# Les Bases de Données Relationnelles

## BDR, SGBDR, L'Algèbre Relationnelle, SQL, SQLite

Anis SAIED

Insitut Préparatoire aux Etudes d'Ingénieurs de Nabeul
.:: UNIVERSITE DE CARTHAGE ::.

AU 2016-2017

https://cahier-de-prepa.fr/info-ipein/
anis.saieed@gmail.com

# Base de données relationnelle : Définition

- En informatique, une base de données relationnelle est une base de données où l'information est organisée dans des tableaux à deux dimensions appelés des relations ou tables.

# Base de données relationnelle : Définition

- En informatique, une base de données relationnelle est une base de données où l'information est organisée dans des tableaux à deux dimensions appelés des relations ou tables.
- Selon ce modèle relationnel, une base de données consiste en une ou plusieurs relations.

# Base de données relationnelle : Définition

- En informatique, une base de données relationnelle est une base de données où l'information est organisée dans des tableaux à deux dimensions appelés des relations ou tables.
- Selon ce modèle relationnel, une base de données consiste en une ou plusieurs relations.
- Les lignes de ces relations sont appelées des *nuplets* ou enregistrements.

# Base de données relationnelle : Définition

- En informatique, une base de données relationnelle est une base de données où l'information est organisée dans des tableaux à deux dimensions appelés des relations ou tables.
- Selon ce modèle relationnel, une base de données consiste en une ou plusieurs relations.
- Les lignes de ces relations sont appelées des *nuplets* ou enregistrements.
- Les noms des colonnes sont appelées des attributs.

# Base de données relationnelle : Définition

- En informatique, une base de données relationnelle est une base de données où l'information est organisée dans des tableaux à deux dimensions appelés des relations ou tables.
- Selon ce modèle relationnel, une base de données consiste en une ou plusieurs relations.
- Les lignes de ces relations sont appelées des *nuplets* ou enregistrements.
- Les noms des colonnes sont appelées des attributs.
- Les logiciels qui permettent de créer, utiliser et maintenir des bases de données relationnelles sont des systèmes de gestion de base de données relationnels (SGBDR).

# Base de données relationnelle : Définition

- En informatique, une base de données relationnelle est une base de données où l'information est organisée dans des tableaux à deux dimensions appelés des relations ou tables.
- Selon ce modèle relationnel, une base de données consiste en une ou plusieurs relations.
- Les lignes de ces relations sont appelées des *nuplets* ou enregistrements.
- Les noms des colonnes sont appelées des attributs.
- Les logiciels qui permettent de créer, utiliser et maintenir des bases de données relationnelles sont des systèmes de gestion de base de données relationnels (SGBDR).
- La théorie : l'algèbre relationnelle.

# Base de données relationnelle : Définition

- En informatique, une base de données relationnelle est une base de données où l'information est organisée dans des tableaux à deux dimensions appelés des relations ou tables.
- Selon ce modèle relationnel, une base de données consiste en une ou plusieurs relations.
- Les lignes de ces relations sont appelées des *nuplets* ou enregistrements.
- Les noms des colonnes sont appelées des attributs.
- Les logiciels qui permettent de créer, utiliser et maintenir des bases de données relationnelles sont des systèmes de gestion de base de données relationnels (SGBDR).
- La théorie : l'algèbre relationnelle.
- La pratique : SQL (Voir TP).

# Algèbre relationnelle : Définition

- L'algèbre relationnelle permet de faire des recherches dans les relations
- Ensemble d'**opérateurs** qui s'appliquent aux relations
- **Opérandes :** relations du modèle relationnel
- Résultat : nouvelle relation qui peut à son tour être manipulée

# DB Browser for SQLite

- `DB Browser for SQLite` est un outil visuel utilisé pour créer et éditer des fichiers de bases de données relationnelles avec `SQLite`.

