

## Le langage SQL - Autres ordres



P.Mathieu

LP DA2I Lille  
<http://www.iut-a.univ-lille.fr>  
[prenom.nom@univ-lille.fr](mailto:prenom.nom@univ-lille.fr)

16 septembre 2018

### 1 Ordres de mise à jour

### 2 Exercice

### 3 La définition des données : Le DDL

### 4 La création de tables

### 5 Abstraction logique

### 6 Autres ordres du DDL

## Ordres de mise à jour

### Insertion en extension



2 formes possibles : en extension (VALUES) ou en intention (SELECT)

*ajouter un nouveau produit*

```
INSERT INTO produits (pno,design,prix,poids,couleur)
VALUES (8,'ecrou',5,12,'vert') ;
```

ou encore

*ajouter un nouveau produit (2eme méthode)*

```
INSERT INTO produits
VALUES (8,'ecrou',5,12,'vert') ;
```

## Ordres de mise à jour

### Insertion en extension



Les colonnes non citées sont mises à NULL

*ajouter un nouveau produit dont on ne connaît pas le poids et la couleur*

```
INSERT INTO produits (pno,design,prix)
VALUES (8,'ecrou',5) ;
```

ou aussi

```
INSERT INTO produits (pno,design,prix,poids,couleur)
VALUES (8,'ecrou',5,NULL,NULL) ;
```

ou encore

```
INSERT INTO produits
VALUES (8,'ecrou',5,NULL,NULL) ;
```

## Ordres de mise à jour

### L'insertion avec Select

*ajouter dans la table petites\_commandes les numéros et quantités des produits commandés en moins de 5 exemplaires*

```
INSERT INTO petites_commandes (pno, qute)
  SELECT pno, qute FROM commandes
  WHERE qute < 5 ;
```

## Ordres de mise à jour

### Utilisation d'une constante à l'insertion

*ajouter dans la table braderie les numéros de produits entre 100€ et 500€ et les afficher à un prix de 100€*

```
INSERT INTO braderie (pno, prix)
  SELECT pno, 100 FROM produits
  WHERE produit.prix < 500 AND produit.prix > 100 ;
```

## Ordres de mise à jour

### Le Update

L'expression caractérisant la modification à effectuer peut être une constante, une expression arithmétique ou le résultat d'un SELECT.

*Augmenter le prix de tous les produits rouges de 5%*

```
UPDATE produits
  SET prix = prix*1.05
  WHERE couleur='rouge' ;
```

## Ordres de mise à jour

### Problèmes d'intégrité ...

*Incrémentation de 1 de toutes les clés*

```
UPDATE TABLE
  SET cle=cle+1 ;
```

Quand doit-on vérifier l'intégrité de la clé unique ?

## Ordres de mise à jour

La suppression.

*Supprimer les produits dont le prix est supérieur à 100F*

```
DELETE FROM produits WHERE prix>100 ;
```

L'ordre DELETE FROM <table>; permet de vider complètement une table. Néanmoins, dans ce cas, la table existe toujours bien qu'elle soit vide.

1 Ordres de mise à jour

2 Exercice

3 La définition des données : Le DDL

4 La création de tables

5 Abstraction logique

6 Autres ordres du DDL

## Exercice



On considère une base de données permettant de gérer des pièces de théâtre. De cette base nous avons extrait 3 relations :

```
spectacle(num-spec, titre-spec, salle, monteur)
jouer(nom-acteur, num-spec)
représentations(date, heure, num-spec, tarif)
```

## Exercice



Ecrire en SQL les requêtes suivantes

- 1 Ajouter l'acteur Giraudeau dans le spectacle 10
- 2 Ajouter dans la table promos(num-spec,titre,prix) les spectacles moins chers que 200F
- 3 Augmenter tous les tarifs de 10%
- 4 Changer l'heure de 16 :00 à 15 :00 pour les spectacles du 17/10/2002
- 5 Supprimer les représentations dont la date est dépassée (date système today())
- 6 Supprimer les représentations du 12/10/2002 dont le prix est supérieur à 300F

1 Ordres de mise à jour

2 Exercice

3 La définition des données : Le DDL

4 La création de tables

5 Abstraction logique

6 Autres ordres du DDL

## La définition des données : Le DDL

### Les types de données

Dépendants des machines, donc des SGBD

Le type alphanumérique :

CHAR (n)	Longueur fixe de n caractères. n_max : 16 383
VARCHAR(n)	Longueur variable, n représente le maximum

## La définition des données : Le DDL

Le type numérique :

