Bases de données relationnelles

Olivier Caron

Polytech Lille
Avenue Paul Langevin Cité Scientifique Lille 1
59655 Villeneuve d'Ascq cedex

http://ocaron.plil.fr
Olivier.Caron@polytech-lille.fr

Plan

Partie I

Plan

Partie I

- *Le modèle relationnel*

Plan

Partie I

- *Le modèle relationnel*
- Le langage SQL, partie DDL (Data Definition Language)

Plan

Plan

Partie I

- *Le modèle relationnel*
- Le langage SQL, partie DDL (Data Definition Language)

Partie II

- *L'algèbre relationnelle (langage de requêtes)*

Plan

Partie I

- *Le modèle relationnel*
- Le langage SQL, partie DDL (Data Definition Language)

Partie II

- *L'algèbre relationnelle (langage de requêtes)*
- Le langage SQL, partie requêtes et modification de données

Le modèle relationnel en un seul slide

- L'universitaire Codd (1970) a inventé le modèle relationnel basé sur des concepts simples :

Le modèle relationnel en un seul slide

- L'universitaire Codd (1970) a inventé le modèle relationnel basé sur des concepts simples :
  - Une **base** de données correspond à un ensemble de **tables** (ou relations)

Le modèle relationnel en un seul slide

- L'universitaire Codd (1970) a inventé le modèle relationnel basé sur des concepts simples :
  - Une **base** de données correspond à un ensemble de **tables** (ou relations)
  - Une **table** est identifiée par un nom
    (pas deux tables de même nom dans une même base)

Le modèle relationnel en un seul slide

- L'universitaire Codd (1970) a inventé le modèle relationnel basé sur des concepts simples :
  - Une **base** de données correspond à un ensemble de **tables** (ou relations)
  - Une **table** est identifiée par un nom
    (pas deux tables de même nom dans une même base)
  - Une **table** contient des **colonnes**

Le modèle relationnel en un seul slide

- L'universitaire Codd (1970) a inventé le modèle relationnel basé sur des concepts simples :
  - Une **base** de données correspond à un ensemble de **tables** (ou relations)
  - Une **table** est identifiée par un nom
    (pas deux tables de même nom dans une même base)
  - Une **table** contient des **colonnes**
  - Une **colonne** est identifiée par un nom et correspond à un **type** (entier, réel, date,. . .)

Le modèle relationnel en un seul slide

- L'universitaire Codd (1970) a inventé le modèle relationnel basé sur des concepts simples :
  - Une **base** de données correspond à un ensemble de **tables** (ou relations)
  - Une **table** est identifiée par un nom
    (pas deux tables de même nom dans une même base)
  - Une **table** contient des **colonnes**
  - Une **colonne** est identifiée par un nom et correspond à un **type** (entier, réel, date,. . .)
  - Certaines colonnes peuvent être des **clés**.

Le modèle relationnel en un seul slide

- L'universitaire Codd (1970) a inventé le modèle relationnel basé sur des concepts simples :
  - Une **base** de données correspond à un ensemble de **tables** (ou relations)
  - Une **table** est identifiée par un nom
    (pas deux tables de même nom dans une même base)
  - Une **table** contient des **colonnes**
  - Une **colonne** est identifiée par un nom et correspond à un **type** (entier, réel, date,. . .)
  - Certaines colonnes peuvent être des **clés**.
  - Les lignes d'une table sont parfois appelés les **t-uples**.

Clés primaires et clés étrangères

- La valeur d'une colonne **clé primaire** permet d'**identifier** une ligne d'une table :
  Il n'existe donc pas deux lignes dans une même table qui ont une valeur identique pour la colonne de **clé primaire**.

Clés primaires et clés étrangères

- La valeur d'une colonne **clé primaire** permet d'**identifier** une ligne d'une table :
  Il n'existe donc pas deux lignes dans une même table qui ont une valeur identique pour la colonne de **clé primaire**.
- Une valeur d'une colonne **clé étrangère** correspond à une valeur d'une **clé primaire** dans la base.
  Cela permet de relier les tables entre elles.

Exemple : table Poste

Table: Poste

code PK integer	**nom** varchar(30)
1	Ingénieur Informaticien
2	Secrétaire
3	Président Directeur Général
4	...

