Licence Sciences et Technologies
Mention Informatique

Conception de Bases de Données

Introduction

Verónika Peralta

2021-2022

Plan

- **Concepts de base**
  - SI, BD et SGBD
  - Modèles de données, schémas et instances
  - Méthodologies de conception
- **Rappel du modèle relationnel**
  - Structures de données
  - Opérations d'interrogation : SQL-QL
- **Contraintes dans le modèle relationnel**
  - Contraintes
  - Commandes de mise à jour : SQL-DML
  - Commandes de définition de données : SQL-DDL

Information et entreprises

- **Toute entreprise crée de la valeur en traitant de l'information :**
  - Celles qui ont pour activité principale le traitement de l'information :
    - Ex. compagnies d'assurance, banques, compagnies financières…
  - Celles qui gèrent d'autres types de produits/services mais qui guident leurs décisions par l'information :
    - Ex. usines, fournisseurs, compagnies de livraison, sociétés de services…
- **L'information possède une valeur d'autant plus grande qu'elle contribue à atteindre les objectifs de l'organisation.**

Systèmes d'information (SI)

- **Un SI représente l'ensemble des éléments participant à la gestion, au traitement, au transport et à la diffusion de l'information au sein de l'organisation :**
  - C'est un ensemble organisée de ressources :
    - personnel, données, procédures, matériel, logiciel, …
  - permettant :
    - d'acquérir, de stocker, de structurer et de communiquer des informations,
  - sous forme de :
    - textes, images, sons ou de données codées.

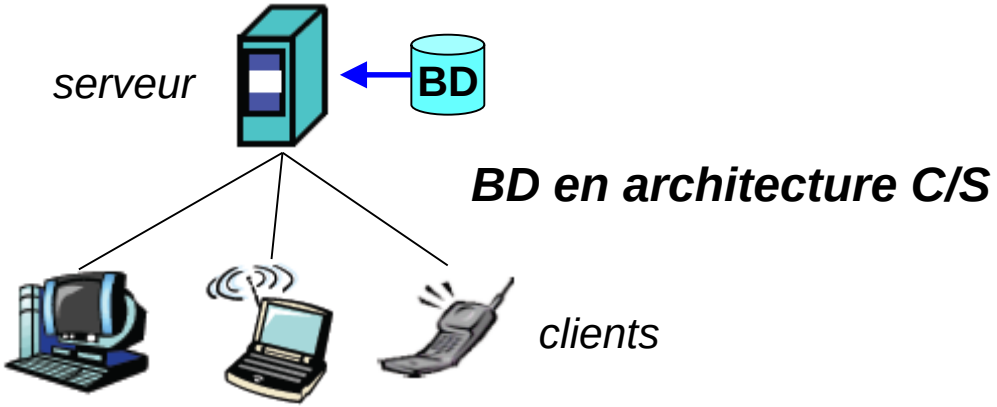
Bases de données (BD)

- **Entité dans laquelle il est possible de stocker des données de façon structurée et avec le moins de redondance possible.**

- **Ces données doivent pouvoir être utilisées par des programmes et des utilisateurs différents.**
  - Ainsi, la notion de base de données est généralement couplée à celle de réseau, afin de pouvoir partager, diffuser, mais également protéger ces informations.

Bases de données et réseaux

BD locale



BD en architecture C/S

- **Une base de données peut être :**
  - Locale :
    - La BD est stocké sur la machine d'un utilisateur.
  - Distante (architecture client/serveur) :
    - La BD est stocké sur un serveur.
    - Plusieurs utilisateurs, sur plusieurs machines distantes, accèdent aux données via des réseaux.
- **L'avantage majeur de l'utilisation de BD C/S est la possibilité pour plusieurs utilisateurs d'accéder simultanément aux données.**

Systèmes de gestion de bases de données (SGBD)

- **Logiciel responsable de la gestion de BD :**
  - Exécution des requêtes.
  - Maintenance des données, de leur cohérence.
  - Contrôle des utilisateurs et des actions…
- **Objectifs d'un SGBD :**
  - Permettre la création de nouvelles bases de données, en définissant leur schéma.
  - Permettre d'ajouter des données, de modifier les données, d'interroger la base de façon simple.
  - Gérer le stockage de très grandes quantités de données de façon efficace et sécurisée face aux pannes, aux accès non autorisés, etc.
  - Permettre l'accès simultané de plusieurs utilisateurs à la base.

