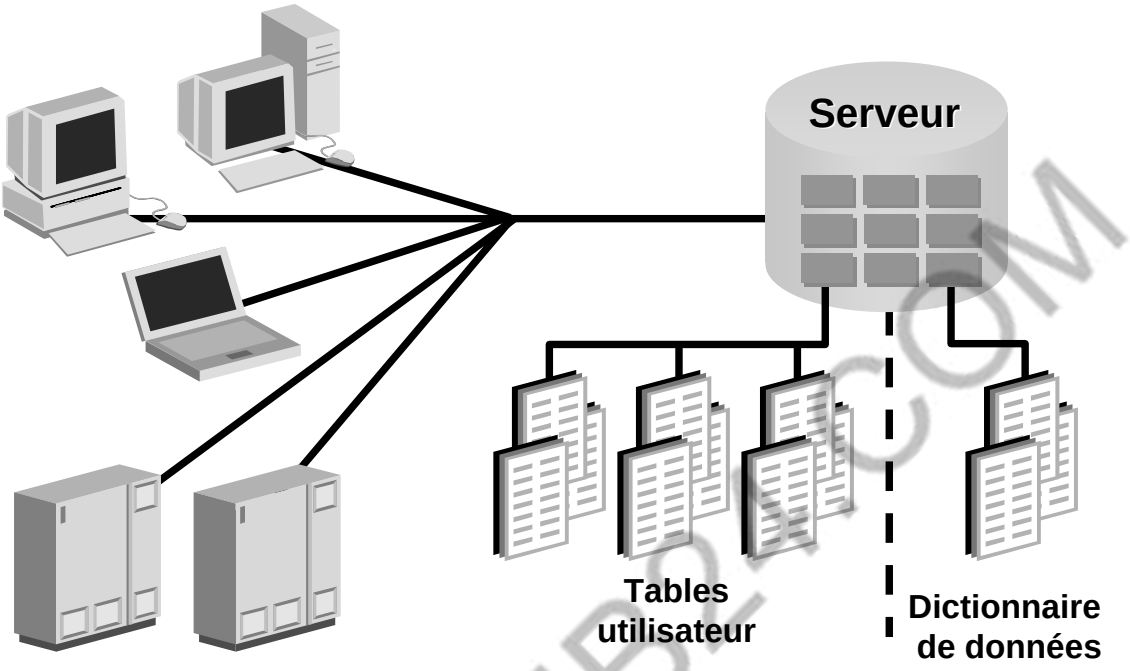
I-15

SQL

SQL permet de communiquer avec le serveur et présente plusieurs avantages. Il est :

- Efficace
- Facile à apprendre et à utiliser
- Complet sur le plan fonctionnel. SQL permet de définir, d'extraire et de manipuler des données des tables.

Système de Gestion de Bases de Données Relationnelles



I-16

Système de Gestion de Base de Données Relationnelle

Oracle offre un SGBDR flexible, Oracle Server. Ce système permet de stocker et gérer des données, avec tous les avantages que procure une structure relationnelle combinée à PL/SQL, moteur permettant de stocker et d'exécuter des unités de programme. Oracle Server permet aux utilisateurs des accès optimisés aux données. Il inclut des fonctions de sécurité qui contrôlent les accès à la base de données et son utilisation. Il comprend aussi des mécanismes de verrouillage qui assurent la cohérence et la protection des données.

Les applications Oracle peuvent être exécutées sur le même ordinateur qu'Oracle Server. Mais il est également possible d'exécuter les applications sur un ordinateur local et Oracle Server sur un autre système (architecture client-serveur). Un tel environnement client-serveur peut faire appel à de nombreuses ressources de traitement. Par exemple, une application de réservation de billets d'avion peut tourner sur un PC client, avec accès aux données de vol gérées par un système Oracle Server sur ordinateur central.

Pour plus d'information, reportez-vous à
Oracle8 Server Concepts Manual, Release 8.

Oracle8 : Système de Gestion de Bases de Données Relationnelles Objet

- **Types de données et objets définis par l'utilisateur**
- **Compatibilité relationnelle totale**
- **Support des objets de type multimedia et des large objects**
- **Fonctions de serveur de bases de données haut de gamme**

I-17

Oracle8

Oracle8 Server est la première base de données objet développée par Oracle. Les fonctions de modélisation des données d'Oracle7 Server ont été étendues pour prendre en charge le nouveau modèle de base de données relationnelle objet. Oracle8 Server fonctionne avec un nouveau moteur permettant la programmation orientée objet, les types de données et les objets de gestion complexes, ainsi qu'une compatibilité totale avec le monde relationnel.

Oracle8 Server offre de nombreuses améliorations par rapport à Oracle7 Server. Notamment, il inclut de nombreuses fonctions destinées à augmenter les performances et fonctionnalités des applications de traitement transactionnel en ligne (OLTP) comme, par exemple, un meilleur partage des structures de données lors de l'exécution, des buffers et des caches de taille supérieure, et des contraintes dont il est désormais possible de différer la vérification. Les applications de data warehouse vont bénéficier d'améliorations telles que l'exécution en parallèle des opérations d'insertion, de mise à jour et de suppression, le partitionnement et l'optimisation des requêtes en parallèle. Parce qu'il fonctionne dans le cadre de l'architecture NCA (Network Computing Architecture), Oracle8 Server est compatible avec les applications client-serveur et Internet distribuées et multi-niveaux.

Oracle8 Server accepte des dizaines de milliers d'utilisateurs simultanés, prend en charge jusqu'à 512 péta-octets et peut manipuler n'importe quel type de données (texte, spatial, image, son, image vidéo, série chronologique, mais aussi des données structurées traditionnelles).

Pour plus d'informations, reportez-vous à
Oracle8 Concepts Manual, Release 8

Définition de l'Objet

Un objet :

- **Peut être une personne, un lieu ou une chose**
- **Connaît ses caractéristiques et exécute des actions**
- **A une identité**

Je suis une pendule, je connais mon *fuseau horaire* et je peux *indiquer l'heure*

I-18

Les Objets

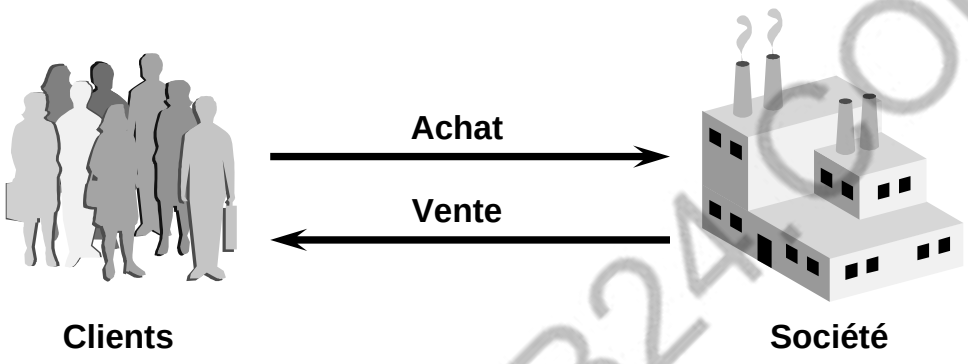
Les objets sont souvent considérés comme étant une représentation d'une chose appartenant au monde réel. On dit d'eux qu'ils "savent" faire des choses. Il est possible de "demander" à un objet employé de calculer ses charges salariales ou à un objet commande de s'auto-expédier.

Voici quelques définitions :

- "Un objet est un 'package' logiciel contenant un ensemble de procédures (méthodes) et de données (variables) associés." David Taylor, *Object-Oriented Technology: A Manager's Guide* (Reading, Mass.: Addison-Wesley, 1981)
- "Un objet est un concept, une abstraction ou une chose ayant des limites et un sens clairs pour le problème à résoudre." James Rumbaugh, et. al., *Object-Oriented Modeling and Design* (Englewood Cliffs, N.J.: Prentice-Hall, 1991)

Utilisation d'un Modèle Objet

- **Les objets modélisent le problème à résoudre.**
- **Le modèle exprime les interactions existant entre les objets**
- **Les modèles objet reflètent la réalité**



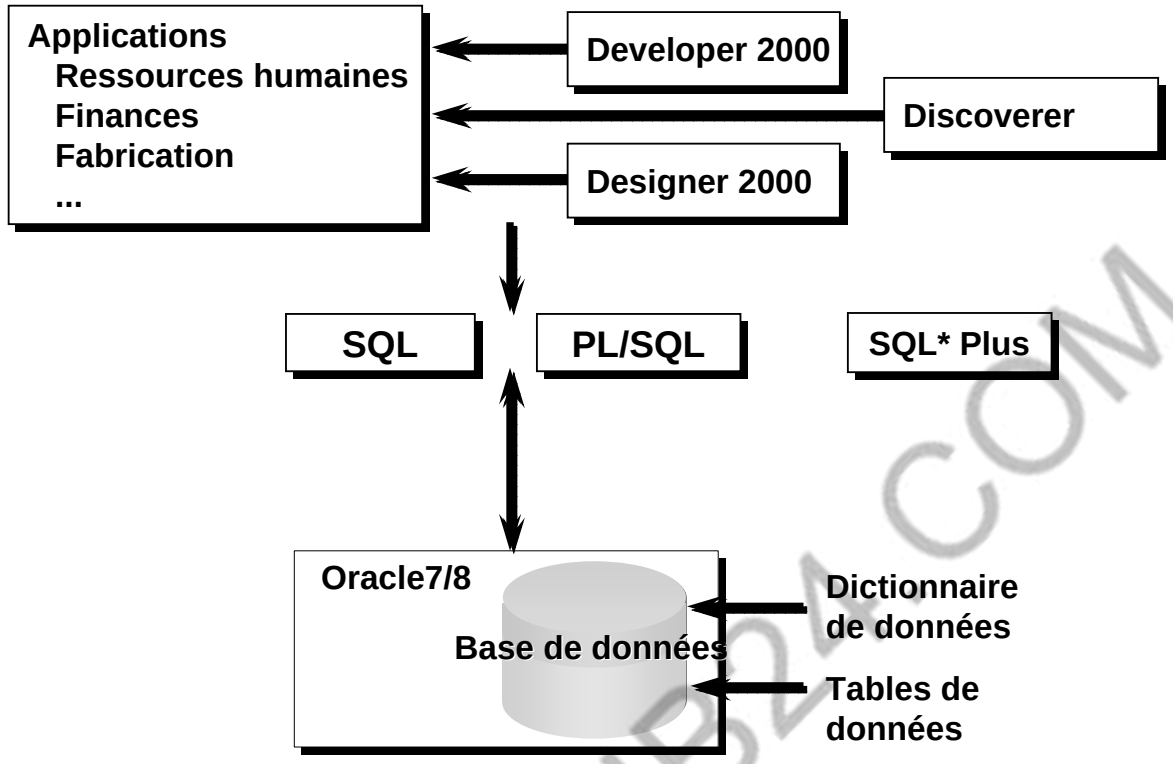
I-19

Les Modèles Objet

La technologie objet a été créée dans le but de modéliser des problèmes de gestion. Le modèle exprime les interactions qui existent entre les objets.

Lorsqu'ils travaillent avec des objets, les développeurs réfléchissent davantage en termes de besoins de l'application qu'en termes d'architecture des systèmes d'exploitation et d'exigences de l'environnement de développement.

La Solution Complète Oracle



I-20

La Solution Complète Oracle

Le système de gestion de bases de données relationnelles Oracle est le principal produit de l'offre Oracle. Il comprend Oracle Server ainsi que divers outils destinés à aider l'utilisateur pour la mise à jour, le contrôle et l'utilisation des données. Le dictionnaire de données Oracle est l'un des composants les plus importants de Oracle Server. Il est constitué d'un ensemble de tables et de vues permettant un accès en lecture seule à la base de données.

Le SGBDR gère des tâches telles que les suivantes :

- Stockage et définition des données
- Contrôle et restriction des accès aux données
- Sauvegarde et restauration
- Interprétation des ordres SQL et PL/SQL

Remarque : PL/SQL est un langage procédural qui complète les possibilités de SQL avec une logique d'application.

Les ordres SQL et PL/SQL sont systématiquement utilisés pour rechercher et manipuler les données stockées dans la base Oracle. Cependant, dans le cadre de programmes d'application, il arrive souvent que vous accédiez à la base de données sans utiliser directement SQL ou PL/SQL : vous vous contentez de cliquer sur un bouton ou de cocher une case par exemple, mais en fait, les applications utilisent implicitement SQL ou PL/SQL lorsqu'elles exécutent la requête.

SQL*Plus est un outil Oracle qui reconnaît les ordres SQL et PL/SQL et en lance l'exécution à partir du serveur. Il a son propre langage de commande.

Oracle propose une large gamme d'outils pilotés par une interface graphique (GUI) de pointe pour la conception d'applications de gestion, ainsi qu'un vaste éventail d'applicatifs destinés à de nombreux secteurs du commerce et de l'industrie.

Remarque : les dictionnaires de données Oracle sont traités plus en détails dans les chapitres suivants.

Les Ordres SQL

SELECT	**Recherche de données**
INSERT **UPDATE** **DELETE**	**Langage de manipulation des données (LMD)**
CREATE **ALTER** **DROP** **RENAME** **TRUNCATE**	**Langage de définition des données (LDD)**
COMMIT **ROLLBACK** **SAVEPOINT**	**Contrôle des transactions**
GRANT **REVOKE**	**Langage de contrôle des données (LCD)**

I-21

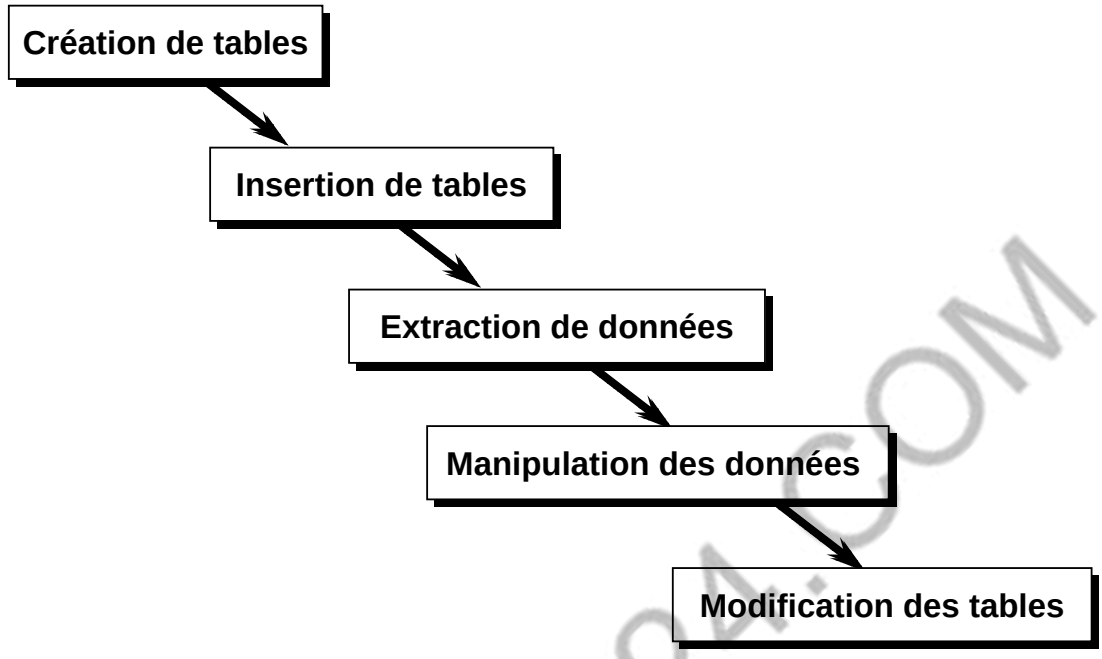
Les Ordres SQL

Le SQL d'Oracle est conforme aux standards reconnus. La société Oracle garantit la conformité future avec l'évolution des standards en faisant participer certains de ses principaux collaborateurs aux comités de normalisation SQL. Les deux organismes reconnus sont l'ANSI (American Standards Institute) et l'ISO (International Standards Organization). Tous deux ont adopté SQL comme langage standard des bases de données relationnelles.

Ordre	**Description**
SELECT	Extraction des données de la base de données.
INSERT UPDATE DELETE	Respectivement, ajout, modification et suppression de lignes dans les tables de la base de données. L'ensemble est appelé *langage de manipulation des données* (LMD).
CREATE ALTER DROP RENAME TRUNCATE	Initialisation, modification et suppression des structures de données dans une table. L'ensemble est appelé *langage de définition des données* (LDD).
COMMIT ROLLBACK SAVEPOINT	Gestion des modifications apportées par les ordres LMD. Les modifications apportées aux données peuvent être regroupées en transactions logiques.
GRANT REVOKE	Accorde ou retire les droits d'accès à la base de données Oracle et aux structures qu'elle renferme. L'ensemble est appelé *langage de contrôle des données* (LCD).

M. Hanoune

Présentation du Cours



I-22

Présentation du Cours

Ce cours comprend dix-neuf chapitres couvrant les sujets suivants :

- Création de tables
- Insertion de données
- Extraction de données
- Manipulation des données
- Modification des tables

Chaque chapitre débute par une présentation des objectifs et se termine par des exercices pratiques.

Ce cours s'attache aux aspects relationnels des systèmes de gestion de bases de données Oracle. Il ne traite pas des objets.

Vous utiliserez SQL*Plus pour saisir et exécuter des ordres SQL.

Tables Utilisées dans le Cours

EMP

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		17-NOV-81	5000		10
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	1500		10
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7902	FORD	ANALYST	7566	03-DEC-81	3000		20
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7788	SCOTT	ANALYST	7566	09-DEC-82	3000		20
7876	ADAMS	CLERK	7788	12-JAN-83	1100		20
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

DEPT

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

SALGRADE

GRADE	LOSAL	HISAL
1	700	1200
2	1201	1400
3	1401	2000
4	2001	3000
5	3001	9999

Tables Utilisées dans le Cours

Vous utiliserez principalement trois tables dans ce cours :

- La table EMP qui contient des informations sur tous les employés
- La table DEPT, qui contient des informations sur tous les départements
- La table SALGRADE , contenant des informations sur les différents niveaux de salaires en fonction de l'échelon

La structure et les données de chaque table sont données dans l'annexe B.

Résumé

- **Les bases de données relationnelles:**
  - **Sont composées de relations**
  - **Possèdent des opérateurs relationnels**
  - **Sont régies par des contraintes d'intégrité des données**
- **Oracle Server permet de stocker et gérer l'information au moyen du langage SQL.**
- **Oracle8 est fondé sur le système de gestion de bases de données relationnelles objet.**

Résumé

Les systèmes de gestion de bases de données relationnelles sont constitués de relations. Ils possèdent des opérateurs et sont régis par des contraintes d'intégrité des données.

Oracle Corporation offre des produits et services qui répondent à vos besoins en matière de systèmes de gestion de bases de données relationnelles. Le produit central, Oracle Server, permet de stocker et gérer l'information au moyen de SQL et du moteur PL/SQL pour les structures procédurales.

Oracle Server est fondé sur le standard ANSI pour SQL et comprend des extensions. SQL est le langage qui permet de communiquer avec le serveur pour accéder aux données, les manipuler et les contrôler.

1

L'Ordre SELECT Elémentaire

Objectifs

A la fin de ce chapitre, vous saurez :

- **Enumérer toutes les possibilités de l'ordre SQL SELECT**
- **Exécuter un ordre SELECT élémentaire**
- **Faire la différence entre les ordres SQL et les commandes SQL*Plus**

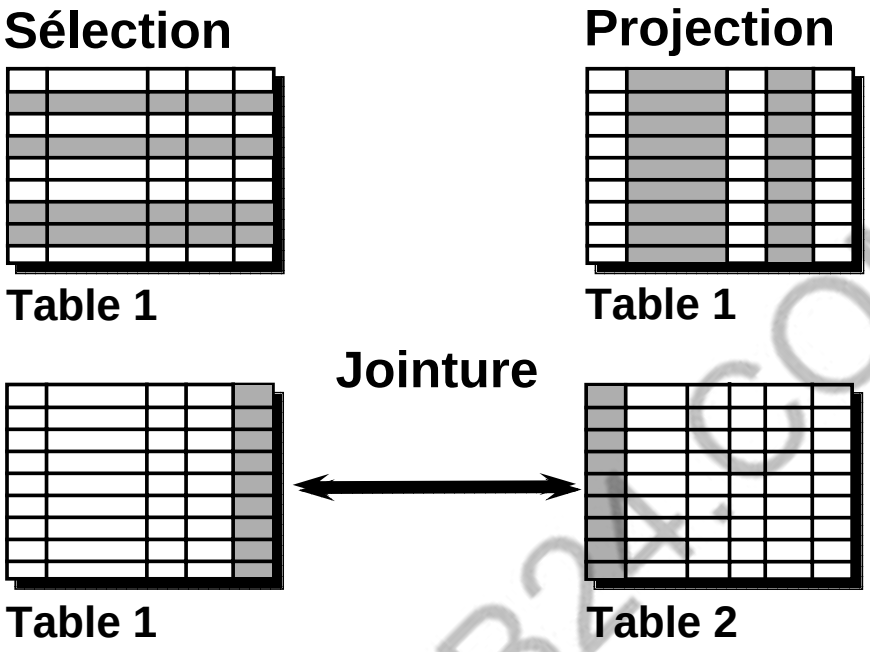
1-2

Objectifs

Pour extraire des données d'une base de données, on utilise l'ordre SQL SELECT. Il est parfois nécessaire de restreindre le nombre de colonnes à l'affichage. Ce chapitre décrit tous les ordres SQL nécessaires à l'exécution de ces actions.

Il peut être utile de créer des ordres SELECT destinés à être réutilisés de nombreuses fois. Ce chapitre explique également comment utiliser les commandes SQL*Plus pour exécuter les ordres SQL.

Les Possibilités de l'Ordre SQL SELECT



1-3

Possibilités de l'ordre SQL SELECT

Un ordre *SELECT* permet d'extraire des informations d'une base de données. L'utilisation d'un ordre SELECT offre les possibilités suivantes :

- *Sélection* : SQL permet de choisir dans une table, les lignes que l'on souhaite ramener au moyen d'une requête. Divers critères de sélection sont disponibles à cet effet.
- *Projection* : SQL permet de choisir dans une table, les colonnes que l'on souhaite ramener au moyen d'une requête. Vous pouvez déterminer autant de colonnes que vous le souhaitez.
- *Jointure* : SQL permet de joindre des données stockées dans différentes tables, en créant un lien par le biais d'une colonne commune à chacune des tables. Les jointures seront décrites en détail dans la suite de ce cours.

Ordre SELECT Elémentaire

```
SELECT   [DISTINCT] {*, column [alias],...}
FROM     table;
```

- **SELECT indique *quelles* colonnes rapporter**
- **FROM indique dans *quelle* table rechercher**

1-4

Ordre SELECT Elémentaire

Dans sa forme la plus simple, l'ordre SELECT comprend :

- Une clause SELECT précisant les colonnes à afficher
- Une clause FROM spécifiant la table qui contient les colonnes indiquées dans la clause SELECT

Syntaxe :

SELECT	liste d'une ou plusieurs colonnes
DISTINCT	suppression des doublons
*	sélection de toutes les colonnes
column	sélection de la colonne désignée
alias	attribue des en-têtes différents aux colonnes sélectionnées
FROM *table*	spécifie la table qui contient les colonnes

Remarque : nous utiliserons les termes mot-clé, clause et ordre tout au long de ce cours.

- Un *mot-clé* est un élément SQL individuel.
  Par exemple, SELECT et FROM sont des mots-clés.
- Une *clause* est une partie d'un ordre SQL.
  Par exemple, SELECT empno, ename, ... est une clause.
- Un *ordre* est la combinaison de deux clauses ou plus.
  Par exemple, SELECT * FROM emp est un ordre SQL.

M.Hanoune

Ecriture des Ordres SQL

- **Les ordres SQL peuvent être écrits indifféremment en majuscules et/ou minuscules.**
- **Les ordres SQL peuvent être écrits sur plusieurs lignes.**
- **Les mots-clés ne doivent pas être abrégés ni scindés sur deux lignes différentes.**
- **Les clauses sont généralement placées sur des lignes distinctes.**
- **Les tabulations et indentations permettent une meilleure lisibilité.**

1-5

Ecriture des Ordres SQL

En suivant les règles et indications simples ci-dessous, vous pourrez créer des ordres corrects, simples à lire et à éditer.

- Sauf indication contraire, les ordres SQL peuvent être écrits indifféremment en majuscules ou en minuscules.
- Les ordres SQL peuvent être saisis sur plusieurs lignes.
- Les mots-clés ne doivent pas être scindés sur deux lignes différentes, ni abrégés.
- Les clauses se placent généralement sur des lignes distinctes pour en faciliter la lecture et l'édition.
- L'utilisation de tabulations et d'indentations permet une meilleure lisibilité.
- Généralement, les mots-clés sont saisis en majuscules, et tous les autres termes, tels que les noms de tables et de colonnes, sont saisis en minuscules.
- Dans le SQL*Plus, un ordre SQL est saisi au prompt SQL, et les lignes qui suivent sont numérotées. C'est ce qu'on appelle le *buffer SQL*. Il ne peut y avoir qu'un seul ordre courant à la fois dans le buffer.

Exécution d'Ordres SQL

- Placer un point-virgule (;) à la fin de la dernière clause.
- Placer un slash (/) seul sur la dernière ligne du buffer.
- Placer un slash (/) au prompt SQL.
- Entrer la commande RUN SQL*Plus au prompt SQL.

Sélection de Toutes les Colonnes

```
SQL> SELECT *
  2  FROM   dept;
```

```
   DEPTNO DNAME          LOC
--------- -------------- -------------
       10 ACCOUNTING     NEW YORK
       20 RESEARCH       DALLAS
       30 SALES          CHICAGO
       40 OPERATIONS     BOSTON
```

1-6

Sélection de Toutes les Colonnes et de Toutes les Lignes

Pour afficher *toutes* les colonnes d'une table, placez un astérisque à la suite du mot-clé SELECT (*). Dans l'exemple de la diapositive, la table des départements (DEPT) comporte trois colonnes : DEPTNO, DNAME et LOC. La table comporte également quatre lignes, une pour chaque département.

Vous pouvez aussi afficher *toutes* les colonnes de la table en les énumérant toutes à la suite du mot-clé SELECT. Par exemple, l'ordre SQL suivant affiche toutes les colonnes et toutes les lignes de la table DEPT :

```
SQL> SELECT deptno, dname, loc
  2  FROM   dept;
```

Sélection d'Une ou Plusieurs Colonnes Spécifiques

```
SQL> SELECT deptno, loc
  2  FROM   dept;
```

```
   DEPTNO LOC
--------- -------------
       10 NEW YORK
       20 DALLAS
       30 CHICAGO
       40 BOSTON
```

1-7

Sélection de Colonnes Spécifiques et de Toutes les Lignes

L'ordre SELECT peut être utilisé pour afficher des colonnes spécifiques de la table. Pour cela, indiquez les noms de colonnes séparés par des virgules. L'exemple ci-dessus affiche tous les numéros de département de la table DEPT, ainsi que leur localisation.

Dans la clause SELECT, indiquez les colonnes dans l'ordre où vous souhaitez qu'elles vous soient rapportées. Par exemple, si vous voulez que la colonne "LOC" soit placée avant la colonne "DEPTNO", utilisez l'ordre suivant :

```
SQL> SELECT loc, deptno
  2  FROM   dept;
```

```
LOC             DEPTNO
------------- ---------
NEW YORK             10
DALLAS               20
CHICAGO              30
BOSTON               40
```

.

Valeurs par Défaut des En-têtes de Colonne

- **Justification par défaut**
  - **A gauche : date et données alphanumériques**
  - **A droite : données numériques**
- **Affichage par défaut : en majuscules**

1-8

Valeurs par Défaut des En-têtes de Colonnes

Les en-têtes de colonnes, comme les données, alphanumériques et dates sont cadrées à gauche. Les en-têtes de colonnes, comme les données, numériques sont cadrées à droite.

```
SQL> SELECT ename, hiredate, sal
  2  FROM   emp;
```

```
ENAME      HIREDATE        SAL
---------- --------- ---------
KING       17-NOV-81      5000
BLAKE      01-MAY-81      2850
CLARK      09-JUN-81      2450
JONES      02-APR-81      2975
MARTIN     28-SEP-81      1250
ALLEN      20-FEB-81      1600
...
14 rows selected.
```

Les en-têtes de colonnes alphanumériques peuvent être tronquées, mais pas les numériques. Par défaut, l'affichage se fait en majuscules. Vous pouvez substituer un alias à un en-tête de colonne à l'affichage. Les alias de colonnes sont traités un peu plus loin dans ce chapitre.

M.Hanoune

Expressions Arithmétiques

Possibilité de créer des expressions avec des données de type NUMBER et DATE au moyen d'opérateurs arithmétiques

Opérateur	Description
+	Addition
-	Soustraction
*	Multiplication
/	Division

1-9

Expressions Arithmétiques

Si nécessaire, vous pouvez modifier l'affichage des données, effectuer des calculs ou étudier différentes hypothèses au moyen d'expressions arithmétiques. Une expression arithmétique peut contenir des noms de colonnes, des valeurs numériques constantes et des opérateurs arithmétiques.

Opérateurs Arithmétiques

Cette diapositive présente les opérateurs arithmétiques disponibles dans SQL. Vous pouvez les utiliser dans n'importe quelle clause d'un ordre SQL, excepté dans la clause FROM.

Utilisation des Opérateurs Arithmétiques

```
SQL> SELECT ename, sal, sal+300
  2  FROM   emp;
```

```
ENAME             SAL   SAL+300
---------- --------- ---------
KING             5000      5300
BLAKE            2850      3150
CLARK            2450      2750
JONES            2975      3275
MARTIN           1250      1550
ALLEN            1600      1900
...
14 rows selected.
```

1-10

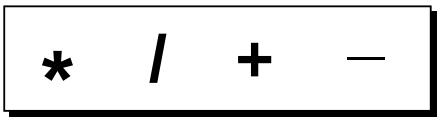
Utilisation des Opérateurs Arithmétiques

L'exemple de la diapositive utilise l'opérateur d'addition pour calculer l'augmentation de salaire de $300 pour tous les employés, puis affiche une nouvelle colonne SAL+300.

A noter que la colonne SAL+300 qui résulte de ce calcul n'est pas une nouvelle colonne de la table EMP ; elle n'est que l'affichage d'un résultat. Par défaut, le nom d'une nouvelle colonne est issu du calcul dont elle provient : dans ce cas précis, sal+300.

Remarque : SQL*Plus ignore les espaces situés avant et après l'opérateur arithmétique.

Priorité des Opérateurs



- **La multiplication et la division ont priorité sur l'addition et la soustraction.**
- **A niveau de priorité identique, les opérateurs sont évalués de gauche à droite.**
- **Les parenthèses forcent la priorité d'évaluation et permettent de clarifier les ordres.**

1-11

Priorité des Opérateurs

Lorsqu'une expression comporte plusieurs opérateurs, les opérateurs de multiplication et de division sont évalués en premier. Lorsque les opérateurs d'une expression sont de priorité identique, ils sont évalués de la gauche vers la droite.

Pour forcer la priorité d'évaluation d'une expression, placez-la entre parenthèses.

Priorité des Opérateurs

```
SQL> SELECT ename, sal, 12*sal+100
  2  FROM   emp;
```

```
ENAME             SAL 12*SAL+100
---------- --------- ----------
KING             5000      60100
BLAKE            2850      34300
CLARK            2450      29500
JONES            2975      35800
MARTIN           1250      15100
ALLEN            1600      19300
...
14 rows selected.
```

1-12

Priorité des Opérateurs (suite)

L'exemple de la diapositive présente le nom, le salaire et le revenu annuel des employés. Le revenu annuel est calculé en multipliant le salaire mensuel par 12, puis en ajoutant un bonus exceptionnel de $100. Remarquez que la multiplication a été effectuée avant l'addition.

Remarque : l'utilisation des parenthèses renforce l'ordre normal de priorité des opérateurs et améliore la clarté. L'expression ci-dessus, par exemple, peut être écrite comme suit : (12*sal)+100, sans que le résultat en soit modifié.

Utilisation des Parenthèses

```
SQL> SELECT ename, sal, 12*(sal+100)
  2  FROM   emp;
```

```
ENAME             SAL 12*(SAL+100)
---------- --------- -----------
KING             5000       61200
BLAKE            2850       35400
CLARK            2450       30600
JONES            2975       36900
MARTIN           1250       16200
...
14 rows selected.
```

1-13

Utilisation des Parenthèses

Vous pouvez modifier les règles de priorité en utilisant des parenthèses pour préciser l'ordre dans lequel les opérateurs doivent être évalués.

L'exemple de la diapositive présente les noms, salaires et revenus annuels des employés. Le calcul du revenu annuel est effectué en multipliant le salaire mensuel ainsi que le bonus de $100 par 12. Les parenthèses rendent l'addition prioritaire sur la multiplication.

La Valeur NULL

- **NULL représente une valeur non disponible, non affectée, inconnue ou inapplicable.**
- **La valeur NULL est différente du zéro ou de l'espace.**

```
SQL> SELECT   ename, job, comm
  2  FROM     emp;
```

```
ENAME      JOB          COMM
---------- --------- ---------
KING       PRESIDENT
BLAKE      MANAGER
...
TURNER     SALESMAN          0
...
14 rows selected.
```

1-14

Les Valeurs NULL

Lorsqu'il manque une valeur dans une colonne sur une ligne, la valeur est dite *NULL*.

Une valeur NULL est une valeur non disponible, non affectée, inconnue ou inapplicable. Une valeur NULL est différente du zéro ou de l'espace. Le zéro est un chiffre et l'espace est un caractère.

Quel que soit le type de données d'une colonne, celle-ci peut contenir des valeurs NULL, excepté lorsque cette colonne a été définie comme NOT NULL ou comme CLE PRIMAIRE lors de sa création.

Dans la colonne COMM de la table EMP, vous pouvez remarquer que seul un employé occupant le poste de vendeur est habilité à toucher une commission. Ainsi, pour les non vendeurs COMM contient NULL. Turner, vendeur, n'a pas touché de commission : la valeur de sa commission est 0 et par conséquent non NULL.

Valeurs NULL dans les Expressions Arithmétiques

Les expressions arithmétiques comportant une valeur NULL sont évaluées à NULL

```
SQL> select ename , 12*sal+comm
  2  from   emp
  3  WHERE  ename='KING';
```

```
ENAME      12*SAL+COMM
---------- -----------
KING
```

1-15

Les Valeurs NULL (suite)

Lorsqu'une valeur NULL est utilisée dans une expression arithmétique, le résultat de cette expression est NULL. Si vous essayez de diviser par zéro, vous obtenez une erreur. En revanche, si vous divisez un nombre par NULL, le résultat sera NULL ou inconnu.

Dans l'exemple ci-dessus, l'employé KING ne fait pas partie de la catégorie SALESMAN et ne touche donc pas de commission. La valeur de la colonne COMM dans l'expression arithmétique étant NULL, le résultat est donc NULL.

Pour plus d'informations, reportez-vous à
Oracle8 Server SQL Reference, Release 8, "Elements of SQL."

L’Alias de Colonne

- **Renomme un en-tête de colonne**
- **Est utile dans les calculs**
- **Suit immédiatement le nom de la colonne ; le mot-clé AS placé entre le nom et l’alias est optionnel**
- **Doit obligatoirement être inclus entre guillemets s’il contient des espaces, des caractères spéciaux ou si les majuscules/minuscules doivent être différenciées**

1-16

Les Alias de Colonnes

Lors de l'affichage des résultats d'une requête, SQL*Plus prend le nom de la colonne sélectionnée comme en-tête de colonne. La plupart du temps, cet en-tête n'est pas explicite et par conséquent est difficile à comprendre. L'alias de colonne permet de modifier l'en-tête d'une colonne.

On spécifie l'alias à la suite de la colonne dans la liste SELECT en utilisant le caractère espace comme séparateur. Par défaut, les en-têtes de colonne sont en majuscules. Si l'alias contient des espaces ou des caractères spéciaux (tels que # ou $), ou si la différence entre majuscules et minuscules est importante, placez l'alias entre guillemets (" ").

M.Hanoune

Utilisation des Alias de Colonnes

```
SQL> SELECT ename AS name, sal salary
  2  FROM   emp;
```

```
NAME           SALARY
-------------- ---------
...
```

```
SQL> SELECT ename "Name",
  2         sal*12 "Annual Salary"
  3  FROM   emp;
```

```
Name           Annual Salary
-------------- -------------
...
```

1-17

Les Alias de Colonnes (suite)

Le premier exemple présente le nom et le salaire mensuel de tous les employés. Remarquez que le mot-clé optionnel AS a été placé avant l'alias de colonne. L'utilisation ou non de ce mot-clé ne modifie pas le résultat de la requête. Notez également que dans l'ordre SQL, les alias de colonne 'name' et 'salary' sont spécifiés en minuscules, alors que les en-têtes de colonne apparaissent en majuscules dans le résultat de la requête. Comme il a été précisé sur la diapositive précédente, c'est le mode d'affichage par défaut.

Le second exemple présente le nom et le salaire annuel de tous les employés. L'alias Annual Salary a été placé entre guillemets car il contient un espace. Remarquez que cette fois l'en-tête de colonne affiché est strictement identique à l'alias de colonne.

L’Opérateur de Concaténation

- **Concatène des colonnes ou chaînes de caractères avec d’autres colonnes**
- **Est représenté par deux barres verticales (||)**
- **La colonne résultante est une expression caractère**

1-18

L'Opérateur de Concaténation

L'opérateur de concaténation (||) permet de concaténer des colonnes à d'autres colonnes, à des expressions arithmétiques ou à des valeurs constantes afin de créer une expression caractère. Les colonnes situées de part et d'autre de l'opérateur se combinent pour donner une colonne unique lors de la restitution des données.

Utilisation de l'Opérateur de Concaténation

```
SQL> SELECT   ename||job AS "Employees"
  2  FROM     emp;
```

```
Employees
------------------
KINGPRESIDENT
BLAKEMANAGER
CLARKMANAGER
JONESMANAGER
MARTINSALESMAN
ALLENSALESMAN
...
14 rows selected.
```

1-19

L'Opérateur de Concaténation (suite)

Dans cet exemple, ENAME et JOB sont concaténés et reçoivent l'alias Employees. On peut remarquer que noms et emplois sont combinés pour ne donner qu'une colonne à l'affichage.

Le mot-clé AS placé devant l'alias simplifie la lecture de la clause SELECT.

Littéral

- **Un littéral est un caractère, une expression, ou un nombre inclus dans la liste SELECT.**
- **Les valeurs littérales de type date et caractère doivent être placées entre simples quotes.**
- **Chaque littéral apparaît sur chaque ligne ramenée.**

1-20

Chaînes de Caractères Littérales

Un littéral est un caractère, une expression, ou un nombre quelconque inclus dans la liste SELECT, et qui n'est ni un nom de colonne ni un alias de colonne. Il apparaît sur chaque ligne ramenée. Des chaînes de texte littérales en format libre peuvent être intégrées au résultat de la requête. Elles sont traitées comme les colonnes dans une liste SELECT.

Les littéraux date et alphanumérique *doivent* être inclus entre simples quotes (' '), mais pas les littéraux numériques.

Utilisation des Chaînes de Caractères Littérales

```
SQL> SELECT ename ||' '||'is a'||' '||job
  2               AS "Employee Details"
  3  FROM   emp;
```

```
Employee Details
-------------------------
KING is a PRESIDENT
BLAKE is a MANAGER
CLARK is a MANAGER
JONES is a MANAGER
MARTIN is a SALESMAN
...
14 rows selected.
```


Chaînes de Caractères Littérales (suite)

L'exemple ci-dessus présente le nom et le poste de tous les employés. L'en-tête de la colonne est Employee Details. Remarquez l'espace situé entre les simples quotes dans l'ordre SELECT. Les espaces améliorent la lisibilité des résultats.

Dans l'exemple suivant, le nom et le salaire de chaque employé est concaténé avec un littéral pour que le résultat soit plus parlant.

```
SQL> SELECT ename ||': 1 Month salary = '||sal Monthly
  2  FROM   emp;
```

```
MONTHLY
----------------------------------------------------------------
KING: 1 Month salary = 5000
BLAKE: 1 Month salary = 2850
CLARK: 1 Month salary = 2450
JONES: 1 Month salary = 2975
MARTIN: 1 Month salary = 1250
ALLEN: 1 Month salary = 1600
TURNER: 1 Month salary = 1500
...
14 rows selected.
```

Doublons

Par défaut, le résultat d'une requête affiche toutes les lignes, y compris les doublons.

```
SQL> SELECT deptno
  2  FROM   emp;
```

```
   DEPTNO
---------
       10
       30
       10
       20
...
14 rows selected.
```

1-22

Doublons

Par défaut, l'instruction SELECT retourne un résultat sans éliminer les doublons. L'exemple ci-dessus affiche la totalité des numéros de département de la table EMP. Vous pouvez remarquer que certains numéros de département apparaissent plusieurs fois.

M.Hanoune

Elimination des Doublons

Pour éliminer les doublons il faut ajouter le mot-clé DISTINCT à la clause SELECT.

```
SQL> SELECT DISTINCT deptno
  2  FROM   emp;
```

```
   DEPTNO
---------
        10
        20
        30
```

1-23

Doublons (suite)

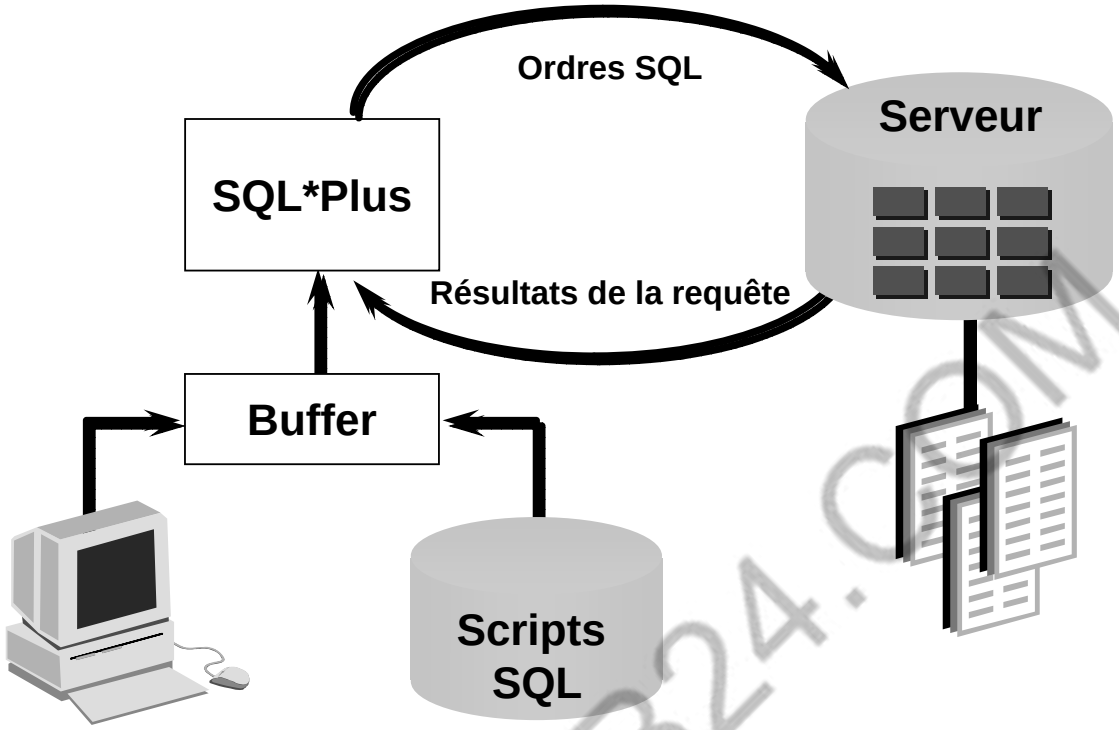
Pour éliminer les doublons du résultat d'une requête, ajoutez le mot-clé DISTINCT à la clause SELECT, directement à la suite du mot-clé SELECT. Dans l'exemple ci-dessus, la table EMP comporte 14 lignes, mais seulement trois numéros de département distincts.

Il est possible de spécifier plusieurs colonnes à la suite du paramètre DISTINCT. Ce paramètre agit sur toutes les colonnes sélectionnées, on obtient en résultat les combinaisons distinctes de ces colonnes.

```
SQL> SELECT DISTINCT deptno, job
  2  FROM   emp;
```

```
 DEPTNO JOB
------ ---------
    10 CLERK
    10 MANAGER
    10 PRESIDENT
    20 ANALYST
...
9 rows selected.
```

Interaction entre SQL et SQL*Plus



1-24

SQL et SQL*Plus

SQL est un langage de commande qui permet de communiquer avec Oracle Server quels que soient l'outil ou l'application utilisés. Le SQL d'Oracle comprend de nombreuses extensions.

*SQL*Plus* est un outil Oracle possédant son propre langage de commande qui reconnaît les ordres SQL et les envoie à Oracle Server pour qu'ils soient exécutés. Lorsque vous saisissez un ordre SQL, celui-ci est stocké dans un espace mémoire appelé *buffer SQL* où il reste jusqu'à la saisie d'un nouvel ordre.

Caractéristiques de SQL

- Peut être employé par de nombreux utilisateurs, y compris ceux ayant peu ou pas d'expérience de programmation
- C'est un langage non procédural
- Il réduit le temps de création et de mise à jour des systèmes
- C'est un langage proche de la langue anglaise

Caractéristiques de SQL*Plus

- Accepte la saisie d'ordres ad hoc
- Accepte l'exécution d'ordres SQL à partir de fichiers
- Contient un éditeur de lignes pour la modification des ordres SQL
- Contrôle les paramètres d'environnement
- Formate les résultats des requêtes dans des états élémentaires
- Accède aux bases de données locales ou distantes

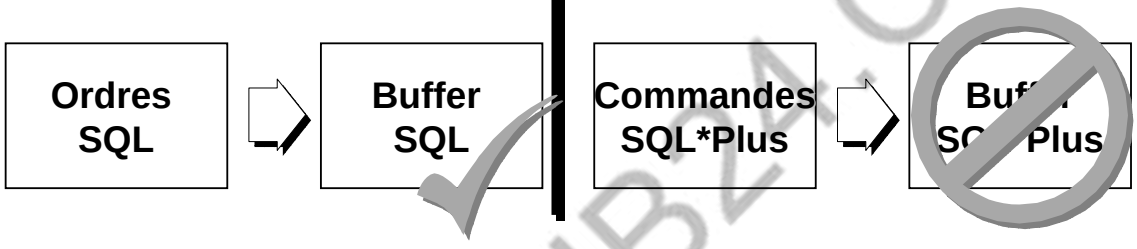
Comparatif entre Ordres SQL et Commandes SQL*Plus

SQL

- **Un langage**
- **Standard ANSI**
- **Abréviation des mots-clés impossible**
- **Les ordres agissent sur le contenu et la définition des tables de la base de données**

SQL*Plus

- **Un environnement**
- **Produit propriétaire Oracle**
- **Abréviation des mots-clés possible**
- **Les commandes ne permettent d'agir ni sur le contenu, ni sur la définition des tables.**



1-25

SQL et SQL*Plus (suite)

La table ci-dessous présente un comparatif de SQL et SQL*Plus.

SQL	SQL*Plus
Langage de communication avec Oracle Server pour accéder aux données	Reconnaît les ordres SQL et les envoie au serveur
Basé sur le SQL standard ANSI (American National Standards Institute)	Interface propriétaire Oracle pour l'exécution des ordres SQL
Manipule les données et les définitions de tables dans la base de données	Les commandes SQL*Plus ne permettent pas la manipulation de valeurs dans la base de données
Est entré dans le buffer SQL sur une ou plusieurs lignes.	Entrée des lignes une par une, pas de stockage dans le buffer SQL
Ne comporte pas de caractère de continuation	Le tiret (-) est utilisé comme caractère de continuation lorsque la commande fait plus d'une ligne
Ne peut être abrégé	Peut être abrégé
Comporte un caractère de fin pour l'exécution immédiate des commandes	Ne nécessite pas de caractère de fin ; les commandes sont immédiatement exécutées
Utilise des fonctions pour effectuer les formatages	Utilise des commandes pour formater les données

Présentation de SQL*Plus

- **Connexion à SQL*Plus.**
- **Affichage de la structure d'une table.**
- **Edition d'un ordre SQL.**
- **Exécution de SQL à partir de SQL*Plus.**
- **Enregistrement ou ajout d'ordres SQL dans un fichier.**
- **Exécution d'ordres contenus dans un fichier.**
- **Chargement dans le buffer de l'ordre contenu dans un fichier.**

1-26

SQL*Plus

L'environnement SQL*Plus permet :

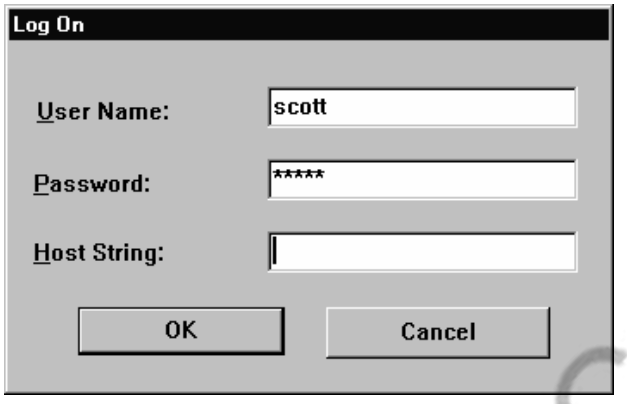
- L'extraction, la modification, l'ajout et la suppression de données dans la base de données au moyen d'ordres SQL
- Le formatage, le calcul, le stockage et l'impression des résultats de requête sous la forme d'états
- La création de fichiers script pour le stockage d'ordres SQL dans le but d'une utilisation répétée

Les commandes SQL*Plus se divisent selon les catégories suivantes :

Catégorie	Objet
Environnement	Influencer le comportement général des ordres SQL pour la session en cours
Formatage	Formater les résultats de requêtes
Manipulation de fichiers	Sauvegarder, charger et exécuter des fichiers scripts
Exécution	Envoyer l'ordre SQL du buffer SQL vers Oracle Server
Edition	Modifier l' ordre SQL contenu dans le buffer
Interaction	Créer et transmettre des variables aux ordres SQL, imprimer les valeurs des variables et afficher des messages.
Divers	Nombreuses commandes permettant de se connecter à la base de données, de manipuler l'environnement SQL*Plus et d'afficher les définitions de colonnes

Connexion à SQL*Plus

- **Depuis l'environnement Windows :**



- **Depuis une ligne de commande :**

sqlplus [*username*[/*password* [@*database*]]]

1-27

Connexion à SQL*Plus

La procédure d'appel de SQL*Plus varie en fonction de votre système d'exploitation ou de votre environnement Windows.

Pour se connecter depuis l'environnement Windows NT :

1. Cliquez sur Démarrer—>Programmes—>Oracle for Windows NT—>SQL*Plus
2. Entrez votre nom d'utilisateur, votre mot de passe et le nom de la chaîne de connexion à la base de données.

Pour se connecter depuis une ligne de commande :

1. Ouvrez une session sur votre machine
2. Saisissez la commande SQL*Plus comme indiqué sur la diapositive ci-dessus.

Syntaxe de la commande :

username	votre nom d'utilisateur de base de données
password	votre mot de passe ; à ce niveau, le mot de passe est visible
@database	la chaîne de connexion à la base de données

Remarque : pour garantir l'intégrité de votre mot de passe, ne le saisissez pas au prompt du système d'exploitation. Saisissez uniquement votre nom d'utilisateur, puis saisissez votre mot de passe au prompt Mot de passe.

Lorsque vous êtes bien connecté à SQL*Plus, un message équivalent au message suivant apparaît :

```
SQL*Plus : Release 8.0.3.0.0 - Production on Mon Oct 06 16:03:43
1997
(c) Copyright 1997 Oracle Corporation. All rights reserved.
```

Affichage de la Structure d'une Table

Utilisez la commande SQL*Plus DESCRIBE pour afficher la structure d'une table.

```
DESC[RIBE] tablename
```

1-28

Affichage de la Structure d'une Table

Dans SQL*Plus, vous pouvez afficher la structure d'une table à l'aide de la commande DESCRIBE. Cette commande affiche les noms de colonnes et les types de données, ainsi qu'une information indiquant si les colonnes *doivent* obligatoirement contenir des données ou non.

Syntaxe :

tablename	nom d'une table, d'une vue ou d'un synonyme existants, accessibles à l'utilisateur

Affichage de la Structure d'une Table

```
SQL> DESCRIBE dept
```

```
Name                 Null?    Type
-------------------- -------- ----
DEPTNO               NOT NULL NUMBER(2)
DNAME                         VARCHAR2(14)
LOC                           VARCHAR2(13)
```


Affichage de la Structure d'une Table (suite)

L'exemple ci-dessus affiche des informations sur la structure de la table DEPT.

Dans le résultat :

Null? indique si une colonne *doit ou non* contenir des données; NOT NULL signifie que la colonne doit contenir des données

Type affiche le type de données d'une colonne

Les types de données sont décrits dans le tableau suivant :

Type de données	**Description**
NUMBER(*p*,*s*)	Valeur numérique contenant au maximum *p* chiffres, dont *s* chiffres après la virgule
VARCHAR2(*s*)	Valeur caractère de longueur variable, d'une taille maximale *s*
DATE	Valeurs de date et heure situées entre le 1er janvier 4712 av.J.-C. et le 31 décembre 9999 apr.J.-C.
CHAR(*s*)	Valeur caractère de longueur fixe *s*

Commandes d'Edition SQL*Plus

- **A[PPEND] *text***
- **C[HANGE] / *old* / *new***
- **C[HANGE] / *text* /**
- **CL[EAR] BUFF[ER]**
- **DEL**
- **DEL *n***
- **DEL *m n***

1-30

Commandes d'Editions SQL*Plus

Les commandes SQL*Plus sont saisies une à une et ne sont pas stockées dans le buffer SQL.

Commande	Description
A[PPEND] *text*	Ajoute du texte à la fin de la ligne courante
C[HANGE] / *old* /*new*	Remplace le texte *old* par le *new* dans la ligne courante
C[HANGE] / *text* /	Supprime le *text* dans la ligne courante
CL[EAR] BUFF[ER]	Supprime toutes les lignes du buffer SQL
DEL	Supprime la ligne courante

Conseils

- Si vous appuyez sur la touche [Entrée] avant d'avoir terminé une commande SQL, SQL*Plus vous demande un numéro de ligne.
- Pour terminer une saisie dans le buffer SQL, saisissez un des caractères de terminaison (point-virgule ou slash) ou appuyez deux fois sur la touche [Entrée]. Le prompt SQL apparaît alors.

Commandes d'Edition SQL*Plus

- **I[NPUT]**
- **I[NPUT]** ***text***
- **L[IST]**
- **L[IST]** ***n***
- **L[IST]** ***m n***
- **R[UN]**
- ***n***
- ***n text***
- **0** ***text***

1-31

Commandes d'Edition SQL*Plus (suite)

Commande	**Description**
I[NPU T]	Insère un nombre indéfini de lignes
I[NPUT] *text*	Insère la ligne *text*
L[IST]	Affiche toutes les lignes du buffer SQL
L[IST] *n*	Affiche une ligne (indiquée par *n*)
L[IST] *m n*	Affiche une série de lignes (*m à n)*
R[UN]	Affiche et exécute l'ordre SQL actuellement dans le buffer
n	Indique quelle doit être la ligne courante
n text	Remplace la ligne *n* par *text*
0 *text*	Insère une ligne avant la ligne 1

On ne peut saisir qu'une seule commande SQL*Plus par prompt SQL. Les commandes SQL*Plus ne sont pas stockées dans le buffer. Pour poursuivre la saisie d'une commande SQL*Plus sur la ligne suivante, entrez un tiret (-) à la fin de la ligne courante.

Commandes de Fichiers SQL*Plus

- **SAVE *filename***
- **GET *filename***
- **START *filename***
- **@ *filename***
- **EDIT *filename***
- **SPOOL *filename***
- **EXIT**

1-32

Commandes de Fichiers SQL*Plus

Les ordres SQL communiquent avec Oracle. Les commandes SQL*Plus contrôlent l'environnement, formatent les résultats des requêtes et gèrent les fichiers. Pour gérer les fichiers,vous disposez des commandes décrites dans le tableau suivant :

Commande	**Description**
SAV[E] *filename* [.ext] [REP[LACE]APP[END]]	Sauvegarde dans un fichier le contenu actuel du buffer SQL. APPEND ajoute le contenu à un fichier existant ; REPLACE écrase un fichier existant. L'extension par défaut est .sql.
GET *filename* [.ext]	Charge le contenu d'un fichier déjà sauvegardé dans le buffer SQL. Par défaut, l'extension du fichier est .sql.
STA[RT] *filename* [.ext]	Exécute un fichier de commandes déjà sauvegardé.
@ *filename*	Exécute un fichier de commandes déjà sauvegardé (identique à START).
ED[IT]	Appelle l'éditeur et sauvegarde le contenu du buffer dans un fichier nommé *afiedt.buf*.
ED[IT] [*filename*[.ext]]	Appelle l'éditeur et édite le contenu d'un fichier déjà sauvegardé.
SPO[OL] [*filename*[.ext]| OFF|OUT]	Stocke les résultats de requêtes dans un fichier. OFF ferme le fichier spool. OUT ferme le fichier spool et envoie les résultats du fichier à l'imprimante du système.
EXIT	Quitte SQL*Plus.

Résumé

```
SELECT   [DISTINCT] {*,column[alias],...}
FROM     table;
```

L'environnement SQL*Plus permet :

- **D'exécuter des ordres SQL**
- **D'éditer des ordres SQL**

Ordre SELECT

Dans ce chapitre, vous avez appris à extraire des données d'une table dans une base de données à l'aide de l'ordre SELECT.

```
SELECT  [DISTINCT] {*,column[alias],...}
FROM    table;
```

Où :	SELECT	Est une liste contenant au moins une colonne
	DISTINCT	Supprime les doublons
	*	Sélectionne toutes les colonnes
	column	Sélectionne la colonne nommée
	alias	Remplace l'en-tête de la colonne sélectionnée
	FROM *table*	Spécifie la table qui contient les colonnes

SQL*Plus

SQL*Plus est un environnement d'exécution qui vous permet d'envoyer des commandes SQL au serveur de la base de données, ainsi que d'éditer et de sauvegarder des commandes SQL. Les commandes peuvent être exécutées à partir du prompt SQL ou depuis un fichier script.

Présentation des Exercices

- **Afficher les données de différentes tables.**
- **Afficher la structure des tables.**
- **Calculs arithmétiques et spécifications des noms de colonnes.**
- **Utilisation de l'éditeur SQL*Plus.**

1-34

Présentation des Exercices

Voici le premier d'une longue série d'exercices. Les solutions (si vous en avez besoin) sont dans l'annexe A. Les exercices ont pour objet de présenter tous les sujets abordés dans ce chapitre. Répondez aux questions 2 à 4 sur papier.

Certains exercices contiennent des questions "si vous avez du temps" ou "pour aller plus loin". Ne les faites que si vous avez répondu à toutes les autres questions dans le temps imparti et si vous souhaitez tester vos connaissances plus avant

Commencez les exercices sans vous presser et consciencieusement. Vous pouvez vous entraîner à sauvegarder et à exécuter des fichiers de commande. Si nécessaire, n'hésitez pas à poser des questions à votre instructeur.

Questions sur Papier

Pour les questions 2 à 4, entourez Vrai ou Faux.

