



UNIVERSITE DE LA MANNOUBA

Ecole Nationale des Sciences de l'Informatique

Module : Systèmes de Gestion de Bases de Données (SGBD)

Dr. Nizar Sghaier
nizar.sghaier@ensi-uma.tn

Introduction

Objectifs des SGBD

- Une Base de données (**BD**) est :
 - Collection de données structurées, sûres, cohérentes, et partageables.
 - Correspond à une représentation fidèle des données et de leur structure avec le minimum de contraintes imposées par le matériel
 - Susceptible d'être utilisée par toutes les applications sans duplication
- Un **SGBD** est un logiciel permettant de:

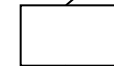
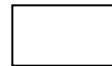
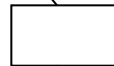
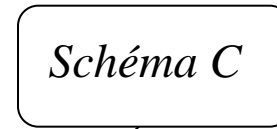
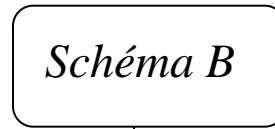
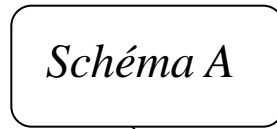
– Décrire		- la confidentialité
– Manipuler	<i>en assurant</i>	- l'intégrité
– Consulter les données		- la sécurité
– Définir des CI		- le partage des données

Fonctions des SGBD

- Description des données : Langage de définition de données (LDD)
 - Recherche des données
 - Mise à jour des données
 - Transformation des données
- } Langage de Manipulation de Données (LMD)
- Contrôle de l'intégrité des données – respect des contraintes d'intégrité:
 - Schéma = structure + contraintes
 - Formule logique (E.g. Nom character 20, non NULL; age integer between 0 and 120; debit <= credit).
 - But: protéger les données
 - Gestion de transactions (*atomicité* des transactions) et sécurité (mots de passe, etc.)

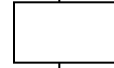
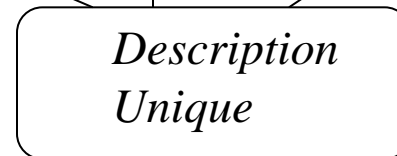
Architecture à niveaux Ansi/Sparc

*Niveau Externe
(vue d'un user)*



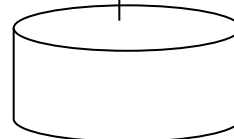
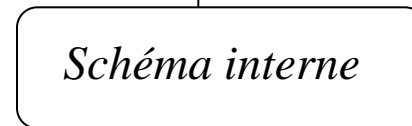
*Correspondance
externe/conceptuelle*

Niveau Conceptuel



*Correspondance
Conceptuelle/physique*

*Niveau Interne
(vue physique)*



Conception de BD

Historique du Langage SQL

- E. F. CODD : premiers articles dans les années 70
- IBM crée le langage SEQUEL (*Structured English Query Language*) ancêtre du langage SQL
- Ce langage devient SQL (*Structured Query Language*, prononcer *eskuel*)
- En 1979, Relational Software Inc. (devenu depuis Oracle) met en place la première version commerciale de SQL