Utilisateurs des bases de données

- **Utilisateurs finaux** (end-users) : vous & moi, applications :
  - Requêtes
  - Ajouts, suppressions, modifications des données
- **Concepteurs-développeurs :**
  - Requêtes
  - Ajouts, suppressions, modifications des données
  - Modifications d'objets
- **Administrateurs (DBA) :**
  - Gestion de tous les objets
  - Performance
  - Sécurité

Modèles de données

- **Qu'est-ce qu'un modèle de données :**
  - Abstraction mathématique selon laquelle l'utilisateur voit les données.
  - Abstraction qui permet d'étudier des propriétés et tirer des conclusions.
  - Langage pour spécifier des BD.
- **Permet d'exprimer :**
  - Structures :
    - Description des données ; objets du problème.
    - Ex: on veut représenter des cours (ue, nom, heuresCM, heuresTD)
  - Contraintes :
    - Règles que les données doivent satisfaire.
    - Ex: ue est unique ; heuresCM et heuresTD sont positives
  - Opérations :
    - Ensemble d'opérations pour manipuler les données.
    - Ex: comment insérer un cours ; comment lister les cours existants

Schémas et instances

- **Schéma:**
  - Indique les entités manipulées par le SGBD
  - Types de données existants
  - Exemple :
    - COMPTE (numero, solde, type)
    - CLIENT (nss, nom, prenom, adresse, email)

Très stable

- **Instance:**
  - Données courantes dans une BD
  - État d'une base de données

Très volatile

compte :	numero	solde	type
	12345	845.69	courant
	67890	5500.20	epargne
	...	...	...

Classification de modèles de données

Analogie

- **Classification selon le niveau d'abstraction:**
  - Conceptuels
    - Représentation à haut niveau des objets de la réalité et leurs interrelations
    - Indépendamment de l'implémentation BD.
    - Utilisé dans la phase d'analyse du SI

    Diagramme de classes

  Dans ce module

  - Logiques
    - Représentation dans un formalisme que peut être interprété et manipulé (opérations) par le SGBD.
    - Utilisé dans les phases de conception et implémentation du SI.

    Définition de classes dans un langage OO

  - Physiques
    - Implémentation des structures de données.
    - Aspects d'administration, d'optimisation…
    - Utilisé dans la phase d'implémentation du SI

    Code binaire

Méthodologies de conception BD

- **Deux méthodologies complémentaires pour la conception d'une BD relationnelle :**
  - Modélisation conceptuelle + traduction
    - Modéliser le problème à haut niveau
    - Traduire le schéma conceptuel au modèle relationnel
    - Si besoin, adapter le schéma relationnel

  BD & Web

  - Modélisation directe relationnelle
    - Partir d'une liste d'attributs et contraintes
    - Concevoir le schéma relationnel

  CBD

- **Le processus de conception/adaptation s'appelle normalisation**

Dans le monde des entreprises

- **Plusieurs scénarios de conception BD :**
  1. Conception d'une nouvelle BD :
     - Schéma conceptuel + traduction + normalisation si besoin
  2. Modification d'une BD existante (bien modélisée) :
     - Mise à jour du schéma conceptuel + mise à jour de la traduction + normalisation si besoin + migration de données
  3. Modification d'une BD existante (pas modélisée/documentée) :
     - Modification des relations + normalisation + migration des données
     - Ou réingénierie :
       - Utilisation de la BD (ou fichier) existante pour comprendre le problème et produire un schéma conceptuel
       - On procède comme dans le cas 1 mais avec migration de données

Plan

- **Concepts de base**
  - SI, BD et SGBD
  - Modèles de données
  - Schémas et instances
- **Rappel du modèle relationnel**
  - Structures de données
  - Opérations d'interrogation : SQL-QL
- **Contraintes dans le modèle relationnel**
  - Contraintes
  - Commandes de mise à jour : SQL-DML
  - Commandes de définition de données : SQL-DDL

Modèle relationnel

- **Créé par Codd en 1970**

- **C'est un modèle logique de données**
  - Utilisé comme modèle pour l'implémentation de SGBD
  - De nos jours, c'est le modèle logique dominant.

Vision informel du modèle

- **Les données sont organisés en tables : RELATIONS**
  - Les lignes correspondent à TUPLES.
  - Les colonnes correspondent à ATTRIBUTS de type atomique.
- **Les opérations sont orientées à la manipulation des relations** (ensembles de tuples)
- **Exemple :**
  - FILMS (title, director, year)

title	director	year
Star wars	Lucas	1977
Nikita	Besson	1990
Dune	Lynch	1984
Lady and the tramp	Geronimi	1955

- **C'est un modèle de données très simple et claire mais très puissant pour la plupart des applications BD**

Schémas de relation

- **Schéma de relation : $R(A_1,...,A_n)$**
  - Consiste d'un nom de relation, R, et d'une liste d'attributs, $A_1,...,A_n$, avec domaines $D_1,...,D_n$.
  - Exemple :
    - FILM (title, director, year)
    - SCHEDULE (theater, title, time)

- **Sort d'une relation : sort(R)**
  - Est une fonction qui associe à chaque nom de relation un ensemble fini d'attributs.
  - L'arité d'une relation R est **arity(R)** = |sort(R)|
  - Exemple :
    - sort (FILM) = {title, director, year}
    - sort (SCHEDULE) = {theater, title, time}
    - arity (FILM) = arity (SCHEDULE) = 3

Schémas de bases de données

- **Schéma de BD : B = { $R_1(U_1)$, . . . ,$R_n(U_n)$ }**
  - Est un ensemble fini non vide, B, de schémas de relations
  - Indique les schémas de relations de B.
  - Exemple :
    - BD-Cinéma = { FILM (title, director, year), SCHEDULE (theater, title, time) }

