Base de données
-
SQL

7 avril 2010

1. Base de données
2. Modèle relationnel
3. Le langage SQL
   - LDD
   - LMD
   - SELECT
4. Requêtes plus évoluées
5. Opérateurs et fonctions
   - Regroupement
   - Jointures externes

Base de données (BDD)

Définition

- ensemble **structuré** et **organisé** de données mis à la disposition d'ensembles d'utilisateurs
- gérée par un administateur
- **exhaustive** : contient toutes les données nécessaires à la problématique
- **unicité** : centralise les informations d'une manière **non redondante**
- **cohérence** : vérifie les mises à jour, modifications...

Base de données (BDD)

Bdd de prêt de dvd

- entités à modeliser
- données à sauvegarder :
  - dvd :
  - ami :
  - pret :
- pb unicité :
- pb redondance :

Base de données (BDD)

Bdd de prêt de dvd

- entités à modeliser : dvd, ami, pret
- données à sauvegarder :
  - dvd :
  - ami :
  - pret :
- pb unicité :
- pb redondance :

Base de données (BDD)

Bdd de prêt de dvd

- entités à modeliser : dvd, ami, pret
- données à sauvegarder :
  - dvd :
  - ami :
  - pret :
- pb unicité :
- pb redondance :

Base de données (BDD)

Bdd de prêt de dvd

- entités à modeliser : dvd, ami, pret
- données à sauvegarder :
  - dvd :
  - ami :
  - pret :
- pb unicité :
- pb redondance :

Base de données (BDD)

Bdd de prêt de dvd

- entités à modeliser : dvd, ami, pret
- données à sauvegarder :
  - dvd : titre, réalisateur, durée, genre, résumé
  - ami :
  - pret :
- pb unicité :
- pb redondance :

Base de données (BDD)

Bdd de prêt de dvd

- entités à modeliser : dvd, ami, pret
- données à sauvegarder :
  - dvd : titre, réalisateur, durée, genre, résumé
  - ami :
  - pret :
- pb unicité :
- pb redondance :

Base de données (BDD)

Bdd de prêt de dvd

- entités à modeliser : dvd, ami, pret
- données à sauvegarder :
  - dvd : titre, réalisateur, durée, genre, résumé
  - ami : nom, prénom, tel
  - pret :
- pb unicité :
- pb redondance :

Base de données (BDD)

Bdd de prêt de dvd

- entités à modeliser : dvd, ami, pret
- données à sauvegarder :
  - dvd : titre, réalisateur, durée, genre, résumé
  - ami : nom, prénom, tel
  - pret :
- pb unicité :
- pb redondance :

Base de données (BDD)

Bdd de prêt de dvd

- entités à modeliser : dvd, ami, pret
- données à sauvegarder :
  - dvd : titre, réalisateur, durée, genre, résumé
  - ami : nom, prénom, tel
  - pret : dvd, ami, date de pret
- pb unicité :
- pb redondance :

Base de données (BDD)

Bdd de prêt de dvd

- entités à modeliser : dvd, ami, pret
- données à sauvegarder :
  - dvd : titre, réalisateur, durée, genre, résumé
  - ami : nom, prénom, tel
  - pret : dvd, ami, date de pret
- pb unicité :
- pb redondance :

Base de données (BDD)

Bdd de prêt de dvd

- entités à modeliser : dvd, ami, pret
- données à sauvegarder :
  - dvd : titre, réalisateur, durée, genre, résumé
  - ami : nom, prénom, tel
  - pret : dvd, ami, date de pret
- pb unicité : identifiants dvd, ami
- pb redondance :

Base de données (BDD)

Bdd de prêt de dvd

- entités à modeliser : dvd, ami, pret
- données à sauvegarder :
  - dvd : titre, réalisateur, durée, genre, résumé
  - ami : nom, prénom, tel
  - pret : dvd, ami, date de pret
- pb unicité : identifiants dvd, ami
- pb redondance :

Base de données (BDD)

Bdd de prêt de dvd

- entités à modeliser : dvd, ami, pret
- données à sauvegarder :
  - dvd : titre, réalisateur, durée, genre, résumé
  - ami : nom, prénom, tel
  - pret : dvd, ami, date de pret
- pb unicité : identifiants dvd, ami
- pb redondance : identifiants numerique, factorisation des genres : nouvelle entite