NUMBER (n,[d])	Nombre de n chiffres dont d après la virgule.
SMALLINT	Mot signé de 16 bits (-32768 à 32767)
INTEGER	Double mot signé de bits (-2E31 à 2E31 -1)
FLOAT	Numérique flottant

Le type gestion du temps :

DATE	Champ date (ex 24/02/1997)
TIME	Champ heure (ex 14 :45 :10.95)
TIMESTAMP	regroupe DATE et TIME

1 Ordres de mise à jour

2 Exercice

3 La définition des données : Le DDL

4 La création de tables

5 Abstraction logique

6 Autres ordres du DDL

## La création de tables

### Forme élémentaire

```
CREATE [temp] TABLE <nom-de-TABLE> [ IF NOT EXISTS ]
( <nom-de-colonne> <TYPE de données> <contr> ,
  <contraintes de TABLE> , ....
);
```

Dans la plupart des SGBD, le nom de la table doit commencer par une lettre et on autorise 254 colonnes maximum par table.

*création de la table département avec un numéro et un nom :*

```
CREATE TABLE département
  (numdept NUMBER(3),
   nomdept CHAR(10));
```

## La création de tables

### Insertion de données à la création

```
CREATE [temp] TABLE <nom-de-TABLE> [ IF NOT EXISTS ]
( <nom-de-colonne> , ...)
  AS SELECT <nom-de-champ>, ... FROM <nom-de-TAB>
  WHERE <prédicat>;
```

Attention : seuls les types et les données sont copiées, pas les contraintes de l'autre table.

*création de la table bonus avec insertion des noms et des salaires des chefs de services de la table employé.*

```
CREATE TEMP TABLE BONUS (nom,salaire)
AS SELECT nom,salaire FROM employé
WHERE métier = 'Chef de service';
```

## La création de tables

### Expression des contraintes d'intégrité

Tous les types de contraintes (clés, domaines, intégrité) sont exprimables dans le DDL au moment du CREATE

Deux types de contraintes :

- Les contraintes de colonnes  
not null, unique, check
- Les contraintes de tables  
Primary key, foreign key

## La création de tables

### Contraintes de colonnes

- Nommer une contrainte de colonne : CONSTRAINT nom  
nom utilisé par le SGBD pour signaler les erreurs
- Définir une valeur par défaut : DEFAULT  
automatiquement introduite si l'utilisateur ne fournit pas de valeur par INSERT.  
Valeurs admises :
  - ▶ Une constante numérique ou alphanumérique
  - ▶ USER
  - ▶ NULL
  - ▶ CURRENT\_DATE, CURRENT\_TIME
  - ▶ CURRENT\_TIMESTAMP
- Oblige à des valeurs distinctes : UNIQUE

## La création de tables

### Contraintes de tables

S'appliquent en général sur plusieurs colonnes

- Nommer une contrainte de table : `CONSTRAINT nom`.
- Définition d'une clé : `PRIMARY KEY(liste de cols)`  
les colonnes qui constituent la clé primaire ne peuvent plus être nulles et chaque clé doit être unique. Création d'un INDEX
- Intégrité référentielle :  
`FOREIGN KEY (liste-col1) REFERENCES table(liste-col2)`  
Il peut bien sûr y avoir plusieurs clés étrangères dans une même table.

## La création de tables

### Contraintes d'intégrité référentielles

- Modifications automatiques à faire sur les clés étrangères en cas de changement de la clé primaire associée :  
`ON DELETE {RESTRICT | CASCADE | SET NULL | SET DEFAULT}`  
`ON UPDATE {RESTRICT | CASCADE | SET NULL | SET DEFAULT}`

Les contraintes de référence ne sont vérifiées pour un tuple que quand toutes les valeurs qui constituent la clé étrangère sont différentes de NULL.

## La création de tables

### Conventions de nommage

Une convention généralement admise consiste à nommer les contraintes d'intégrité référentielle par un nom préfixé par PK pour la clé primaire et FK pour les clés étrangères.

```
CREATE TABLE commandes
(
  fno integer,
  pno integer,
  qute integer,
  CONSTRAINT pk_commandes PRIMARY KEY (fno,pno) ,
  CONSTRAINT fk_fournisseur FOREIGN KEY (fno)
    REFERENCES fournisseurs(fno)
    ON DELETE CASCADE ,
  CONSTRAINT fk_produits FOREIGN KEY (pno)
    REFERENCES produits(pno)
    ON DELETE CASCADE
)
```

## La création de tables

### Déclenchement des contraintes

```
CREATE TABLE T1
(
  a integer,
  b integer,
  CONSTRAINT pk_T1 PRIMARY KEY (a) ,
  CONSTRAINT fk_T2 FOREIGN KEY (b)
    REFERENCES T2(b)
    ON UPDATE CASCADE
)
```