Exercices 1

1. Initialisez une session SQL*Plus avec votre ID et le mot de passe que votre instructeur vous a remis.
2. Les commandes SQL*Plus accèdent aux bases de données.
   Vrai/Faux
3. L'ordre SELECT suivant sera convenablement exécuté.
   Vrai/Faux

```
SQL> SELECT       ename, job, sal Salary
  2  FROM         emp;
```

4. L'ordre SELECT suivant sera convenablement exécuté.
   Vrai/Faux

```
SQL> SELECT       *
  2  FROM         salgrade;
```

5. Cet ordre comporte trois erreurs de code ; pouvez-vous les trouver ?

```
SQL> SELECT       empno, ename
  2               sal x 12 ANNUAL SALARY
  3  FROM         emp;
```

6. Affichez la structure de la table DEPT. Sélectionnez toutes les données de la table DEPT.

```
Name          Null?    Type
------------ -------- ----
DEPTNO        NOT NULL NUMBER(2)
DNAME                  VARCHAR2(14)
LOC                    VARCHAR2(13)
```

```
DEPTNO DNAME      LOC
------ ---------- -------------
    10 ACCOUNTING NEW YORK
    20 RESEARCH   DALLAS
    30 SALES      CHICAGO
    40 OPERATIONS BOSTON
```

Exercices 1

7. Affichez la structure de la table EMP . Créez une requête pour afficher le nom (ename), le poste (job), la date d'embauche (hiredate) et le matricule (empno) de chaque employé, en plaçant le matricule en premier. Enregistrez votre ordre SQL dans un fichier nommé *p1q7.sql*.

```
Name            Null?    Type
--------------- -------- ----
EMPNO           NOT NULL NUMBER(4)
ENAME                    VARCHAR2(10)
JOB                      VARCHAR2(9)
MGR                      NUMBER(4)
HIREDATE                 DATE
SAL                      NUMBER(7,2)
COMM                     NUMBER(7,2)
DEPTNO                   NOT NULL NUMBER(2)
```

8. Exécutez la requête que vous avez placée dans le fichier *p1q7.sql*.

```
EMPNO ENAME   JOB          HIREDATE
----- ------- ----------- ---------
 7839 KING    PRESIDENT   17-NOV-81
 7698 BLAKE   MANAGER     01-MAY-81
 7782 CLARK   MANAGER     09-JUN-81
 7566 JONES   MANAGER     02-APR-81
 7654 MARTIN  SALESMAN    28-SEP-81
 7499 ALLEN   SALESMAN    20-FEB-81
 7844 TURNER  SALESMAN    08-SEP-81
 7900 JAMES   CLERK       03-DEC-81
 7521 WARD    SALESMAN    22-FEB-81
 7902 FORD    ANALYST     03-DEC-81
 7369 SMITH   CLERK       17-DEC-80
 7788 SCOTT   ANALYST     09-DEC-82
 7876 ADAMS   CLERK       12-JAN-83
 7934 MILLER  CLERK       23-JAN-82
14 rows selected.
```

Exercices 1

9. Créez une requête pour afficher les différents types de poste existant dans la table EMP.

```
JOB
-----------
ANALYST
CLERK
MANAGER
PRESIDENT
SALESMAN
```

Si vous avez du temps, faites les exercices suivants :

10. Editez *p1q7.sql*. Donnez respectivement les noms suivants aux en-têtes de colonne : Emp #, Employee, Job, et Hire Date. Exécutez à nouveau votre requête.

```
Emp # Employee  Job        Hire Date
----- --------- ---------- --------------
 7839 KING      PRESIDENT  17-NOV-81
 7698 BLAKE     MANAGER    01-MAY-81
 7782 CLARK     MANAGER    09-JUN-81
 7566 JONES     MANAGER    02-APR-81
 7654 MARTIN    SALESMAN   28-SEP-81
 7499 ALLEN     SALESMAN   20-FEB-81
 7844 TURNER    SALESMAN   08-SEP-81
 7900 JAMES     CLERK      03-DEC-81
 7521 WARD      SALESMAN   22-FEB-81
 7902 FORD      ANALYST    03-DEC-81
 7369 SMITH     CLERK      17-DEC-80
 7788 SCOTT     ANALYST    09-DEC-82
 7876 ADAMS     CLERK      12-JAN-83
 7934 MILLER    CLERK      23-JAN-82
14 rows selected.
```

Exercices 1

11. Affichez le nom concaténé avec le poste en les séparant par une virgule suivie d'un espace, puis donnez comme titre à la colonne Employee and Title.

```
Employee and Title
------------------
KING, PRESIDENT
BLAKE, MANAGER
CLARK, MANAGER
JONES, MANAGER
MARTIN, SALESMAN
ALLEN, SALESMAN
TURNER, SALESMAN
JAMES, CLERK
WARD, SALESMAN
FORD, ANALYST
SMITH, CLERK
SCOTT, ANALYST
ADAMS, CLERK
MILLER, CLERK
14 rows selected.
```

Si vous souhaitez aller plus loin dans la difficulté, faites l' exercice suivant :

12. Créez une requête pour afficher toutes les données de la table EMP dans une seule colonne d'affichage. Séparez chaque colonne par une virgule. Nommez la colonne d'affichage THE_OUTPUT.

```
THE_OUTPUT
--------------------------------------------
7839,KING,PRESIDENT,,17-NOV-81,5000,,10
7698,BLAKE,MANAGER,7839,01-MAY-81,2850,,30
7782,CLARK,MANAGER,7839,09-JUN-81,2450,,10
7566,JONES,MANAGER,7839,02-APR-81,2975,,20
7654,MARTIN,SALESMAN,7698,28-SEP-81,1250,1400,30
7499,ALLEN,SALESMAN,7698,20-FEB-81,1600,300,30
7844,TURNER,SALESMAN,7698,08-SEP-81,1500,0,30
7900,JAMES,CLERK,7698,03-DEC-81,950,,30
7521,WARD,SALESMAN,7698,22-FEB-81,1250,500,30
7902,FORD,ANALYST,7566,03-DEC-81,3000,,20
7369,SMITH,CLERK,7902,17-DEC-80,800,,20
7788,SCOTT,ANALYST,7566,09-DEC-82,3000,,20
7876,ADAMS,CLERK,7788,12-JAN-83,1100,,20
7934,MILLER,CLERK,7782,23-JAN-82,1300,,10
14 rows selected.
```

2

Sélection et Tri des Lignes Retournées par un SELECT

Objectifs

A la fin de ce chapitre, vous saurez :

- **Limiter le nombre de lignes retournées par une requête**
- **Trier les lignes retournées par une requête**

2-2

Objectifs

Lors d'une recherche de données dans une base de données, il est parfois nécessaire de restreindre le nombre de lignes retournées ou de préciser l'ordre d'affichage de ces lignes. Ce chapitre explique quelles clauses SQL utiliser à cet effet.

Hanoune

Sélectionner les Lignes

EMP

EMPNO	ENAME	JOB	...	DEPTNO
7839	KING	PRESIDENT		10
7698	BLAKE	MANAGER		30
7782	CLARK	MANAGER		10
7566	JONES	MANAGER		20
...				

"...rechercher tous les employés du département 10"

EMP

EMPNO	ENAME	JOB	...	DEPTNO
7839	KING	PRESIDENT		10
7782	CLARK	MANAGER		10
7934	MILLER	CLERK		10

2-3

Restriction du Nombre de Lignes d'une Sélection

Dans cet exemple, nous souhaitons afficher tous les employés du département 10. Les seules lignes ramenées, mises en évidence dans le cadre du bas, sont celles dont la colonne DEPTNO contient la valeur 10. La séléction des lignes est effectuée par la clause SQL WHERE.

Sélectionner les Lignes

- **Restreindre la sélection au moyen de la clause WHERE.**

```
SELECT      [DISTINCT] {*, column [alias], ...}
FROM        table
[WHERE      condition(s)];
```

- **La clause WHERE se place après la clause FROM.**

2-4

Sélectionner les Lignes

Vous pouvez limiter le nombre de lignes ramenées par la requête au moyen de la clause WHERE. La clause WHERE permet de spécifier une condition à satisfaire. Elle se place immédiatement après la clause FROM.

Syntaxe :

WHERE	limite la requête aux lignes remplissant la condition spécifiée
condition	se compose de noms de colonnes, d'expressions, de constantes et d'un opérateur de comparaison

La clause WHERE peut comparer des valeurs dans des colonnes, des littéraux, des expressions arithmétiques ou des fonctions. Elle se compose de trois éléments :

- Nom de colonne
- Opérateur de comparaison
- Nom de colonne, constante ou liste de valeurs

Utilisation de la Clause WHERE

```
SQL> SELECT ename, job, deptno
  2  FROM   emp
  3  WHERE  job='CLERK';
```

ENAME	JOB	DEPTNO
JAMES	CLERK	30
SMITH	CLERK	20
ADAMS	CLERK	20
MILLER	CLERK	10

2-5

Utilisation de la Clause WHERE

Dans l'exemple, l'ordre SELECT recherche le nom, le poste et le numéro de département de tous les employés dont le poste (job) est CLERK.

A noter que le poste CLERK a été indiqué en majuscules pour garantir la correspondance avec la colonne "job" de la table EMP. En effet, la recherche tient compte des majuscules et minuscules.

Chaînes de Caractères et Dates

- **Les constantes chaînes de caractères et dates doivent être placées entre simples quotes.**
- **La recherche tient compte des majuscules et minuscules (pour les chaînes de caractère) et du format (pour les dates.)**
- **Le format de date par défaut est 'DD-MON-YY'.**

```
SQL> SELECT   ename, job, deptno
  2  FROM     emp
  3  WHERE    ename = 'JAMES';
```

2-6

Chaînes de Caractères et Dates

Dans la clause WHERE, les chaînes alphanumériques et les dates doivent être incluses entre simples quotes (' '), mais pas les constantes numériques.

Toutes les recherches de caractères tiennent compte des majuscules et des minuscules. Dans l'exemple ci-dessous, aucune ligne n'est ramenée car, dans la table EMP, toutes les données sont stockées en lettres majuscules.

```
SQL> SELECT ename, empno, job, deptno
   2  FROM   emp
   3  WHERE  job='clerk';
```

Oracle stocke les dates dans un certain format numérique interne, représentant le siècle, l'année, le mois, le jour, les heures, les minutes et les secondes. Le format de date par défaut est :
DD-MON-YY.

Remarque : la modification du format de date par défaut est expliquée dans le chapitre 3.
Les valeurs numériques ne sont pas incluses entre simples quotes.

Opérateurs de Comparaison

Opérateur	Signification
=	Egal à
>	Supérieur à
>=	Supérieur ou égal à
<	Inférieur à
<=	Inférieur ou égal à
<>	Différent de

2-7

Opérateurs de Comparaison

Ces opérateurs de comparaison s'utilisent dans les conditions qui comparent deux expressions. Dans la clause WHERE, ils s'utilisent de la façon suivante:

Syntaxe

```
… WHERE expr operator value
```

Exemples

```
… WHERE hiredate='01-JAN-95'
… WHERE sal>=1500
… WHERE ename='SMITH'
```

Utilisation des Opérateurs de Comparaison

```
SQL> SELECT ename, sal, comm
  2  FROM   emp
  3  WHERE  sal<=comm;
```

```
ENAME          SAL       COMM
---------- --------- ---------
MARTIN        1250 <--> 1400
```

2-8

Utilisation des Opérateurs de Comparaison

Dans l'exemple, l'ordre SELECT recherche le nom, le salaire et la commission dans la table EMP, la condition (where) étant que le salaire de l'employé soit inférieur ou égal à la commission. A noter qu'ici, aucune valeur explicite n'est fournie dans la clause WHERE. Les deux valeurs à comparer sont prises dans les colonnes SAL et COMM de la table EMP.

Autres Opérateurs de Comparaison

Opérateur	Signification
BETWEEN ...AND...	Compris entre ... et ... (bornes comprises)
IN (liste)	Correspond à une valeur de la liste
LIKE	Ressemblance partielle de chaînes de caractères
IS NULL	Correspond à une valeur NULL

2-9

Utilisation de l'Opérateur BETWEEN

BETWEEN permet de tester l'appartenance à une fourchette de valeurs.

```
SQL> SELECT   ename, sal
  2  FROM     emp
  3  WHERE    sal BETWEEN 1000 AND 1500;
```

ENAME	SAL
MARTIN	1250
TURNER	1500
WARD	1250
ADAMS	1100
MILLER	1300

Limite inférieure

Limite supérieure

2-10

L'Opérateur BETWEEN

L'opérateur BETWEEN permet d'afficher des lignes en fonction d'un intervalle de valeurs. Vous spécifiez un intervalle comprenant une limite inférieure et une limite supérieure.

Dans l'exemple ci-dessus, l'ordre SELECT ramène les lignes correspondant aux employés ayant un salaire compris entre $1000 et $1500.

Les valeurs spécifiées avec l'opérateur BETWEEN sont inclusives. Vous devez spécifier la limite inférieure en premier.

Utilisation de l'Opérateur IN

IN permet de comparer une expression avec une liste de valeurs.

```
SQL> SELECT   empno, ename, sal, mgr
  2  FROM     emp
  3  WHERE    mgr IN (7902, 7566, 7788);
```

```
     EMPNO ENAME              SAL        MGR
---------- ---------- ---------- ----------
      7902 FORD             3000       7566
      7369 SMITH             800       7902
      7788 SCOTT            3000       7566
      7876 ADAMS            1100       7788
```

2-11

L'Opérateur IN

Pour comparer une expression avec une liste de valeurs, utilisez l'opérateur IN.

Dans cet exemple, on affiche le matricule, le nom, le salaire et le numéro de manager de tous les employés ayant un manager dont le matricule est 7902, 7566 ou 7788.

L'opérateur IN peut s'utiliser avec n'importe quel type de données. Le SELECT suivant ramène une ligne par employé dont le nom figure dans la liste.

```
SQL> SELECT  empno,  ename,  mgr, deptno
  2  FROM    emp
  3  WHERE   ename IN ('FORD' , 'ALLEN');
```

Si vous spécifiez des caractères ou des dates dans la liste, il faut les inclure entre simples quotes (' ').

Hanoune

Utilisation de l'Opérateur LIKE

- **LIKE permet de rechercher des chaînes de caractères à l'aide de caractères génériques**
- **Les conditions de recherche peuvent contenir des caractères ou des nombres littéraux.**
  - **(%) représente zéro ou plusieurs caractères**
  - **(_) représente un caractère**

```
SQL> SELECT   ename
  2  FROM     emp
  3  WHERE    ename LIKE 'S%';
```

2-12

L'Opérateur LIKE

Vous ne connaissez pas toujours les valeurs exactes à rechercher. Vous pouvez sélectionner des lignes correspondant à une suite de caractères au moyen de l'opérateur LIKE. L'opération ainsi exécutée est appelée recherche ***générique***. Deux symboles sont utilisables pour construire la chaîne de recherche :

Symbole	Description
%	Représente zéro caractère ou une séquence quelconque de caractères
_	Représente un caractère quelconque

L'ordre SELECT ci-dessus ramène tous les employés dont le nom commence par "S" (il s'agit bien du S majuscule). Les noms commençant par "s" ne seront pas sélectionnés. L'opérateur LIKE peut s'utiliser comme raccourci dans certaines comparaisons BETWEEN. L'exemple suivant affiche le nom et la date d'embauche de tous les employés ayant rejoint la société entre Janvier 1981 et Décembre 1981.

```
SQL>  SELECT ename, hiredate
2     FROM     emp
3     WHERE    hiredate LIKE '%81';
```

Utilisation de l'Opérateur LIKE

- **Vous pouvez combiner plusieurs caractères génériques de recherche.**

```
SQL> SELECT   ename
  2  FROM     emp
  3  WHERE    ename LIKE '_A%';
```

```
ENAME
----------
JAMES
WARD
```

- **Vous pouvez utiliser l'identifiant ESCAPE pour rechercher "%" ou "_".**

2-13

Combinaison de Caractères Génériques

Les symboles % et _ peuvent être combinés de manière quelconque avec des caractères littéraux. L'exemple de la diapositive affiche le nom des employés dont la deuxième lettre est un "A".

L'Option ESCAPE

Lorsque vous voulez rechercher les caractères "%" et "_" proprement dits, utilisez l'option ESCAPE, qui permet de définir un caractère ESCAPE. Pour afficher les noms des employés contenant les caractères "A_B", spécifiez l'ordre SQL suivant :

```
SQL> SELECT   ename
  2  FROM     emp
  3  WHERE    ename  LIKE  '%A\_B%'  ESCAPE  '\';
```

L'option ESCAPE définit la barre oblique inverse (\ ) comme caractère ESCAPE. Ainsi, le caractère suivant ce caractère ESCAPE ne sera pas interprété comme le caractère générique (_), mais comme le simple caractère (_).

Utilisation de l'Opérateur IS NULL

Recherche de valeurs NULL avec l'opérateur IS NULL

```
SQL> SELECT  ename, mgr
  2  FROM    emp
  3  WHERE   mgr IS NULL;
```

```
ENAME          MGR
---------- ---------
KING
```

2-14

L'Opérateur IS NULL

L'opérateur IS NULL teste les valeurs NULL. Une valeur NULL est indisponible, non attribuée, inconnue ou inapplicable. Il est donc impossible de tester une valeur NULL à l'aide de l'opérateur (=) puisqu'elle ne peut répondre à aucune condition d'égalité ou d'inégalité. L'exemple ci-dessus recherche le nom et le manager de tous les employés n'ayant pas de manager.

Par exemple, pour afficher le nom, le poste et la commission de tous les employés non habilités à toucher une commission, utilisez l'instruction SQL suivante :

```
SQL> SELECT    ename,  job, comm
  2  FROM      emp
  3  WHERE     comm  IS  NULL;
```

```
ENAME    JOB           COMM
-------- ----------- ------
KING     PRESIDENT
BLAKE    MANAGER
CLARK    MANAGER
...
```

Opérateurs Logiques

Opérateur	Signification
AND	Retourne TRUE si *les deux* conditions sont VRAIES
OR	Retourne TRUE si *l'une au moins* des conditions est VRAIE
NOT	Ramène la valeur TRUE si la condition qui suit l'opérateur est FAUSSE

2-15

Opérateurs Logiques

Un opérateur logique combine le résultat de deux conditions pour produire un résultat unique ou inverse le résultat d'une condition unique. SQL inclut trois opérateurs logiques :

- AND
- OR
- NOT

Tous les exemples que vous avez vus jusqu'ici ne spécifiaient qu'une seule condition dans la clause WHERE. En utilisant les opérateurs AND et OR, vous pouvez inclure plusieurs conditions dans une même clause WHERE.

Utilisation de l'Opérateur AND

Avec AND, les deux conditions doivent être VRAIES.

```
SQL> SELECT empno, ename, job, sal
  2  FROM   emp
  3  WHERE  sal>=1100
  4  AND    job='CLERK';
```

```
     EMPNO ENAME      JOB              SAL
---------- ---------- --------- ---------
      7876 ADAMS      CLERK            1100
      7934 MILLER     CLERK            1300
```

2-16

L'Opérateur AND

Dans l'exemple, les deux conditions doivent être vraies pour qu'un enregistrement soit sélectionné. Ainsi, tout employé dont le poste est CLERK *et* qui gagne plus de $1100 sera sélectionné.

Toutes les recherches de caractères font la distinction entre les majuscules et les minuscules. Aucune ligne n'est ramenée si le mot CLERK n'est pas entièrement en majuscules. Les chaînes de caractères doivent obligatoirement être incluses entre simples quotes.

Table de Vérité de l'Opérateur AND

Le tableau suivant montre les résultats obtenus en combinant deux expressions avec AND :

AND	TRUE	FALSE	UNKNOWN
TRUE	TRUE	FALSE	UNKNOWN
FALSE	FALSE	FALSE	FALSE
UNKNOWN	UNKNOWN	FALSE	UNKNOWN

Utilisation de l'Opérateur OR

Avec OR, l'une ou l'autre des deux conditions doit être VRAIE.

```
SQL> SELECT empno, ename, job, sal
  2  FROM   emp
  3  WHERE  sal>=1100
  4  OR     job='CLERK';
```

```
     EMPNO ENAME      JOB              SAL
---------- ---------- --------- ---------

      7839 KING       PRESIDENT       5000
      7698 BLAKE      MANAGER         2850
      7782 CLARK      MANAGER         2450
      7566 JONES      MANAGER         2975
      7654 MARTIN     SALESMAN        1250
...
14 rows selected.
```

2-17

L'Opérateur OR

Dans l'exemple, l'une ou l'autre des deux conditions doit être vraie pour qu'un enregistrement quelconque soit sélectionné. Ainsi, tout employé dont le poste est CLERK *ou* qui gagne plus de $1100 est sélectionné.

Table de Vérité de l'Opérateur OR

Le tableau suivant montre les résultats obtenus en combinant deux expressions avec OR :

OR	**TRUE**	**FALSE**	**UNKNOWN**
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	UNKNOWN
UNKNOWN	TRUE	UNKNOWN	UNKNOWN

Hanoune

Utilisation de l'Opérateur NOT

```
SQL> SELECT ename, job
  2  FROM   emp
  3  WHERE  job NOT IN ('CLERK','MANAGER','ANALYST');
```

```
ENAME      JOB
---------- ---------
KING       PRESIDENT
MARTIN     SALESMAN
ALLEN      SALESMAN
TURNER     SALESMAN
WARD       SALESMAN
```

```
... WHERE  sal  NOT  BETWEEN  1000 AND  1500
... WHERE  ename NOT LIKE '%A%'
... WHERE  comm  IS  NOT  NULL
```

2-18

L'Opérateur NOT

L'exemple ci-dessus affiche le nom et le poste de tous les employés dont l'intitulé du poste *n'est pas* CLERK, MANAGER ni ANALYST.

Table de Vérité de l'Opérateur NOT

Le tableau suivant montre le résultat de l'utilisation de l'opérateur NOT avec une condition :

NOT	TRUE	FALSE	UNKNOWN
	FALSE	TRUE	UNKNOWN

Remarque : l'opérateur NOT peut également s'utiliser avec d'autres opérateurs SQL comme BETWEEN, LIKE et NULL.

```
... WHERE  sal  NOT  BETWEEN  1000 AND  1500
... WHERE  ename NOT LIKE '%A%'
... WHERE  comm  IS  NOT  NULL
```

Règles de Priorité

Ordre de priorité	Opérateur
1	Tous les opérateurs de comparaison
2	NOT
3	AND
4	OR

Les parenthèses permettent de modifier les règles de priorité

Règles de Priorité

```
SQL> SELECT ename, job, sal
  2  FROM   emp
  3  WHERE  job='SALESMAN'
  4  OR     job='PRESIDENT'
  5  AND    sal>1500;
```

```
ENAME      JOB              SAL
---------- --------- ---------
KING       PRESIDENT       5000
MARTIN     SALESMAN        1250
ALLEN      SALESMAN        1600
TURNER     SALESMAN        1500
WARD       SALESMAN        1250
```

2-20

Exemple de Priorité de l'Opérateur AND

L'exemple contient deux conditions :

- Première condition : le poste (job) est PRESIDENT *et* le salaire est supérieur à 1500.
- Deuxième condition : le poste est SALESMAN.

L'ordre SELECT se lit donc comme suit :

"Sélectionner la ligne si un employé est PRESIDENT *et* s'il gagne plus de $1500 *ou* s'il est SALESMAN."

Règles de Priorité

Utilisation de parenthèses pour forcer la priorité.

```
SQL> SELECT   ename, job, sal
  2  FROM     emp
  3  WHERE    (job='SALESMAN'
  4  OR       job='PRESIDENT')
  5  AND      sal>1500;
```

```
ENAME      JOB              SAL
---------- --------- ---------
KING       PRESIDENT       5000
ALLEN      SALESMAN        1600
```


Utilisation de Parenthèses

Cet exemple contient deux conditions :

- Première condition : le poste (job) est PRESIDENT *ou* SALESMAN.
- Deuxième condition : le salaire est supérieur à 1500.

L'ordre SELECT se lit donc comme suit :

"Sélectionner la ligne si un employé est PRESIDENT ou SALESMAN et s'il gagne plus de $1500."

Clause ORDER BY

- **Tri des lignes avec la clause ORDER BY**
  - **ASC : ordre croissant (par défaut)**
  - **DESC : ordre décroissant**
- **La clause ORDER BY se place à la fin de l'ordre SELECT**

```
SQL> SELECT   ename, job, deptno, hiredate
  2  FROM     emp
  3  ORDER BY hiredate;
```

```
ENAME      JOB           DEPTNO HIREDATE
---------- --------- --------- ---------
SMITH      CLERK             20 17-DEC-80
ALLEN      SALESMAN          30 20-FEB-81
...
14 rows selected.
```

2-22

La Clause ORDER BY

Les lignes trouvées par une requête sont ramenées dans un ordre quelconque. La clause ORDER BY sert à trier les lignes. Si vous l'utilisez, vous devez la placer en dernier dans l'ordre SELECT. Vous pouvez spécifier une expression ou un alias sur lesquels le tri sera effectué.

Syntaxe

```
SELECT    expr
FROM      table
[WHERE    condition (s)]
[ORDER BY {column, expr} [ASC|DESC]];
```

Où : ORDER BY précise l'ordre d'affichage des lignes trouvées.

ASC classe les lignes en ordre croissant. C'est l'ordre par défaut.

DESC classe les lignes en ordre décroissant.

Tri par Ordre Décroissant

```
SQL> SELECT   ename, job, deptno, hiredate
  2  FROM     emp
  3  ORDER BY hiredate DESC;
```

```
ENAME      JOB           DEPTNO HIREDATE
---------- --------- --------- ---------
ADAMS      CLERK             20 12-JAN-83
SCOTT      ANALYST           20 09-DEC-82
MILLER     CLERK             10 23-JAN-82
JAMES      CLERK             30 03-DEC-81
FORD       ANALYST           20 03-DEC-81
KING       PRESIDENT         10 17-NOV-81
MARTIN     SALESMAN          30 28-SEP-81
...
14 rows selected.
```

2-23

Classement des Données par Défaut

L'ordre de tri par défaut est croissant :

- Les valeurs numériques sont affichées à partir des plus petites, par exemple de 1 à 999.
- Les dates sont affichées à partir de la plus ancienne ; par exemple, le 01-JAN-92 sera placé avant le 01-JAN-95.
- Les valeurs caractères sont affichées par ordre alphabétique, par exemple A en premier et Z en dernier.
- Les valeurs NULL sont affichées en dernier dans le cas d'un ordre croissant et en premier dans le cas d'un ordre décroissant.

Inversion de l'Ordre par Défaut

Pour inverser l'ordre d'affichage des lignes, spécifiez le mot-clé DESC après le nom de colonne dans la clause ORDER BY. Dans l'exemple ci-dessus, les résultats sont triés à partir du dernier salarié embauché.

Tri sur l'Alias de Colonne

```
SQL> SELECT   empno, ename, sal*12 annsal
  2  FROM     emp
  3  ORDER BY annsal;
```

```
    EMPNO ENAME          ANNSAL
--------- ---------- ---------
     7369 SMITH          9600
     7900 JAMES         11400
     7876 ADAMS         13200
     7654 MARTIN        15000
     7521 WARD          15000
     7934 MILLER        15600
     7844 TURNER        18000
...
14 rows selected.
```

2-24

Tri sur les Alias de Colonnes

Vous pouvez utiliser un alias de colonne dans la clause ORDER BY. Dans l'exemple ci-dessus, les données sont triées par salaire annuel.

Tri sur Plusieurs Colonnes

- **L'ordre des éléments de la liste ORDER BY donne l'ordre du tri.**

```
SQL> SELECT   ename, deptno, sal
  2  FROM     emp
  3  ORDER BY deptno, sal DESC;
```

```
ENAME          DEPTNO        SAL
---------- --------- ---------
KING               10        5000
CLARK              10        2450
MILLER             10        1300
FORD               20        3000
...
14 rows selected.
```

- **Vous pouvez effectuer un tri sur une colonne ne figurant pas dans la liste SELECT.**

2-25

Tri sur Plusieurs Colonnes

Vous pouvez trier les résultats d'une requête sur plusieurs colonnes, à concurrence du nombre de colonnes présentes dans la table concernée.

Dans la clause ORDER BY, spécifiez les noms de colonnes en les séparant par une virgule. Pour inverser l'ordre de tri d'une colonne, faites suivre son nom du mot-clé DESC. Vous pouvez faire un tri sur des colonnes non incluses dans la clause SELECT.

Exemple

Affichez le nom et le salaire de tous les employés et classez le résultat par numéro de département croissant, puis par salaire décroissant.

```
SQL> SELECT    ename,  sal
  2  FROM      emp
  3  ORDER BY  deptno, sal DESC;
```

Hanoune

Résumé

```
SELECT      [DISTINCT] {*, column [alias], ...}
FROM        table
[WHERE      condition(s)]
[ORDER BY   {column, expr, alias} [ASC|DESC]];
```

2-26

Résumé

Dans ce chapitre, vous avez appris à limiter et trier les lignes ramenées par l'ordre SELECT. Vous avez également vu comment utiliser différents opérateurs.

Présentation des Exercices

- **Interrogation de données et modification de l'ordre des lignes affichées**
- **Restriction des lignes avec la clause WHERE**
- **Utilisation des guillemets dans les alias de colonne**

2-27

Présentation des Exercices

Vous allez effectuer différents exercices faisant appel aux clauses WHERE et ORDER BY.

Exercices 2

1. Créez une requête destinée à afficher le nom et le salaire des employés gagnant plus de $2850.
   Enregistrez l'ordre SQL créé dans un fichier appelé *p2q1.sql*. Exécutez votre requête.

```
ENAME    SAL
-------- ----
KING     5000
JONES    2975
FORD     3000
SCOTT    3000
```

2. Créez une requête destinée à afficher le nom et le numéro de département de l'employé dont le matricule est 7566.

```
ENAME  DEPTNO
------ ------
JONES      20
```

3. Modifiez *p2q1.sql* de manière à afficher le nom et le salaire de tous les employés dont le salaire n'est pas compris entre $1500 et $2850. Enregistrez ce nouvel ordre SQL dans un fichier appelé *p2q3.sql*. Exécutez cette requête.

```
ENAME     SAL
------- -----
KING     5000
JONES    2975
MARTIN   1250
JAMES     950
WARD     1250
FORD     3000
SMITH     800
SCOTT    3000
ADAMS    1100
MILLER   1300
10 rows selected.
```

Exercices 2

4. Affichez le nom, le poste et la date d'entrée (hiredate) des employés embauchés entre le 20 février 1981 et le 1 mai 1981. Classez le résultat par date d'embauche croissante.

```
ENAME   JOB       HIREDATE
------- --------- ----------------
ALLEN   SALESMAN  20-FEB-81
WARD    SALESMAN  22-FEB-81
JONES   MANAGER   02-APR-81
BLAKE   MANAGER   01-MAY-81
```

5. Affichez le nom et le numéro de département de tous les employés des départements 10 et 30 classés par ordre alphabétique des noms.

```
ENAME          DEPTNO
---------- ---------
ALLEN              30
BLAKE              30
CLARK              10
JAMES              30
KING               10
MARTIN             30
MILLER             10
TURNER             30
WARD               30
9 rows selected.
```

6. Modifiez *p2q3.sql* pour afficher la liste des noms et salaires des employés gagnant plus de $1500 et travaillant dans le département 10 ou 30. Nommez les colonnes Employee et Monthly Salary, respectivement. Enregistrez à nouveau votre ordre SQL dans un fichier appelé *p2q6.sql*. Réexécutez votre requête.

```
Employee  Monthly Salary
--------  --------------
KING                5000
BLAKE               2850
CLARK               2450
ALLEN               1600
```

Exercices 2

7. Affichez le nom et la date d'embauche de chaque employé entré en 1982.

```
ENAME  HIREDATE
------ ---------
SCOTT  09-DEC-82
MILLER 23-JAN-82
```

Si vous avez le temps, faites les exercices suivants :

8. Affichez le nom et le poste de tous les employés n'ayant pas de manager.

```
ENAME JOB
----- ---------
KING  PRESIDENT
```

9. Affichez le nom, le salaire et la commission de tous les employés qui perçoivent des commissions. Triez les données dans l'ordre décroissant des salaires et des commissions.

```
ENAME   SAL   COMM
------- ----  ----
ALLEN   1600   300
TURNER  1500     0
MARTIN  1250  1400
WARD    1250   500
```

10. Affichez le nom de tous les employés dont la troisième lettre du nom est un *A*.

```
ENAME
-----------
BLAKE
CLARK
ADAMS
```

11. Affichez le nom de tous les employés dont le nom contient deux *L* et travaillant dans le département 30 ou dont le manager est 7782.

```
ENAME
----------
ALLEN
MILLER
```

Exercices 2

Si vous voulez aller plus loin dans la difficulté, faites les exercices suivants :

12. Affichez le nom, le poste et le salaire de tous les ' CLERK' ou 'ANALYST' dont le salaire est différent de $1000, $3000 ou $5000.

```
ENAME  JOB       SAL
------ ------- -----
JAMES  CLERK     950
SMITH  CLERK     800
ADAMS  CLERK    1100
MILLER CLERK    1300
```

13. Afficher le nom, le salaire et la commission de tous les employés dont le montant de commission est de plus de 10% supérieur au salaire. Enregistrez votre requête sous le nom *p2q13.sql.*

```
ENAME      SAL  COMM
------- ----- -----
MARTIN    1250  1400
```

3

Fonctions Mono-Ligne

Objectifs

A la fin de ce chapitre, vous saurez :

- **Décrire différents types de fonctions SQL**
- **Utiliser les fonctions caractère, numériques et date dans les ordres SELECT**
- **Expliquer les fonctions de conversion**

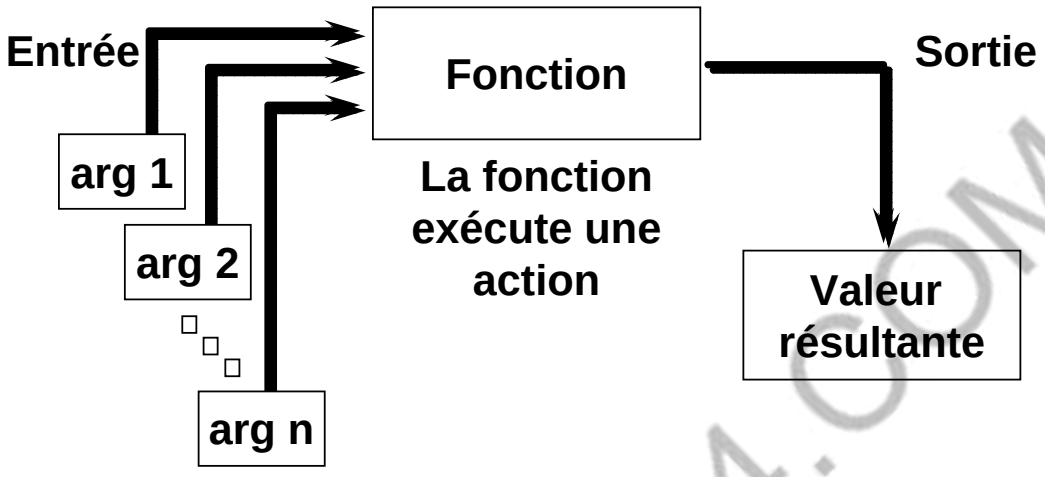
3-2

Objectifs

Les fonctions rendent l'instruction SELECT plus puissante en permettant de manipuler des valeurs de données. Ce chapitre est le premier de deux consacrés aux fonctions. Il explique en particulier les fonctions caractère, numériques et date, ainsi que les fonctions qui convertissent des données d'un certain type en un autre type, par exemple, des données de type caractère en données numériques.

M.Hanoune

Fonctions SQL



3-3

Fonctions SQL

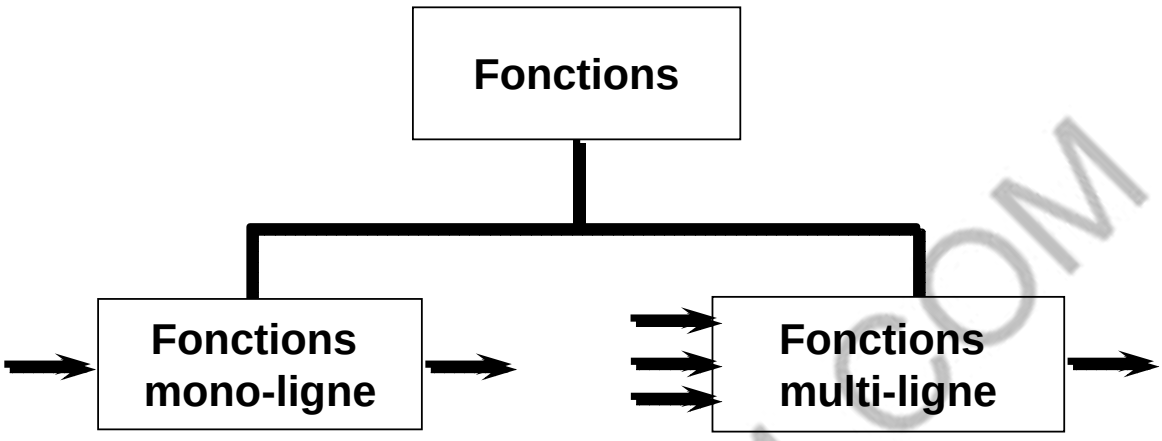
Les fonctions représentent ue caractéristique très puissant de SQL et sont utilisées pour :

- Effectuer des calculs sur des données
- Transformer des données
- Effectuer des calculs sur des groupes de lignes
- Formater des dates et des nombres pour l'affichage
- Convertir des types de données de colonnes

Les fonctions SQL acceptent les arguments et ramènent des valeurs.

Remarque: la plupart des fonctions SQL décrites dans ce chapitre sont spécifiques au SQL d'Oracle.

Deux Types de Fonctions SQL



3-4

Fonctions SQL

Il existe deux types de fonctions :

- Les fonctions mono-ligne
- Les fonctions multi-ligne

Fonctions Mono-Ligne

Ces fonctions agissent sur une seule ligne à la fois et ramènent un seul résultat. Il existe plusieurs types de fonctions mono-ligne. Ce chapitre décrit les quatre suivantes :

- Caractère
- Numérique
- Date
- Conversion

Fonctions Multi-Ligne

Ces fonctions manipulent des groupes de lignes et ramènent un seul résultat par groupe de lignes.

Pour obtenir la syntaxe et la liste complète des fonctions disponibles, reportez-vous à *Oracle 8 Server SQL Reference, Release 8*

M.Hanoune

Fonctions Mono-Ligne

- **Manipulent des éléments de données**
- **Acceptent des arguments et ramènent une valeur**
- **Agissent sur chacune des lignes rapportées**
- **Ramènent un seul résultat par ligne**
- **Peuvent modifier les types de données**
- **Peuvent être imbriquées**

```
function_name (column|expression, [arg1, arg2,...])
```

3-5

Fonctions Mono-Ligne

Les fonctions mono-ligne sont utilisées pour manipuler des éléments de données. Elles acceptent un ou plusieurs arguments et ramènent une seule valeur par ligne issue de la requête. Un argument peut être l'un des éléments suivants :

- Une valeur constante utilisateur
- Une variable
- Un nom de colonne
- Une expression

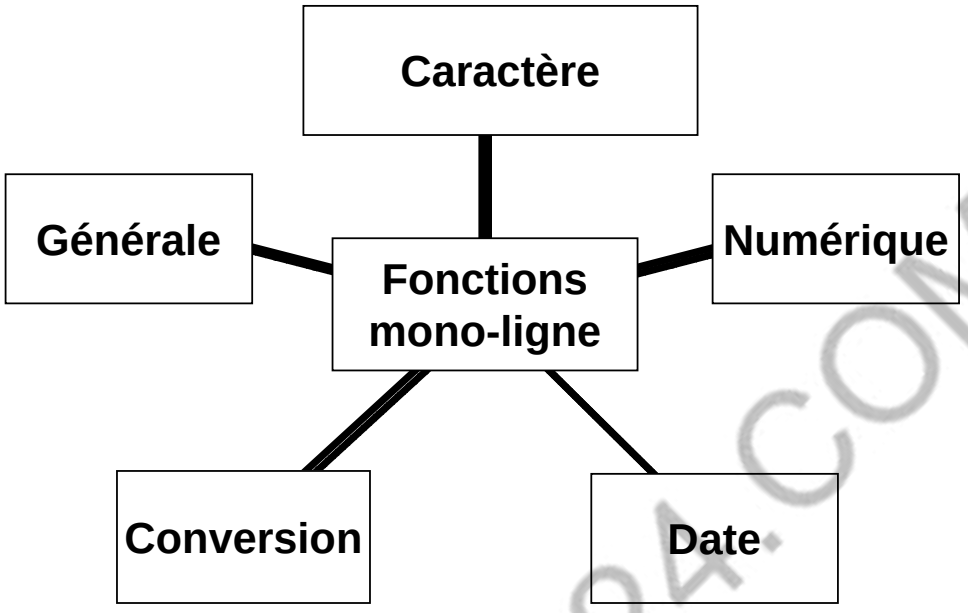
Caractéristiques des Fonctions Mono-Ligne

- Acceptent un ou plusieurs arguments.
- Agissent sur chacune des lignes issues de la requête.
- Ramènent un résultat par ligne.
- Peuvent ramener une valeur d'un type différent de celui mentionné.
- S'utilisent dans les clauses SELECT, WHERE, et ORDER BY.
- Peuvent être imbriquées.

Syntaxe :

function_name	nom de la fonction
column	nom d'une colonne de la base de données
expression	chaîne de caractères ou expression calculée
arg1, arg2	argument utilisé dans la fonction

Fonctions Mono-Ligne



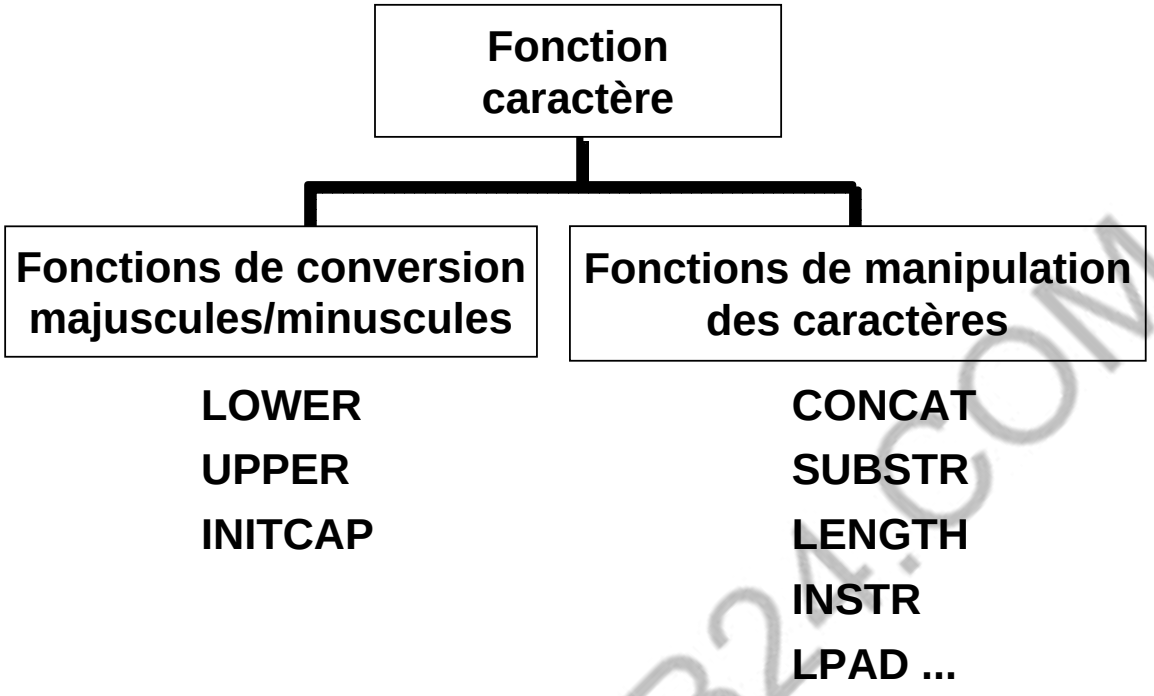
3-6

Fonctions Mono-Ligne

Ce chapitre présente les fonctions mono-ligne suivantes :

- Fonctions caractère : elles acceptent des caractères en entrée et peuvent ramener des valeurs caractère ou numériques.
- Fonctions numériques : elles acceptent des nombres en entrée et ramènent des valeurs numériques.
- Fonctions date : elles opèrent sur des valeurs de type date et ramènent des valeurs de type date. Seule la fonction MONTHS_BETWEEN ramène une valeur numérique.
- Fonctions de conversion : elles convertissent une valeur d'un certain type dans un autre type.
- Fonctions générales
  - fonction NVL
  - fonction DECODE

Fonctions Caractère



3-7

Fonctions caractère

Les fonctions mono-ligne caractère acceptent des données caractère en entrée et ramènent des données caractère ou numériques. Les fonctions caractère se divisent en deux groupes :

- Les fonctions de conversion majuscules/minuscules
- Les fonctions de manipulation des caractères

Fonction	**Modification**
LOWER(*column*|*expression*)	Convertit les caractères alphabétiques en minuscules.
UPPER(*column*|*expression*)	Convertit les caractères alphabétiques en majuscules.
INITCAP(*column*|*expression*)	Convertit l'initiale de chaque mot en majuscule et les caractères suivants en minuscules.
CONCAT(*column1*|*expression1*, *column2*|*expression2*)	Concatène la première chaîne de caractère à la seconde. Equivaut à l'opérateur de concaténation (||).
SUBSTR(*column*|*expression*,*m*[,*n*])	Extrait une partie de la chaîne de caractères en commençant au caractère situé à la position *m* et sur une longueur de *n* caractères. Si *m* est une valeur négative, le décompte s'effectue dans le sens inverse (à partir du dernier caractère de la chaîne). Si *n* est omis, tous les caractères jusqu'à la fin de la chaîne sont ramenés.
LENGTH(*column*|*expression*)	Ramène le nombre de caractères d'une chaîne de caractères.
INSTR(*column*|*expression*,*m*)	Ramène une valeur égale à la position du caractère *m*.
LPAD(*column*|*expression*, *n*, '*string*')	Complète une chaîne de caractère sur la gauche avec la chaîne *'string'* pour parvenir à une longueur totale de *n* caractères.

Remarque : cette liste de fonctions caractère n'est pas complète :

Pour plus d'informations, reportez-vous à
Oracle8 Server SQL Reference, Release 8, "Character Functions."

Fonctions de Conversion Majuscules/Minuscules

Fonction	Résultat
LOWER('Cours SQL')	cours sql
UPPER('Cours SQL')	COURS SQL
INITCAP('Cours SQL')	Cours Sql

Fonctions de Conversion Majuscules/Minuscules

LOWER, UPPER, et INITCAP sont les trois fonctions qui modifient la casse.

- LOWER : Convertit tous les caractères d'une chaîne en minuscules
- UPPER : Convertit tous les caractères d'une chaîne en majuscules
- INITCAP : Convertit la première lettre de chaque mot en majuscule et les lettres suivantes en minuscules

```
SQL> SELECT  'The job title for '||INITCAP(ename)||' is '
  2           ||LOWER(job) AS "EMPLOYEE DETAILS"
  3  FROM    emp;
```

```
EMPLOYEE DETAILS
----------------------------------------
The job title for King is president
The job title for Blake is manager
The job title for Clark is manager
...
14 rows selected.
```

M.Hanoune

Utilisation des Fonctions de Conversion Majuscules/Minuscules

Afficher le matricule, le nom et le numéro de département de l'employé Blake.

```
SQL> SELECT   empno, ename, deptno
  2  FROM     emp
  3  WHERE    ename = 'blake';
no rows selected
```

```
SQL> SELECT   empno, ename, deptno
  2  FROM     emp
  3  WHERE    LOWER(ename) = 'blake';
```

```
     EMPNO ENAME          DEPTNO
--------- ---------- ---------
      7698 BLAKE              30
```

3-9

Fonctions de Conversion Majuscules/Minuscules

L'exemple ci-dessus affiche le matricule, le nom et le numéro de département de l'employé BLAKE.

La clause WHERE du premier ordre SQL spécifie le nom de l'employé sous la forme 'blake'. Comme les données de la table EMP sont stockées en majuscules, il est impossible de trouver le nom correspondant dans la table EMP, et par conséquent de sélectionner des lignes.

La clause WHERE du second ordre SQL indique que le nom d'employé dans la table EMP doit être converti en minuscules pour être ensuite comparé à 'blake'. Les deux noms étant maintenant en minuscules, la ligne correspondante est ramenée. La clause WHERE, réécrite de la manière suivante, produit le même résultat :

```
… WHERE   ename = 'BLAKE'
```

Le nom de l'employé apparaît à l'affichage tel qu'il est stocké dans la base de données. Pour obtenir le nom avec seulement l'initiale en majuscule, il suffit d'utiliser la fonction INITCAP dans l'ordre SELECT.

```
SQL> SELECT   empno, INITCAP(ename), deptno
  2  FROM     emp
  3  WHERE    LOWER(ename) = 'blake';
```

M.Hanoune

Fonctions de Manipulation des Caractères

Manipulation de chaînes de caractères

Fonction	Résultat
CONCAT('Une', 'Chaîne')	UneChaîne
SUBSTR('Chaîne',1,3)	Cha
LENGTH('Chaîne')	6
INSTR('Chaîne', 'a')	3
LPAD(sal,10,'*')	******5000

3-10

Fonctions de Manipulation des Caractères

CONCAT, SUBSTR, LENGTH, INSTR et LPAD sont les cinq fonctions de manipulation des caractères étudiées dans ce chapitre.

- CONCAT : Concatène des valeurs. Le nombre de paramètres avec CONCAT est limité à deux.
- SUBSTR : Extrait une chaîne de longueur déterminée.
- LENGTH : Fournit la valeur numérique correspondant au nombre de caractères d'une chaîne.
- INSTR : Fournit la valeur numérique correspondant à la position d'un caractère.
- LPAD : Ajoute des caractères de remplissage à la gauche d'une valeur alphanumérique qui sera ainsi cadrée à droite.

Remarque : la fonction de manipulation des caractères RPAD ajoute des caractères de remplissage à la droite d'une valeur alphanumérique qui sera ainsi cadrée à gauche.

Utilisation des Fonctions de Manipulation des Caractères

```
SQL> SELECT ename, CONCAT (ename, job), LENGTH(ename),
  2         INSTR(ename, 'A')
  3 FROM    emp
  4 WHERE   SUBSTR(job,1,5) = 'SALES';
```

ENAME	CONCAT(ENAME,JOB)	LENGTH(ENAME)	INSTR(ENAME,'A')
MARTIN	MARTINSALESMAN	6	2
ALLEN	ALLENSALESMAN	5	1
TURNER	TURNERSALESMAN	6	0
WARD	WARDSALESMAN	4	2

3-11

Fonctions de Manipulation des Caractères

L'exemple ci-dessus affiche le nom des employés concaténé à leur poste, le nombre de caractères du nom, ainsi que la position de la lettre A dans leur nom et ce, pour tous les employés dont le poste commence par la chaîne 'SALES'.

Exemple

Modifier l'ordre SQL ci-dessus afin d'afficher les données concernant les employés dont le nom se termine par un N.

```
SQL> SELECT  ename, CONCAT(ename, job), LENGTH(ename),
        INSTR(ename, 'A')
  2  FROM    emp
  3  WHERE   SUBSTR(ename, -1, 1) = 'N';
```

```
ENAME    CONCAT(ENAME,JOB)   LENGTH(ENAME) INSTR(ENAME,'A')
-------- ------------------- ------------- ----------------
MARTIN   MARTINSALESMAN                  6                2
ALLEN    ALLENSALESMAN                   5                1
```

M.Hanoune

Fonctions Numériques

- **ROUND : Arrondit la valeur à la précision spécifiée**

  ROUND(45.926, 2) ⟶ 45.93

- **TRUNC : Tronque la valeur à la précision spécifiée**

- **TRUNC(45.926, 2) ⟶ 45.92**

- **MOD : Ramène le reste d'une division**

  MOD(1600, 300) ⟶ 100

3-12

Fonctions numériques

Les fonctions numériques utilisent et ramènent des valeurs numériques. Cette section décrit quelques-unes de ces fonctions.

Fonction	Modification
ROUND(*column*|*expression*, *n*)	Arrondit la valeur de la colonne ou de l'expression à une précision de 10^{-n}. Si *n* est positif, le nombre sera arrondi à *n* décimales. Si *n* est omis, il n'y aura pas de décimale. Si *n* est négatif, l'arrondi portera sur la partie du nombre située à gauche de la virgule (dizaine, centaine...)
TRUNC(*column*|*expression*,*n*)	Tronque la valeur de la colonne ou de l'expression à une précision de 10^{-n}. Si *n* est positif, le nombre sera tronqué à *n* décimales. Si *n* est omis, il n'y aura pas de décimale. Si *n* est négatif, ce sera la partie du nombre située à gauche de la virgule (dizaine, centaine...) qui sera tronquée.
MOD(*m*,*n*)	Ramène le reste de la division de *m* par *n*.

Remarque : Cette liste de fonctions numériques n'est pas exhaustive.

Pour plus d'informations, reportez-vous à
Oracle8 Server SQL Reference, Release 8, "Number Functions."

Utilisation de la Fonction ROUND

Affichage de la valeur 45.923 arrondie au centième, à 0 décimale et à la dizaine supérieure.

```
SQL> SELECT ROUND(45.923,2), ROUND(45.923,0),
  2         ROUND(45.923,-1)
  3  FROM   DUAL;
```

```
ROUND(45.923,2) ROUND(45.923,0) ROUND(45.923,-1)
--------------- --------------- ----------------
          45.92              46               50
```

3-13

Fonction ROUND

La fonction ROUND arrondit la valeur d'une colonne ou d'une expression à une précision égale à 10^{-n}. Lorsque *n* vaut 0 ou est absent, la valeur est arrondie à zéro décimale. Si *n* vaut 2, la valeur est arrondie à deux décimales (soit au centième). Inversement, si *n* vaut -2, la valeur est arrondie de deux chiffres à gauche de la virgule, soit à la centaine.

La fonction ROUND peut aussi être utilisée avec les fonctions date. Nous verrons quelques exemples un peu plus loin dans ce chapitre.

La table DUAL est une table factice. Nous y reviendrons ultérieurement.

Utilisation de la Fonction TRUNC

Affichage de la valeur 45.923 tronquée au centième, à 0 décimale et à la dizaine.

```
SQL> SELECT TRUNC(45.923,2), TRUNC(45.923),
  2         TRUNC(45.923,-1)
  3  FROM   DUAL;
```

TRUNC(45.923,2)	TRUNC(45.923)	TRUNC(45.923,-1)
45.92	45	40

3-14

Fonction TRUNC

La fonction TRUNC tronque la valeur de la colonne ou de l'expression à une précision égale à 10^{-n}.

La fonction TRUNC fonctionne avec des arguments identiques à ceux de la fonction ROUND. Lorsque *n* vaut 0 ou est absent, la valeur est tronquée à zéro décimale. Si *n* vaut 2, la valeur est tronquée à deux décimales (soit au centième). Inversement, si *n* vaut -2, la valeur est tronquée de deux chiffres à gauche de la virgule, soit à la centaine.

Comme la fonction ROUND, la fonction TRUNC peut aussi être utilisée avec les fonctions date.

Utilisation de la Fonction MOD

Calculer le reste de la division salaire par commission pour l'ensemble des employés ayant un poste de vendeur.

```
SQL> SELECT   ename, sal, comm, MOD(sal, comm)
  2  FROM     emp
  3  WHERE    job = 'SALESMAN';
```

ENAME	SAL	COMM	MOD(SAL,COMM)
MARTIN	1250	1400	1250
ALLEN	1600	300	100
TURNER	1500	0	1500
WARD	1250	500	250

3-15

Fonction MOD

La fonction MOD calcule le reste de la division d'une valeur1 par une valeur2. L'exemple ci-dessus donne le reste de la division du salaire par la commission, pour tous les employés occupant un poste de vendeur (SALESMAN).

M.Hanoune

Utilisation des Dates

- **Oracle stocke les dates dans un format numérique interne : siècle, année, mois, jour, heures, minutes, secondes.**
- **Le format de date par défaut est DD-MON-YY.**
- **La fonction SYSDATE ramène la date et l'heure courante.**
- **DUAL est une table factice qu'on peut utiliser pour visualiser SYSDATE.**

3-16

Format de Date Oracle

Oracle stocke les dates dans un format numérique interne représentant le siècle, l'année, le mois, le jour, les heures, les minutes et les secondes.

Le format d'entrée et d'affichage par défaut des dates est DD-MON-YY. Les dates valides pour Oracle sont comprises entre le 1er janvier 4712 av.J.-C. et le 31 décembre 9999 apr.J.-C.

SYSDATE

SYSDATE est une fonction date qui permet d'obtenir la date et l'heure courante. SYSDATE s'utilise de la même façon qu'un nom de colonne quelconque. Il est usuel d'interroger la table "factice" DUAL.

DUAL

La table DUAL appartient à l'utilisateur SYS, mais tous les utilisateurs peuvent y accéder. Elle contient une seule colonne, DUMMY, et une seule ligne contenant la valeur X. La table DUAL est utile lorsque vous souhaitez ramener une valeur une seule fois, par exemple, la valeur d'une constante, d'une pseudo-colonne ou d'une expression qui ne dépend pas d'une table de données utilisateur.

Exemple

Afficher la date courante au moyen de la table DUAL.

```
SQL> SELECT SYSDATE
  2  FROM   DUAL;
```

Opérations Arithmétiques sur les Dates

- **Ajout ou soustraction d'un nombre à une date pour obtenir un résultat de type *date*.**
- **Soustraction de deux dates afin de déterminer le *nombre* de jours entre ces deux dates.**
- **Ajout *d'un nombre d'heures* à une date en divisant le nombre d'heures par 24.**

3-17

Opérations Arithmétiques sur les Dates

Comme la base de donnée stocke les dates en tant que données numériques, il est possible d'effectuer des calculs tels que l'addition ou la soustraction au moyen d'opérateurs arithmétiques. Il est possible d'ajouter et soustraire des constantes numériques aussi bien que des dates.

Les opérations possibles sont les suivantes :

Opération	Résultat	Description
date + nombre de jours	Date	Ajoute un certain nombre de jours à une date
date - nombre de jours	Date	Soustrait un certain nombre de jours d'une date
date - date	Nombre de jours	Soustrait une date d'une autre
date + (nombre d'heures)/24	Date	Ajoute un certain nombre d'heures à une date

Utilisation d'Opérateurs Arithmétiques avec les Dates

```
SQL> SELECT ename, (SYSDATE-hiredate)/7 WEEKS
  2  FROM   emp
  3  WHERE  deptno = 10;
```

```
ENAME          WEEKS
---------- ---------
KING       830.93709
CLARK      853.93709
MILLER     821.36566
```

3-18

Opérations Arithmétiques sur les Dates (suite)

L'exemple de la diapositive affiche le nom et le nombre de semaines d'ancienneté de tous les employés du département 10. La date courante (SYSDATE) est soustraite de la date d'embauche de l'employé, puis le résultat est divisé par 7 pour obtenir le nombre de semaines d'ancienneté.

Remarque : SYSDATE est une fonction SQL qui donne l'heure et la date courantes. Vos résultats peuvent donc être différents de ceux de l'exemple.

M.Hanoune

Fonctions Date

FONCTION	DESCRIPTION
MONTHS_BETWEEN	Nombre de mois situés entre deux dates
ADD_MONTHS	Ajoute des mois calendaires à une date
NEXT_DAY	Jour qui suit la date spécifiée
LAST_DAY	Dernier jour du mois
ROUND	Arrondit une date
TRUNC	Tronque une date

3-19

Fonctions Date

Les fonctions date s'appliquent aux données de type DATE. Toutes les fonctions date ramènent une valeur de type DATE, à l'exception de MONTHS_BETWEEN qui ramène une valeur numérique.

- MONTHS_BETWEEN(*date1, date2*) : Donne le nombre de mois situés entre une date (*date1)* et une autre date (*date2)*. Le résultat peut être positif ou négatif. Si *date1* est postérieure à *date2*, le résultat est positif ; si *date1* est antérieure à *date2*, le résultat est négatif. La partie non entière du résultat représente une portion de mois.
- ADD_MONTHS(*date, n*) : Ajoute un nombre *n* de mois à une *date*. *n* doit être un nombre entier et peut être négatif.
- NEXT_DAY(*date,* '*char*') : Fournit la date de la première occurrence du jour spécifié ('*char*') après la *date* fournie. *char* peut être, soit un numéro de jour de semaine, soit une chaîne de caractères.
- LAST_DAY(*date*) : Indique la date du dernier jour du mois auquel appartient la *date* indiquée.
- ROUND(*date*[,'*fmt*']) : Ramène la *date,* arrondie à l'unité précisée par le modèle de format *fmt*. Lorsque *fmt* est omis, la *date* est arrondie est arrondie au jour le plus proche.
- TRUNC(*date*[, '*fmt*']) : Ramène la *date*, tronquée à l'unité précisée par le modèle de format *fmt*. Lorsque *fmt* est omis, la date est tronquée au jour.

Cette liste des fonctions date n'est pas exhaustive. Les modèles de format sont expliqués dans la suite de ce chapitre. Le mois ou l'année sont des exemples de modèles de format.

M.Hanoune

Utilisation des Fonctions Date

- MONTHS_BETWEEN ('01-SEP-95','11-JAN-94') → 19.6774194
- ADD_MONTHS ('11-JAN-94',6) → '11-JUL-94'
- NEXT_DAY ('01-SEP-95','FRIDAY') → '08-SEP-95'
- LAST_DAY('01-SEP-95') → '30-SEP-95'

3-20

Fonctions Date (suite)

Pour tous les employés ayant moins de 200 mois d'ancienneté, affichez les données suivantes : le matricule, la date d'embauche, le nombre de mois d'ancienneté, la date correspondant à la révision de salaire après 6 mois , le premier vendredi suivant la date d'embauche et le dernier jour du mois d'embauche.

```
SQL> SELECT  empno, hiredate,
  2          MONTHS_BETWEEN(SYSDATE, hiredate) TENURE,
  3          ADD_MONTHS(hiredate, 6) REVIEW,
  4          NEXT_DAY(hiredate, 'FRIDAY'), LAST_DAY(hiredate)
  5  FROM    emp
  6  WHERE   MONTHS_BETWEEN (SYSDATE, hiredate)<200;
```

```
    EMPNO HIREDATE     TENURE REVIEW    NEXT_DAY( LAST_DAY(  -------
-- --------- --------- --------- --------- ---------
     7839 17-NOV-81 192.24794 17-MAY-82 20-NOV-81 30-NOV-81
     7698 01-MAY-81 198.76407 01-NOV-81 08-MAY-81 31-MAY-81
...
11 rows selected.
```

M.Hanoune

Utilisation des Fonctions Date

- ROUND('25-JUL-95','MONTH') → 01-AUG-95
- ROUND('25-JUL-95','YEAR') → 01-JAN-96
- TRUNC('25-JUL-95','MONTH') → 01-JUL-95
- TRUNC('25-JUL-95','YEAR') → 01-JAN-95

3-21

Fonctions Date (suite)

Les fonctions ROUND et TRUNC peuvent être utilisées avec des valeurs de type numérique ou date. Utilisées avec des dates, ces fonctions arrondissent ou tronquent au modèle de format spécifié. Vous pouvez par conséquent arrondir les dates au premier jour du mois ou de l'année les plus proches.