# DB Browser for SQLite

- `DB Browser for SQLite` est un outil visuel utilisé pour créer et éditer des fichiers de bases de données relationnelles avec `SQLite`.
- Créer, définir, modifier et supprimer des tables

# DB Browser for SQLite

- `DB Browser for SQLite` est un outil visuel utilisé pour créer et éditer des fichiers de bases de données relationnelles avec `SQLite`.
- Créer, définir, modifier et supprimer des tables
- Parcourir, modifier, ajouter et supprimer des enregistrements

# DB Browser for SQLite

- `DB Browser for SQLite` est un outil visuel utilisé pour créer et éditer des fichiers de bases de données relationnelles avec `SQLite`.
- Créer, définir, modifier et supprimer des tables
- Parcourir, modifier, ajouter et supprimer des enregistrements
- Rechercher des enregistrements

# DB Browser for SQLite

- `DB Browser for SQLite` est un outil visuel utilisé pour créer et éditer des fichiers de bases de données relationnelles avec `SQLite`.
- Créer, définir, modifier et supprimer des tables
- Parcourir, modifier, ajouter et supprimer des enregistrements
- Rechercher des enregistrements
- Émettre des requêtes SQL et afficher les résultats

# DB Browser for SQLite

- `DB Browser for SQLite` est un outil visuel utilisé pour créer et éditer des fichiers de bases de données relationnelles avec `SQLite`.
- Créer, définir, modifier et supprimer des tables
- Parcourir, modifier, ajouter et supprimer des enregistrements
- Rechercher des enregistrements
- Émettre des requêtes SQL et afficher les résultats
- ...

# DB Browser for SQLite

- `DB Browser for SQLite` est un outil visuel utilisé pour créer et éditer des fichiers de bases de données relationnelles avec `SQLite`.
- Créer, définir, modifier et supprimer des tables
- Parcourir, modifier, ajouter et supprimer des enregistrements
- Rechercher des enregistrements
- Émettre des requêtes SQL et afficher les résultats
- ...
- Essayer !

# Plan du cours

# Plan du cours

# Plan du cours

# Notions de base : Attributs

Description de problème ≪ Gestion d'une bibliothèque ≫ par le modèle relationnel. On se donne

**$\mathcal{A}$, l'ensemble des *attributs* (on dit aussi *champs ou colonnes*)**

Les livres ont comme attributs
$\mathcal{A} = \{$numéro, titre, auteur, année$\}$

# Notions de base : Domaines

**$\mathcal{D}$, l'ensemble des *valeurs* qui peuvent être prises par les attributs**

Pour un attribut $A \in \mathcal{A}$, $D(A)$ est le domaine (type) de $A$.
ex : $D(\text{numéro}) = \text{int}$, $D(\text{titre}) = \text{string}$, $D(\text{auteur}) = \text{string}$, $D(\text{année}) = \text{int}$.

Chaque domaine vient avec son propre ensemble d'opérations :

- Opérations booléennes pour **BOOLEAN**, non-applicables pour les autres types.
- Opérations arithmétiques pour **INTEGER**, **FLOAT**, **DOUBLE**.
- Opérations spécifiques sur les chaînes de caractères **STRING**.

# Schéma

Un *schéma relationnel* $S$ est donné par une famille finie d'attributs $S = (A_1, \ldots, A_n) \in \mathcal{A}^n$, avec $i \neq j \Rightarrow A_i \neq A_j$,
et le domaine associée pour chaque attribut.

On le note aussi $S = ((A_1, D(A_1)), \ldots, (A_n, D(A_n))$
ex :
Livre = ((numéro,int),(titre,string),(auteur,string),(année,int))

# Relation, table, enregistrement

Une *relation* $R$ de schéma $S$ est
$R \subset D(A_1) \times \ldots D(A_n)$, avec $R$ finie.

Notée $R_S$ ou $R(S)$, appelée aussi *table*.
Les éléments de $R$ sont appelés *enregistrements* de la table $R$.

ex : la table $R_1$ du schéma *Livre* : $R_1$=R(Livre)
$R_1$= ((numéro,int),(titre,string),(auteur,string),(année,int))

| numéro | titre | auteur | année |
|---|---|---|---|
| 1 | Anna Karénine | Léon Tolstoï | 1877 |
| 2 | Algèbre I | Nicolas Bourbaki | 1970 |
| 3 | Topologie générale | Nicolas Bourbaki | 1971 |
| 4 | Informatique pour tous | Wack,Courant, De Falco,... | 2013 |

# Les opérations ensemblistes : Définitions

- Il existe six opérateurs d'algèbre relationnelle (Sélection, Projection, Renommage, Union, Différence et Produit cartésien).

