SGBD

VUES, SÉQUENCES, INDEX ET TRANSACTIONS

PLAN

- Introduction
- Rappels
- Les vues
- Les séquences
- Les index
- Les transactions

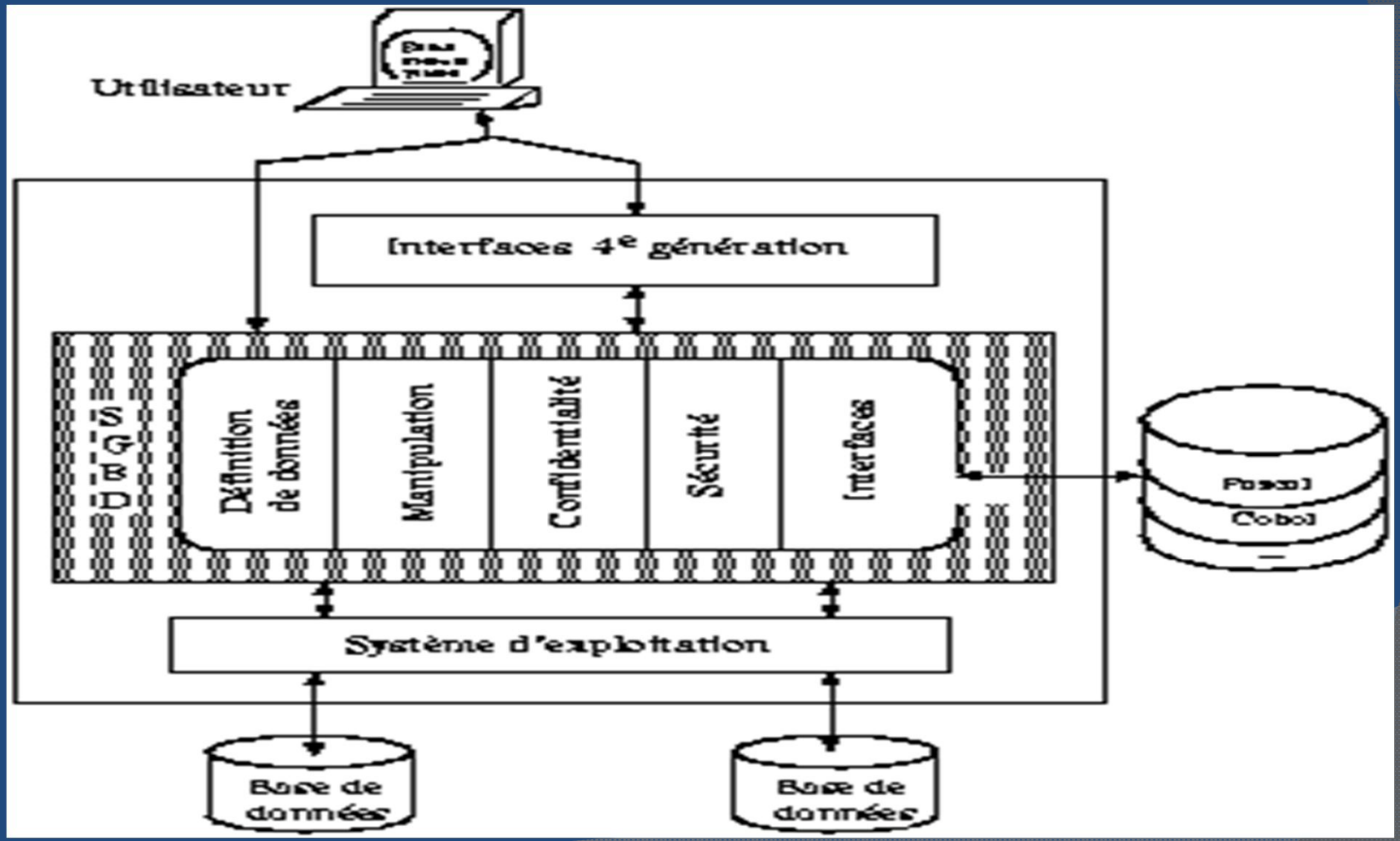
INTRODUCTION

- SGBD= ensemble d'outils logiciels permettant la création et l'exploitation des bases de données
- Doit assurer les fonctions suivantes:
  1. *Description et Définition des Données* :créer, définir les paramètres et les objets de la base
  2. *Manipulation*: recherche, ajout, suppression, modification

INTRODUCTION

3. *Intégrité*: garantir la cohérence des données et respecter les règles de gestion du système
4. *Confidentialité*: chaque utilisateur n'effectue que les opérations autorisées
5. *Gestion des transactions* (concurrence d'accès) et *sécurité* (mot de passe, …)