Base de données (BDD)

Définition

Le **système de gestion de base de données**(SGBD) est un ensemble d'applications logicielles permettant de gérer la base de données

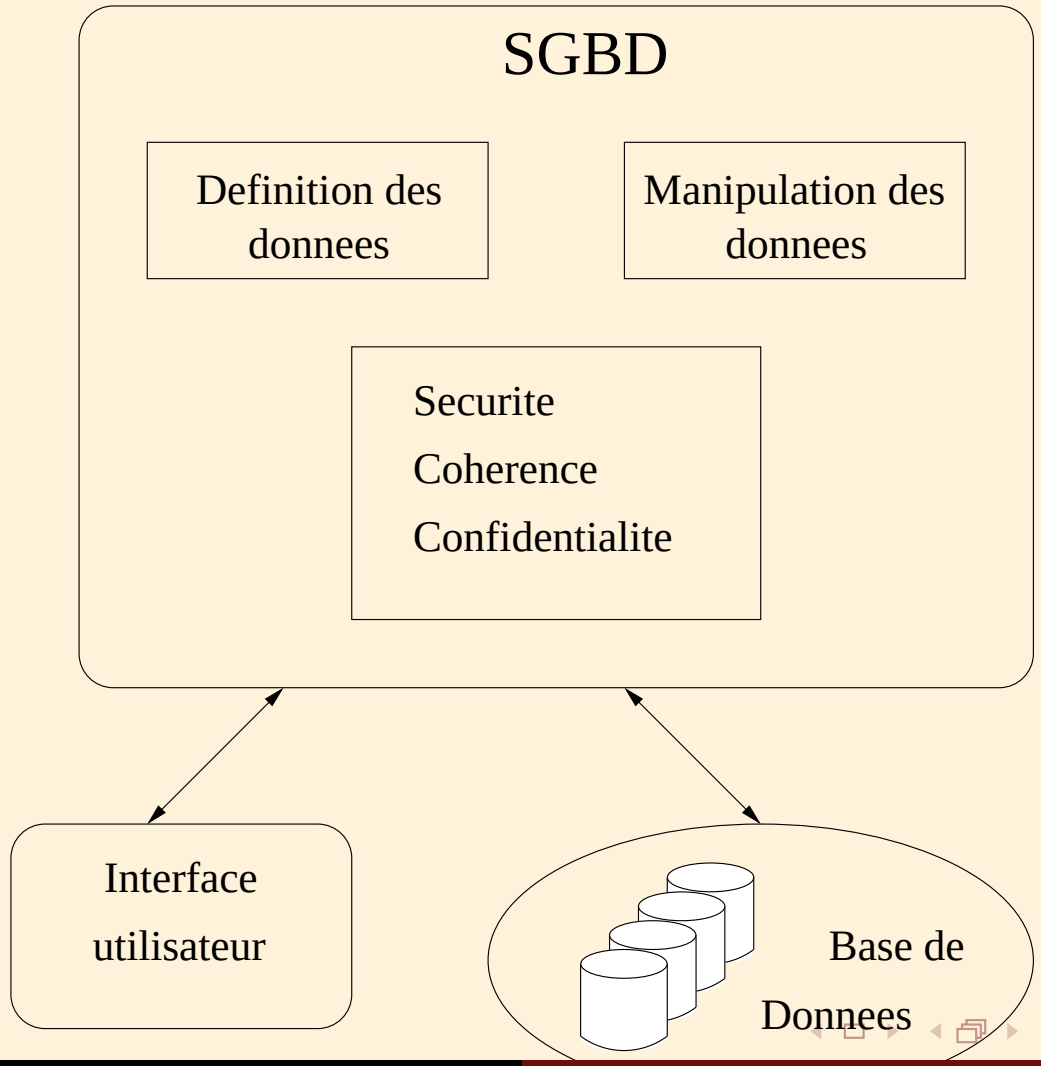
- gère les accès aux données, les relations entre ces données, leur manipulation
- différents SGBD : Orable, IBM DB2, Microsoft Access et SQL serveur, MySQL...

