Bases de données

L2 sciences et technologies, mention informatique

SQL

ou : le côté obscure de la jolie théorie

films	titre	réalisateur	année
	starwars	lucas	1977
	nikita	besson	1990
	locataires	ki-duk	2005
	dune	lynch	1984

realisateurs	nom	nationalité
	lucas	américaine
	lynch	américaine
	besson	française
	ki-duk	coréenne

patrick.marcel@univ-tours.fr
http://celene.univ-tours.fr/course/view.php?id=3131

plan du cours

étude d'un langage standard du commerce pour

- définir les données (schéma, instance)
- exprimer des requêtes conjonctives
- exprimer des requêtes de l'algèbre relationnelle
- exprimer des requêtes avec agrégation, récursivité...

SGBD (DBMS)

SGBD : Système de Gestion de Bases de Données

- gestion des utilisateurs
- organisation physique des données sur disque
- gestion des accès concurrents
- reprise après pannes
- ...
- définition et mise à jour des données
- évaluation/optimisation de requêtes

SQL

Structured Query Language (IBM 1974)

basé sur l'algèbre relationnelle

normalisation ISO :

- SQL1 (86-87, révisé en 89)
- SQL2 (92)
- SQL3 ou SQL:1999 (99)
- SQL:2003
- SQL:2006
- SQL:2008

SQL est...

un langage de définition de données

Data Definition Language (DDL)

- création, suppression, modification, typage, ...
- de bases de données, de relations, ...

définition d'une base de données

création d'un nom de base de données

```
CREATE DATABASE cinema;
USE cinema;
```

on peut aussi...

```
SHOW DATABASES;
SHOW TABLES;
DROP DATABASE cinema;
```

définition de relation

définition d'un schéma de relation

```
CREATE TABLE films (titre VARCHAR(20),
                    realisateur VARCHAR(20),
                    annee YEAR);
```

on peut aussi...

```
DESCRIBE films;

ALTER TABLE films ADD COLUMN budget INTEGER;

DROP TABLE films;
```

définition d'instances

```
INSERT INTO films VALUES('starwars','lucas',1977);

LOAD DATA LOCAL INFILE 'film_data.txt'
INTO TABLE films
FIELDS TERMINATED BY '|';
```

modification et suppression d'instances

```
DELETE FROM films WHERE titre='starwars';

UPDATE films
SET titre='starwars4'
WHERE realisateur='lucas'
AND annee=1977;
```

attention !

on parle de *tables* plutôt que de relations

- une table est un *multiensemble* :
- les *duplicats* sont autorisés

on parle de *lignes* (row) plutôt que de tuples

dans la suite...

CREATE TABLE film(titre VARCHAR(20), realisateur VARCHAR(20), acteur VARCHAR(20));

CREATE TABLE salle(nom VARCHAR(20), titre VARCHAR(20), horaire TIME);

CREATE TABLE produit(producteur VARCHAR(20), titre VARCHAR(20));

CREATE TABLE vu(spectateur VARCHAR(20), titre VARCHAR(20));

CREATE TABLE aime(spectateur VARCHAR(20), titre VARCHAR(20));

SQL est aussi...

un langage de manipulation de données/de requêtes

Data Manipulation Language/ Query Language

privilégiant les requêtes conjonctives

requêtes conjonctives

SELECT	liste d'attributs à sélectionner
FROM	liste de noms de tables
WHERE	condition

exemple

```
SELECT titre
FROM film
WHERE realisateur='lucas';
```

exemple

```
SELECT film.titre,nom
FROM salle, film
WHERE film.titre = salle.titre
AND film.realisateur='lucas';
```

correspondance

donc :
SELECT correspond à π
FROM correspond à $\times$
WHERE correspond à σ

equi-jointure

```
SELECT salle.titre,nom
FROM salle JOIN film ON salle.titre=film.titre;
```

jointure naturelle

```
SELECT titre,nom
FROM salle NATURAL JOIN film;
```

formules de sélection

sont utilisables dans WHERE:

- AND, OR, ...
- comparateurs $=$,$!=$, $>$, $>=$,...
- IN, NOT IN, EXISTS, NOT EXISTS, ...

exemple

```
SELECT titre, nom
FROM salle NATURAL JOIN film
WHERE titre = 'starwars'
OR horaire > '18:00';
```

on peut aussi...

SELECT * FROM film;

* représente la liste de tous les attributs de la table

on peut aussi...