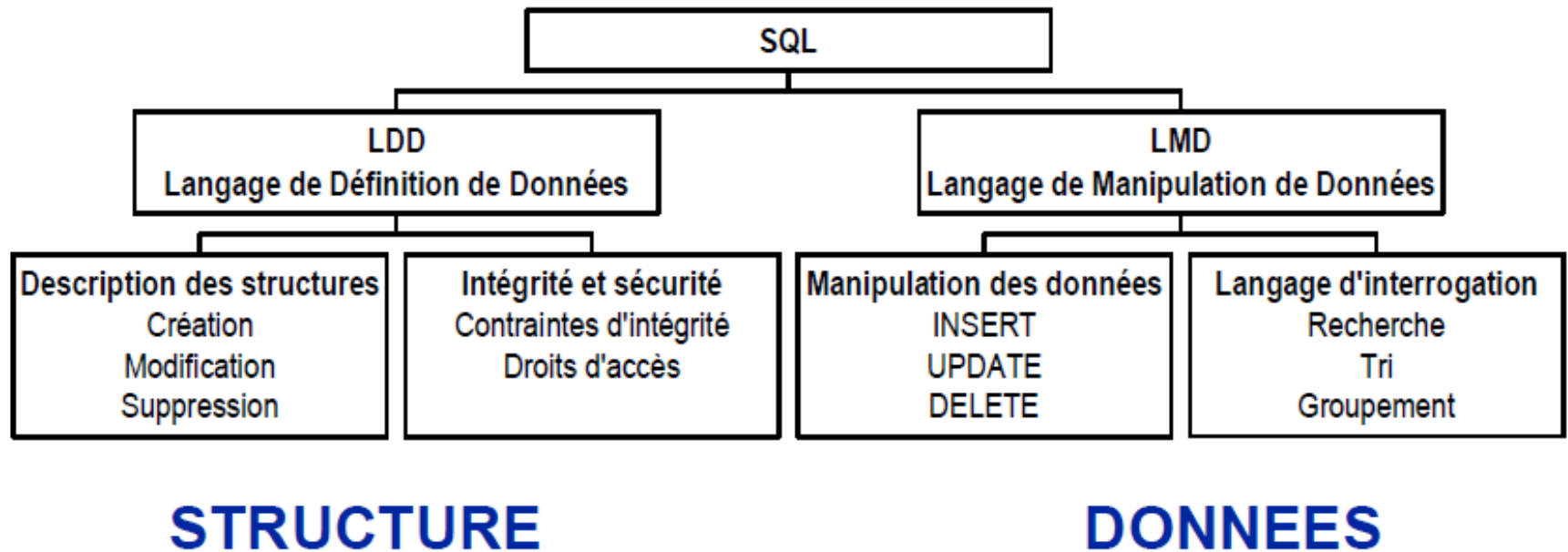
Norme SQL et Editeurs de SGBDR (1)

- Base du succès de SQL
- Fin des SGBD constructeurs
- ANSI (*American National Standards Institute*) et de l'ISO (*International Standards Organization*) qui est affilié à l'IEC (*International Electrotechnical Commission*)
- L'ANSI et l'ISO/IEC ont accepté SQL comme le langage standardisé. La dernière norme publiée par l'ANSI et l'ISO est SQL92 (SQL2)
- On attend la norme SQL3

Norme SQL et Editeurs de SGBDR (2)

- Principaux SGBDR « propriétaires »: Oracle, IBM DB2, Informix, Microsoft SQL-Server, Ingres, Interbase...
- Principaux SGBDR « libres »: MySQL, PostGresql, Firebird...
- Version libre et limitée de SGBDR « propriétaires »: Microsoft Desktop Engine, Oracle 10 G Express, DB2 Express...

Catégories d'instruction SQL



SQL avancé

- Langage de bloc pour augmenter la puissance de SQL :
 - Fonctions itératives et alternatives
 - PL/SQL avec Oracle, Transact-SQL avec SQL-Server
- Notion de Déclencheur ou Trigger
 - MAJ automatique de colonnes dérivées
 - Contraintes complexes
- Notion de Procédure Stockée
 - Programme SQL stocké (compilé) dans la base
- SQL encapsulé : SQL embarqué dans un langage externe
 - Géré par le SGBD : PRO*C, PRO*ADA, ...
 - Extérieur au SGBD : VB, C#, ...

Apprendre SQL avec Oracle

- SGBD le plus répandu dans le monde (gros, moyens et petits systèmes)
- SGBD le plus normalisé
- Produit téléchargeable sur oracle.com à des fins d'apprentissage
- Interface SQL*Plus pour dialoguer avec le langage SQL

Offre complète d'Oracle : Produits proposés

- Noyau Oracle Serveur
 - DBMS : gestionnaire de bases de données
 - Création d'une ou plusieurs instances
 - Licence serveur minimale
 - Toutes plates-formes acceptées
 - Driver SQL*Net serveur
 - PL/SQL : langage de bloc propriétaire
- SQL*Plus
 - Interface minimale pour exécuter des requêtes SQL
`SQL> SELECT * FROM emp ;`
 - Envoi de requêtes et retour des résultats sur écran
 - Appel de blocs, procédures, fonctions...