Tuples

- **Tuple** (ou n-uplet) : t(U)
  - Est un élément du produit cartésien des domaines des attributs : $t \in D_1 \times ... \times D_n$
  - Peut être interprété comme :
    - un vecteur :
      - t2 = <Nikita, Besson, 1990>
    - une fonction totale qui associe à chaque attribut une valeur dans son domaine : $t:U \rightarrow D_1 \cup ... \cup D_n$
      - t2 [title] = Nikita
      - t2 [director] = Besson
      - t2 [year] = 1990

title	director	year
Star wars	Lucas	1977
Nikita	Besson	1990
Dune	Lynch	1984
Lady and the tramp	Geronimi	1955

Instances de relation

- **Instance de relation** (ou relation) : **I(R)**
  - Est un instance d'un schéma de relation
  - Consiste d'un ensemble fini de tuples dont le sort est R
    - Pas ordonné, pas de répétés
  - Peut être interprété comme : $I(R) \subseteq (D_1 \times ... \times D_n)$
  - Exemple :
    - I(FILM) = {t1,t2,t3,t4}
      - t1 = <Star wars, Lucas, 1977>
      - t2 = <Nikita, Besson, 1990>
      - …

title	director	year
Star wars	Lucas	1977
Nikita	Besson	1990
Dune	Lynch	1984
Lady and the tramp	Geronimi	1955

Opérations

- **Le modèle relationnel propose des opérations pour :**
  - insérer, supprimer, modifier des données dans la BD
  - interroger la BD

- **Langages d'interrogation :**
  - Algèbre relationnelle
  - Calcul relationnel
  - SQL – QL (query language)

Structured Query Language - SQL

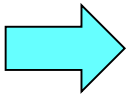
- **Un peu d'histoire :**
  - SEQUEL (IBM-1970) → SQL
  - QUEL (Ingres-1970)
  - QBE (IBM-1970)
- **SQL est le langage commercial le plus accepté**
  - Standard ANSI e ISO en 1986.
  - Différents dialectes :
    - SQL1 (1986-1987, révisée en 1989)
    - SQL2 (1992)
    - SQL3 ou SQL:1999
    - SQL:2003
    - SQL:2006
    - SQL:2008
    - SQL:2011

Structured Query Language - SQL

- **Data Definition Language (DDL)**
  - Décrit les types des entités et des relations entre les types d'entité en suivant un modèle de données
  - Exemple:
    - CREATE TABLE Fligths (number: int, date: char(6), seats : int, from: char(3), to: char(3));
- **Data Manipulation Language (DML)**
  - Permet l'interrogation (requêtes) et la mise à jour (insertion, suppression, modification) des données de la BD
    - Query Language (QL) est le sous-langage d'interrogation
  - Exemple:
    - INSERT INTO Fligths VALUES (456, 'Aug 21', 100, 'CDG', 'JFK');

Devoir

- **Chercher / construire un aide mémoire des commandes de SQL**
  - Inclure :
    - Syntaxe des clauses principales
      - Select, insert, delete, update, create table, drop table, alter table
    - Syntaxe des fonctions de
      - Comparaison et agrégation
      - Manipulation de nombres, dates, strings…
      - Conversion de types
    - Syntaxe des fonctions spécifiques d'Oracle
  - Ajouter des liens à descriptions de commandes, exemples et tutoriaux



DML - Interrogation (QL)

- **Caractéristiques de SQL-QL:**
  - Langage non procédural.
  - Manipule des multi-ensembles de tuples (tables):
    - N'élimine pas les tuples doublons .
  - Le même langage peut être utilisé :
    - De façon interactive (Ex.: console).
    - Imbriqué dans un langage de programmation.
- **SQL-QL est plus expressif que l'algèbre relationnelle**
  - On peut exprimer toutes les requêtes exprimables en algèbre
  - et plus.
- **Il privilégie les requêtes conjonctives**

Requêtes conjonctives

- **Syntaxe :**

  SELECT $A_1,...,A_n$

  FROM $R_1,...,R_m$

  WHERE cond

  où:

  - $R_1,...R_m$ sont des noms de tables
  - $A_1,...A_n$ sont des noms d'attributs de $R_1,...R_m$ (* = tous)
  - cond est une condition

Projection (π)

- **Syntaxe SQL :**
  - SELECT $X_1,\ldots X_k$ FROM R
- **Exemple :**

  SELECT titre FROM films;
- **Remarques :**
  - Projeter tous les attributs de la table : *

    SELECT * FROM films;
  - Eliminer les doublons : DISTINCT

    SELECT DISTINCT titre FROM films;
  - Ordonner les tuples : ORDER BY
    - On peut indiquer les sens (ASC o DESC)

    SELECT titre FROM films ORDER BY titre ASC;

FILMS

titre	réalisateur	année
Starwars	Lucas	1977
Nikita	Besson	1990
Dune	Lynch	1984
Antz	Darnell	1998
Antz	Johnson	1998

T1

titre
Starwars
Nikita
Dune
Antz
Antz

Sélection (σ)

- **Syntaxe SQL :**
  - SELECT * FROM R WHERE cond

- **Exemple :**
  - Quels sont les réalisateurs français ?