Exemple

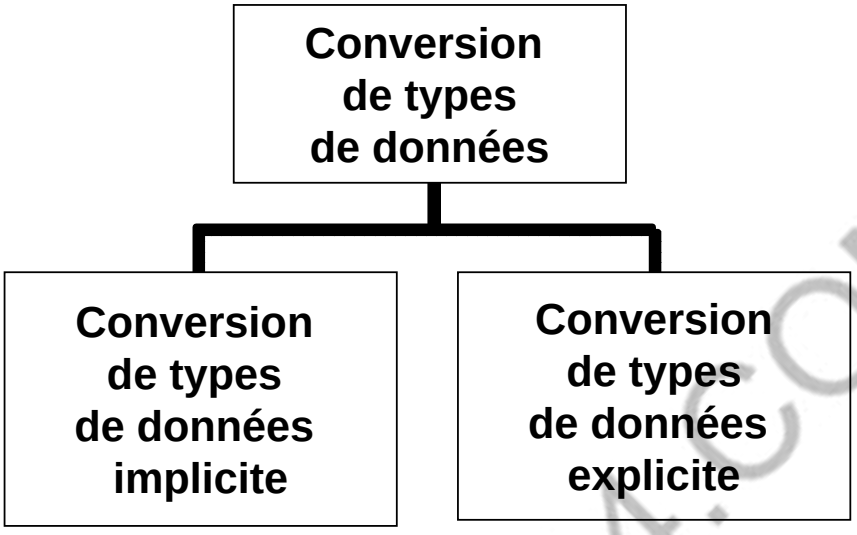
Afficher les dates d'embauche de tous les employés ayant commencé en 1987. Affichez le matricule, la date d'embauche et le mois de début d'activité en utilisant les fonctions ROUND et TRUNC.

```
SQL> SELECT empno, hiredate,
  2         ROUND(hiredate, 'MONTH'), TRUNC(hiredate, 'MONTH')
  3  FROM   emp
  4  WHERE  hiredate like '%87';
```

```
    EMPNO HIREDATE  ROUND(HIR TRUNC(HIR
--------- --------- --------- ---------
     7788 19-APR-87 01-MAY-87 01-APR-87
     7876 23-MAY-87 01-JUN-87 01-MAY-87
```

M.Hanoune

Fonctions de Conversion



3-22

Fonctions de Conversion

Il est possible d'utiliser des types de données ANSI, DB2 et SQL/DS, en plus des types de données Oracle, pour définir les colonnes d'une table dans une base de données Oracle8. Toutefois, Oracle8 Server convertit en interne ces types de données en types de données Oracle8.

Dans certains cas, Oracle8 peut accepter des données d'un type différent de celui normalement attendu, sous réserve qu'Oracle Server puisse effectuer une conversion automatique de ces données. Cette conversion de types de données est réalisée, soit de manière *implicite* par Oracle8 Server, soit de manière *explicite* par l'utilisateur.

Les conversions implicites de types de données fonctionnent selon des règles que nous allons expliquer dans les deux diapositives suivantes.

Les conversions explicites des types de données se font au moyen des fonctions de conversion, qui convertissent une valeur d'un type de données en un autre type. En principe, le format de la fonction suit la convention *datatype* TO *datatype*, le premier *datatype* étant le type de données d'entrée, le second *datatype* étant le type de données restitué.

Remarque : Bien que la conversion implicite des types de données soit possible, il est recommandé d'effectuer des conversions explicites afin d'assurer une meilleure efficacité des ordres SQL.

M.Hanoune

Conversion de Types de Données Implicite

Pour les affectations, Oracle effectue automatiquement les conversions suivantes

De	Vers
VARCHAR2 ou CHAR	NUMBER
VARCHAR2 ou CHAR	DATE
NUMBER	VARCHAR2
DATE	VARCHAR2

3-23

Conversion de Types de Données Implicite

Pour les affectations, Oracle8 Server peut convertir automatiquement les types de données suivants :

- VARCHAR2 ou CHAR vers NUMBER
- VARCHAR2 ou CHAR vers DATE
- NUMBER vers VARCHAR2
- DATE vers VARCHAR2

L'affectation réussit si Oracle Server parvient à convertir le type de données de la valeur à affecter dans le type de données de la cible de l'affectation.

M.Hanoune

Conversion de Types de Données Implicite

Pour l’évaluation d’expressions, Oracle effectue automatiquement les conversions suivantes

De	Vers
VARCHAR2 ou CHAR	NUMBER
VARCHAR2 ou CHAR	DATE

3-24

Conversion de Types de Données Implicite

Pour l'évaluation d'expressions, Oracle Server peut convertir automatiquement les valeurs suivantes :

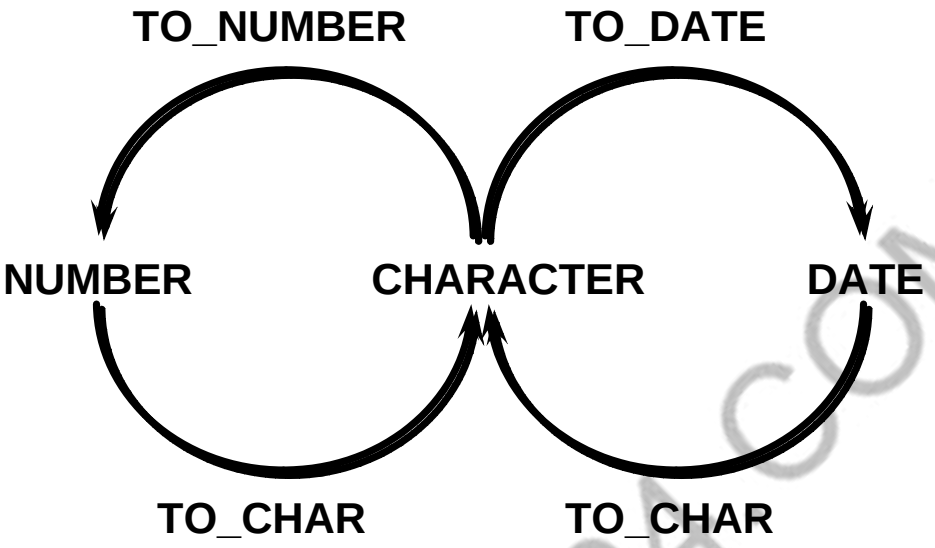
- VARCHAR2 ou CHAR vers NUMBER
- VARCHAR2 ou CHAR vers DATE

Pour des raisons de lisibilité, de performance et d'évolution ultérieure des algorithmes Oracle, il est recommandé d'utiliser la conversion explicite des types de données.

Remarque : les conversions CHAR vers NUMBER ne sont possibles que lorsque le nombre représenté par la chaîne de caractères est valide. Les conversions CHAR vers DATE ne fonctionnent que si la chaîne de caractères est dans le format par défaut DD-MON-YY.

M.Hanoune

Conversion de Types de Données Explicite



3-25

Conversion de Types de Données Explicite

SQL offre trois fonctions pour convertir une valeur d'un certain type de données dans autre type.

Fonction	**Résultat**
TO_CHAR(*number*|*date*,[‘*fmt*’])	Convertit un nombre ou une date en une chaîne de caractères de type VARCHAR2 et de format *fmt*.
TO_NUMBER(*char*)	Convertit une chaîne de caractères en un nombre.
TO_DATE(*char*,[‘*fmt*’])	Convertit une chaîne de caractères représentant une date au format *fmt* en une date Oracle. Lorsque *fmt* est omis, le format est DD-MON-YY.

Remarque : cette liste de fonctions de conversion n'est pas exhaustive.

Pour plus d'informations, reportez-vous à
Oracle8 Server SQL Reference, Release 8, "Conversion Functions."

M.Hanoune

Utilisation de la Fonction TO_CHAR avec les Dates

```
TO_CHAR(date, 'fmt')
```

Le modèle de format :

- **Doit être placé entre simples quotes et différencie les majuscules et minuscules.**
- **Peut inclure tout élément valide de format date**
- **Comporte un élément *fm* qui supprime les espaces de remplissage ou les zéros de tête**
- **Est séparé de la valeur date par une virgule**

3-26

Affichage d'une Date dans un Format Spécifique

Nous avons vu que Oracle Server affiche les dates au format DD-MON-YY. La fonction TO-CHAR permet de convertir ces dates en un autre format de votre choix.

Conseils

- Le modèle de format doit être placé entre simples quotes et différencie les majuscules et minuscules.
- Il peut comprendre tout élément valide de format date. N'oubliez pas de séparer la date et le modèle de format par une virgule.
- Les noms de jours et de mois sont automatiquement complétés par des espaces.
- Pour supprimer les espaces de remplissage ou les zéros de tête, utilisez l'élément *fm*. *fm* siginfie fill mode, ou mode de remplissage.
- La colonne résultante a une largeur par défaut de 80 caractères.
- Vous pouvez redimensionner la largeur de la colonne résultante avec la commande SQL*Plus COLUMN.

```
SQL> SELECT  empno, TO_CHAR(hiredate, 'MM/YY') Month_Hired
  2  FROM    emp
  3  WHERE   ename = 'BLAKE';
```

M.Hanoune

Modèles de Format Date

YYYY	Année exprimée avec 4 chiffres
YEAR	Année exprimée en toutes lettres
MM	Mois exprimé avec 2 chiffres
MONTH	Mois exprimé en toutes lettres
DY	3 premières lettres du nom du jour
DAY	Jour exprimé en toutes lettres

3-27

Eléments de Format de Date Valides

Elément	Description
SCC ou CC	Siècle; le S fait précéder les dates av. J.C. d'un signe -
YYYY ou SYYYY	Année; le S fait précéder les dates av. J.C. d'un signe -
YYY ou YY ou Y	Les 3, 2 ou 1 derniers chiffres de l'année
Y,YYY	Année avec une virgule insérée
IYYY, IYY, IY, I	Les 4, 3, 2 ou 1 derniers chiffres de l'année (norme ISO)
SYEAR ou YEAR	Année en toutes lettres ; le S fait précéder les dates av. J.C. d'un signe -
BC ou AD	Respectivement, av. JC ou apr. JC
B.C. ou A.D.	Respectivement, av. J.C. ou apr. J.C.
Q	Numéro du trimestre
MM	Mois exprimé avec 2 chiffres
MONTH	Mois en toutes lettres complété par des blancs à concurrence de 9 caractères
MON	3 premières lettres du nom du mois
RM	Numéro du mois en chiffres romains
WW ou W	Numéro de la semaine dans l'année ou le mois
DDD ou DD ou D	Numéros du jour dans l'année, le mois ou la semaine
DAY	Nom du jour exprimé en toutes lettres et complété par des blancs à concurrence de 9 caractères
DY	3 premières lettres du nom du jour
J	Jour du calendrier Julien ; nombre de jours depuis le 31 décembre 4713 av.J.C

M.Hanoune

Modèles de Format pour les Dates

- **Les éléments horaires formatent la partie horaire de la date.**

HH24:MI:SS AM	15:45:32 PM

- **Pour ajouter des chaînes de caractères, les placer entre guillemets.**

DD "of" MONTH	12 of OCTOBER

- **Différents suffixes existent pour les nombres.**

ddspth	fourteenth

3-28

Formats Horaires

Utilisez les formats suivants pour afficher des informations et littéraux de type heure, et pour transformer des valeurs numériques en caractères.

Elément	Description
AM ou PM	Respectivement, matin ou après-midi
A.M. ou P.M.	Respectivement, matin ou après-midi avec points
HH ou HH12 ou HH24	Heure du jour ou heure (1 à 12) ou heure (0 à 23)
MI	Minutes (0 à 59)
SS	Secondes (0 à 59)
SSSSS	Secondes après minuit (0 à 86399)

Autres Formats

Elément	Description
/ . , :	La ponctuation est reproduite dans le résultat
‘‘ of the ’’	Les chaînes entre guillemets sont reproduites telles quelles dans le résultat

Ajout de Suffixes pour Modifier L'Affichage des Données Numériques

Elément	Description
TH	Nombre ordinal (par exemple, DDTH pour 4TH)
SP	Nombre en toutes lettres (par exemple, DDSP pour FOUR)
SPTH ou THSP	Nombres ordinaux en toutes lettres (par exemple, DDSPTH pour FOURTH)

M.Hanoune

Format de Date RR

Année en Cours	Date Spécifiée	Format RR	Format YY
1995	27-OCT-95	1995	1995
1995	27-OCT-17	2017	1917
2001	27-OCT-17	2017	2017
2001	27-OCT-95	1995	2095

		Si l'année spécifiée est située entre	
		0-49	50-99
Si les 2 chiffres de l'année en cours sont	0-49	La nouvelle date appartient au siècle courant.	La nouvelle date appartient au siècle précédent.
	50-99	La nouvelle date appartient au siècle suivant.	La nouvelle date appartient au siècle courant.

3-29

Elément de Format de Date RR

Le format de date RR est identique à l'élément YY, mais permet en outre de changer de siècle. Vous pouvez l'utiliser à la place du format YY pour faire varier le siècle en fonction des 2 chiffres de l'année spécifiée et des deux derniers chiffres de l'année en cours. Le tableau de la diapositive résume le fonctionnement de l'élément RR.

Année en cours	Date spécifiée	Interprétée RR	Interprétée YY
1994	27-OCT-95	1995	1995
1994	27-OCT-17	2017	1917
2001	27-OCT-17	2017	2017

M.Hanoune

Utilisation de la Fonction TO_CHAR avec les Dates

```
SQL> SELECT ename,
  2         TO_CHAR(hiredate, 'fmDD Month YYYY') HIREDATE
  3  FROM   emp;
```

```
ENAME      HIREDATE
---------- -----------------
KING       17 November 1981
BLAKE      1 May 1981
CLARK      9 June 1981
JONES      2 April 1981
MARTIN     28 September 1981
ALLEN      20 February 1981
...
14 rows selected.
```

3-30

Utilisation de la Fonction TO_CHAR avec les Dates

L'ordre SQL ci-dessus affiche le nom et la date d'embauche de tous les employés. La date est affichée sous la forme 17 November 1981.

Exemple

Modifier l'exemple ci-dessus afin d'obtenir l'affichage de la date dans le format suivant : Seventh of February 1981 08:00:00 AM.

```
SQL>SELECT ename,
  2        TO_CHAR(hiredate, 'fmDdspth "of" Month YYYY fmHH:MI:SS AM')
  3        HIREDATE
  4  FROM   emp;
```

```
ENAME      HIREDATE
---------- ------------------------------------------------
KING       Seventeenth of November 1981 12:00:00 AM
BLAKE      First of May 1981 12:00:00 AM
...
14 rows selected.
```

Remarquez que le mois suit le modèle de format spécifié (INITCAP).

M.Hanoune

Utilisation de la Fonction TO_CHAR avec les Nombres

```
TO_CHAR(number, 'fmt')
```

Utilisez les formats suivants avec TO_CHAR pour afficher un nombre sous la forme d'une chaîne de caractère.

9	Représente un chiffre
0	Force l'affichage du zéro
$	Place un signe dollar flottant
L	Utilise le symbole monétaire local flottant
.	Imprime un point décimal
,	Imprime un séparateur de milliers

3-31

Utilisation de la Fonction TO_CHAR avec les Nombres

Pour pouvoir afficher des valeurs numériques sous la forme de chaînes de caractères, il convient de convertir ces nombres en données de type caractère avec la fonction TO_CHAR, qui transforme une valeur de type NUMBER en un type VARCHAR2. Cette technique est très utile pour la concaténation.

Eléments de Format Numérique

Pour convertir un nombre en un type caractère, utilisez les éléments suivants :

Elément	Description	Exemple	Résultat
9	Le nombre de 9 détermine la largeur maximum de l'affichage	999999	1234
0	Affichage des zéros de gauche	099999	001234
$	Signe dollar flottant	$999999	$1234
L	Symbole monétaire local flottant	L999999	FF1234
.	Point décimal à l'emplacement spécifié	999999.99	1234.00
,	Séparateur de milliers à l'emplacement spécifié	999,999	1,234
MI	Signe moins à droite (valeurs négatives)	999999MI	1234-
PR	Place les nombres négatifs entre crochets angulaires	999999PR	<1234>
EEEE	Notation scientifique (indiquer obligatoirement quatre E)	99.999EEEE	1.234E+03
V	Multiplie par 10 *n* fois (*n* = nombre de chiffres aprèsV)	9999V99	123400
B	N'affiche pas les zéros non significatifs.	B9999.99	1234.00

Utilisation de la Fonction TO_CHAR avec les Nombres

```
SQL> SELECT   TO_CHAR(sal,'$99,999') SALARY
  2  FROM     emp
  3  WHERE    ename = 'SCOTT';
```

```
  SALARY
--------
  $3,000
```

3-32

Conseils

- Oracle Server affiche une chaîne de signes dièse (#) lorsque le nombre de chiffres de la valeur excède le nombre de chiffres spécifié dans le modèle de format.
- Oracle Server arrondit la valeur décimale au nombre de décimales spécifié dans le modèle de format.

M.Hanoune

Fonctions TO_NUMBER et TO_DATE

- **Conversion d'une chaîne de caractères en format numérique avec la fonction TO_NUMBER**

```
TO_NUMBER(char)
```

- **Conversion d'une chaîne de caractères en format date avec la fonction TO_DATE**

```
TO_DATE(char[, 'fmt'])
```

3-33

Fonctions TO_NUMBER et TO_DATE

Vous pouvez convertir une chaîne de caractères en format numérique ou date en utilisant respectivement les fonctions TO_NUMBER ou TO_DATE. Pour TO_DATE, le modèle de format à spécifier est basé sur les éléments de format déjà expliqués. Ce modèle de format doit décrire le format de la chaîne fournie en entrée.

Exemple

Afficher le nom et la date d'embauche de tous les employés entrés le "February 22, 1981".

```
SQL> SELECT ename, hiredate
  2  FROM   emp
  3  WHERE  hiredate = TO_DATE('February 22, 1981', 'Month dd, YYYY');
```

```
ENAME      HIREDATE
---------- --------
WARD       22-FEB-81
```

Fonction NVL

Convertit une valeur NULL en une valeur réelle

- **Fonctionne avec les données de type date, caractère et numérique.**
- **Les types de données doivent correspondre**
  - **NVL(comm,0)**
  - **NVL(hiredate,'01-JAN-97')**
  - **NVL(job,'No Job Yet')**

3-34

La fonction NVL

Pour transformer une valeur NULL en une valeur réelle, on utilise la fonction NVL.

Syntaxe

```
NVL (expr1, expr2)
```

où : *expr1* est l'expression ou la valeur source suceptible de contenir une valeur NULL

expr2 est la valeur de remplacement pour la valeur NULL

La fonction NVL permet de convertir n'importe quel type de données, mais toutefois, la valeur de remplacement doit être de même type que la valeur de l'expression *expr1*.

Conversions NVL pour Divers Types de Données

Type de données	Exemple de conversion
NUMBER	NVL(*number_column*,9)
DATE	NVL(*date_column*, '01-JAN-95')
CHAR ou VARCHAR2	NVL(*character_column*, 'Unavailable')

Utilisation de la Fonction NVL

```
SQL> SELECT ename, sal, comm, (sal*12)+NVL(comm,0)
  2  FROM   emp;
```

```
ENAME             SAL       COMM (SAL*12)+NVL(COMM,0)
---------- --------- --------- --------------------
KING             5000                         60000
BLAKE            2850                         34200
CLARK            2450                         29400
JONES            2975                         35700
MARTIN           1250      1400               16400
ALLEN            1600       300               19500
...
14 rows selected.
```


Fonction NVL

Pour calculer la rémunération annuelle de chaque employé, il faut multiplier son salaire mensuel par 12 puis ajouter la commission au résultat obtenu.

```
SQL> SELECT ename, sal, comm, (sal*12)+comm
  2  FROM   emp;
```

```
ENAME      JOB       (SAL*12)+COMM
---------- --------- -------------
KING       PRESIDENT
BLAKE      MANAGER
CLARK      MANAGER
JONES      MANAGER
MARTIN     SALESMAN          16400
...
14 rows selected.
```

Notez que le résultat de ce calcul de rémunération annuelle est renseigné uniquement pour les employés qui perçoivent une commission. Lorsqu'un argument dans une expression arithmétique est NULL, le résultat de cette expression sera NULL. Pour calculer la rémunération annuelle de tous les employés, il faut convertir la valeur NULL en une valeur numérique avant d'appliquer l'opérateur arithmétique. Dans l'exemple ci-dessus, la fonction NVL est utilisée pour convertir les valeurs NULL en zéro.

M.Hanoune

Fonction DECODE

Facilite les recherches conditionnelles en jouant le rôle de CASE ou IF-THEN-ELSE

```
DECODE(col/expression, search1, result1
                       [, search2, result2,...,]
                       [, default])
```

3-36

La Fonction DECODE

La fonction DECODE permet de décoder les expressions de la même manière que l'ordre logique IF-THEN-ELSE utilisé dans de nombreux langages. Elle décode l'*expression* après l'avoir comparée à chacune des valeurs de recherche (*search)*. Si l'expression est identique à *search*, le résultat (*result)* est ramené.

Si la valeur par défaut (default) est omise, on obtient une valeur NULL chaque fois que la colonne ou expression ne correspond à aucune valeur *search*.

Utilisation de la Fonction DECODE

```
SQL> SELECT job, sal,
  2         DECODE(job, 'ANALYST', SAL*1.1,
  3                     'CLERK',   SAL*1.15,
  4                     'MANAGER', SAL*1.20,
  5                                SAL)
  6               REVISED_SALARY
  7  FROM   emp;
```

```
JOB              SAL REVISED_SALARY
--------- --------- --------------
PRESIDENT       5000           5000
MANAGER         2850           3420
MANAGER         2450           2940
...
14 rows selected.
```

3-37

Utilisation de la Fonction DECODE

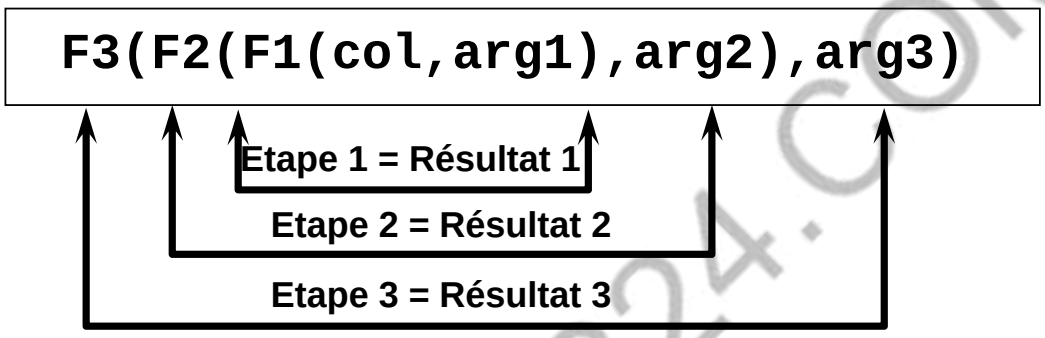
Dans l'ordre SQL ci-dessus, la valeur de JOB est décodée. Si la valeur de JOB correspond à ANALYST, alors l'augmentation de salaire est de 10% ; si elle correspond à CLERK, elle est de 15% et si elle correspond à MANAGER, elle est de 20%. Il n'y a aucune augmentation de salaire pour toutes les autres valeurs de JOB.

Cette instruction correspondrait à l'ordre IF-THEN-ELSE suivant :

```
IF job = 'ANALYST'    THEN  sal = sal*1.1
IF job = 'CLERK'      THEN  sal = sal*1.15
IF job = 'MANAGER'    THEN  sal = sal*1.20
ELSE sal = sal
```

Imbrication des Fonctions

- **Le niveau d'imbrication des fonctions mono-ligne est illimité**
- **Les fonctions imbriquées sont évaluées de l'intérieur vers l'extérieur**



3-38

Imbrication des Fonctions

Le niveau d'imbrication des fonctions mono-ligne est illimité. L'évaluation se fait du niveau le plus interne vers le niveau le plus externe. Les exemples qui suivent montrent la flexibilité de ces fonctions.

Imbrication des Fonctions

```
SQL> SELECT   ename,
  2           NVL(TO_CHAR(mgr),'No Manager')
  3  FROM     emp
  4  WHERE    mgr IS NULL;
```

```
ENAME      NVL(TO_CHAR(MGR),'NOMANAGER')
---------- ------------------------------
KING       No Manager
```

3-39

Imbrication des Fonctions

L'exemple ci-dessus affiche le nom du directeur de l'entreprise (qui n'a pas de manager). L'évaluation de l'ordre SQL se fait en deux temps :

1. Evaluation de la fonction interne qui convertit une valeur numérique en chaîne de caractères.
   - Result1 = TO_CHAR(mgr)
2. Evaluation de la fonction externe qui remplace la valeur NULL par une chaîne de texte.
   - NVL(Result1, 'No Manager')

L'expression entière devient l'en-tête de colonne puisqu'aucun alias de colonne n'a été donné.

Exemple

Afficher la date correspondant au premier vendredi qui tombe six mois après la date d'embauche. La date résultante doit se présenter comme suit : Friday, March 12th, 1982. Classer les résultats par date d'embauche.

```
SQL> SELECT    TO_CHAR(NEXT_DAY(ADD_MONTHS
  2            (hiredate, 6), 'FRIDAY'),
  3            'fmDay, Month ddth, YYYY')
  4            "Next 6 Month Review"
  5  FROM      emp
  6  ORDER BY  hiredate;
```

Résumé

Utilisez des fonctions mono-ligne pour :

- **Transformer des données**
- **Formater des dates et des nombres pour l'affichage**
- **Convertir des types de données de colonnes**

3-40

Fonctions Mono-Ligne

Les fonctions mono-ligne peuvent être imbriquées à l'infini. Elles peuvent manipuler :

- Des données caractères
  - LOWER, UPPER, INITCAP, CONCAT, SUBSTR, INSTR, LENGTH
- Des données numériques
  - ROUND, TRUNC, MOD
- Des dates
  - MONTHS_BETWEEN, ADD_MONTHS, NEXT_DAY, LAST_DAY, ROUND, TRUNC
  - Il est aussi possible d'utiliser les opérateurs arithmétiques + et - sur des données de type DATE.
- Les fonctions de conversion agissent sur les données de type caractère, date et numérique.
  - TO_CHAR, TO_DATE, TO_NUMBER

SYSDATE et DUAL

SYSDATE est une fonction date qui ramène la date et l'heure courantes. SYSDATE est généralement sélectionnée dans une table factice appelée DUAL.

M.Hanoune

Présentation des Exercices

- **Création de requêtes utilisant les fonctions numériques, caractère et date**
- **Utilisation de la concaténation avec les fonctions**
- **Ecriture de requêtes insensibles à la casse pour illustrer l'utilité des fonctions de type caractère**
- **Calcul des années et mois d'ancienneté d'un employé**
- **Détermination de la date de révision de salaire d'un employé**

3-41

Présentation des Exercices

La variété des exercices qui suivent est destinée à vous permettre de mettre en pratique les différentes fonctions utilisables avec des données de type caractère, numérique et date.

Rappelez-vous que dans les fonctions imbriquées, l'évaluation se fait de la fonction la plus interne vers la fonction la plus externe.

Exercices 3

1. Ecrivez une requête pour afficher la date courante. Nommez la colonne Date.

```
Date
---------
28-OCT-97
```

2. Pour chaque employé, affichez le matricule, le nom, le salaire et le salaire augmenté de 15% sous la forme d'un nombre entier. Nommez cette colonne New Salary. Enregistrez votre ordre SQL dans un fichier appelé *p3q2.sql*.

3. Exécutez votre requête à partir du fichier *p3q2.sql*.

```
EMPNO ENAME      SAL New Salary
----- ------- ----- ----------
 7839 KING     5000       5750
 7698 BLAKE    2850       3278
 7782 CLARK    2450       2818
 7566 JONES    2975       3421
 7654 MARTIN   1250       1438
 7499 ALLEN    1600       1840
 7844 TURNER   1500       1725
 7900 JAMES     950       1093
 7521 WARD     1250       1438
 7902 FORD     3000       3450
 7369 SMITH     800        920
 7788 SCOTT    3000       3450
 7876 ADAMS    1100       1265
 7934 MILLER   1300       1495
14 rows selected.
```

4. Modifiez votre requête *p3q2.sql* en ajoutant une colonne dans laquelle l'ancien salaire est soustrait du nouveau salaire. Nommez cette colonne Increase. Exécutez à nouveau votre requête.

```
EMPNO ENAME    SAL New Salary Increase
----- ------ ----- ---------- --------
 7839 KING    5000       5750      750
 7698 BLAKE   2850       3278      428
 7782 CLARK   2450       2818      368
 7566 JONES   2975       3421      446
```

Exercices 3

5. Affichez le nom et la date d'embauche de chaque employé ainsi que la date de révision du salaire qui sera le premier lundi tombant après 6 mois d'activité. Nommez la colonne REVIEW. Les dates devront apparaître dans le format suivant : "Sunday, the Seventh of September, 1981."

```
ENAME  HIREDATE  REVIEW
-----  --------  -------------------------------------
KING   17-NOV-81 Monday, the Twenty-Fourth of May, 1982
BLAKE  01-MAY-81 Monday, the Second of November, 1981
CLARK  09-JUN-81 Monday, the Fourteenth of December, 1981
JONES  02-APR-81 Monday, the Fifth of October, 1981
MARTIN 28-SEP-81 Monday, the Twenty-Ninth of March, 1982
ALLEN  20-FEB-81 Monday, the Twenty-Fourth of August, 1981
TURNER 08-SEP-81 Monday, the Fifteenth of March, 1982
JAMES  03-DEC-81 Monday, the Seventh of June, 1982
WARD   22-FEB-81 Monday, the Twenty-Fourth of August, 1981
FORD   03-DEC-81 Monday, the Seventh of June, 1982
SMITH  17-DEC-80 Monday, the Twenty-Second of June, 1981
SCOTT  09-DEC-82 Monday, the Thirteenth of June, 1983
ADAMS  12-JAN-83 Monday, the Eighteenth of July, 1983
MILLER 23-JAN-82 Monday, the Twenty-Sixth of July, 1982
14 rows selected.
```

6. Affichez le nom de chaque employé et calculez le nombre de mois travaillés depuis la date d'embauche. Nommez la colonne MONTHS_WORKED. Classez les résultats en fonction du nombre de mois d'ancienneté. Arrondissez le nombre de mois à l'entier le plus proche.

```
ENAME   MONTHS_WORKED
------- -------------
ADAMS               177
SCOTT               178
MILLER              188
JAMES               190
FORD                190
KING                191
MARTIN              192
TURNER              193
CLARK               196
BLAKE               197
JONES               198
WARD                199
ALLEN               199
SMITH               202
14 rows selected
```

Exercices 3

Si vous avez le temps, faites les exercices suivants :

7. Ecrivez une requête affichant les informations suivantes pour chaque employé : <nom de l'employé> gagne <salaire> par mois, mais veut <3 fois son salaire>. Nommez la colonne Salaires de Rêve.

```
Salaires de Rêve
----------------------------------------------------------
KING gagne $5,000.00 par mois, mais veut $15,000.00.
BLAKE gagne $2,850.00 par mois, mais veut $8,550.00.
CLARK gagne $2,450.00 par mois, mais veut $7,350.00.
JONES gagne $2,975.00 par mois, mais veut $8,925.00.
MARTIN gagne $1,250.00 par mois, mais veut $3,750.00.
ALLEN gagne $1,600.00 par mois, mais veut $4,800.00.
TURNER gagne $1,500.00 par mois, mais veut $4,500.00.
JAMES gagne $950.00 par mois, mais veut $2,850.00.
WARD gagne $1,250.00 par mois, mais veut $3,750.00.
FORD gagne $3,000.00 par mois, mais veut $9,000.00.
SMITH gagne $800.00 par mois, mais veut $2,400.00.
SCOTT gagne $3,000.00 par mois, mais veut $9,000.00.
ADAMS gagne $1,100.00 par mois, mais veut $3,300.00.
MILLER gagne $1,300.00 par mois, mais veut $3,900.00.

14 rows selected.
```

8. Créez une requête pour afficher le nom et le salaire de tous les employés. Le salaire sera formaté de façon à avoir 15 caractères de long, la valeur du salaire étant complétée à gauche par des $. Nommez la colonne SALARY.

```
ENAME      SALARY
--------   ---------------
SMITH      $$$$$$$$$$$800
ALLEN      $$$$$$$$$$1600
WARD       $$$$$$$$$$1250
JONES      $$$$$$$$$$2975
MARTIN     $$$$$$$$$$1250
BLAKE      $$$$$$$$$$2850
CLARK      $$$$$$$$$$2450
SCOTT      $$$$$$$$$$3000
KING       $$$$$$$$$$5000
TURNER     $$$$$$$$$$1500
ADAMS      $$$$$$$$$$1100
JAMES      $$$$$$$$$$950
FORD       $$$$$$$$$$3000
MILLER     $$$$$$$$$$1300

14 rows selected.
```

Exercices 3

9. Ecrivez une requête pour afficher tous les noms d'employé commençant par les lettres *J*, *A*, ou *M*, ainsi que la longueur du nom. Le nom doit apparaître en minuscules, sauf l'initiale qui sera en majuscules. Donnez à chaque colonne un nom approprié.

```
Name    Length
------- ------
Jones        5
Martin       6
Allen        5
James        5
Adams        5
Miller       6

6 rows selected.
```

10. Affichez le nom, la date d'embauche ainsi que le jour de la semaine où l'employé à débuté.
Nommez la colonne JOUR. Classez les résultats dans l'ordre des jours de la semaine à partir du lundi (monday).

```
ENAME   HIREDATE  JOUR
------- --------- ----------
MARTIN  28-SEP-81 MONDAY
CLARK   09-JUN-81 TUESDAY
KING    17-NOV-81 TUESDAY
TURNER  08-SEP-81 TUESDAY
SMITH   17-DEC-80 WEDNESDAY
ADAMS   12-JAN-83 WEDNESDAY
JONES   02-APR-81 THURSDAY
FORD    03-DEC-81 THURSDAY
SCOTT   09-DEC-82 THURSDAY
JAMES   03-DEC-81 THURSDAY
ALLEN   20-FEB-81 FRIDAY
BLAKE   01-MAY-81 FRIDAY
MILLER  23-JAN-82 SATURDAY
WARD    22-FEB-81 SUNDAY

14 rows selected
```

Exercice 3

Si vous souhaitez aller plus loin dans la difficulté, faites les exercices suivants :

11. Créez une requête pour afficher le nom et le montant de la commission de chaque employé. Pour les employés ne touchant aucune commission, affichez "No Commission". Nommez la colonne COMM.

```
ENAME   COMM
------  -------------------------
SMITH   No Commission
ALLEN   300
WARD    500
JONES   No Commission
MARTIN  1400
BLAKE   No Commission
CLARK   No Commission
SCOTT   No Commission
KING    No Commission
TURNER  0
ADAMS   No Commission
JAMES   No Commission
FORD    No Commission
MILLER  No Commission
14 rows selected.
```

12. Créez une requête pour afficher le nom des employés et leur salaire indiqué par des astérisques. Chaque astérisque représente cent dollars. Triez les données dans l'ordre décroissant des salaires. Nommez la colonne EMPLOYEE_AND_THEIR_SALARIES.

```
EMPLOYEE_AND_THEIR_SALARIES
----------------------------------------------------------
KING     ***********************************************
FORD     ****************************
SCOTT    ****************************
JONES    ****************************
BLAKE    ************************
CLARK    ***********************
ALLEN    ***************
TURNER   **************
MILLER   ************
MARTIN   ***********
WARD     ***********
ADAMS    **********
JAMES    ********
SMITH    *******

14 rows selected.
```

4

Afficher des Données Issues de Plusieurs Tables

Objectifs

A la fin de ce chapitre, vous saurez :

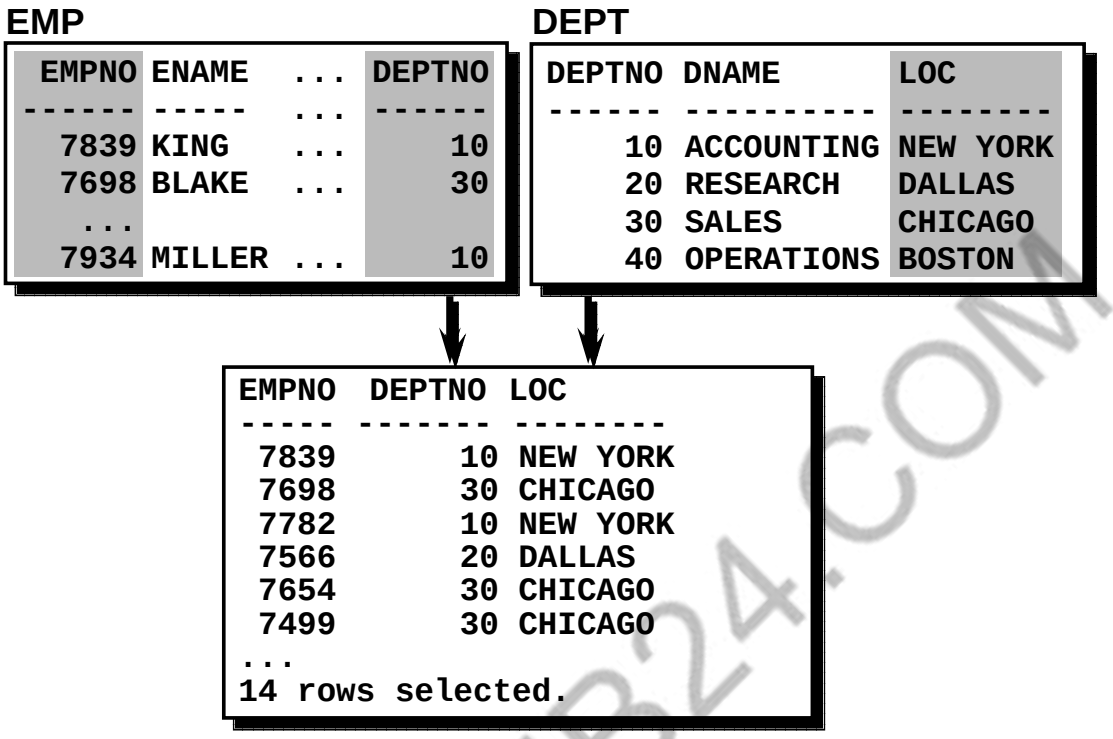
- **Ecrire des ordres SELECT pour accéder aux données de plusieurs tables en utilisant des équijointures et des non-équijointures**
- **Visualiser des données ne répondant pas aux conditions de jointure, en utilisant les jointures externes**
- **Relier une table à elle-même**

4-2

Objectifs

Au cours de ce chapitre, vous allez étudier les différentes façons d'obtenir des données de plusieurs tables.

Afficher des Données Issues de Plusieurs Tables



4-3

Afficher des Données Issues de Plusieurs Tables

On a parfois besoin d'obtenir des données de plusieurs tables. Dans l'exemple ci-dessus, l'état affiche les données de deux tables différentes.

- EMPNO appartient à la table EMP.
- DEPTNO appartient aux tables EMP et DEPT.
- LOC appartient à la table DEPT.

Pour obtenir cet état, il faut relier les tables EMP et DEPT et accéder aux données de ces deux tables.

M.Hanoune

Qu'est-ce qu'une Jointure ?

Une jointure sert à extraire des données de plusieurs tables.

```
SELECT   table1.column, table2.column
FROM     table1, table2
WHERE    table1.column1 = table2.column2;
```

- **Ecrivez la condition de jointure dans la clause WHERE.**
- **Placez le nom de la table avant le nom de la colonne lorsque celui-ci figure dans plusieurs tables.**

4-4

Définition des Jointures

Pour obtenir des données appartenant à différentes tables de la base de données, vous devez utiliser une condition de *jointure*. Les lignes d'une table peuvent être reliées aux lignes d'une autre table en fonction de valeurs communes éxistant dans des colonnes se correspondant, en général la colonne clé primaire et la colonne clé étrangère.

Pour afficher les données issues de deux ou plusieurs tables , écrivez une condition de jointure simple dans la clause WHERE.
Syntaxe :

table.column	indique la table et la colonne d'où sont extraites les données
table1.column1 = table2.column2	représente la condition qui joint (ou lie) les tables entre-elles

Conseils

- Lorque vous écrivez un ordre SELECT pour joindre des tables, il est recommandé, par souci de clarté et de facilité d'accès, de placer le nom de la table avant le nom de la colonne.
- Lorsque le même nom de colonne apparaît dans plusieurs tables, il doit obligatoirement être préfixé par le nom de la table.
- Pour joindre *n* tables entre elles, il faut au minimum (*n-1*) conditions de jointure. C'est pourquoi, par exemple, trois jointures au moins sont nécessaires pour lier quatre tables. Cette règle ne s'applique pas si votre table contient une clé primaire concaténée, auquel cas il faut plus d'une colonne pour permettre d'identifier chaque ligne de manière unique.

Pour plus d'information, reportez-vous à
Oracle8 Server SQL Language Reference Manual, "SELECT."

M.Hanoune

Produit Cartésien

- **On obtient un produit cartésien lorsque :**
  - **Une condition de jointure est omise**
  - **Une condition de jointure est incorrecte**
- **Toutes les lignes de la première table sont jointes à toutes les lignes de la seconde**
- **Pour éviter un produit cartésien, toujours insérer une condition de jointure correcte dans la clause WHERE.**

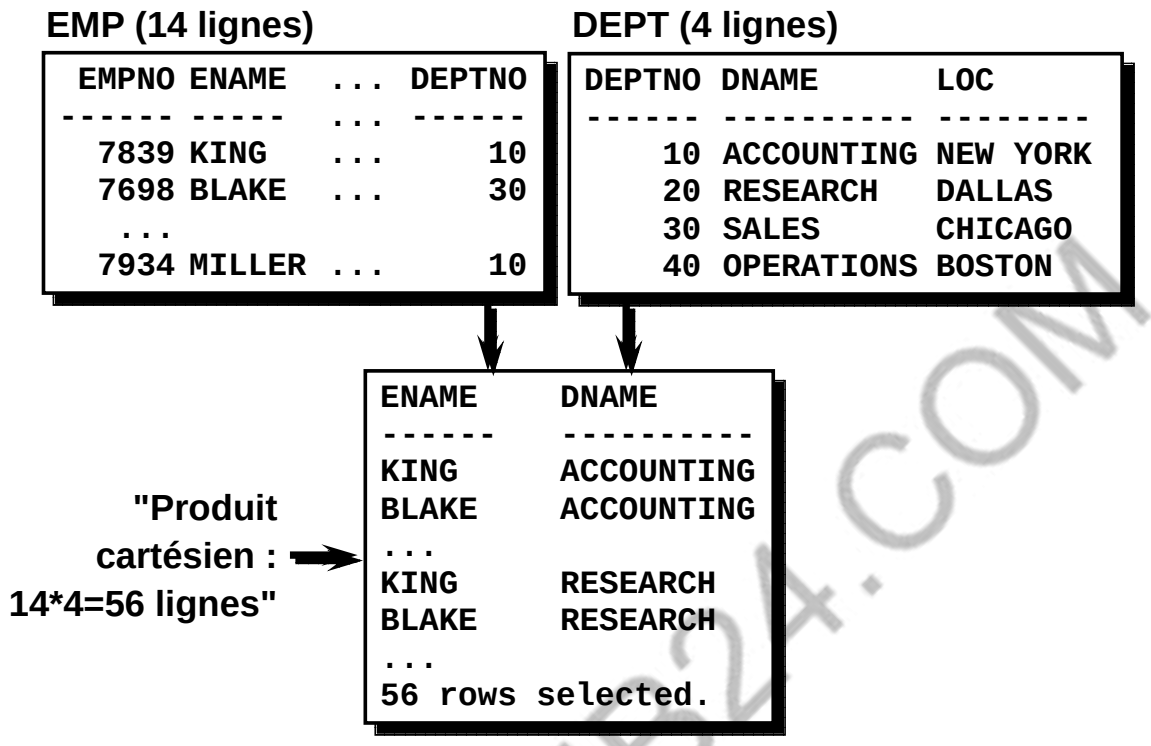
4-5

Produit Cartésien

Lorsqu'une condition de jointure est incorrecte ou tout simplement omise, on obtient un *produit cartésien* dans lequel sont affichées toutes les combinaisons de lignes. Toutes les lignes de la première table sont jointes à toutes les lignes de la seconde.

Un produit cartésien fournit en général un nombre important de lignes, donnant un résultat rarement exploitable. C'est pourquoi il faut toujours inclure une condition de jointure correcte dans une clause WHERE, à moins que vous n'ayez réellement besoin de combiner toutes les lignes de toutes les tables.

Génération d'un Produit Cartésien



4-6

Produit Cartésien

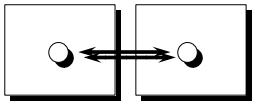
L'omission d'une condition de jointure génère un produit cartésien. L'exemple de la diapositive affiche le nom d'employé et le nom de département des tables EMP et DEPT. Comme aucune clause WHERE n'a été spécifiée, toutes les lignes (14) de la table EMP ont été jointes à l'ensemble des lignes (4) de la table DEPT, donnant ainsi un résultat de 56 lignes.

```
SQL> SELECT ename, dname
  2  FROM   emp, dept;
```

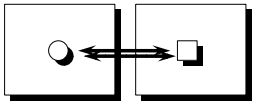
```
ENAME      DNAME
---------- --------------
KING       ACCOUNTING
BLAKE      ACCOUNTING
...
KING       RESEARCH
BLAKE      RESEARCH
...
56 rows selected.
```

Types de Jointures

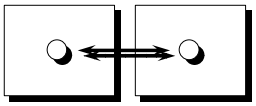
Equijointure



Non-équijointure



Jointure externe



Autojointure

4-7

Types de Jointures

Il existe deux principaux types de conditions de jointure :

- Les équijointures
- Les non-équijointures

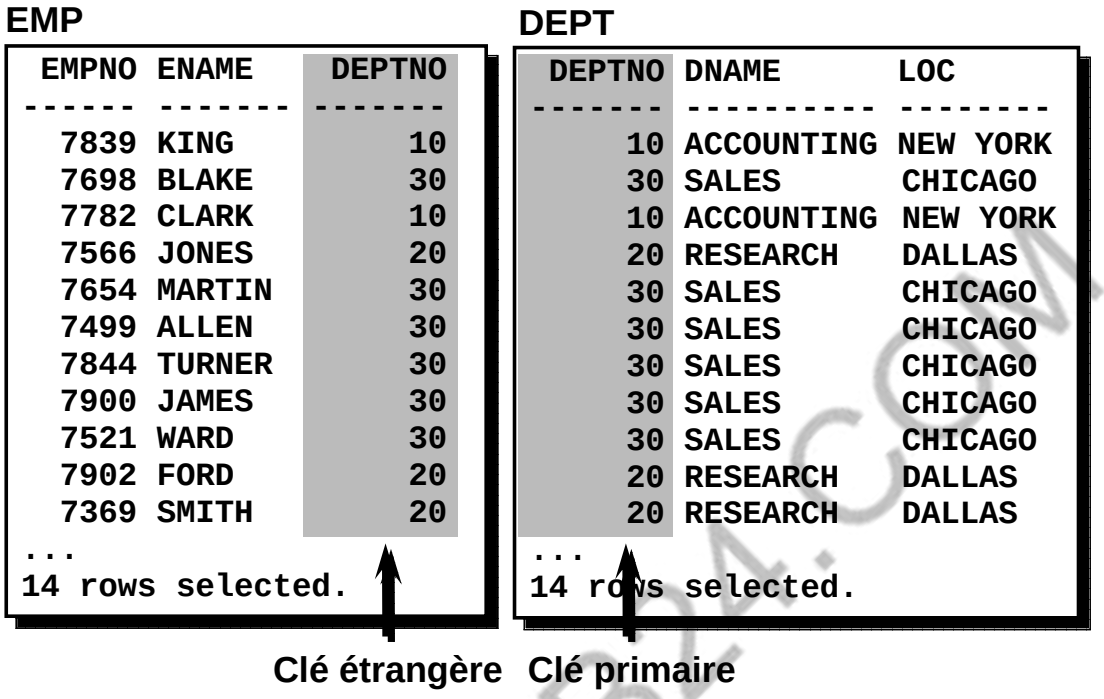
Les autres méthodes de jointures sont les suivantes :

- Jointures externes
- Autojointures
- Les opérateurs ensemblistes

Remarque : les opérateurs ensemblistes seront étudiés dans un chapitre ultérieur

M.Hanoune

Qu'est-ce qu'une Equijointure ?



4-8

Equijointures

Pour déterminer le département auquel appartient un employé, vous devez comparer les valeurs de la colonne DEPTNO de la table EMP avec les valeurs de la colonne DEPTNO de la table DEPT. La relation établie entre les tables EMP et DEPT est une équijointure : les valeurs de la colonne DEPTNO appartenant aux deux tables doivent être identiques. Ce type de relation fait souvent appel aux clés primaires et étrangères.

Remarque : les équijointures sont aussi appelées jointures simples ou jointures internes.

Extraction d'Enregistrements avec les Equijointures

```
SQL> SELECT   emp.empno, emp.ename, emp.deptno,
  2           dept.deptno, dept.loc
  3  FROM     emp, dept
  4  WHERE    emp.deptno=dept.deptno;
```

```
EMPNO ENAME   DEPTNO DEPTNO LOC
----- ------ ------ ------ ---------
 7839 KING       10     10 NEW YORK
 7698 BLAKE      30     30 CHICAGO
 7782 CLARK      10     10 NEW YORK
 7566 JONES      20     20 DALLAS
...
14 rows selected.
```

4-9

Extraction d'Enregistrements avec les Equijointures

Dans l'exemple ci-dessus :

- La clause SELECT spécifie les noms des colonnes à extraire :
  - colonnes ename (nom des employés), empno (matricule des employés) et deptno (numéro de département) dans la table EMP
  - colonnes deptno (numéro de département) et loc (localisation) dans la table DEPT
- La clause FROM spécifie les deux tables de la base de données auxquelles on souhaite accéder :
  - la table EMP
  - la table DEPT
- La clause WHERE spécifie la façon dont les deux tables sont jointes :

  EMP.DEPTNO=DEPT.DEPTNO

La colonne DEPTNO étant commune aux deux tables, vous devez la préfixer du nom de la table d'appartenance afin d'éviter toute ambiguïté.

Différencier les Noms de Colonne Ambigus

- **Préfixer avec le nom de la table pour différencier les noms de colonnes appartenant à plusieurs tables.**
- **Ces préfixes de table améliorent les performances.**
- **Différencier des colonnes de même nom appartenant à plusieurs tables en utilisant des alias de colonne.**

4-10

Différenciation des Noms de Colonne

Pour éviter toute ambiguïté, vous devez préfixer dans la clause WHERE les noms de colonne avec le nom de la table. Ainsi, sans autre précision, la colonne DEPTNO peut tout aussi bien appartenir à la table DEPT qu'à la table EMP. Il faut donc ajouter le préfixe de table pour pouvoir exécuter la requête.

Lorsqu'aucune colonne n'est commune aux deux tables, la qualification n'est pas indispensable. Toutefois, vous obtiendrez de meilleurs résultats avec les préfixes de table, car ils indiquent précisément à Oracle où il peut trouver les colonnes.

La nécessité de qualifier les noms de colonne s'applique aussi quand une colonne ambiguë est présente dans d'autres clauses, par exemple dans les clauses SELECT ou ORDER BY.

Ajout de Conditions de Recherche avec l'Opérateur AND

EMP

EMPNO	ENAME	DEPTNO
7839	KING	10
7698	BLAKE	30
7782	CLARK	10
7566	JONES	20
7654	MARTIN	30
7499	ALLEN	30
7844	TURNER	30
7900	JAMES	30
7521	WARD	30
7902	FORD	20
7369	SMITH	20

...
14 rows selected.

DEPT

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
30	SALES	CHICAGO
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
30	SALES	CHICAGO
30	SALES	CHICAGO
30	SALES	CHICAGO
30	SALES	CHICAGO
20	RESEARCH	DALLAS
20	RESEARCH	DALLAS

...
14 rows selected.

4-11

Conditions de Recherche Supplémentaires

Outre la jointure, vous pouvez spécifier des critères supplémentaires dans la clause WHERE. Par exemple, pour afficher le matricule, le nom, le numéro de département et la localisation de l'employé King, vous devez ajouter une condition dans la clause WHERE.

```
SQL> SELECT empno, ename, emp.deptno, loc
  2  FROM   emp, dept
  3  WHERE  emp.deptno = dept.deptno
  4  AND    INITCAP(ename) = 'King';
```

```
    EMPNO ENAME          DEPTNO LOC
--------- ---------- --------- -------------
     7839 KING               10 NEW YORK
```

M.Hanoune

Utilisation d'Alias de Table

Simplifiez les requêtes avec les alias de table.

```
SQL> SELECT emp.empno, emp.ename, emp.deptno,
  2         dept.deptno, dept.loc
  3  FROM   emp, dept
  4  WHERE  emp.deptno=dept.deptno;
```

```
SQL> SELECT e.empno, e.ename, e.deptno,
  2         d.deptno, d.loc
  3  FROM   emp e, dept d
  4  WHERE  e.deptno=d.deptno;
```

4-12

Alias de Table

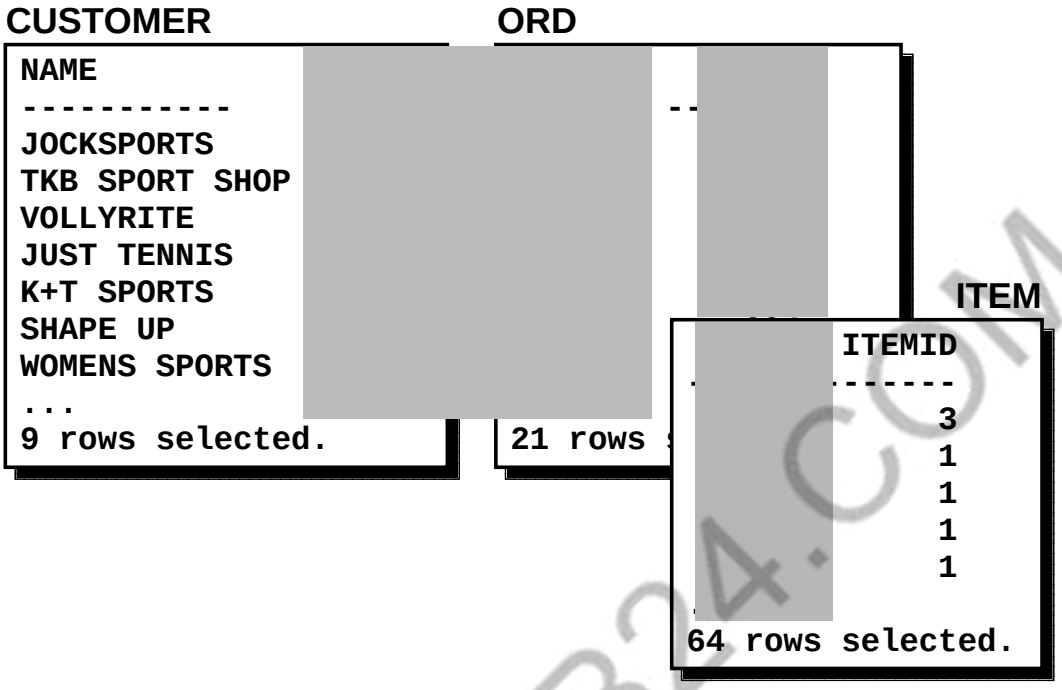
La qualification des noms de colonne à l'aide des noms de table peut prendre beaucoup de temps, en particulier si les noms de table sont longs. Vous pouvez substituer des *alias* de table aux noms de table. De la même manière qu'un alias de colonne renomme une colonne, un alias de table donne un nouveau nom à une table. Les alias de table permettent ainsi de réduire le volume du code SQL et donc, de gagner de la place en mémoire.

Notez la manière dont les alias de table sont identifiés dans la clause FROM de l'exemple. Le nom de la table spécifié en entier est suivi d'un espace puis de l'alias de table. E est l'alias de la table EMP, et D l'alias de la table DEPT.

Conseils

- Bien qu'un alias de table puisse compter jusqu'à 30 caractères, il est préférable qu'il soit le plus court possible.
- Lorsqu'un alias de table est substitué à un nom de table dans la clause FROM, cette substitution doit s'opérer dans la totalité de l'ordre SELECT.
- Choisissez de préférence des alias "parlants".
- Un alias de table ne s'applique que dans l'ordre SELECT courant.

Jointures de Plus de Deux Tables



4-13

Jointures Mettant en Jeu Plus de Deux Tables

Vous pouvez avoir besoin de lier plus de deux tables. Par exemple, si vous souhaitez afficher le nom, les commandes passées, les numéros d'articles, le total par article et le total par commande pour le client TKB SPORT SHOP, vous devez lier les tables CUSTOMER, ORD et ITEM.

```
SQL> SELECT c.name, o.ordid, i.itemid, i.itemtot, o.total
  2  FROM   customer c, ord o, item i
  3  WHERE  c.custid = o.custid
  4  AND    o.ordid = i.ordid
  5  AND    c.name = 'TKB SPORT SHOP';
```

```
NAME            ORDID    ITEMID   ITEMTOT     TOTAL
----------- --------- --------- --------- ---------
TKB SPORT SHOP     610         3        58     101.4
TKB SPORT SHOP     610         1        35     101.4
TKB SPORT SHOP     610         2       8.4     101.4
```

Non-Equijointures

EMP

EMPNO	ENAME	SAL
7839	KING	5000
7698	BLAKE	2850
7782	CLARK	2450
7566	JONES	2975
7654	MARTIN	1250
7499	ALLEN	1600
7844	TURNER	1500
7900	JAMES	950

...
14 rows selected.

SALGRADE

GRADE	LOSAL	HISAL
1	700	1200
2	1201	1400
3	1401	2000
4	2001	3000
5	3001	9999

"Les salaires (SAL) de la table EMP sont compris entre le salaire minimum (LOSAL) et le salaire maximum (HISAL) de la table SALGRADE"

4-14

Non-Equijointures

La relation entre la table EMP et la table SALGRADE est une non-équijointure car aucune colonne de la table EMP ne correspond directement à une colonne de la table SALGRADE. La relation existant entre les deux tables est la suivante: les valeurs de la colonne SAL de la table EMP sont comprises entre celles des colonnes LOSAL et HISAL de la table SALGRADE. Il faut donc utiliser un autre opérateur que le signe égal (=) pour effectuer une jointure.

Extraction d'Enregistrements avec les Non-Equijointures

```
SQL>  SELECT   e.ename, e.sal, s.grade
  2   FROM     emp e, salgrade s
  3   WHERE    e.sal
  4   BETWEEN  s.losal AND s.hisal;
```

```
ENAME             SAL      GRADE
---------- --------- ---------
JAMES             950          1
SMITH             800          1
ADAMS            1100          1
...
14 rows selected.
```

4-15

Non-Equijointures (suite)

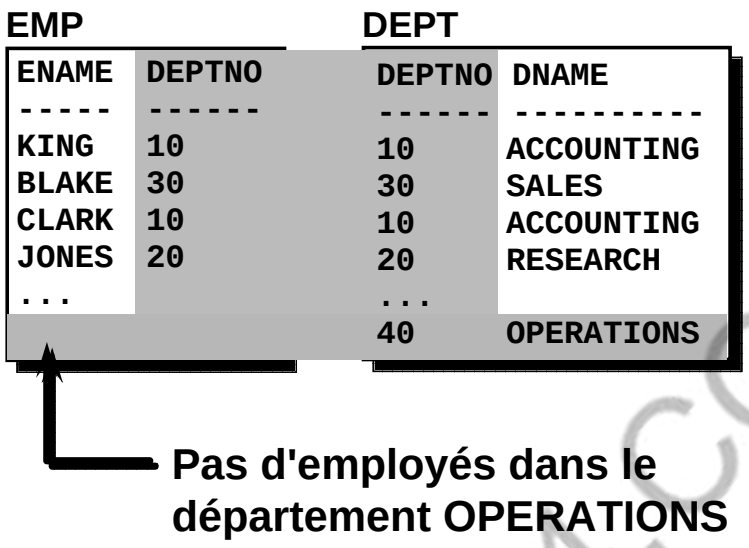
L'exemple ci-dessus crée une non-équijointure pour évaluer l'échelon de salaire d'un employé. Le salaire est obligatoirement compris *entre* deux valeurs délimitant une tranche salariale.

Il est important de noter que tous les employés n'apparaissent qu'une seule fois dans la liste lorsque la requête est exécutée, et ce pour deux raisons :

- Aucune ligne de la table des échelons de salaire ne déborde sur une autre. Autrement dit, le salaire d'un employé se situe nécessairement entre la valeur minimale et la valeur maximale d'une des lignes de la table.
- Tous les salaires des employés entrent dans les limites prévues par la table des échelons de salaire. Aucun employé ne peut gagner moins que le salaire minimal de la colonne LOSAL ni plus que le salaire maximal de la colonne HISAL.

