CHAPITRE 2.3: INTERROGATION DES BASES DE DONNÉES

Introduction à SQL

LANGAGE DE REQUETES RELATIONNEL

Le langage SQL

SQL = Structured Query Langage

- Langage standard - 4ème génération (SQL89, SQL2, SQL3)

- Langages de requêtes SQL pour BdD relationnelles
  - LANGAGE DE DESCRIPTION DE DONNÉES (LDD)
    - Création de tables, contraintes, modification, …
  - LANGAGE DE MANIPULATION DE DONNÉES (LMD)
    - Interrogation/Modification de la BdD

SQL

- **Langage de Description de Données**
  - Instructions: CREATE, DROP, ALTER TABLE
- **Langage de Manipulation de Données**
  - Instructions: SELECT, INSERT, DELETE, UPDATE

Introduction à SQL

DÉFINITION DES DONNÉES LDD

DÉFINITION DU SCHEMA DE LA BDD

Création de table

Une *table* (*relation* en algèbre relationnelle) se créé par l'instruction CREATE TABLE

CREATE TABLE *nom_table*
(
nom_col1 *type_col1* [*contrainte1*]
[, *nom_col2* *type_col2* [*contrainte2*] …]

);

Contraintes sur les colonnes

- PRIMARY KEY

  Clé primaire de la table. Pour désigner plus d'une colonne, on l'utilise comme une clause PRIMARY KEY(col1, col2, …) à côté des définitions de colonnes

- NOT NULL

  Doit avoir une valeur pour chaque enregistrement

- UNIQUE

  Chaque enregistrement doit avoir une valeur différente

- CHECK *condition*

  La valeur doit respecter une condition donnée

Exemple de création de table

```
CREATE TABLE Etudiant
    (
    Matricule INT PRIMARY KEY,
    Nom VARCHAR(30) NOT NULL,
    Prénom VARCHAR(30) NOT NULL
    DateNais  date,
    Résidence VARCHAR(30),
    Spécialité VARCHAR(15) CHECK (type IN ('Tcommun', 'management', 'comptabilité','autres')
    );
```

Suppression & renommage d'une table

Une *table* est supprimée par l'instruction DROP TABLE

DROP TABLE *nom_table*;

Une *table* est renommée par l'instruction RENAME TABLE

RENAME TABLE *ancien_nom* TO *nouveau_nom*;

Modification d'une table

Le contenu d'une table peut être modifié par l'instruction ALTER TABLE

ALTER TABLE *nom_table*
modification1,
modification 2…;

Les modifications portent sur des ajouts, suppressions ou remplacements de colonnes avec pour syntaxe :

ADD (*nom_col type_col* [*contrainte*])

DROP *nom_col*

MODIFY *nom_col type_col* [*contrainte*]

Exemple de modification de table

```
ALTER TABLE ETUDIANT
    ADD (Université VARCHAR(30)),
    MODIFY Nom VARCHAR(20) NOT NULL,
    DROP DateNais;
```

Introduction à SQL

LANGAGE DE *MANIPULATION DES DONNÉES* *LMD*

Insertion d'enregistrement (TUPLE)

Un *enregistrement* (*tuple* en algèbre relationnelle) s'insère dans une table par l'instruction INSERT INTO

INSERT INTO *nom_table*[*(nom_col1, nom_col2, …)*]
VALUES (*val1*, *val2*, …)
[(*val3*, *val4*, …)];

Exemple :

INSERT INTO ETUDIANT
VALUES (1, ' Hamidi', 'Omar', 19/08/1989, 'alger', 'autre'),
(2, ' Achouri', 'Karim', 12/08/1988, 'Ourgla', 'Tcommun');

Suppression d'enregistrement

Un ou plusieurs enregistrements se suppriment par l'instruction DELETE

DELETE FROM *nom_table*
WHERE *conditions*;

conditions exprime un ensemble de condition pour sélectionner (à la manière de l'opérateur relationnel de sélection) les enregistrements à effacer.

Exemple :

```
DELETE FROM Client
        WHERE nom = ' Messoudi';
```

Modification d'enregistrement

Un ou plusieurs enregistrements se modifient par l'instruction UPDATE

UPDATE *nom_table*
SET *nom_col1 = expr1* [, *nom_col2 = expr2*]…
WHERE *conditions*;

conditions exprime un ensemble de condition pour sélectionner les enregistrements à modifier et les clauses SET indiquent les modifications à effectuer.