    SELECT * FROM realisateurs

    WHERE nationnalite='française'

- **Exemple :** sélection et projection
  - Quels sont les noms des réalisateurs français ?

    SELECT nom FROM realisateurs

    WHERE nationnalite='française'

REALISATEURS

nom	nationalité
Lucas	américaine
Lynch	américaine
Besson	française
Darnell	américaine
Johnson	américaine

T1

nom	nationalité
Besson	française

T2

nom
Besson

Sélection : remarques

- **Formules de sélection (utilisables dans la clause WHERE) :**
  - AND, OR, ...
  - Comparateurs : = , != , > , >= ,...
  - IS NULL, IS NOT NULL
- **Expressions (utilisables dans les clauses WHERE et SELECT) :**
  - Opérations arithmétiques : +, -, *, /, round(n), abs(n), mod(m, n), sign(n)
  - Opérations sur des strings : upper(s), lower(s), s || t
  - Changements de format, …
  - Exemple: Lister les budgets des films en francs

productions (titre, entreprise, annee, budget)

```
SELECT titre, budget * 6.56
FROM productions;
```

Produit cartésien (x)

- **Syntaxe SQL :**
  - SELECT * FROM R, S
- **Exemple :**
  - SELECT * FROM films, realisateurs

FILMS

titre	réalisateur	année
starwars	lucas	1977
nikita	besson	1990

REALISATEURS

nom	nationalité
lucas	américaine
lynch	américaine
besson	française

T

titre	réalisateur	année	nom	nationalité
starwars	lucas	1977	lucas	américaine
starwars	lucas	1977	lynch	américaine
starwars	lucas	1977	besson	française
nikita	besson	1990	lucas	américaine
nikita	besson	1990	lynch	américaine
nikita	besson	1990	besson	française

Jointure (⋈)

- **Syntaxe SQL :**
  - SELECT * FROM R, S WHERE cond
- **Exemple :**
  - Donner les films, leurs réalisateurs et leurs nationalités ?
    - Jointure par égalité des réalisateurs

SELECT * FROM films, realisateurs

WHERE realisateur=nom

FILMS

titre	réalisateur	année
starwars	lucas	1977
nikita	besson	1990

REALISATEURS

nom	nationalité
lucas	américaine
lynch	américaine
besson	française

T

titre	réalisateur	année	nom	nationalité
starwars	lucas	1977	lucas	américaine
nikita	besson	1990	besson	française

Jointure

- **Exemple : Jointure + sélection**
  - Quels sont les films réalisés par des américains ?
    1. Jointure par égalité des réalisateurs
    2. Sélection des tuples des réalisateurs américains

```
SELECT * FROM realisateurs, films
WHERE realisateur=nom AND nationalite='americaine'
```

REALISATEURS

nom	nationalité
Lucas	américaine
Lynch	américaine
Besson	française

FILMS

titre	réalisateur	année
Starwars	Lucas	1977
Nikita	Besson	1990
Dune	Lynch	1984
Antz	Darnell	1998
Antz	Johnson	1998

T

nom	nationalité	titre	réalisateur	année
lucas	américaine	starwars	lucas	1977
lynch	américaine	dune	lynch	1984

Jointures : remarques

- **Forme générale :**

```
SELECT *
FROM T1, T2
WHERE T1.A = T2.A;
```

ou :

```
SELECT *
FROM T1 JOIN T2 ON T1.A = T2.A;
```

T1

A	**B**
1	a
2	b
3	c

T2

A	C
1	xxx
3	yyy
5	zzz

```
+------+------+------+------+
| A    | B    | A    | C    |
+------+------+------+------+
| 1    | a    | 1    | xxx  |
| 3    | c    | 3    | yyy  |
+------+------+------+------+
```

- **Jointure naturelle :**

```
SELECT *
FROM T1 NATURAL JOIN T2;
```

```
+------+------+------+
| A    | B    | C    |
+------+------+------+
| 1    | a    | xxx  |
| 3    | c    | yyy  |
+------+------+------+
```

Jointures : remarques

T1

A	B
1	a
2	b
3	c

T2

A	C
1	xxx
3	yyy
5	zzz

- **Jointure gauche (left join) :**
  - Inclut des tuples de T1 qui n'ont pas de correspondant dans T2
  - en complétant par des valeurs nulles

```
SELECT *
FROM T1 LEFT JOIN T2 ON T1.A = T2.A;
```

A	B	A	C
1	a	1	xxx
2	b	NULL	NULL
3	c	3	yyy

- **Jointure droite (right join) :**
  - Analogue, en incluant toutes les tuples de T2

```
SELECT *
FROM T1 RIGHT JOIN T2 ON T1.A = T2.A;
```

A	B	A	C
1	a	1	xxx
3	c	3	yyy
NULL	NULL	5	zzz

- **Jointure complète (full join) :**
  - Union de jointure gauche et droite

```
SELECT *
FROM T1 FULL JOIN T2 ON T1.A = T2.A;
```

A	B	A	C
1	a	1	xxx
2	b	NULL	NULL
3	c	3	yyy
NULL	NULL	5	zzz

Jointure : Retour du TP SQL en 2017

- **Exemple :** Quels sont les films réalisés par des américains ?