- Chaque foreign key n'est vérifiée que si non null
- En cas d'insertion sur T1 : Vérifie que b existe dans T2
- En cas de modification sur T2 : mise à jour de T1 en cascade

## La création de tables

Description d'une table

*Afficher la structure de la table Fournisseurs*

```
DESCRIBE fournisseurs;
```

## La création de tables

Script DDL de création de base

```
-- =====
--  Creation Table : Fournisseurs
-- =====

CREATE TABLE fournisseurs
(
  fno          integer NOT NULL CHECK (fno < 20)
  nom          char(20) NOT NULL,
  adresse      char(50),
  ville        char(10) DEFAULT 'Lille',
  CONSTRAINT pk_fournisseur PRIMARY KEY (fno)
)
```

## La création de tables

```
-- =====
--  Creation Table : Produits
-- =====

CREATE TABLE produits
(
  pno          integer CHECK (pno < 200) ,
  design       char(10),
  prix         integer CHECK (prix BETWEEN 1000 AND 9000)
  poids        integer CHECK (poids < 100),
  couleur      char(10) CHECK (couleur IN ('rouge', 'vert',
                                           'jaune', 'bleu', 'gris'))
  CONSTRAINT pk_produits PRIMARY KEY (pno)
)
```

## La création de tables

```
-- =====
--  Creation Table : Commandes
-- =====

CREATE TABLE commandes
(
  cno integer UNIQUE,
  fno integer,
  pno integer,
  qute integer,
  CONSTRAINT pk_commandes PRIMARY KEY (fno,pno) ,
  CONSTRAINT fk_fournisseur FOREIGN KEY (fno)
    REFERENCES fournisseurs(fno)
    ON DELETE CASCADE ,
  CONSTRAINT fk_produits FOREIGN KEY (pno)
    REFERENCES produits(pno)
    ON DELETE CASCADE
)
```

## La création de tables

```
-- =====  
--  Creation données : Produits  
-- =====  
  
INSERT INTO produits(pno,design,prix,poids,couleur)  
VALUES (102, 'fauteuil' ,1500 , 9 , 'rouge')  
  
INSERT INTO produits(pno,design,prix,poids,couleur)  
VALUES (103, 'bureau' ,3500 , 30 , 'vert')
```

1 Ordres de mise à jour

2 Exercice

3 La définition des données : Le DDL

4 La création de tables

5 Abstraction logique

6 Autres ordres du DDL

## Abstraction logique

### Principe

Fenêtre sur la base de données permettant à chacun de voir les données comme il le souhaite.

- Abstraction logique sur la base
- La vue est calculée dynamiquement (interprétée) à chaque exécution d'une requête qui y fait référence.
- Les données ne sont stockées que dans les tables d'origine.
- La définition d'une vue se fait simplement à l'aide de l'ordre `SELECT` pour sélectionner les colonnes.
- Une vue peut utiliser plusieurs tables
- il peut y avoir plusieurs vues sur une même table

## Abstraction logique

### Syntaxe

```
CREATE VIEW <nom-de-vue>  
( <nom-de-colonne> , ... )  
AS <sélection>;
```

**Remarque :** Les requêtes sauvegardées de MS-ACCESS sont des vues bien qu'elles n'en portent pas le nom.



## Abstraction logique

### Exemple

*création d'une vue sur la jointure fournisseurs-commandes ne permettant de voir que les fournisseurs avec leur total de commandes.*

```
CREATE VIEW vuefournisseur
AS SELECT f.nom,
          f.adresse,
          SUM(C.qute) AS somme
FROM fournisseurs f, commandes C
WHERE f.fno = C.fno
GROUP BY f.nom, f.adresse;
```

## Abstraction logique

### requête sur la vue ... traduction ... exécution

#### Requête sur une vue

```
SELECT * FROM vuefournisseur WHERE somme > 50 ;
```

sera traduite en ...

```
SELECT f.nom, f.adresse, SUM(qute)
FROM fournisseurs f, commandes C
WHERE f.fno=C.fno
GROUP BY f.nom, f.adresse
HAVING SUM(qute) > 50 ;
```

## Abstraction logique

### Vues adaptées aux différents utilisateurs

```
CREATE VIEW fourniss-lille
AS SELECT f.nom, f.pno, f.qute
FROM fournisseurs f , commandes C
WHERE f.fno=C.fno
AND f.ville='Lille';

CREATE VIEW fourniss-lyon
AS SELECT f.nom, f.pno, f.qute
FROM fournisseurs f , commandes C
WHERE f.fno=C.fno
AND f.ville='Lyon';
```

De cette manière, aucun commercial ne voit ni les tables réelles, ni les données de ces tables qui ne lui sont pas destinées.