PK : Primary Key

Exemple : table Employé

Table: Employe

code	**Nom**	**Prénom**	**refPoste**
PK			FK (poste)
integer	varchar(30)	varchar(20)	integer
1	Dupont	Paul	1
2	Durant	André	3
3	Dubois	Maxime	1
4	Dupont	Jean	2

PK : Primary Key, FK : Foreign Key

Exercice

Schéma relationnel

Donnez un schéma relationnel qui permet de décrire des étudiants caractérisés par un NIP, nom, prenom et date de naissance et des modules caractérisés par un code et un libellé. Décrire également le fait que des étudiants suivent certains modules.

Implantation dans les SGBDR

- 80% des SGBD (Systèmes de Gestion de Bases de données) sont des SGBD relationnels (SGBDR)

Implantation dans les SGBDR

- 80% des SGBD (Systèmes de Gestion de Bases de données) sont des SGBD relationnels (SGBDR)
- Tous désormais exploitent un même langage: le langage SQL (Structured Query Language)

Implantation dans les SGBDR

- 80% des SGBD (Systèmes de Gestion de Bases de données) sont des SGBD relationnels (SGBDR)
- Tous désormais exploitent un même langage: le langage SQL (Structured Query Language)
- SQL sert pour la consultation de la base (les requêtes)

Implantation dans les SGBDR

- 80% des SGBD (Systèmes de Gestion de Bases de données) sont des SGBD relationnels (SGBDR)
- Tous désormais exploitent un même langage: le langage SQL (Structured Query Language)
- SQL sert pour la consultation de la base (les requêtes)
- SQL sert pour la création et modification de la base

Syntaxe générale des commandes

- Pas de distinction minuscule et majuscule.

Syntaxe générale des commandes

- Pas de distinction minuscule et majuscule.
- Les commandes commencent par un mot clé servant à nommer l'opération de base à exécuter.

Syntaxe générale des commandes

- Pas de distinction minuscule et majuscule.
- Les commandes commencent par un mot clé servant à nommer l'opération de base à exécuter.
- Chaque commande SQL doit remplir deux exigences :

Syntaxe générale des commandes

- Pas de distinction minuscule et majuscule.
- Les commandes commencent par un mot clé servant à nommer l'opération de base à exécuter.
- Chaque commande SQL doit remplir deux exigences :
  - Indiquer les données sur lesquelles elle opère (un ensemble de lignes stockées dans une ou plusieurs classes)

Syntaxe générale des commandes

- Pas de distinction minuscule et majuscule.
- Les commandes commencent par un mot clé servant à nommer l'opération de base à exécuter.
- Chaque commande SQL doit remplir deux exigences :
  - Indiquer les données sur lesquelles elle opère (un ensemble de lignes stockées dans une ou plusieurs classes)
  - Indiquer l'opération à exécuter sur ces données

Les domaines principaux par défaut

- Les numériques : `integer, smallint, double, float,...`
  exemple : `23, -122, 3.4, -6.7,...`

Les domaines principaux par défaut

- Les numériques : `integer, smallint, double, float,...`
  exemple : `23, -122, 3.4, -6.7,...`
- Les alphanumériques : `char, varchar(n), char(n), text`
  exemples : `'v', 'la vie est un long fleuve ...'`

Les domaines principaux par défaut

- Les numériques : `integer, smallint, double, float,...`
  exemple : `23, -122, 3.4, -6.7,...`
- Les alphanumériques : `char, varchar(n), char(n), text`
  exemples : `'v', 'la vie est un long fleuve ...'`
- Le type date (format configurable)
  exemple : `'2002-02-28'`

Les domaines principaux par défaut

- Les numériques : `integer, smallint, double, float,...`
  exemple : `23, -122, 3.4, -6.7,...`
- Les alphanumériques : `char, varchar(n), char(n), text`
  exemples : `'v', 'la vie est un long fleuve ...'`
- Le type date (format configurable)
  exemple : `'2002-02-28'`
- Le type boolean : `boolean` ou `bool`
  exemples : `true` ou `'t'` et `false` ou `'f'`

Les domaines principaux par défaut

- Les numériques : `integer, smallint, double, float,...`
  exemple : `23, -122, 3.4, -6.7,...`
- Les alphanumériques : `char, varchar(n), char(n), text`
  exemples : `'v', 'la vie est un long fleuve ...'`
- Le type date (format configurable)
  exemple : `'2002-02-28'`
- Le type boolean : `boolean` ou `bool`
  exemples : `true` ou `'t'` et `false` ou `'f'`
- Et bien d'autres encore (time,timestamp, serial, ...)