REALISATEURS

nom	nationalité
Lucas	américaine
Lynch	américaine
Besson	française

FILMS

titre	réalisateur	année
Starwars	Lucas	1977
Nikita	Besson	1990
Dune	Lynch	1984
Antz	Darnell	1998
Antz	Johnson	1998

T

titre
starwars
dune

```
SELECT titre FROM realisateurs, films
    WHERE realisateur=nom AND nationalite='americaine';

SELECT titre FROM realisateurs JOIN films ON realisateur=nom
    WHERE nationalite='americaine';

SELECT titre FROM films WHERE realisateur IN (
        SELECT nom FROM realisateurs WHERE nationalite='americaine');
```

Renommage : remarques

T1

titre_film
Starwars
Nikita
Dune
Antz
Antz

- **Pour renommer un attribut :** AS
  - Exemple: Retrouver les films réalisés par Lucas.

```
SELECT titre AS titre_film
FROM films
WHERE realisateur='lucas';
```

- **Pour lever l'ambigüité sur les noms d'attribut** : préfixer avec le nom de la table
  - Exemple: Retrouver les films de Lucas et les salles qui les projettent.

```
SELECT films.titre, salle
FROM salles, films
WHERE films.titre = salles.titre
AND realisateur = 'lucas';
```

films (titre, realisateur, annee)
salles (salle, titre, capacite)

Renommage : remarques

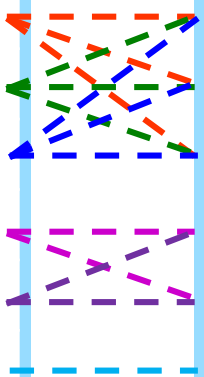
- **Pour lever l'ambigüité sur les noms de tables** : utiliser des *alias*
  - Exemple: Trouver les acteurs qui jouent dans le même film

```
SELECT M1.acteur, M2.acteur
FROM joue M1, joue M2
WHERE M1.titre = M2.titre
```

Trouver le problème de cette solution
(dans cet exemple précis)

JOUE M1

titre	acteur
starwars	hamil
starwars	ford
starwars	fisher
nikita	parillaud
nikita	duret
dune	madsen



JOUE M2

titre	acteur
starwars	hamil
starwars	ford
starwars	fisher
nikita	parillaud
nikita	duret
dune	madsen

T

M1.acteur	M2.acteur
hamil	hamil
hamil	ford
hamil	fisher
ford	hamil
…	
madsen	madsen

Renommage : remarques

- **Pour lever l'ambigüité sur les noms de tables** : utiliser des *alias*
  - Exemple: Trouver les acteurs qui jouent dans le même film

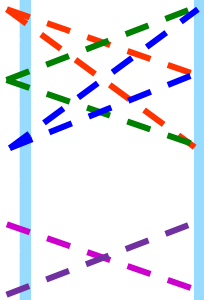
```
SELECT M1.acteur, M2.acteur
FROM joue M1, joue M2
WHERE M1.titre = M2.titre
AND M1.acteur <> M2.acteur
```

JOUE

titre	acteur
starwars	hamil
starwars	ford
starwars	fisher
nikita	parillaud
nikita	duret
dune	madsen

JOUE M2

titre	acteur
starwars	hamil
starwars	ford
starwars	fisher
nikita	parillaud
nikita	duret
dune	madsen



T

M1.acteur	M2.acteur
hamil	ford
hamil	fisher
ford	hamil
ford	fisher
fisher	hamil
fisher	ford
parrillaud	duret
duret	parrillaud

Renommage : remarques

- **Pour lever l'ambigüité sur les noms de tables** : utiliser des *alias*
  - Exemple: Trouver les acteurs qui jouent dans le même film

```
SELECT M1.acteur, M2.acteur
FROM joue M1, joue M2
WHERE M1.titre = M2.titre
AND M1.acteur > M2.acteur
```

JOUE

titre	acteur
starwars	hamil
starwars	ford
starwars	fisher
nikita	parillaud
nikita	duret
dune	madsen

JOUE M2

titre	acteur
starwars	hamil
starwars	ford
starwars	fisher
nikita	parillaud
nikita	duret
dune	madsen

T'

M1.acteur	M2.acteur
hamil	ford
hamil	fisher
ford	fisher
parrillaud	duret

Operateurs ensemblistes

- **Union :** UNION
  - Exemple : Trouver les acteurs et réalisateurs de films

    SELECT realisateur FROM film UNION SELECT acteur FROM joue;
  - Attention : l'union élimine des tuples répetés
- **Différence :** MINUS
  - Exemple : Trouver les réalisateurs qui ne sont pas acteurs

    SELECT realisateur FROM film MINUS SELECT acteur FROM joue;
- **Intersection :** INTERSECT
  - Exemple : Trouver les acteurs qui sont aussi réalisateurs