Offre complète d'Oracle (suite)

- Enterprise Manager
 - Interface graphique pour administrer la base de données distante (DBA)
 - Administration système Oracle
 - Ajout, modification et suppression de tous les objets de la base
 - Surveillance des activités
- SQL*Net
 - Driver propriétaire de communication client et serveur
 - Nécessaire en Client - Serveur et les BD réparties
- Oracle Application Server
 - Pilier de NCA (*Network Computing Architecture*)
 - Serveur Web Transactionnel
 - Cartouches PL/SQL, Java...
 - Intègre le standard CORBA et IIOP

Offre complète d'Oracle (suite)

- Oracle WareHouse
 - Serveur OLAP
 - Analyse décisionnelle
- Oracle Database Designer
 - Atelier de génie logiciel
 - Construction du système d'information (données et programme)
 - Reverse engineering
- Oracle Developer 2000
 - Outil graphique de développement propriétaire
 - Intègre le produit SQL*Forms

Offre complète d'Oracle (fin)

- Oracle Inter-Office
 - Outil de Workflow
 - Gestion des flux de documents électronique
 - Concurrents : Lotus Notes et Exchange
- Oracle Portal
 - Portail d'entreprise
 - Point d'entrée unique de travail
- Certains produits sont aujourd'hui proposés en standard avec Oracle 9i

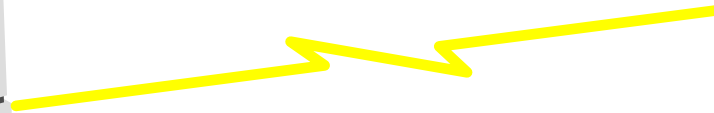
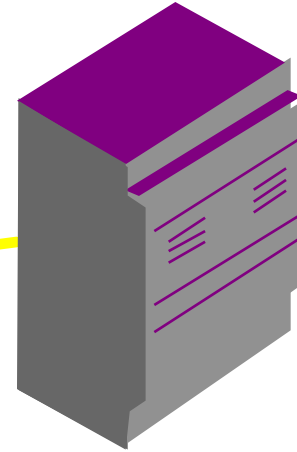
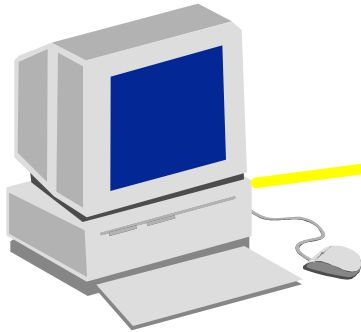
Oracle version libre

- Offre d'un logiciel libre : Oracle XE
- Disponible sur : www.oracle.com
- Version 10g limitée
 - Mémoire (SGA) : 512 Mégas
 - Disque : 2 gigas
 - Pas de limitation du nombre d'utilisateurs
- Utilisation commerciale autorisée
- Interface Windows conviviale

SQL* Plus: en mode ligne de graphique

- Creation d'utilisateur
- – Se connecter en tant que SYS/ORACLE
- – Sys et System sont deux utilisateurs privilégiés d'oracle
- ils ont le rôle DBA (Administrateur)
- – SQL>Create user II2 identified by motpass;
- – SQL>Grant connect to ii2;
- – SQL>Grant resource to ii2;
- – SQL>Connect ii2/motpass
- – SQL> Show user;
- – NB: Ne pas travailler avec le compte SYS et SYSTEM

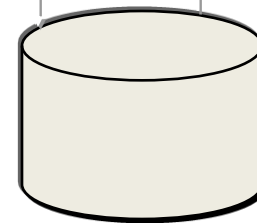
SQL*Plus



```
sqlplus user@connect_string
Password : *****
SQL> SELECT ..... ;
SQL> SAVE req
SQL> START req
SQL> EXIT
```

SAVE

START



req.sql

Le Langage de Définition de Données LDD

Les ordres et les objets

- Ordre CREATE
 - Création de la structure de l'objet → DD
- Ordre DROP
 - Suppression des données et de la structure
- Ordre ALTER
 - Modification de la structure (contenant)
- Syntaxe <Ordre> <Objet> < nom_objet>
- Objet TABLE
- Objet INDEX
- Objet CLUSTER
- Objet SEQUENCE