Base de données (BDD)

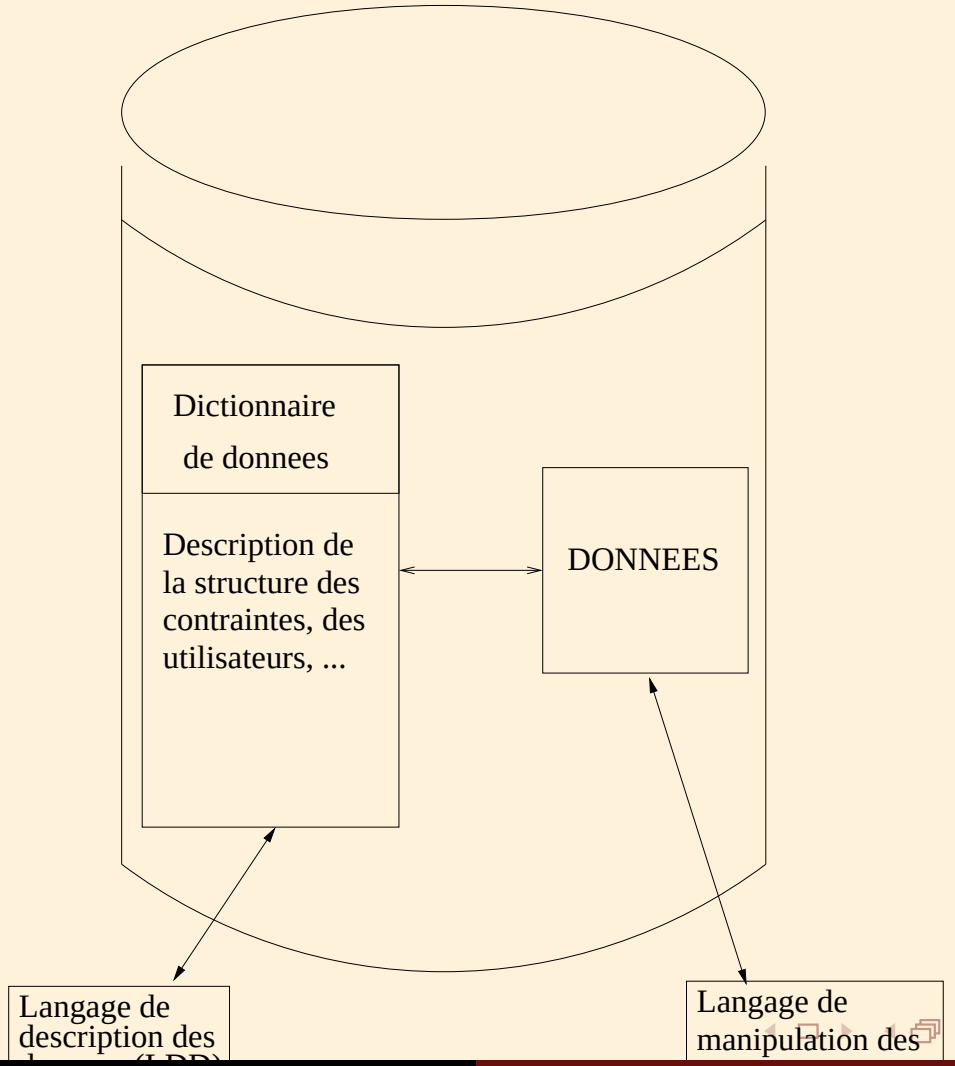
Principes

- indépendance données / traitements
- partage des données (confidentialité, concurrence)
- intégrité des données (contraintes d'intégrité)
- cohérence des données
- **utilisation pas des non informaticiens**

Base de données (BDD)



Base de données (BDD)



Base de données (BDD)

Conception d'une BDD

Ensemble de procédures d'analyse d'une problématique afin de construire un modèle persistant de données

- définition des données, structures, méthodes et contraintes
- établissement d'un modèle conceptuel (MERISE, UML)
- déduction d'un modèle physique de données qui sera implémenté

Modèle relationnel

Définition

- Modèle logique basé sur la théorie mathématique de l'algèbre relationnelle
- Somme, différence, produit, intersection d'ensembles de données