INTRODUCTION

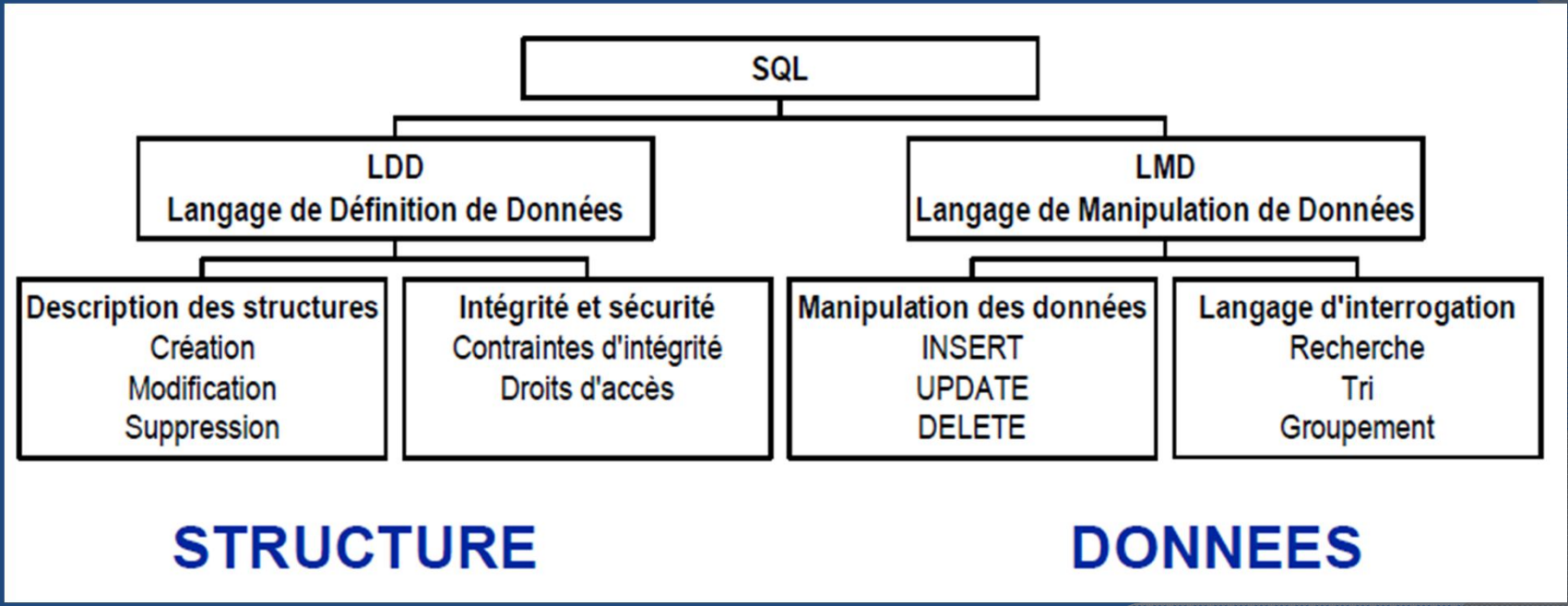


RAPPELS (1/11)

- Lexique de SQL
  - Mots clé
    - Les mots clé du langage peuvent être en majuscule, minuscule ou encore en mixte
  - Noms des objets et des colonnes (identifiants)
    - Les identifiants peuvent être utilisés indifféremment en majuscule, minuscule ou mixte
  - Constantes
    - Les constantes de type chaîne de caractères doivent être entourées par des quotes simples
    - les constantes numériques sont utilisées sans quotes
  - Commentaire
    - il est possible d'inclure des retours chariots, des tabulations, espaces et commentaires (sur une ligne précédée de deux tirets --, sur plusieurs lignes entre /* et */).