Exemple :

```
UPDATE Client
    SET spécialité = 'comptabilité'
    WHERE nom = ' Hamidi ';
```

Requête SELECT

Une requête d'interrogation de la BdD s'écrit à l'aide de l'instruction SELECT ….Consultation

SELECT [DISTINCT] *table1.attr1* [, *table2.attr2*]… | *expr*
FROM *table1* [, *table2*]…
[WHERE *condition*]
[ORDER BY *expr* [DESC]]
[GROUP BY *expr*]
[HAVING *expr*]

Tables des exemples - ÉTUDIANT

ÉTUDIANT

matricule	Nom	Prénom	AnnéeNais	résidence	Université
0012/06	Betouche	Malek	1990	Alger	ESC
0045/07	Trabelssi	Ismail	1988	boumerdes	USTHB
0124/05	Guessoum	Lila	1989	Medea	ESC
0166/07	Bouchardi	Ahmed	1992	Mascara	ESI
0485/07	Dilmi	Ahmed	1991	Adrar	ESC
0120/08	Ziadi	Mounir	1988	Alger	USTHB
0225/08	Othmani	Djamila	1990	Blida	ESI

Interrogation de la BdD

Projection

immatriculation	*marque*	*proprietaire*
3452 AZ 13	BMW	13
9835 EI 07	Renault	76
1234 WW 42	Peugeot	5
9878 DG 94	BMW	76

$\Pi_{\{immatriculation,\ proprietaire\}}$

immatriculation	*proprietaire*
3452 AZ 13	13
9835 EI 07	76
1234 WW 42	5
9878 DG 94	76

Sélection

immatriculation	*marque*	*proprietaire*
3452 AZ 13	BMW	13
9835 EI 07	Renault	76
1234 WW 42	Peugeot	5
9878 DG 94	BMW	76

$\sigma_{proprietaire=76}$

immatriculation	*marque*	*proprietaire*
9835 EI 07	Renault	76
9878 DG 94	BMW	76

Requêtes de projection

L'ensemble des enregistrements d'une table est désigné par le symbole *.

```
SELECT * FROM ETUDIANT;
```

Une projection se réalise en précisant les colonnes à conserver.

```
SELECT Etudiant.prenom, Etudiant.nom FROM Etudiant;
```

DISTINCT ne conserve pas les doublons.

```
SELECT DISTINCT Etudiant.prenom, Etudiant.nom FROM Etudiant;
```

Projection sur plusieurs champs

SELECT Nom, Prénom, Université FROM ÉTUDIANT

Nom	Prénom	Université
Betouche	Malek	ESC
Trabelssi	Ismail	USTHB
Guessoum	Lila	ESC
Bouchardi	Ahmed	ESI
Dilmi	Ahmed	ESC
Dilmi	Mounir	USTHB
Othmani	Djamila	ESI

- La liste des étudiants avec l'Université où ils sont inscrits

Requêtes de sélection

Les conditions d'une sélection s'expriment après la clause WHERE

```
SELECT * FROM Etudiant WHERE nom=‘ Messoudi ’;
```

Plusieurs conditions peuvent être exprimées et combinées par les opérateurs logiques OR et AND

```
SELECT * FROM Etudiant
        WHERE (spécialité=‘autre’)
        AND ((nom=‘Messoudi’) OR (nom=‘Hamidi’));
```

Sélection – Exemples avec 2 critères (OU et ET)

SELECT * FROM ÉTUDIANT WHERE

[AnnéeNais = 1991] AND [Université = 'ESC']

matricule	Nom	Prénom	AnnéeNais	résidence	Université
0485/07	Dilmi	Ahmed	1991	Adrar	ESC

SELECT * FROM ÉTUDIANT WHERE

(Résidance= ' boumerdes') OR (Université = 'ESI')

matricule	Nom	Prénom	AnnéeNais	résidence	Université
0045/07	Trabelssi	Ismail	1988	boumerdes	USTHB
0225/08	Othmani	Djamila	1990	Blida	ESI
0166/07	Bouchardi	Ahmed	1992	Mascara	ESI

Expression des conditions

- Opérateurs simples de comparaison

  =, !=, <, >, <=, >=

- Autres opérateurs
  - expr1 BETWEEN expr2 AND expr3

    Ex : salaire BETWEEN 5000 AND 10000
  - expr1 IN (expr2, expr3, …)

    Ex : nom IN (‘Malki’, ‘Hamidi’)
  - expr1 LIKE chaîne

    Ex : nom LIKE H%

Ordre du résultat

Le résultat d'une requête peut être trié selon les valeurs d'une ou plusieurs colonne(s).

