

# Le Langage SQL

## Traitement des données

### **Le Langage de Manipulation de Données**

**L'ajout, la suppression et la modification de données**

A decorative graphic on the left side of the slide consists of a vertical arrangement of blue and yellow hexagons of various sizes, some solid and some outlined, creating a pixelated or molecular-like structure.

# Introduction

## ■ Les ordres SQL :

- INSERT
- UPDATE
- DELETE

## ■ Ces instructions SQL influent directement sur les données, contrairement à l'instruction SELECT

# L'insertion de données (1)

## ■ La syntaxe générale

**INSERT INTO** nom\_de\_la\_table\_cible [(liste\_des\_colonnes\_visées)] **VALUES**  
(liste\_des\_valeurs)

*[] optionnel dans la requête*

## ■ 3 types d'insertion

- Insertion explicite
- Insertion multiple
- Insertion à base de sous requête **SELECT**

# L'insertion de données (2)

## ■ Insertion simple explicite

On insère directement les valeurs dans les colonnes de la table,  
***!! insertion d'une seule ligne !!***

Exemple :

*insertion d'une ligne dans la table T\_MODE\_PAIEMENT,  
elle contient deux colonnes (code et libelle)*

```
INSERT INTO T_MODE_PAIEMENT (CODE, LIBELLE)  
VALUES('VIR','virement bancaire');
```

OU

```
INSERT INTO T_MODE_PAIEMENT  
VALUES('VIR','virement bancaire');
```

# L'insertion de données (3)

## ■ Insertion multiple

On insère plusieurs lignes dans la table cible,

**Exemple :**

*insertion de trois lignes dans la table T\_MODE\_PAIEMENT,  
elle contient deux colonnes (code et libelle)*

```
INSERT INTO T_MODE_PAIEMENT (CODE, LIBELLE)  
      VALUES      ('VIR','virement bancaire'),  
                   ('CB', 'Carte Bancaire'),  
                   ('CHQ','Chèque');
```

# L'insertion de données (4)

## ■ Insertion partiellement explicite

Exemple de création de table

```
CREATE TABLE CONNEXION
```

```
(  
    C_USER    VARCHAR(128)          NOT NULL DEFAULT 'Administrateur',  
    DATE_HEURE TIMESTAMP            NOT NULL DEFAULT CURRENT_TIMESTAMP  
);
```

```
INSERT INTO CONNEXION (C_USER, DATE_HEURE) VALUES('Dupont', DEFAULT);
```

```
INSERT INTO CONNEXION VALUES('Durant', DEFAULT);
```

```
INSERT INTO CONNEXION (C_USER) VALUES('Duchemol');
```

```
INSERT INTO CONNEXION VALUES ( DEFAULT, DEFAULT);
```

c_user	date_heure
Dupont	2015-02-05 11:18:22.654

c_user	date_heure
Dupont	2015-02-05 11:18:22.654
Durant	2015-02-05 11:19:37.743

c_user	date_heure
Dupont	2015-02-05 11:18:22.654
Durant	2015-02-05 11:19:37.743
Duchemol	2015-02-05 11:20:33.544

c_user	date_heure
Dupont	2015-02-05 11:18:22.654
Durant	2015-02-05 11:19:37.743
Duchemol	2015-02-05 11:20:33.544
Administrateur	2015-02-05 11:21:29.957

# L'insertion de données (5)

## ■ Insertion à base de sous requêtes

### Sous requête sur une autre table

*-- sous requête qui retourne une seule colonne et une seule ligne*

```
INSERT INTO CONNEXION  
VALUES( (select NOM from CLIENT where CLI_ID=4), default);
```

*-- sous requête qui retourne plus d'une ligne*

```
INSERT INTO CLIENT (CLI_ID, CODE, NOM, PRENOM)  
SELECT PRP_ID, PRP_CODE_TITRE, PRP_NOM, PRP_PRENOM  
FROM T_PROSPECT ;
```

### Sous requête sur la même table

```
INSERT INTO CLIENT  
VALUES( MAX(CLIENT .CLI_ID)+1, 'M.', 'Durant', 'Cyril') ;
```