Modèle relationnel

Relations

- Les données de la bases sont organisées en structures, stockées dans des tables
- les tables sont modélisées par des **relations**

Modèle relationnel

Schéma de relation

$$\mathbf{R}(a1,a2,\ldots,an)$$

- R : nom de la relation
- a1,...,an : *attributs* de la relation
- (a1,...,an) : *n-uplet*
- n : *degré* de la relation (relation *n-aires*)

Modèle relationnel

Exemple

AMI(nom,prenom,tel)
DVD(titre,realisateur,resume,duree,genre)

Modèle relationnel

Contraintes d'intégrité

permettent de gérer la cohérence et l'intégrité des données. Des contrôles sont effectués lors de la manipulation des données (ajout, mise à jour, suppression)

- valeur nulle, par défaut, ensemble
- clé primaire
- clé étrangère

Contraintes d'intégrité

valeur nulle, par défaut, ensemble

- *nulle* : fixe le droit, ou non de laisser une valeur nulle pour un attribut de la relation
- *défaut* : fixe une valeur par défaut, modifiable, pour un attribut de la relation
- *ensemble* : fixe un ensemble de valeurs possibles pour un attribut de la relation (liste, intervalle)

Contraintes d'intégrité

clé primaire

- définit l'unicité d'une structure dans la relation
- formalisée par un attribut *id_nom_de_la_structure*

id_musicien	**nom**	**prenom**	**age_deces**
12	Mozart	W. A.	
22	Beethoven	Ludwig V.	
32	Ravel	Maurice	
12	Vilard	Hervé	

- impossible d'ajouter Hervé Vilard...

Contraintes d'intégrité

clé étrangère

- permet d'éviter la redondance de données
- fait référence à une clé primaire d'une autre relation afin d'afficher les données de 2 relations
- sa valeur est inclue dans l'ensemble des valeurs de la clé primaire à laquelle elle se réfère
- formalisée par *#id_nom_de_la_structure_de_référence*

Contraintes d'intégrité

id	titre	duree	annee	nom_real	prenom
1	Le chat	87	1971	Granier−Deferre	Pierre
2	Un singe en hiver	105	1962	Verneuil	Henri
3	Adieu Poulet	91	1975	Granier−Deferre	Pierre
4	Z	125	1969	Costa−Gavras	
3	L'aveu	140	1970	Costa−Gavras	

Contraintes d'intégrité

id	titre	duree	annee	id_real
1	Le chat	87	1971	1
2	Un singe en hiver	105	1962	3
3	Adieu Poulet	91	1975	1
4	Z	125	1969	2
3	L'aveu	140	1970	2

Film

id_real	nom_real	prenom
1	Granier−Deferre	Pierre
2	Costa−Gavras	
3	Verneuil	Henri
4	Miller	Claude

Contraintes d'intégrité

clé étrangère

id_musicien	**nom**	**prenom**	**age_deces**
12	Mozart	W. A.	
22	Beethoven	Ludwig V.	
32	Ravel	Maurice	

id_morceau	**nom**	**#id_musicien**
112	Requiem	12
122	9e symphonie	22
122	La truite	
132	La flute enchantée	12
142	bo le lavabo	**02**

incohérence de la donnée : ajout impossible

Contraintes d'intégrité

Formalisation dans un schéma relationnel

- Genre
  Genre(id_genre,libelle)
- Dvd
  Dvd(id_dvd,titre,real,resume,duree,#id_genre)
- Ami
  Ami(id_ami,nom,prenom,tel)
- Pret
  Pret(#id_dvd,#id_ami,date)

SQL

Définition

- Standard Query Language
- Langage utilisé pour communiquer avec la base de données par système de requêtes
- intuitif, simple et standardisé
- incluable dans d'autres langages (php, java c++...)
- deux sous-langages SQL :
  - Langage de définition des données (LDD)
  - Langage de manipulation des données (LMD)

Langage de définition de données (LDD)

Définition

Le langage de définition de données permet la création, modification, destruction **des tables** contenant les données

Langage de définition de données (LDD)