Remarque : il serait possible d'utiliser d'autres opérateurs tels que <= et >= , mais BETWEEN est le plus simple. Avec BETWEEN, n'oubliez pas de spécifier d'abord la valeur la plus basse puis la valeur la plus haute. Dans l'exemple, des alias de table ont été spécifiés pour améliorer les performances et non à cause d'une possible ambiguïté.

Jointures Externes



4-16

Affichage d'Enregistrements sans Lien Direct, au moyen de Jointures Externes

Lorsqu'une ligne ne satisfait pas à une condition de jointure, elle n'apparaît pas dans le résultat de la requête. Par exemple, quand on fait l'équijointure entre les tables EMP et DEPT, le département OPERATIONS n'apparaît pas car personne ne travaille dans ce département.

```
SQL> SELECT e.ename, e.deptno, d.dname
  2  FROM   emp e, dept d
  3  WHERE  e.deptno = d.deptno;
```

```
ENAME          DEPTNO DNAME
---------- --------- -------------
KING               10 ACCOUNTING
BLAKE              30 SALES
CLARK              10 ACCOUNTING
JONES              20 RESEARCH
...
ALLEN              30 SALES
TURNER             30 SALES
JAMES              30 SALES
...
14 rows selected.
```

Jointures Externes

- **Les jointures externes permettent de visualiser des lignes qui ne répondent pas à la condition de jointure.**
- **L'opérateur de jointure externe est le signe (+).**

```
SELECT table.column, table.column
FROM   table1, table2
WHERE  table1.column(+) = table2.column;
```

```
SELECT table.column, table.column
FROM   table1, table2
WHERE  table1.column = table2.column(+);
```

4-17

Affichage d'Enregistrements sans Lien Direct, au moyen de Jointures Externes

Il est néanmoins possible de ramener la ou les lignes manquantes en plaçant un opérateur de *jointure externe* dans la condition de jointure. Cet opérateur se présente sous la forme d'un signe plus inclus entre parenthèses, et *se place du "côté" de la jointure où l'information est incomplète.* Il crée une ou plusieurs lignes NULL, auxquelles une ou plusieurs lignes de la table complète peuvent être liées.

Syntaxe :

table1.column =	condition qui joint (ou lie) les tables entre-elles.
table2.column (+)	symbole de jointure externe ; se place d'un côté ou de l'autre de la condition de la clause WHERE, jamais des deux côtés. Placez le symbole de jointure externe après le nom de la colonne appartenant à la table où manquent les lignes correspondantes

Utilisation des Jointures Externes

```
SQL> SELECT   e.ename, d.deptno, d.dname
  2  FROM     emp e, dept d
  3  WHERE    e.deptno(+) = d.deptno
  4  ORDER BY e.deptno;
```

```
ENAME          DEPTNO DNAME
---------- --------- -------------
KING               10 ACCOUNTING
CLARK              10 ACCOUNTING
...
                   40 OPERATIONS
15 rows selected.
```

4-18

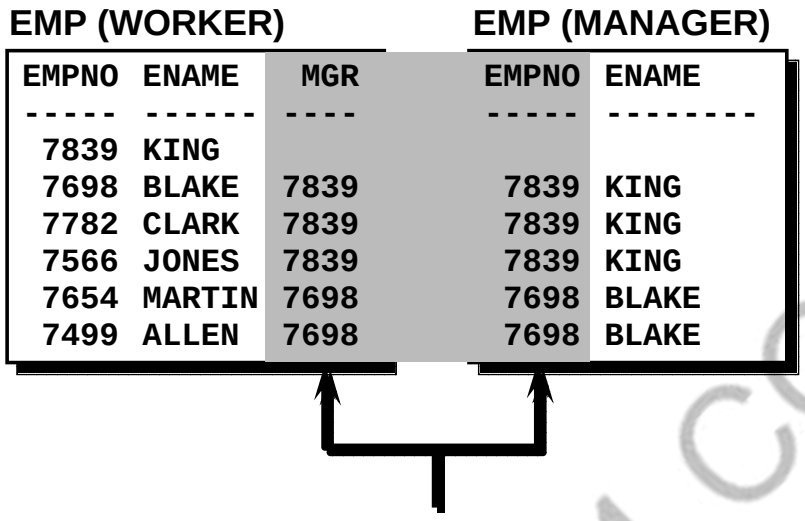
Affichage d'Enregistrements sans Lien Direct, au moyen de Jointures Externes

L'exemple ci-dessus affiche les numéros et les noms de tous les départements. Le département OPERATIONS, qui ne comprend aucun employé, est également affiché.

Restrictions Applicables aux Jointures Externes

- L'opérateur de jointure externe ne peut être placé que d'*un* seul côté de l'expression, à savoir le côté où l'information manque. Il permet de ramener les lignes d'une table n'ayant pas de jointure directe avec l'autre table.
- Une condition comportant une jointure externe ne peut pas utiliser l'opérateur IN ni être liée à une autre condition par l'opérateur OR.

Autojointures



"Dans la table WORKER, MGR équivaut à EMPNO dans la table MANAGER"

4-19

Liaison d'une Table à Elle-même

Il se peut que vous ayez besoin de relier une table à elle-même. Ici, pour retrouver le nom du manager de chaque employé, il faut que la table EMP soit reliée à elle-même. Par exemple, pour retrouver le nom du manager de l'employé Blake, vous devez :

- Trouver Blake dans la table EMP en cherchant dans la colonne ENAME
- Trouver le matricule du manager de Blake en cherchant dans la colonne MGR.
- Trouver le nom du manager dont le matricule est 7839 dans la colonne EMPNO, puis regarder le nom correspondant dans la colonne ENAME. Le matricule 7839 appartient à King, donc King est le manager de Blake.

Dans ce processus, vous utilisez la même table deux fois : la première, pour rechercher le nom de Blake dans la colonne ENAME et la valeur correspondante dans la colonne MRG ; la seconde, pour rechercher le matricule 7839 dans la colonne EMPNO et le nom correspondant (King) dans la colonne ENAME.

Liaison d'une Table à Elle-même

```
SQL> SELECT worker.ename||' works for '||manager.ename
  2  FROM   emp worker, emp manager
  3  WHERE  worker.mgr = manager.empno;

WORKER.ENAME||'WORKSFOR'||MANAG
------------------------------
BLAKE works for KING
CLARK works for KING
JONES works for KING
MARTIN works for BLAKE
...
13 rows selected.
```

4-20

Liaison d'une Table à Elle-même (suite)

L'exemple ci-dessus relie la table EMP à elle-même. Afin de simuler l'existence de deux tables dans la clause FROM, on a mis deux alias, WORKER et MANAGER, pour la même table EMP.

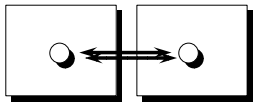
Dans cet exemple, la clause WHERE contient une jointure dont la signification est "lorsque le matricule du manager d'un employé correspond au matricule du manager".

M.Hanoune

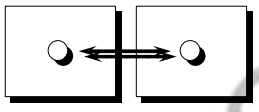
Résumé

```
SELECT   table.column, table.column
FROM     table1, table2
WHERE    table1.column1 = table2.column2;
```

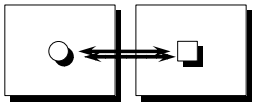
Equijointure



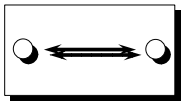
Jointure externe



Non-équijointure



Autojointure



4-21

Résumé

Il existe de nombreuses manières de lier des tables. Cependant, toutes sont fondées sur une condition spécifiée dans la clause WHERE. La méthode que vous choisirez dépend du résultat souhaité et des structures de données que vous utilisez.

```
SELECT      table.column, table.column
FROM        table1, table2
WHERE       table1.column1 = table2.column2;
```

Types de Jointures

- Equijointure
- Non-équijointure
- Jointure externe
- Autojointure

Produits Cartésiens

L'omission de la condition de jointure dans la clause WHERE génère un produit cartésien dans lequel toutes les combinaisons de lignes sont affichées.

Alias de Table

- Les alias de table accélèrent l'accès aux données.
- Les alias de table permettent de réduire le volume du code et donc, de gagner de la place en mémoire.

Présentation des Exercices

- **Liaison de tables au moyen d'équijointures**
- **Exécution de jointures externes et d'autojointures**
- **Ajout de conditions supplémentaires**

Présentation des Exercices

Les exercices qui suivent ont pour but de vous entraîner à extraire des données provenant de plusieurs tables. Ils vont vous demander de lier et de sélectionner des lignes dans la clause WHERE.

Exercices 4

1. Ecrivez une requête pour afficher le nom, le numéro de département et le département de tous les employés.

```
ENAME  DEPTNO DNAME
------ ------ ---------
CLARK      10 ACCOUNTING
KING       10 ACCOUNTING
MILLER     10 ACCOUNTING
SMITH      20 RESEARCH
ADAMS      20 RESEARCH
FORD       20 RESEARCH
SCOTT      20 RESEARCH
JONES      20 RESEARCH
ALLEN      30 SALES
BLAKE      30 SALES
MARTIN     30 SALES
JAMES      30 SALES
TURNER     30 SALES
WARD       30 SALES
14 rows selected.
```

2. Créez une liste unique de tous les postes du département 30.

```
JOB       LOC
--------- -------------
CLERK     CHICAGO
MANAGER   CHICAGO
SALESMAN  CHICAGO
```

3. Ecrivez une requête pour afficher le nom, le nom du département et la localisation de tous les employés qui touchent une commission.

```
ENAME  DNAME  LOC
------ ------ -------------
ALLEN  SALES  CHICAGO
WARD   SALES  CHICAGO
MARTIN SALES  CHICAGO
TURNER SALES  CHICAGO
```

Exercices 4

4. Affichez le nom et le nom du département pour tous les employés dont le nom contient la lettre A. Enregistrez votre ordre SQL dans un fichier que vous nommerez *p4q4.sql*.

```
ENAME    DNAME
-------  ------------
CLARK    ACCOUNTING
ADAMS    RESEARCH
ALLEN    SALES
WARD     SALES
JAMES    SALES
MARTIN   SALES
BLAKE    SALES
7 rows selected.
```

5. Ecrivez une requête pour afficher le nom, le poste, le numéro de département et le nom du département de tous les employés basés à DALLAS.

```
ENAME   JOB       DEPTNO DNAME
------- --------- ------ ----------
SMITH   CLERK         20 RESEARCH
ADAMS   CLERK         20 RESEARCH
FORD    ANALYST       20 RESEARCH
SCOTT   ANALYST       20 RESEARCH
JONES   MANAGER       20 RESEARCH
```

6. Affichez le nom et le matricule des employés et de leur manager. Nommez les colonnes Employee, Emp#, Manager, et Mgr#, respectivement.
Enregistrez votre ordre SQL dans un fichier nommé *p4q6.sql*.

```
Employee    Emp# Manager Mgr#
--------  ------ ------- -----
SCOTT       7788 JONES    7566
FORD        7902 JONES    7566
ALLEN       7499 BLAKE    7698
WARD        7521 BLAKE    7698
JAMES       7900 BLAKE    7698
TURNER      7844 BLAKE    7698
MARTIN      7654 BLAKE    7698
MILLER      7934 CLARK    7782
ADAMS       7876 SCOTT    7788
JONES       7566 KING     7839
CLARK       7782 KING     7839
BLAKE       7698 KING     7839
SMITH       7369 FORD     7902
13 rows selected.
```

Exercices 4

7. Modifiez le fichier *p4q6.sql* pour afficher tous les employés, y compris King, n'ayant pas de manager. Enregistrez à nouveau dans un fichier *p4q7.sql*.Exécutez *p4q7.sql*.

```
Employee   Emp# Manager  Mgr#
-------- ------ ------- -----
SCOTT      7788 JONES    7566
FORD       7902 JONES    7566
ALLEN      7499 BLAKE    7698
WARD       7521 BLAKE    7698
JAMES      7900 BLAKE    7698
TURNER     7844 BLAKE    7698
MARTIN     7654 BLAKE    7698
MILLER     7934 CLARK    7782
ADAMS      7876 SCOTT    7788
JONES      7566 KING     7839
CLARK      7782 KING     7839
BLAKE      7698 KING     7839
SMITH      7369 FORD     7902
KING       7839
14 rows selected.
```

Si vous avez le temps, faites les exercices suivants :

8. Créez une requête pour afficher le numéro de département et le nom de tous les employés qui travaillent dans le même département qu'un certain employé. Donnez à chaque colonne un en-tête approprié.

```
    DEPARTMENT EMPLOYEE COLLEAGUE
    ---------- -------- ---------
            10 CLARK    KING
            10 CLARK    MILLER
            10 KING     CLARK
            10 KING     MILLER
            10 MILLER   CLARK
            10 MILLER   KING
            20 ADAMS    FORD
            20 ADAMS    JONES
            20 ADAMS    SCOTT
            20 ADAMS    SMITH
            20 FORD     ADAMS
            20 FORD     JONES
            20 FORD     SCOTT
    ...
56 rows selected.
```

Exercices 4

9. Affichez la structure de la table SALGRADE. Créez une requête pour afficher le nom, le poste, le département, le salaire et l'échelon de tous les employés.

```
Name         NULL?    Type
------------ -------- ----
GRADE                 NUMBER
LOSAL                 NUMBER
HISAL                 NUMBER
```

```
ENAME   JOB        DNAME         SAL GRADE
------  ---------- ---------- ----- -----
MILLER  CLERK      ACCOUNTING  1300     2
CLARK   MANAGER    ACCOUNTING  2450     4
KING    PRESIDENT  ACCOUNTING  5000     5
SMITH   CLERK      RESEARCH     800     1
SCOTT   ANALYST    RESEARCH    3000     4
FORD    ANALYST    RESEARCH    3000     4
ADAMS   CLERK      RESEARCH    1100     1
JONES   MANAGER    RESEARCH    2975     4
JAMES   CLERK      SALES        950     1
BLAKE   MANAGER    SALES       2850     4
TURNER  SALESMAN   SALES       1500     3
ALLEN   SALESMAN   SALES       1600     3
WARD    SALESMAN   SALES       1250     2
MARTIN  SALESMAN   SALES       1250     2
14 rows selected.
```

Si vous souhaitez aller plus loin dans la difficulté, faites les exercices suivants :

10. Créez une requête pour afficher le nom et la date d'embauche de tous les employés arrivés avant l'employé Blake. Trier les lignes sur la date d'embauche.

```
ENAME   HIREDATE
------- ---------
SMITH   17-DEC-80
ALLEN   20-FEB-81
WARD    22-FEB-81
JONES   02-APR-81
```

Exercices 4

11. Affichez les noms et date d'embauche des employés et de leur manager, pour tous les employés ayant été embauchés avant leur manager. Nommez les colonnes Employee, Emp Hiredate, Manager et Mgr Hiredate, respectivement.

```
Employee Emp Hiredate  Manager Mgr Hiredate
-------- ------------- ------- ------------
ALLEN    20-FEB-81     BLAKE   01-MAY-81
WARD     22-FEB-81     BLAKE   01-MAY-81
JONES    02-APR-81     KING    17-NOV-81
CLARK    09-JUN-81     KING    17-NOV-81
BLAKE    01-MAY-81     KING    17-NOV-81
SMITH    17-DEC-80     FORD    03-DEC-81
6 rows selected.
```

5

Regrouper les Données avec les Fonctions de Groupe

Objectifs

A la fin de ce chapitre, vous saurez :

- **Identifier les fonctions de groupe disponibles**
- **Expliquer l'utilisation des fonctions de groupe**
- **Regrouper les données avec la clause GROUP BY**
- **Inclure ou exclure des groupes de lignes avec la clause HAVING**

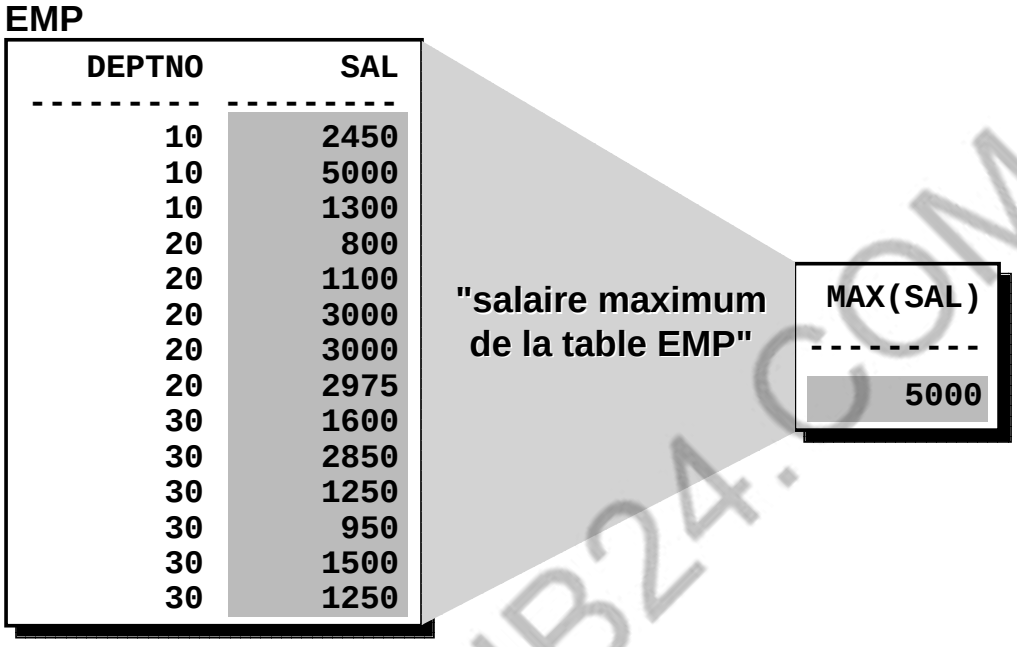
5-2

Objectifs

Ce chapitre poursuit l'étude des fonctions. Il explique comment effectuer des calculs statistiques, tels que les moyennes, sur des groupes de lignes. Il présente les différentes manières de grouper les lignes d'une table en sous-ensembles et comment spécifier des critères de recherche sur ces groupes de lignes.

Fonctions de Groupe

Les fonctions de groupe agissent sur des groupes de lignes et donnent un résultat par groupe.



5-3

Fonctions de Groupe

Contrairement aux fonctions mono-ligne, les fonctions de groupe agissent sur des groupes de lignes et donnent un résultat par groupe. Un groupe peut être une table entière ou un ensemble de lignes d'une table.

Hanoune

Types de Fonctions de Groupe

- **AVG ([DISTINCT|ALL]*n*)**
- **COUNT ({ *|[DISTINCT|ALL]*expr*})**
- **MAX ([DISTINCT|ALL]*expr*)**
- **MIN ([DISTINCT|ALL]*expr*)**
- **STDDEV ([DISTINCT|ALL]*n*)**
- **SUM ([DISTINCT|ALL]*n*)**
- **VARIANCE ([DISTINCT|ALL]*n*)**

5-4

Fonctions de Groupe

Chaque fonction accepte un argument. La table suivante présente les différentes options de syntaxe possibles.

Fonction	**Description**
AVG([DISTINCT|ALL]*n*)	Valeur moyenne de *n*, en ignorant les valeurs NULL
COUNT({*|[DISTINCT|ALL]expr})	Nombre de lignes, où expr est différent de NULL. Le caractère * comptabilise toutes les lignes sélectionnées y compris les doublons et les lignes NULL
MAX([DISTINCT|ALL]*expr*)	Valeur maximale de *expr*, en ignorant les valeurs NULL
MIN([DISTINCT|ALL]*expr*)	Valeur minimale de *expr*, en ignorant les valeurs NULL
STDDEV([DISTINCT|ALL]*x*)	Ecart standard de *n*, en ignorant les valeurs NULL
SUM([DISTINCT|ALL]*n*)	Somme des valeurs de *n*, en ignorant les valeurs NULL
VARIANCE([DISTINCT|ALL]*x*)	Variance de *n*, en ignorant les valeurs NULL

Types de Fonctions de Groupe

- **AVG ([DISTINCT|ALL]*n*)**
- **COUNT ({ *|[DISTINCT|ALL]*expr*})**
- **MAX ([DISTINCT|ALL]*expr*)**
- **MIN ([DISTINCT|ALL]*expr*)**
- **STDDEV ([DISTINCT|ALL]*x*)**
- **SUM ([DISTINCT|ALL]*n*)**
- **VARIANCE ([DISTINCT|ALL]*x*)**

5-5

Conseils pour l'Utilisation des Fonctions de Groupe

- Avec DISTINCT, la fonction ne prend en compte que les valeurs distinctes ; avec ALL, elle tient compte de toutes les valeurs y compris les doublons. La valeur par défaut est ALL, par conséquent il n'est pas nécessaire de la spécifier.
- Si *expr* est spécifié, les différents types de données possibles pour les arguments sont CHAR, VARCHAR2, NUMBER ou DATE.
- Toutes les fonctions de groupe, à l'exception de COUNT(*), ignorent les valeurs NULL. Pour substituer une valeur à une valeur NULL, utilisez la fonction NVL.

Fonctions AVG et SUM

AVG et SUM s'utilisent avec des données numériques.

```
SQL> SELECT   AVG(sal), MAX(sal),
  2           MIN(sal), SUM(sal)
  3  FROM     emp
  4  WHERE    job LIKE 'SALES%';
```

```
AVG(SAL)  MAX(SAL)  MIN(SAL)  SUM(SAL)
--------  --------  --------  --------
    1400      1600      1250      5600
```

5-6

Fonctions de Groupe

Vous pouvez utiliser les fonctions AVG, SUM, MIN et MAX avec des colonnes de données numériques. L'exemple ci-dessus affiche la moyenne, le maximum, le minimum et la somme des salaires mensuels pour tous les vendeurs.

Fonctions MIN et MAX

MIN et MAX s'utilisent avec tous types de données.

```
SQL> SELECT  MIN(hiredate), MAX(hiredate)
  2  FROM    emp;
```

```
MIN(HIRED MAX(HIRED
--------- ---------
17-DEC-80 12-JAN-83
```


Fonctions de Groupe (suite)

Vous pouvez utiliser les fonctions MAX et MIN pour tous les types de données. L'exemple ci-dessus affiche la date d'embauche de l'employé le plus récemment entré et celle du plus ancien.

L'exemple ci-dessous affiche le nom des employés apparaissant en premier et en dernier dans le classement alphabétique de tous les employés.

```
SQL> SELECT    MIN(ename), MAX(ename)
  2  FROM      emp;
```

```
MIN(ENAME) MAX(ENAME)
---------- ----------
ADAMS      WARD
```

Remarque : les fonctions AVG, SUM, VARIANCE et STDDEV n'acceptent que des données de type numérique.

Utilisation de la Fonction COUNT

COUNT(*) ramène le nombre de lignes d'une table.

```
SQL> SELECT  COUNT(*)
  2  FROM    emp
  3  WHERE   deptno = 30;
```

```
 COUNT(*)
---------
         6
```

5-8

Fonction COUNT

La fonction COUNT se présente sous deux formats :

- COUNT(*)
- COUNT(*expr*).

COUNT(*) ramène le nombre de lignes d'une table, y compris les lignes en double et celles contenant des valeurs NULL.

A l'opposé, COUNT(*expr*) ramène le nombre de lignes pour lesquelles la colonne identifiée par *expr* est non NULL.

L'exemple ci-dessus affiche le nombre d'employés du département 30.

Utilisation de la Fonction COUNT

COUNT(*expr*) ramène le nombre de lignes non NULL.

```
SQL> SELECT   COUNT(comm)
  2  FROM     emp
  3  WHERE    deptno = 30;
```

```
COUNT(COMM)
-----------
          4
```

5-9

Fonction COUNT (suite)

L'exemple ci-dessus affiche le nombre d'employés du département 30 qui touchent une commission. Notez que le nombre total de lignes affiché est 4, car deux employés du département 30 ne sont pas habilités à toucher de commission et présentent par conséquent une valeur NULL dans la colonne COMM.

Exemple

Afficher le nombre de départements de la table EMP.

```
SQL> SELECT    COUNT(deptno)
  2  FROM      emp;
```

```
COUNT(DEPTNO)
-------------
           14
```

Afficher le nombre de départements distincts de la table EMP.

```
SQL> SELECT    COUNT(DISTINCT (deptno))
  2  FROM      emp;
```

```
COUNT(DISTINCT(DEPTNO))
-----------------------
                      3
```

Fonctions de Groupe et Valeurs NULL

Les fonctions de groupe ignorent les valeurs NULL des colonnes.

```
SQL> SELECT AVG(comm)
  2  FROM   emp;
```

```
 AVG(COMM)
---------
      550
```

5-10

Fonctions de Groupe et Valeurs Null

Toutes les fonctions de groupe, à l'exception de COUNT (*), ignorent les valeurs NULL des colonnes. Dans l'exemple ci-dessus, la moyenne est calculée uniquement sur les lignes pour lesquelles la colonne COMM est renseignée. Le calcul de la moyenne s'effectue en divisant le total des commissions versées à tous les employés par le nombre d'employés touchant une commission (4).

Utilisation de la Fonction NVL avec les Fonctions de Groupe

La fonction NVL force la prise en compte des valeurs NULL dans les fonctions de groupe.

```
SQL> SELECT AVG(NVL(comm,0))
  2  FROM   emp;
```

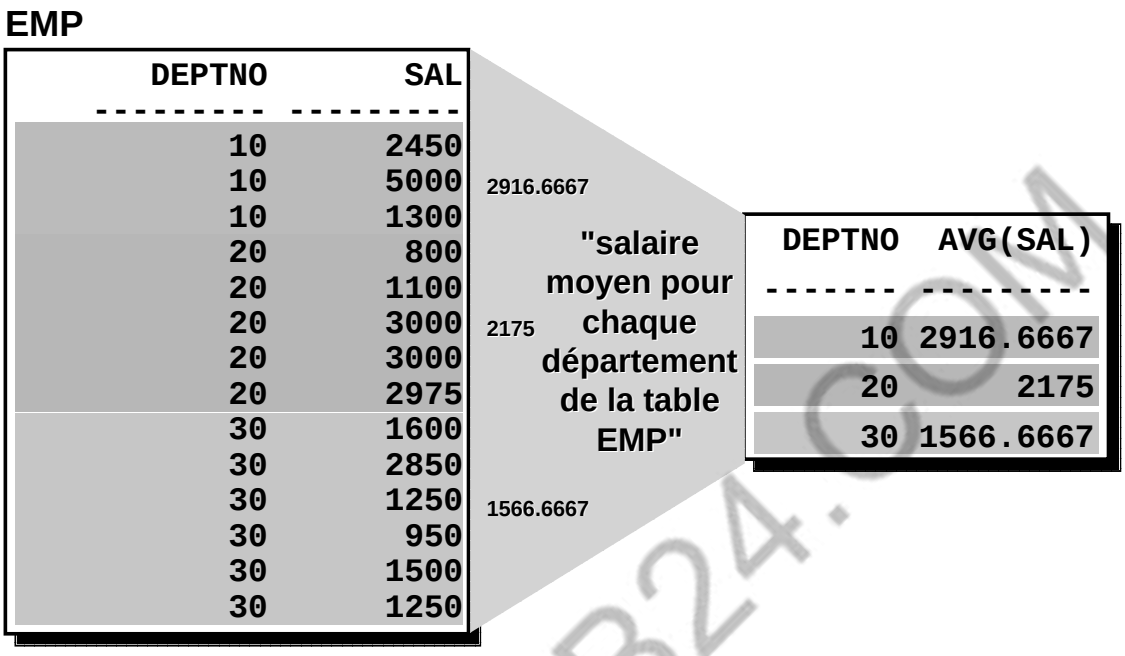
```
AVG(NVL(COMM,0))
----------------
      157.14286
```

5-11

Fonctions de Groupe et Valeurs Null (suite)

La fonction NVL force les fonctions de groupe à prendre en compte les valeurs NULL. Dans l'exemple ci-dessus, la moyenne est calculée sur *toutes* les lignes de la table y compris celles dont la colonne COMM contient des valeurs NULL. Le calcul de la moyenne s'effectue en divisant le total des commissions de tous les employés par le nombre total d'employés de l'entreprise (14).

Création de Groupes de Données



5-12

Groupes de Données

Jusqu'ici, la table a été prise en compte par les fonctions de groupe, comme un seul groupe d'informations. Mais il est parfois nécessaire de diviser les informations d'une table en groupes plus petits. Pour cela, il faut utiliser la clause GROUP BY.

Création de Groupes de Données : la Clause GROUP BY

```
SELECT      column, group_function
FROM        table
[WHERE      condition]
[GROUP BY   group_by_expression]
[ORDER BY   column];
```

Divisez une table en groupes de lignes avec la clause GROUP BY.

5-13

La Clause GROUP BY

Vous pouvez utiliser la clause GROUP BY pour diviser une table en groupes de lignes. Vous pouvez alors utiliser les fonctions de groupe pour effectuer des calculs statistiques sur chaque groupe.

Syntaxe :

group_by_expression spécifie les colonnes dont les valeurs déterminent les différents groupes

Conseils

- Lorsque vous intégrez une fonction de groupe dans une clause SELECT, vous ne pouvez pas en même temps sélectionner des résultats individuels, *à moins que* la colonne individuelle ne soit mentionnée dans la clause GROUP BY. Si vous omettez de spécifier un de ces colonnes, un message d'erreur s'affichera.
- Avec la clause WHERE, vous pouvez exclure des lignes avant de créer des groupes.
- Vous devez inclure les *"column"* de la liste SELECT dans la clause GROUP BY.
- Vous ne pouvez pas utiliser l'alias de colonne dans la clause GROUP BY.
- Par défaut, les lignes sont triées dans l'ordre croissant des colonnes incluses dans la liste GROUP BY. Vous pouvez changer cet ordre en utilisant la clause ORDER BY.

Utilisation de la Clause GROUP BY

La clause GROUP BY doit inclure toutes les colonnes de la liste SELECT qui ne figurent pas dans des fonctions de groupe.

```
SQL> SELECT   deptno, AVG(sal)
  2  FROM     emp
  3  GROUP BY deptno;
```

```
   DEPTNO  AVG(SAL)
--------- ---------
       10 2916.6667
       20      2175
       30 1566.6667
```

5-14

La Clause GROUP BY (suite)

Lorsque vous utilisez la clause GROUP BY, pensez à inclure dans cette clause toutes les colonnes de la liste SELECT qui ne figurent pas dans les fonctions de groupe. L'exemple ci-dessus affiche le numéro et le salaire moyen de chaque département. Voici comment est évalué l'ordre SELECT ci-dessus, qui contient une clause GROUP BY :

- La clause SELECT indique les colonnes à extraire :
  - Colonne numéro de département de la table EMP
  - La moyenne de tous les salaires du groupe que vous avez spécifié dans la clause GROUP BY
- La clause FROM indique les tables de la base de données auxquelles il faut accéder : la table EMP.
- La clause WHERE indique les lignes à extraire. Comme il n'y a pas de clause WHERE, toutes les lignes seront extraites.
- La clause GROUP BY spécifie la manière dont les lignes doivent être groupées. Ici, elle sont groupées par numéro de département pour permettre à la fonction AVG, appliquée à la colonne des salaires (sal), de calculer la *moyenne des salaires de chaque département*.

Utilisation de la Clause GROUP BY

La colonne citée en GROUP BY ne doit pas nécessairement figurer dans la liste SELECT.

```
SQL> SELECT   AVG(sal)
  2  FROM     emp
  3  GROUP BY deptno;
```

```
 AVG(SAL)
---------
2916.6667
     2175
1566.6667
```

5-15

La Clause GROUP BY (suite)

Il n'est pas obligatoire que la colonne de la clause GROUP BY soit également dans la clause SELECT. Par exemple, l'ordre SELECT ci-dessus affiche la moyenne des salaires de chaque département sans afficher les numéros de département correspondants. Toutefois, sans les numéros de département, les résultats ne sont pas très parlants.

Vous pouvez utiliser une fonction de groupe dans la clause ORDER BY.

```
SQL> SELECT   deptno, AVG(sal)
  2  FROM     emp
  3  GROUP BY  deptno
  4  ORDER BY   AVG(sal);
```

```
    DEPTNO     AVG(SAL)
---------- ------------
        30    1566.6667
        20         2175
        10    2916.6667
```

Regroupement sur Plusieurs Colonnes

EMP

DEPTNO	JOB	SAL
10	MANAGER	2450
10	PRESIDENT	5000
10	CLERK	1300
20	CLERK	800
20	CLERK	1100
20	ANALYST	3000
20	ANALYST	3000
20	MANAGER	2975
30	SALESMAN	1600
30	MANAGER	2850
30	SALESMAN	1250
30	CLERK	950
30	SALESMAN	1500
30	SALESMAN	1250

'"somme des salaires de la table EMP pour chaque poste, regroupés par département"

DEPTNO	JOB	SUM(SAL)
10	CLERK	1300
10	MANAGER	2450
10	PRESIDENT	5000
20	ANALYST	6000
20	CLERK	1900
20	MANAGER	2975
30	CLERK	950
30	MANAGER	2850
30	SALESMAN	5600

5-16

Sous-groupes

Il est parfois nécessaire d'obtenir des résultats pour des sous-groupes de lignes. Sur la diapositive ci-dessus, l'état affiche le salaire total pour chacun des postes à l'intérieur de chaque département.

Ici, les lignes de la table EMP présentent un groupement par numéro de département, puis des sous-groupes qui sont fonction des poste. Par exemple, les deux employés 'CLERK' du département 20 ont été groupés; un résultat unique (salaire total) est produit pour tous les vendeurs (salesmen) du département 30.

Utilisation de la Clause GROUP BY sur Plusieurs Colonnes

```
SQL> SELECT   deptno, job, sum(sal)
  2  FROM     emp
  3  GROUP BY deptno, job;
```

```
   DEPTNO JOB         SUM(SAL)
--------- --------- ---------
       10 CLERK          1300
       10 MANAGER        2450
       10 PRESIDENT      5000
       20 ANALYST        6000
       20 CLERK          1900
...
9 rows selected.
```

5-17

Sous-groupes (suite)

Vous pouvez effectuer des calculs statistiques sur des groupes et des sous-groupes en spécifiant plusieurs colonnes dans la clause GROUP BY. Par défaut, le tri des lignes retournées est conditionné par l'ordre des colonnes dans la clause GROUP BY.

Voici comment est évalué l'ordre SELECT ci-dessus, qui contient une clause GROUP BY :

- La clause SELECT indique les colonnes à extraire :
  - Colonne numéro de département de la table EMP
  - Colonne intitulé des postes dans la table EMP
  - La somme de tous les salaires du groupe que vous avez spécifié dans la clause GROUP BY
- La clause FROM indique les tables de la base de données auxquelles il faut accéder : la table EMP.
- La clause GROUP BY indique comment grouper les lignes :
  - Tout d'abord, les lignes sont groupées par numéro de département.
  - Ensuite, à l'intérieur de ces groupes, les lignes sont groupées par intitulé de poste.

La fonction SUM est alors appliquée à la colonne des salaires de tous les postes à l'intérieur de chaque département.

Hanoune

Erreurs d'Utilisation des Fonctions de Groupe dans une Requête

Toute colonne ou expression de la liste SELECT autre qu'une fonction de groupe, doit être incluse dans la clause GROUP BY.

```
SQL> SELECT   deptno, COUNT(ename)
  2  FROM     emp;
```

```
SELECT deptno, COUNT(ename)
       *
ERROR at line 1:
ORA-00937: not a single-group group function
```

Colonne manquante dans la clause GROUP BY

5-18

Erreurs d'utilisation des Fonctions de Groupe dans une Requête

Chaque fois que vous mélangez des éléments individuels (DEPTNO) et des fonctions de groupe (COUNT) dans le même ordre SELECT, vous devez obligatoirement inclure une clause GROUP BY qui spécifie les éléments individuels (ici, DEPTNO). Si la clause GROUP BY est absente, vous verrez apparaître le message d'erreur "not a single-group group function", la colonne incorrecte étant indiquée par un astérisque (*). L'erreur ci-dessus peut être corrigée en ajoutant la clause GROUP BY.

```
SQL> SELECT    deptno,COUNT(ename)
  2  FROM      emp
  3  GROUP BY  deptno;
```

```
    DEPTNO COUNT(ENAME)
---------- ------------
        10            3
        20            5
        30            6
```

Toute colonne ou expression de la liste SELECT autre qu'une fonction de groupe doit figurer dans la clause GROUP BY.

Hanoune

Erreurs d'utilisation des Fonctions de Groupe dans une Requête

- **Vous ne pouvez utiliser la clause WHERE pour limiter les groupes.**
- **Utilisez la clause HAVING.**

```
SQL> SELECT   deptno, AVG(sal)
  2  FROM     emp
  3  WHERE    AVG(sal) > 2000
  4  GROUP BY deptno;
```

```
WHERE AVG(sal) > 2000
      *
ERROR at line 3:
ORA-00934: group function is not allowed here
```

N'utilisez pas la clause WHERE pour limiter les groupes

5-19

Erreurs d'utilisation des Fonctions de Groupe dans une Requête (suite)

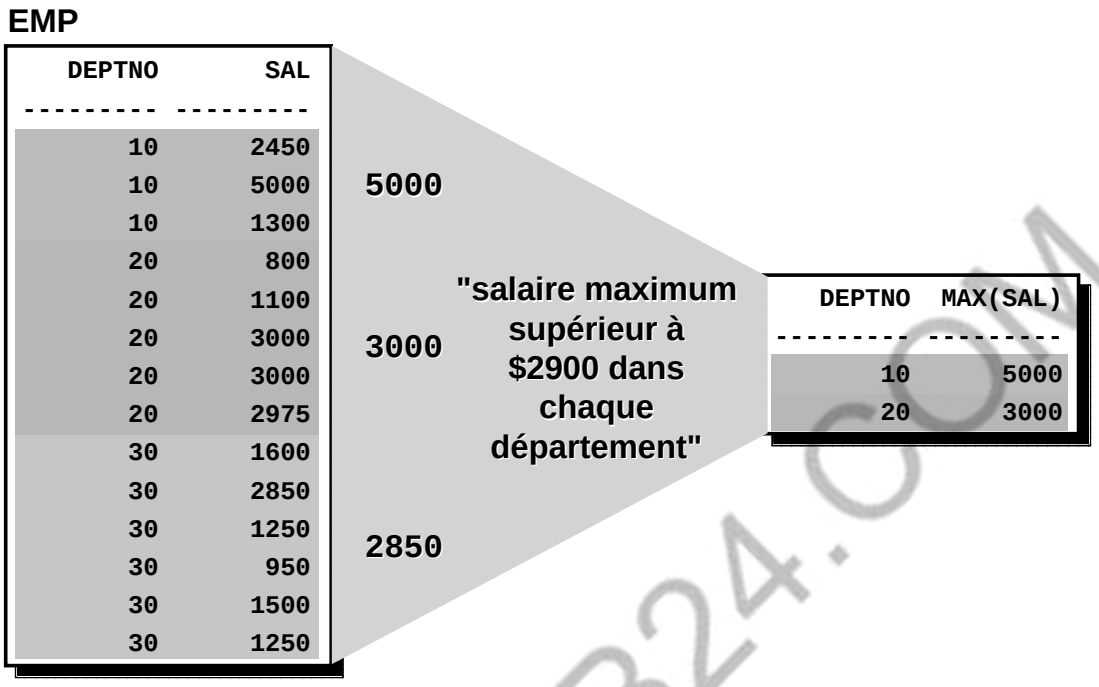
La clause WHERE ne peut pas être utilisée pour limiter les groupes retournés. L'ordre SELECT ci-dessus provoque une erreur car il utilise la clause WHERE pour restreindre l'affichage aux départements dans lesquels le salaire moyen est supérieur à $2000.

Vous pouvez corriger l'erreur ci-dessus en limitant les groupes avec la clause HAVING.

```
SQL> SELECT   deptno, AVG(sal)
  2  FROM     emp
  3  GROUP BY deptno
  4  HAVING   AVG(sal) > 2000;
```

```
    DEPTNO      AVG(SAL)
---------- --------------
        10     2916.6667
        20          2175
```

Exclusion de Groupes



5-20

Exclusion de Groupes

De la même manière que vous utilisez la clause WHERE pour limiter les lignes que vous sélectionnez, vous pouvez utiliser la clause HAVING pour restreindre des groupes. Pour afficher le salaire maximum de chaque département, en ne retenant que les départements ayant un salaire maximum supérieur à 2000 $, vous devez procéder comme suit :

1. Trouvez le salaire moyen pour chaque département en créant des groupes par numéro de département.
2. Restreindre les groupes retournés aux départements qui ont un salaire maximum supérieur à $2900.

Hanoune

Exclusion de Groupes : la Clause HAVING

Utilisez la clause HAVING pour restreindre les groupes

- Les lignes sont regroupées.
- La fonction de groupe est appliquée.
- Les groupes qui correspondent à la clause HAVING sont affichés.

```
SELECT      column, group_function
FROM        table
[WHERE      condition]
[GROUP BY   group_by_expression]
[HAVING     group_condition]
[ORDER BY   column];
```

5-21

La Clause HAVING

Utilisez la clause HAVING pour indiquer les groupes que vous souhaitez afficher.

Syntaxe :

group_condition limite les groupes de lignes ramenés aux groupes pour lesquels la condition spécifiée est VRAIE

Oracle Server opère de la manière suivante avec la clause HAVING :

- Les lignes sont groupées.
- La fonction de groupe s'applique au groupe.
- Les groupes qui correspondent aux critères de la clause HAVING sont affichés.

Il est possible de placer la clause HAVING avant la clause GROUP BY ; néanmoins, pour des raisons de logique, il est recommandé de placer la clause GROUP BY en tête. Les groupes sont formés puis les fonctions de groupe calculées avant que la clause HAVING ne soit appliquée aux groupes de la liste SELECT.

Utilisation de la clause HAVING

```
SQL> SELECT   deptno, max(sal)
  2  FROM     emp
  3  GROUP BY deptno
  4  HAVING   max(sal)>2900;
```

```
    DEPTNO  MAX(SAL)
---------- ---------
        10      5000
        20      3000
```

5-22

La Clause HAVING (suite)

L'exemple ci-dessus affiche le salaire maximum et le numéro de département dont le salaire maximum est supérieur à $2900.

Vous pouvez utiliser la clause GROUP BY sans utiliser de fonction de groupe dans la liste SELECT.

Si vous souhaitez, pour restreindre les lignes retournées, vous baser sur le résultat d'une fonction de groupe, vous devez utiliser une clause HAVING plus une clause GROUP BY.

L'exemple suivant affiche le numéro de département et le salaire moyen pour les départements dont le salaire maximum est supérieur à $2900.

```
SQL> SELECT    deptno, AVG(sal)
  2  FROM emp
  3  GROUP BY  deptno
  4 HAVING     MAX(sal) > 2900;
```

```
DEPTNO  AVG(SAL)
--------- ---------
       10 2916.6667
       20      2175
```

Utilisation de la Clause HAVING

```
SQL> SELECT    job, SUM(sal) PAYROLL
  2  FROM      emp
  3  WHERE      job NOT LIKE 'SALES%'
  3  GROUP BY  job
  4  HAVING    SUM(sal)>5000
  5  ORDER BY  SUM(sal);
```

```
JOB          PAYROLL
--------- ---------
ANALYST         6000
MANAGER         8275
```

5-23

La Clause HAVING (suite)

L'exemple ci-dessus affiche l'intitulé de poste et le total des salaires mensuels pour chaque poste. On ne retiendra que les postes ayant un salaire total > $5000. On excluera les employés dont le poste est 'SALESMAN'. Le résultat sera trié par salaire total croissant.

Imbrication des Fonctions de Groupe

Afficher le salaire moyen maximum.

```
SQL> SELECT   max(avg(sal))
  2  FROM     emp
  3  GROUP BY deptno;
```

```
MAX(AVG(SAL))
-------------
    2916.6667
```

5-24

Imbrication des Fonctions de Groupe

Les fonctions de groupe peuvent être imbriquées. L'exemple ci-dessus affiche le salaire moyen maximum.

Résumé

```
SELECT      column, group_function
FROM        table
[WHERE      condition]
[GROUP BY   group_by_expression]
[HAVING     group_condition]
[ORDER BY   column];
```

5-25

Résumé

SQL comprend sept fonctions de groupe :

- AVG
- COUNT
- MAX
- MIN
- SUM
- STDDEV
- VARIANCE

Vous pouvez créer des sous-groupes en utilisant la clause GROUP BY. La clause HAVING permet d'exclure des groupes.

Dans un ordre, les clauses HAVING et GROUP BY doivent être placées après la clause WHERE. Placez la clause ORDER BY en dernier.

Oracle Server évalue les clauses dans l'ordre suivant :

- Si l'ordre contient une clause WHERE, le serveur établit les lignes candidates.
- Le serveur constitue les groupes spécifiés dans la clause GROUP BY.
- La clause HAVING permet de restreindre encore davantage les résultats en excluant les groupes qui ne répondent pas aux critères de la clause HAVING.

Présentation des Exercices

- **Montrer différentes requêtes utilisant les fonctions de groupe**
- **Grouper des lignes pour obtenir plusieurs résultats**
- **Exclure des groupes au moyen de la clause HAVING**

5-26

Présentation des Exercices

A la fin des exercices, vous saurez utiliser sans problème les fonctions de groupe et sélectionner des groupes de données.

Questions sur Papier

Pour les questions 1 à 3, entourez Vrai ou Faux.

Remarque : des alias de colonne sont utilisés pour les requêtes.

Exercices 5

Déterminez si les affirmations suivantes sont vraies ou fausses et entourez la réponse correspondante.

1. Les fonctions de groupe agissent sur plusieurs lignes pour produire un seul résultat.
   Vrai/Faux
2. Les fonctions de groupe intègrent les valeurs NULL dans leurs calculs.
   Vrai/Faux
3. La clause WHERE restreint les lignes avant qu'elles soient incluses dans un calcul de groupe.
   Vrai/Faux

4. Affichez le salaire maximum, le salaire minimum, la somme des salaires et le salaire moyen de tous les employés. Nommez respectivement les colonnes Maximum, Minimum, Sum et Average. Arrondissez les résultats à zéro décimale. Enregistrez votre ordre SQL dans un fichier nommé *p5q4.sql*.

```
Maximum  Minimum    Sum Average
-------  ------- ------ -------
   5000      800  29025    2073
```

5. Modifiez le fichier *p5q4.sql* pour afficher le salaire maximum, le salaire minimum, la somme des salaires et le salaire moyen pour chaque type de poste.
   Enregistrez votre fichier sous *p5q5.sql*. Exécutez à nouveau votre requête.

```
JOB         Maximum  Minimum     Sum  Average
---------- -------- -------- ------- --------
ANALYST        3000     3000    6000     3000
CLERK          1300      800    4150     1038
MANAGER        2975     2450    8275     2758
PRESIDENT      5000     5000    5000     5000
SALESMAN       1600     1250    5600     1400
```

6. Ecrivez une requête pour afficher le nombre de personnes qui occupent le même poste.

```
JOB       COUNT(*)
---------- --------
ANALYST           2
CLERK             4
MANAGER           3
PRESIDENT         1
SALESMAN          4
```

Exercices 5

Si vous avez le temps, faites les exercices suivants :

7. Déterminez le nombre de personnes ayant des subordonnés, sans en donner la liste. Nommez la colonne "Nombre de Chefs".

```
Nombre de Chefs
---------------
              6
```

8. Ecrivez une requête pour afficher la différence existant entre le salaire maximum et le salaire minimum. Nommez la colonne DIFFERENCE.

```
DIFFERENCE
----------
      4200
```

9. Affichez le matricule des différents managers et le niveau de salaire le plus bas de leurs employés.
Excluez toute ligne où le manager n'est pas identifié. Excluez tout groupe dans lequel le salaire minimum est inférieur à $1000. Triez les résultats par ordre décroissant des salaires.

```
       MGR  MIN(SAL)
---------  --------
      7566      3000
      7839      2450
      7782      1300
      7788      1100
```

10. Ecrivez une requête pour afficher le nom du département, la localisation, le nombre d'employés et le salaire moyen pour tous les employés de ce département. Nommez les colonnes dname, loc, Nombre d'Employés et Salaire, respectivement.

```
DNAME          LOC       Nombre d'Employés  Salaire
-------------- --------- -----------------  -------
ACCOUNTING     NEW YORK                  3  2916.67
RESEARCH       DALLAS                    5     2175
SALES          CHICAGO                   6  1566.67
```

Exercices 5

Si vous souhaitez aller plus loin dans la difficulté, faites les exercices suivants :

11. Créez une requête pour afficher le nombre total d'employés puis, parmi ces employés, ceux qui ont été embauchés en 1980, 1981, 1982 et 1983. Nommez les colonnes de façon appropriée.

```
TOTAL  1980   1981   1982  1983
----- -----  -----  ----- -----
   14     1     10      2     1
```

12. Créez une requête pour afficher les postes, le salaire de ces postes par numéro de département et le salaire total de ces postes incluant tous les départements. Nommez les colonnes de façon appropriée.

```
Job        Dept 10  Dept 20  Dept 30   Total
--------- -------- -------- -------- -------
ANALYST                6000             6000
CLERK          1300     1900      950    4150
MANAGER        2450     2975     2850    8275
PRESIDENT      5000                       5000
SALESMAN                          5600    5600
```

6

Opérateurs Ensemblistes

Objectifs

A la fin de ce chapitre, vous saurez :

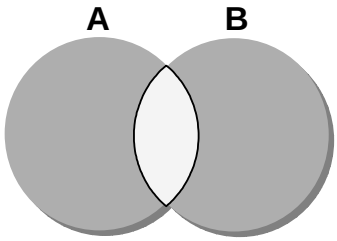
- **Décrire les opérateurs ensemblistes**
- **Utiliser un opérateur ensembliste pour combiner plusieurs requêtes en une seule**
- **Vérifier l'ordre des lignes ramenées**

6-2

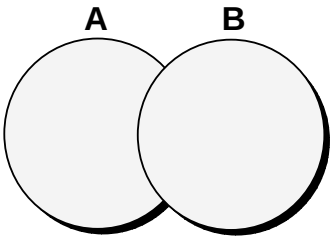
Objectifs

Au cours de ce chapitre, vous allez apprendre à écrire des requêtes avec des opérateurs ensemblistes.

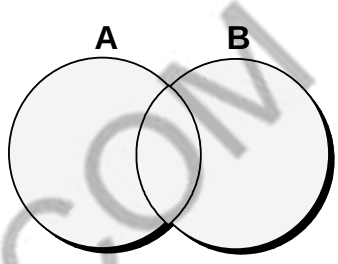
Opérateurs Ensemblistes

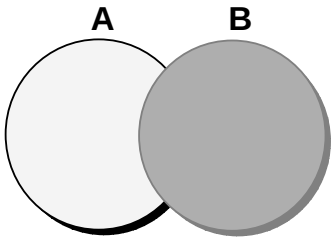


Intersect



Union / Union All





Minus

6-3

Présentation

Les opérateurs ensemblistes combinent les résultats de deux ou plusieurs requêtes en un seul résultat. Une requête composée est une requête contenant des opérateurs ensemblistes.

L'opérateur	**Ramène**
INTERSECT	Toutes les lignes communes aux deux requêtes. INTERSECT combine les deux requêtes et ramène les lignes du premier ordre SELECT identiques aux lignes du second ordre SELECT.
UNION	Toutes les lignes distinctes ramenées par les deux requêtes.
UNION ALL	Toutes les lignes sélectionnées par les deux requêtes, y compris les doublons.
MINUS	Toutes les lignes sélectionnées par le premier ordre SELECT moins les lignes sélectionnées dans le second ordre SELECT.

Tous les opérateurs ensemblistes ont la même priorité. Si un ordre SQL en contient plusieurs, la base de données les évalue de gauche à droite ou de haut en bas, si aucune parenthèse n'indique explicitement un autre ordre. Pour être conforme aux nouvelles normes SQL, la prochaine version de la base de données accordera une priorité plus importante à l'opérateur INTERSECT qu'aux autres opérateurs ensemblistes. Vous devez donc indiquer explicitement par des parenthèses l'ordre d'exécution des requêtes contenant l'opérateur INTERSECT et d'autres opérateurs ensemblistes.

Tables Utilisées dans ce Chapitre

EMP

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		17-NOV-81	5000		10
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	1500		10
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN					30
7499	ALLEN	SALESMAN					30
7844	TURNER	SALESMAN					30
7900	JAMES	CLERK					30
7521	WARD	SALESMAN					30
7902	FORD	ANALYST					20
7369	SMITH	CLERK					20

EMP_HISTORY

EMPID	NAME	TITLE	DATE_OUT	DEPTID
6087	SPENCER	OPERATOR	27-NOV-81	20
6185	VANDYKE	MANAGER	17-JAN-81	10
6235	BALFORD	CLERK	22-FEB-80	20
7788	SCOTT	ANALYST	05-MAY-81	20
7901	LEWEDEC-80	ANALYST	10-JUN-81	30

6-4

Tables utilisées dans ce chapitre

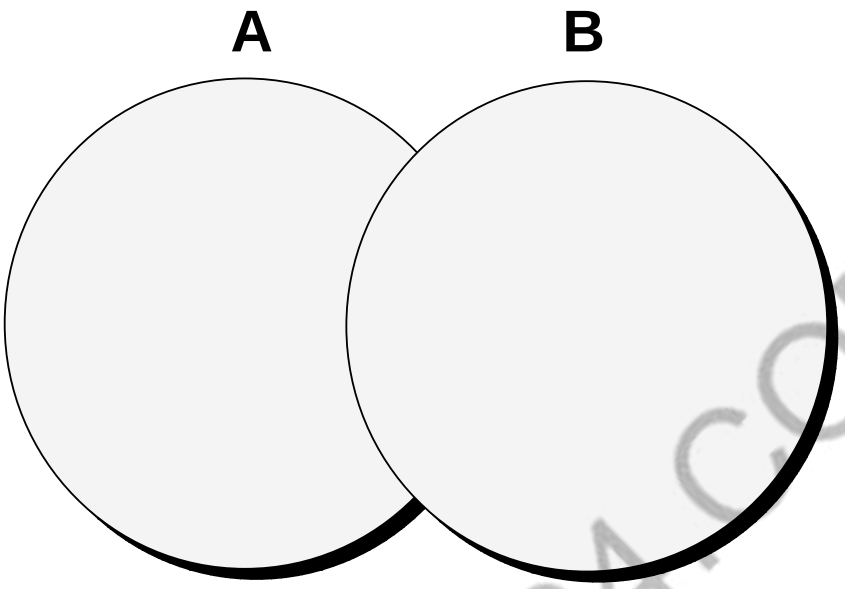
Deux tables sont utilisées dans ce cours :

- La table EMP donne des informations sur tous les employés
- EMP_HISTORY donne des informations sur les employés ayant quitté la société

Le script nécessaire à la création de la table EMP_HISTORY est indiqué dans l'exercice à la fin de ce chapitre.

Hanoune

UNION



6-5

L'Opérateur UNION

Cet opérateur combine le résultat de deux requêtes. Il permet de ramener toutes les lignes issues de plusieurs requêtes et d'éliminer les doublons.

Instructions

- Le nombre de colonnes et les types de données des colonnes doivent être identiques dans les deux ordres SELECT. En revanche, les noms de colonnes peuvent être différents.
- L'opérateur UNION intervient sur toutes les colonnes sélectionnées. Par exemple, si vous modifiez la requête de la page suivante pour sélectionner uniquement les noms des employés et leur poste, ALLEN n'apparaîtra qu'une seule fois dans les résultats.
- Les colonnes NULL sont ignorées lors du contrôle des doublons. Par exemple, si la colonne DEPTNO correspondant à ALLEN contenait une valeur NULL dans le premier ordre SELECT (si DEPTNO n'était pas une colonne NOT NULL) à la place de la valeur 30 comme dans le second ordre SELECT, ALLEN n'apparaîtrait qu'une seule fois dans les résultats.
- L'opérateur IN a une priorité plus élevée que l'opérateur UNION.
- Les requêtes incluant l'opérateur UNION dans la clause WHERE doivent comprendre le même nombre de colonnes et des colonnes du même type que celles de la clause SELECT.
- Par défaut, les données sont affichées par ordre ascendant.

Utilisation de l'Opérateur UNION

Affichez le nom, le poste et le département de tous les employés.

```
SQL> SELECT ename, job, deptno
  2  FROM   emp
  3  UNION
  4  SELECT name, title, deptid
  5  FROM emp_history;
```

```
ENAME      JOB           DEPTNO
---------- --------- ---------
ADAMS      CLERK             30
ALLEN      SALESMAN          30
ALLEN      SALESMAN          20
BALFORD    CLERK             20
BLAKE      MANAGER           30
...
20 rows selected.
```

6-6

L'Opérateur UNION

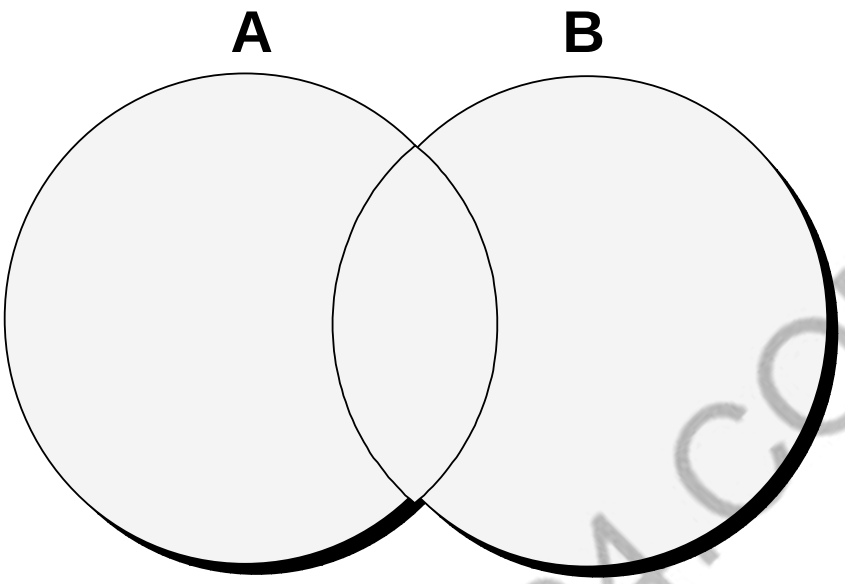
Dans la diapositive ci-dessus, 20 lignes ont été sélectionnées. Même si la combinaison des deux tables totalise plus de 20 enregistrements, seuls 20 sont ramenés, car l'opérateur UNION élimine tous les doublons.

Les tables EMP et EMP_HISTORY comprennent plusieurs colonnes en commun. Par exemple, ENAME et NAME, JOB et TITLE, EMPNO et EMPID. Que se passerait-il si vous souhaitiez afficher les noms des employés, leur poste et leur salaire à l'aide de l'opérateur UNION, sachant que leur salaire ne figure pas dans ces tables ? L'ordre suivant établit une correspondance entre les colonnes ENAME et NAME, JOB et TITLE, puis ajoute le littéral 0 dans l'ordre SELECT de la table EMP_HISTORY pour établir une correspondance avec la colonne numérique SAL dans l'ordre SELECT de la table EMP.

```
SELECT ename, job, sal FROM   emp
UNION
SELECT name, title, 0 FROM   emp_history;
```

```
ENAME      JOB              SAL
---------- --------- ---------
ADAMS      CLERK           1100
ALLEN      SALESMAN           0
ALLEN      SALESMAN        1600
BALFORD    CLERK              0
...
```

UNION ALL



6-7

L'Opérateur UNION ALL

Cet opérateur permet de ramener toutes les lignes issues de plusieurs requêtes.

Règles

- Contrairement à l'opérateur UNION, les doublons ne sont pas éliminés et le résultat n'est pas trié par défaut.
- Il n'est pas possible d'utiliser le mot-clé DISTINCT.

Remarque : Les règles relatives aux opérateurs UNION et UNION ALL sont les mêmes, excepté les deux points ci-dessus.

Utilisation de l'Opérateur UNION ALL

Affichez le nom, le numéro et le poste de tous les employés.

```
SQL> SELECT ename, empno, job
  2  FROM   emp
  3  UNION ALL
  4  SELECT name, empid, title
  5  FROM   emp_history;
```

```
ENAME           EMPNO JOB
---------- --------- ---------
KING             7839 PRESIDENT
BLAKE            7698 MANAGER
CLARK            7782 MANAGER
CLARK            7782 MANAGER
MARTIN           7654 SALESMAN
...
23 rows selected.
```

6-8

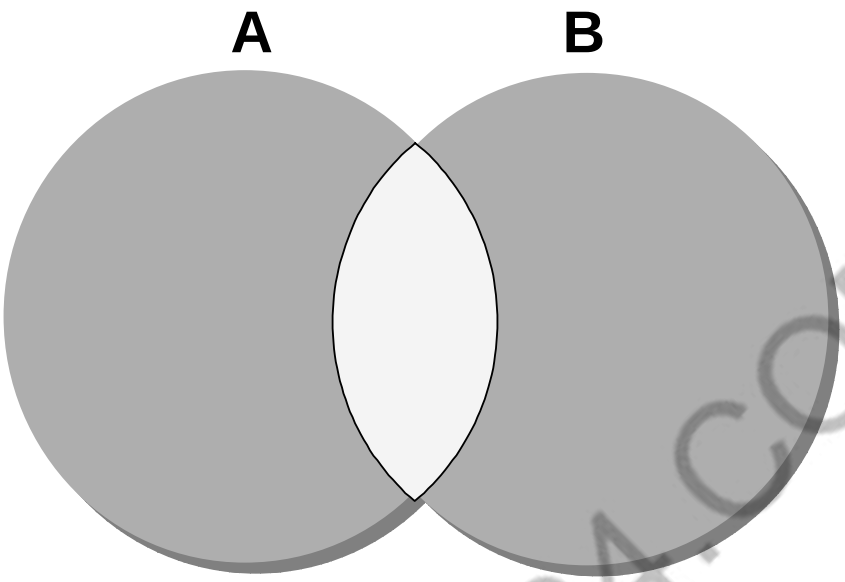
L'Opérateur UNION ALL (suite)

Dans l'exemple ci-dessus, 23 lignes ont été sélectionnées et la combinaison des deux tables totalise 23 enregistrements, ce qui prouve que l'opérateur UNION ALL n'élimine pas les doublons.