Par ordre croissant :

```
SELECT * FROM Etudiant ORDER BY Annéenais;
```

Par ordre décroissant (ici le tri est d'abord fait par le nom puis par le prenom pour les enregistrements de même nom) :

```
SELECT * FROM Etudiant ORDER BY nom, prenom DESC;
```

Jointure

- Une jointure s'exprime par une sélection couvrant plusieurs tables
- Il suffit de tester l'égalité d'attributs de 2 tables différentes

```
SELECT * FROM Vehicule, Proprietaire
        WHERE Vehicule.proprietaire = Proprietaire.numero;
```

Interrogation de la BdD

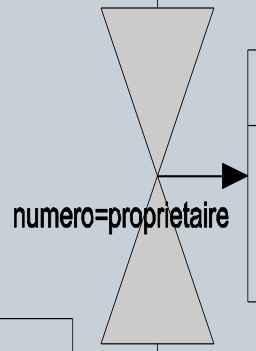
Jointure naturelle

R

immatriculation	*marque*	*proprietaire*
3452 AZ 13	BMW	13
9835 EI 07	Renault	76
1234 WW 42	Peugeot	5
9878 DG 94	BMW	76

R'

numero	*nom*	*prenom*	*naissance*
5	Martin	Paul	4 février 1980
13	Dupont	Marcel	15 mai 1976
76	Durand	Pierre	29 novembre 1978



immatriculation	*marque*	*proprietaire*	*numero*	*nom*	*prenom*	*naissance*
3452 AZ 13	BMW	13	13	Dupont	Marcel	15 mai 1976
9835 EI 07	Renault	76	76	Durand	Pierre	29 novembre 1978
1234 WW 42	Peugeot	5	5	Martin	Paul	4 février 1980
9878 DG 94	BMW	76	76	Durand	Pierre	29 novembre 1978

Opérateurs de groupe

SUM (*attribut*)	Renvoie la somme des valeurs de l'attribut
COUNT(*)	Renvoie le nombre d'enregistrements
MAX(*expr*)	Renvoie la valeur maximale d'une expression (qui peut être un attribut)
MIN(*expr*)	Renvoie la valeur minimale d'une expression (qui peut être un attribut)
AVG(*attribut*)	Renvoie la moyenne des valeurs de l'attribut

Exemples d'opérateurs de groupe

- SELECT COUNT(*) FROM Vehicule;
- SELECT COUNT(*) FROM Vehicule WHERE marque='BMW ';
- SELECT MAX(solde) FROM Client;
- SELECT SUM(solde) FROM Client WHERE nom = 'Durand ';

Groupement des résultats

- Par défaut les opérateurs de groupe s'appliquent sur l'ensemble des enregistrements sélectionnés
- L'instruction GROUP BY permet d'appliquer un opérateur à des groupes séparés d'enregistrements
- Par exemple, pour obtenir le nombre de vehicule de chaque marque on peut écrire :

```
SELECT COUNT(*) FROM Vehicule GROUP BY marque;
```

Clause HAVING

- La clause HAVING réalise une sélection (à la manière du WHERE) sur les groupes à retenir.
- Par exemple, si on souhaite connaître le nombre de véhicules de chaque marque en ayant plus de 5, on peut écrire :

```
SELECT COUNT(*) FROM Vehicule GROUP
    BY marque HAVING COUNT(*) > 5;
```

Sous-interrogations

- Un critère de recherche employé dans une clause WHERE peut être lui-même le résultat d'un SELECT
- Par exemple, pour connaître le nom des personnes nées le même jour que le client « Martin » on peut écrire :

```
SELECT nom FROM Client
   WHERE naissance = (SELECT naissance FROM Client

         WHERE nom = ' Martin ');
```

Sous-interrogations (2)

- Dans le cas où plusieurs enregistrements peuvent être renvoyés on peut utiliser :
  - IN : pour tester une appartenance à l'ensemble de ce qui est renvoyé
  - Un opérateur de comparaison (<, >, ...) suivi de ANY ou de ALL pour tester si la comparaison est vraie au moins une fois ou pour tous les enregistrements.

- Par exemple pour connaître le client le plus jeune on peut écrire :

```
SELECT nom FROM Client
   WHERE naissance <= ALL (SELECT naissance
         FROM Client);
```

Insertion utilisant une sélection

- Une requête imbriquée peut-être utilisée pour réaliser des insertions.
- Par exemple, pour insérer dans la table 2A tous les élèves de la table 1A dont l'attribut note est supérieur à 10, on peut écrire :

```
INSERT INTO 2A
                SELECT * FROM 1A
                    WHERE note > 10;
```