La commande DESCRIBE

La commande DESCRIBE permet de retrouver le format d'une table

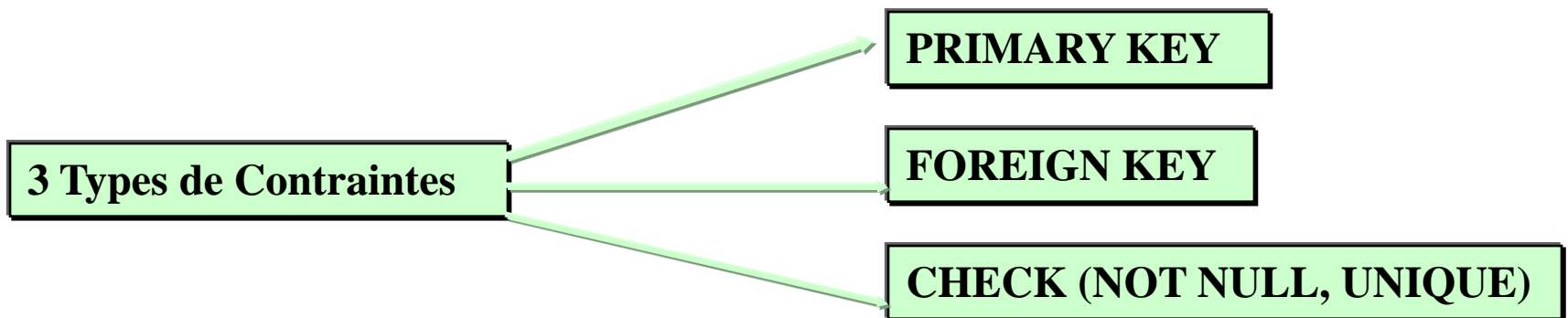
```
SQL> describe s_dept;
```

Name	Null?	Type
-----	-----	-----
ID	NOT NULL	NUMBER (7)
NAME	NOT NULL	VARCHAR2 (25)
REGION_ID		NUMBER (7)

Objet Table et Contraintes

CREATE : Syntaxe

```
create table nom_table
  (colonne1          type1(longueur1) ,
   colonne2          type2(longueur2) ,
   .....
   constraint nom_constraint1
     type_constraint1,
   .....
  ) ;
```



Objet Table et Contraintes

Les types de données

<code>VARCHAR(<i>size</i>)</code>	Données caractères de longueur variable
<code>CHAR(<i>size</i>)</code>	Données caractères de longueur fixe
<code>NUMBER(<i>p</i>, <i>s</i>)</code>	Numérique de longueur variable
<code>DATE</code>	Valeurs de date et d'heure
<code>LONG</code>	Données caractères de longueur variable (2 Go)
<code>CLOB</code>	Données caractères (4 Go)
<code>RAW</code>	Binaire
<code>BLOB</code>	Binaire, jusqu'à 4 giga-octets
<code>BFILE</code>	Binaire, stocké dans un fichier externe, (4 Go)

Objet Table et Contraintes

CREATE : Exemples

```
-- Table 'Mère'

CREATE TABLE service
  (IdService          CHAR(3) ,
   NomService         VARCHAR(30) ,
   CONSTRAINT pk_service
     PRIMARY KEY (IdService)
  ) ;
```


Objet Table et Contraintes

CREATE : Exemples (suite)

```
-- Table 'Fille'
CREATE TABLE employe
    (IdEmploye          NUMBER(5) ,
     NomEmploye         VARCHAR(30) ,
     Indice             NUMBER(3) ,
     DateEmbauche       DATE DEFAULT SYSDATE,
     IdService          CHAR(3)
      CONSTRAINT nn_emp_ser NOT NULL,
      CONSTRAINT pk_employe
        PRIMARY KEY(IdEmploye) ,
      CONSTRAINT fk_emp_ser FOREIGN KEY(IdService)
        REFERENCES service(IdService) ,
      CONSTRAINT ck_emp_indice CHECK
        (indice BETWEEN 100 AND 900)
    );
```

Objet Table : DROP

```
DROP TABLE nom_table;
```

Suppression complète de la table : définition et données

```
DROP TABLE nom_table CASCADE CONSTRAINTS;
```