CREATE/DROP

- créer une table :
  CREATE TABLE nom_table(
  attr1 TYPE [PRIMARY KEY] [NOT NULL][REFERENCES nom_table],
  attr1 TYPE [PRIMARY KEY] [NOT NULL][REFERENCES nom_table],
  ...
  attrn TYPE [PRIMARY KEY] [NOT NULL][REFERENCES nom_table]
) ;
- supprimer une table
  DROP TABLE nom_table ;
- seulement le contenu
  TRUNCATE TABLE nom_table ;
- renommer la table
  RENAME TABLE ancien_nom TO nv_nom ;

Langage de définition de données (LDD)

Exemple

```
CREATE TABLE personne(
    id_personne NUMBER CONSTRAINT pk_per PRIMARY KEY,
    nom VARCHAR2(20),
    salaire NUMBER(12,2) CONSTRAINT const_sal NOT NULL,
    codepost NUMBER(5) CHECK(codepost BETWEEN 00001 AND 99999),
    date DATE DEFAULT current_date
);
```

Langage de définition de données (LDD)

ALTER

modifie la structure de la table

- ajouter une colonne
  ALTER TABLE nom_table ADD nom_colonne Type_donnees
- supprimer une colonne
  ALTER TABLE nom_table DROP nom_colonne
- modifie le type d'une colonne
  ALTER TABLE nom_table MODIFY nom_colonne Type_donnees

Langage de définition de données (LDD)

ALTER

ajoute/supprime des contraintes

- ajouter une clé primaire
  ALTER TABLE nom_table ADD CONSTRAINT nom_contrainte
  PRIMARY KEY(nom_colonne) ;
- ajouter une clé étrangère
  ALTER TABLE nom_table ADD CONSTRAINT nom_contrainte
  nom_colonne REFERENCES nom_table(nom_colonne)
  ON [*DELETE*|*UPDATE*][*SET NULL*|*CASCADE*] ;
- supprimer une contrainte
  ALTER TABLE nom_table DROP CONSTRAINT nom_contrainte

Langage de définition de données (LDD)

Contraintes

liste des contraintes

- PRIMARY KEY
- REFERENCES
- UNIQUE
- CHECK
- NOT NULL

Langage de manipulation de données (LMD)

Définition

Le langage de manipulation de données permet la création, modification, suppression et interrogation **des données**

Langage de manipulation de données (LMD)

Insertion

- INSERT INTO nom_table
  VALUES (val1,val2,...,valn) ; n, degré de la relation
- INSERT INTO nom_table
  VALUES (val1,null,...,valn) ; n, degré de la relation
- INSERT INTO nom_table(atti,...,attj,...,attk)
  VALUES (vali,...,valj,...,valk) ;

Langage de manipulation de données (LMD)

Suppression/Modification

modifie les données **qui correspondent aux critères**

- supression
  DELETE FROM nom_table
  WHERE critère ;
- modification
  UPDATE nom_table
  SET atti=vali, attj=valj ...
  WHERE critère ;

- DELETE FROM dvd WHERE genre='COMEDIE' ;
  supprime tous les films du genre COMEDIE
- UPDATE ami SET prenom=NULL WHERE nom='Dupont' ;
  met à nul le prénom de tous les amis dont le nom est Dupont

Principes algébriques d'interrogation

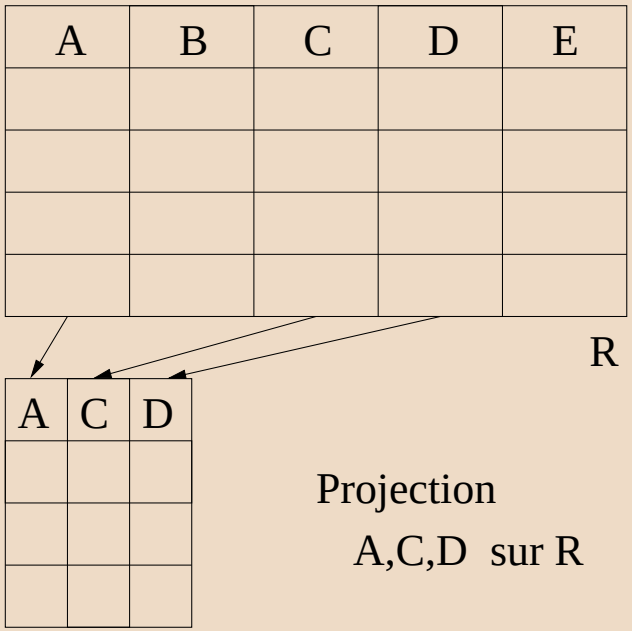
7 opérations algébriques :