    SELECT realisateur FROM film INTERSECT SELECT acteur FROM joue;

films (titre, realisateur, annee)
joue (titre, acteur)

Agrégation

- **De nombreuses fonctions sont disponibles :** avg, count, sum, max, min...
- **Exemples :**

  SALLES (salle, titre, capacite);

  – Compter les salles

    SELECT count distinct (salle) FROM salles;

  – Trouver la salle à capacité maximale

    SELECT max (capacite) FROM salles;

Agrégation

- **Agrégation par groupes :** GROUP BY
  - Divise le résultat en groupes selon la valeur des attributs indiqués
    - Les expressions de la clause SELECT ne peuvent être que :
      - Les attributs de la clause GROUP BY.
      - Fonctions d'agrégation.
      - Expressions sur les précédents.
    - Le groupement se réalise après la sélection (clause WHERE)
      - Donc sur les tuples qui satisfont la condition
  - Exemple : Lister les entreprises et leur budget moyen

```
SELECT entreprise, avg(budget)
FROM productions
GROUP BY entreprise;
```

productions (titre, entreprise, annee, budget)

Agrégation

- **Conditions sur les agrégats :** HAVING
  - Exemple : Lister les entreprises et leur budget moyen, mais seulement si ce dernier est supérieur à 1.000.000 EUR

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

  - Exemple : Lister les entreprises qui ont réalisé plus de 3 films

```
SELECT entreprise
FROM productions
GROUP BY entreprise
HAVING count(*) > 3;
```

productions (titre, entreprise, annee, budget)

Requêtes imbriquées

- **Exemples :**
  - Lister les films qui ont le plus grand budget.

```
SELECT titre FROM productions
WHERE budget = (
   SELECT max(budget) FROM productions);
```

  - Lister les entreprises qui ont co-réalisé un film avec Canal+.

```
SELECT entreprise FROM productions
WHERE titre IN (
   SELECT titre FROM productions
   WHERE entreprise = 'Canal+');
```

productions (titre, entreprise, annee, budget)

Requêtes imbriquées

- **Attention :**
  - Beaucoup de requêtes imbriquées peuvent se résoudre avec une requête plus simple.
- **Exemple :**
  - Lister les entreprises qui ont réalisé un seul film.

```
SELECT entreprise FROM productions X
WHERE NOT EXISTS (
   SELECT Y.entreprise FROM productions Y
   WHERE X. entreprise = Y. entreprise
   AND X.titre != Y.titre);

SELECT entreprise FROM productions X
GROUP BY entreprise
HAVING count(*)=1;
```

productions (titre, entreprise, annee, budget)

Requêtes imbriquées

- **Les opérations ensemblistes peuvent s'implémenter :**
  - Avec les opérateurs ensemblistes INTERSECT, MINUS
  - Avec IN, NOT IN dans le WHERE pour des requêtes imbriquées
- **Intersection :**
  - Exemple : Trouver les films qui sont à l'affiche

```
SELECT titre FROM film
WHERE titre IN
   (SELECT titre FROM salles);
```

- **Différence :**
  - Exemple : Trouver les films qui ne sont pas à l'affiche

```
SELECT titre FROM film
WHERE titre NOT IN
   (SELECT titre FROM salles);
```

films (titre, realisateur, annee)
salles (salle, titre, capacite)

Résumé

- **SQL-QL fait tout ça... et bien plus encore !**
  - Il évolue encore... à suivre

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

Plan

- **Concepts de base**
  - SI, BD et SGBD
  - Modèles de données
  - Schémas et instances
- **Rappel du modèle relationnel**
  - Structures de données
  - Opérations d'interrogation : SQL-QL
- **Contraintes dans le modèle relationnel**
  - Contraintes
  - Commandes de mise à jour : SQL-DML
  - Commandes de définition de données : SQL-DDL

Contraintes

- **Clés**
  - Sous ensemble d'attributs à valeur unique
  - Etant donné R(A1,...,An), X $\subseteq$ {A1,...,An}

    X est clé de R s'il ne peut pas exister aucune instance I(R) contenant deux tuples avec les mêmes valeurs de X (t1[X] = t2[X])

  - Une relation peut avoir plusieurs clés
    - L'une est déclaré comme clé primaire et bénéficie d'indexes optimisés
    - Les autres sont des clés secondaires

  - Exemple :
    - PERSONNEL (code, nom, prenom, email, telephone, etage, bureau, numPoste)
    - constraint pk_personnel primary key (code)
    - constraint pk_personnel_email unique (email)
    - constraint uk_personnel_poste unique (bureau, numPoste)

Contraintes

- **Clés étrangères (intégrité référentiel)**
  - Sous ensemble d'attributs qui doivent être présents dans une autre table
  - Etant donné R(A1,...,An), S(B1,...,Bm), X $\subseteq$ {A1,...,An},

    X est clé étrangère de R référençant S si
    - Les attributs de X coïncident en domaine avec une clé Y de S
    - Pour tout tuple de tout instance I(R), les valeurs de X correspondent à des valeurs de Y d'une tuple de I(S)
  - Exemple :
    - constraint fk_personne foreign key (ville_naiss) references VILLE (villeID)