Le SGBD Postgres

- Travaux de Stonebraker (Univ. Berkeley)

Le SGBD Postgres

- Travaux de Stonebraker (Univ. Berkeley)
- `http://www.postgresql.org/`

Le SGBD Postgres

- Travaux de Stonebraker (Univ. Berkeley)
- `http://www.postgresql.org/`
- Fonctionne sous Unix (linux, Solaris, Mac OS X) et Windows

Le SGBD Postgres

- Travaux de Stonebraker (Univ. Berkeley)
- `http://www.postgresql.org/`
- Fonctionne sous Unix (linux, Solaris, Mac OS X) et Windows
- Logiciel libre

Le SGBD Postgres

- Travaux de Stonebraker (Univ. Berkeley)
- `http://www.postgresql.org/`
- Fonctionne sous Unix (linux, Solaris, Mac OS X) et Windows
- Logiciel libre
- Architecture client/serveur (processus, TCP-IP)

Le SGBD Postgres

- Travaux de Stonebraker (Univ. Berkeley)
- `http://www.postgresql.org/`
- Fonctionne sous Unix (linux, Solaris, Mac OS X) et Windows
- Logiciel libre
- Architecture client/serveur (processus, TCP-IP)
- Les types d'utilisateurs Postgres :

Le SGBD Postgres

- Travaux de Stonebraker (Univ. Berkeley)
- `http://www.postgresql.org/`
- Fonctionne sous Unix (linux, Solaris, Mac OS X) et Windows
- Logiciel libre
- Architecture client/serveur (processus, TCP-IP)
- Les types d'utilisateurs Postgres :
  - L'utilisateur "**postgres**" dispose de tous les droits (root), création bases, utilisateurs, destruction,...

Le SGBD Postgres

- Travaux de Stonebraker (Univ. Berkeley)
- `http://www.postgresql.org/`
- Fonctionne sous Unix (linux, Solaris, Mac OS X) et Windows
- Logiciel libre
- Architecture client/serveur (processus, TCP-IP)
- Les types d'utilisateurs Postgres :
  - L'utilisateur **"postgres"** dispose de tous les droits (root), création bases, utilisateurs, destruction,...
  - Les **utilisateurs-administrateurs** : peuvent créer des bases, peuvent autoriser d'autres utilisateurs à accéder à ces bases.
    Certains utilisateurs-administrateurs peuvent créer d'autres utilisateurs.

Le SGBD Postgres

- Travaux de Stonebraker (Univ. Berkeley)
- `http://www.postgresql.org/`
- Fonctionne sous Unix (linux, Solaris, Mac OS X) et Windows
- Logiciel libre
- Architecture client/serveur (processus, TCP-IP)
- Les types d'utilisateurs Postgres :
  - L'utilisateur **"postgres"** dispose de tous les droits (root), création bases, utilisateurs, destruction,...
  - Les **utilisateurs-administrateurs** : peuvent créer des bases, peuvent autoriser d'autres utilisateurs à accéder à ces bases.
    Certains utilisateurs-administrateurs peuvent créer d'autres utilisateurs.
  - Les **utilisateurs** peuvent accéder à des bases (selon les droits)

Le réseau Polytech Lille

- Un serveur postgres situé sur le serveur `houplin.studserv.deule.net`
- Chaque étudiant dispose d'un compte **utilisateur-administrateur**
- Accès disponible à distance sur toutes les machines Polytech
- Configuration initiale (`.bashrc`) :

  ```
  export PGHOST=houplin.studserv.deule.net
  ```

- Par défaut : comptePostgres = compteUnix (sauf pour login avec '-'), password (postgres)

Les commandes de base POSTGRES

- Commandes sous unix

Les commandes de base POSTGRES

- Commandes sous unix
- Création d'une base ([] facultatif) :

```
createdb  nomBase [ -U comptePostgres ]
```

Les commandes de base POSTGRES

- Commandes sous unix
- Création d'une base ([] facultatif) :

```
createdb  nomBase [ -U comptePostgres ]
```

