

Chapitre 7 : SQL Avancé Oracle ©

INF3080 BASES DE DONNÉES (SGBD)

Guy Francoeur

Aucune reproduction sans autorisation

3 septembre 2019

UQÀM | **Département d'informatique**

- ▶ Les droits de lecture sont accordés aux étudiants inscrits au cours INF3080-030 A2019 uniquement;
- ▶ Aucun droit pédagogique ou reproduction n'est accordé sans autorisation;

1. Au dernier cours

2. SQL Avancé

- la clause WHERE

- la clause IN

- la clause BETWEEN

- la clause LIKE

- la clause GROUP BY

- la clause HAVING

- la clause ORDER BY

3. Les sous-requêtes

Table des matières

1. Au dernier cours

2. SQL Avancé

la clause WHERE

la clause IN

la clause BETWEEN

la clause LIKE

la clause GROUP BY

la clause HAVING

la clause ORDER BY

3. Les sous-requêtes

- ▶ Questions, précisions, ...

conclusion - résumé

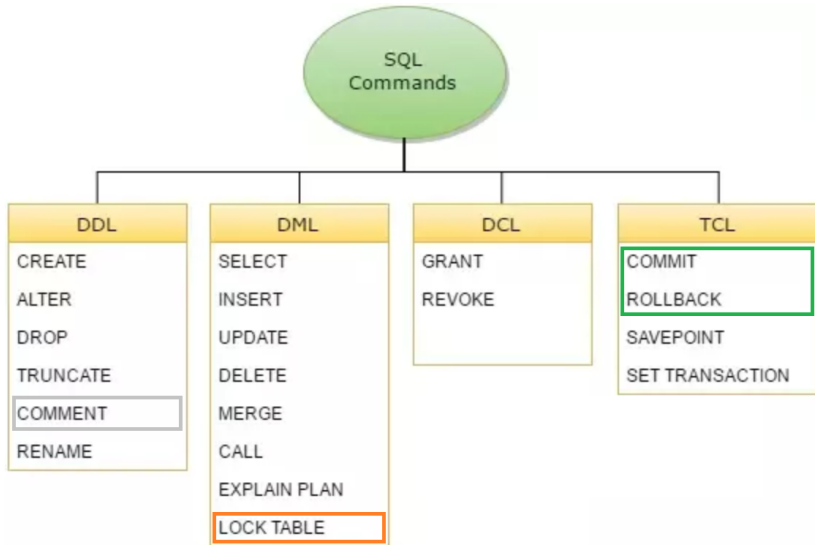


Table des matières

1. Au dernier cours

2. SQL Avancé

Contraintes

Jointures

la clause WHERE

la clause IN

la clause BETWEEN

la clause LIKE

groupement

la clause GROUP BY

la clause HAVING

tri de projection

la clause ORDER BY

3. Les sous-requêtes

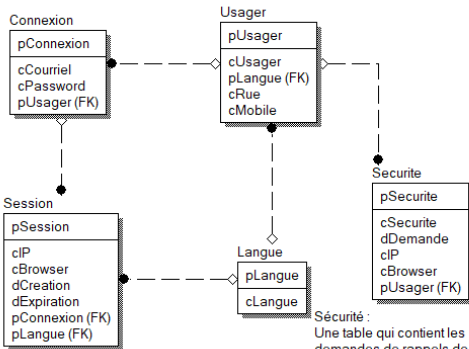
Les contraintes sont des règles qui limitent (ajoutent de la précision) les données qui seront maintenues dans une colonne.

- ▶ Clé primaire - *Primary Key*;
- ▶ Clé unique - *Unique Key*;
- ▶ Clé étrangère - *Referential integrity*;
- ▶ Validation - *Check option*;
- ▶ Base;

Contrainte - modèle entité association

Connexion:

Une table qui contient les informations pour qu'un usager puisse se connecter au système.



Session :

Une table qui garde les informations nécessaire pour garantir un usage sécuritaire du système.

Historique:

Une fois que la session est terminée, les informations seront bougées (moved) dans la table Historique. Nous maintenons l'information de façon anonyme. pLangue est aussi maintenu mais sans contrainte d'intégrité.

Sécurité :

Une table qui contient les demandes de rappels de mot de passe. Nous voulons savoir quand l'Usager a fait une demande en plus de certaines informations utiles.

Contrainte - exemple

```
SQL> CREATE TABLE Langue (  
    pLangue          NUMBER NOT NULL ,  
    cLangue          VARCHAR2(50) NULL  
);  
  
ALTER TABLE Langue  
    ADD CONSTRAINT  XPKLangue PRIMARY KEY (pLangue);  
  
CREATE UNIQUE INDEX XPKLangue ON Langue  
(pLangue    ASC);
```

```
SQL> CREATE TABLE Langue (  
    pLangue          NUMBER NOT NULL ,  
    cLangue          VARCHAR2(50) NULL ,  
    CONSTRAINT XPKLangue PRIMARY KEY (pLangue)  
);  
  
CREATE UNIQUE INDEX XPKLangue ON Langue  
(pLangue    ASC);
```