Suppression aussi des contraintes de référence filles

Modification de la structure

ALTER TABLE

Ajout de colonnes

```
ALTER TABLE nom_table  
    ADD (colonne1 type1, colonne2 type2);
```

Modification de colonnes

```
ALTER TABLE nom_table  
    MODIFY (colonne1 type1, colonne2 type2);
```

Suppression de colonnes

```
ALTER TABLE nom_table  
    DROP COLUMN (colonne1, colonne2);
```



ALTER TABLE

Exemples de modifications

```
ALTER TABLE client  
    ADD ChiffreAffaire NUMBER (10,2);
```

```
ALTER TABLE client MODIFY nom VARCHAR(60);
```

```
ALTER TABLE etudiant  
MODIFY idDiplome CONSTRAINT nn_etu_dip NOT NULL;
```

```
ALTER TABLE client  
    DROP COLUMN ChiffreAffaire ;
```

Contraintes

```
constraint nomcontrainte
{ unique | primary key (col1[,col2]...)
      | foreign key (col1[,col2]...)
references [schema].table (col1[,col2]...)
[ON DELETE CASCADE]
      | check (condition) }
```



Attention : suppression de tous les fils !

Modification des contraintes

Ajout et Suppression

Ajout de contraintes

```
ALTER TABLE nom_table  
    ADD CONSTRAINT nom_contrainte  
        type_contrainte;
```



Comme à la création d'une table

Suppression de contraintes

```
ALTER TABLE nom_table  
    DROP CONSTRAINT nom_contrainte;
```

Modification des contraintes

Exemples

```
ALTER TABLE client
  ADD CONSTRAINT fk_client_cat
  FOREIGN KEY(idCat)
  REFERENCES categorie(idCat);
```

```
ALTER TABLE client
  DROP CONSTRAINT fk_client_cat;
```

Activation et désactivation de contraintes

Désactivation de contraintes

```
ALTER TABLE nom_table  
        DISABLE CONSTRAINT nom_contrainte;
```

```
ALTER TABLE nom_table  
        DISABLE CONSTRAINT PRIMARY KEY;
```

➡ Les contraintes existent toujours dans le dictionnaire de données
mais ne sont pas actives

➡ Chargement de données volumineuses extérieures à la base

Activation d'une contrainte désactivée

Activation de contraintes


```
ALTER TABLE nom_table  
    ENABLE CONSTRAINT nom_contrainte ;
```

```
ALTER TABLE nom_table  
    ENABLE CONSTRAINT PRIMARY KEY;
```

Ajout ou activation de contraintes :

Récupération des lignes en erreur

Création d'une table Rejets

```
CREATE TABLE rejets
(ligne          rowid,  Adresse ligne
proprietaire    varchar(30) ,
nom_table       varchar(30) ,
contrainte      varchar(30) ) ;
```

Activation de contraintes

```
ALTER TABLE nom_table
ENABLE CONSTRAINT nom_contrainte
EXCEPTIONS INTO rejets;
```

Vérification de Contraintes différées

- **Une contrainte peut-elle être différée ?**
 - **NOT DEFERRABLE** (par défaut)
 - **DEFERRABLE**
- **Comportement par défaut de la contrainte :**
 - **INITIALLY IMMEDIATE** (par défaut)
 - **INITIALLY DEFERRED**
- **Utiliser la clause SET CONSTRAINTS ou ALTER SESSION SET CONSTRAINTS pour modifier le comportement d'une contrainte**

Vérification de Contraintes différées

Exemples

```
CREATE TABLE emps ...  
    dept_id      NUMBER(6)  
                CONSTRAINT fk_emp_dept REFERENCES depts  
                    DEFERRABLE INITIALLY IMMEDIATE);
```