## Abstraction logique

### Avantages

- Les vues permettent de rendre les programmes moins dépendants de l'évolution des tables.
- Une vue permet de restreindre l'accès d'une table à un sous-ensemble de colonnes et un sous-ensemble de lignes
- Il est possible de rassembler dans un seul objet des données "éparpillées" dans plusieurs tables.
- La programmation est simplifiée pour l'utilisateur puisqu'une partie de l'ordre SELECT est déjà codée dans la vue.
- Les vues permettent de simplifier les ordres SELECT avec sous-requêtes complexes (une vue par sous-requête).
- Les vues permettent de protéger finement l'accès aux tables en fonction des utilisateurs.

## Abstraction logique

Problèmes de Mise à jour

R	A	B
	a	1
	b	2
	c	2

S	C	D
	v	1
	x	2
	y	2
	z	3

T	A	C
	a	v
	b	x
	b	y
	c	x
	c	y

```
CREATE VIEW T
AS SELECT A, C FROM R, S
WHERE R.B=S.D ;
```

Comment répercuter la suppression du tuple (b, y) de la vue T ?

## Abstraction logique

M.A.J. autorisée si :

- La clause FROM principale ne fait référence qu'à une seule table ou une vue accessible en mise à jour.
- Elle ne comporte pas de DISTINCT ou une fonction sur colonne.
- Elle ne contient pas de clause GROUP BY ou HAVING.
- Elle n'utilise pas les opérateurs ensemblistes UNION, INTERSECT ou EXCEPT
- Elle ne contient que des références aux colonnes de la table source (pas de COUNT, SUM etc ...).
- Elle ne contient pas de sous-requête dont la clause FROM contient la même table que la clause FROM principale.

**WITH CHECK OPTION** permet de forcer la vérification de possibilité de maj

## Abstraction logique

Exemple

```
CREATE VIEW vuefournisseur
AS SELECT f.nom,
          f.adresse,
          SUM(C.qute) AS somme
FROM fournisseurs f, commandes C
WHERE f.fno = C.fno
GROUP BY f.nom, f.adresse;
```

```
UPDATE vuefournisseur
SET somme=somme*2
WHERE nom = 'DUPONT';
```

- 1 Ordres de mise à jour
- 2 Exercice
- 3 La définition des données : Le DDL
- 4 La création de tables
- 5 Abstraction logique
- 6 Autres ordres du DDL

## Autres ordres du DDL

### Optimisation des accès

- Un index permet d'accéder rapidement aux données
- Une clé primaire implique la création d'un index
- Plusieurs index par table autorisés pour une table
- La création d'index ne peut pas se faire sur une vue.
- Mis à jour automatiquement lors de chaque mise à jour d'une table.

```
CREATE [UNIQUE] INDEX <nom-de-l'INDEX>  
ON <nom-de-TABLE>  
(<Nom-de-champ> [ASC / DESC]  
 , ... );
```

## Autres ordres du DDL

### Les index

*création d'un index sur les colonnes nom et prénom de la table fournisseurs :*

```
CREATE INDEX i_founiss  
ON fournisseurs (nom ASC, prénom DESC);
```

Attention :

Un index alourdit et ralentit la saisie de données dans la base.

Eviter des index sur des champs avec beaucoup de doublons

## Autres ordres du DDL

### Modification de tables

ALTER TABLE + (ADD | MODIFY | DROP)

```
ALTER TABLE fournisseurs  
ADD prenom CHAR(15);
```

```
ALTER TABLE commandes  
MODIFY qute NUMBER(8,2);
```

```
ALTER TABLE <nom-de-TABLE>  
DROP <nom-de-colonne>;
```

## Autres ordres du DDL

### Suppression de table, de vue ou d'index

```
DROP TABLE <nom-de-TABLE>;  
DROP VIEW <nom-de-la-vue>;  
DROP INDEX <nom-de-l'INDEX>;
```

## Autres ordres du DDL

Nombreux autres objets à manier

### C'est le rôle du DBA

- (CREATE | ALTER | DROP) DATABASE
- (CREATE | ALTER | DROP) DOMAIN
- (CREATE | ALTER | DROP) FUNCTION
- (CREATE | ALTER | DROP) GROUP
- (CREATE | ALTER | DROP) LANGUAGE
- (CREATE | ALTER | DROP) INDEX
- (CREATE | ALTER | DROP) SCHEMA
- (CREATE | ALTER | DROP) SEQUENCE
- (CREATE | ALTER | DROP) TABLE
- (CREATE | ALTER | DROP) TRIGGER