SELECT * FROM film;

* représente la liste de tous les attributs de la table

SELECT DISTINCT titre FROM film;

permet d'éliminer les duplicats

opérateurs ensemblistes

union, intersection, différence :

```
SELECT realisateur FROM film UNION SELECT acteur FROM film;
SELECT realisateur FROM film EXCEPT SELECT acteur FROM film;
SELECT realisateur FROM film INTERSECT SELECT acteur FROM film;
```

renommage

plusieurs cas d'utilisation :

1. préfixer avec le nom de la table
2. utiliser une variable
3. utiliser AS

préfixer avec le nom de la table

```
SELECT film.titre,nom
FROM salle, film
WHERE film.titre = salle.titre
AND film.realisateur='lucas';
```

permet de lever l'ambiguité sur les noms d'attribut

utiliser des variables

```
SELECT M1.acteur, M2.acteur
FROM film M1, film M2
WHERE M1.titre = M2.titre
AND M1.realisateur = M2.realisateur;
```

permet de lever l'ambiguité sur les noms d'attribut

AS

```
SELECT film.acteur AS acteur_producteur
FROM film, produit
WHERE acteur = producteur
AND film.titre = produit.titre;
```

requêtes imbriquées

```
SELECT nom
FROM salle
WHERE titre IN
(SELECT titre
FROM film
WHERE realisateur = 'lucas');
```

requêtes imbriquées

```
SELECT S.nom
FROM (SELECT * FROM salle WHERE horaire = '20:00') S
WHERE titre IN
(SELECT titre
FROM film
WHERE realisateur = 'lucas');
```

opérations ensemblistes

- avec les opérateurs ensemblistes UNION, INTERSECT, EXCEPT
- ou avec les IN, NOT IN dans le WHERE
  pour des requêtes imbriquées

intersection

```
SELECT realisateur
FROM film
WHERE realisateur IN
(SELECT producteur FROM produit);
```

différence

```
SELECT spectateur
FROM vu
WHERE spectateur NOT IN
(SELECT spectateur
FROM aime);
```

division

quel spectateur a vu tous les films ?

```
SELECT DISTINCT spectateur FROM vu WHERE
spectateur NOT IN
(SELECT spectateur FROM vu,film
WHERE (spectateur,film.titre)
NOT IN (SELECT spectateur,titre FROM vu));
```

vues

une vue est

- une table temporaire (non stockée sur disque)
- définie par une requête
- accessible comme toute autre table

vues

```
CREATE VIEW salle_film_non_lucas AS
(SELECT nom
FROM salle, film
WHERE salle.titre = film.titre
AND realisateur != 'lucas');
```

vues et nr-datalog¬

exemple : quels sont les salles qui ne proposent que des films de 'lucas' ?

```
SELECT nom
FROM salle
WHERE nom NOT IN
(SELECT * FROM salle_film_non_lucas);
```

conclusion (1)

SQL permet d'exprimer toutes les requêtes exprimables en algèbre relationnelle

agrégation

```
ALTER TABLE film ADD COLUMN budget integer;
SELECT realisateur, avg(budget)
FROM film
GROUP BY realisateur;
```

de nombreuses fonctions d'agrégation sont disponibles : avg, count, sum, ...

having

```
SELECT realisateur, avg(budget) AS budget_moyen
FROM film
GROUP BY realisateur
HAVING budget_moyen > 1000000;
```

order by

```
SELECT realisateur, avg(budget) AS budget_moyen
FROM film
GROUP BY realisateur
HAVING budget_moyen > 1000000
ORDER BY budget_moyen DESC;
```

récursivité, depuis SQL3

exemple : sachant la table *enfant* je cherche les ancêtres de 'george'

```
WITH
RECURSIVE ancetre(ascendant,descendant) AS
        (SELECT * FROM enfant)
        UNION
        (SELECT ad1.ascendant,ad2.descendant
        FROM ancetre ad1, ancetre ad2
        WHERE ad1.descendant = ad2.ascendant)
SELECT ascendant
FROM ancetre
WHERE descendant = 'george';
```

conclusion (2)

SQL

- fait tout ça... et bien plus encore !
- évolue encore... à suivre

conclusion (3)

SQL est un standard... comme tous les standards il donne lieu à différentes implémentations

le SQL de MySQL

- n'a pas d'opérateur EXCEPT ou INTERSECT
- ne permet pas les requêtes récursives
- ...