RAPPELS (2/11)

- Types d'instructions SQL



RAPPELS (3/11)

- Règles de création des tables
  - Nom_table : **toutes les tables de la base doivent avoir des noms différents**
  - Définition des attributs :
    - **Donner un nom pertinent à la colonne**
    - **Affecter un type à la colonne (les types de données)**
    - **Préciser les valeurs éventuelles par défaut de la colonne**
    - **Associer une contrainte éventuelle à la colonne**
  - Mots clé:
    - CONSTRAINT : **permet de donner un nom à la contrainte**
    - PRIMARY KEY : **déclare que la colonne est la clé primaire**
    - NOT NULL : **interdit aux valeurs de la colonne d'être indéfinies**
    - UNIQUE : **indique que toutes les valeurs de la colonne doivent être différentes.**
    - CHECK : **permet de définir une contrainte booléenne sur la colonne**
    - REFERENCES**: permet de déclarer les colonnes clés étrangères**

RAPPELS (4/11)

- Contraintes

  Quatre types de contraintes sont possibles :

  - **CONSTRAINT** *nomContrainte*
    UNIQUE (*colonne1* [,*colonne2*]…)
    PRIMARY KEY (*colonne1* [,*colonne2*]…)
    FOREIGN KEY (colonne1 [,*colonne2*]…) REFERENCES
    [*schéma*.]*nomTablePere* (*colonne1* [,*colonne2*]…)
    [ON DELETE { CASCADE | SET NULL | SET DEFAULT }]
    [ON UPDATE {NO ACTION | CASCADE | SET NULL
    | SET DEFAULT }]
    CHECK (*condition*)

RAPPELS (5/11)

- **ON DELETE:** indique l'action à effectuer sur la table indiquée lors d'une suppression dans la table référencée.
- **ON UPDATE:** indique l'action à effectuer sur la table indiquée lors d'une modification dans la table référencée.
  - **CASCADE :**effacer ou mettre à jour les tuples correspondant dans la table indiquée.
  - **SET NULL** : le tuple cible est supprimé ou modifié et les tuples clé étrangère associés sont initialisé à zéro.
  - **SET DEFAULT :** le tuple cible est supprimé ou modifié et les tuples clé étrangère associés sont initialisé à leur valeur par défaut.
  - **NO ACTION:** équivalent à l'omission des clauses ON DELETE, ON UPDATE

RAPPELS (6/11)

- Structure d'une table (DESC)
  - DESC (raccourci de DESCRIBE) est une commande SQL*Plus. Elle permet d'extraire la structure brute d'une table.
  - Elle peut aussi s'appliquer à une vue.
  - Enfin, elle révèle également les paramètres d'une fonction ou procédure.

DESC[RIBE] [*schéma*.]*élément;*

RAPPELS (7/11)

- Renomage d'une table (RENAME)
  - Il faut être propriétaire de l'objet que l'on renomme.

    RENAME *ancienNom* TO *nouveauNom*;

  - Les contraintes d'intégrité, index et prérogatives associés à l'ancienne table sont automatiquement transférés sur la nouvelle.
  - En revanche, les vues, synonymes et procédures catalogués sont invalidés et doivent être recréés.

RAPPELS (8/11)

- Modifications structurelles (ALTER TABLE)

  Ajout de colonnes

  - La directive ADD de l'instruction ALTER TABLE permet d'ajouter une nouvelle colonne à une table.
  - Cette colonne est initialisée à NULL pour tous les enregistrements (à moins de spécifier une contrainte DEFAULT, auquel cas tous les enregistrements de la table sont mis à jour avec une valeur non nulle).

RAPPELS (9/11)

Renomage de colonnes

- Il faut utiliser la directive RENAME COLUMN de l'instruction ALTER TABLE pour renommer une colonne existante.
- Le nom de la nouvelle colonne ne doit pas être déjà utilisé par une colonne de la table.

ALTER TABLE nom table

RENAME COLUMN *ancienNom* TO n*ouveauNom*;

RAPPELS (10/11)

Suppression de colonnes

- La directive DROP COLUMN de l'instruction ALTER TABLE permet de supprimer une colonne.
- Il n'est pas possible de supprimer avec cette instruction :
  - des clés primaires (ou candidates par UNIQUE) référencées par des clés étrangères ;
  - des colonnes à partir desquelles un index a été construit ;
  - toutes les colonnes d'une table.

RAPPELS (11/11)

Modification de contraintes

- La directive ADD CONSTRAINT de l'instruction ALTER TABLE permet d'ajouter une contrainte à une table. La syntaxe générale est la suivante

ALTER TABLE nom table

ADD CONSTRAINT *Nom contrainte* ;

Suppression de contraintes

ALTER TABLE nom table

DROP CONSTRAINT *Nom contrainte [CASCADE]* ;

(Cette commande supprime à la fois la clé primaire de la table indiquée mais aussi les contraintes clés étrangères des tables dépendantes)

DEFINITION

- Un objet est une composante stockée dans la base de données.
- Liste des objets d'une BD:
  - Table, contrainte, Vue, séquence, index, procédure, …
- Visualiser la liste des objets créés par un utilisateur:

```
SELECT * FROM user_catalog;
```

LES VUES (1/6)

- Définition
  - Une vue est une table virtuelle constituée par des sous-ensembles de données d'une ou plusieurs tables. Elle est définie par une requête prédéfinie stockée dans la base de données. Elle est exploitée comme une table mais elle ne nécessite pas d'espace physique de stockage.