# La suppression de données (1)

## ■ La syntaxe générale

**DELETE FROM** nom\_table\_cible

**[WHERE condition]**

**[RETURNING \*]**

**[ ]** *optionnel dans la requête*

## ■ les types de suppression

- Suppression totale
- Suppression conditionnelle
- Suppression et sous requête



# La suppression de données (2)

- **Suppression totale**

```
DELETE FROM PROSPECT;
```

- **Suppression conditionnelle**

```
DELETE FROM PROSPECT  
WHERE prenom LIKE '%A%' ;
```

- **Suppression et sous requête**

```
DELETE FROM CLIENT  
WHERE cli_id IN (SELECT prp_id FROM PROSPECT);
```

- **Avec retour des lignes supprimées**

```
DELETE FROM CLIENT  
WHERE cli_id IN (SELECT prp_id FROM PROSPECT)  
RETURNING *; -- * Pour toute les colonnes ou sinon lister les colonnes
```

# La suppression de données (3)

## ■ Autre instruction SQL

**TRUNCATE** supprime rapidement toutes les lignes d'un ensemble de tables. Elle a le même effet qu'un DELETE sans condition, mais comme elle ne parcourt pas la table, elle est plus rapide.

■ **Syntaxe :** **TRUNCATE** nomtable

■ **Exemple :**

Vide la table personne : **TRUNCATE** personne ;

Vide la table personne et adresse : **TRUNCATE** personne, adresse ;

Vide les tables en cascade : **TRUNCATE** cereale **CASCADE**;

*suppression en cascade de toutes les tables qui référencent cereale via des contraintes de clés étrangères*

# La modification de données (1)

## ■ La syntaxe générale

**UPDATE** nom\_table\_cible

**SET** colonne = valeur

[, colonne2 = valeur2 ...]

[**WHERE** condition]

[**RETURNING** \*];

[ ] *optionnel dans la requête*

## La modification de données (2)

### ■ Mise à jour sans condition

*Attention modification de ou des colonnes spécifiées sur toutes les lignes de la table*

// mise à jour du tarif

```
UPDATE TARIF
```

```
    SET  PETIT_DEJEUNE = 10 ;
```

//majoration de 15% du tarif

```
UPDATE TARIF
```

```
    SET  PETIT_DEJEUNE = PETIT_DEJEUNE * 1.15;
```

# La modification de données (3)

## ■ Mise à jour sans condition

*Attention modification de ou des colonnes spécifiées sur toutes les lignes de la table*

**UPDATE** CLIENT

**SET**    nom= **UPPER**(nom),  
         prenom = **UPPER**(prenom),  
         enseigne = **UPPER**(enseigne);

**Ou**

**UPDATE** CLIENT

**SET** (nom, prenom, enseigne) =( **UPPER**(nom), **UPPER**(prenom), **UPPER**(enseigne)) ;

# La modification de données (4)

## ■ Mise à jour conditionnelle

```
UPDATE TARIF  
SET  PETIT_DEJEUNE = PETIT_DEJEUNE * 1.15  
WHERE EXTRACT(YEAR FROM DATE_DEBUT) > 1999 ;
```

# La modification de données (5)

## ■ Mise à jour et les sous requêtes

```
UPDATE PROSPECT  
SET  nom = 'bis_' || nom  
WHERE prp_id IN (SELECT cli_id FROM CLIENT)  
RETURNING nom , prenom , email;
```


prenom	nom	email
MARY	bis_SMITH	MARY.SMITH@sakilacustomer.org
PATRICIA	bis_JOHNSON	PATRICIA.JOHNSON@sakilacustomer.org

**Modifie le nom si le l'identifiant des prospects est dans celui de client**  
**Retourne le nom et prénom des prospects modifiés**

# La modification de données (6)

## ■ Mise à jour avec sous requête corrélée

```
UPDATE PROSPECT AS P  
SET  nom= ( SELECT nom  
             FROM CLIENT  
             WHERE id_cli=P.prp_id);
```



Le nom du prospect est égal à celui du client dont l'identifiant est égal à celui du prospect courant





# FIN