- Destruction d'une base :

```
dropdb nomBase [ -U comptePostgres ]
```

Les commandes de base POSTGRES

- Commandes sous unix
- Création d'une base ([] facultatif) :

```
createdb  nomBase [ -U comptePostgres ]
```

- Destruction d'une base :

```
dropdb nomBase [ -U comptePostgres ]
```

- Accès à une base :

```
psql nomBase [ -U comptePostgres ]
```

Environnements POSTGRES

- Utilitaire `psql`

Environnements POSTGRES

- Utilitaire `psql`
  - Interface texte :-) ou :-(

Environnements POSTGRES

- Utilitaire `psql`
  - Interface texte :-) ou :-(
  - Reconnaît toutes les requêtes SQL !

Environnements POSTGRES

- Utilitaire `psql`
  - Interface texte :-) ou :-(
  - Reconnaît toutes les requêtes SQL !
  - Et bien plus encore (aide en ligne,. . .)

Environnements POSTGRES

- Utilitaire `psql`
  - Interface texte :-) ou :-(
  - Reconnaît toutes les requêtes SQL !
  - Et bien plus encore (aide en ligne,. . .)
- Interface Web `http://pgsql.polytech-lille.fr`

Environnements POSTGRES

- Utilitaire `psql`
  - Interface texte :-) ou :-(
  - Reconnaît toutes les requêtes SQL !
  - Et bien plus encore (aide en ligne,. . .)
- Interface Web `http://pgsql.polytech-lille.fr`
  - Plus convivial,

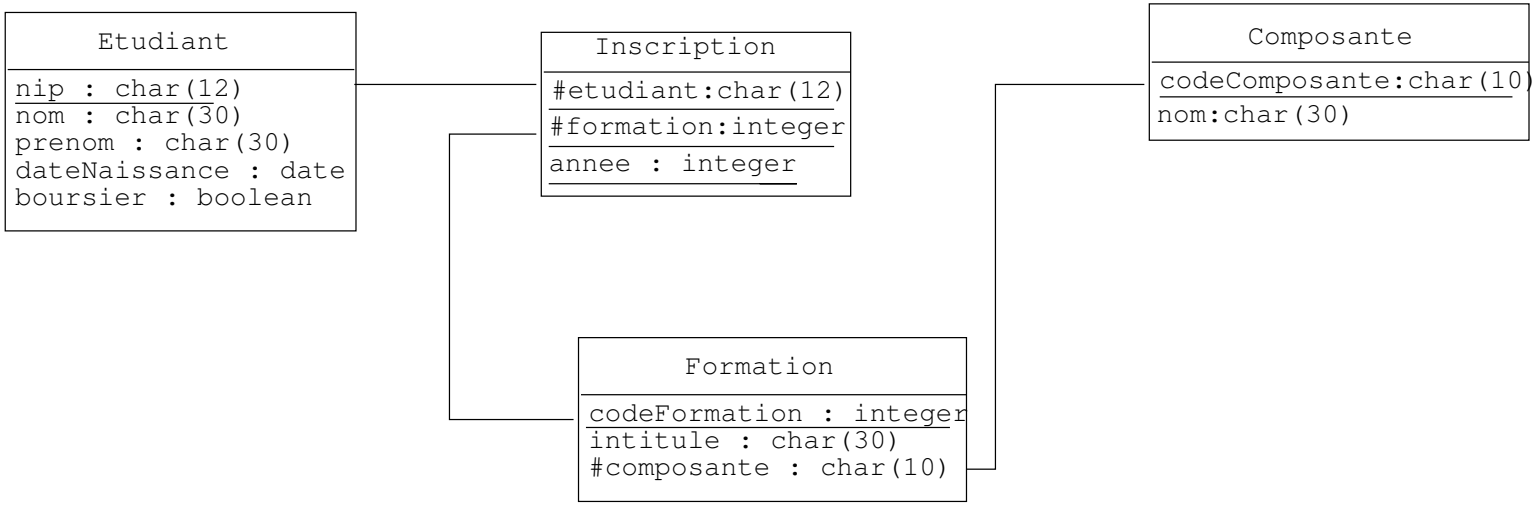
Environnements POSTGRES

- Utilitaire `psql`
  - Interface texte :-) ou :-(
  - Reconnaît toutes les requêtes SQL !
  - Et bien plus encore (aide en ligne,...)
- Interface Web `http://pgsql.polytech-lille.fr`
  - Plus convivial,
  - Ne nécessite pas la connaissance de SQL