Le résultat de l'exemple ci-dessus contient trois groupes de doublons :

```
ENAME           EMPNO JOB
---------- --------- ---------
KING             7839 PRESIDENT
BLAKE            7698 MANAGER
CLARK            7782 MANAGER
CLARK            7782 MANAGER
MARTIN           7654 SALESMAN
ALLEN            7499 SALESMAN
TURNER           7844 SALESMAN
JAMES            7900 CLERK
SMITH            7369 CLERK
SCOTT            7788 ANALYST
ADAMS            7876 CLERK
MILLER           7934 CLERK
BALFORD          6235 CLERK
SCOTT            7788 ANALYST
JEWELL           7001 ANALYST
ALLEN            7499 SALESMAN
BRIGGS           7225 PAY CLERK
...
23 rows selected.
```

INTERSECT



6-9

L'Opérateur INTERSECT

Cet opérateur permet de ramener toutes les lignes communes aux deux requêtes.

- Le nombre de colonnes et les types de données des deux colonnes doivent être identiques dans les deux ordres SELECT. En revanche, les noms de colonnes peuvent être différents.
- L'inversion de l'ordre des tables interrogées ne modifie pas le résultat.
- Comme l'opérateur UNION, l'opérateur INTERSECT ignore les colonnes NULL.
- Les requêtes incluant l'opérateur INTERSECT dans la clause WHERE doivent comprendre le même nombre et des colonnes du même type que celles de la clause SELECT.

Utilisation de l'Opérateur INTERSECT

Affichez les différents noms, numéros et postes des employés présents dans les tables EMP et EMP_HISTORY.

```
SQL> SELECT ename, empno, job
  2  FROM   emp
  3  INTERSECT
  4  SELECT name, empid, title
  5  FROM   emp_history;
```

```
ENAME           EMPNO JOB
---------- --------- ---------
ALLEN            7499 SALESMAN
CLARK            7782 MANAGER
SCOTT            7788 ANALYST
```

6-10

L'Opérateur INTERSECT

Dans l'exemple ci-dessus, seuls les enregistrements ayant les mêmes valeurs dans les colonnes sélectionnées des deux tables sont ramenés par la requête.

Quels seraient les résultats si on ajoutait la colonne DEPTNO dans l'ordre SELECT de la table EMP et la colonne DEPTID dans l'ordre SELECT de la table EMP_HISTORY et si on exécutait la requête ? Les résultats seraient différents en raison de l'ajout d'une autre colonne dont les valeurs peuvent être des doublons.

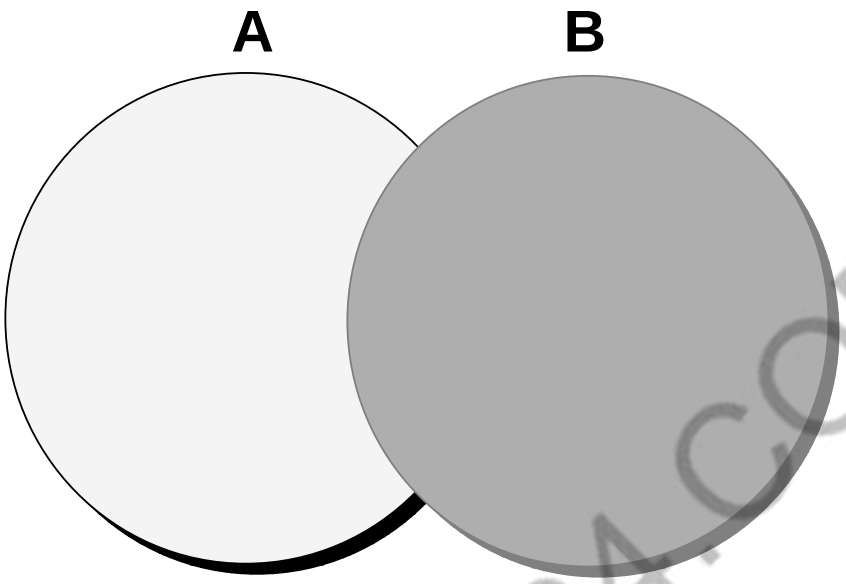
Exemple

```
SQL> SELECT ename, empno, job, deptno
  2  FROM   emp
  3  INTERSECT
  4  SELECT name, empid, title, deptid
  5  FROM   emp_history;
```

```
ENAME           EMPNO JOB           DEPTNO
---------- --------- --------- ---------
CLARK            7782 MANAGER          10
SCOTT            7788 ANALYST          20
```

L’employé ALLEN ne fait plus partie du résultat parce que la valeur de EMP.DEPTNO est différente de la valeur de EMP_HISTORY.DEPTID.

MINUS



6-11

L'opérateur MINUS

Cet opérateur ramène les lignes retournées par la première requête, qui ne le sont pas par la seconde (premier ordre SELECT moins le second).

- Le nombre de colonnes et les types de données des colonnes doivent être identiques dans les deux ordres SELECT. En revanche, les noms de colonnes peuvent être différents.
- Toutes les colonnes incluses dans la clause WHERE doivent également être incluses dans la clause SELECT pour que la requête de l'opérateur MINUS puisse être exécutée.
- Les requêtes incluant l'opérateur MINUS dans la clause WHERE doivent comprendre le même nombre de colonnes et des colonnes du même type que celles de la clause SELECT.

MINUS

Affichez le nom,le numéro et le poste de tous les employés ayant quitté la société.

```
SQL> SELECT name, empid, title
  2  FROM   emp_history
  3  MINUS
  4  SELECT ename, empno, job
  5  FROM   emp;
```

```
NAME           EMPID TITLE
---------- --------- ---------
BALFORD         6235 CLERK
BRIGGS          7225 PAY CLERK
JEWELL          7001 ANALYST
SPENCER         6087 OPERATOR
...
6 rows selected.
```


L'Opérateur MINUS

Dans l'exemple ci-dessus, les noms et les postes des employés de la table EMP sont soustraits de ceux de la table EMP_HISTORY. Cette table ne contient maintenant que les employés actuels et leur poste respectif.

Règles des Opérateurs Ensemblistes

- **Les expressions citées dans la clause SELECT doivent être égales en nombre et avoir des données du même type.**
- **Les doublons sont automatiquement éliminés, sauf avec l'opérateur UNION ALL.**
- **Les noms de colonnes apparaissant dans le résultat sont ceux de la première requête.**
- **Par défaut, le résultat est trié par ordre croissant, sauf avec l'opérateur UNION ALL.**
- **Utilisez des parenthèses pour modifier la séquence d'exécution.**

6-13

Règles des Opérateurs Ensemblistes

- Si deux requêtes sélectionnent des valeurs du type de données CHAR, les valeurs retournées sont des données du type CHAR.
- Si l'une ou l'autre requête sélectionne des valeurs du type de données VARCHAR2, les valeurs retournées sont des données du type VARCHAR2.
- La clause ORDER BY :
  - ne peut apparaître que tout à la fin de l'ordre.
  - accepte un nom de colonne, un alias ou la position de la colonne.
- Les noms de colonne ou alias, si ils sont utilisés dans la clause ORDER BY doivent provenir du premier ordre SELECT.
- Il est possible d'utiliser des opérateurs ensemblistes dans des sous-requêtes.
- Les ordres SELECT sont exécutés de gauche à droite ou de haut en bas.
- Vous pouvez modifier la priorité des opérateurs à l'aide de parenthèses.
- Les requêtes utilisant les opérateurs UNION, INTERSECT, et MINUS dans leur clause WHERE doivent avoir le même nombre et le même type de colonnes que celles sitées dans leur clause SELECT. Par exemple:

```
SQL> SELECT ename, deptno
  2  FROM   emp
  3  WHERE  (ename, deptno) IN (SELECT ename, deptno
  4                             FROM   emp) INTERSECT
  5                            (SELECT name, deptid
  6                             FROM   emp_history);
```

Correspondance des clauses SELECT

Affichez le numéro du département, le lieu et la date d'embauche de tous les employés.

```
SQL> SELECT deptno, TO_CHAR(NULL) location, hiredate
  2  FROM   emp
  3  UNION
  4  SELECT deptno, loc, TO_DATE(NULL)
  5  FROM   dept;
```

```
DEPTNO LOCATION      HIREDATE
--------- ------------- ---------
       10 NEW YORK
       10               09-JUN-81
       10               17-NOV-81
       10               23-JAN-82
       10
       20 DALLAS
       20               17-DEC-80
...
19 rows selected.
```


Correspondance des expressions dans les clauses SELECT

Comme les expressions de la clause SELECT des requêtes composées doivent être égales en nombre et contenir des données du même type, vous pouvez créer des colonnes factices et utiliser les fonctions de conversion de types de données pour respecter cette règle. Dans l'exemple ci-dessus, le nom "location" (lieu) a été attribué à l'en-tête de la colonne factice.

```
   DEPTNO LOCATION      HIREDATE
--------- ------------- ---------
       10 NEW YORK
       10               09-JUN-81
       10               17-NOV-81
       10               23-JAN-82
       10
       20 DALLAS
       20               17-DEC-80
...
       30               03-DEC-81
       40 BOSTON
19 rows selected.
```

Contrôler l'Ordre des Lignes

Créez une phrase anglaise à l'aide de deux opérateurs UNION.

```
SQL> COLUMN a_dummy NOPRINT
SQL> SELECT 'to sing' "My dream", 3 a_dummy
  2  FROM dual
  3  UNION
  4  SELECT 'I''d like to teach', 1
  5  FROM dual
  6  UNION
  7  SELECT 'the world', 2
  8  FROM dual
  9  ORDER BY 2;
```

```
My dream
------------------------
I'd like to teach
the world
to sing
```

6-15

Contrôler l'Ordre des Lignes

Par défaut, le résultat est trié par ordre croissant. Vous pouvez utiliser la clause ORDER BY pour modifier le tri.

Utilisation de la Clause ORDER BY pour Trier des Lignes

Dans une requête composée, la clause ORDER BY ne peut être utilisée qu'une seule fois et elle doit être placée à la fin de la requête. Cette clause accepte le nom de la colonne, un alias ou la position de la colonne.

Remarque : La clause ORDER BY, lorsuq'elle est utilisée dans une requête composée avec l'opérateur UNION (utilisé plus d'une fois), peut seulement utiliser les positions, pas les expressions.

Résumé

- **L'opérateur UNION ramène toutes les lignes distinctes.**
- **L'opérateur UNION ALL ramène toutes les lignes, y compris les doublons.**
- **L'opérateur INTERSECT ramène toutes les lignes partagées par deux requêtes.**
- **L'opérateur MINUS ramène toutes les lignes distinctes sélectionnées par la première requête, et non par la seconde.**
- **La clause ORDER BY doit être placée à la fin de l'ordre.**

6-16

Résumé

La clause ORDER BY ne peut être placée qu'à la fin d'une requête composée.

Les expressions correspondantes des listes SELECT doivent être égales en nombre et comprendre des données du même type.

Présentation des Exercices

Dans ces exercices, vous allez écrire des requêtes en incluant des opérateurs ensemblistes.

- **Utilisation d'autres méthodes de jointure**
- **Ecriture de requêtes composées sous forme d'ordres IF**

6-17

Remarque : Pour créer la table EMP_HISTORY, exécuter le script *emphis.sql*.

Exercice 6

1. Affichez le département qui ne comprend aucun employé.

```
   DEPTNO DNAME
--------- --------------
       40 OPERATIONS
```

2. Retrouvez le poste qui a été attribué dans le deuxième semestre des années 1981 et 1982.

```
JOB
---------
ANALYST
```

3. Ecrivez une requête composée pour afficher la liste des produits (utiliser la table PRICE) en indiquant le pourcentage de remise, l'identifiant des produits (PRODID), ainsi que le nouveau prix (avec remise) et l'ancien prix (STDPRICE):
   - les produits dont le prix est inférieur à $10 sont réduits de 10%
   - ceux dont le prix est compris entre $10 et $30 sont réduits de 15%
   - ceux dont le prix est supérieur à $30 sont réduits de 20%
   - ceux dont le prix est supérieur à $40 ne sont pas réduits.

```
DISCOUNT    PRODID  STDPRICE  ACTPRICE
-------- --------- --------- ---------
10% off     100870       2.4      2.16
10% off     100870       2.8      2.52
10% off     100871       4.8      4.32
10% off     100871       5.6      5.04
10% off     102130       3.4      3.06
10% off     200376       2.4      2.16
10% off     200380         4       3.6
15% off     100860        30      25.5
15% off     101860        24      20.4
15% off     101863      12.5    10.625
20% off     100860        32      25.6
20% off     100860        35        28
20% off     100861        39      31.2
no disc     100861        42        42
no disc     100861        45        45
no disc     100890        54        54
no disc     100890        58        58
```

Exercice 6

Si vous avez le temps, faites les exercices suivants :

4. Affichez la liste des postes dans les départements 10, 30 et 20, en conservant cet ordre. Affichez le poste et le numéro du département.

   Note : La commande SQL*Plus suivante `permet de ne pas afficher la colonne DUMMY(DUMMY est le nom d'une colonne ramenée par le SELECT) : COL dummy NOPRINT`

```
JOB         DEPTNO
--------- --------
CLERK           10
MANAGER         10
PRESIDENT       10
CLERK           30
MANAGER         30
SALESMAN        30
ANALYST         20
CLERK           20
MANAGER         20
```

5. Affichez le numéro des départements dans lesquels on ne trouve pas de poste ANALYST .

```
DEPTNO
--------
    10
    30
    40
```

6. Affichez tous les postes des départements 10 et 20 qui n'existent que dans l'un ou l'autre de ces départements.

```
JOB
--------
ANALYST
PRESIDENT
```

7

Sous-Interrogations

Objectifs

A la fin de ce chapitre, vous saurez :

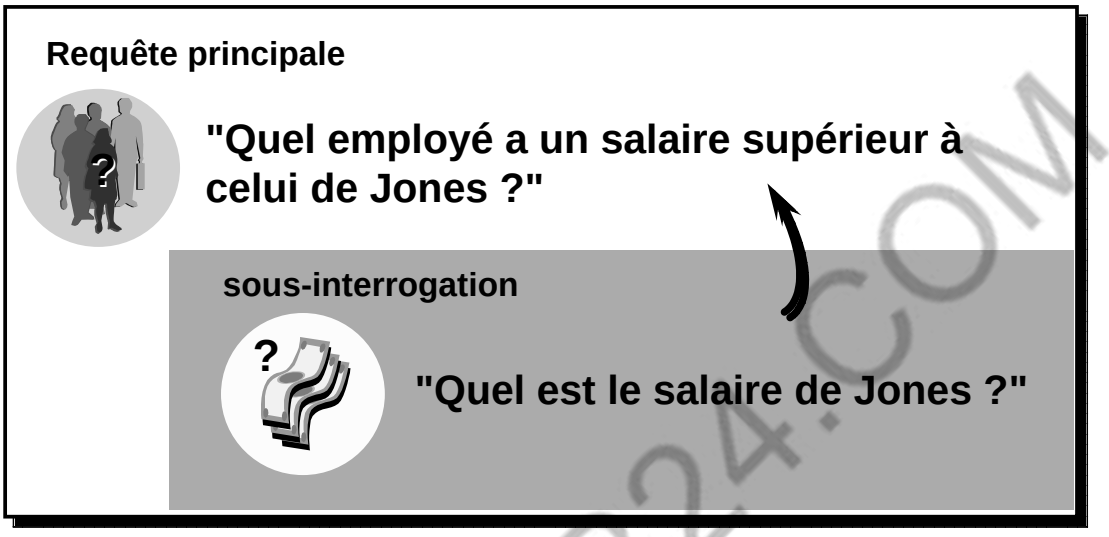
- **Décrire les types de problèmes que les sous-interrogations peuvent résoudre**
- **Définir des sous-interrogations**
- **Enumérer les types de sous-interrogations**
- **Ecrire des sous-interrogations mono-ligne et multi-ligne**

Objectifs

Au cours de ce chapitre, vous allez étudier des aspects plus complexes de l'ordre SELECT. Il est possible d'inclure des sous-interrogations dans la clause WHERE de tout ordre SQL pour obtenir les valeurs de différentes inconnues. Ce chapitre traite des sous-interrogations mono-ligne et multi-ligne.

Utilisation d'une Sous-Interrogation pour Résoudre un Problème

"Qui a un salaire supérieur à celui de Jones ?"



Utilisation d'une Sous-Interrogation pour Résoudre un Problème

Supposons que vous souhaitiez écrire une requête pour trouver qui gagne plus que l'employé Jones.

Pour résoudre ce problème, *deux* requêtes sont nécessaires : une qui trouve le salaire de Jones, et l'autre qui trouve quel employé a un salaire supérieur.

A cet effet, vous pouvez combiner deux requêtes, en en plaçant l'une *à l'intérieur* de l'autre.

La requête interne, ou *sous-interrogation*, ramène une valeur utilisée par la requête externe, ou principale. Utiliser une sous-interrogation revient à exécuter deux requêtes successives en utilisant le résultat de la première comme valeur de recherche de la seconde.

Sous-Interrogations

```
SELECT   select_list
FROM     table
WHERE    expr operator
                    (SELECT      select_list
                    FROM         table);
```

- **La sous-interrogation (requête interne) est exécutée une fois avant la requête principale.**
- **Le résultat de la sous-interrogation est utilisé par la requête principale (externe).**

7-4

sous-interrogations

Une sous-interrogation est un ordre SELECT imbriqué dans une clause d'un autre ordre SELECT. Les sous-interrogations vous permettent de construire des ordres puissants à partir d'instructions toutes simples. Elle s'avèrent très utiles pour sélectionner des lignes d'une table lorsqu'une condition dépend des données de la table elle-même.

Vous pouvez placer une sous-interrogation dans les clauses SQL suivantes :

- WHERE
- HAVING
- FROM

Syntaxe :

operator est un opérateur de comparaison tel que >, = ou IN.

Remarque : les opérateurs de comparaison se classent en deux catégories : les opérateurs mono-ligne (>, =, >=, <, <>, <=) et les opérateurs multi-ligne (IN, ANY, ALL).

Une sous-interrogation est souvent désignée sous le nom "d'ordre SELECT imbriqué", "sous-ordre SELECT", ou encore "ordre SELECT interne". Elle est exécutée en premier, et son résultat sert à évaluer la condition définie dans l'interrogation principale ou externe.

Utilisation d'une Sous-Interrogation

```
SQL> SELECT ename
  2  FROM   emp
  3  WHERE  sal >
  4                (SELECT sal
  5                 FROM   emp
  6                 WHERE  empno=7566);
```

2975

```
ENAME
----------
KING
FORD
SCOTT
```

7-5

Utilisation d'une Sous-Interrogation

Sur la diapositive, la requête interne recherche quel est le salaire de l'employé 7566. La requête externe prend le résultat de la requête interne et l'utilise pour afficher tous les employés qui gagnent plus que cette somme.

Conventions d'Utilisation des Sous-Interrogations

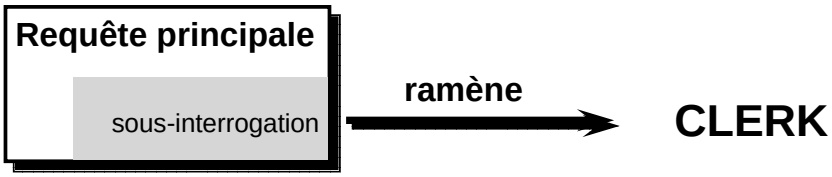
- **Placez les sous-interrogations entre parenthèses.**
- **Placez les sous-interrogations à droite de l'opérateur de comparaison.**
- **N'ajoutez jamais de clause ORDER BY à une sous-interrogation.**
- **Utilisez les opérateurs mono-ligne avec les sous-interrogations mono-ligne.**
- **Utilisez les opérateurs multi-ligne avec les sous-interrogations multi-ligne.**

Convention d'Utilisation des sous-interrogations

- Une sous-interrogation doit être incluse entre parenthèses.
- Une sous-interrogation doit être placée à droite de l'opérateur de comparaison.
- Les sous-interrogations ne doivent pas contenir de clause ORDER BY. Il ne peut y avoir qu'une seule clause ORDER BY par ordre SELECT. Si vous en spécifiez une, elle doit obligatoirement figurer en dernier dans l'ordre SELECT principal.
- Deux catégories d'opérateurs de comparaison sont utilisées dans les sous-interrogations : les opérateurs mono-ligne et les opérateurs multi-ligne .

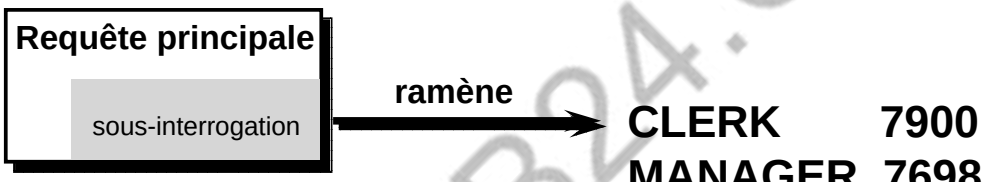
Types de Sous-Interrogations

- **Sous-interrogation mono-ligne**



- **Sous-interrogation multi-ligne**

- **Sous-interrogation multi-colonne**



Types de sous-interrogations

- Sous-interrogation mono-ligne : ordre SELECT interne qui ne ramène qu'une seule ligne
- Sous-interrogation multi-ligne : ordre SELECT interne qui ramène plusieurs lignes
- Sous-interrogation multi-colonne : ordre SELECT interne qui ramène plusieurs colonnes

Sous-Interrogations Mono-ligne

- **Ne ramènent qu'une seule ligne**
- **Utilisent des opérateurs de comparaison mono-ligne**

Opérateur	Signification
=	Egal à
>	Supérieur à
>=	Supérieur ou égal à
<	Inférieur à
<=	Inférieur ou égal à
<>	Différent de

Sous-Interrogations Mono-ligne

Une *sous-interrogation* mono-ligne ne ramène qu'une seule ligne. Elle utilise un opérateur mono-ligne. Les opérateurs mono-ligne sont énumérés sur la diapositive.

Exemple

Afficher les employés occupant le même poste que l'employé 7369.

```
SQL> SELECT   ename, job
  2  FROM     emp
  3  WHERE    job =
  4              (SELECT  job
  5              FROM     emp
  6              WHERE    empno = 7369);
```

```
ENAME      JOB
---------- ---------
JAMES      CLERK
SMITH      CLERK
ADAMS      CLERK
MILLER     CLERK
```

Exécution de Sous-Interrogations Mono-ligne

```
SQL> SELECT   ename, job
  2  FROM     emp
  3  WHERE    job =                    CLERK
  4                  (SELECT     job
  5                  FROM        emp
  6                  WHERE       empno = 7369)
  7  AND      sal >                    1100
  8                  (SELECT     sal
  9                  FROM        emp
  10                 WHERE       empno = 7876);
```

```
ENAME      JOB
---------- ---------
MILLER     CLERK
```


Exécution de Sous-Interrogations Mono-ligne

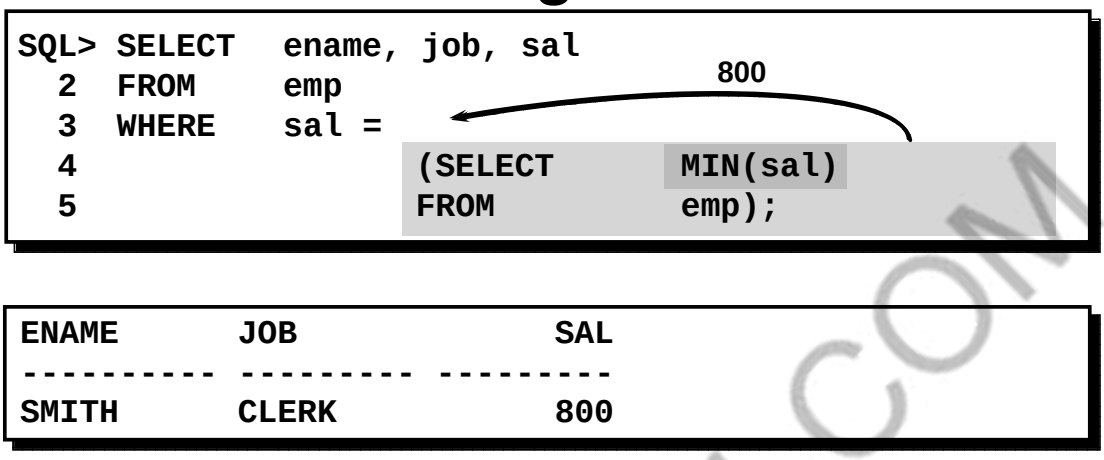
Un ordre SELECT est comparable à un bloc de requêtes. L'exemple ci-dessus affiche les employés occupant le même poste que l'employé 7369 et gagnant plus que l'employé 7876.

Cet exemple se compose de trois blocs de requêtes : la requête externe et deux requêtes internes. Les blocs de requêtes internes sont exécutés en premier, donnant les résultats CLERK et 1100, respectivement. Le bloc de requête externe est ensuite traité et utilise les valeurs ramenées par les requêtes internes pour définir sa condition de recherche.

Comme les deux requêtes internes ramènent des valeurs uniques (CLERK et 1100, respectivement), ce type d'ordre est appelé sous-interrogation mono-ligne.

Remarque: les requêtes internes et externe peuvent rapporter des données issues de différentes tables.

Utilisation de Fonctions de Groupe dans une Sous-Interrogation



Utilisation de Fonctions de Groupe dans une Sous-Interrogation

Vous pouvez afficher les données d'une requête principale en utilisant une fonction de groupe dans une sous-interrogation pour ramener une ligne unique. La sous-interrogation est incluse entre parenthèses et placée à la suite de l'opérateur de comparaison.

L'exemple ci-dessus affiche le nom, le poste et le salaire de tous les employés dont le salaire est égal au salaire minimum. La fonction de groupe MIN transmet une valeur unique (800) à la requête externe.

Clause HAVING avec Sous-Interrogations

- **Oracle Server exécute les sous-interrogations en premier.**
- **Oracle Server ramène les résultats dans la clause HAVING de la requête principale.**

```
SQL> SELECT       deptno, MIN(sal)
  2  FROM         emp
  3  GROUP BY     deptno
  4  HAVING       MIN(sal) >
  5                          (SELECT    MIN(sal)
  6                          FROM       emp
  7                          WHERE      deptno = 20);
```

800

Clause HAVING Comportant des Sous-Interrogations

En plus de la clause WHERE, vous pouvez aussi utiliser des sous-interrogations dans la clause HAVING. Oracle Server exécute la sous-interrogation, et ramène les résultats dans la clause HAVING de la requête externe.

Sur la diapositive, l'ordre SQL affiche tous les départements dont le salaire minimum est supérieur au salaire minimum du département 20.

```
   DEPTNO   MIN(SAL)
--------- ---------
       10      1300
       30       950
```

Exemple

Trouver le poste ayant le salaire moyen le moins élevé.

```
SQL> SELECT   job, AVG(sal)
  2  FROM     emp
  3  GROUP BY job
  4  HAVING   AVG(sal) = (SELECT  MIN(AVG(sal))
                          FROM     EMP
                          GROUP BY job);
```

Qu'est-ce Qui ne Va pas dans cet Ordre ?

```
SQL> SELECT empno, ename
  2  FROM   emp
  3  WHERE  sal =
  4                 (SELECT   MIN(sal)
  5                 FROM      emp
  6                 GROUP BY deptno);
```

```
ERROR:
ORA-01427: single-row sub-query returns more than
one row

no rows selected
```

Opérateur mono-ligne avec sous-interrogation multi-ligne

Erreur Liée aux Sous-Interrogations

L'affichage de plusieurs lignes en réponse à une sous-interrogation mono-ligne est une erreur courante.

Dans l'ordre SQL ci-dessus, la sous-interrogation contient une clause "GROUP BY deptno", qui implique que la sous-interrogation ramènera plusieurs lignes, à savoir une pour chaque groupe trouvé. Ici, le résultat de la sous-interrogation sera 800, 1300 et 950.

La requête externe utilise les résultats de la sous-interrogation (800, 950, 1300) dans sa clause WHERE. Celle-ci contient un opérateur égal (=), c'est-à-dire un opérateur de comparaison mono-ligne qui ne porte que sur une valeur. L'opérateur = ne peut donc accepter les valeurs multiples issues de la sous-interrogation et génère une erreur.

Pour remédier à cette erreur, remplacez l'opérateur = par IN.

Cet Ordre Va-t-il Fonctionner ?

```
SQL> SELECT ename, job
  2  FROM   emp
  3  WHERE  job =
  4                (SELECT job
  5                FROM    emp
  6                WHERE   ename='SMYTHE');
```

```
no rows selected
```

La sous-interrogation ne ramène aucune valeur

Problème lié aux Sous-Interrogations

L'absence de lignes ramenées par la requête interne est une autre erreur courante avec les sous-interrogations.

Dans l'ordre SQL ci-dessus, la sous-interrogation contient la clause "WHERE ename='SMYTHE'", dont le but est, a priori, de trouver l'employé dont le nom est Smythe. L'ordre semble correct et cependant, aucune ligne n'est ramenée lors de l'exécution.

Le problème est que le nom Smythe est mal orthographié. Comme aucun employé ne s'appelle Smythe, la sous-interrogation ramène zéro ligne. Cette sous-interrogation ramenant zéro ligne, l'interrogation principale va, elle aussi, ramener zéro ligne.

Sous-Interrogation Multi-ligne

- **Ramène plusieurs lignes**
- **Utilise des opérateurs de comparaison multi-ligne**

Opérateur	Signification
IN	Egal à un élément quelconque de la liste
ANY	Compare la valeur à chaque valeur ramenée par la sous-interrogation
ALL	Compare la valeur à toutes les valeurs ramenées par la sous-interrogation

Sous-Interrogations Multi-ligne

Les sous-interrogations qui ramènent plusieurs lignes sont appelées *sous-interrogations multi-ligne*. Elles s'utilisent avec un opérateur multi-ligne. Un opérateur multi-ligne porte sur une ou plusieurs valeurs.

Exemple

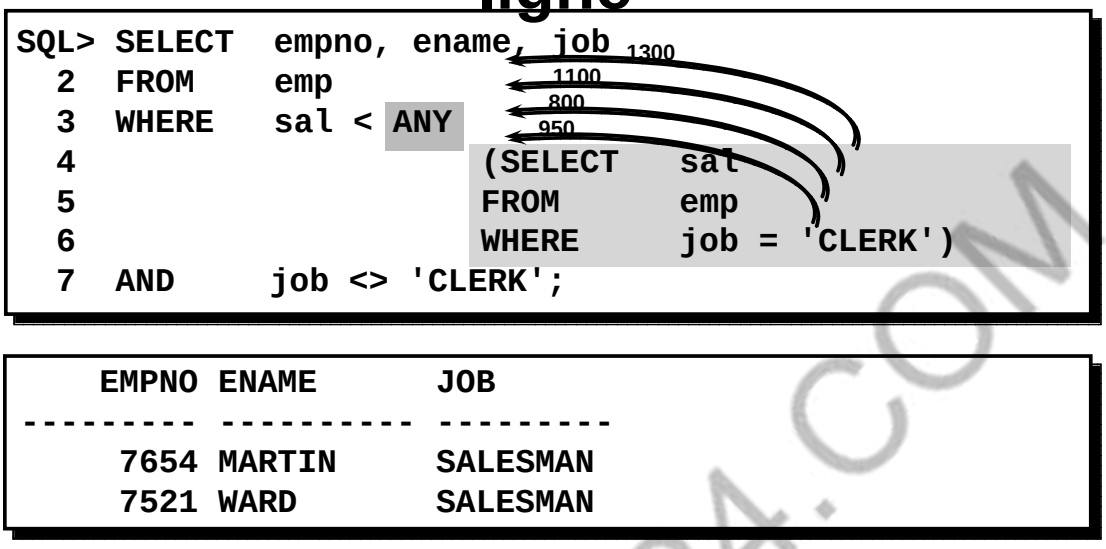
Trouver les employés qui gagnent l'équivalent d'un salaire minimum de département.

```
SQL> SELECT   ename, sal, deptno
  2  FROM     emp
  3  WHERE    sal IN  (SELECT   MIN(sal)
  4                    FROM     emp
  5                    GROUP BY deptno);
```

La requête interne est exécutée en premier, produisant un résultat de trois lignes : 800, 950, 1300. La requête principale est traitée ensuite. Elle utilise les valeurs ramenées par la requête interne pour établir sa condition de recherche. Autrement dit, Oracle Server reçoit la requête principale suivante:

```
SQL> SELECT   ename, sal, deptno
  2  FROM     emp
  3  WHERE    sal IN (800, 950, 1300);
```

Utilisation de l'Opérateur ANY dans les Sous-Interrogations Multi-ligne



Sous-interrogations Multi-ligne

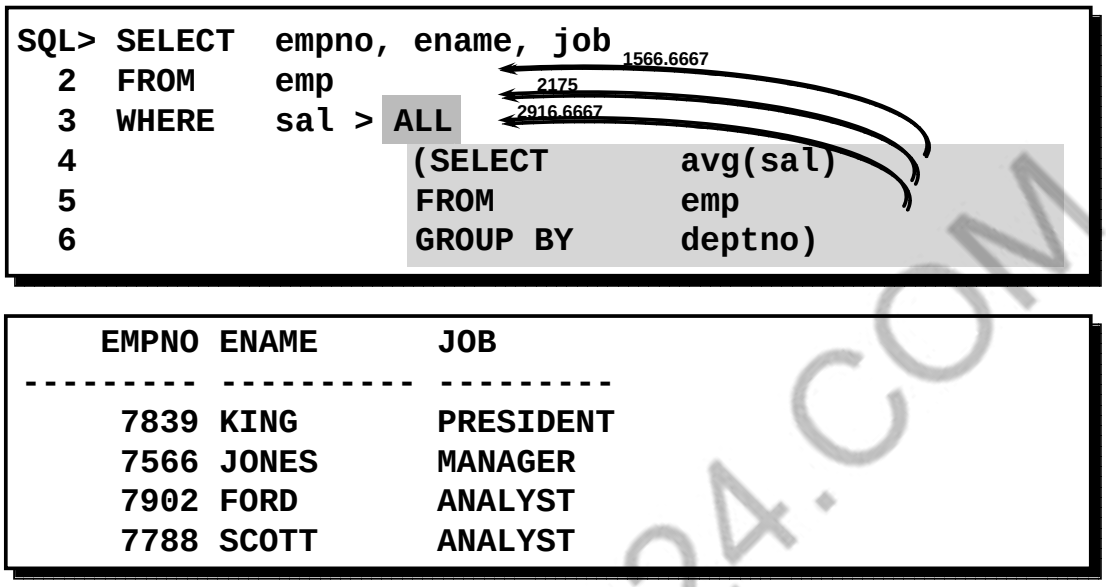
L'opérateur ANY (et son synonyme SOME) compare une valeur à *chaque* valeur ramenée par une sous-interrogation. L'exemple ci-dessus affiche les employés dont le salaire est inférieur à celui de n'importe quel employé 'CLERK' et qui ne sont pas eux-même 'CLERK'. Le salaire maximum d'un 'CLERK' est $1300. L'ordre SQL affiche tous les employés qui ne sont pas 'CLERK' mais qui gagnent moins de $1300.

<ANY signifie inférieur à au moins une des valeurs, donc inférieur au maximum.

>ANY signifie supérieur à au moins une des valeurs, donc supérieur au minimum.

=ANY équivaut à IN.

Utilisation de l'Opérateur ALL dans les Sous-Interrogations Multi-ligne



Sous-Interrogations Multi-ligne

L'opérateur ALL compare une valeur à *toutes les* valeurs renvoyées par une sous-interrogation. L'exemple ci-dessus affiche les employés dont le salaire est supérieur au salaire moyen de tous les départements. Le salaire moyen le plus élevé étant de $2916.66 dans un département, la requête renvoie tous les employés gagnant plus de $2916.66.

>ALL signifie supérieur au maximum et <ALL signifie inférieur au minimum.

L'opérateur NOT peut s'utiliser avec les opérateurs IN, ANY et ALL.

Résumé

Les sous-interrogations sont utiles lorsqu'une requête fait appel à des valeurs inconnues.

```
SELECT   select_list
FROM     table
WHERE    expr operator
                    (SELECT select_list
                    FROM    table);
```


Résumé

Une sous-interrogation est un ordre SELECT imbriqué dans une clause d'un autre ordre SQL. Les sous-interrogations sont utiles lorsqu'une requête est fondée sur des critères inconnus.

Caractéristiques des sous-interrogations :

- Elles peuvent transmettre une ligne de données à un ordre principal contenant un opérateur mono-ligne tel que =, <>, >, >=, < ou <=.
- Elles peuvent transmettre plusieurs lignes de données à un ordre principal contenant un opérateur multi-ligne tel que IN.
- Elles sont traitées en premier par Oracle Server, et leurs résultats sont employés par les clauses WHERE ou HAVING.
- Elles peuvent contenir des fonctions de groupe.

Présentation des Exercices

- **Création de sous-interrogations pour rechercher des valeurs en fonction de critères inconnus**
- **Utilisation de sous-interrogations pour trouver quelles valeurs existent dans un groupe de données et pas dans un autre**

Présentation des Exercices

Au cours de cette série d'exercices, vous allez écrire des requêtes utilisant des ordres SELECT imbriqués.

Vous souhaiterez peut-être créer d'abord la requête interne avant d'écrire la requête principale. Dans ce cas, assurez-vous que la requête interne fonctionne et qu'elle produit bien les données prévues avant d'écrire la requête externe.

Exercices 7

1. Créez une requête pour afficher le nom et la date d'embauche de tous les employés travaillant dans le même département que Blake, à l'exclusion de Blake.

```
ENAME   HIREDATE
------- ---------------
MARTIN  28-SEP-81
ALLEN   20-FEB-81
TURNER  08-SEP-81
JAMES   03-DEC-81
WARD    22-FEB-81

5 rows selected.
```

2. Créez une requête pour afficher le matricule et le nom de tous les employés qui gagnent plus que le salaire moyen. Triez les résultats par ordre décroissant des salaires.

```
EMPNO ENAME
----- -----------
 7839 KING
 7902 FORD
 7788 SCOTT
 7566 JONES
 7698 BLAKE
 7782 CLARK
6 rows selected.
```

3. Ecrivez une requête pour afficher le matricule et le nom de tous les employés qui travaillent dans le même département que tout employé dont le nom contient un *T*. Enregistrez votre ordre SQL dans un fichier nommé *p7q3.sql*.

```
 EMPNO ENAME
------ --------
  7566 JONES
  7788 SCOTT
  7876 ADAMS
  7369 SMITH
  7902 FORD
  7698 BLAKE
  7654 MARTIN
  7499 ALLEN
  7844 TURNER
  7900 JAMES
  7521 WARD

11 rows selected.
```

Exercices 7

4. Modifiez *p7q3.sql* afin d'afficher le matricule, le nom et le salaire de tous les employés qui gagnent plus que le salaire moyen et qui travaillent dans un département avec tout employé dont le nom contient un *T*. Enregistrez à nouveau votre requête sous le nom *p7q4.sql*, puis réexécutez-la.

```
EMPNO ENAME      SAL
----- ------ ------
 7566 JONES    2975
 7788 SCOTT    3000
 7902 FORD     3000
 7698 BLAKE    2850
```

Si vous avez le temps, faites les exercices suivants.

5. Affichez le nom, le numéro de département et le poste de tous les employés dont le département est situé à Dallas.

```
ENAME  DEPTNO JOB
------ ------ ---------
JONES      20 MANAGER
FORD       20 ANALYST
SMITH      20 CLERK
SCOTT      20 ANALYST
ADAMS      20 CLERK
```

6. Affichez le nom et le salaire de tous les employés dont le manager est King.

```
ENAME      SAL
------- ------
BLAKE     2850
CLARK     2450
JONES     2975
```

7. Affichez le numéro de département, le nom et le poste de tous les employés travaillant dans le département des ventes ('SALES').

```
DEPTNO ENAME    JOB
------ -------- ---------
    30 BLAKE    MANAGER
    30 MARTIN   SALESMAN
    30 ALLEN    SALESMAN
    30 TURNER   SALESMAN
    30 JAMES    CLERK
    30 WARD     SALESMAN
6 rows selected.
```

Exercices 7

8. Créez une requête pour afficher les employés qui perçoivent un salaire supérieur à tout employé dont le poste est CLERK. Triez le résultat par ordre décroissant des salaires.

```
ENAME      JOB              SAL
---------- --------- ---------
KING       PRESIDENT       5000
FORD       ANALYST         3000
SCOTT      ANALYST         3000
JONES      MANAGER         2975
BLAKE      MANAGER         2850
CLARK      MANAGER         2450
ALLEN      SALESMAN        1600
TURNER     SALESMAN        1500
8 rows selected.
```

8

Sous-Interrogations Multi-colonne

Objectifs

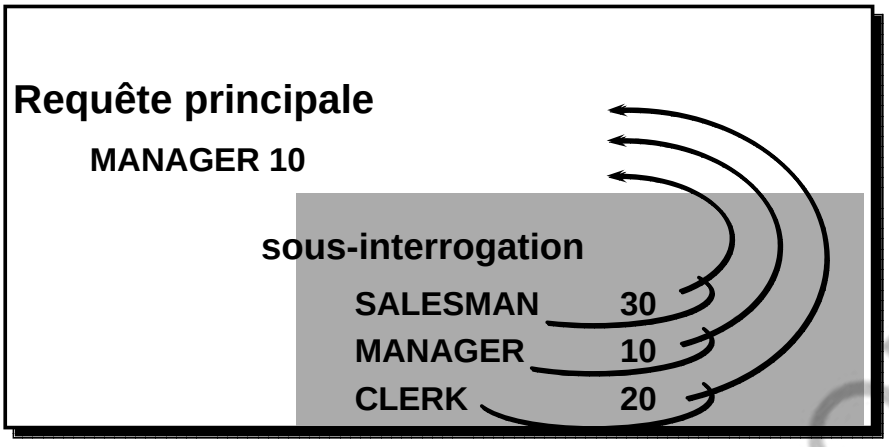
A la fin de ce chapitre, vous saurez :

- **Ecrire une sous-interrogation multi-colonne**
- **Décrire et expliquer le comportement des sous-interrogations lorsqu'elles trouvent des valeurs NULL**
- **Ecrire une sous-interrogation dans une clause FROM**

Objectifs

Au cours de ce chapitre, vous allez apprendre à écrire des sous-interrogations multi-colonne et des sous-interrogations dans la clause FROM d'un ordre SELECT.

Sous-Interrogations Multi-colonne



La requête principale compare **aux** **Valeurs d'une sous-interrogation multi-ligne et multi-colonne**

MANAGER 10

SALESMAN 30
MANAGER 10
CLERK 20

Sous-Interrogations Multi-colonne

Jusqu'à maintenant, vous avez écrit des sous-interrogations mono-ligne et multi-ligne dans lesquelles une seule colonne était comparée via la clause WHERE ou HAVING de l'ordre SELECT principal. Si vous voulez comparer plusieurs colonnes, vous devez écrire une clause WHERE composée utilisant des opérateurs logiques. Les sous-interrogations multi-colonne vous permettent de combiner plusieurs conditions WHERE dans une même clause WHERE.

Syntaxe

```
SELECT   column, column, ...
FROM     table
WHERE    (column, column, ...) IN
                              (SELECT column, column, ...
                               FROM   table
                               WHERE  condition);
```

Utilisation des Sous-Interrogations Multi-colonne

Afficher le nom, le numéro de département, le salaire et la commission de tout employé dont le salaire et la commission correspondent à la fois aux salaire et commission d'un des employé du département 30.

```
SQL> SELECT   ename, deptno, sal, comm
  2  FROM     emp
  3  WHERE    (sal, NVL(comm,-1)) IN
  4                          (SELECT sal, NVL(comm,-1)
  5                           FROM   emp
  6                           WHERE  deptno = 30);
```


Utilisation des Sous-Interrogations Multi-colonne

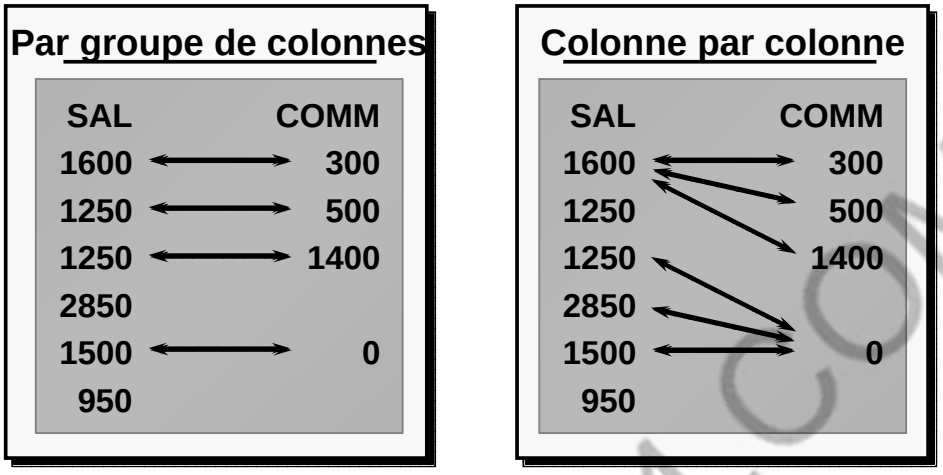
La sous-interrogation ci-dessus, qui ramène plusieurs colonnes, est un exemple de sous-interrogation multi-colonne. Elle compare la colonne SAL et la colonne COMM. Elle affiche le nom, le numéro de département, le salaire et la commission de tout employé dont le salaire et la commission correspondent *à la fois* aux salaire et commission d'un des employés du département 30.

L'ordre SQL ci-dessus donnera les résultats suivants :

```
ENAME      DEPTNO        SAL       COMM
---------- --------- --------- ---------
JAMES              30       950
WARD               30      1250        500
MARTIN             30      1250       1400
TURNER             30      1500          0
ALLEN              30      1600        300
BLAKE              30      2850

6 rows selected.
```

Comparaison de Colonnes



Comparaisons " par Groupe de Colonnes" versus "Colonne par Colone"

Dans une sous-interrogation multi-colonne, les comparaisons peuvent être effectuées "par groupe de colonnes" ou "colonne par colonne". Dans l'exemple de la diapositive précédente, la comparaison exécutée dans la clause WHERE est faite par groupe de colonnes: chaque ligne rapportée par l'ordre SELECT doit comporter *à la fois* le même salaire et la même commission qu'un employé du département 30.

Pour effectuer une comparaison par groupe de colonnes, utilisez une clause WHERE avec seule condition faisant intervenir les différentes colonnes.

Pour effectuer une comparaison colonne par colonne (produit cartésien), utilisez une clause WHERE avec de multiples conditions, chaque condition faisant intervenir une colonne.

Sous-Interrogation avec Comparaison Colonne par Colonne

Afficher le nom, le n° de département, le salaire et la commission de tout employé dont le salaire et la commission correspondent au salaire et à la commission d'un des employés du département 30.

```
SQL> SELECT   ename, deptno, sal, comm
  2  FROM     emp
  3  WHERE    sal IN           (SELECT sal
  4                            FROM    emp
  5                            WHERE   deptno = 30)
  6  AND
  7           NVL(comm,-1) IN  (SELECT NVL(comm,-1)
  8                            FROM    emp
  9                            WHERE   deptno = 30);
```

8-6

Sous-Interrogation Effectuant une Comparaison Colonne par Colonne

L'exemple ci-dessus effectue une comparaison colonne par colonne. Il affiche le nom, le numéro de département, le salaire et la commission de tout employé dont le salaire et la commission correspondent au salaire et à la commission d'un employé du département 30.

L'ordre SQL ci-dessus donnera le résultat suivant :

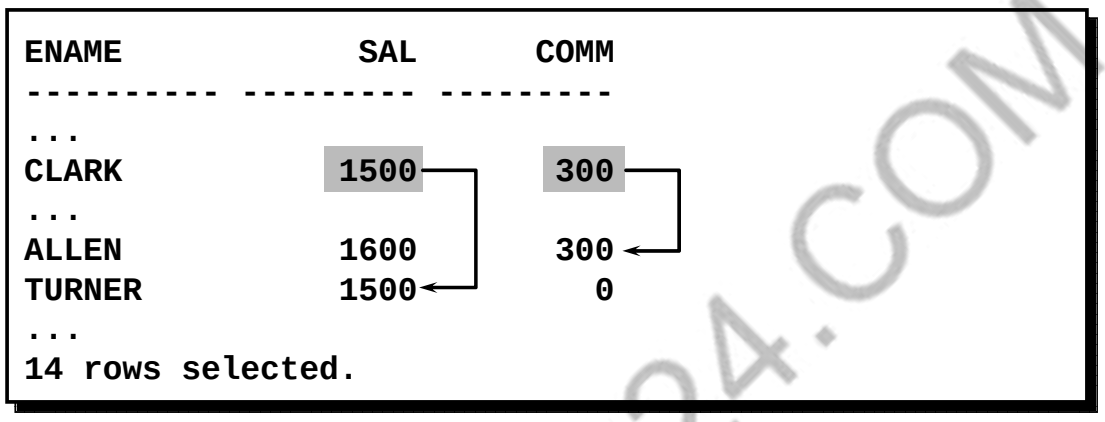
```
ENAME          DEPTNO        SAL       COMM
---------- --------- --------- ---------
JAMES              30        950
BLAKE              30       2850
TURNER             30       1500          0
ALLEN              30       1600        300
WARD               30       1250        500
MARTIN             30       1250       1400

6 rows selected.
```

Le résultat des deux dernières requêtes est identique bien que les conditions de comparaison diffèrent. Cela s'explique par la nature des données présentes dans la table EMP.

Modification de la Table EMP

- **Supposons que le salaire et la commission de l'employé Clark soient modifiés.**
- **Le salaire passe à $1500 et la commission à $300.**



Exemple

Supposons que le salaire et la commission de l'employé Clark soient modifiés de telle sorte qu'il ait le même salaire qu'un employé du département 30 et la même commission qu'un autre employé du département 30.

Le salaire de Clark est désormais égal à celui de Turner ($1500) et sa commission est la même que celle d'Allen ($300).

Nous allons maintenant exécuter une comparaison par groupe de colonnes puis colonne par colonne pour voir combien de lignes sont ramenées dans chaque cas.

Remarque : la syntaxe à utiliser pour mettre à jour les données d'une table sera traitée au cours du chapitre 9.

Comparaison par Groupe de Colonne

```
SQL> SELECT   ename, deptno, sal, comm
  2  FROM     emp
  3  WHERE    (sal, NVL(comm,-1)) IN
  4                          (SELECT sal, NVL(comm,-1)
  5                           FROM   emp
  6                           WHERE  deptno = 30);
```

```
ENAME          DEPTNO        SAL       COMM
---------- --------- --------- ---------
JAMES              30       950
WARD               30      1250        500
MARTIN             30      1250       1400
TURNER             30      1500          0
ALLEN              30      1600        300
BLAKE              30      2850

6 rows selected.
```


Sous-Interrogation avec Comparaison par Groupe de Colonnes

Les résultats de la sous-interrogation avec comparaison par groupe de colonnes ne changent pas et ramènent toujours six lignes.

Comparaison Colonne par Colonne

```
SQL> SELECT   ename,deptno, sal, comm
  2  FROM     emp
  3  WHERE    sal IN           (SELECT sal
  4                             FROM   emp
  5                             WHERE  deptno = 30)
  6  AND
  7           NVL(comm,-1) IN (SELECT NVL(comm,-1)
  8                            FROM   emp
  9                            WHERE  deptno = 30);
```

```
ENAME          DEPTNO        SAL       COMM
---------- --------- --------- ---------
JAMES              30       950
BLAKE              30      2850
TURNER             30      1500          0
CLARK              10      1500        300
...
7 rows selected.
```

8-9

Sous-Interrogation avec Comparaison Colonne par Colonne

Le résultat de la sous-interrogation avec comparaison colonne par colonne inclut l'employé Clark puisque maintenant, son salaire est le même que celui de Turner et sa commission identique à celle d'Allen.

Valeurs NULL dans une Sous-Interrogation

```
SQL> SELECT   employee.ename
  2  FROM     emp employee
  3  WHERE    employee.empno NOT IN
                            (SELECT manager.mgr
                             FROM   emp manager);
no rows selected.
```


Présence d'une Valeur NULL dans les Résultats d'une Sous-Interrogation

L'ordre SQL ci-dessus tente d'afficher tous les employés n'ayant pas de subordonné. L'ordre SQL ne ramène aucune ligne alors que logiquement, il aurait dû en renvoyer huit. En fait, l'une des valeurs ramenées par la sous-interrogation étant une valeur NULL, la requête principale ne renvoie aucune ligne. En effet, quand dans une interrogation, on utilise un opérateur de comparaison (=,>, <,...) pour comparer une valeur à NULL, cette interrogation retourne zéro ligne. NULL ne peut être testé qu'avec les opérateurs IS NULL et IS NOT NULL.

C'est pourquoi, chaque fois que les résultats d'une sous-interrogation risquent de contenir des valeurs NULL, vous ne devez pas utiliser l'opérateur NOT IN. L'opérateur NOT IN est l'équivalent de !=ALL.

En revanche, la présence d'une valeur NULL dans le résultat d'une sous-interrogation ne posera pas de problème si vous utilisez l'opérateur IN. Celui-ci est l'équivalent de =ANY. Par exemple, pour afficher les employés qui ont des subordonnés, utilisez l'ordre SQL suivant :

```
SQL> SELECT    employee.ename
  2  FROM      emp employee
  3  WHERE     employee.empno IN (SELECT manager.mgr
  4                               FROM    emp manager);
```

```
ENAME
----------
KING
…
6 rows selected.
```

Utilisation d'une Sous-Interrogation dans la Clause FROM

```
SQL> SELECT  a.ename, a.sal, a.deptno, b.salavg
  2  FROM    emp a, (SELECT   deptno, avg(sal) salavg
  3                  FROM     emp
  4                  GROUP BY deptno) b
  5  WHERE   a.deptno = b.deptno
  6  AND     a.sal > b.salavg;
```

```
ENAME             SAL     DEPTNO     SALAVG
---------- --------- --------- ----------
KING             5000         10  2916.6667
JONES            2975         20       2175
SCOTT            3000         20       2175
...
6 rows selected.
```


Utilisation d'une Sous-Interrogation dans la Clause FROM

Vous pouvez utiliser une sous-interrogation dans la clause FROM d'un ordre SELECT. Cette méthode présente de nombreuses similitudes avec le mode d'utilisation des vues. L'exemple ci-dessus affiche le nom, le salaire, le numéro de département et le salaire moyen du département pour tous les employés gagnant plus que le salaire moyen de leur département.

Résumé

- **Une sous-interrogation multi-colonne ramène plus d'une colonne.**
- **Dans une sous-interrogation multi-colonne, les comparaisons de colonnes peuvent être effectuées par groupe de colonne ou colonne par colonne.**
- **Une sous-interrogation multi-colonne peut également s'utiliser dans la clause FROM d'un ordre SELECT.**

Présentation des Exercices

Création de sous-interrogations multi-colonne

Présentation des Exercices

Au cours de cette série d'exercices, vous allez écrire des sous-interrogations multi-colonne.

Exercices 8

1. Ecrivez une requête pour afficher le nom, le numéro de département et le salaire de tout employé dont le numéro de département et le salaire correspondent au numéro de département et au salaire d'un des employés touchant une commission.

```
ENAME    DEPTNO    SAL
-------- ------ ------
MARTIN       30   1250
WARD         30   1250
TURNER       30   1500
ALLEN        30   1600
```

2. Affichez le nom de l'employé, le nom du département et le salaire de tout employé dont le salaire et la commission sont tous les deux à la fois équivalents au salaire et à la commission de n'importe quel employé basé à Dallas.

```
ENAME   DNAME         SAL
------- --------- ------
SMITH   RESEARCH      800
ADAMS   RESEARCH     1100
JONES   RESEARCH     2975
FORD    RESEARCH     3000
SCOTT   RESEARCH     3000
```

3. Créez une requête pour afficher le nom, la date d'embauche et le salaire pour tous les employés touchant le même salaire et la même commission que Scott.

```
ENAME   HIREDATE    SAL
------- --------- ------
FORD    03-DEC-81   3000
```

9

Ecriture de Sous-Interrogations Synchronisées

Objectifs

A la fin de ce chapitre, vous saurez :

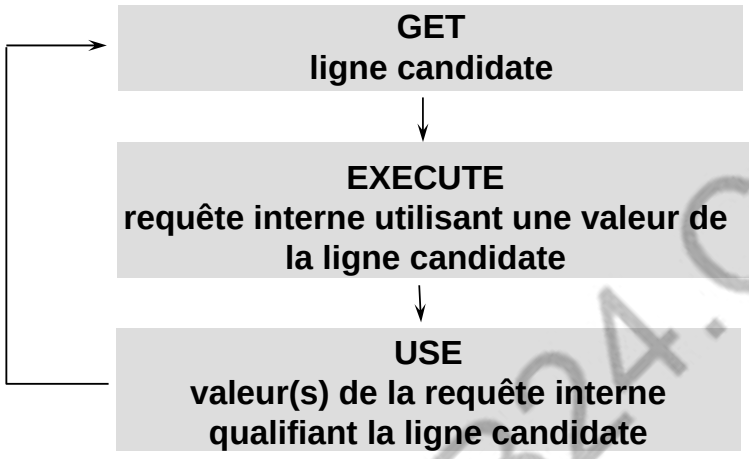
- **Décrire les types de problèmes qui peuvent être résolus à l'aide de Sous-Interrogations Synchronisées**
- **Ecrire des Sous-Interrogations Synchronisées**
- **Utiliser les opérateurs EXISTS et NOT EXISTS**

Objectifs

Dans ce chapitre, vous allez apprendre à résoudre différents problèmes à l'aide de Sous-Interrogations Synchronisées.

Sous-Interrogations Synchronisées

Conçue pour un traitement ligne à ligne, chaque sous-interrogation est exécutée une seule fois pour chaque ligne de la requête externe.



9-3

Sous-Interrogations Synchronisées

Une Sous-Interrogation Synchronisée ou Corrrélée est une sous-interrogation imbriquée qui est exécutée une seule fois pour chaque ligne traitée par la requête principale et qui, pendant son exécution, utilise une valeur d'une colonne de la requête externe.

sous-interrogations Imbriquées et Sous-Interrogations Synchronisées

Avec une sous-interrogation imbriquée normale, l'ordre SELECT interne est d'abord exécuté pour retourner des valeurs qui seront récupérées par la requête principale. Par contre, une Sous-Interrogation Synchronisée est exécutée une fois pour chaque ligne candidate prise en compte par la requête externe. Autrement dit, la requête interne est pilotée par la requête externe.

Exécution d'une Sous-Interrogation Synchronisée

1. Récupérer une ligne candidate (extraite par la requête externe).
2. Exécuter la requête interne avec la valeur de la ligne candidate.
3. Utiliser la ou les valeurs résultant de la requête interne pour sélectionner ou ne pas sélectionner la ligne candidate.
4. Répéter ces étapes jusqu'à ce qu'il ne reste plus de lignes candidates.

Même si cette description est centrée principalement sur les Sous-Interrogations Synchronisées des ordres SELECT, elle s'applique également aux ordres UPDATE et DELETE (voir chapitre concernant la Manipulation des Données).

Sous-Interrogations Synchronisées

Syntaxe

```
SELECT outer1, outer2, ...
FROM   table1 alias1
WHERE  outer1 operator (SELECT inner1
                        FROM   table2 alias2
                        WHERE  alias1.outer2 = alias2.inner1);
```

La sous-interrogation fait référence à une colonne d'une table de la requête principale.

Sous-Interrogations Synchronisées (suite)

Une Sous-Interrogation Synchronisée permet de "lire" chaque ligne d'une table et de comparer les valeurs de chaque ligne aux données associées. Elle est utilisée chaque fois qu'une sous-interrogation doit retourner un résultat ou un ensemble de résultats différent pour chaque ligne candidate prise en compte par la requête principale. Autrement dit, une Sous-Interrogation Synchronisée permet de répondre à une question à plusieurs choix, dont la réponse dépend de la valeur de chaque ligne traitée par l'ordre maître.

Oracle exécute une Sous-Interrogation Synchronisée lorsque la sous-interrogation fait référence à une colonne d'une table de la requête principale.