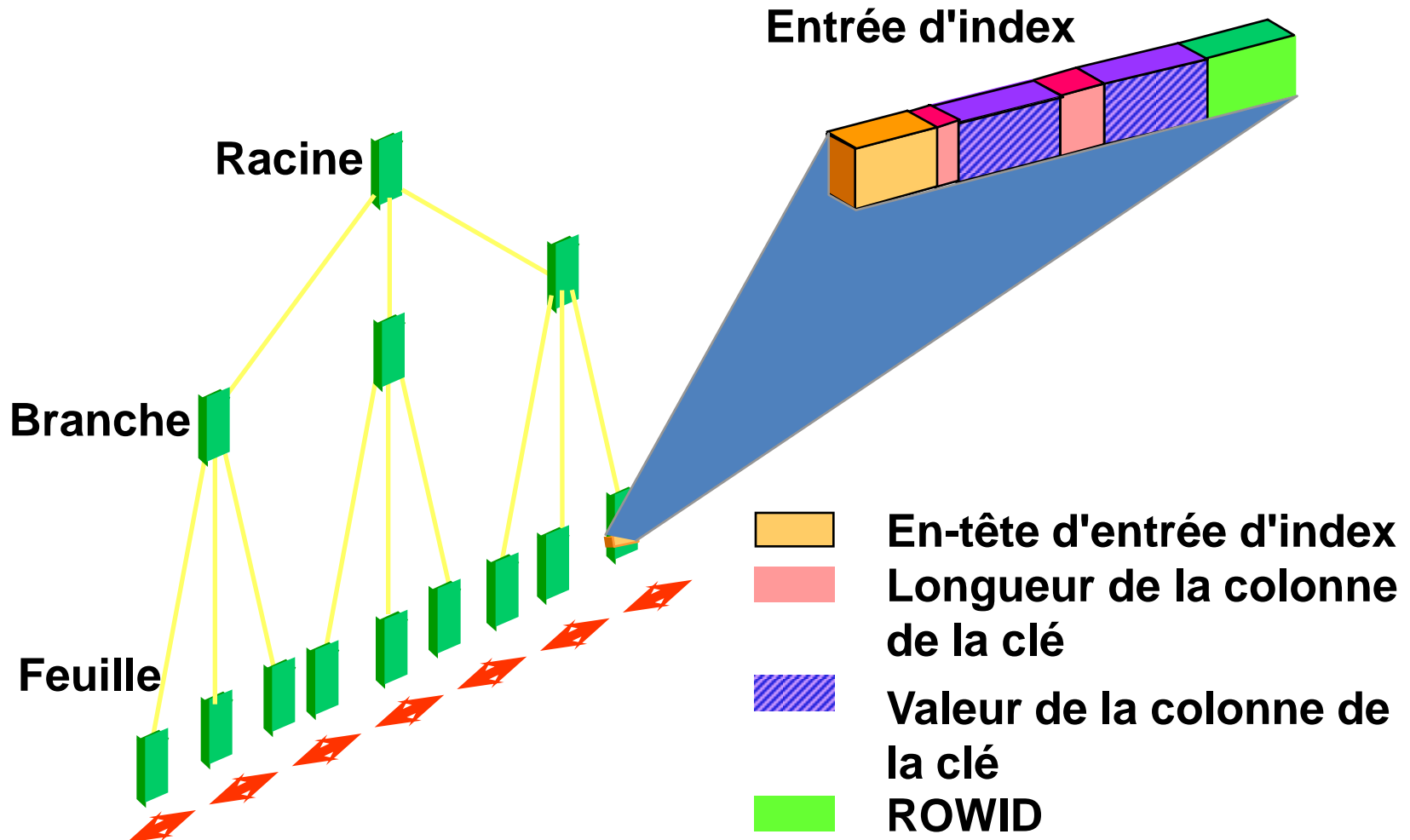
```
SQL> INSERT INTO emps VALUES (1, 'Laurent', 2)  
      *  
ERROR at line 1:  
ORA-02291: integrity constraint (MICHEL.FK_EMP_DEPT_ID)  
violated - parent key not found ;
```

```
SET CONSTRAINTS ALL DEFERRED;
```

```
SQL> INSERT INTO emps VALUES (1, 'Laurent', 2);  
1 row created.
```

```
SQL> COMMIT;  
COMMIT  
*  
ERROR at line 1:  
ORA-02091: transaction rolled back  
ORA-02291: integrity constraint (MICHEL.FK_EMP_DEPT_ID)  
violated - parent key not found
```

Les fichiers Index : organisation en B-Arbre



Création et suppression d'Index

Création d'un index Unique

```
CREATE UNIQUE INDEX nom_index  
ON nom_table(colonne[,colonne2 ...]);
```