Exemple schéma relationnel (1/2)

- Soit le schéma relationnel suivant :
  Etudiant (nip:char(12), nom:char(30), prenom:char(30), dateNaissance:date, boursier:boolean)
  Formation (codeFormation:integer,intitule:char(30), ♯composante(Composante):char(10))
  Composante (codeComposante:char(10),nom:char(30))
  Inscription (♯etudiant(Etudiant):char(12), ♯formation(Formation):integer,annee:integer)

Exemple schéma relationnel (2/2)



Représentation SQL d'un schéma relationnel (1/2)

```
CREATE TABLE Etudiant (
    nip CHAR(12) PRIMARY KEY,
    nom CHAR(30), prenom CHAR(30),
    dateNaissance date, boursier boolean
) ;

CREATE TABLE Composante (
    codeComposante CHAR(10) PRIMARY KEY,
    nom CHAR(30)
) ;
```

Représentation SQL d'un schéma relationnel (2/2)

```
CREATE TABLE Formation (
    codeFormation serial PRIMARY KEY, --generation automatique
    intitule CHAR(30),
    composante CHAR(10) REFERENCES Composante
) ;

CREATE TABLE Inscription (
    etudiant CHAR(12) REFERENCES Etudiant,
    formation INTEGER REFERENCES Formation,
    annee INTEGER,
    PRIMARY KEY (etudiant, formation, annee)
) ;
```

Schéma relationnel en SQL

- Mots-clefs en majuscules ou minuscules.

Schéma relationnel en SQL

- Mots-clefs en majuscules ou minuscules.
- `primary key` : valeur unique et non nulle au sein de la table

Schéma relationnel en SQL

- Mots-clefs en majuscules ou minuscules.
- `primary key` : valeur unique et non nulle au sein de la table
- Possibilité de définir la clé au niveau de l'attribut ou au niveau de la table (obligatoire si clé composée)

Suppression de tables

- A chaque commande `CREATE`, une commande `DROP`
- Syntaxes :

```
DROP TABLE name {, name}
```

Insertion de lignes

- Parfois considéré comme une commande purement administrateur
- Syntaxe (simplifiée) :

```
INSERT INTO table [ ( column {, ...} ) ]
    VALUES ( expression {, ...} ) | SELECT query
```

Exemples insertion

- On insère une ligne complète :

```
INSERT INTO composante VALUES ('EPU',
'Polytech') ;
```

Respecter l'ordre des colonnes définies à la création (et le type)

Exemples insertion

- On insère une ligne complète :

```
INSERT INTO composante VALUES ('EPU',
'Polytech') ;
```

  Respecter l'ordre des colonnes définies à la création (et le type)

- On insère une ligne complète en spécifiant les colonnes :

```
INSERT INTO composante(codeComposante,nom)
VALUES ('EPU', 'Polytech') ;
```

Exemples insertion

- On insère une ligne complète :

```
INSERT INTO composante VALUES ('EPU',
'Polytech') ;
```

  Respecter l'ordre des colonnes définies à la création (et le type)

- On insère une ligne complète en spécifiant les colonnes :

```
INSERT INTO composante(codeComposante,nom)
VALUES ('EPU', 'Polytech') ;
```

- On insère une ligne incomplète :

```
INSERT INTO formation(intitule,composante)
VALUES ('GIS', 'EPU');
```

Exemples insertion

- On insère une ligne complète :

```
INSERT INTO composante VALUES ('EPU',
'Polytech') ;
```

  Respecter l'ordre des colonnes définies à la création (et le type)

- On insère une ligne complète en spécifiant les colonnes :

```
INSERT INTO composante(codeComposante,nom)
VALUES ('EPU', 'Polytech') ;
```

- On insère une ligne incomplète :

```
INSERT INTO formation(intitule,composante)
VALUES ('GIS', 'EPU');
```

- Attention aux incohérences de la base !