PERSONNE				
PersID	nom	prénom	date_naiss	ville_naiss
1	Dupont	bob	01-01-1950	1
2	yyyy	merise	29-04-1999	2
3	zzzz	codd	26-12-2000	1

VILLE				
VilleID	nom	population	superficie	region
1	Paris	123456	123456	12
2	Lyon	12345	12345	22
3	Grenoble	1234	1234	22

Contraintes

- **Contraintes de domaine**
  - Restrictions du type des attributs
  - Exemple :
    - PERSONNEL (code, nom, prenom, email, telephone, etage, bureau, numPoste)
    - constraint ch_personnel_etage check (etage >= 0 AND etage <= 4)
    - constraint ch_personnel_tel check (telephone LIKE '02545%')
- **Valeurs nulles / non nulles**
  - Spécifiés dans la déclaration d'attributs
  - Exemple : nom varchar2(25) not null
- **D'autres contraintes existent mais ne sont pas contrôlés directement par le SGBD**
  - Exemple :
    - Si deux personnes travaillent dans le même bureau, elles sont au même étage
    - bureau → etage

Contraintes

- **Une BD est valide si :**
  - Toutes les relations r satisfont les contraintes de la BD.
- **Propriétés importantes :**
  - Les contraintes surgissent de :
    - L'observation de la réalité.
    - PAS de l'observation des relations.
  - Les contraintes sont définis :
    - A niveau de schéma de relation
    - PAS à niveau d'instance.
  - Les contraintes sont violés :
    - Par les relations.
    - PAS par les schémas de relations.

DML - Insertion

- **Syntaxe :**

```
INSERT INTO Nom_table (colonne1,colonne2,colonne3,...)
VALUES (Valeur1,Valeur2,Valeur3,...)
```

```
INSERT INTO Nom_table (colonne1,colonne2,...)
SELECT colonne1,colonne2,...
FROM Nom_de_la_table2
WHERE qualification
```

films (titre, realisateur, annee)
new_films (titre, annee)

- **Exemples :**
  - INSERT INTO new_films (titre, annee) VALUES ('Cindirella',2015)
    - Ajoute le tuple dans l'instance de la table new_films.
  - INSERT INTO films (annee, titre) SELECT annee, titre FROM new_films
    - Ajoute les tuples retournés par la sous-requête dans l'instance de la table films.
- **Les tuples ajoutés doivent satisfaire les contraintes.**

DML - Mise à jour

- **Syntaxe :**

```
UPDATE Nom_table
SET Colonne = Valeur_Ou_Expression
WHERE qualification
```

- **Exemple :**

salles (salle, titre, capacite)

  - UPDATE salles SET capacite = 200 WHERE salle='A101'
    - Modifie les tuples de l'instance de salles qui satisfont la condition en changeant la valeur de capacite

- **Les tuples modifiés doivent satisfaire les contraintes.**

DML - Suppression

- **Syntaxe :**

```
DELETE FROM Nom_de_la_table
WHERE qualification
```

new_films (titre, annee)

- **Exemple :**
  - delete from new_films where annee < 2014
    - Supprime les tuples de l'instance de new_films qui satisfont la condition.

- **La suppression doit satisfaire les contraintes (clés étrangères)**

DDL - Création de tables

- **Syntaxe :**

```
CREATE TABLE [schéma.]nomtable
(colonne1 type1 [contrainte1] [contrainte2]…
[,colonne2 type2 [contrainte3] [contrainte4]…]
[,contrainteglobal5] [,contrainteglobal6] …)
```

- **Exemple simple** (sans contraintes) **:**

```
CREATE TABLE personne (
code NUMBER(4),
nom VARCHAR2(32),
prenom VARCHAR2(32),
sexe CHAR(1),
date_naissance DATE,
departement NUMBER(2))
```

- **Génère une instance vide**

DDL - Création de tables

- **Contrainte NULL / NOT NULL :**
  - On précise NOT NULL pour indiquer un champ qui doit obligatoirement être renseigné.

- **Exemple :**

```
CREATE TABLE personne (
code NUMBER(4) NOT NULL,
nom VARCHAR2(32) NOT NULL,
prenom VARCHAR2(32) NOT NULL,
sexe CHAR(1),
date_naissance DATE,
departement NUMBER(2) NULL)
```

DDL - Création de tables

- **Valeur par défaut :**
  - On précise la valeur par défaut d'un attribut.