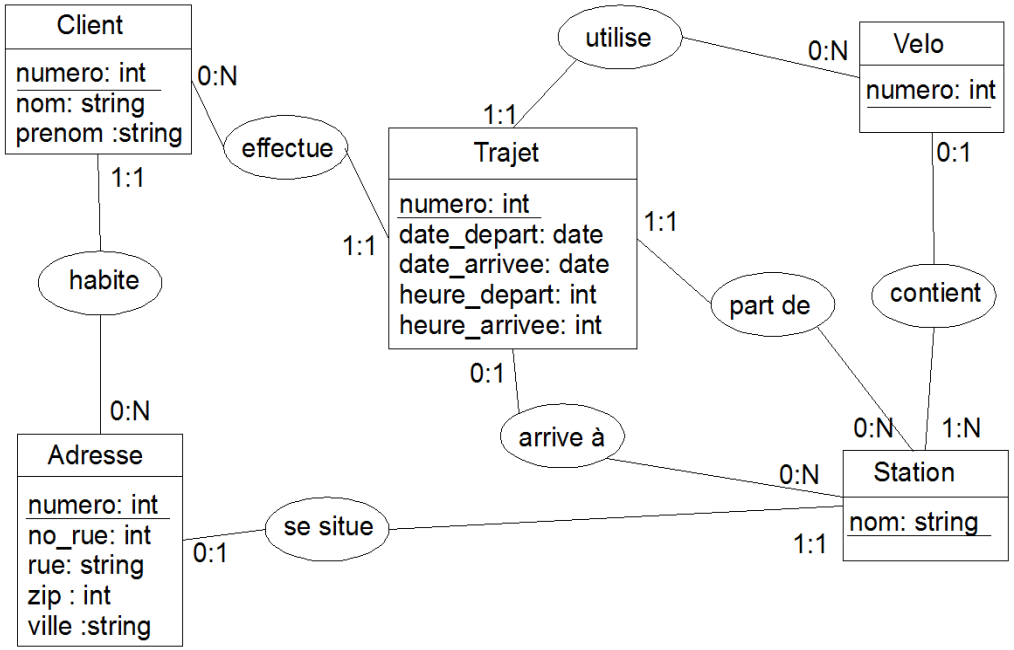
Les instructions UPDATE et DELETE peuvent aussi utiliser une requête imbriquée

Création utilisant une sélection

- Une requête imbriquée peut-être utilisée pour créer et remplir une table.
- Par exemple, pour une table 2A comprenant les colonnes nom et prenom et remplie avec tous les noms et prenoms des enregistrements d'une rable 1A, on peut écrire :

```
CREATE TABLE 2A
        AS SELECT nom, prenom FROM 1A;
```

Introduction à SQL



En relationnel :

Client (<u>numero: int</u>, nom: string, prenom: string, no_adresse: int)

Adresse (<u>numero: int</u>, no_rue: int, rue: string, zip: int, ville: string)

Station (<u>nom: string</u>, no_adresse: int)

Velo(<u>numero: int</u>, nom_station: string)

Trajet(<u>numero: int</u>, no_client: int, no_velo: int, date_depart: Date, heure_depart: int, station_depart: int, date_arrivee: Date, heure_arrivee: int, station_arrivee: int)

Requêtes en algèbre relationnelle

- l'ensemble des noms des stations

$$\prod_{\{nom\}}(Station)$$

- le prénom du client dont le nom est « Martin »

$$\prod_{\{prenom\}}(\sigma_{\{nom=\text{« Martin »}\}}(Client))$$

- les identifiants des vélos situés sur la station nommée « Gare SNCF »

$$\prod_{\{numero\}}(\sigma_{\{nom\_station=\text{« Gare SNCF »}\}}(\text{ Velo}))$$

Requêtes en algèbre relationnelle

- le nom des personnes qui habitent à la même adresse que la station nommée « Gare SNCF »

  $\prod_{\{nom\}}(Client \mid><\mid_{\{no\_adresse=no\_adresse\}} (\sigma_{\{nom=« Gare SNCF »\}}(Station)))$

- le nom des personnes ayant utilisé le vélo numero 2 entre le 10/12/1980 et le 01/01/1990

  $\prod_{\{nom\}}((\sigma_{\{no\_velo=2,\ 10/12/1980 < date\_depart < 01/01/1990\}}(Trajet)))$

Requêtes en SQL

- l'ensemble des noms des stations

```
SELECT Station.nom
FROM Station;
```

- le prénom du client dont le nom est « Martin »

```
SELECT Client.prenom FROM Client
        WHERE (Client.nom = "martin");
```

- les identifiants des vélos situés sur la station nommée « Gare SNCF »

```
SELECT Velo.numero FROM Velo
WHERE (Velo.nom_station = "Gare SNCF");
```

Requêtes en SQL

- le nom des personnes qui habitent à la même adresse que la station nommée « Gare SNCF »

```
SELECT Client.nom FROM Client, Station
WHERE Client.adresse = Station.no_adresse
AND Station.nom = "Gare SNCF";
```

- le nom des personnes ayant utilisé le vélo numero 2 entre deux le 10/12/1980 et le 01/01/1990

```
SELECT Client.nom FROM Client, Trajet
WHERE Client.numero = Trajet.no_client
AND Trajet.no_velo = 2
AND Trajet.date_depart>#10/12/1980#
AND Trajet.date_depart<#1/1/1990#;
```

Des questions ?