Exemples insertion

- On insère une ligne complète :

```
INSERT INTO composante VALUES ('EPU',
'Polytech') ;
```

  Respecter l'ordre des colonnes définies à la création (et le type)

- On insère une ligne complète en spécifiant les colonnes :

```
INSERT INTO composante(codeComposante,nom)
VALUES ('EPU', 'Polytech') ;
```

- On insère une ligne incomplète :

```
INSERT INTO formation(intitule,composante)
VALUES ('GIS', 'EPU');
```

- Attention aux incohérences de la base !
- Valeurs nulles ou valeurs par défaut (notamment `serial`)

Suppression de lignes

- Syntaxe :

```
DELETE FROM nom_table [ WHERE condition ]
```

- Exemples :

```
DELETE FROM composante ;
DELETE FROM Etudiant where nip='cr0245zs' ;
```

Modification de lignes

- Syntaxe :

```
UPDATE nom_table SET col = expression {, col =
expression}
[ WHERE condition ]
```

- Exemples :
  - Modifier une colonne, pour une ligne :

```
UPDATE Composante SET nom='Polytech Lille'
WHERE codeComposante='EPU'
```

  - Modifier plusieurs colonnes pour une ligne :

```
UPDATE Etudiant SET
dateNaissance='27/02/1991',
boursier=true where nip='cr0123dz'
```

  - Modifier toutes les lignes :

```
UPDATE personnel SET
salaire=salaire+0.10*salaire
```

Application covoiturage (1/4)

Une startup désire mettre en place une application web d'aide au covoiturage. Voici la liste des fonctionnalités de cette application :

Inscription d'un membre Un membre peut aussi bien jouer le rôle de passager ou de conducteur de trajets. Toute personne peut s'inscrire à l'application en fournissant les informations suivantes : nom, prénom, date de naissance, le modèle de la voiture qu'il possède éventuellement, son n° de tél et son e-mail (il ne peut y avoir deux membres de même e-mail), un mot de passe. A ces informations de base, quelques informations booléennes supplémentaires permettent de définir le profil du membre : accepte ou pas les fumeurs, accepte ou pas les animaux, accepte la musique, aime peu (bla) ou beaucoup (blabla) discuter.

Application covoiturage (2/4)

Modification d'un membre Un membre peut consulter ses informations et les modifier. Pour accéder à ses informations, il doit fournir son email et mot de passe.

Application covoiturage (2/4)

Modification d'un membre Un membre peut consulter ses informations et les modifier. Pour accéder à ses informations, il doit fournir son email et mot de passe.

Informations sur un membre Tout membre peut obtenir des informations (à l'exception du mot de passe) sur un autre membre, il suffit d'indiquer l'e-mail du membre recherché.

Application covoiturage (3/4)

Proposition d'un trajet Tout membre disposant d'une voiture peut proposer des trajets. Les informations à fournir pour un trajet sont : la ville de départ, la ville d'arrivée, les adresses précises des départ et arrivée, la date et heure de départ, le nombre de places passagers disponibles, le prix d'une place, une estimation du nombre de kilomètres.

Application covoiturage (3/4)

Proposition d'un trajet Tout membre disposant d'une voiture peut proposer des trajets. Les informations à fournir pour un trajet sont : la ville de départ, la ville d'arrivée, les adresses précises des départ et arrivée, la date et heure de départ, le nombre de places passagers disponibles, le prix d'une place, une estimation du nombre de kilomètres.

Liste des trajets L'application doit permettre de fournir une liste de tous les trajets proposés.

Application covoiturage (3/4)

Proposition d'un trajet Tout membre disposant d'une voiture peut proposer des trajets. Les informations à fournir pour un trajet sont : la ville de départ, la ville d'arrivée, les adresses précises des départ et arrivée, la date et heure de départ, le nombre de places passagers disponibles, le prix d'une place, une estimation du nombre de kilomètres.

Liste des trajets L'application doit permettre de fournir une liste de tous les trajets proposés.

Affectation d'un trajet Tout membre peut se proposer comme passager d'un trajet. Le passager précise s'il emporte ou pas des bagages. L'acceptation est automatique.

Application covoiturage (4/4)

Administration Sur le site, des pages dédiées à l'administration permettent de tout modifier (supprimer un membre, un trajet, etc). L'accès à ces pages est sécurisé par mot de passe.

Application covoiturage (4/4)

Administration Sur le site, des pages dédiées à l'administration permettent de tout modifier (supprimer un membre, un trajet, etc). L'accès à ces pages est sécurisé par mot de passe.

schéma relationnel

Concevoir le schéma relationnel des informations nécessaires pour cette application. Ecrire ce schéma relationnel dans le langage SQL.