- Projection
- Restriction
- Difference
- Union
- Produit
- Jointure
- Division

Principes algébriques d'interrogation

Projection

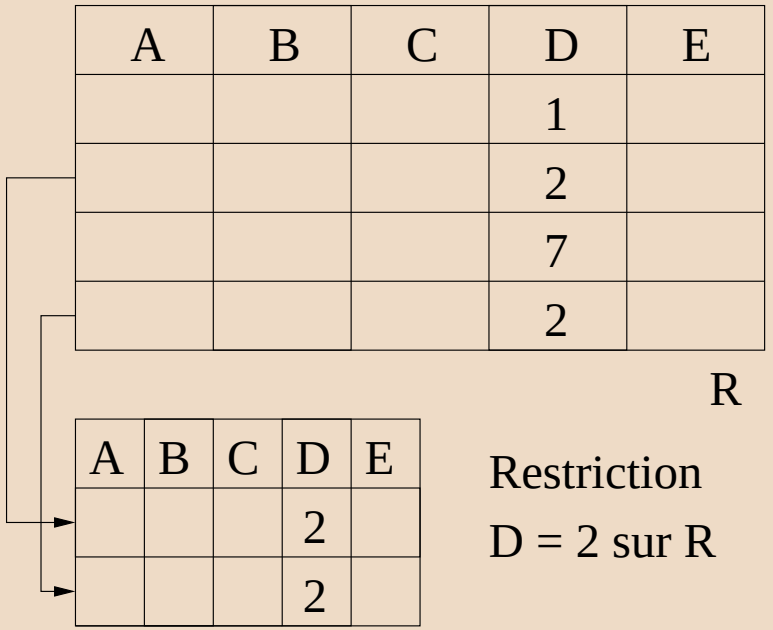
permet de sélectionner n-upplets d'attributs d'une relation.



Principes algébriques d'interrogation

Restriction

permet de sélectionner un n-upplets d'une relation.
Utilisation des des conditions simples $=, \neq, <, >, \ldots$ et des opérateurs logiques et, ou, non.



Principes algébriques d'interrogation

Difference

permet de soustraire deux relations.
R contient les éléments de R1 qui ne sont pas dans R2.
R1 et R2 doivent respecter le même schéma.

A	B	C
a1	a2	a3
b1	b2	b3

R1

A	B	C
a4	a5	a6
b1	b2	b3
a1	a2	a7

R2

A	B	C
a1	a2	a3

Difference R = R1 − R2

a4	a5	a6
a1	a2	a7

Difference R = R2 − R1

Principes algébriques d'interrogation

Union

permet d'ajouter deux relations.
R contient les éléments de R1 et ceux de R2 qui ne sont pas dans R1. R1 et R2 doivent respecter le même schéma.

A	B	C
a1	a2	a3
b1	b2	b3

R1

A	B	C
a4	a5	a6
b1	b2	b3
a1	a2	a7

R2

A	B	C
a1	a2	a3
b1	b2	b3
a4	a5	a6
a1	a2	a7

Union R = R1 U R2

Principes algébriques d'interrogation

Produit

réalise le produit cartésien de deux relations.
R contient les éléments de R1 associés à chacun des éléments de R2. Si R1 est un n-upplet et R2 un k-upplet, R est un (*kxn*)-upplet.

A	B	C	D	E
r11		...		
r12		...		

R1

F	G	H
r21	...	
r22	...	

R2

A	B	C	D	E	F	G	H
r11		...			r21	...	
r11		...			r22	...	
r12		...			r21	...	
r12		...			r22	...	