Contrainte intégrité et validation - exemple

Version avec ALTER TABLE ... ADD CONSTRAINT

```
SQL> CREATE TABLE Usager (  
  pUsager          NUMBER NOT NULL ,  
  cUsager          VARCHAR2(50) NULL ,  
  pLangue          NUMBER NULL ,  
  cRue             VARCHAR2(50) NULL ,  
  cMobile          VARCHAR2(50) NULL ,  
  bActif           NUMERIC(1) NOT NULL CONSTRAINT  
    Validation_Rule_234 CHECK (bActif IN (0, 1)),  
  CONSTRAINT XPKUsager PRIMARY KEY (pUsager)  
);  
  
ALTER TABLE Usager  
  ADD (CONSTRAINT R_4 FOREIGN KEY (pLangue) REFERENCES Langue (  
    pLangue));
```

Contrainte intégrité et validation - exemple

- ▶ La contrainte d'intégrité lors de la création de la table.
- ▶ La contrainte de validation est associée à une colonne lors de la création de la table.

```
SQL> CREATE TABLE Usager (  
  pUsager          NUMBER NOT NULL ,  
  cUsager          VARCHAR2(50) NULL ,  
  pLangue          NUMBER NULL , -- Attention  
  cRue             VARCHAR2(50) NULL ,  
  cMobile          VARCHAR2(50) NULL ,  
  bActif           NUMERIC(1) NOT NULL CONSTRAINT --Attention  
    Validation_Rule_234 CHECK (bActif IN (0, 1)),  
  CONSTRAINT XPKUsager PRIMARY KEY (pUsager),  
  CONSTRAINT R_4 FOREIGN KEY (pLangue) REFERENCES Langue (pLangue)  
);
```

filtres, jointure - la clause WHERE

- ▶ Utilisé pour lier des tables;
- ▶ Utilisé pour filtrer des données;
- ▶ Pourquoi est-il utile d'appliquer un filtre ?

```
SQL> SELECT pLangue FROM Langue WHERE pLangue=1;
```

```
SQL> SELECT u.* FROM Usager u, Langue l  
WHERE u.pLangue = l.pLangue  
AND pLangue=1;
```

filtres, jointure - la clause IN

La clause la plus simple qui exprime très bien la théorie des ensembles.

```
SQL> SELECT u.*  
FROM Usager u  
WHERE pLangue IN (2,3);
```

```
SQL> SELECT u.*  
FROM Usager u  
WHERE pLangue NOT IN (2,3);
```

filtres, jointure - la clause WHERE

Un filtre avec la clause BETWEEN

- ▶ Utilisé pour définir des intervalles;
- ▶ Utilisé avec les type date ou entier;

```
SQL> SELECT u.cUsager, l.cLangue
FROM Usager u, Langue l
WHERE u.pLangue = l.pLangue
AND l.pLangue BETWEEN 1 AND 3;
```

filtres, jointure - la clause WHERE

Un filtre avec la clause LIKE

- Utilisé avec les chaînes de caractères;

```
SQL> SELECT u.cUsager, l.cLangue
FROM Usager u, Langue l
WHERE u.pLangue = l.pLangue
AND cLangue LIKE 'A%';
```


Sommaire des opérateurs qui sont utilisés

- ▶ IN, NOT, LIKE, BETWEEN
- ▶ >, <, <=, >=, =, <>
- ▶ IS [NOT] NULL, IS [NOT] TRUE | FALSE
- ▶ AND, OR

groupement - la clause GROUP BY

- ▶ Utilisé pour grouper les données afin de faire des sommaires;
- ▶ Pourquoi est-il utile d'utiliser le groupement ?

```
SQL> SELECT cLangue, sum(1) C1, count(*)  
FROM Usager u, Langue l  
WHERE u.pLangue = l.pLangue  
GROUP BY cLangue;
```

```
SQL> SELECT cLangue, sum(1) C1, count(*)  
FROM Usager u, Langue l  
WHERE u.pLangue = l.pLangue  
GROUP BY cLangue  
HAVING count(*) > 1;
```

groupement - la clause GROUP BY

- ▶ Utilisé afin d'ajouter un filtre sur une fonction de groupement;
- ▶ Joue un rôle similaire à la clause WHERE;

```
SQL> SELECT cLangue, sum(1) C1, count(*)  
FROM Usager u, Langue l  
WHERE u.pLangue = l.pLangue  
GROUP BY cLangue  
HAVING count(*) > 1;
```

tri - la clause ORDER BY

- ▶ Utilisé pour ordonner notre projection selon un ordre précis;
- ▶ Il est possible d'utiliser l'index dans la projection;

```
SQL> SELECT cLangue, sum(bActif), count(*)  
FROM Usager u, Langue l  
WHERE u.pLangue = l.pLangue  
GROUP BY cLangue  
HAVING count(*) > 1  
ORDER BY 3;
```

```
SQL> SELECT cLangue, sum(bActif), count(*)  
FROM Usager u, Langue l  
WHERE u.pLangue = l.pLangue  
GROUP BY cLangue  
HAVING count(*) > 1  
ORDER BY cLangue, 3;
```

1. Au dernier cours

2. SQL Avancé

- la clause WHERE

- la clause IN

- la clause BETWEEN

- la clause LIKE

- la clause GROUP BY

- la clause HAVING

- la clause ORDER BY

3. Les sous-requêtes

sous-requêtes - avec IN

```
SQL> SELECT u.*  
FROM Usager u  
WHERE pLangue IN (  
    SELECT pLangue  
    FROM Langue  
    WHERE cLangue IN ('Francais','Anglais')  
);
```

```
SQL> SELECT u.cUsager, l.cLangue
FROM Usager u,
      (SELECT pLangue, cLangue FROM Langue WHERE pLangue > 1) l
WHERE u.pLangue=l.pLangue;
```