Utilisation de Sous-Interrogations Synchronisées

Recherchez tous les employés dont le salaire est supérieur au salaire moyen de leur département.

```
SQL> SELECT empno, sal, deptno
  2  FROM   emp outer
  3  WHERE  sal > (SELECT AVG(sal)
  4                FROM   emp inner
  5                WHERE  outer.deptno = inner.deptno);
```

Chaque fois que la requête externe est traitée, la requête interne est exécutée.

```
    EMPNO        SAL     DEPTNO
--------- ---------- ---------
     7839       5000         10
     7698       2850         30
     7566       2975         20
 ...
 6 rows selected.
```


Utilisation de Sous-Interrogations Synchronisées

Dans l'exemple ci-dessus, nous recherchons les employés dont le salaire est supérieur au salaire moyen de leur département. Dans ce cas, la Sous-Interrogation Synchronisée calcule explicitement le salaire moyen de chaque département.

Comme les requêtes externe et interne utilisent toutes les deux la table EMP dans la clause FROM, un alias est attribué à chaque ordre SELECT pour plus de clarté. Grâce aux alias, les ordres SELECT deviennent plus lisibles et la requête s'exécute correctement, car elle peut distinguer la colonne de la table interne de celle de la table externe, ce qui ne serait pas possible sans les alias.

Utilisation de l'Opérateur EXISTS

Dès qu'il trouve une ligne par la sous-interrogation :

- **La recherche dans la requête interne est interrompue.**
- **La condition est vraie (TRUE).**

S'il ne trouve aucune ligne par la sous-interrogation :

- **La condition est fausse (FALSE).**

L'Opérateur EXISTS

Avec des ordres SELECT imbriqués, tous les opérateurs logiques sont valides. De plus, vous pouvez recourir à l'opérateur EXISTS. Cet opérateur est souvent utilisé dans les Sous-Interrogations Synchronisées, car il permet de tester s'il existe une valeur. Si la valeur existe, il retourne la valeur TRUE (vrai) ; sinon, il retourne la valeur FALSE (faux). De la même façon, l'opérateur NOT EXISTS garantit qu'il n'existe aucune valeur.

Utilisation de l'Opérateur EXISTS

Recherchez les employés ayant au moins une personne sous leur responsabilité.

```
SQL> SELECT empno, ename, job, deptno
  2  FROM   emp outer
  3  WHERE  EXISTS (SELECT empno
  4                 FROM   emp inner
  5                 WHERE  inner.mgr = outer.empno);
```

```
     EMPNO ENAME      JOB           DEPTNO
---------- ---------- --------- ----------
      7839 KING       PRESIDENT         10
      7698 BLAKE      MANAGER           30
      7782 CLARK      MANAGER           10
      7566 JONES      MANAGER           20
...
6 rows selected.
```


Utilisation de l'Opérateur EXISTS

Cet opérateur garantit que la recherche dans la requête interne ne se poursuivra pas s'il existe au moins une correspondance entre un responsable et des numéros d'employés.

Utilisation de l'Opérateur NOT EXISTS

Recherchez tous les départements qui ne comprennent pas d'employés.

```
SQL> SELECT  deptno, dname
  2  FROM    dept d
  3  WHERE   NOT EXISTS (SELECT '1'
  4                      FROM   emp e
  5                      WHERE  d.deptno = e.deptno);
```

```
DEPTNO    DNAME
--------- ----------
       40 OPERATIONS
```


Utilisation de l'Opérateur EXISTS (suite)

Notez que l'ordre interne SELECT ne doit pas retourner nécessairement une valeur spécifique, étant donné qu'aucun littéral ne peut être sélectionné. Pour des raisons de performances, il est plus rapide de sélectionner une constante qu'une colonne.

Autre solution

```
SQL> SELECT deptno, dname
  2  FROM   dept
  3  WHERE deptno NOT IN (SELECT deptno
  4                       FROM   emp);
```

Comme le montre cet exemple, il est possible d'utiliser une structure NOT IN à la place de l'opérateur NOT EXISTS. Cependant, vous devez l'utiliser avec prudence, car elle retourne la valeur FALSE si un membre du groupe a une valeur NULL. Dans ce cas, votre requête ne ramènera aucune ligne.

Résumé

- **Les Sous-Interrogations Synchronisées s'avèrent très utiles chaque fois qu'une sous-interrogation doit retourner un résultat différent pour chaque ligne candidate.**
- **EXISTS est un opérateur booléen qui permet de tester l'existence d'une valeur.**
- **Il est possible d'utiliser des Sous-Interrogations Synchronisées avec les ordres SELECT, UPDATE et DELETE.**

Même si cette description est centrée principalement sur les Sous-Interrogations Synchronisées des ordres SELECT, elle s'applique également aux ordres UPDATE et DELETE (voir chapitre concernant la Manipulation des Données).

Présentation des Exercices

- **Ecriture de Sous-Interrogations Synchronisées**
- **Utilisation de l'opérateur EXISTS**

Présentation des Exercices

Au cours de ces exercices, vous allez écrire des Sous-Interrogations Synchronisées.

Exercice 9

1. Ecrivez une requête pour afficher les trois meilleurs salaires dans la table EMP. Affichez les noms des employés et leur salaire.

```
ENAME             SAL
---------- ---------
KING             5000
FORD             3000
SCOTT            3000
```

2. Recherchez tous les employés qui ne sont pas des responsables.

   a. Utilisez d'abord l'opérateur EXISTS.

```
ENAME
----------
MARTIN
ALLEN
TURNER
JAMES
WARD
SMITH
ADAMS
MILLER
```

   b. Pouvez-vous effectuer cette opération à l'aide de l'opérateur IN ? Pourquoi ?

```
no rows selected
```

Exercice 9

Si vous avez le temps, faites les exercices suivants.

3. Ecrivez une requête pour rechercher tous les employés dont le salaire est supérieur au salaire moyen de leur département. Affichez le numéro de chaque employé, son salaire, son numéro de département et le salaire moyen du département. Triez le résultat en fonction du salaire moyen.

```
ENAME          SALARY     DEPTNO  DEPT_AVG
---------- --------- --------- ---------
ALLEN            1600         30 1566.6667
BLAKE            2850         30 1566.6667
JONES            2975         20      2175
FORD             3000         20      2175
SCOTT            3000         20      2175
KING             5000         10 2916.6667
```

4. Ecrivez une requête pour afficher les employés dont le salaire est inférieur à la moitié du salaire moyen de leur département.

```
ENAME
----------
SMITH
MILLER
```

5. Ecrivez une requête pour afficher les employés ayant un ou plusieurs collègues de leur département dont les dates d'embauche sont postérieures aux leurs et dont les salaires sont plus élevés que les leurs.

```
ENAME
----------
CLARK
JONES
ALLEN
WARD
SMITH
```

10

Mise en Forme des Résultats avec SQL*Plus

Objectifs

A la fin de ce chapitre, vous saurez :

- **Créer des requêtes nécessitant la saisie d'une variable**
- **Personnaliser l'environnement SQL*Plus**
- **Afficher des résultats formatés**
- **Créer et exécuter des fichiers script**
- **Enregistrer les paramètres personnalisés**

10-2

Objectifs

Au cours de ce chapitre, vous allez apprendre à utiliser certaines commandes SQL*Plus. Ces commandes vont permettre de rendre le résultat de vos requêtes SQL plus lisible.

Prenons un fichier de commande contenant une clause WHERE pour sélectionner les lignes à afficher. Pour pouvoir modifier la condition à chaque exécution du fichier, il faut utiliser des variables de substitution. Les variables de substitution peuvent remplacer des valeurs de la clause WHERE, une chaîne de texte et même un nom de colonne ou de table.

Etats Interactifs

Etats Interactifs

Tous les exemples d'états étudiés jusqu'à maintenant n'avaient absolument aucun caractère interactif. Un tel état, inclus dans une application en production, s'éxécuterait sans émettre le moindre message. Les données rapportées seraient uniquement déterminées par la clause WHERE écrite en dur dans le fichier script SQL*Plus.

SQL*Plus vous permet de créer des états qui demandent à l'utilisateur de fournir lui-même des valeurs pour sélectionner les données ramenées. Pour créer de tels états interactifs, il faut placer des *variables de substitution* dans un fichier de commande ou dans un ordre SQL. On peut comparer une variable à un conteneur dans lequel des valeurs sont temporairement stockées.

Variables de Substitution

- **Les variables de substitution SQL*Plus permettent de stocker temporairement des valeurs**
  - **Simple "et commercial" (&)**
  - **Double "et commercial" (&&)**
  - **Commandes DEFINE et ACCEPT**
- **Modification dynamique des en-têtes et pieds de page**

Variables de Substitution

Dans SQL*Plus, vous pouvez utiliser des variables de substitution à simple "et commercial" (&) pour stocker temporairement des valeurs.

Les commandes ACCEPT et DEFINE permettent de prédéfinir des variables SQL*Plus. ACCEPT lit une chaîne de caractères saisie par l'utilisateur et la stocke dans une variable. DEFINE crée une variable et lui affecte une valeur.

Exemples de Sélection de Données

- Etat affichant les chiffres du trimestre en cours ou d'une période spécifiée
- Etat fournissant à son demandeur des informations qui lui sont pertinentes
- Affichage du personnel d'un département donné

Autres Actions Interactives

L'utilisation des variables de substitution ne se limite pas à la clause WHERE. Grâce à ces dernières, il est possible de :

- Modifier dynamiquement les en-têtes et pieds de page
- Récupérer les paramètres d'entrée d'un fichier script sans saisie de l'utilisateur

SQL*Plus n'effectue pas de validation des entrées utilisateur (sauf pour le type de données). Veillez à ce que les messages que vous adressez à l'utilisateur soient simples et sans ambiguïté.

Utilisation de la Variable de Substitution &

Pour demander une valeur à un utilisateur, utilisez une variable préfixée par un simple "et commercial" (&).

```
SQL> SELECT   empno, ename, sal, deptno
  2  FROM     emp
  3  WHERE    empno = &employee_num;
```

```
Enter value for employee_num: 7369

    EMPNO ENAME             SAL    DEPTNO
--------- ---------- --------- ---------
     7369 SMITH             800        20
```


Variable de Substitution Précédée d'un simple "et commercial" (&)

Lors de l'exécution d'un état, il est fréquent que les utilisateurs veuillent limiter dynamiquement les données ramenées. SQL*Plus offre cette possibilité par le biais des variables utilisateur. Lorsque vous utilisez, à l'intérieur d'un ordre SQL, une variable préfixée par un "et commercial" (&), vous n'avez pas à définir la valeur de cette variable.

Notation	Description
&user_variable	Désigne une variable dans un ordre SQL ; si la variable n'existe pas, SQL*Plus invite l'utilisateur à saisir une valeur. SQL*Plus supprimera cette variable dès sa première utilisation.

L'ordre SQL ci-dessus, à son exécution, demandera à l'utilisateur de saisir le matricule d'un employé, puis affichera le matricule, le nom, le salaire et le numéro de département de cet employé.

Lorsqu'un simple "et commercial" (&) préfixe une variable et que cette variable n'est pas définie auparavant, l'utilisateur est invité à saisir une valeur à chaque exécution de la commande.

Utilisation de la Commande SET VERIFY

Affichage du texte de la commande avant et après que SQL*Plus remplace les variables de substitution par les valeurs.

```
SQL> SET VERIFY ON
SQL> SELECT   empno, ename, sal, deptno
  2  FROM     emp
  3  WHERE    empno = &employee_num;
```

```
Enter value for employee_num: 7369
old   3: WHERE empno = &employee_num
new   3: WHERE empno = 7369

...
```


La Commande SET VERIFY

Pour vérifier les changements intervenus dans l'ordre SQL, utilisez la commande SQL*Plus SET VERIFY. La commande SQL*Plus SET VERIFY ON demande à SQL*Plus d'afficher le texte d'une commande avant et après le remplacement des variables de substitution par des valeurs.

L'exemple ci-dessus affiche le texte de la commande tel qu'il se présente avant et après le remplacement de la variable de substitution &employee_num.

Valeurs Caractères et Dates dans les Variables de Substitution

Placez les valeurs de type caractère et date entre simples quotes.

```
SQL> SELECT ename, deptno, sal*12
  2  FROM   emp
  3  WHERE  job='&job_title';
```

```
Enter value for job_title: ANALYST

ENAME          DEPTNO    SAL*12
---------- --------- ---------
SCOTT              20     36000
FORD               20     36000
```


Spécification de Valeurs Caractères et Dates avec les Variables de Substitution

Dans une clause WHERE, les valeurs de type caractère et date doivent être incluses entre simples quotes. La même règle s'applique aux variables de substitution.

Incluez la variable entre simples quotes dans l'ordre SQL lui-même pour éviter d'avoir à les saisir au moment de l'exécution.

La diapositive illustre une requête dont le but est de rechercher le nom, le numéro de département et le salaire annuel de tous les employés en fonction de l'intitulé du poste saisi par l'utilisateur en réponse à une question.

Remarque : vous pouvez appliquer à une variable de substitution des fonctions comme UPPER et LOWER. Par exemple, spécifiez UPPER('&job_title') pour que l'utilisateur ne soit pas obligé de saisir l'intitulé en majuscules.

Spécification de Noms de Colonnes, d'Expressions et de Texte lors de l'Exécution

Les variables de substitution peuvent remplacer :

- Une condition WHERE
- Une clause ORDER BY
- Un nom de colonne
- Un nom de table
- Une expression
- Un ordre SELECT

Spécification de Noms de Colonnes, d'Expressions et de Texte au moment de l'Exécution

Les variables de substitution utilisables dans la clause WHERE d'un ordre SQL peuvent également remplacer des noms de colonnes ou de tables, des expressions ou du texte.

Exemple

Afficher le matricule et n'importe quelle autre colonne et condition relative aux employés.

```
SQL> SELECT empno, &column_name
  2  FROM   emp
  3  WHERE  &condition;
```

```
Enter value for column_name: job
Enter value for condition: deptno = 10

    EMPNO JOB
--------- ---------
     7839 PRESIDENT
     7782 MANAGER
     7934 CLERK
```

Si vous ne saisissez pas de valeur pour la variable de substitution, une erreur se produira lors de l'exécution de l'ordre ci-dessus.

Spécification de Noms de Colonnes, d'Expressions et de Texte lors de l'Exécution

```
SQL> SELECT          empno, ename, job, &column_name
  2  FROM            emp
  3  WHERE           &condition
  4  ORDER BY        &order_column;
```

```
Enter value for column_name: sal
Enter value for condition: sal>=3000
Enter value for order_column: ename

     EMPNO ENAME      JOB              SAL
---------- ---------- --------- ---------
      7902 FORD       ANALYST        3000
      7839 KING       PRESIDENT      5000
      7788 SCOTT      ANALYST        3000
```


Spécification de Noms de Colonnes, d'Expressions et de Texte au moment de l'Exécution

L'exemple ci-dessus interroge la table EMP et affiche le matricule, le nom et le poste de l'employé, ainsi que toute autre colonne spécifiée par l'utilisateur. Celui-ci doit également spécifier la condition de recherche des lignes, ainsi que le nom de la colonne qui sera utilisée comme critère de tri.

Utilisation de la Variable de Substitution &&

Spécifiez un double "et commercial" (&&) si vous voulez réutiliser la valeur de la variable sans interroger l'utilisateur à chaque fois.

```
SQL> SELECT         empno, ename, job, &&column_name
  2  FROM           emp
  3  ORDER BY       &column_name;
```

```
Enter value for column_name: deptno
     EMPNO ENAME      JOB           DEPTNO
--------- ---------- --------- ---------
      7839 KING       PRESIDENT         10
      7782 CLARK      MANAGER           10
      7934 MILLER     CLERK             10
...
14 rows selected.
```

10-10

Variable de Substitution à Double "et commercial" (&&)

Si vous souhaitez réutiliser la valeur de la variable à plusieurs reprises sans demander à l'utilisateur de saisir une valeur à chaque fois, spécifiez une variable de substitution à double "et commercial" (&&). Dans ce cas, le message invitant l'utilisateur à entrer une valeur ne s'affichera qu'une seule fois. Ainsi, dans l'exemple de la diapositive, il n'est demandé qu'une fois de fournir une valeur pour la variable *column_name*. La valeur saisie (deptno) va servir à la fois pour l'affichage et pour le tri des données.

SQL*Plus stocke la valeur fournie au moyen de la commande DEFINE et pourra la réutiliser chaque fois que vous mentionnerez ce nom de variable. Pour supprimer une variable utilisateur ainsi stockée, recourez à la commande UNDEFINE.

Définition des Variables Utilisateur

- **Deux commandes SQL*Plus permettent de prédéfinir des variables :**
  - **DEFINE : crée une variable utilisateur de type CHAR**
  - **ACCEPT : lit la valeur saisie par l'utilisateur et la stocke dans une variable**
- **Si, avec la commande DEFINE, vous voulez spécifier comme valeur une chaîne de caractères contenant un ou plusieurs espaces, vous devez l'inclure entre simples quotes.**

10-11

Définition des Variables Utilisateur

Vous pouvez prédéfinir des variables utilisateur avant d'exécuter un ordre SELECT. SQL*Plus fournit deux commandes à cet effet : DEFINE et ACCEPT.

Commande	Description
DEFINE *variable* = *value*	Crée une variable utilisateur de type CHAR et lui affecte une valeur
DEFINE *variable*	Affiche la variable, sa valeur et son type de donnée
DEFINE	Affiche toutes les variables utilisateur avec leur valeur et leur type de donnée
ACCEPT (*voir la syntaxe sur la diapositive suivante*)	Lit une ligne saisie par l'utilisateur et la stocke dans une variable

La Commande ACCEPT

- **Affiche un message d'invite personnalisé et récupère la saisie de l'utilisateur**
- **Permet un contrôle du type de donnée (CHAR, NUMBER ou DATE)**
- **Peut masquer la valeur saisie par l'utilisateur à des fins de confidentialité**

```
ACCEPT variable [datatype] [FORMAT format]
                [PROMPT text] {HIDE}
```


La Commande ACCEPT

Syntaxe :

variable nom de la variable où sera stockée la valeur. Si la variable n'existe pas, SQL*Plus la crée, sinon, il la redéfinit.
datatype type de données à contrôler, à savoir NUMBER, CHAR ou DATE. Avec le type CHAR, la longueur est limitée à 240 octets. Le type DATE permet de vérifier la confirmité à un modèle de format, mais le type de données de la variable sera CHAR.
FOR[MAT] *format* définit le modèle de format, par exemple, A10 ou 9.999 ou DD/MM/YY.

PROMPT *text* affiche le message spécifié par *text* invitant l'utilisateur à saisir une valeur.
HIDE n'affiche pas la valeur entrée par l'utilisateur, par exemple un mot de passe.

Remarque : lorsque vous incluez un paramètre de substitution SQL*Plus dans une commande ACCEPT, ne le faites pas précéder d'un "et commercial" (&).

Utilisation de la Commande ACCEPT

```
ACCEPT   dept PROMPT 'Provide the department name: '
SELECT   *
FROM     dept
WHERE    dname = UPPER('&dept')
/
```

```
Provide the department name: Sales

   DEPTNO DNAME          LOC
--------- -------------- -------------
       30 SALES          CHICAGO
```


Utilisation de la Commande ACCEPT

Ci-dessus, la commande ACCEPT va récupérer une valeur et la placer dans une variable appelée DEPT. Pour cela, elle affiche le message d'invite suivant : "Provide the department name:". Ensuite, l'ordre SELECT utilise la valeur saisie par l'utilisateur (un nom de département) pour rechercher la ligne appropriée dans la table DEPT.

Si l'utilisateur saisit un nom de département *valide*, l'ordre SELECT s'exécute comme n'importe quel autre ordre SELECT. Autrement dit, il prend cette valeur utilisateur et l'utilise dans la clause WHERE pour effectuer une comparaison avec DNAME.

Remarquez que le caractère & ne précède pas la variable DEPT dans la commande ACCEPT. Il ne figure que dans l'ordre SELECT.

Conseils

- Les deux commandes ACCEPT et DEFINE créent une variable si elle n'existe pas et la redéfinissent automatiquement si elle existe.
- Avec la commande DEFINE, si une chaîne contient un espace, placez-la entre simples quotes (' ').
- La commande ACCEPT permet :
  - D'afficher un message d'invite personnalisé. Si vous n'utilisez pas cette commande, le message "Enter value for variable" apparaîtra par défaut.
  - De contrôler le type de donnée de la valeur saisie, ainsi que son format.
  - De masquer les valeurs saisies par l'utilisateur à des fins de confidentialité.

Commandes DEFINE et UNDEFINE

- **Une variable reste définie jusqu'à ce que vous exécutiez l'une des actions suivantes :**
  - **Utiliser la commande UNDEFINE**
  - **Quitter SQL*Plus**
- **La commande DEFINE permet de lister les variables définies.**
- **Pour définir des variables réutilisables à chaque session, modifiez le fichier *login.sql* afin de les créer au démarrage de SQL*Plus.**

Les Commandes DEFINE et UNDEFINE

Une variable reste définie jusqu'à ce que vous exécutiez l'une des actions suivantes :

- Utiliser la commande UNDEFINE
- Quitter SQL*Plus

Lorsque vous supprimez des variables au moyen de UNDEFINE, vous pouvez vérifier vos suppressions au moyen de la commande DEFINE; de plus, utilisée seule, la commande DEFINE permet de lister l'ensemble des variables définies. Lorsque vous quittez SQL*Plus, les variables définies au cours de cette session sont perdues. Pour définir automatiquement des variables utilisables à chaque session, modifiez votre fichier *login.sql* de manière à créer ces variables au démarrage.

Utilisation de la Commande DEFINE

- **Créer une variable pour stocker le nom de département.**

```
SQL> DEFINE deptname = sales
SQL> DEFINE deptname
```

```
DEFINE DEPTNAME        = "sales" (CHAR)
```

- **Utiliser la variable normalement.**

```
SQL> SELECT *
  2  FROM   dept
  3  WHERE  dname = UPPER('&deptname');
```


Utilisation de la Commande DEFINE

La commande DEFINE vous permet de créer une variable que vous utilisez ensuite comme n'importe quelle autre variable. L'exemple ci-dessus crée une variable DEPTNAME contenant le nom du département SALES. L'ordre SQL utilise ensuite cette variable pour afficher le numéro, le nom et la localisation de ce département.

```
   DEPTNO DNAME          LOC
--------- -------------- -------------
       30 SALES          CHICAGO
```

Pour effacer la variable, utilisez la commande UNDEFINE :

```
SQL> UNDEFINE deptname
SQL> DEFINE deptname
symbol deptname is UNDEFINED
```

Personnalisation de l'Environnement SQL*Plus

- **Utilisez les commandes SET pour contrôler la session courante.**

```
SET system_variable value
```

- **Vérifiez les paramètres définis au moyen de la commande SHOW.**

```
SQL> SET ECHO ON
```

```
SQL> SHOW ECHO
echo ON
```


Personnalisation de l'Environnement SQL*Plus

Vous pouvez contrôler l'environnement de fonctionnement de SQL*Plus au moyen des commandes SET.

Syntaxe :

system_variable	variable qui contrôle un aspect de l'environnement de la session
value	valeur de la variable système

Vous pouvez vérifier les éléments que vous avez définis au moyen de la commande SHOW. Sur la diapositive, la commande SHOW vérifie si le paramètre ECHO a été activé ou désactivé.

Pour vérifier toutes les valeurs définies par la commande SET, utilisez la commande SHOW ALL.

Pour plus d'informations, reportez-vous à *SQL*Plus User's Guide and Reference, Release8,* "Command Reference".

Variables de la Commande SET

- **ARRAYSIZE { 20 | *n* }**
- **COLSEP { _ | *text* }**
- **FEEDBACK { 6 | *n* |OFF | ON }**
- **HEADING { OFF | ON }**
- **LINESIZE { 80 | *n* }**
- **LONG { 80 | *n* }**
- **PAGESIZE { 24 | *n* }**
- **PAUSE { OFF | ON | *text* }**
- **TERMOUT { OFF | ON }**

10-17

Variables de la Commande SET

Variables et valeurs de SET	Description
ARRAY[SIZE] {20| n}	Détermine le nombre de lignes qui seront rapportées de la base en une fois.
COLSEP {_|*text*}	Définit le texte à imprimer entre 2 colonnes. Par défaut, le séparateur est un espace unique
FEED[BACK] {6|*n*|OFF|ON}	Affiche le nombre d'enregistrements ramené par une requête lorsque celle-ci sélectionne au moins *n* enregistrements
HEA[DING] {OFF|ON}	Détermine si les en-têtes de colonnes apparaissent dans les états
LIN[ESIZE] {80|*n*}	Fixe à *n* le nombre de caractères par ligne dans les états
LONG {80|*n*}	Définit la largeur maximale d'affichage des valeurs de type LONG
PAGES[IZE] {24|*n*}	Spécifie le nombre de lignes par page
PAU[SE] {OFF|ON|*text*}	Permet de contrôler le défilement de l'affichage sur le terminal. L'utilisateur doit appuyer sur [Return] pour voir la suite de l'affichage après chaque pause
TERM[OUT] {OFF|ON}	Détermine si les résultats apparaissent à l'écran

Remarque : *n* représente une valeur numérique. Dans le tableau ci-dessus, les valeurs soulignées indiquent les valeurs par défaut. Si vous n'attribuez pas de valeur à une variable, SQL*Plus utilise la valeur par défaut.

Variables de la Commande SQL*Plus SET

La commande SET permet de valoriser des variables système pour contrôler l'environnement d'une session SQL*Plus. Les éléments suivants sont contrôlés par des variables système :

- **Nombre de lignes blanches entre les enregistrements**
- **Nombre d'espaces entre les colonnes**
- **Caractères utilisés pour souligner les en-têtes de colonnes**
- **Valeur à afficher pour les valeurs NULL**

10-18

Remarque : Pour voir la valeur courante d'une variable de la commande SET, vous pouvez utiliser la commande SHOW. Au prompt SQL*Plus, entrez SHOW suivi du nom de la variable. La commande SHOW ALL permet de visualiser toutes les variables à la fois.

Enregistrement d'une Configuration Personnalisée dans le Fichier *login.sql*

- **Le fichier *login.sql* contient des commandes SET et d'autres commandes SQL*Plus qui s'exécuteront lors de la connexion.**
- **Vous pouvez ajouter d'autres commandes SET au fichier *login.sql*.**

10-19

Valeurs par Défaut Issues du Fichier *login.sql*

Le fichier *login.sql* contient des commandes SET et d'autres commandes SQL*Plus standard dont vous pouvez avoir besoin à chaque session. Le fichier est lu lors de la connexion et les commandes qu'il contient sont implémentées.

Modification des Valeurs par Défaut

Les valeurs implémentées par *login.sql* peuvent être modifiées pendant la session courante, mais les changements ne s'appliquent que durant cette session. Dès que vous quittez, les nouvelles valeurs sont perdues.

Pérennisez vos modifications de valeurs en les plaçant dans le fichier *login.sql.*

Commandes de Format SQL*Plus

- **COLUMN [*column option*]**
- **TTITLE [*text* | OFF | ON]**
- **BTITLE [*text* | OFF | ON]**
- **BREAK [ON *report_element*]**

Rendre un Etat plus Lisible

Contrôlez la mise en forme des états au moyen des fonctions suivantes :

Commande	Description
COL[UMN] [*column option*]	Contrôle le format des colonnes
TTI[TLE] [*text*|OFF|ON]	Définit un en-tête à afficher en haut de chaque page
BTI[TLE] [*text*|OFF|ON]	Définit un pied de page à afficher en bas de chaque page de l'état
BRE[AK] [ON report_element]	Supprime les doublons et sépare les groupes de lignes en insérant des sauts de ligne

Conseils

- Toutes les commandes de format restent en vigueur jusqu'à la fin de la session SQL*Plus ou jusqu'à ce que les paramètres qu'elles définissent soient écrasés ou supprimés.
- Pensez à rétablir les valeurs par défaut de SQL*Plus après chaque création d'état.
- Il n'existe pas de commande spécifique pour rétablir la valeur par défaut d'une variable SQL*Plus. Vous devez, soit connaître cette valeur précisément, soit quitter la session et vous reconnecter.
- Si vous attribuez un alias à une colonne, vous devez faire référence à cette colonne par son alias et non plus par son nom.

Commande COLUMN

Contrôle l'affichage d'une colonne

```
COL[UMN] [{column|alias} [option]]
```

- **CLE[AR]–Efface le formatage de la colonne**
- **FOR[MAT] *format*–Modifie le format d'affichage d'une colonne en fonction d'un modèle**
- **HEA[DING] *text*–Définit l'en-tête de colonne**
- **JUS[TIFY] {*align*}–Cadre l'en-tête de colonne à gauche, au centre ou à droite**

Options de la Commande COLUMN

Option	Description
CLE[AR]	Efface le formatage de la colonne. Les attributs d'affichage de la colonne reprennent les valeurs par défaut.
FOR[MAT] *format*	Modifie l'affichage des données de la colonne
HEA[DING] *text*	Définit l'en-tête de colonne. Un trait vertical (|) impose un saut de ligne dans l'en-tête si vous ne définissez pas de cadrage
JUS[TIFY] {align}	Justifie l'en-tête de colonne (pas les données) à gauche, au centre ou à droite
NOPRI[NT]	N'affiche pas la colonne
NUL[L] *text*	Spécifie que *text* sera affiché à la place des valeurs NULL
PRI[NT]	Affiche la colonne
TRU[NCATED]	Tronque la chaîne à la fin de la première ligne de l'affichage
WRA[PPED]	Renvoie la fin de la ligne sur la ligne suivante

Utilisation de la Commande COLUMN

- **Définir des en-têtes de colonnes.**

```
COLUMN ename HEADING 'Employee|Name' FORMAT A15
COLUMN sal JUSTIFY LEFT FORMAT $99,990.00
```

- **Afficher le paramètrage courant de la colonne ENAME.**

```
COLUMN ename
```

- **Effacer le paramètrage la colonne ENAME.**

```
COLUMN ename CLEAR
```

10-22

Affichage ou Suppression de Paramètrage de Colonne

Avec la commande COLUMN, vous définissez vos paramétrages de colonnes; pour afficher ou effacer les paramètrages de colonne courants, utilisez les commandes suivantes :

Commande	**Description**
COL[UMN] *column*	Affiche le paramètrage courant de la colonne spécifiée
COL[UMN]	Affiche le paramètrage courant de toutes les colonnes
COL[UMN] column CLE[AR]	Efface le paramètrage de la colonne spécifiée
CLE[AR] COL[UMN]	Efface le paramètrage de toutes les colonnes

Si votre commande est longue, vous pouvez la poursuivre sur la ligne suivante en plaçant un tiret (-) à la fin de la ligne en cours.

Modèles de Format de la Commande COLUMN

Elément	Description	Exemple	Résultat
A*n*	Définit une largeur de *n*	a10	
9	Suppression du zéro de gauche	999999	1234
0	Affiche les zéro de tête	099999	01234
$	Signe dollar flottant	$9999	$1234
L	Symbole monétaire local flottant	L9999	L1234
.	Position du point décimal	9999.99	1234.00
,	Séparateur des milliers	9,999	1,234

Modèles de Format de la Commande COLUMN

La diapositive donne des exemples de modèles de format de la commande COLUMN.

Oracle8 Server affiche une chaîne de signes dièse (#) à la place d'un nombre entier lorsque celui-ci contient plus de chiffres que ne l'autorise le modèle de format. De même, il va afficher une chaîne de signes dièse (#) à la place d'une valeur numérique dont le modèle est alphanumérique.

Autres Options de la Commande COLUMN

Contrôlez l'affichage des colonnes et des en-têtes

```
COL[UMN] [{column|alias} [option]]
```

- **NEW_VALUE** Spécifie un nom de variable pour mémoriser le contenu d'une colonne
- **NOPRINT** Exclut des données de la sortie imprimée
- **CLEAR** Rétablit les attributs d'affichage des colonnes

Options de la Commande COLUMN

Option	Description
NEW_V[ALUE] *variable*	Spécifie une variable pour stocker la valeur d'une colonne à laquelle la commande TTITLE peut accéder.
NOPRI[NT] PRI[NT]	Indique si une colonne doit être imprimée ou non.
CLE[AR] {*column* | COL[UMNS]}	Rétablit les valeurs par défaut des attributs d'affichage de la colonne citée ou de toutes les colonnes.

Utilisation des Commandes TTITLE et BTITLE

Affichage d'en-têtes et de pieds de pages

```
TTI[TLE] [text|OFF|ON]
```

- **Définition d'un en-tête d'état**

```
SQL> TTITLE 'Salary|Report'
```

- **Définition d'un pied de page d'état**

```
SQL> BTITLE 'Confidential'
```

10-25

Les Commandes TTITLE et BTITLE

Utilisez la commande TTITLE pour mettre en forme les en-têtes de pages et la commande BTITLE pour les pieds de page. Ces derniers apparaissent en bas de page, leur position étant en fonction de la valeur définie par PAGESIZE.

Comme les syntaxes de BTITLE et TTITLE sont identiques, nous étudierons seulement la syntaxe de TTITLE. Vous pouvez utiliser le trait vertical (|) pour répartir le texte du titre sur plusieurs lignes.

Syntaxe :

text représente le texte du titre. S'il se compose de plusieurs mots, placez-le entre simples quotes.

L'exemple de TTITLE présenté sur la diapositive définit l'en-tête suivant : Salary sera centré sur une ligne et le mot Report sera centré au-dessous. L'exemple de BTITLE définit le pied de page Confidential.

Utilisée sans options, la commande TTITLE affiche automatiquement la date et le numéro de page sur l'état.

Remarque : sur la diapositive, la syntaxe de TTITLE et de BTITLE est abrégée. TTITLE et BTITLE possèdent différentes options qui sont décrites dans la suite de ce chapitre.

Utilisation de la Commande BTITLE

```
SQL> BTITLE 'Confidential'
SQL> SELECT empno, ename, sal, mgr
  2  FROM   emp;
```

```
     EMPNO ENAME             SAL        MGR
--------- ---------- --------- ---------
      7839 KING             5000 NULL
...
      7876 ADAMS            1100       7788
      7934 MILLER           1300       7782

                    Confidential

14 rows selected.
```


Utilisation de la Commande BTITLE

Dans l'exemple ci-dessus, la commande BTITLE ajoute le pied de page "Confidential" dans le bas de page de l'état.

Conseils d'Utilisation des Commandes BTITLE et TTITLE

- Pour disposer l'en-tête ou le pied de page sur plusieurs lignes, utilisez la barre verticale (|).
- Les commandes TTITLE et BTITLE restent en vigueur jusqu'à ce que vous désactiviez ou remplaciez le titre existant, ou tant que la session SQL*Plus n'est pas terminée.
- Par défaut, la commande TTITLE centre l'en-tête de l'état, affiche la date en haut à gauche et le numéro de page en haut à droite de la page.
- Utilisez les options d'impression pour personnaliser sa mise en forme.

Utilisation de la Commande NEW_VALUE

```
COLUMN deptno NEW_VALUE deptnum FORMAT 99
TTITLE SKIP 1 CENTER 'Report for Dept:' deptnum -
SKIP 2 CENTER
BREAK on deptno SKIP PAGE
SELECT   ename, mgr, deptno, sal
FROM     emp
ORDER BY deptno
/
```

```
...
        Report for Dept: 20

ENAME            MGR DEPTNO       SAL
---------- --------- ------ ---------
JONES           7839     20      2975
...
```


Utilisation de l'Option NEW_VALUE dans la Commande COLUMN

L'exemple ci-dessus illustre la création d'un état dont les résultats sont répartis sur des pages séparées en fonction du département. L'option NEW_VALUE de la commande COLUMN crée une variable (deptnum) associée au numéro de département, elle est ensuite utilisée dans l'ordre TTITLE en tant que partie intégrante de l'en-tête de chaque page de l'état. La commande BREAK ON crée une nouvelle section pour chaque département :

```
          Report for Dept:       10

ENAME            MGR DEPTNO       SAL
---------- --------- ------ ---------
KING                    10      5000
CLARK           7839            2450
MILLER          7782            1300


          Report for Dept:       20

ENAME            MGR DEPTNO       SAL
---------- --------- ------ ---------
JONES           7839     20      2975
...
14 rows selected.
```

Utilisation des Variables de la Commande SET

SET NEWPAGE

```
SET NEWPAGE 3
SELECT empno, ename, mgr
FROM emp
/
```

```
    EMPNO ENAME             SAL        MGR
--------- ---------- --------- ---------
     7839 KING             5000
     7698 BLAKE            2850      7839
     7782 CLARK            2450      7839
...
14 rows selected.
```


Utilisation des Variables de la Commande SET (suite)

Dans l'exemple ci-dessus, la commande SET NEWPAGE 3 insère trois lignes blanches au début de chaque page du résultat.

Utilisation de la Commande BREAK

Supprime les doublons et divise les lignes en sections

- **Suppression des doublons**

```
SQL> BREAK ON ename ON job
```

- **Production de totaux généraux**

```
SQL> BREAK ON report
```

- **Saut de ligne(s) ou de page au niveau des ruptures**

```
SQL> BREAK ON ename SKIP 4 ON job SKIP2
```

10-29

La Commande BREAK

Utilisez la commande BREAK pour diviser les lignes en différentes sections et supprimer les doublons. Pour garantir que la commande BREAK fonctionne bien, il est nécessaire de trier le résultat de la requête sur la ou les colonnes de rupture (clause ORDER BY)

Syntaxe

```
BREAK on column[|alias|row] [skip n|dup|page] on .. [on report]
```

où : *page* débute une nouvelle page lorsque la valeur de rupture change.

skip n saute *n* lignes lorsque la valeur de rupture change.

Un BREAK peut être défini sur les éléments suivants :
- Colonne
- Ligne
- Page
- Etat

dup(licate) affiche les doublons.

Pour effacer toutes les paramétrages définis par BREAK, utilisez la commande CLEAR :

```
CLEAR BREAK
```

Création d'un Fichier Script pour Exécuter un Etat

1. Créez l'ordre SQL SELECT.
2. Enregistrez l'ordre SELECT dans un fichier script.
3. Chargez le fichier script dans un éditeur.
4. Ajoutez des commandes de formatage avant l'ordre SELECT.
5. Vérifiez que l'ordre SELECT est suivi du caractère de terminaison.

Création d'un Fichier Script pour Exécuter un Etat

Vous pouvez entrer toutes les commandes SQL*Plus au niveau du prompt SQL ou bien placer toutes les commandes, y compris l'ordre SELECT, dans un fichier de commande (ou script). Le fichier script type comprend au moins un ordre SELECT et plusieurs commandes SQL*Plus.

Etapes de Création d'un Fichier Script

1. Créez l'ordre SQL SELECT au niveau du prompt SQL. Vérifiez que les données requises pour l'état sont correctes avant d'enregistrer l'ordre dans un fichier. De plus, si vous prévoyez de définir des ruptures, assurez-vous d'avoir spécifié la clause ORDER BY appropriée.
2. Enregistrez l'ordre SELECT dans un fichier script.
3. Entrez les commandes SQL*Plus dans le fichier script.
4. Ajoutez les commandes de formatage requises avant l'ordre SELECT. Veillez bien à ne pas insérer de commandes SQL*Plus à l'intérieur de l'ordre SELECT.
5. Vérifiez que l'ordre SELECT est suivi d'un caractère d'exécution, c'est-à-dire un point-virgule (;) ou un slash (/).

Création d'un Fichier Script pour Exécuter un Etat

6. Effacez les définitions de formatage après l'ordre SELECT.
7. Enregistrez le fichier script.
8. Entrez "START *filename*" pour exécuter le script.

Etapes de Création d'un Fichier Script

6. Ajoutez les commandes SQL*Plus d'effacement de format après le caractère d'exécution. Vous avez aussi la possibilité d'appeler un fichier de réinitialisation qui contient toutes les commandes d'effacement de format.
7. Enregistrez les modifications apportées au fichier script.
8. Dans SQL*Plus, exécutez le fichier script en entrant START *filename* ou @ *filename*. Cette commande est indispensable.

Conseils

- Vous pouvez insérer des lignes blanches entre les commandes SQL*Plus dans un script.
- Vous pouvez abréger les commandes SQL*Plus.
- Placez les commandes de réinitialisation en fin de fichier pour rétablir l'environnement SQL*Plus d'origine.

Utilisation de la Commande CLEAR

Utilisez cette commande pour rétablir les valeurs par défaut des attributs d'affichage des colonnes et des en-têtes.

```
SQL> COLUMN deptno CLEAR
SQL> COLUMN dname CLEAR
SQL> CLEAR BREAK
```

Commande COMPUTE

- **Calcule et affiche les lignes de totaux**

```
COMP[UTE] [function [LABEL labelname] …
       OF {expr|column|alias} …
       ON {expr|column|alias|REPORT|FORM}]
```

- **Utilise différents calculs standard, notamment :**
  - **AVG**
  - **COUNT**
  - **MAXIMUM**
  - **MINIMUM**
  - **NUMBER**
  - **STD**
  - **SUM**
  - **VARIANCE**

10-33

Commande COMPUTE

Cette commande utilise les fonctions de calcul standard suivantes pour traiter et imprimer des totaux par sous-ensembles de lignes sélectionnées. La commande COMPUTE utilisée seule permet d'afficher tous les calculs définits dans la session courante.

Fonction	Calcul	S'applique aux Types de Données
AVG	Moyenne des valeurs non-NULL	NUMBER
COUNT	Comptage des valeurs non-NULL	Tous les types
MAX[IMUM]	Valeur maximale	NUMBER, CHAR, VARCHAR2(VARCHAR)
MIN[IMUM]	Valeur minimale	NUMBER, CHAR, VARCHAR2(VARCHAR)
NUM[BER]	Comptage des lignes	Tous les types
STD	Ecart-type des valeurs non-NULL	NUMBER
SUM	Somme des valeurs non-NULL	NUMBER
VAR[IANCE]	Ecart des valeurs non-NULL	NUMBER

Utilisation de la Commande COMPUTE

```
BREAK ON JOB SKIP 1
COMPUTE SUM OF sal ON job
SELECT job, ename, sal
FROM   emp
WHERE  job IN ('CLERK', 'ANALYST', 'SALESMAN')
ORDER BY job, sal
/
```

```
JOB       ENAME          SAL
--------- ---------- ---------
ANALYST   FORD             3000
          SCOTT            3000
********* ---------- ---------
sum                        6000

CLERK     SMITH             800
…
10 rows selected.
```


Utilisation de la Commande COMPUTE

Dans l'exemple ci-dessus, la commande COMPUTE calcule et affiche le salaire total de chaque poste cité dans la clause WHERE. On obtient la liste des employés correspondant à chaque poste, avec le salaire total à la fin de chaque section.

- Il est possible de diviser le résultat en sections dans toutes les colonnes de la table.
- Dans la colonne JOB, la clause ORDER BY garantit le regroupement des postes pour chaque total.

Remarque : La commande COMPUTE n'a aucun effet si la commande BREAK correspondante n'a pas été passée.

Utilisation de l'Option BREAK avec COMPUTE

```
BREAK ON deptno SKIP 2
COMPUTE MAX OF sal ON deptno
SELECT deptno, ename, sal
FROM   emp
WHERE  job IN ('CLERK', 'ANALYST', 'SALESMAN')
ORDER BY deptno, sal
/
```

```
   DEPTNO ENAME             SAL
--------- ---------- ---------
       10 MILLER          1300
********* ----------
maximum                   1300


       20 SMITH            800
          ADAMS           1100
…
10 rows selected.
```


Utilisation de l'Option BREAK avec la Commande COMPUTE

Dans l'exemple ci-dessus, l'option SKIP de la commande BREAK permet d'insérer des lignes blanches entre le total et les lignes détails du groupe suivant.

La ligne "maximum" générée par la commande COMPUTE représente le salaire maximum parmis les employés de bureau (clerk), les experts (analyst) et les vendeurs (salesman) dans chaque département.

Etat Simple

Fri Oct 24 page 1

Employee
Report

Job Category	Employee	Salary
CLERK	ADAMS	$1,100.00
CLERK	JAMES	$950.00
CLERK	MILLER	$1,300.00
CLERK	SMITH	$800.00
MANAGER	BLAKE	$2,850.00
MANAGER	CLARK	$2,450.00
MANAGER	JONES	$2,975.00
SALESMAN	ALLEN	$1,600.00
SALESMAN	MARTIN	$1,250.00
SALESMAN	TURNER	$1,500.00
SALESMAN	WARD	$1,250.00

Confidential

Exemple

Créer un fichier script pour obtenir un état affichant le poste, le nom et le salaire de chaque employé gagnant moins de $3000. Ajouter un titre intitulé Employee Report, le centrer et l'afficher sur deux lignes, ainsi qu'un pied de page centré dont le libellé est Confidential. Donnez à la colonne 'job title' l'alias 'Job Category' en l'affichant sur deux lignes. Renommez la colonne 'employee name' en 'Employee'. Nommez la colonne des salaires 'Salary' et formatez-la comme suit: $2,500.00.

```
SET PAGESIZE 37
SET LINESIZE 60
SET FEEDBACK OFF
TTITLE 'Employee|Report'
BTITLE 'Confidential'
COLUMN job HEADING 'Job|Category' FORMAT A15
COLUMN ename HEADING 'Employee' FORMAT A15
COLUMN sal HEADING 'Salary' FORMAT $99,999.99
REM ** Insert SELECT statement
SELECT   job, ename, sal
FROM     emp
WHERE    sal < 3000
ORDER BY  job, ename
/
```

Dans SQL*Plus, REM représente une remarque ou un commentaire.

Exemple avec Calculs

```
BREAK ON deptno SKIP 2
COMPUTE MAX LABEL Max_Sal OF sal ON deptno
SELECT deptno, ename, sal
FROM   emp
WHERE  job IN ('CLERK', 'ANALYST', 'SALESMAN')
ORDER BY deptno, sal
/
```

```
   DEPTNO  ENAME                SAL
---------  ----------  ---------
       10  MILLER             1300
*********                 ---------
Max_Sal                        1300


       20  SMITH               800
…
10 rows selected.
```


Utilisation de l'Option LABEL avec la Commande COMPUTE

Dans l'exemple ci-dessus, l'option LABEL utilisée avec la commande COMPUTE permet de changer le libellé par défaut "maximum" généré par la commande COMPUTE MAX OF ....

Exemple avec Calculs

```
1 COLUMN job NOPRINT NEW_VALUE jobname FORMAT A9
2 TTITLE -
  SKIP 1 CENTER 'Salaries for' -
  SKIP 1 CENTER jobname -
  SKIP 2
3 BREAK on job SKIP PAGE
  COMPUTE AVG LABEL '' OF sal ON job 4
5 BTITLE 'Top Secret'
  SELECT job, ename, hiredate, sal
  FROM   emp                          6
  ORDER BY job, sal
  /
```


Création d'un Script d'Etat

1. A l'aide de l'option NEW_VALUE de la commande COLUMN, créez une variable pour afficher le nom du poste dans l'en-tête de chaque page de l'état.
2. A l'aide de la commande TTITLE, créez un en-tête de page :
   - Ajoutez et centrez l'en-tête "Salaries for *jobname*" au début de l'état. Disposez-le sur deux lignes.
   - Insérez une ligne blanche.
3. A l'aide de la commande BREAK, commencez une nouvelle page à chaque nouveau poste.
4. A l'aide de la commande COMPUTE, calculez le salaire moyen de chaque poste.
5. A l'aide de la commande BTITLE, ajoutez "Top Secret" dans le bas de chaque page de l'état.
6. Ajoutez la requête de l'état permettant d'afficher tous les employés. Triez le résultat par poste (JOB) puis par salaire (SAL) pour être cohérent avec les ruptures définies par la commande BREAK.

Résumé

- Utiliser les variables de substitution SQL*Plus pour stocker des valeurs
- Utiliser les commandes SET pour contrôler l'environnement SQL*Plus
- Utiliser la commande COLUMN pour contrôler l'affichage d'une colonne
- Utiliser la commande BREAK pour supprimer les doublons et répartir les lignes en sections
- Utiliser TTITLE et BTITLE pour afficher des en-têtes et des pieds de page

Résumé

Les variables de substitution sont très utiles pour créer des états. Elles permettent de remplacer facilement des valeurs dans une clause WHERE, ainsi que des noms de colonnes et des expressions. Vous pouvez personnaliser des états en écrivant des fichiers script contenant les éléments suivants :

- Variables de substitution à simple et double "et commercial"
- Commande ACCEPT
- Commande DEFINE
- Commande UNDEFINE
- Variables de substitution dans la ligne de commande

Vous pouvez rendre votre état encore plus lisible en utilisant les commandes suivantes:

- COLUMN
- TTITLE
- BTITLE
- BREAK

Présentation des Exercices

- **Création d'une requête utilisant des variables de substitution**
- **Lancement d'un fichier de commande contenant des variables**
- **Utilisation de la commande ACCEPT**

Présentation des Exercices

Les exercices qui suivent vont vous permettre de créer des fichiers qui pourront être exécutés de façon interactive en utilisant des variables de substitution. Ces dernières vont permettre à l'utilisateur de spécifier des critères de sélection au moment de l'exécution de la requête.

Exercices 10

Indiquez si les affirmations ci-dessous sont vraies ou fausses :

1. Une variable de substitution à simple "et commercial" (&) n'affiche un message d'invite qu'une seule fois.
   Vrai/Faux
2. La commande ACCEPT est une commande SQL.
   Vrai/Faux

3. Les tâches de la colonne de gauche ne correspondent pas aux commandes de la colonne de droite. Tracez des lignes pour indiquer la commande appropriée à chaque tâche.

Tâche		**Commande**
Modifier les caractéristiques d'affichage d'une colonne		SET LINESIZE
Calculer et imprimer des totaux		TTITLE
Contrôler le nombre de caractères par ligne		COLUMN
Ajouter un titre à l'état		BREAK
Forcer un saut de page à une rupture		SET NEWPAGE
Déterminer le nombre de lignes blanches en haut de page		COMPUTE

4. Ecrivez un script pour afficher le nom, le poste et le nom du département des employés travaillant dans un département déterminé. La recherche doit pouvoir s'effectuer indifféremment sur des majuscules ou des minuscules. Enregistrez le fichier script sous *p10q4.sql*.

```
Please enter the location name: Dallas
EMPLOYEE NAME JOB          DEPARTMENT NAME
------------- ------------ --------------
JONES         MANAGER      RESEARCH
FORD          ANALYST      RESEARCH
SMITH         CLERK        RESEARCH
SCOTT         ANALYST      RESEARCH
ADAMS         CLERK        RESEARCH
```

Exercices 10

5. Ecrivez un fichier script pour afficher le nom, le poste et la date d'embauche de tous les employés ayant commencé entre deux dates spécifiées. Concaténez le nom et le poste en les séparant par une virgule et un espace, et nommez la colonne 'Employees'. Demandez à l'utilisateur d'entrer l'intervalle de dates au moyen de la commande ACCEPT. Utilisez le format MM/DD/YY. Enregistrez le fichier sous *p10q5.sql*.

```
Please enter the low date range ('MM/DD/YY'): 01/01/81
Please enter the high date range ('MM/DD/YY'): 01/01/82
EMPLOYEES         HIREDATE
----------------- ---------------
KING, PRESIDENT   17-NOV-81
BLAKE, MANAGER    01-MAY-81
CLARK, MANAGER    09-JUN-81
JONES, MANAGER    02-APR-81
MARTIN, SALESMAN  28-SEP-81
ALLEN, SALESMAN   20-FEB-81
TURNER, SALESMAN  08-SEP-81
JAMES, CLERK      03-DEC-81
WARD, SALESMAN    22-FEB-81
FORD, ANALYST     03-DEC-81

10 rows selected.
```

6. Modifiez *p10q4.sql* pour créer un état contenant le nom du département, le nom, la date d'embauche, le salaire et le salaire annuel de tous les employés d'une même localisation. Demandez à l'utilisateur d'entrer cette localisation. Nommez les colonnes DEPARTMENT NAME, EMPLOYEE NAME, START DATE, SALARY et ANNUAL SALARY, en répartissant les en-têtes sur plusieurs lignes. Enregistrez de nouveau le script sous le nom *p10q6.sql*.

```
Please enter the location name: Chicago

DEPARTMENT     EMPLOYEE        START                          ANNUAL
NAME           NAME            DATE                 SALARY    SALARY
-------------- --------------- --------------- ----------- -----------
SALES          BLAKE           01-MAY-81         $2,850.00  $34,200.00
               MARTIN          28-SEP-81         $1,250.00  $15,000.00
               ALLEN           20-FEB-81         $1,600.00  $19,200.00
               TURNER          08-SEP-81         $1,500.00  $18,000.00
               JAMES           03-DEC-81           $950.00  $11,400.00
               WARD            22-FEB-81         $1,250.00  $15,000.00
```

Exercice 10

7. Produisez un état semblable à celui ci. N'oubliez pas de mettre en forme les données, puis enregistrez votre travail dans un fichier nommé *p10_rep.sql.* Le fichier *p10_sel.sql* contient l'ordre SELECT sur lequel vous pouvez vous baser pour créer votre état.

```
SQL>SELECT to_char(sysdate,'mm/dd/yy') today, d.deptno dep,
    2 job, ename, sal, comm, sal * 12 +nvl(comm,0) totsal
    3 FROM emp e, dept d
    4 WHERE e.deptno = d.deptno
    5 ORDER BY d.deptno, job;
```

```
                              Monthly    Annual
 Dept Job       Name           Salary      Comm       Total
----- --------- ------- ---------- --------- -----------
   10 CLERK     MILLER    1,300.00             15,600.00
   10 MANAGER   CLARK     2,450.00             29,400.00
   10 PRESIDENT KING      5,000.00             60,000.00
   20 ANALYST   FORD      3,000.00             36,000.00
   30 SALESMAN  ALLEN     1,600.00     300.00  19,500.00
  ...
```

Exercice 10

8. Améliorez la présentation de l'état que vous avez créé précédemment pour obtenir un état semblable à celui ci-après.

```
02/19/98                 Employee Report          Page:   1

                           Monthly    Annual
 Dept Job       Name         Salary      Comm       Total
----- --------- ------- ---------- --------- -----------
   10 CLERK     MILLER    1,300.00      0.00   15,600.00
      MANAGER   CLARK     2,450.00      0.00   29,400.00
      PRESIDENT KING      5,000.00      0.00   60,000.00
***** *********          ---------- --------- -----------
total                     8,750.00      0.00  105,000.00

. . .
                    Company Confidential

02/19/98                 Employee Report          Page:   2

                           Monthly    Annual
 Dept Job       Name         Salary      Comm       Total
----- --------- ------- ---------- --------- -----------
   30 MANAGER   BLAKE     2,850.00      0.00   34,200.00
      SALESMAN  MARTIN    1,250.00  1,400.00   16,400.00
                ALLEN     1,600.00    300.00   19,500.00
                WARD      1,250.00    500.00   15,500.00
                TURNER    1,500.00      0.00   18,000.00
***** *********          ---------- --------- -----------
total                     9,400.00  2,200.00  115,000.00

                         ---------- --------- -----------
                          29,025.00  2,200.00  350,500.00
                    Company Confidential
```

11

Extraction Hiérarchique

Objectifs

A la fin de ce chapitre, vous saurez :

- Décrire le concept d'une requête hiérarchique
- Créer un état sous forme d'arbre
- Mettre en forme des données hiérarchiques
- Exclure des branches de la structure arborescente

Objectifs

Dans ce chapitre, vous allez apprendre à utiliser des requêtes hiérarchiques pour créer des états sous forme d'arbre.

Quand est-il Possible d'Utiliser une Requête Hiérarchique ?

EMPNO	ENAME	JOB	MGR
7839	KING	PRESIDENT	
7698	BLAKE	MANAGER	7839
7782	CLARK	MANAGER	7839
7566	JONES	MANAGER	7839
7654	MARTIN	SALESMAN	7698
7499	ALLEN	SALESMAN	7698
7844	TURNER	SALESMAN	7698
7900	JAMES	CLERK	7698
7521	WARD	SALESMAN	7698
7902	FORD	ANALYST	7566
7369	SMITH	CLERK	7902
7788	SCOTT	ANALYST	7566
7876	ADAMS	CLERK	7788
7934	MILLER	CLERK	7782

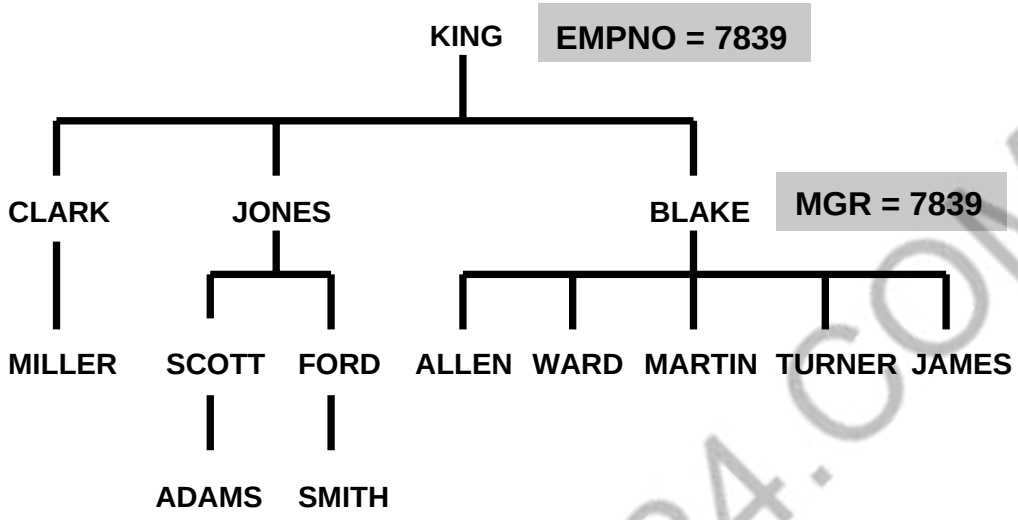
Présentation

Les requêtes hiérarchiques permettent d'extraire des données basées sur une relation hiérarchique entre les lignes d'une table.

Une base de données relationnelle ne stocke pas les enregistrements de façon hiérarchique. Toutefois, lorsqu'il existe une relation hiérarchique entre les lignes d'une même table, le processus appelé *parcours d'arbre* vous permet d'établir une hiérarchie. Une requête hiérarchique est une méthode qui permet d'afficher les branches d'un arbre.

Une requête hiérarchique est possible quand une relation existe entre différentes lignes d'une même table. Par exemple, dans ci-dessus, vous pouvez voir qu'un employé occupant le poste de MANAGER rapporte directement au président de la société. Nous voyons cela parce que la colonne MGR contient le numéro d'employé 7839, lequel correspond à celui du président.

Structure Arborescente



Structure Arborescente

La table EMP présente une structure arborescente qui représente la ligne des responsables. Pour créer une hiérarchie, utilisez la relation existant entre les valeurs équivalentes des colonnes EMPNO et MGR. Cette relation a été exploitée en utilisant une jointure d'une table sur elle-même. Le numéro de responsable (MGR) d'un employé est lui-même le numéro d'employé (EMPNO) de son responsable.

La relation maître-détail des structures arborescentes vous permettent de déterminer :

- La *direction* du parcours de la hiérarchie
- Le *point de départ* de la hiérarchie

Remarque : Le schéma ci-dessus montre une structure d'arbre inversée, représentant la hiérarchie des employés de la table EMP.

Syntaxe Hiérarchique

```
SELECT[LEVEL], column, expr...
FROM table
[WHERE condition(s)]
[START WITH condition(s)]
[CONNECT BY PRIOR condition(s)];
```

condition:

```
expr comparison_operator expr
```


Mots-Clés et Clauses

La présence des clauses CONNECT BY et START WITH permettent d'identifier les requêtes hiérarchiques**.**

Dans la syntaxe :

SELECT	Clause SELECT standard, avec la pseudocolonne LEVEL
LEVEL	Pseudocolonne. LEVEL retourne 1 pour le noeud racine, lorsqu'il vaut 2 il représente un enfant de la racine, etc... LEVEL indique le numéro de niveau hiérarchique de l'élément dans l'arbre.
FROM *table*	Indique le nom de la table, vue ou snapshot contenant les colonnes. Vous ne pouvez effectuer une sélection qu'à partir d'*une seule* table
WHERE	Restreint les lignes ramenées par la requête sans affecter les autres lignes de la hiérarchie.
START WITH	Spécifie les lignes racine de la hiérarchie (point de départ). Cette clause est obligatoire pour une vraie requête hiérarchique.
CONNECT BY PRIOR	Indique les colonnes où il existe une relation entre des lignes parent et enfant. Cette clause est obligatoire dans une requête hiérarchique. PRIOR indique la direction du parcours de l'arbre. Permet également de'éliminer certaines branches de l'arborescence.

L'ordre SELECT ne peut pas contenir de jointure ou être basé sur une vue issue d'une jointure.

Parcours de l'Arbre

DIRECTION

DU HAUT VERS LE BAS ⟶ **Column1 = CLE PARENT**
Column2 = CLE ENFANT

DU BAS VERS LE HAUT ⟶ **Column1 = CLE ENFANT**
Column2 = CLE PARENT

```
CONNECT BY PRIOR column1 = column2
```

Parcours du haut vers le bas sur la table EMP.

```
... CONNECT BY PRIOR empno = mgr
```


Parcours de l'arbre

La direction de la requête peut aller du parent vers l'enfant ou inversement. La direction est indiquée dans la clause CONNECT BY par le positionnement du mot-clé PRIOR. La colonne citée derière l'opérateur PRIOR indique quel est le parent. Pour trouver les enfants d'un parent Oracle Server évalue l'expression PRIOR pour la ligne parent et l'autre expression pour chaque ligne de la table. Les lignes pour lesquelles la condition est vraie sont les enfants du parent. Le serveur Oracle sélectionne toujours les enfants en évaluant la condition CONNECT BY en correspondance avec la ligne parent courante.