Principes algébriques d'interrogation

Jointure

permet de joindre deux relations selon un ou plusieurs prédicats.
R contient les éléments de R1 combinés avec ceux de R2 selon le ou les prédicats de jointure.
On appelle *jointure naturelle* un prédicat de jointure portant sur l'égalité des attributs communs aux deux relations

A	B
a1	a2
b1	b2
a3	b5

R1

D	E	F
a4	a5	a6
b1	b3	b6
a1	a4	a7

R2

A	B	D	E	F
a1	a2	a1	a4	a7
b1	b2	b1	b3	b6

Jointure

R = R1 joint R2
sur R1.A = R2.D

Principes algébriques d'interrogation

Division

R contient les éléments de R1 combinés avec **tous ceux** de R2.

A	B
a1	a2
b1	b2
a3	b5
b1	a2
a1	b2
a1	b5

R1

A
a1
b1

R2

B
a2
b2

R

Division R = R1/R2

Interrogation SQL

Requête simple

Les requêtes s'écrivent sous la forme :
SELECT *Liste des attributs*
FROM *Liste des tables nécessaires*
WHERE *Liste des prédicats*;

- SELECT : attributs de projection
- WHERE : conditions de jointure et de restriction

Interrogation SQL

Musicien(<u>idmusicien</u>)

Restriction

SELECT * FROM *nom_table* WHERE *prédicat de restriction* ;

Exemple

SELECT * FROM Compositeur WHERE age_deces $>$ 35 ;

Interrogation SQL

Projection

SELECT *liste des attributs de projection* FROM *table* ;

Exemple

SELECT DISTINCT nom,prenom FROM Compositeur ;

Interrogation SQL

Difference

Requete1 **MINUS** Requete2

Exemple

```
SELECT nom compositeur FROM musicien
MINUS
SELECT mu.nom compositeur
FROM musicien mu, morceau mo
WHERE mu.id_musicien=mo.id_musicien ;
```

Interrogation SQL

Difference

SELECT *attributs*
FROM *table*
WHERE *attribut*
NOT IN (SELECT *attribut*
FROM *table*
WHERE *Condition de restriction*) ;

Interrogation SQL

Difference

SELECT *attributs*
FROM *table* T1
WHERE **NOT EXISTS** (SELECT *attribut*
FROM *table* T2
WHERE T1.attribut=T2.attribut) ;

Interrogation SQL

Union

Requete1 **UNION** Requete2

Exemple

```
SELECT nom compositeur FROM musicien
UNION
SELECT nom compositeur FROM morceau ;
```

Interrogation SQL

Produit

SELECT *liste attributs*
FROM *table1*, *table2*;

Interrogation SQL

Jointure

SELECT *liste attributs*
FROM *table1, table2*
WHERE *prédicat de jointure* ;

Exemple

```
SELECT mu.nom compositeur, mo.nom titre
FROM musicien mu, morceau mo
WHERE mu.id_musicien=mo.id_musicien ;
```

Opérateurs

Opérateurs particuliers

permettent d'affiner les requêtes

- AND, OR
- LIKE
- BETWEEN
- ORDER BY . . . ASC/DESC
- IN, NOT IN, ALL, ANY
- EXISTS, NOT EXISTS
- UPPER, LOWER

Opérateurs

LIKE

permet de tester le contenu d'une chaîne de caractères.
2 jokers :

- _ : pour exactement un caractère
- % : pour 0 ou n caractères

Opérateurs

IN, NOT IN, ALL, ANY

permet de tester un ensemble de valeurs

- IN : valeur testée contenue dans l'ensemble
- NOT IN : valeur testée non contenue dans l'ensemble
- ALL : valeur testée doit valider toutes les valeurs de l'ensemble
- ANY : valeur testée doit valider au moins une valeur de l'ensemble

Opérateurs

EXISTS, NOT EXISTS

permet de quantifier les ensembles, utilisés notament pour la division, les requêtes du type tous/aucun

Opérateurs

Exemples

Nom et prénom des compositeurs dont le nom commence par M :

```
SELECT nom, prenom FROM Musicien mu
WHERE nom LIKE 'M%' ORDER BY nom DESC ;
```