- **Exemple :**

```
CREATE TABLE personne (
code NUMBER(4) NOT NULL,
nom VARCHAR2(32) NOT NULL,
prenom VARCHAR2(32) NOT NULL,
sexe CHAR(1) DEFAULT 'M',
date_naissance DATE DEFAULT CURRENT_DATE,
departement NUMBER(2) NULL)
```

Fonction dépendante du SGBD

DDL - Création de tables

- **Contraintes de domaine :**
  - On précise des contraintes sur la valeur d'un attribut ou entre les valeurs de plusieurs attributs.
  - On peut spécifier des contraintes de validation très complexes.
- **Syntaxe :** [CONSTRAINT nom_contrainte] CHECK (prédicat)
- **Exemple :**

```
CREATE TABLE personne (
code NUMBER(4) NOT NULL CHECK (code>0),
nom VARCHAR2(32) NOT NULL CHECK (LENGTH(nom)>2),
prenom VARCHAR2(32) NOT NULL,
sexe CHAR(1) DEFAULT 'M' CHECK (sexe IN ('M','F')),
date_naissance DATE DEFAULT CURRENT_DATE,
departement NUMBER(2) NULL,
CHECK (nom <> prenom))
```

DDL - Création de tables

- **Clé primaire :**
  - On indique les attributs qui composent la clé.
- **Syntaxe :** [CONSTRAINT nom_contr] PRIMARY KEY(liste_attributs)
- **Exemple :**

```
CREATE TABLE personne2 (
code NUMBER(4) NOT NULL PRIMARY KEY,
nom VARCHAR2(32) NOT NULL,
prenom VARCHAR2(32) NOT NULL)
```

```
CREATE TABLE personne3 (
nom VARCHAR2(32) NOT NULL,
prenom VARCHAR2(32) NOT NULL),
PRIMARY KEY (nom, prenom))
```

DDL - Création de tables

- **Clés secondaires :**
  - On indique les attributs qui composent les clés.
- **Syntaxe :** [CONSTRAINT nom_contr] UNIQUE (liste_attributs)
- **Exemple :**

```
CREATE TABLE personne (
code NUMBER(4) NOT NULL PRIMARY KEY,
nss NUMBER(13) UNIQUE,
nom VARCHAR2(32) NOT NULL,
prenom VARCHAR2(32) NOT NULL,
sexe CHAR(1) DEFAULT 'M',
date_naissance DATE DEFAULT CURRENT_DATE,
departement NUMBER(2) NULL,
UNIQUE (nom, prenom))
```

DDL - Création de tables

- **Clés étrangères :**
  - On indique les attributs qui référencent une autre table.
- **Syntaxe :** [CONSTRAINT nom_contr] FOREIGN KEY (attributs)
  REFERENCES table (attributs_clé_primaire)
- **Exemple :**

```
CREATE TABLE FACTURE (
id_fact NUMBER(10) PRIMARY KEY,
date_fact DATE,
client INTEGER,
FOREIGN KEY (client) REFERENCES personne(code)
)
```

DDL - Création de tables

- **Nommer les contraintes permet de comprendre plus facilement les messages d'erreur.**
- **Exemple :**

```
CREATE TABLE personne (
code NUMBER(4) NOT NULL,
nss NUMBER(13),
nom VARCHAR2(32) NOT NULL,
prenom VARCHAR2(32) NOT NULL,
sexe CHAR(1) DEFAULT 'M',
date_naissance DATE DEFAULT CURRENT_DATE,
departement NUMBER(2) NULL,
CONSTRAINT pk_personne PRIMARY KEY (code),
CONSTRAINT uk_personne_nss UNIQUE (nss),
CONSTRAINT ch_personne_sexe CHECK (sexe IN ('M','F')),
CONSTRAINT ch_personne_n_p CHECK (nom <> prenom))
```

DDL - Modification de tables

- **Modifie le schéma d'une relation**
  - changement de noms d'attributs, changement de types, ajout/suppression d'attributs, ajout/suppression de contraintes…
- **Forme :**
  - ALTER TABLE nom_table …
- **Exemples :**

```
ALTER TABLE personne RENAME COLUMN date_naissance TO naissance;

ALTER TABLE personne DROP COLUMN sexe;
ALTER TABLE personne ADD (sexe char(1) default 'M');

ALTER TABLE personne DROP CONSTRAINT uk_personne_nss;
ALTER TABLE personne ADD CONSTRAINT ch_personne_nss CHECK (nss>0
```

- **Exercice : Réviser le reste des options**

DDL - Suppression de tables

- **Supprime le schéma de relation R**
- **Forme :**
  - DROP TABLE nom_table [options]

Langages et interfaces spécialisés

- **Host Languages**
  - Programmes d'application: écrits dans un langage de programmation (ex. C, COBOL, Java).
  - Les opérations sur la base de données (DML) sont imbriqués dans le code.
- **Interfaces graphiques de requête.**
  - Permettent de visualiser les structures de la BD de façon graphique.
  - Les résultats se visualisent comme des graphiques (camembert, barres, etc.).
- **Interfaces d'administration.**
  - Environnements spécialisés pour l'administration BD.

Résumé

- **Le modèle relationnel nous fournit :**
  - Un modèle de données logique pour structurer une BD
  - Un langage de définition de données (DDL)
    - Permet la définition d'objets (ex. relations) et de contraintes
  - Un langage de manipulation de données (DML)
    - Permet des opérations d'interrogation et de mise à jour

- **SQL est un standard... Comme tout standard**
  - il donne lieu à différentes implémentations
    - Ex.: le SQL de MySQL n'a pas d'opérateur MINUS ou INTERSECT
  - il évolue encore... à suivre