Exemples

Parcourir la hiérarchie de haut en bas à partir de la table EMP. Définir une relation hiérarchique dans laquelle la valeur de EMPNO de la ligne parent est égale à la valeur de MGR de la ligne enfant.

```
... CONNECT BY PRIOR empno = mgr
```

Parcours de la table EMP de bas en haut.

```
... CONNECT BY PRIOR mgr = empno
```

Il n'est pas nécessaire de coder l'opérateur PRIOR immédiatement après la clause CONNECT BY. La clause CONNECT BY PRIOR ci-après donne donc le même résultat que celui de l'exemple ci-dessus.

```
... CONNECT BY empno = PRIOR mgr
```

Remarque : La clause CONNECT BY ne peut pas contenir de sous-interrogation.

Parcours de l'Arbre

POINT DE DEPART

- **Indique la condition à remplir**
- **Accepte tous les prédicats valides**

```
START WITH column1 = value
```

Sur la table EMP, commencer par l'employé Blake.

```
... START WITH ename = 'BLAKE'
```


Parcours de l'arbre (suite)

La ligne à utiliser à la racine de l'arbre est déterminée par la clause START WITH. Cette clause peut être associée à n'importe quel prédicat valide**.**

Exemples

Sur la table EMP, commencer par KING, le président de la société.

```
... START WITH mgr IS NULL
```

Sur la table EMP, commencer avec l'employé SMITH. La condition START WITH peut contenir une sous-interrogation.

```
... START WITH empno = (SELECT empno
                        FROM   emp
                        WHERE  ename = 'SMITH')
```

Si la clause START WITH est omise, le parcours commence en considérant toutes les lignes de la table comme des lignes racines. Si une clause WHERE est utilisée, le parcours commence à partir de toutes les lignes qui satisfont la clause WHERE. Ceci ne reflète plus une vraie hiérarchie.

Parcours de l'Arbre

```
SQL> SELECT empno, ename, job, mgr
  2  FROM   emp
  3  CONNECT BY PRIOR mgr = empno
  4  START WITH empno = 7698;
```

```
  EMPNO ENAME      JOB              MGR
------- ---------- --------- ---------
   7698 BLAKE      MANAGER         7839
   7839 KING       PRESIDENT
```


Parcours de bas en haut

Dans l'exemple ci-dessus, un parcours de bas en haut affiche la liste des responsables en commençant par l'employé numéro 7698.

Remarque : Une expression peut porter sur plus d'une colonne.

Exemple

Dans cet example les valeurs de EMPNO sont évaluées pour la ligne parent et les valeurs de MGR, SAL, et COMM sont évaluée pour les lignes enfant. L'opérateur PRIOR s'applique seulement sur la valeur de EMPNO.

```
... CONNECT BY PRIOR empno = mgr AND sal > comm
```

Pour être considérée comme ligne enfant, une ligne doit avoir la même valeur dans la colonne MGR que la colonne EMPNO de la ligne parent et, de plus, avoir la valeur de SAL plus grande que celle de COMM.

Parcours de l'Arbre

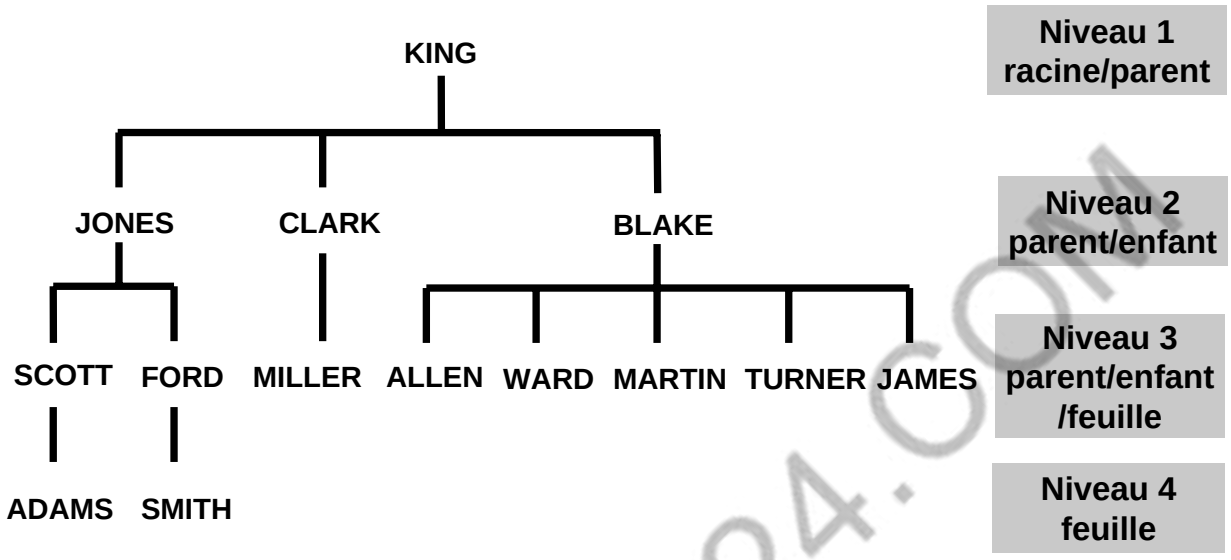
```
SQL> SELECT ename||' reports to '||PRIOR ename "Walk"
  2  FROM   emp
  3  CONNECT BY PRIOR empno = mgr
  4  START WITH ename = 'KING';
```

```
Walk
------------------------------
KING reports to
BLAKE reports to KING
MARTIN reports to BLAKE
ALLEN reports to BLAKE
TURNER reports to BLAKE
JAMES reports to BLAKE
...
14 rows selected.
```


Parcours de haut en bas

Parcours de haut en bas, afficher les noms des employés et le nom de leur responsable. Utiliser l'employé KING comme point de départ. N'imprimer qu'une seule colonne.

Classement des Lignes avec la Pseudocolonne LEVEL



Classement des Lignes Ramenées

Vous pouvez explicitement afficher le classement ou le niveau d'une ligne dans la hiérarchie à l'aide de la pseudocolonne LEVEL. Votre état devient ainsi plus lisible. Les embranchements où la ligne principale se divise en une ou plusieurs branches sont appelés des noeuds, et l'extrémité d'une branche est appelée feuille, ou noeud feuille. Le diagramme ci-dessus affiche les noeuds de l'arbre inversé ainsi que les valeurs de LEVEL. Par exemple, l'employé FORD est à la fois un parent et un enfant, alors que l'employé ALLEN est un enfant et une feuille.

La pseudocolonne LEVEL

Valeur LEVEL	Pour
1	un noeud racine
2	un enfant d'un noeud racine
3	un enfant d'un enfant, etc.
. . .	

Note: Un noeud racine est le plus haut d'un arbre inversé. Tous les noeuds non racine, sont des *noeuds enfants*. Tout noeud qui a des enfant est un noeud parent. Tous les noeuds qui n'ont pas d'enfant sont des neouds feuille.
Le nombre de niveaux retournés par une requête hiérarchique peut être limité par la mémoire disponible pour l'utilisateur.

Formatage des Etats Hiérarchiques avec LEVEL et LPAD

Créer un rapport affichant les niveaux hiérarchiques de la société en commençant par le plus haut, marquer chaque niveau suivant par une indentation jusqu'au niveau le plus bas.

```
SQL> COLUMN org_chart FORMAT A15
SQL> SELECT LPAD(' ', 3 * LEVEL-3)||ename org_chart,
  2  LEVEL, empno, mgr, deptno
  3  FROM   emp
  4  CONNECT BY PRIOR empno = mgr
  5  START WITH mgr is NULL;
```


Mise en Forme des Etats Hiérarchiques à l'Aide de la Pseudocolonne LEVEL

Des numéros de niveau sont attribués aux noeuds d'un arbre à partir de la racine. La fonction LPAD associée à la pseudcolonne LEVEL permet d'afficher un état hiérarchique sous la forme d'un arbre indenté**.**

```
ORG_CHART            LEVEL      EMPNO       MGR     DEPTNO
--------------- --------- --------- --------- ---------
KING                     1      7839                  10
   JONES                 2      7566      7839        20
      SCOTT              3      7788      7566        20
         ADAMS           4      7876      7788        20
      FORD               3      7902      7566        20
         SMITH           4      7369      7902        20
   BLAKE                 2      7698      7839        30
      ALLEN              3      7499      7698        30
      WARD               3      7521      7698        30
      MARTIN             3      7654      7698        30
      TURNER             3      7844      7698        30
      JAMES              3      7900      7698        30
   CLARK                 2      7782      7839        10
       MILLER            3      7934      7782        10
14 ROWS SELECTED.
```

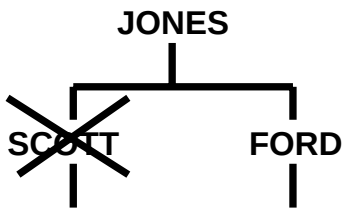
LPAD(' ', 3*LEVEL-3) définit le format d'affichage. “3 * LEVEL-3” est la longueur, n'ayant pas de chaîne de caractère de départ, un espace (' ') est utilisé. Ainsi, cela indique à SQL de prendre cette chaîne pour un espace et complèter à gauche jusqu'à ce que la longueur de la chaîne atteigne celle indiquée par "3* LEVEL-3". Finallement, chaque ligne sera décalée de trois espaces vers la droite par rapport à celle du niveau supérieur.

Elagage des Branches

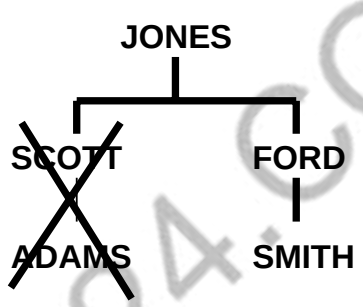
Utilisez la clause WHERE pour éliminer un noeud individuel.

Utilisez la clause CONNECT BY pour éliminer une branche.

WHERE ename != 'SCOTT'



CONNECT BY PRIOR empno = mgr AND ename != 'SCOTT'



Elagage des Branches

Les clauses WHERE et CONNECT BY permettent d'élaguer l'arbre, c'est-à-dire de déterminer les noeuds ou les lignes à afficher. Le prédicat que vous utilisez opère comme une condition booléenne.

Exemple 1

En commençant à la racine, parcourez l'arbre de haut en bas et supprimez l'employé Scott dans le résultat tout en traitant les lignes détail.

```
SQL>SELECT  deptno,empno,ename,job,sal
  2  FROM emp
  3  WHERE ename  != 'SCOTT'
  4  CONNECT BY PRIOR empno = mgr
  5  START WITH mgr IS NULL;
```

Exemple 2

En commençant à la racine, parcourez l'arbre de haut en bas et supprimez l'employé Scott ainsi que toutes les lignes détail qui en dépendent.

```
SQL> SELECT deptno,empno,ename,job,sal
  2  FROM emp
  3  CONNECT BY PRIOR empno = mgr
  4  AND ename != 'SCOTT'
  5  START WITH mgr IS NULL;
```

Tri des Données

Créez un état hiérarchique trié par numéro de département

```
SQL>BREAK ON deptno
SQL>SELECT LEVEL,deptno,empno,ename,job,sal
  2 FROM emp
  3 CONNECT BY PRIOR empno = mgr
  4 START WITH mgr is NULL
  5 ORDER BY deptno;
```


Tri des Données

Il n'est pas conseillé d'utiliser la clause ORDER BY dans les requêtes hiérarchiques, car vous risquez de détruire l'ordre normal implicite. N'utilisez cette clause qu'avec la pseudocolonne LEVEL.

```
   LEVEL     DEPTNO    EMPNO ENAME      JOB
-------- ---------- -------- ---------- ----------
       1         10     7839 KING       PRESIDENT
       2                7782 CLARK      MANAGER
       3                7934 MILLER     CLERK
       2         20     7566 JONES      MANAGER
       3                7788 SCOTT      ANALYST
       4                7876 ADAMS      CLERK
       3                7902 FORD       ANALYST
       4                7369 SMITH      CLERK
       2         30     7698 BLAKE      MANAGER
       3                7499 ALLEN      SALESMAN
       3                7521 WARD       SALESMAN
       3                7654 MARTIN     SALESMAN
       3                7844 TURNER     SALESMAN
       3                7900 JAMES      CLERK

14 rows selected.
```

RESUME

- **Les requêtes hiérarchiques permettent d'afficher une relation hiérarchique existant entre des lignes d'une table.**
- **Il est possible de déterminer la direction et le point de départ du parcours.**
- **L'élagage permet d'éliminer des noeuds ou des branches.**

Présentation des Exercices

- **Différence entre les requêtes hiérarchiques et les requêtes non hiérarchiques**
- **Parcours d'un arbre**
- **Création d'un état indenté à l'aide de la pseudocolonne LEVEL**
- **Elagage de la structure arborescente**
- **Tri des résultats**

Présentation des Exercices

Au cours de ces exercices, vous allez créer des états hiérarchiques.

Exercice 11

1. Examinez les exemples suivants. Sont-ils le résultat d'une requête hiérarchique ? Expliquez pourquoi.

Exemple 1

EMPNO	ENAME	MGR	SAL	DEPTNO
7839	KING		5000	10
7902	FORD	7566	3000	20
7788	SCOTT	7566	3000	20
7566	JONES	7839	2975	20
7698	BLAKE	7839	2850	30
7782	CLARK	7839	2450	10
7499	ALLEN	7698	1600	30

Exemple 2

EMPNO	MGR	DNAME	LOC
7698	7839	SALES	CHICAGO
7782	7839	ACCOUNTING	NEW YORK
7566	7839	RESEARCH	DALLAS

Exemple 3

RANK	ENAME	MGR
1	KING	
2	BLAKE	7839
3	MARTIN	7698
3	ALLEN	7698
3	TURNER	7698
3	JAMES	7698
3	WARD	7698

Exercice 11

2. Créez un état représentant l'organigramme du département de Jones. Imprimez les noms des employés, leur salaire et leur numéro de département.

```
ENAME                SAL     DEPTNO
---------- --------- ----------
JONES             2975         20
FORD              3000         20
SMITH              800         20
SCOTT             3000         20
ADAMS             1100         20
```

3. Créez un état dans lequel figurent les noms de tous les responsables pour lesquels travaille Adams.

```
ENAME
------------
SCOTT
JONES
KING
```

Exercice 11

4. Créez un état représentant la hiérarchie des dirigeants par une indentation. Affichez les noms des employés, le numéro de leur département ainsi que le numéro de leur responsable. Commencez par l'employé ayant le grade le plus élevé.

```
NAME                            MGR     DEPTNO
-------------------- --------- ---------
KING                                        10
   BLAKE                       7839         30
      MARTIN                   7698         30
      ALLEN                    7698         30
      TURNER                   7698         30
      JAMES                    7698         30
      WARD                     7698         30
   CLARK                       7839         10
      MILLER                   7782         10
   JONES                       7839         20
      FORD                     7566         20
         SMITH                 7902         20
      SCOTT                    7566         20
         ADAMS                 7788         20
```

S'il vous reste encore du temps, effectuez l'exercice suivant :

5. Créez l'organigramme d'une société représentant la hiérarchie des dirigeants. Commencez par la personne ayant le grade le plus élevé et excluez tous les employés occupant le poste ANALYST, ainsi que la branche de CLARK.

```
 ENAME          EMPNO        MGR
---------- ---------- ----------
KING             7839
BLAKE            7698       7839
MARTIN           7654       7698
ALLEN            7499       7698
TURNER           7844       7698
JAMES            7900       7698
WARD             7521       7698
JONES            7566       7839
SMITH            7369       7902
ADAMS            7876       7788
```

12

Manipulation des Données

Objectifs

A la fin de ce chapitre, vous saurez :

- **Décrire chaque ordre du LMD**
- **Insérer des lignes dans une table**
- **Mettre à jour des lignes dans une table**
- **Supprimer des lignes d'une table**
- **Contrôler les transactions**

Objectifs

Dans ce chapitre, vous allez apprendre à insérer des lignes dans une table, mettre à jour des lignes existantes et supprimer des lignes d'une table. Vous verrez également comment contrôler des transactions au moyen des ordres COMMIT, SAVEPOINT et ROLLBACK.

Langage de Manipulation des Données

- **Un ordre du LMD est exécuté lorsque :**
  - **Vous ajoutez des lignes à une table**
  - **Vous modifiez des lignes existantes dans une table**
  - **Vous supprimez des lignes d'une table**
- **Une *transaction* est un ensemble d'ordres du LMD formant une unité de travail logique.**

Langage de Manipulation des Données

Le langage de manipulation des données (LMD) joue un rôle central dans SQL. Chaque fois que vous ajoutez, modifiez ou supprimez des données dans la base de données, vous exécutez un ordre du LMD. Un ensemble d'ordres du LMD groupé en une unité de travail logique constitue ce qu'on appelle une *transaction.*

Considérons unebase de données d'opérations bancaires. Quand un client de la banque transfère de l'argent d'un compte d'épargne vers un compte courant, la transaction doit donner lieu à trois opérations différentes : débit du compte d'épargne, crédit du compte courant, et enregistrer la transaction dans le journal des transactions. Oracle Server doit garantir que l'ensemble des trois ordres SQL sont exécutés pour maintenir la balance des comptes équilibrée. Quand quelque chose empêche la bonne exécution de l'un des ordres de la transaction, les autres ordres de la même transaction doivent être annulés.

Ajout d'une Nouvelle Ligne dans une Table

50	DEVELOPMENT	DETROIT

Nouvelle ligne

DEPT

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

"...insérer une nouvelle ligne dans la table DEPT ..."

DEPT

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON
50	DEVELOPMENT	DETROIT

Ajout d'une Nouvelle Ligne dans une Table

L'exemple ci-dessus ajoute un nouveau département dans la table DEPT.

L'Ordre INSERT

- **L'ordre INSERT permet d'ajouter de nouvelles lignes dans une table.**

```
INSERT INTO      table [(column [, column...])]
VALUES           (value [, value...]);
```

- **Cette syntaxe n'insère qu'une seule ligne à la fois.**

Ajout d'une Nouvelle Ligne dans une Table

Pour ajouter de nouvelles lignes dans une table, utilisez l'ordre INSERT.

Syntaxe :

table	nom de la table
column	nom de la colonne dans la table à remplir
value	valeur qui figurera dans la colonne

Remarque : lorsque cet ordre est utilisé avec la clause VALUES, il n'insère qu'une seule ligne à la fois.

Insertion de Nouvelles Lignes

- **Insérez une nouvelle ligne en précisant une valeur pour chaque colonne.**
- **Eventuellement, énumérez les colonnes dans la clause INSERT.**

```
SQL> INSERT INTO    dept (deptno, dname, loc)
  2  VALUES         (50, 'DEVELOPMENT', 'DETROIT');
1 row created.
```

- **Indiquez les valeurs dans l'ordre par défaut des colonnes dans la table.**
- **Placez les valeurs de type caractère et date entre simples quotes.**

Ajout d'une Nouvelle Ligne dans une Table

Etant donné que vous pouvez insérer une nouvelle ligne en précisant une valeur pour chaque colonne, il n'est pas obligatoire de lister les colonnes dans la clause INSERT. Dans ce cas, les valeurs doivent être fournies dans l'ordre par défaut des colonnes dans la table.

```
SQL> DESCRIBE  dept
```

```
Name                           NULL?    Type
------------------------------ -------- ------------
DEPTNO                         NOT NULL NUMBER(2)
DNAME                                   VARCHAR2(14)
LOC                                     VARCHAR2(13)
```

Pour éviter toute ambiguïté, listez les colonnes dans la clause INSERT.
Placez les valeurs de type caractère et date entre simples quotes, mais pas les valeurs numériques.

Insertion de Lignes Contenant des Valeurs NULL

- **Méthode implicite : ne spécifiez pas la colonne dans la liste.**

```
SQL> INSERT INTO    dept (deptno, dname )
  2  VALUES         (60, 'MIS');
1 row created.
```

- **Méthode explicite : spécifiez le mot-clé NULL.**

```
SQL> INSERT INTO    dept
  2  VALUES         (70, 'FINANCE', NULL);
1 row created.
```


Méthodes d'Insertion de Valeurs NULL

Méthode	Description
Implicite	Omettez la colonne dans la liste
Explicite	Spécifiez le mot-clé NULL dans la liste VALUES Vous pouvez aussi spécifier une chaîne vide (' ') dans la liste VALUES, mais uniquement pour les chaînes de caractères et les dates

Assurez-vous que la colonne cible admet les valeurs NULL en vérifiant l'état NULL? au moyen de la commande SQL*Plus DESCRIBE.

Oracle8 Server applique automatiquement toutes les contraintes applicables aux types de données, aux intervalles de données et à l'intégrité. Si une colonne n'est pas explicitement spécifiée dans la liste, elle reçoit automatiquement une valeur NULL dans la nouvelle ligne.

Insertion de Valeurs Spéciales

La fonction SYSDATE renvoie la date et l'heure courantes.

```
SQL> INSERT INTO    emp (empno, ename, job,
  2                 mgr, hiredate, sal, comm,
  3                 deptno)
  4  VALUES         (7196, 'GREEN', 'SALESMAN',
  5                 7782, SYSDATE, 2000, NULL,
  6                 10);
1 row created.
```


Insertion de Valeurs Spéciales au Moyen de Fonctions SQL

Vous pouvez utiliser les pseudo-colonnes pour entrer des valeurs spéciales dans une table.

L'exemple ci-dessus enregistre des informations sur l'employé Green dans la table EMP. Dans la colonne HIREDATE, il insère la date et l'heure courantes issues de la fonction *SYSDATE*.

Lorsque vous insérez des lignes dans une table, vous pouvez aussi utiliser la fonction *USER* qui renvoie le nom de l'utilisateur courant.

Vérification des Ajouts

```
SQL> SELECT  empno, ename, job, hiredate, comm
  2  FROM    emp
  3  WHERE   empno = 7196;
```

```
    EMPNO ENAME      JOB       HIREDATE       COMM
--------- ---------- --------- --------- ---------
     7196 GREEN      SALESMAN  01-DEC-97
```

Insertion de Dates dans un Format Spécifique

- **Ajout d'un nouvel employé.**

```
SQL> INSERT INTO emp
  2  VALUES      (2296,'AROMANO','SALESMAN',7782,
  3               TO_DATE('FEB 3,97', 'MON DD,YY'),
  4               1300, NULL, 10);
1 row created.
```

- **Vérification de l'ajout.**

```
EMPNO ENAME   JOB       MGR   HIREDATE  SAL COMM DEPTNO
----- ------- --------- ----  --------- ---- ----- ------
 2296 AROMANO SALESMAN  7782  03-FEB-97 1300           10
```


Insertion de Valeurs de Date et d'Heure Spécifiques

Pour insérer une date, on utilise généralement le format DD-MON-YY. Par défaut, ce format interprète le siècle comme étant le siècle en cours. Comme la date est systématiquement associée à des informations horaires, une heure par défaut est fournie : minuit (00:00:00).

Si vous devez entrer une date appartenant à un autre siècle, ou indiquer une heure spécifique, il faut utiliser la fonction TO_DATE.

L'exemple ci-dessus enregistre des informations concernant l'employé Aromano dans la table EMP en entrant la date d'embauche du 3 février 1997 dans la colonne HIREDATE.

Si le format RR est utiliser, le siècle peut ne pas être le siècle courant.

Insertion de Valeurs au Moyen de Variables de Substitution

Création d'un script interactif au moyen de paramètres de substitution SQL*Plus.

```
SQL> INSERT INTO    dept (deptno, dname, loc)
  2  VALUES         (&department_id,
  3                 '&department_name', '&location');
```

```
Enter value for department_id: 80
Enter value for department_name: EDUCATION
Enter value for location: ATLANTA

1 row created.
```

12-10

Insertion de Valeurs au Moyen de Variables de Substitution

Grâce aux variables de substitution SQL*Plus, vous pouvez créer un ordre INSERT permettant à l'utilisateur d'ajouter des valeurs en mode interactif.

L'exemple ci-dessus enregistre des informations relatives à un département dans la table DEPT. Il demande à l'utilisateur de saisir le numéro de département, son nom et sa localisation.

Dans le cas de valeurs de type date et caractère, incluez le signe & et le nom de la variable entre simples quotes.

Création d'un Script Contenant des Messages Personnalisés

- **ACCEPT stocke la valeur dans une variable.**
- **PROMPT affiche votre texte.**

```
ACCEPT       department_id PROMPT 'Please enter the -
             department number:'
ACCEPT       department_name PROMPT 'Please enter -
             the department name:'
ACCEPT       location PROMPT 'Please enter the -
             location:'
INSERT INTO  dept (deptno, dname, loc)
VALUES       (&department_id, '&department_name',
             '&location');
```


Création d'un Script de Manipulation de Données

Vous pouvez enregistrer la commande contenant des variables de substitution dans un fichier et exécuter celui-ci. A chaque exécution, des messages vous demanderont d'entrer de nouvelles valeurs. Il est possible de personnaliser ces messages au moyen de la commande SQL*Plus ACCEPT.

L'exemple ci-dessus enregistre des informations relatives à un département de la table DEPT. Par l'intermédiaire de messages personnalisés, il demande à l'utilisateur de saisir le numéro de département, son nom et sa localisation.

```
Please enter the department number: 90
Please enter the department name: PAYROLL
Please enter the location: HOUSTON

1 row created.
```

Lorsque vous placez un paramètre de substitution SQL*Plus dans la commande ACCEPT, ne le faites pas précéder de la perluète (&). Pour pousuivre une commande SQL*Plus sur la ligne suivante, placez un tiret (-) en fin de ligne.

Copie de Lignes d'une Autre Table

- **Ecrivez votre ordre INSERT en spécifiant une sous-interrogation.**

```
SQL> INSERT INTO managers(id, name, salary, hiredate)
  2                 SELECT empno, ename, sal, hiredate
  3                 FROM   emp
  4                 WHERE  job = 'MANAGER';
3 rows created.
```

- **N'utilisez pas la clause VALUES.**
- **Le nombre de colonnes de la clause INSERT doit correspondre à celui de la sous-interrogation.**

Copie de Lignes d'une Autre Table

Vous pouvez utiliser l'ordre INSERT pour ajouter des lignes dans une table lorsque les valeurs proviennent de tables existantes. Dans ce cas, à la place de la clause VALUES, vous utilisez une sous-interrogation.

Syntaxe

```
INSERT INTO table [ column (, column) ]
        subquery;
```

où :	*table*	représente le nom de la table
	column	représente le nom de la colonne dans la table à remplir
	subquery	représente la sous-requête qui ramène les lignes dans la table

Pour plus d'informations, reportez-vous à *Oracle8 Server SQL Reference, Release 8.0,* "SELECT,"section Subqueries.

Le nombre et le type de données des colonnes énumérées dans la clause INSERT doivent correspondre au nombre de valeurs et à leur type de données dans la sous-interrogation.

Modification des Données d'une Table

EMP

EMPNO	ENAME	JOB	...	DEPTNO
7839	KING	PRESIDENT		10
7698	BLAKE	MANAGER		30
7782	CLARK	MANAGER		10
7566	JONES	MANAGER		20
...				

"...modifier une ligne de la table EMP..."

EMP

EMPNO	ENAME	JOB	...	DEPTNO
7839	KING	PRESIDENT		10
7698	BLAKE	MANAGER		30
7782	CLARK	MANAGER		20
7566	JONES	MANAGER		20
...				

12-13

Modification des Données d'une Table

L'exemple ci-dessus fait passer le numéro de département de l'employé Clark de 10 à 20.

L'Ordre UPDATE

- **Utilisez l'ordre UPDATE pour modifier des lignes existantes.**

```
UPDATE      table
SET         column = value [, column = value]
[WHERE      condition];
```

- **Si nécessaire, vous pouvez modifier plusieurs lignes à la fois.**

Modification de Lignes

Vous pouvez modifier des lignes existantes au moyen de l'ordre UPDATE.

Dans la syntaxe présentée ci-dessus :

table	est le nom de la table
column	est le nom de la colonne à modifier dans la table
value	est la nouvelle valeur qui figurera dans la colonne, ou une sous-interrogation fournissant cette valeur
condition	identifie les lignes à mettre à jour. Se compose de noms de colonne, d'expressions, de constantes, de sous-interrogations et d'opérateurs de comparaison.

Vérifiez la mise à jour en interrogeant la table de manière à afficher les lignes modifiées.

Pour plus d'informations, reportez-vous à
Oracle8 Server SQL Reference, Release 8.0, "UPDATE."

Remarque : de manière générale, utilisez la clé primaire pour identifier une ligne unique. L'utilisation d'autres colonnescomme critère de sélection risque de provoquer la modification de plusieurs lignes par inadvertance. Par exemple, il est dangereux d'utiliser le nom d'employé pour désigner une seule ligne de la table EMP car plusieurs personnes peuvent porter le même nom.

Modification de Lignes d'une Table

- La clause WHERE permet de modifier une ou plusieurs lignes spécifiques.

```
SQL> UPDATE   emp
  2  SET      deptno = 20
  3  WHERE    empno = 7782;
1 row updated.
```

- Si vous omettez la clause WHERE, toutes les lignes sont modifiées.

```
SQL> UPDATE   employee
  2  SET      deptno = 20;
14 rows updated.
```


Modification de Lignes

L'ordre UPDATE modifie une ou plusieurs lignes spécifiques lorsque la clause WHERE est spécifiée. Dans l'exemple ci-dessus, l'employé 7782 (Clark) est transféré dans le département 20.

Si vous omettez la clause WHERE, toutes les lignes de la table sont modifiées.

```
SQL> SELECT  ename, deptno
  2  FROM    employee;
```

```
ENAME          DEPTNO
---------- ---------
KING               20
BLAKE              20
CLARK              20
JONES              20
MARTIN             20
ALLEN              20
TURNER             20
...
14 rows selected.
```

Remarque : la table EMPLOYEE contient les mêmes données que la table EMP.

Modification avec une Sous-Interrogation Multi-colonne

Modifier le poste et le n° de département de l'employé 7698 à l'identique de l'employé 7499.

```
SQL> UPDATE  emp
  2  SET     (job, deptno) =
  3                          (SELECT job, deptno
  4                          FROM    emp
  5                          WHERE   empno = 7499)
  6  WHERE   empno = 7698;
1 row updated.
```


Modification de Lignes au moyen d'une Sous-interrogation Multi-colonne

Il est possible d'utiliser des sous-interrogations multi-colonne dans la clause SET d'un ordre UPDATE.

Syntaxe

```
UPDATE   table
SET      (column, column, ...) =
                (SELECT    column, column,
                 FROM      table
                 WHERE     condition)
WHERE    condition;
```

Modification de Lignes en Fonction d'une Autre Table

Utilisez des sous-interrogations dans l'ordre UPDATE pour modifier des lignes d'une table à l'aide de valeurs d'une autre table.

```
SQL> UPDATE   employee
  2  SET      deptno =  (SELECT   deptno
  3                      FROM     emp
  4                      WHERE    empno = 7788)
  5  WHERE    job    =  (SELECT   job
  6                      FROM     emp
  7                      WHERE    empno = 7788);
2 rows updated.
```

12-17

Modification de Lignes en Fonction d'une Autre Table

Vous pouvez modifier les lignes d'une table en utilisant des sous-interrogations dans les ordres UPDATE. Ainsi, l'exemple ci-dessus modifie la table EMPLOYEE en fonction des valeurs de la table EMP. Il remplace le numéro de département de tous les employés occupant le même poste que l'employé 7788 par l'actuel numéro de département de ce dernier.

Ordre UPDATE Synchronisé

Syntaxe

```
UPDATE table1 alias1
SET    column = (SELECT expression
                 FROM table2 alias2
                 WHERE alias1.column = alias2.column);
```

Utilisez une Sous-Interrogation Synchronisée pour mettre à jour les lignes d'une table basée sur des lignes d'une autre table.

Utilisation de Sous-Interrogations Synchronisées

Dans le cas d'un ordre UPDATE, une sous-interrogation synchronisée permet de mettre à jour les lignes d'une table basée sur des lignes d'une autre table.

Exemple

Modifiez la structure de la table EMP en ajoutant une colonne pour stocker le nom du département. Puis remplissez la table à l'aide d'un ordre UPDATE Synchronisé.

```
SQL> ALTER TABLE emp
  2  ADD(dname VARCHAR2(14));
```

```
SQL> UPDATE emp e
  2  SET dname = (SELECT dname
                  FROM   dept d
                  WHERE  e.deptno = d.deptno);
```

Modification de Lignes : Erreur de Contrainte d'Intégrité

```
SQL> UPDATE   emp
  2  SET      deptno = 55
  3  WHERE    deptno = 10;
```

Le numéro de département 55 n'existe pas

```
UPDATE emp
       *
ERROR at line 1:
ORA-02291: integrity constraint (USR.EMP_DEPTNO_FK)
violated - parent key not found
```


Erreur de Contrainte d'Intégrité

Si vous tentez de mettre à jour un enregistrement en utilisant une valeur ne respectant pas une contrainte d'intégrité, vous obtiendrez une erreur.

Dans l'exemple ci-dessus, comme le numéro de département 55 n'existe pas dans la table maître DEPT, il se produit une erreur de violation de clé maître ("parent key") ORA-02291.

Remarque : les contraintes d'intégrité garantissent que les données se conforment à un ensemble de règles pré-définies. Les contraintes d'intégrité seront étudiées en détail dans le Chapitre 14.

Suppression d'une Ligne d'une Table

DEPT

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON
50	DEVELOPMENT	DETROIT
60	MIS	
...		

"…supprime une ligne de la table DEPT…"

DEPT

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON
60	MIS	
...		

Suppression d'une Ligne d'une Table

Sur ce shéma, le département DEVELOPMENT est supprimé de la table DEPT (on suppose qu'aucune contrainte n'est définie dans la table DEPT).

L'Ordre DELETE

Vous pouvez supprimer des lignes d'une table au moyen de l'ordre DELETE.

```
DELETE [FROM]   table
[WHERE          condition];
```


Suppression de Lignes

Vous pouvez supprimer des lignes existantes au moyen de l'ordre DELETE.

Syntaxe :

table	nom de la table
condition	identifie les lignes à supprimer. Se compose de noms de colonnes, d'expressions, de constantes, de sous-requêtes et d'opérateurs de comparaison.

Pour plus d'informations, reportez-vous à
Oracle8 Server SQL Reference, Release 8.0, "DELETE."

Suppression de Lignes d'une Table

- **La clause WHERE permet de supprimer une ou plusieurs lignes spécifiques.**

```
SQL> DELETE FROM    department
  2  WHERE          dname = 'DEVELOPMENT';
1 row deleted.
```

- **Si vous omettez la clause WHERE, toutes les lignes sont supprimées.**

```
SQL> DELETE FROM    department;
4 rows deleted.
```


Suppression de Lignes

Vous pouvez supprimer une ou plusieurs lignes spécifiques en précisant la clause WHERE dans l'ordre DELETE. Dans l'exemple ci-dessus, le département DEVELOPMENT est supprimé de la table DEPARTMENT. Vous pouvez vérifier que la suppression a bien été effectuée en utilisant l'ordre SELECT avec la même clause WHERE que l'ordre DELETE, la requête ne devrait trouver aucune ligne.

```
SQL> SELECT  *
  2  FROM    department
  3  WHERE   dname = 'DEVELOPMENT';
no rows selected.
```

Exemple

Supprimer tous les employés ayant été embauché après le 1 janvier 1997.

```
SQL> DELETE FROM  emp
  2  WHERE        hiredate > TO_DATE('01.01.97', 'DD.MM.YY');
1 row deleted.
```

Si vous omettez la clause WHERE, toutes les lignes de la table seront supprimées. Le second exemple de la diapositive supprime toutes les lignes de la table DEPARTMENT car aucune clause WHERE n'a été spécifiée.

Remarque : la table DEPARTMENT contient les mêmes données que la table DEPT.

Suppression de Lignes en Faisant Référence à une Autre Table

Utilisez des sous-interrogations dans l'ordre DELETE pour supprimer des lignes dont certaines valeurs correspondent à celles d'une autre table.

```
SQL> DELETE FROM    employee
  2  WHERE           deptno =
  3                          (SELECT   deptno
  4                           FROM     dept
  5                           WHERE    dname ='SALES');
6 rows deleted.
```


Suppression de Lignes Associées à des Valeurs d'une Autre Table

Vous pouvez utiliser des sous-interrogations pour supprimer des lignes d'une table associées à des valeurs d'une autre table. L'exemple ci-dessus supprime tous les employés du département 30. La sous-interrogation recherche le numéro du département SALES dans la table DEPT. Elle transmet ensuite ce numéro à la requête principale, qui va supprimer les lignes de la table EMPLOYEE correspondant à ce numéro de département.

Ordre DELETE Synchronisé

Syntaxe

```
SQL> DELETE FROM table1 alias1
  2  WHERE  column operator
  3          (SELECT expression
  3           FROM   table2 alias2
  4           WHERE  alias1.column = alias2.column);
```

Utilisez une Sous-Interrogation Synchronisée pour supprimer uniquement les lignes existant également dans une autre table.

Utilisation de Sous-Interrogations Synchronisées (suite)

Avec un ordre DELETE, une Sous-Interrogation Synchronisée permet de ne supprimer que les lignes existant également dans une autre table.

Exemple

Ecrivez une requête pour rechercher tous les numéros d'employés en double et supprimer les doublons dans la table EMP.

```
SQL> SELECT ename
  2  FROM   emp outer
  3  WHERE  ROWID > (SELECT MIN(ROWID)
  4                  FROM   emp inner
  5                  WHERE  outer.empno = inner.empno)
  6  FOR UPDATE;
```

```
SQL> DELETE FROM emp outer
  2  WHERE  ROWID > (SELECT MIN(ROWID)
                     FROM   emp inner
                     WHERE  outer.empno = inner.empno);
```

Remarque : La clause FOR UPDATE verrouille les lignes ramenées par la requête. Les autres utilisateurs ne peuvent pas verrouiller ou mettre à jour ces mêmes lignes tant que vous n'aurez pas terminé votre transaction.

Suppression de Lignes : Erreur de Contrainte d'Intégrité

```
SQL> DELETE FROM    dept
  2  WHERE          deptno = 10;
```

```
DELETE FROM dept
            *
ERROR at line 1:
ORA-02292: integrity constraint (USR.EMP_DEPTNO_FK)
violated - child record found
```

Vous ne pouvez pas supprimer une ligne qui contient une clé primaire utilisée comme clé étrangère dans une autre table.

Erreur de Contrainte d'Intégrité

Si vous tentez de mettre un enregistrement à jour en utilisant une valeur liée à une contrainte d'intégrité FOREIGN KEY, il se produit une erreur.

Dans l'exemple ci-dessus, on tente de supprimer le numéro de département 10 de la table DEPT, mais il provoque une erreur car ce numéro de département est utilisé comme clé étrangère dans la table EMP. Si l'enregistrement maître que vous tentez de supprimer possède des enregistrements détail, il se produira une erreur de violation d'intégrité pour enregistrement détail existant ("child record found") ORA-02292.

Transactions de Base de Données

Une transaction se compose des éléments suivants :

- **Ensemble d'ordres du LMD effectuant une modification cohérente des données**
- **Un ordre du LDD**
- **Un ordre du LCD**

Transactions de Base de Données

Oracle8 Server garantit la cohérence des données par le biais des transactions. Les transactions vous offrent davantage de souplesse et un meilleur contrôle lors de la modification de données. Elles garantissent la cohérence des données en cas d'échec du processus utilisateur ou de panne du système.

Les transactions consistent en un ensemble d'ordres du LMD qui réalisent une modification cohérente des données. Par exemple, un transfert de fonds entre deux comptes implique de débiter un compte et d'en créditer un autre du même montant. Les deux actions doivent, soit réussir, soit échouer en même temps : un crédit ne peut pas être validé sans le débit correspondant.

Types de Transactions

Type	Description
Langage de manipulation des données (LMD)	Comprend un nombre quelconque d'ordres LMD qu'Oracle Server traite en tant qu'entité unique ou unité de travail logique.
Langage de définition des données (LDD)	Comprend un seul ordre LDD
Langage de contrôle des données (LCD)	Comprend un seul ordre LCD

Transactions de Base de Données

Une transaction :

- **Commence à l'exécution du premier ordre SQL**
- **Se termine par l'un des événements suivants :**
  - **COMMIT ou ROLLBACK**
  - **Exécution d'un ordre LDD ou LCD (validation automatique)**
  - **Fin de session utilisateur**
  - **Panne du système**

12-27

Quand Commence et Finit une Transaction ?

Une transaction commence dès le premier ordre SQL exécutable rencontré et se termine lorsque l'un des événements suivants se produit :

- Un ordre COMMIT ou ROLLBACK est lancé
- Un ordre LDD, tel que CREATE, est lancé
- Un ordre LCD est lancé
- L'utilisateur quitte SQL*Plus
- Il se produit une panne de machine ou du système d'exploitation

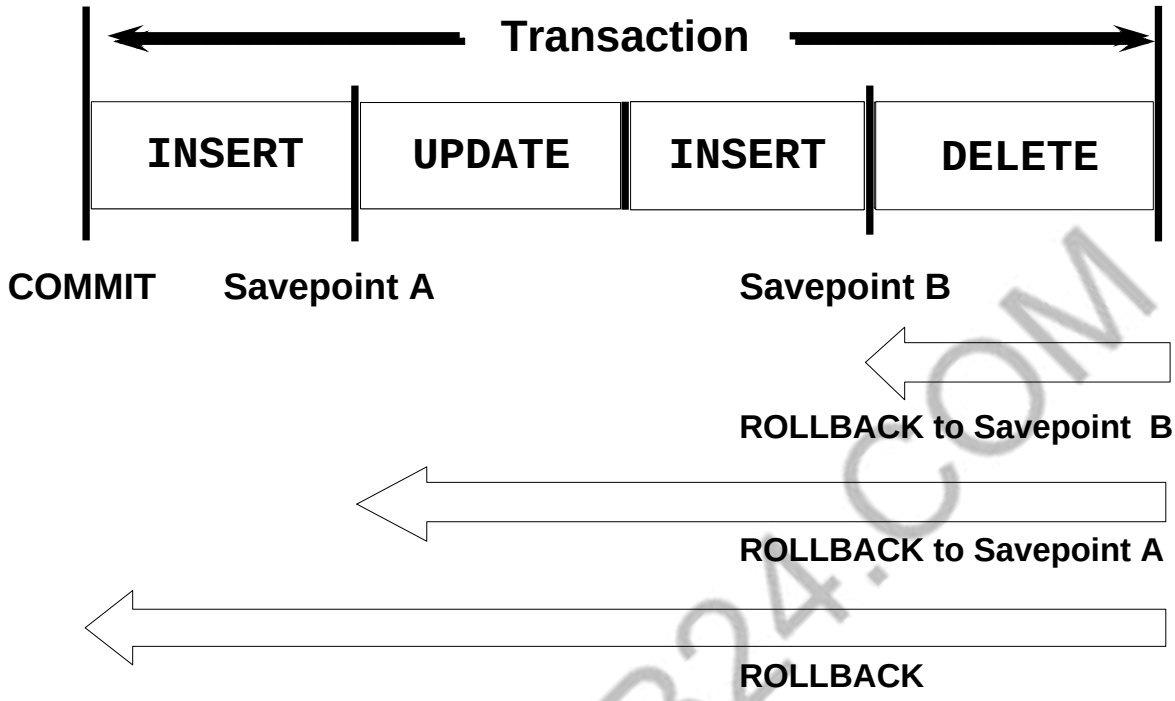
Lorsqu'une transaction prend fin, le prochain ordre SQL exécutable démarrera automatiquement la transaction suivante.

Comme un ordre LDD ou LCD est automatiquement validé, il termine implicitement une transaction.

Avantages des Ordres COMMIT et ROLLBACK

- Garantit la cohérence des données
- Possibilité d'afficher le résultat des modifications avant qu'elles ne soient définitives
- Regroupement logique d'opérations

Contrôle des Transactions



Ordres de Contrôle Explicite des Transactions

Vous avez la possibilité de contrôler la logique des transactions au moyen des ordres COMMIT, SAVEPOINT et ROLLBACK.

Ordre	Description
COMMIT	Met fin à la transaction courante en rendant définitives toutes les modifications de données en instance.
SAVEPOINT *name*	Pose une étiquette dans la transaction courante.
ROLLBACK [TO *SAVEPOINT name*]	ROLLBACK met fin à la transaction courante et rejette toutes les modifications de données en instance. ROLLBACK TO *SAVEPOINT name* annule toutes les modifications jusqu'au savepoint et supprime celui-ci.

Remarque : SAVEPOINT n'est pas un ordre SQL standard ANSI.

Traitement Implicite des Transactions

- **Une validation automatique a lieu dans les situations suivantes :**
  - **Exécution d'un ordre du LDD**
  - **Exécution d'un ordre du LCD**
  - **Sortie normale de SQL*Plus, sans ordre COMMIT ou ROLLBACK explicite**
- **Il se produit un rollback automatique en cas de sortie anormale de SQL*Plus ou d'une panne du système**

Traitement Implicite des Transactions

Etat	Circonstances
Commit automatique	Exécution d'un ordre du LDD ou du LCD Sortie normale de SQL*Plus, sans précision d'un ordre COMMIT ou ROLLBACK explicite
Rollback automatique	Fin anormale de SQL*Plus ou panne du système

Remarque : SQL*Plus propose également la commande AUTOCOMMIT. Cette commande peut être activée (ON) ou désactivée (OFF). Lorsqu'elle est mise sur ON, chaque ordre du LMD est individuellement validé dès son exécution. Vous ne pouvez pas annuler les modifications. Lorsqu'elle est mise sur OFF, vous pouvez préciser explicitement COMMIT. Un ordre COMMIT est également lancé lors de l'exécution d'un ordre du LDD ou de la fin de session SQL*Plus.

Pannes du Système

Lorsqu'une transaction est interrompue par une panne du système, elle est annulée en totalité. Ainsi, SQL*Plus protège l'intégrité des tables en empêchant les modifications intempestives de données et en restaurant les tables dans l'état où elles se trouvaient lors de la dernière validation.

Etat des Données Avant COMMIT ou ROLLBACK

- **Il est possible de restaurer l'état précédent des données.**
- **L'utilisateur courant peut afficher le résultat des opérations du LMD au moyen de l'ordre SELECT.**
- **Les résultats des ordres du LMD exécutés par l'utilisateur courant *ne peuvent pas* être affichés par d'autres utilisateurs.**
- **Les lignes concernées sont *verrouillées*. Aucun autre utilisateur ne peut les modifier.**

Validation des Modifications

Toutes les modifications de données effectuées au cours d'une transaction restent temporaires tant que la transaction n'est pas validée.

Etat des Données Avant COMMIT ou ROLLBACK

- Les opérations de manipulation des données se déroulant principalement dans le buffer de la base de données, il est possible de restaurer l'état précédent des données.
- L'utilisateur courant peut afficher le résultat de ses opérations de manipulation de données en interrogeant les tables.
- Les autres utilisateurs ne peuvent pas voir le résultat des opérations de manipulation de données réalisées par l'utilisateur courant. Oracle met en œuvre un principe de lecture cohérente qui garantit que l'utilisateur voit les données telles qu'elles se présentaient lors de la dernière validation.
- Les lignes concernées par la transaction sont verrouillées ; aucun autre utilisateur ne peut modifier les données qu'elles contiennent.

Etat des Données Après COMMIT

- **Les modifications des données dans la base sont définitives.**
- **L'état précédent des données est irrémédiablement perdu.**
- **Tous les utilisateurs peuvent voir le résultat des modifications.**
- **Les lignes verrouillées sont libérées et peuvent de nouveau être manipulées par d'autres utilisateurs.**
- **Tous les savepoints sont effacés.**

12-32

Validation des Modifications

Pour enregistrer définitivement les modifications en instance, utilisez l'ordre COMMIT. Après l'exécution d'un ordre COMMIT :

- Les modifications de données sont écrites définitivement dans la base de données.
- L'état précédent des données est irrémédiablement perdu.
- Tous les utilisateurs peuvent voir les résultats de la transaction.
- Les lignes qui étaient verrouillées sont libérées et redeviennent accessibles à d'autres utilisateurs pour modification.
- Tous les savepoints sont effacés.

Validation de Données

- **Effectuez les modifications.**

```
SQL> UPDATE   emp
  2  SET      deptno = 10
  3  WHERE    empno = 7782;
1 row updated.
```

- **Validez les modifications.**

```
SQL> COMMIT;
Commit complete.
```


Validation des Modifications

L'exemple ci-dessus met à jour la table EMP en remplaçant par 10 le numéro de département de l'employé 7782 (Clark). Il rend ensuite la modification permanente en spécifiant l'ordre COMMIT.

Exemple

Créer un nouveau département ADVERTISING comportant au moins un employé. Enregistrer définitivement cette modification.

```
SQL> INSERT INTO department(deptno, dname, loc)
  2  VALUES      (50, 'ADVERTISING', 'MIAMI');
1 row created.
```

```
SQL> UPDATE  employee
  2  SET     deptno = 50
  3  WHERE   empno = 7876;
1 row updated.
```

```
SQL> COMMIT;
Commit complete.
```

Etat des Données Après ROLLBACK

L'ordre ROLLBACK rejette toutes les modifications de données en instance.

- **Les modifications sont annulées.**
- **L'état précédent des données est restauré.**
- **Les lignes verrouillées sont libérées.**

```
SQL> DELETE FROM    employee;
14 rows deleted.
SQL> ROLLBACK;
Rollback complete.
```


Annulation des Modifications

Vous pouvez annuler toutes les modifications de données en instance au moyen de l'ordre ROLLBACK. Après l'exécution d'un ordre ROLLBACK :

- Les modifications apportées aux données sont annulées.
- Les données sont retrouvent leur état précédent.
- Les lignes verrouillées sont libérées.

Exemple

En essayant de supprimer un enregistrement de la table TEST, vous pouvez la vider accidentellement. Si cela vous arrive, corrigez l'erreur, spécifiez de nouveau l'ordre en veillant à ce qu'il soit correct, puis validez les modifications.

```
SQL> DELETE FROM  test;
25,000 rows deleted.
SQL> ROLLBACK;
Rollback complete.
SQL> DELETE FROM  test
  2  WHERE       id = 100;
1 row deleted.
SQL> SELECT   *
  2  FROM     test
  3  WHERE    id = 100;
No rows selected.
SQL> COMMIT;
Commit complete.
```

Annulation des Modifications Jusqu'à une Etiquette

- **Posez une étiquette dans la transaction courante au moyen de l'ordre SAVEPOINT.**
- **Annulez la transaction jusqu'à cette étiquette en utilisant l'ordre ROLLBACK TO SAVEPOINT.**

```
SQL> UPDATE...
SQL> SAVEPOINT update_done;
Savepoint created.
SQL> INSERT...
SQL> ROLLBACK TO update_done;
Rollback complete.
```


Annulation des Modifications Jusqu'à une Etiquette de Savepoint

Vous pouvez poser une étiquette dans la transaction courante en utilisant l'ordre SAVEPOINT. Cette procédure permet de diviser une transaction en plusieurs parties plus petites. Il vous est ensuite possible de rejeter les modifications en instance jusqu'à la hauteur de l'étiquette au moyen de l'ordre ROLLBACK TO SAVEPOINT.

Si vous créez un savepoint portant le même nom qu'un savepoint précédemment défini, celui-ci est supprimé.

Rollback au Niveau Ordre

- **Si un seul ordre du LMD dans la transaction échoue, seul cet ordre est annulé.**
- **Oracle8 met en œuvre un savepoint implicite.**
- **Toutes les autres modifications sont conservées.**
- **L'utilisateur doit terminer explicitement les transactions en exécutant un ordre COMMIT ou ROLLBACK.**

Rollback au Niveau Ordre

Il est possible d’annuler une partie d’une transaction au moyen d'un rollback implicite si une erreur d'exécution d'un ordre est détectée. Si un seul ordre du LMD échoue pendant l'exécution d'une transaction, son effet est annulé par un rollback au niveau ordre, mais les modifications effectuées par des ordres du LMD précédents de la transaction ne sont pas annulées. Elles peuvent ensuite être validées ou annulées explicitement par l'utilisateur.

Oracle émet un ordre COMMIT implicite avant et après tout ordre du langage de définition des données (LDD). De cette manière, même si cet ordre ne s'exécute pas correctement, il sera impossible d'annuler l'ordre précédent.

Terminez vos transactions en exécutant explicitement un ordre COMMIT ou ROLLBACK.

Lecture Cohérente

- **La lecture cohérente garantit à tout moment une vue homogène des données.**
- **Les modifications effectuées par un utilisateur n'entrent pas en conflit avec celles d'un autre utilisateur.**
- **Sur les mêmes données, garantit que :**
  - **la lecture ignore les écritures en cours**
  - **l'écriture ne perturbe pas la lecture**

12-37

Lecture Cohérente

On peut accéder à une base de données de deux manières différentes :

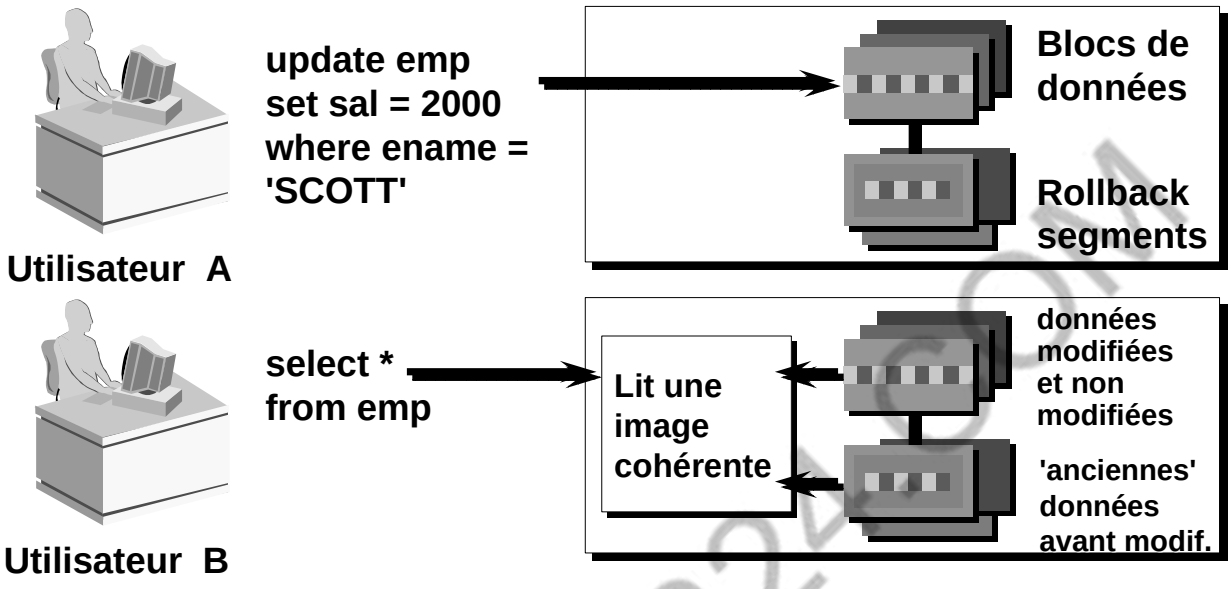
- En lecture (ordre SELECT)
- En écriture (ordres INSERT, UPDATE, DELETE)

Le principe de lecture cohérente a les effets suivants :

- L'utilisateur qui lit la base de données et celui qui y écrit ont une vue cohérente des données
- Les utilisateurs qui accèdent en lecture ne peuvent pas voir les données en cours de modification
- Les utilisateurs qui accèdent en écriture sont sûrs que leurs modifications seront cohérentes
- Les modifications effectuées par un utilisateur ne peuvent pas interrompre ou gêner les modifications en cours d'un autre utilisateur

L'objectif de la lecture cohérente est de garantir que chaque utilisateur voit les données telles qu'elles se présentaient lors de la dernière validation, c'est-à-dire avant le démarrage d'une opération du LMD.

Implémentation de la Lecture Cohérente



Implémentation de la Lecture Cohérente

Le principe de lecture cohérente s'applique automatiquement. Il consiste à conserver une copie partielle de la base de données dans des "rollback segments".

Lorsqu'une opération d'insertion, de mise à jour ou de suppression a lieu dans la base de données, Oracle8 Server copie les données avant leur modification dans un *rollback segment*.

Tous les utilisateurs, excepté celui à l'origine de la modification, continuent à voir les données telles qu'elles se présentaient avant les modifications. En réalité, ils voient un "cliché" des données copiées dans les rollback segments.

Tant que les modifications ne sont pas validées dans la base de données, seul l'auteur des mises à jour voit comment se présentera la base de données une fois modifiée. Tous les autres continuent à voir des copies des "anciennes" données. Ainsi, les utilisateurs ont toujours une vue cohérente des données qui ne sont pas en cours de modification.

Après la validation d'un ordre LMD, les modifications effectuées deviennent visibles par tout utilisateur exécutant un ordre SELECT.

L'espace occupé par les "anciennes" données dans le fichier des rollback segments est libéré et redevient disponible.

Si un ordre ROLLBACK est exécuté, les modifications sont "annulées" et :

- Les données d'origine présentes dans le rollback segment sont récrites dans la table.
- Tous les utilisateurs voient la base de données telle qu'elle était avant le début de la transaction.

Verrouillage

Les verrous Oracle8 :

- **Evitent les risques de destruction des données en cas de transactions simultanées**
- **N'exigent aucune intervention de l'utilisateur**
- **S'appliquent au niveau de restriction le plus bas**
- **Sont actifs durant toute la transaction**
- **Fonctionnent en deux modes de base :**
  - **Exclusif**
  - **Partagé**

12-39

Définition des Verrous

Les verrous sont des mécanismes qui empêchent les destructions dues aux interactions entre des transactions qui accèdent à la même ressource, qu'il s'agisse d'un objet utilisateur (comme des tables ou des lignes), ou d'objets système non visibles par les utilisateurs (tels que des structures de données partagées et des lignes du dictionnaire de données).

Procédure Oracle de Verrouillage des Données

Dans une base de données Oracle, le processus de verrouillage est entièrement automatique et ne requiert aucune intervention de l'utilisateur. Un verrouillage implicite a lieu pour tous les ordres SQL. Le mécanisme de verrouillage d'Oracle s'applique par défaut au niveau de restriction le plus bas possible, afin de permettre un degré élevé de concurrence transactionnelle tout en offrant une intégrité maximale des données. Il est également possible de verrouiller les données manuellement.

Modes de Verrouillage

Oracle utilise deux modes de verrouillage dans une base de données multi-utilisateur.

Mode	Description
exclusif	Interdit le partage d'une ressource. La première transaction qui verrouille une ressource en mode exclusif est la seule à pouvoir modifier cette ressource jusqu'à la levée du verrou.
partagé	Permet le partage d'une ressource. Plusieurs utilisateurs détenant des verrous partagés pour empêcher l'accès simultané en écriture (qui nécessite un verrou exclusif), peuvent partager des données en lecture. Plusieurs transactions peuvent disposer de verrous partagés sur la même ressource

Résumé

Ordre	Description
INSERT	Ajoute une nouvelle ligne dans une table
UPDATE	Modifie des lignes dans une table
DELETE	Supprime des lignes d'une table
COMMIT	Valide toutes les modifications de données en instance
SAVEPOINT	Permet un rollback partiel
ROLLBACK	Annule toutes les modifications de données en instance

Résumé

Pour manipuler les données dans une base de données Oracle, utilisez les ordres INSERT, UPDATE et DELETE. Contrôlez vos modifications au moyen des ordres COMMIT, SAVEPOINT et ROLLBACK.

Oracle8 garantit en permanence une vue cohérente des données.

Le verrouillage peut-être implicite ou explicite.

Présentation des Exercices

- **Insertion de lignes dans une table.**
- **Mise à jour et suppression de lignes dans une table.**
- **Contrôle des transactions.**

Présentation des Exercices

Au cours des exercices qui suivent, vous allez ajouter des lignes dans la table MY_EMPLOYEE, mettre à jour et supprimer des données de cette table, et contrôler vos transactions.

Exercices 12

Insérez des données dans la table MY_EMPLOYEE.

1. Exécutez le script \LABS*lab12_1.sql* pour créer la table MY_EMPLOYEE qui va servir pour cette série d'exercices.
2. Affichez la structure de la table MY_EMPLOYEE pour trouver les noms de colonnes.

```
Name          NULL?     Type
------------ --------- ------
ID            NOT NULL  NUMBER(4)
LAST_NAME               VARCHAR2(25)
FIRST_NAME              VARCHAR2(25)
USERID                  VARCHAR2(8)
SALARY                  NUMBER(9,2)
```

3. Ajoutez **la première ligne** de données du tableau ci-dessous dans la table MY_EMPLOYEE. N'énumérez pas les colonnes dans la clause INSERT.

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	rpatel	795
2	Dancs	Betty	bdancs	860
3	Biri	Ben	bbiri	1100
4	Newman	Chad	cnewman	750
5	Ropeburn	Audry	aropebur	1550

4. Continuez à remplir la table MY_EMPLOYEE en insérant **la seconde ligne** des données ci-dessus. Cette fois, mentionnez explicitement les colonnes dans la clause INSERT.