Morceaux de Mozart dont la durée est supérieure à tous les morceaux de Ravel :

```
SELECT titre FROM Musicien mu, Morceau mo
WHERE mo.id_musicien=mo.id_musicien AND UPPER(nom)=UPPER('Mozart')
AND duree > ALL (SELECT duree FROM Musicien mu, Morceau mo WHERE
mo.id_musicien=mo.id_musicien AND LOWER(nom)=LOWER('Ravel')) ;
```

Musiciens n'ayant rien composé :

```
SELECT nom, prenom FROM Musicien mu
WHERE NOT EXISTS (
SELECT id_morceau FROM Morceau mo  WHERE mu.id_morceau=mo.id_musicien) ;
```

Opérateurs

Utilisation de fonctions

- Moyenne : AVG([DISTINCT]attribut)
- Comptage : COUNT([DISTINCT]attribut)
- Maximum : MAX([DISTINCT]attribut)
- Minimum : MIN([DISTINCT]attribut)
- Somme : SUM([DISTINCT]attribut)

Exemples

```
SELECT COUNT(*) FROM Musicien ;
SELECT MAX(age) FROM Musicien ;
Sélection du plus vieux musicien :
SELECT nom, prenom FROM Musicien
WHERE age=(SELECT MAX(age) FROM Musicien) ;
```

GROUP BY

Regroupement

- SELECT *attributs*
  FROM *table*
  GROUP BY *Liste attributs de regroupement* ;

permet de créer des sous-ensembles par rapport à un ou plusieurs critères de regroupement.
Les regroupements se font par égalité de valeur de l'attribut ; il y a autant de sous-groupes que de valeurs différentes.
La liste des attributs de regroupement doit contenir **exactement** la liste des attributs de l'instruction SELECT permettant le regroupement.

GROUP BY

Exemples

Nombre de morceaux par musicien

```
SELECT nom, prenom, COUNT(mo.id_morceau)
FROM Musicien mu,Morceau mo
WHERE mo.id_musicien=mu.id_musicien
GROUP BY nom, prenom ;
```

GROUP BY

Regroupement avec condition de restriction

- SELECT *attributs*
  FROM *table*
  GROUP BY *Liste attributs de regroupement*
  HAVING *prédicat sur attribut de regroupement* ;

permet de créer des sous-ensembles par rapport à un ou plusieurs critères de regroupement sur lesquels on applique un critère de restriction.

GROUP BY

Exemples

Nombre de morceaux par musicien ayant composé plus de 2 morceaux

```
SELECT nom, prenom, COUNT(mo.id_morceau)
FROM Musicien mu,Morceau mo
WHERE mo.id_musicien=mu.id_musicien
GROUP BY nom, prenom
HAVING COUNT(mo.id_morceau)>2;
```

Définition

Définition

Une jointure permet de récupérer les enregistrements dont un critère d'égalité est vérifié entre 2 tables.
Une jointure **externe** permet de récupérer les enregistrements auquels aucune correspondance n'est faite.

Définition

Exemple

Afficher les nom, prénom et nombre de morceaux composés pour chaque musicien

Définition

Exemple

Afficher les nom, prénom et nombre de morceaux composés pour chaque musicien

```
select nom, prenom, count(mo.idmusicien)
from Musicien mu, Morceau mo
where mu.idMusicien = mo.idMusicien
group by nom, prenom ;
```

Définition

Exemple

Afficher les nom, prénom et nombre de morceaux composés pour chaque musicien
PB : cette requête ne retourne que les musiciens pour lesquels il existe une jointure entre ces 2 tables
Solution : jointure externe

Définition

Types

Il existe 3 types de jointure **externe**

- à gauche : retourne tous les enregistrements de la relation de gauche même ceux qui n'ont pas de correspondance à droite
- à droite : l'inverse
- complète : renvoie seulement les enregistrements qui n'ont pas de correspondance

Définition

Syntaxe

select ...
from table_gauche *LEFT*|*RIGHT*|*FULL OUTER JOIN*
table_droite
ON critere_jointure
where critere_restriction

Solution

Exemple

Afficher les nom, prénom et nombre de morceaux composés pour chaque musicien

```
select nom, prenom, count(mo.idmusicien)
from Musicien mu left outer join Morceau mo on mu.idMusicien = mo.idMusicien
group by nom, prenom ;
```