LES VUES (2/6)

- Utilités
  - Pour des raisons de sécurité : restreindre l'accès aux données.
  - Conserver des données de synthèse afin d'éviter les requêtes très complexes.
  - Permettre une certaine indépendance des données.
  - Présenter des différentes vues pour les mêmes données…

LES VUES (3/6)

- Création d'une vue :
  - Cas d'une vue simple (à partir d'une table).

```
CREATE VIEW Nom_Vue  AS
 SELECT * | COLONNE1 [,COLONNE2…]
  FROM  Nom-Table
  WHERE Conditions;
```

Exemple: Créer une vue des noms, postes et salaires des employés du service n°10.

RQ! On peut renommer les noms des colonnes dans la vue en utilisant les alias dans la sous-requête.

```
CREATE VIEW service10 As
SELECT NumP Numero, NomP Nom,
PosteP Poste, SalaireP Salaire
FROM personne WHERE  NumServP = 10;
```

LES VUES (4/6)

- Remarques:
  - On peut décrire et faire des sélections dans les vues comme les tables (➔ *on peut appliquer le langage d'interrogation des données sur les vues*)

```
SELECT * FROM Nom_Vue;
```

- Suppression d'une vue:
  - La suppression d'une vue s'effectue de la même façon qu'une table.

```
Drop Nom_Vue;
```

LES VUES (5/6)

- Cas d'une vue complexe (à partir de plusieurs tables)

```
CREATE VIEW Nom_Vue  AS
SELECT * ! COLONNE1   [, COLONNE2,…]
FROM   TABLE1,    TABLE2
WHERE          Conditions
[GROUP  BY…] ;
```

RQ! Les vues complexes sont souvent utilisées pour la synthèse des données.

Exemple :Créer une vue qui contient, pour chaque service, le nom du service, le nombre d'employés le salaire moyen dans ce service.

```
CREATE VIEW  Serv1  AS
        Select NomServ Nom,
                count(NumP) Nombre_Employe,
                AVG(SalaireP) Salaire_moyen
        FROM personne, service
        WHERE   NumServP = NumServ
        GROUP BY NomServ;
```

LES VUES (6/6)

- **L'option WITH CHECK OPTION**
  - L'option WITH CHECK OPTION est une option de l'instruction CREATE qui peut être ajoutée afin que toutes les instructions UPDATE et INSERT répondent aux conditions de la définition de la vue.
  - Exemple :

```
CREATE VIEW Pers_Embauche AS
SELECT NomP, PosteP, datembp
FROM personne
WHERE datembp IS NOT NULL
With check option;
```

Dans ce cas, l'option WITH CHECK OPTION *refuse toute valeur NULL dans la colonne « datembp »!*

LES SEQUENCES (1/11)

- Définition
  - Une séquence est un objet de base de données, au même titre qu'une table, une vue ou un index, dont le rôle est de générer automatiquement une série de numéros (entiers) uniques. Elle est généralement utilisée pour générer les valeurs des clés primaires et elle peut être partagée par plusieurs utilisateurs.

LES SEQUENCES (2/11)

- Syntaxe

  CREATE SEQUENCE <nom-séquence>
  [INCREMENT BY <valeur-pas>] *(par défaut: 1)*
  [START WITH <valeur-début>]
  [MAXVALUE <valeur-max> | NOMAXVALUE]
  (par défaut: NOMAXVALUE)
  [MINVALUE <valeur-min> | NOMINVALUE]
  [CYCLE | NOCYCLE] *(par défaut: NOCYCLE)*
  [CACHE <nb-valeurs>| NOCACHE];
  (par défaut: NOCACHE)

LES SEQUENCES (3/11)

- **INCREMENT BY** :spécifie l'intervalle d'incrémentation. Cette valeur peut être positive ou négative mais ne peut pas valoir 0. Par défaut, l'incrémentation vaut 1.
- **START WITH** :spécifie la première valeur délivrée par la séquence. Par défaut, cette valeur est la valeur minimum pour les incrémentations positives et la valeur maximum pour les négatives.
- **MINVALUE** :spécifie la valeur minimum de la séquence. Doit être au minimum égal à la valeur spécifiée par START WITH.
- **MAXVALUE** :spécifie la valeur maximum que le compteur peut générer.