5. Vérifiez vos ajouts.

```
ID  LAST_NAME   FIRST_NAME USERID SALARY
--- ----------- ---------- ------ ------
  1 Patel       Ralph      rpatel    795
  2 Dancs       Betty      bdancs    860
```

Exercices 12

6. Créez un script nommé *loademp.sql* pour charger les l ignes dans la table MY_EMPLOYEE en mode interactif.
   Demandez à l'utilisateur de saisir le numéro d'employé (ID), le prénom (FIRST_NAME), le nom (LAST_NAME) et le salaire (SALARY) de chaque employé. Concaténez la première lettre du prénom et les sept premières lettres du nom pour former l'ID utilisateur (USERID).
7. Insérez les deux lignes de données suivantes dans la table en exécutant le script que vous avez créé.

8. Vérifiez vos ajouts dans la table.

```
 ID LAST_NAME  FIRST_NAME USERID  SALARY
--- ---------- ---------- ------ -------
  1 Patel      Ralph      rpatel     795
  2 Dancs      Betty      bdancs     860
  3 Biri       Ben        bbiri     1100
  4 Newman     Chad       cnewman    750
```

9. Validez vos ajouts.

Mettez à jour et supprimez des données de la table MY_EMPLOYEE.

10. Remplacez le nom de l'employé 3 par Drexler.
11. Saisissez un salaire de 1000 pour tous les employés ayant un salaire inférieur à 900.
12. Vérifiez vos modifications.

```
LAST_NAME SALARY
--------- ------
Patel       1000
Dancs       1000
Drexler     1100
Newman      1000
```

13. Supprimez Betty Dancs de la table MY_EMPLOYEE.
14. Vérifiez vos modifications.

```
 ID LAST_NAME  FIRST_NAME USERID SALARY
--- ---------- ---------- ------ ------
  1 Patel      Ralph      rpatel   1000
  3 Drexler    Ben        bbiri    1100
  4 Newman     Chad       cnewman  1000
```

Exercices 12

15. Validez toutes les modifications en instance.

    Contrôlez la transaction de données effectuée dans les tables MY_EMPLOYEE.

16. Insérez la dernière ligne des données d'exemple dans la table en exécutant le script que vous avez créé à l'étape 6 (*loademp.sql)*.

17. Vérifiez l'ajout.

```
    ID LAST_NAME FIRST_NAME  USERID   SALARY
---- --------- ----------- -------- ------
     1 Patel     Ralph       rpatel     1000
     3 Drexler   Ben         bbiri      1100
     4 Newman    Chad        cnewman    1000
     5 Ropeburn  Audry       aropebur   1550
```

18. Définissez une étiquette intermédiaire dans le traitement de la transaction.

19. Videz entièrement la table.

20. Vérifiez que la table est vide.

21. Rejetez la dernière opération DELETE sans annuler l'opération INSERT précédente.

22. Vérifiez que la dernière ligne est restée intacte.

```
 ID LAST_NAME FIRST_NAME  USERID   SALARY
 --- --------- ----------- -------- ------
   1 Patel     Ralph       rpatel     1000
   3 Drexler   Ben         bbiri      1100
   4 Newman    Chad        cnewman    1000
   5 Ropeburn  Audry       aropebur   1550
```

23. Rendez les ajouts définitifs.

13

Création et Gestion de Tables

Objectifs

A la fin de ce chapitre, vous saurez :

- **Décrire les principaux objets d'une base de données**
- **Créer des tables**
- **Décrire les différents types de données utilisables pour les définitions de colonne**
- **Modifier la définition des tables**
- **Supprimer, renommer et tronquer une table**

Objectifs

Dans ce chapitre, vous allez étudier les principaux objets d'une base de données et les relations qu'ils entretiennent. Vous apprendrez également à créer, modifier et supprimer des tables.

Objets d'une Base de Données

Objet	Description
Table	Unité de stockage élémentaire, composée de lignes et de colonnes
Vue	Représente de manière logique des sous-groupes de données issues d'une ou plusieurs tables
Séquence	Génère des valeurs de clés primaires
Index	Améliore les performances de certaines requêtes
Synonyme	Permet de donner un autre nom à un objet

Objets d'une Base de Données

Une base de données Oracle8 peut contenir de nombreuses structures de données. Chaque structure doit être prédéfinie lors de la conception de la base de données pour pouvoir être créée durant la phase de construction de la base.

- Table : stocke les données
- Vue : sous-groupes de données issues d'une ou de plusieurs tables
- Séquence : génère des valeurs de clés primaires
- Index : améliore les performances de certaines requêtes
- Synonyme : permet de donner un autre nom à un objet

Structures des Tables Oracle8

- Vous avez la possibilité de créer des tables à tout moment, y compris lorsque la base de données est déjà en cours d'utilisation.
- Vous n'avez pas à spécifier de taille pour les tables. La taille est en fait définie en fonction de l'espace total alloué à la base de données. Il est important, néanmoins, d'estimer l'espace qu'occupera une table avec le temps.
- La structure des tables peut être modifiée en ligne.

Remarque: Il existe d'autres objets dans la base de données mais ils ne sont pas présentés dans ce cours.

Conventions de Dénomination

Un nom :

- **Doit commencer par une lettre**
- **Peut comporter de 1 à 30 caractères**
- **Ne peut contenir que les caractères A à Z, a à z, 0 à 9, _, $, et #**
- **Ne doit pas porter le nom d'un autre objet appartenant au même utilisateur**
- **Ne doit pas être un mot réservé Oracle8 Server**

13-4 ORACLE®

Règles de Dénomination

Nommez les tables et colonnes de votre base de données en suivant les règles de dénomination applicables à tous les objets d'une base de données Oracle8.

- Les noms de table et de colonnes doivent commencer par une lettre et peuvent comprendre de 1 à 30 caractères.
- Les noms ne doivent pas contenir d'autres caractères que les caractères A à Z, a à z, 0 à 9, _ (trait de soulignement), $ et # (caractères autorisés, mais déconseillés).
- Les noms ne doivent pas être utilisés pour nommer plusieurs objets appartenant au même utilisateur Oracle8 Server.
- Ces noms ne doivent pas être des mots réservés Oracle8 Server.

Conseils

- Utilisez des noms parlants.
- Utilisez une dénomination uniforme pour des entités identiques appartenant à des tables différentes. Par exemple, la colonne numéro du département s'appelle DEPTNO dans la table EMP et dans la table DEPT.

Remarque : les majuscules et minuscules ne sont pas différenciées dans les noms. Par exemple, EMP est identique à eMP ou eMp.

Pour plus d'information, reportez-vous à
Oracle8 Server SQL Reference, Release 8.0, "Object Names and Qualifiers."

L'Ordre CREATE TABLE

- **Vous devez posséder :**
  - **Un privilège CREATE TABLE**
  - **Un espace de stockage**

```
CREATE TABLE [schema.]table
             (column datatype [DEFAULT expr],...
```

- **Spécifiez :**
  - **Un nom de table**
  - **Le nom, le type de données et la taille des colonnes.**

13-5 ORACLE®

L'Ordre CREATE TABLE

Créez des tables pour stocker des données en utilisant l'ordre SQL CREATE TABLE. Cet ordre fait partie d'une série d'ordres appartenant au langage de définition des données (LDD) qui sera étudié dans les prochains chapitres. Les ordres du LDD représentent un sous-ensemble des ordres SQL utilisés pour créer, modifier ou supprimer des structures de données Oracle8. Ils agissent directement sur la base de données, et enregistrent des informations dans le dictionnaire de données.

Pour créer une table, l'utilisateur doit disposer du privilège CREATE TABLE et d'un espace de stockage dans laquelle il pourra créer des objets. L'administrateur de base de données utilise des ordres du LCD (langage de contrôle des données) pour accorder des privilèges aux utilisateurs ; ces ordres seront étudiés dans un prochain chapitre.

Syntaxe :

schema	nom du propriétaire
table	nom de la table
DEFAULT *expr*	spécifie une valeur par défaut à utiliser en cas d'omission d'une valeur dans l'ordre INSERT
column	nom de la colonne
datatype	type de données et longueur de la colonne

Pour plus d'informations, reportez-vous à *Oracle8 Server SQL Reference, Release 8.0*, "CREATE TABLE."

Références aux Tables d'un Autre Utilisateur

- **Les tables appartenant à d'autres utilisateurs ne sont pas dans le schéma utilisateur.**
- **Le nom du propriétaire doit précéder le nom de la table.**

Références aux Tables d'un Autre Utilisateur

Un *schéma* est une collection d'objets. Les schémas sont des structures logiques qui font directement référence aux données de la base. Ils peuvent contenir des tables, des vues, des synonymes, des séquences, des procédures stockées, des index, des clusters et des database link.

Lorsqu'une table n'appartient pas à l'utilisateur, son nom doit être préfixé par le nom de son propriétaire.

L'Option DEFAULT

- **Spécifie la valeur par défaut d'une colonne.**

```
… hiredate DATE DEFAULT SYSDATE, …
```

- **Valeurs autorisées : littéraux, expressions et fonctions SQL.**
- **Valeurs non-autorisées : noms d'autres colonnes ou pseudo-colonnes.**
- **Le type de données par défaut doit correspondre à celui de la colonne.**

13-7

L'Option DEFAULT

On peut déclarer une valeur par défaut pour une colonne en utilisant l'option DEFAULT. Cette option empêche l'insertion de valeurs NULL dans une colonne lors de l'ajout d'une ligne qui ne comporte pas de données pour cette colonne. La valeur par défaut peut être un littéral, une expression ou une fonction SQL telle que SYSDATE et USER ; elle ne peut pas être le nom d'une autre colonne ni d'une pseudo-colonne, telle que NEXTVAL ou CURRVAL. L'expression par défaut doit correspondre au type de données de la colonne.

Création de Tables

- **Créer la table.**

```
SQL> CREATE TABLE dept
  2         (deptno  NUMBER(2),
  3          dname   VARCHAR2(14),
  4          loc     VARCHAR2(13));
Table created.
```

- **Vérifier la création de la table.**

```
SQL> DESCRIBE dept
```

```
Name                            NULL?    Type
------------------------------- -------- ---------
DEPTNO                                   NUMBER(2)
DNAME                                    VARCHAR2(14)
LOC                                      VARCHAR2(13)
```

13-8 ORACLE®

Création de Tables

L'exemple ci-dessus crée la table DEPT avec trois colonnes nommées DEPTNO, DNAME et LOC. Il vérifie ensuite la création de la table avec la commande DESCRIBE.

Comme la création d'une table fait appel à un ordre du LDD, une validation automatique a lieu lors de son exécution.

Interrogation du Dictionnaire de Données

- **Décrire les tables appartenant à l'utilisateur.**

```
SQL> SELECT   *
  2  FROM     user_tables;
```

- **Afficher les différents types d'objets appartenant à l'utilisateur.**

```
SQL> SELECT   DISTINCT object_type
  2  FROM     user_objects;
```

- **Afficher les tables, les vues, les synonymes et les séquences appartenant à l'utilisateur.**

```
SQL> SELECT   *
  2  FROM     user_catalog;
```


Interrogation du Dictionnaire de Données

Vous pouvez interroger les tables du dictionnaire de données pour afficher différents objets de la base vous appartenant. Les tables du dictionnaire de données les plus fréquemment utilisées sont les suivantes :

- USER_TABLES
- USER_OBJECTS
- USER_CATALOG

Remarque : CAT est un synonyme de USER_CATALOG. Vous pouvez l'utiliser à la place de USER_CATALOG dans les ordres SQL.

```
SQL> SELECT *
  2  FROM  CAT;
```

Types de Données

Types de données	Description
VARCHAR2(*size*)	Données caractères de longueur variable
CHAR(*size*)	Données caractères de longueur fixe
NUMBER(*p*,*s*)	Numérique de longueur variable
DATE	Valeurs de date et d'heure
LONG	Données caractères de longueur variable, jusqu'à 2 giga-octets
CLOB	Données caractères mono-octet, jusqu'à 4 giga-octets
RAW et LONG RAW	Binaire
BLOB	Binaire, jusqu'à 4 giga-octets
BFILE	Binaire, stocké dans un fichier externe, jusqu'à 4 giga-octets

Types de Données

Type de Données	Description
VARCHAR2(*size*)	Données caractères de longueur variable. Spécifiez obligatoirement une longueur (*size*) maximum. La longueur minimum est 1, la longueur maximum est 4000.
CHAR(*size*)	Données caractères de longueur fixe d'un nombre d'octets égal à *size*. La longueur (*size*) minimum et par défaut est 1, la longueur maximum est 2000.
NUMBER(*p*,*s*)	Nombre de précision *p* et d'échelle *s* ; la précision est le nombre total de chiffres allant de 1 à 38, et l'échelle est le nombre de chiffres à droite de la virgule allant de -84 à 127.
DATE	Valeurs de date et d'heure allant du 1er janvier 4712 av.J.C. au 31 décembre 9999 apr.J.C.
LONG	Données caractères de longueur variable, jusqu'à 2 giga-octets.
CLOB	Données caractères mono-octet, jusqu'à 4 giga-octets.
RAW(*size*)	Données binaires, de longueur *size*. La longueur maximum est 2000. Spécifiez obligatoirement une longueur maximum.
LONG RAW	Données binaires, de longueur variable ; jusqu'à 2 giga-octets.
BLOB	Données binaires, jusqu'à 4 giga-octets.
BFILE	Données binaires, stockées dans un fichier externe ; jusqu'à 4 giga-octets.

Création d'une Table au Moyen d'une Sous-Interrogation

- **Créez une table et insérez des lignes en associant l'ordre CREATE TABLE et l'option AS *subquery*.**

```
CREATE TABLE table
       [column(, column...)]
AS subquery;
```

- **Le nombre de colonnes spécifiées doit correspondre au nombre de colonnes de la sous-interrogation.**
- **Définissez des colonnes avec des noms de colonne et des valeurs par défaut.**

Création d'une Table avec des Lignes d'une Autre Table.

Il existe une deuxième façon de créer une table, qui consiste à utiliser la clause AS *subquery*. Cette méthode permet à la fois de créer la table et d'y insérer des lignes ramenées par une sous-interrogation.

Syntaxe :

table	nom de la table
column	nom de la colonne, valeur par défaut et contrainte d'intégrité
subquery	ordre SELECT qui définit le groupe de lignes à insérer dans la nouvelle table

Conseils

- La table est créée avec les noms de colonnes spécifiés, puis remplie avec les lignes extraites au moyen de l'ordre SELECT.
- La définition des colonnes ne peut contenir que le nom de la colonne et la valeur par défaut.
- Si des colonnes sont spécifiées, leur nombre doit être le même que celui de la sous-interrogation SELECT.
- Si aucune colonne n'est spécifiée, elles seront du même nom que celles de la sous-interrogation.

Création d'une Table au Moyen d'une Sous-Interrogation

```
SQL> CREATE TABLE   dept30
  2  AS
  3    SELECT   empno, ename, sal*12 ANNSAL, hiredate
  4    FROM     emp
  5    WHERE    deptno = 30;
Table created.
```

```
SQL> DESCRIBE dept30
```

```
Name                            NULL?    Type
------------------------------- -------- -----
EMPNO                                    NUMBER(4)
ENAME                                    VARCHAR2(10)
ANNSAL                                   NUMBER
HIREDATE                                 DATE
```


Création d'une Table avec des Lignes d'une Autre Table

L'exemple ci-dessus crée une table, DEPT30, qui contient des informations concernant tous les employés du département 30. Remarquez que les données destinées à la table DEPT30 proviennent de la table EMP.

Vous pouvez vérifier l'existence d'une table de base de données et contrôler les définitions de colonne en utilisant la commande SQL*Plus DESCRIBE.

Indiquez l'alias de colonne lors de la sélection de l'expression.

L'ordre ALTER TABLE

Utilisez l'ordre ALTER TABLE pour :

- Ajouter une colonne
- Modifier une colonne existante
- Définir une valeur par défaut pour une nouvelle colonne

```
ALTER TABLE table
ADD         (column datatype [DEFAULT expr]
            [, column datatype]...);
```

```
ALTER TABLE table
MODIFY      (column datatype [DEFAULT expr]
            [, column datatype]...);
```


Ordre ALTER TABLE

Après avoir créé vos tables, il peut arriver que vous deviez en modifier la structure pour ajouter une colonne oubliée ou que vous décidiez de changer une définition de colonne. Cela est possible grâce à l'ordre ALTER TABLE.

Vous pouvez ajouter des colonnes à une table en utilisant l'ordre ALTER TABLE avec la clause ADD.

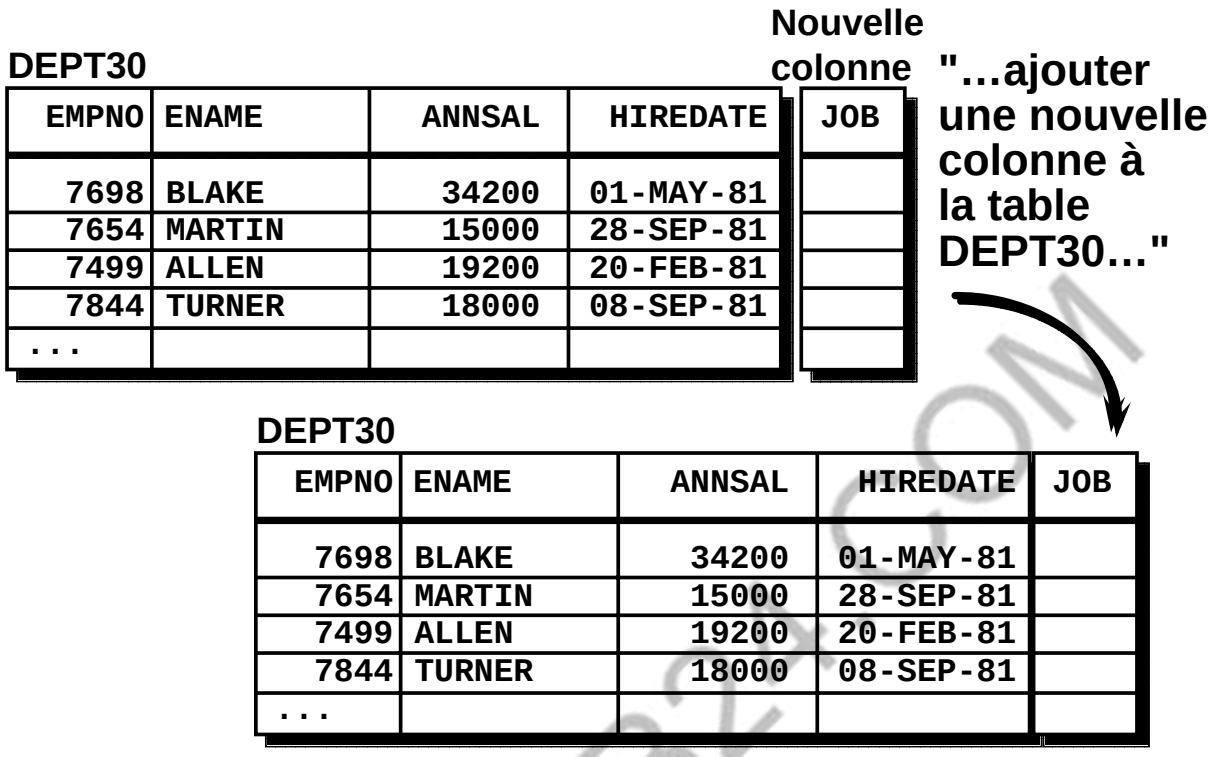
Syntaxe :

table	nom de la table
column	nom de la nouvelle colonne
datatype	type de données et longueur de la nouvelle colonne
DEFAULT *expr*	valeur par défaut de la nouvelle colonne

Vous pouvez modifier des colonnes existantes d'une table au moyen de l'ordre ALTER TABLE avec la clause MODIFY.

Remarque : la diapositive ne présente pas l'ensemble de la syntaxe de l'ordre ALTER TABLE. Celui-ci sera étudié plus en détails dans le chapitre 14.

Ajout de Colonnes



Ajout de Colonnes

L'exemple ci-dessus ajoute la colonne JOB à la table DEPT30. Notez que la nouvelle colonne est placée à la fin de la table.

Ajout de Colonnes

- **Utilisez la clause ADD pour ajouter des colonnes.**

```
SQL> ALTER TABLE dept30
  2  ADD        (job VARCHAR2(9));
Table altered.
```

- **La nouvelle colonne est placée à la fin.**

```
    EMPNO ENAME          ANNSAL HIREDATE  JOB
--------- ---------- --------- --------- ----
     7698 BLAKE           34200 01-MAY-81
     7654 MARTIN          15000 28-SEP-81
     7499 ALLEN           19200 20-FEB-81
     7844 TURNER          18000 08-SEP-81
...
6 rows selected.
```


Conseils pour l'Ajout de Colonnes

- Vous pouvez ajouter ou modifier des colonnes dans une table, mais vous ne pouvez pas en supprimer.
- Vous ne pouvez pas choisir l'emplacement de la nouvelle colonne : elle est systématiquement placée à la fin de la table.

L'exemple ci-dessus ajoute la colonne nommée JOB à la table DEPT30. La colonne JOB devient la dernière colonne de la table.

Remarque : si une table contient déjà des lignes lorsque l'on ajoute une colonne, la nouvelle colonne sera initialisée à NULL pour toutes les lignes sauf si l'on précise une valeur par défaut, dans ce cas pour toutes les lignes de la table, la colonne sera initialisée avec la valeur par défaut.

Modification de Colonnes

- **Vous pouvez modifier le type de données, la taille et la valeur par défaut d'une colonne.**

```
ALTER TABLE  dept30
MODIFY       (ename VARCHAR2(15));
Table altered.
```

- **La modification d'une valeur par défaut ne s'applique qu'aux insertions ultérieures dans la table.**

Modification de Colonnes

Vous pouvez modifier la définition d'une colonne au moyen de l'ordre ALTER TABLE et de la clause MODIFY. Les modifications effectuées peuvent être des modifications du type de données, de taille et de la valeur par défaut.

Conseils

- Vous pouvez augmenter la largeur ou la précision d'une colonne numérique.
- Réduisez la largeur d'une colonne si celle-ci ne contient que des valeurs NULL ou si la table ne contient aucune ligne.
- Modifiez le type de données si la colonne contient des valeurs NULL ou si la table est vide.
- Convertissez une colonne de type CHAR en type de données VARCHAR2 ou inversement, si la colonne contient des valeurs NULL, si vous ne réduisez pas sa taille ou si la table est vide.
- La modification d'une valeur par défaut ne s'appliquera qu'aux insertions ultérieures.

Suppression de Tables

- **La structure et toutes les données de la table sont supprimées.**
- **Tous les index sont supprimés.**
- **La transaction en instance est validée.**
- **Une suppression de table ne peut être annulée.**

```
SQL> DROP TABLE dept30;
Table dropped.
```


Suppression de Tables

L'ordre DROP TABLE supprime la définition d'une table Oracle8. Lorsque vous supprimez une table, la base de données perd toutes les données de la table ainsi que tous les index associés.

Syntaxe

```
DROP TABLE table;
```

où : *table* est le nom de la table

Conseils

- Toutes les données de la table sont supprimées.
- Les vues, synonymes ne sont pas supprimés mais ne sont plus utilisables.
- Toute transaction en instance est validée.
- Seul le créateur de la table ou un utilisateur ayant le privilège DROP ANY TABLE peut supprimer une table.

Une fois exécuté, l'ordre DROP TABLE est irréversible. Oracle8 Server ne demande pas confirmation lorsque vous lancez cet ordre. Si vous êtes le propriétaire de la table ou disposez d'un niveau de privilège élevé, la table sera immédiatement supprimée. Tous les ordres du LDD effectuent une validation qui rend la transaction permanente.

Modification du Nom d'un Objet

- **Pour modifier le nom d'une table, d'une vue, d'une séquence ou d'un synonyme, utilisez l'ordre RENAME.**

```
SQL> RENAME dept TO department;
Table renamed.
```

- **Vous devez être propriétaire de l'objet.**

Modification du Nom d'un Objet

L'ordre RENAME est un ordre du LDD qui permet de renommer une table, une vue, une séquence ou un synonyme.

Syntaxe

```
RENAME    old_name  TO  new_name;
```

où : *old_name* est l'ancien nom de la table, de la vue, de la séquence ou du synonyme

new_name est le nouveau nom de la table, de la vue, de la séquence ou du synonyme

Vous devez être propriétaire de l'objet que vous renommez.

Vider une Table

- **L'ordre TRUNCATE TABLE :**
  - **Supprime toutes les lignes d'une table**
  - **Libère l'espace de stockage utilisé par la table**

```
SQL> TRUNCATE TABLE department;
Table truncated.
```

- **Vous ne pouvez pas annuler un ordre TRUNCATE**
- **Vous pouvez aussi utiliser l'ordre DELETE pour supprimer des lignes**

13-19 ORACLE®

Vider une Table

TRUNCATE TABLE est un autre ordre LDD, qui permet de supprimer toutes les lignes d'une table tout en libérant l'espace utilisé pour stocker cette table. L'ordre TRUNCATE TABLE ne peut être annulé.

Syntaxe

```
TRUNCATE  TABLE   table;
```

où : *table* est le nom de la table

Vous devez être propriétaire de la table ou disposer du privilège système DELETE TABLE pour tronquer une table.

L'ordre DELETE supprime aussi les lignes d'une table, mais il ne libère pas l'espace de stockage.

Ajout de Commentaires à une Table

- **Vous pouvez ajouter des commentaires à une table au moyen de l'ordre COMMENT.**

```
SQL> COMMENT ON TABLE emp
  2  IS 'Employee Information';
Comment created.
```

- **Les commentaires peuvent être affichés grâce aux vues du dictionnaire de données.**
  - **ALL_COL_COMMENTS**
  - **USER_COL_COMMENTS**
  - **ALL_TAB_COMMENTS**
  - **USER_TAB_COMMENTS**

Ajout de Commentaires à une Table

Vous pouvez ajouter un commentaire, comprenant jusqu'à 2000 octets, à une colonne, à une table, à une vue ou à un snapshot au moyen de l'ordre COMMENT. Le commentaire est stocké dans le dictionnaire de données et peut être affiché au moyen de l'une des vues suivantes du dictionnaire de données :

- ALL_COL_COMMENTS
- USER_COL_COMMENTS
- ALL_TAB_COMMENTS
- USER_TAB_COMMENTS

Syntaxe

```
COMMENT ON {TABLE table | COLUMN table.column}
        IS 'text';
```

où :	*table*	est le nom de la table
	column	est le nom de la colonne dans la table
	text	est le texte du commentaire

Vous pouvez supprimer un commentaire d'une base de données en le déclarant comme chaîne de caractère vide (' ').

```
SQL> COMMENT ON TABLE  emp IS '';
```

Résumé

Ordre	Description
CREATE TABLE	Crée une table
ALTER TABLE	Modifie la structure d'une table
DROP TABLE	Supprime les lignes et la structure d'une table
RENAME	Change le nom d'une table, d'une vue, d'une séquence ou d'un synonyme
TRUNCATE	Supprime toutes les lignes d'une table et libère l'espace de stockage de cette table
COMMENT	Ajoute des commentaires à une table ou à une vue

CREATE TABLE

- Crée une table.
- Crée une table sur la base d'une autre table au moyen d'une sous-interrogation.

ALTER TABLE

- Modifie la structure d'une table.
- Change la largeur des colonnes, le type de données des colonnes, ajoute des colonnes.

DROP TABLE

- Supprime les lignes et la structure d'une table.
- Cet ordre ne peut être annulé après exécution.

RENAME

- Renomme une table, une vue, une séquence ou un synonyme.

TRUNCATE

- Supprime toutes les lignes d'une table et libère l'espace occupé par la table.
- L'ordre DELETE supprime seulement les lignes.

COMMENT

- Ajoute un commentaire à une table ou à une colonne.
- Interroger le dictionnaire de données pour afficher un commentaire.

Présentation des Exercices

- Création de nouvelles tables
- Création d'une nouvelle table en utilisant la syntaxe CREATE TABLE AS
- Modification des définitions de colonne
- Vérification de l'existence d'une table
- Ajout de commentaires à une table
- Modification de tables
- Suppression de tables

Présentation des Exercices

Vous allez créer de nouvelles tables avec des contraintes, en utilisant l'ordre CREATE TABLE, puis vérifier l'ajout de cette nouvelle table à la base de données. Pour ce faire, vous devrez placer la syntaxe dans le fichier de commande, puis exécuter celui-ci pour créer la table.

Exercices 13

1. Créez la table DEPARTMENT d'après le tableau suivant. Saisissez la syntaxe dans un script que vous nommerez *p13q1.sql*, puis exécutez ce script pour créer la table. Vérifiez la création de la table.

Colonne	ID	NAME
Type de clé		
Null/Unique		
Table FK		
Colonne FK		
Type de données	Number	Varchar2
Longueur	2	25

```
Name        NULL?    Type
----------- -------- ----
ID                   NUMBER(2)
NAME                 VARCHAR2(25)
```

2. Remplissez la table DEPARTMENT avec les données de la table DEPT. N'utilisez que les colonnes dont vous avez besoin.
3. Créez la table EMPLOYEE d'après le tableau suivant. Saisissez la syntaxe dans un script que vous nommerez *p13q3.sql*, puis exécutez ce script pour créer la table. Vérifiez la création de la table.

Colonne	ID	LAST_NAME	FIRST_NAME	DEPT_ID
Type de clé				
Null/Unique				
Table FK				
Colonne FK				
Type de données	Number	Varchar2	Varchar2	Number
Longueur	7	25	25	2

```
Name           NULL?    Type
-------------- -------- ----
ID                      NUMBER(7)
LAST_NAME               VARCHAR2(25)
FIRST_NAME              VARCHAR2(25)
DEPT_ID                 NUMBER(2)
```

Exercices 13

4. Modifiez la table EMPLOYEE pour pouvoir allonger les noms de famille des employés. Vérifiez votre modification.

```
Name          NULL?    Type
------------- -------- ----
ID                     NUMBER(7)
LAST_NAME              VARCHAR2(50)
FIRST_NAME             VARCHAR2(25)
DEPT_ID                NUMBER(2)
```

5. Vérifiez que les tables DEPARTMENT et EMPLOYEE sont bien enregistrées dans le dictionnaire de données. (*Utiliser* : USER_TABLES)

```
TABLE_NAME
-------------------------
DEPARTMENT
EMPLOYEE
```

6. Créez la table EMPLOYEE2 sur la base de la structure et des données de la table EMP mais n'incluez que les colonnes EMPNO, ENAME et DEPTNO. Nommez les colonnes de votre nouvelle table respectivement ID, LAST_NAME et DEPT_ID,.
7. Supprimez la table EMPLOYEE.
8. Renommez la table EMPLOYEE2 en EMPLOYEE.
9. Ajoutez un commentaire aux définitions de tables DEPARTMENT et EMPLOYEE pour décrire chaque table. Vérifiez vos ajouts dans le dictionnaire de données.

14

Les Contraintes

Objectifs

A la fin de ce chapitre, vous saurez :

- **Définir les contraintes**
- **Créer des contraintes et les maintenir**

Objectifs

Dans ce chapitre, vous allez apprendre à implémenter des règles de gestion en utilisant des contraintes d'intégrité.

Les Contraintes

- **Les contraintes contrôlent des règles de gestion au niveau d'une table.**
- **Les contraintes empêchent la suppression d'une table lorsqu'il existe des dépendances.**
- **Types de contraintes valides dans Oracle :**
  - **NOT NULL**
  - **UNIQUE**
  - **PRIMARY KEY**
  - **FOREIGN KEY**
  - **CHECK**

Contraintes

Oracle8 Server fait appel à des *contraintes* pour empêcher l'entrée de données incorrectes dans des tables.

Vous pouvez utiliser des contraintes pour :

- Appliquer des règles au niveau d'une table chaque fois qu'une ligne est insérée, mise à jour ou supprimée dans cette table. La contrainte doit être satisfaite pour que l'opération réussisse.
- Empêcher la suppression d'une table si il y a des dépendances avec d'autres tables.
- Fournir des règles pour des outils Oracle comme Developer 2000.

Contraintes d'Intégrité des Données

Contrainte	Description
NOT NULL	Spécifie que cette colonne ne doit pas contenir de valeur null
UNIQUE	Spécifie une colonne ou une combinaison de colonnes dont les valeurs doivent être uniques pour toutesles lignes de la table
PRIMARY KEY	Identifie chaque ligne de la table de manière unique
FOREIGN KEY	Etablit et contrôle une relation de clé étrangère entre la colonne et une colonne de la table référencée
CHECK	Spécifie une condition qui doit être vraie

Pour plus d' informations, reportez-vous à *Oracle8 Server SQL Reference, Release 8.0*, "CONSTRAINT Clause".

Conventions Applicables aux Contraintes

- **Si vous ne nommez pas une contrainte, Oracle8 Server créera un nom au format SYS_C*n*.**
- **Vous pouvez créer une contrainte :**
  - **En même temps que la création de la table**
  - **Une fois que la table est créée**
- **Définissez une contrainte peut être définie au niveau table ou colonne.**
- **Consulter le dictionnaire de données pour retrouver une contrainte.**

Conventions Applicables aux Contraintes

Toutes les contraintes sont stockées dans le dictionnaire de données. Elles seront très faciles à manipuler si vous leur donnez un nom parlant. Les noms de contraintes sont soumis aux conventions de dénomination des objets standard. Si vous ne nommez pas une contrainte, Oracle8 génère un nom dont le format est SYS_C*n*, où *n* est un entier permettant de créer un nom de contrainte unique.

Il est possible de définir les contraintes au moment de la création de la table ou plus tard.

Vous pouvez retrouver les contraintes définies pour une table spécifique en consultant la table USER_CONSTRAINTS du dictionnaire de données.

Les Contraintes

```
CREATE TABLE [schema.]table
            (column datatype [DEFAULT expr]
             [column_constraint],
             …
             [table_constraint]);
```

```
CREATE TABLE emp(
             empno  NUMBER(4),
             ename  VARCHAR2(10),
             …
             deptno  NUMBER(2) NOT NULL,
             CONSTRAINT emp_empno_pk
                          PRIMARY KEY (EMPNO));
```


Les Contraintes

La diapositive montre la syntaxe utilisée pour définir des contraintes pendant la création d'une table.

Dans cette syntaxe :

schema	nom du propriétaire de la table
table	nom de la table
DEFAULT *expr*	valeur par défaut à utiliser si une valeur est omise dans l'ordre INSERT
column	nom de la colonne
datatype	type de données et longueur de la colonne
column_constraint	contrainte d'intégrité incluse dans la définition de la colonne
table_constraint	contrainte d'intégrité incluse dans la définition de la table

Pour plus d'informations, reportez-vous à
Oracle8 Server SQL Reference, Release 8.0, "CREATE TABLE."

Les Contraintes

- **Contrainte au niveau colonne**

```
column [CONSTRAINT constraint_name] constraint_type,
```

- **Contrainte au niveau table**

```
column,...
  [CONSTRAINT constraint_name] constraint_type
  (column, ...),
```


Les Contraintes (suite)

En général, on crée les contraintes en même temps que la table. Il est cependant possible d'ajouter des contraintesou d'en désactiver dans une table déjà créée.

Il existe deux niveaux de contraintes :

Niveau de contrainte	Description
Colonne	Référence une seule colonne et se définit dans la spécification de cette colonne ; applicable à n'importe quel type de contrainte d'intégrité
Table	Référence une ou plusieurs colonnes et se définit indépendemment de la définition des colonnes ; applicable à toutes les contraintes sauf NOT NULL

Syntaxe :

constraint_name	nom de la contrainte
constraint_type	type de contrainte

La Contrainte NOT NULL

Interdit la présence de valeurs NULL dans la colonne

EMP

EMPNO	ENAME	JOB	...	COMM	DEPTNO
7839	KING	PRESIDENT			10
7698	BLAKE	MANAGER			30
7782	CLARK	MANAGER			10
7566	JONES	MANAGER			20
...					

Contrainte NOT NULL (aucune ligne ne peut avoir de valeur NULL dans cette colonne)

Absence de contrainte NOT NULL (toute ligne peut avoir une valeur NULL dans cette colonne)

Contrainte NOT NULL

14-7

La contrainte NOT NULL

La contrainte NOT NULL interdit la présence de valeurs NULL dans la colonne à laquelle elle s'applique. Par défaut, les colonnes qui ne sont pas associées à la contrainte NOT NULL peuvent contenir des valeurs NULL.

La Contrainte NOT NULL

Se définit au niveau colonne

```
SQL> CREATE TABLE emp(
  2      empno    NUMBER(4),
  3      ename    VARCHAR2(10) NOT NULL,
  4      job      VARCHAR2(9),
  5      mgr      NUMBER(4),
  6      hiredate DATE,
  7      sal      NUMBER(7,2),
  8      comm     NUMBER(7,2),
  9      deptno   NUMBER(2) NOT NULL);
```

14-8

La contrainte NOT NULL (suite)

La contrainte NOT NULL ne peut être définie qu'au niveau colonne, pas au niveau table.

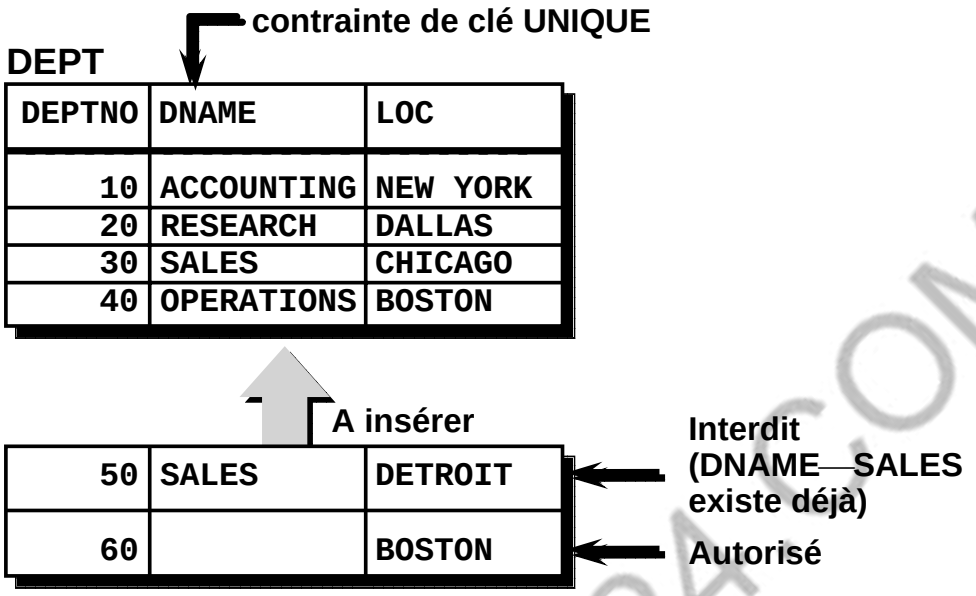
L'exemple ci-dessus applique la contrainte NOT NULL aux colonnes ENAME et DEPTNO de la table EMP. Comme ces contraintes ne sont pas nommées, Oracle8 Server créera des noms pour elles.

Vous pouvez spécifier le nom de la contrainte pendant que vous la définissez.

```
… deptno NUMBER(2)
     CONSTRAINT emp_deptno_nn NOT NULL;
```

Remarque : toutes les contraintes citées en exemple dans ce chapitre ne sont pas nécessairement présentes dans les tables fournies pour les exercices de ce cours. Le cas échéant, il est possible de les ajouter aux tables.

La Contrainte de Clé UNIQUE



La Contrainte de Clé UNIQUE

Une contrainte d'intégrité de type clé UNIQUE exige que chaque valeur dans une colonne ou dans un ensemble de colonnes (la clé) soit unique, c'est-à-dire qu'elle n'existe pas dans plusieurs lignes pour la colonne ou l'ensemble de colonnes spécifiés. La colonne ou l'ensemble de colonne indiqués dans la définition de la contrainte UNIQUE constituent la *clé unique*. Si la clé comprend plusieurs colonnes, le groupe de colonnes est appelé *clé unique composée.*

Les contraintes UNIQUE autorisent la saisie de valeurs NULL, à moins que vous ne définissiez également des contraintes NOT NULL sur les mêmes colonnes. En fait, lorsque des colonnes n'ont pas de contrainte NOT NULL, un nombre quelconque de lignes peuvent contenir des valeurs NULL puisque celles-ci n'équivalent à rien. Une valeur NULL dans une colonne (ou dans toutes les colonnes appartenant à une clé unique composée) satisfait toujours une contrainte de clé UNIQUE.

Remarque : étant donné le mécanisme de recherche pour les contraintes de clé UNIQUE sur plusieurs colonnes, il ne peut y avoir de valeurs identiques dans les colonnes non NULL d'une contrainte de clé unique composée partiellement NULL.

La Contrainte de Clé UNIQUE

Se définit au niveau table ou colonne

```
SQL> CREATE TABLE dept(
  2      deptno   NUMBER(2),
  3      dname    VARCHAR2(14),
  4      loc      VARCHAR2(13),
  5      CONSTRAINT dept_dname_uk UNIQUE(dname));
```

14-10

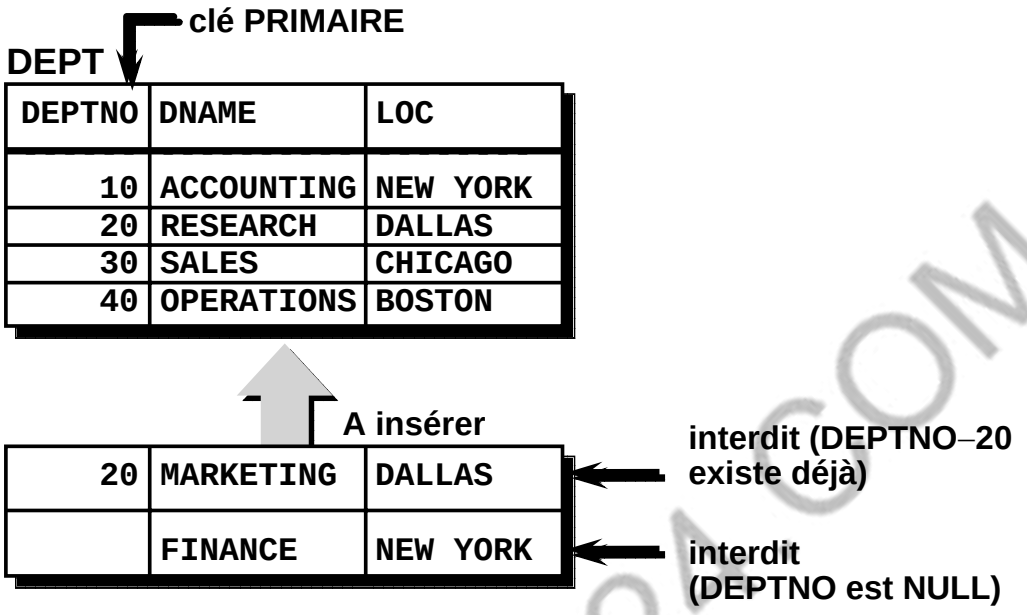
La Contrainte de Clé UNIQUE (suite)

Les contraintes de clé UNIQUE se définissent au niveau colonne ou table. Une clé unique composée est créée à l'aide de définitions de niveau table.

L'exemple de la diapositive applique une contrainte de clé UNIQUE à la colonne DNAME de la table DEPT. Le nom de cette contrainte est DEPT_DNAME_UK.

Remarque : Oracle Server contrôle la contrainte UNIQUE en créant implicitement un index unique sur la clé.

La Contrainte PRIMARY KEY



La Contrainte PRIMARY KEY

Une contrainte PRIMARY KEY crée une clé primaire pour la table. Une seule clé primaire peut être créée par table. La contrainte PRIMARY KEY est une colonne ou un ensemble de colonnes qui identifie de manière unique chaque ligne d'une table. Elle établit une règle d'unicité de la colonne ou d'une combinaison de colonnes et garantit qu'aucune colonne faisant partie de la clé primaire ne contient de valeur NULL.

La Contrainte PRIMARY KEY

Se définit au niveau table ou colonne

```
SQL> CREATE TABLE   dept(
  2      deptno     NUMBER(2),
  3      dname      VARCHAR2(14),
  4      loc        VARCHAR2(13),
  5      CONSTRAINT dept_dname_uk UNIQUE (dname),
  6      CONSTRAINT dept_deptno_pk PRIMARY KEY(deptno));
```

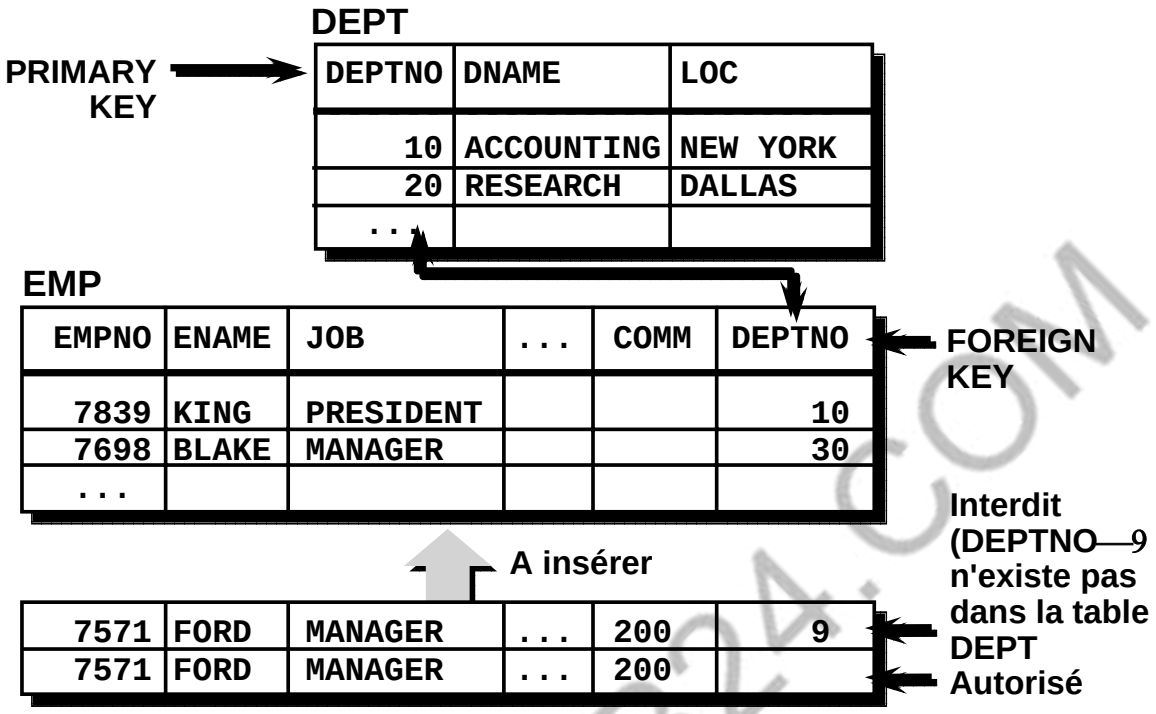

La Contrainte PRIMARY KEY (suite)

Les contraintes PRIMARY KEY peuvent être définies au niveau colonne ou table. Une clé primaire composée se crée à l'aide de la définition de niveau table.

L'exemple de la diapositive définit une contrainte PRIMARY KEY pour la colonne DEPTNO de la table DEPT. Le nom de la contrainte est DEPT_DEPTNO_PK.

Remarque : un index UNIQUE est automatiquement créé pour une colonne PRIMARY KEY.

La Contrainte FOREIGN KEY



La Contrainte FOREIGN KEY

La contrainte FOREIGN KEY, ou contrainte d'intégrité référentielle, désigne une colonne ou une combinaison de colonnes comme étant une clé étrangère et établit une relation avec une clé primaire ou une clé unique de la même table ou d'une table différente. Dans l'exemple ci-dessus, la colonne DEPTNO a été définie comme clé étrangère dans la table EMP (table dépendante ou enfant) ; elle fait référence à la colonne DEPTNO de la table DEPT (table de référence ou parent).

Une valeur de clé étrangère doit obligatoirement correspondre à une valeur existante de la table maître ou être NULL.

Les clés étrangères sont basées sur des valeurs de données et sont des pointeurs purement logiques et non physiques.

La Contrainte FOREIGN KEY

Se définit au niveau table ou colonne

```
SQL> CREATE TABLE emp(
  2      empno    NUMBER(4),
  3      ename    VARCHAR2(10) NOT NULL,
  4      job      VARCHAR2(9),
  5      mgr      NUMBER(4),
  6      hiredate DATE,
  7      sal      NUMBER(7,2),
  8      comm     NUMBER(7,2),
  9      deptno   NUMBER(2) NOT NULL,
 10      CONSTRAINT emp_deptno_fk FOREIGN KEY (deptno)
 11                REFERENCES dept (deptno));
```

14-14

La Contrainte FOREIGN KEY (suite)

Les contraintes FOREIGN KEY peuvent être définies au niveau table ou colonne. Une clé étrangère composée se crée au moyen de la définition de niveau table.

L'exemple ci-dessus définit une contrainte FOREIGN KEY pour la colonne DEPTNO de la table EMP. Le nom de la contrainte est EMP_DEPTNO_FK.

Mots-clés Associés à la Contrainte FOREIGN KEY

- **FOREIGN KEY**
  - **Définit la colonne dans la table détail dans une contrainte de niveau table**
- **REFERENCES**
  - **Identifie la table et la colonne de la table maître**
- **ON DELETE CASCADE**
  - **Autorise la suppression d'une ligne dans la table maître et des lignes dépendantes dans la table détail**

14-15

La Contrainte FOREIGN KEY (suite)

La clé étrangère est définie dans la table détail et la table contenant la colonne référencée est la table maître. Pour définir la clé étrangère, on utilise une combinaison des mots-clés suivants :

- FOREIGN KEY définit une colonne de la table détail dans une contrainte de niveau table.
- REFERENCES identifie la table et la colonne dans la table maître.
- ON DELETE CASCADE indique que si une ligne est supprimée de la table maître, les lignes dépendantes de la table détail seront également supprimées.

En l'absence de l'option ON DELETE CASCADE, la ligne de la table maître ne peut pas être supprimée si elle est référencée dans la table détail.

La Contrainte CHECK

- **Définit une condition que chaque ligne doit obligatoirement satisfaire**
- **Expressions interdites :**
  - **Références aux pseudo-colonnes CURRVAL, NEXTVAL, LEVEL et ROWNUM**
  - **Appels aux fonctions SYSDATE, UID, USER et USERENV**
  - **Requêtes faisant référence à d'autres valeurs dans d'autres lignes**

```
..., deptno  NUMBER(2),
     CONSTRAINT emp_deptno_ck
           CHECK (DEPTNO BETWEEN 10 AND 99),...
```

14-16

La contrainte CHECK

La contrainte CHECK définit une condition que chaque ligne doit obligatoirement satisfaire. La condition peut utiliser les mêmes constructions que les conditions d'une requête, aux exceptions près suivantes :

- Références aux pseudo-colonnes CURRVAL, NEXTVAL, LEVEL et ROWNUM
- Appels aux fonctions SYSDATE, UID, USER et USERENV
- Requêtes faisant référence à d'autres valeurs dans d'autres lignes

Plusieurs contraintes CHECK peuvent être définies sur la même colonne. Le nombre de contraintes CHECK pouvant être associées à une colonne est illimité.

Les contraintes CHECK peuvent être définies au niveau colonne ouau niveau table.

Ajout d'une Contrainte

```
ALTER TABLE table
ADD [CONSTRAINT constraint] type (column);
```

- **Vous pouvez ajouter ou supprimer une contrainte, mais pas la modifier**
- **Vous pouvez activer ou désactiver des contraintes**
- **Pour ajouter une contrainte NOT NULL, utilisez la clause MODIFY**

Ajout d'une Contrainte

Vous pouvez ajouter des contraintes dans une table existante en utilisant l'ordre ALTER TABLE avec la clause ADD.

Syntaxe :

table	nom de la table
constraint	nom de la contrainte
type	type de contrainte
column	nom de la colonne concernée par la contrainte

Il est recommandé de préciser le nom de la contrainte, même si ce n'est pas obligatoire. A défaut, le système générera lui-même des noms de contraintes.

Conseils

- Vous pouvez ajouter, supprimer, activer ou désactiver une contrainte, mais il est impossible d'en modifier la structure.
- Vous pouvez ajouter une contrainte NOT NULL à une colonne existante en utilisant la clause MODIFY à la place de la clause ADD de l'ordre ALTER TABLE.

Remarque : Vous pouvez définir une colonne NOT NULL seulement si la table est vide, ou bien si vous spécifiez une valeur par défaut, celle-ci sera alors automatiquement attribuée à toutes les lignes existantes de la table.

Ajout d'une Contrainte

Ajouter une contrainte FOREIGN KEY à la table EMP précisant qu'un manager doit déjà exister dans la table EMP en tant qu'employé valide.

```
SQL> ALTER TABLE    emp
  2  ADD CONSTRAINT emp_mgr_fk
  3          FOREIGN KEY(mgr) REFERENCES emp(empno);
Table altered.
```

14-18

Ajout d'une Contrainte

L'exemple ci-dessus crée une contrainte FOREIGN KEY dans la table EMP. Cette contrainte vérifie qu'un manager existe en tant qu'employé valide dans la table EMP.

Suppression d'une Contrainte

- **Supprimer de la table EMP la contrainte concernant le manager.**

```
SQL> ALTER TABLE          emp
  2  DROP CONSTRAINTemp_mgr_fk;
Table altered.
```

- **Supprimer la contrainte PRIMARY KEY de la table DEPT, ainsi que la contrainte FOREIGN KEY associée définie sur la colonne EMP.DEPTNO.**

```
SQL> ALTER TABLE          dept
  2  DROP PRIMARY KEY CASCADE;
Table altered.
```

14-19

Suppression d'une Contrainte

Si vous souhaitez supprimer une contrainte, vous pouvez en retrouver le nom dans les vues du dictionnaire de données USER_CONSTRAINTS et USER_CONS_COLUMNS. Utilisez ensuite l'ordre ALTER TABLE avec la clause DROP. L'option CASCADE de la clause DROP provoque également la suppression de toutes les contraintes associées.

Syntaxe

```
ALTER TABLE table
DROP  PRIMARY KEY | UNIQUE (column) |
      CONSTRAINT   constraint  [CASCADE];
```

où :	*table*	représente le nom de la table
	column	représente le nom de la colonne concernée par la contrainte
	constraint	représente le nom de la contrainte

Lorsque vous supprimez une contrainte d'intégrité, elle n'est plus contrôlée par Oracle8 Server et n’existe plus dans le dictionnaire de données.

Désactivation de Contraintes

- **Pour désactiver une contrainte d'intégrité, utiliser la clause DISABLE de l'ordre ALTER TABLE.**
- **Pour désactiver les contraintes d'intégrité dépendantes, ajouter l'option CASCADE.**

```
SQL> ALTER TABLE           emp
  2  DISABLE CONSTRAINT    emp_empno_pk CASCADE;
Table altered.
```


Désactivation d'une Contrainte

Vous pouvez désactiver une contrainte sans la supprimer (ce qui évite d'avoir à la recréer) en utilisant l'ordre ALTER TABLE avec la clause DISABLE.

Syntaxe

```
ALTER TABLE   table
DISABLE  CONSTRAINT constraint [CASCADE];
```

où : *table* représente le nom de la table

constraint représente le nom de la contrainte

Conseils

- Vous pouvez utiliser la clause DISABLE aussi bien dans l'ordre CREATE TABLE que ALTER TABLE.
- La clause CASCADE désactive des contraintes d'intégrité dépendantes.

Activation de Contraintes

- **Pour activer une contrainte d'intégrité actuellement désactivée dans la définition de la table, utiliser la clause ENABLE**

```
SQL> ALTER TABLE           emp
  2  ENABLE CONSTRAINT     emp_empno_pk;
Table altered.
```

- **Si vous activez une contrainte UNIQUE ou PRIMARY KEY, un index correspondant est automatiquement créé.**

14-21

Activation d'une Contrainte

Vous pouvez activer une contrainte, sans la supprimer et la recréer, en utilisant l'ordre ALTER TABLE avec la clause ENABLE.

Syntaxe

```
ALTER TABLE   table
ENABLE  CONSTRAINT constraint;
```

où : *table* représente le nom de la table

constraint représente le nom de la contrainte

Conseils

- Si vous activez une contrainte, celle-ci s'applique à toutes les données de la table. Les données de la table doivent donc toutes respecter la contrainte.
- Si vous activez une contrainte UNIQUE ou PRIMARY KEY, un index unique est automatiquement créé.
- Vous pouvez utiliser la clause ENABLE aussi bien dans l'ordre CREATE TABLE que ALTER TABLE.

Vérification des Contraintes

Pour afficher les définitions et noms de toutes les contraintes, interrogez la table USER_CONSTRAINTS.

```
SQL>  SELECT  constraint_name, constraint_type,
  2           search_condition
  3   FROM    user_constraints
  4   WHERE   table_name = 'EMP';
```

```
CONSTRAINT_NAME           C SEARCH_CONDITION
------------------------- - -------------------------
SYS_C00674                C EMPNO IS NOT NULL
SYS_C00675                C DEPTNO IS NOT NULL
EMP_EMPNO_PK              P
...
```


Vérification des Contraintes

Une fois que vous avez créé une table, vous pouvez en contrôler l'existence en exécutant la commande DESCRIBE. Cependant, la seule contrainte que vous pouvez vérifier de cette manière est NOT NULL. Si vous voulez afficher toutes les contraintes de votre table, interrogez la table USER_CONSTRAINTS.

L'exemple ci-dessus affiche toutes les contraintes de la table EMP.

Remarque : les contraintes qui ne sont pas explicitement nommées par le propriétaire de la table se voient attribuer un nom par le système. Dans la colonne décrivant le type de contrainte, la lettre C signifie CHECK, P représente PRIMARY KEY, R intégrité référentielle et U,UNIQUE. A noter que la contrainte NULL est en réalité une contrainte CHECK.

Affichage des Colonnes Associées aux Contraintes

Affichez les colonnes associées aux noms de contraintes au moyen de la vue USER_CONS_COLUMNS view

```
SQL> SELECT   constraint_name, column_name
  2  FROM     user_cons_columns
  3  WHERE    table_name = 'EMP';
```

```
CONSTRAINT_NAME            COLUMN_NAME
-------------------------- --------------------
EMP_DEPTNO_FK              DEPTNO
EMP_EMPNO_PK               EMPNO
EMP_MGR_FK                 MGR
SYS_C00674                 EMPNO
SYS_C00675                 DEPTNO
```

14-23

Affichage des Contraintes

Vous pouvez afficher le nom des colonnes associées à des contraintes en interrogeant la vue USER_CONS_COLUMNS du dictionnaire de données. Cette vue est particulièrement utile pour les contraintes ayant un nom attribué par le système.

Résumé

- **Vous pouvez créer des contraintes de type :**
  - **NOT NULL**
  - **UNIQUE**
  - **PRIMARY KEY**
  - **FOREIGN KEY**
  - **CHECK**
- **Utilisez la table USER_CONSTRAINTS pour afficher les noms et définitions de toutes les contraintes.**

Résumé

Oracle8 Server empêche la saisie de données incorrectes dans des tables par l'intermédiaire de contraintes.

Les types de contraintes utilisés par Oracle8 sont les suivants :

- NOT NULL
- UNIQUE
- PRIMARY KEY
- FOREIGN KEY
- CHECK

Vous pouvez rechercher les noms et définitions de toutes les contraintes dans la table USER_CONSTRAINTS.

Présentation des Exercices

- **Ajout de contraintes à des tables existantes**
- **Ajout de colonnes supplémentaires à une table**
- **Affichage d'informations des vues du dictionnaire de données**

Présentation des Exercices

Dans les exercices qui suivent, vous allez ajouter des contraintes et des colonnes supplémentaires dans une table en utilisant les ordres étudiés dans ce chapitre.

Exercices 14

1. Ajoutez une contrainte PRIMARY KEY de niveau table dans la table EMPLOYEE en utilisant la colonne ID.
   La contrainte devrait être activée dès sa création.
2. Créez une contrainte PRIMARY KEY sur la table DEPARTMENT en utilisant la colonne ID. La contrainte devrait être activée dès sa création.
3. Ajoutez une clé étrangère dans la table EMPLOYEE qui permettra de contrôler que l'employé n'est pas associé à un département inexistant.
4. Vérifiez que les contraintes ont été ajoutées en les recherchant dans USER_CONSTRAINTS. Prenez note des types et des noms des contraintes. Enregistrez votre ordre dans un fichier nommé *p14q4.sql*.

```
CONSTRAINT_NAME          C
------------------------ --
DEPARTMENT_ID_PK         P
EMPLOYEE_ID_PK           P
EMPLOYEE_DEPT_ID_FK      R
```

5. Recherchez le nom et le type des objets dans la vue USER_OBJECTS du dictionnaire de données correspondant aux tables EMPLOYEE et DEPARTMENT. Vous pouvez mettre les colonnes en forme pour qu'elles soient plus lisibles. Remarquez que pour chaque nouvelle table un nouvel index a été créé.

```
OBJECT_NAME       OBJECT_TYPE
----------------- -----------
DEPARTMENT        TABLE
DEPARTMENT_ID_PK  INDEX
EMPLOYEE          TABLE
EMPLOYEE_ID_PK    INDEX
```

Si vous avez le temps, faites l'exercice suivant :

6. Modifiez la table EMPLOYEE. Ajoutez une colonne SALARY dont le type de donnée est NUMBER, avec une longueur 7.