Algèbre relationnelle

- Egalement définie par Codd (1970)

Algèbre relationnelle

- Egalement définie par Codd (1970)
- Basée sur des opérateurs algébriques simples

Algèbre relationnelle

- Egalement définie par Codd (1970)
- Basée sur des opérateurs algébriques simples
- Construire des requêtes par composition de ces opérateurs

Algèbre relationnelle

- Egalement définie par Codd (1970)
- Basée sur des opérateurs algébriques simples
- Construire des requêtes par composition de ces opérateurs
- SQL est utilisé à nouveau pour les requêtes de consultation

Exemple base

vins1	numero	cru	millesime	degré
	100	Chablis	1974	12
	110	Mercurey	1978	13
	120	Meursault	1977	12

Consultation simple d'une table

— *requete usuelle :*

```
select * from vins1 ;
```

— *definition explicites des colonnes de la table :*
— *ordre des colonnes quelconque*

```
select numero, cru, millesime, degre from vins1 ;
```

— *utilisation des noms de table*

```
select vins1.* from vins1
```

— *utilisation d'une variable pour le nom de la table :*

```
select tablevins.* from vins1 tablevins ;
```

Opérateur de projection

- Projection: on sélectionne certaines colonnes
- Possibilité de renommage des colonnes (clause "as")
- Clause "distinct" pour supprimer des lignes identiques

```
select distinct millesime, degre from vins-3
```

projection	Millésime	Degré
	Chablis	12
	Mercurey	13
	Meursault	12

Opérateur de restriction

- Restriction : on supprime des lignes
- Introduction clause "where"
- La qualification peut être exprimée à l'aide de constantes, comparateurs arithmétiques, opérateurs logiques

```
select * from vins1
where degre=12 and millesime>1974
```

restriction	Numéro	Cru	Millésime	Degré
	120	Meursault	1977	12

Jointure de tables

- **La jointure** permet de relier deux tables (PK=FK)

table Parent		**table Enfant**		
nom	numero	num_e	prenom	num_pere
dupont	1	1	jérome	1
durant	2	2	alain	2
		3	pierre	1

```
select nom, numero, num_e, prenom, num_pere
from parent join enfant on num_pere=numero
```

nom	numero	num_e	prenom	num_pere
dupont	1	1	jérome	1
durant	2	2	alain	2
dupont	1	3	pierre	1

Et des opérateurs de calcul

- Existent dans la plupart des S.G.B.D.
- Donnent un résultat de type relation !
- Quelques illustrations :
  - L'opérateur `count()` calcul du nombre de lignes d'une table T
  - L'opérateur `sum()` calcul de la somme cumulée des valeurs d'un attribut.
  - et aussi `avg(), max(), min()`
  - `upper(), now(), ...`
  - Par défaut, le nom de la colonne résultat est le nom de la fonction (sauf si présence clause de renommage 'as')
  - Exemples :

```
select max(degre) as max_degre from vins1
select count(*) as nbvins from vins1
```

Traitement de chaînes

- Accès très simple : ch1 = ch2
- Opérateur `LIKE`
- Caractère spéciaux : '%' (remplace de 0 à plusieurs caractères) et '_' (remplace exactement un caractère).
- Opérateur de comparaison '>', '<', ... (ordre lexicographique)
- Opérateur de concaténation ||, fonctions prédéfinies (ex: upper)
- Exemple :

```
select * from vins1 where upper(cru) like 'ME%'
```

Présentation des données

- Ordre d'affichage des colonnes
- Clause `distinct`, évite les doublons
- Ordre d'affichage des lignes, clause `Order By`
- Ordre des lignes multi-critères
- Aucun impact sur le traitement algébrique des requêtes

```
select * from vins1 order by degre desc, cru asc
```

En résumé

- Le langage est simple !

En résumé

- Le langage est simple !
- Répond à la majorité des requêtes

En résumé

- Le langage est simple !
- Répond à la majorité des requêtes
- Langage en constante évolution

En résumé

- Le langage est simple !
- Répond à la majorité des requêtes
- Langage en constante évolution
- **Attention !** Cette présentation est une introduction du langage SQL