LES SEQUENCES (4/11)

- **CYCLE** :permet à la séquence de recommencer lorsqu'elle atteint le maximum. NOCYCLE interdit à la séquence de générer des valeurs lorsqu'elle atteint ce seuil.
- **CACHE** permet d'optimiser l'utilisation des séquences en plaçant en mémoire un certain nombre de valeurs générées par la séquence.

Exemple : Créer une séquence Serv_id qui sera utilisée pour les valeurs de la clé primaire de la table service (sans cycle).

```
CREATE SEQUENCE  Serv_id
INCREMENT BY 10
START WITH 10
MAXVALUE 500
NOCYCLE
NOCACHE;
```

LES SEQUENCES (5/11)

Ex: CREATE SEQUENCE Serv_id ….CACHE 100;

indique que les 100 valeurs successives sont générées en même temps et sont placées dans la mémoire.

Un autre ensemble de valeurs est généré seulement si toutes les valeurs de la mémoire sont utilisées.

RQ! Attention, quand la base est arrêtée, le contenu du cache est perdu, ce qui peu provoquer une discontinuité dans les valeurs de la séquence.

LES SEQUENCES (6/11)

- Interrogation d'une Séquence

  - L'interrogation d'une séquence se fait par l'utilisation des pseudo colonnes : CURRVAL et NEXTVAL.

    - *NEXTVAL* : incrémente la séquence et retourne la nouvelle valeur. Si c'est le premier appel, alors elle initialise la séquence.
    - *CURRVAL* : retourne la valeur courante de la séquence. Elle ne peut être appelée que si NEXTVAL a été appelée au moins une fois (pour l'initialisation).

LES SEQUENCES (7/11)

- Utilisation
  - INSERT INTO nom_table VALUES (nom_sequence.NEXTVAL, listes_colonnes …);
  - SELECT nom_sequence.CURRVAL FROM SYS.dual;

- Exemple : Peupler la table service (initialement vide) en utilisant la séquence serv_id pour définir la valeur de la clé primaire.

INSERT INTO serv VALUES (serv_id.NEXTVAL, ‘FINANCE’, ’TUNIS’);

➔ Insertion ligne (**10**, ‘FINANCE’,’TUNIS’)

INSERT INTO serv VALUES (serv_id.NEXTVAL, ‘RECHERCHE’, ’ARIANA’);

➔ Insertion ligne (**20**, ‘RECHERCHE’,’ARIANA’)

INSERT INTO serv VALUES (serv_id.NEXTVAL, ‘VENTES’, ’NABEUL’);

➔ Insertion ligne (**30**, ‘VENTES’,’NABEUL’)

……..

LES SEQUENCES (8/11)

- Modification d'une séquence:
  - Il est possible de modifier tous les paramètres d'une séquence (à part le paramètre START WITH).

```
ALTER SEQUENCE nom_séquence
[INCREMENT BY <valeur-pas>]
[MAXVALUE <valeur-max> | NOMAXVALUE]
[CYCLE | NOCYCLE]
[CACHE <nb-valeurs>| NOCACHE];
```

Remarques:

- La modification des paramètres d'une séquence n'affecte pas sa valeur courante (elle reste inchangée).
- La séquence doit être supprimée puis recréée pour la faire repartir d'un numéro différent.

LES SEQUENCES (9/11)

- Exemple : modifier le pas d'incrément à 20 au lieu de 10 dans Serv_Id.
- Puis insérer un autre service.

- EX: ALTER SEQUENCE serv_id INCREMENT BY 20 MAXVALUE 100;
- INSERT INTO serv_id VALUES (serv_id.NEXTVAL, ‘FABRICATION’, ’MEGRINE’);

LES SEQUENCES (10/11)

- Suppression d'une séquence :

DROP SEQUENCE nom_séquence;

- Consultation d'une séquence
  - Toutes les informations concernant les séquences crées sont disponibles dans la vue user_sequences du dictionnaire des données:

```
SELECT sequence_name, min_value,
max_value increment_by, last_number
FROM user_sequences;
```

Last_number : dernier numéro généré.

LES SEQUENCES (11/11)

- Remarques
  - Etant donnée qu'une séquence peut être interrogée à tout moment par tout utilisateur ayant les droits suffisants, il ne faut pas considérer les séquences comme un moyen de générer une suite de nombre continue
  - Dans le cas de l'alimentation d'une clé primaire, si un enregistrement a été inséré puis la transaction a été annulée, alors la séquence ne revient pas en arrière, et lors de l'insertion suivante, une nouvelle valeur est générée!
  - Une séquence fournit un moyen d'obtenir des valeurs uniques, mais pas forcément continues.