Création d'un index non Unique

```
CREATE INDEX nom_index  
ON nom_table(colonne[,colonne2 ...]);
```

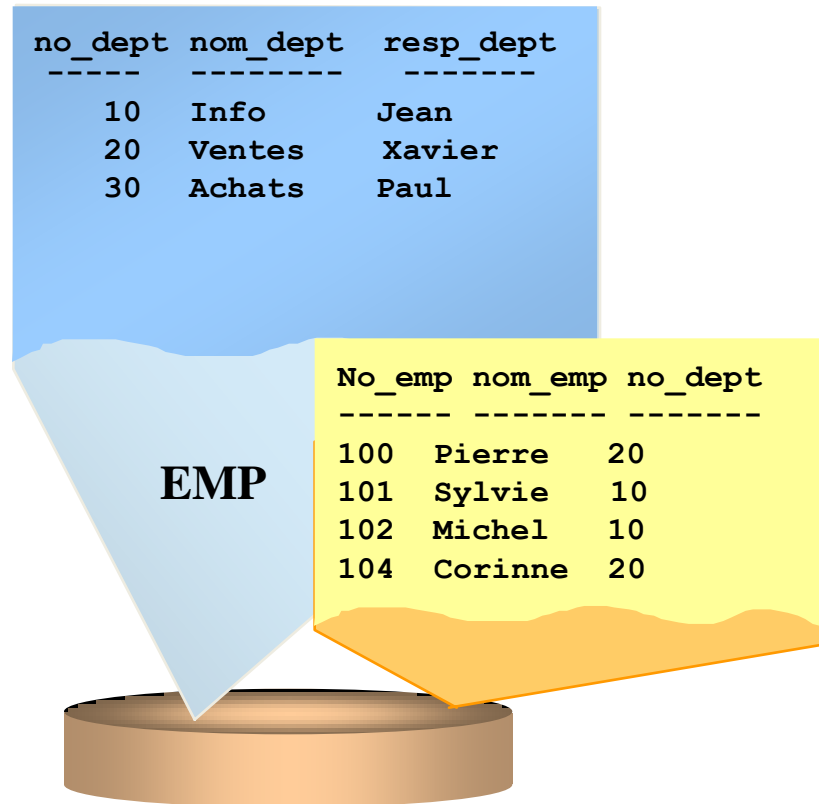
Suppression d'un index

```
DROP INDEX nom_index;
```

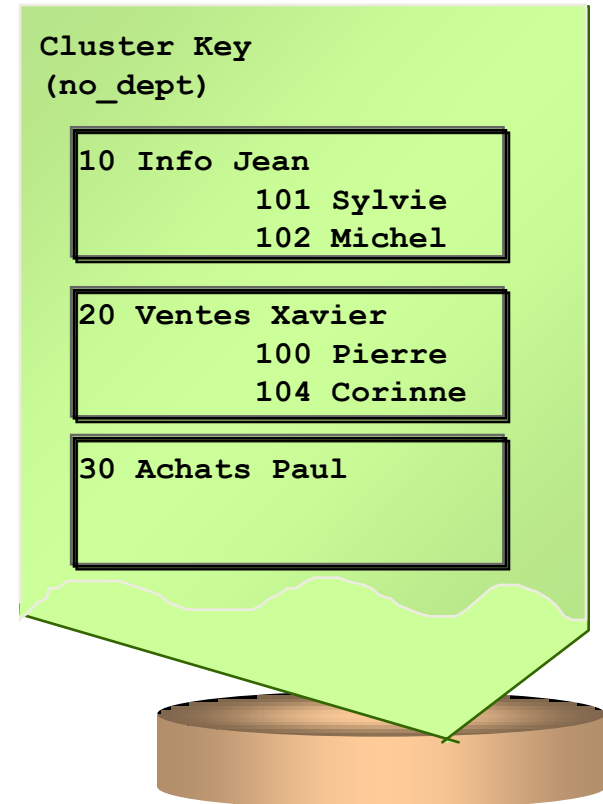
Apport des Index

- Respect de la 4NF : clé primaire → index unique
- Amélioration des accès (sélection) sur les colonnes recherchées
- Optimisation des jointures (équi-jointure entre une clé primaire et sa clé étrangère)
- Attention aux clés primaires composées
- Table USER_INDEXES du dictionnaire

Cluster : Jointure physique



Tables DEPT et EMP
non-clusterisées



Tables DEPT et EMP
clusterisées

Création de Cluster (1)

1. Création du cluster

```
CREATE CLUSTER personnel  
  (no_dept NUMBER(3))  
  SIZE 200 TABLESPACE ts1  
  STORAGE (INITIAL 5M NEXT 5M PCTINCREASE 0) ;
```

Taille du
bloc logique

2. Création de l'index de cluster

```
CREATE INDEX idx_personnel  
ON CLUSTER personnel  
TABLESPACE tsx1  
STORAGE (INITIAL 1M NEXT 1M PCTINCREASE 0) ;
```

Création de Cluster (2)

3- Création des tables dans le cluster

```
CREATE TABLE dept
(no_dept NUMBER(3)
CONSTRAINT pk_dept PRIMARY KEY,
nom_dept VARCHAR(30), resp_dept VARCHAR(30))
CLUSTER personnel (no_dept);
```

```
CREATE TABLE emp
(no_emp NUMBER(3) CONSTRAINT pk_emp PRIMARY KEY,
nom_emp VARCHAR(30),
no_dept NUMBER(3) REFERENCES dept (no_dept))
CLUSTER personnel (no_dept);
```

Administration des Clusters

Modification des paramètres de stockage et de consommation d'espace d'un bloc

```
ALTER CLUSTER personnel  
SIZE 300K STORAGE (NEXT 2M) ;
```

Suppression des Clusters

```
DROP CLUSTER personnel  
INCLUDING TABLES ;
```

OU

```
DROP TABLE emp ;  
DROP TABLE dept ;  
DROP CLUSTER personnel ;
```

Objet Séquence

- Permet d'obtenir des valeurs incrémentales
- Équivalent des colonnes AUTO_INCREMENT de MySql ou IDENTITY de SqlServer
- N'est pas associée à une colonne particulière
- Verrouillage automatique en cas de concurrence d'accès
- Valeur suivante : <nom_séquence>.NEXTVAL
- Valeur courante : <nom_séquence>.CURRVAL

Objet Séquence

création et utilisation

```
CREATE SEQUENCE nom_séquence  
START WITH valeur_départ  
INCREMENT BY incrément;
```

```
INSERT INTO t1 VALUES  
(nom_séquence.NEXTVAL, ....);  
INSERT INTO t2 VALUES  
(....., nom_séquence.CURRVAL);
```

```
DROP SEQUENCE nom_séquence;
```

Objet Séquence

Exemple de mise en oeuvre

```
SQL> CREATE TABLE client (idClient NUMBER PRIMARY KEY,  
    2  nomClient VARCHAR(20));  
Table créée.  
SQL> CREATE TABLE compte (idCompte NUMBER PRIMARY KEY,  
    2  nomCompte VARCHAR(30), idClient REFERENCES client);  
Table créée.  
SQL> CREATE SEQUENCE seq_client START WITH 1 INCREMENT BY 1;  
Séquence créée.  
SQL> CREATE SEQUENCE seq_compte START WITH 1 INCREMENT BY 1;  
Séquence créée.  
SQL> INSERT INTO client VALUES(seq_client.NEXTVAL,'Michel');  
1 ligne créée.  
SQL> SELECT seq_client.CURRVAL FROM dual;  
    CURRVAL  
-----  
          1
```

Objet Séquence

Exemple de mise en œuvre (suite)

```
SQL> INSERT INTO compte VALUES(seq_compte.NEXTVAL,'Compte  
Courant Michel',seq_client.CURRVAL);
```

1 ligne créée.

```
SQL> INSERT INTO compte VALUES(seq_compte.NEXTVAL,'Compte  
Epargne Michel',seq_client.CURRVAL);
```

1 ligne créée.

```
SQL> SELECT * FROM client;
```

IDCLIENT	NOMCLIENT
----------	-----------

1	Michel
---	--------

```
SQL> SELECT * FROM compte;
```

IDCOMPTE	NOMCOMPTE
----------	-----------

IDCLIENT

1	Compte Courant Michel	1
2	Compte Epargne Michel	1