# Les opérations ensemblistes : Définitions

- Il existe six opérateurs d'algèbre relationnelle (Sélection, Projection, Renommage, Union, Différence et Produit cartésien).
- À partir de ces opérateurs, des opérateurs dérivés peuvent être conçus comme la jointure, l'intersection ou la division.

# Les opérations ensemblistes : Définitions

- Il existe six opérateurs d'algèbre relationnelle (Sélection, Projection, Renommage, Union, Différence et Produit cartésien).
- À partir de ces opérateurs, des opérateurs dérivés peuvent être conçus comme la jointure, l'intersection ou la division.
- Deux types d'opérateurs :

# Les opérations ensemblistes : Définitions

- Il existe six opérateurs d'algèbre relationnelle (Sélection, Projection, Renommage, Union, Différence et Produit cartésien).
- À partir de ces opérateurs, des opérateurs dérivés peuvent être conçus comme la jointure, l'intersection ou la division.
- Deux types d'opérateurs :
  - Opérateurs unaires (un seul argument) : fournissent une nouvelle table à partir d'une seule table.

# Les opérations ensemblistes : Définitions

- Il existe six opérateurs d'algèbre relationnelle (Sélection, Projection, Renommage, Union, Différence et Produit cartésien).
- À partir de ces opérateurs, des opérateurs dérivés peuvent être conçus comme la jointure, l'intersection ou la division.
- Deux types d'opérateurs :
  - Opérateurs unaires (un seul argument) : fournissent une nouvelle table à partir d'une seule table.
  - Opérateurs binaires (deux arguments) : fournissent une nouvelle table à partir de deux tables.

# Projection

Soit $S = (A_1, \ldots, A_n)$ un schéma,
pour un enregistrement $e \in R(S)$ et un attribut $A \in S$,
**on note** $e.A$ **ou** $\pi_A(e)$ **la projection de** $e$ **sur** $A$ **:**

$$\forall e \in R(S), e = (e.A_1, \ldots, e.A_n) = (\pi_{A_1}(e), \ldots, \pi_{A_n}(e))$$

$$\pi_A(e) \rightarrow D(A)$$

ex : pour $e = (1, \text{Anna Karénine}, \text{Léon Tolstoï}, 1877)$

$$\pi_{\text{année}}(e) = 1877$$

# Projection sur une famille d'attributs

**On étend la définition de la projection aux familles d'attributs : pour $X = (A_1, \ldots, Ap) \subset S$ on définit**

$$\pi_X(e) = (e.A_1, \ldots, e.A_p)$$

Par extension, $\pi_X(R) = \{\pi_X(e) \mid e \in E\}$

ex : $\pi_{\text{numéro,année}}(e) = (1, 1877)$

# Projection : Exercice

On rappelle $R_1 =$

| numéro | titre | auteur | année |
|---|---|---|---|
| 1 | Anna Karénine | Léon Tolstoï | 1877 |
| 2 | Algèbre I | Nicolas Bourbaki | 1970 |
| 3 | Topologie générale | Nicolas Bourbaki | 1971 |
| 4 | Informatique pour tous | Wack,Courant, De Falco,... | 2013 |

- déterminer $\pi_{\text{numéro,auteur}}(R_1)$ sous forme de table,
- $\pi_{\{\text{numéro,auteur}\}}(R_1) =$

| numéro | auteur |
|---|---|
| 1 | Léon Tolstoï |
| 2 | Nicolas Bourbaki |
| 3 | Nicolas Bourbaki |
| 4 | Wack, Courant, De Falco,... |

# Projection : A retenir

- La projection est un opérateur unaire qui produit une relation R'

# Projection : A retenir

- La projection est un opérateur unaire qui produit une relation R'
- ayant les mêmes tuples/enregistrements que la relation R de départ,

# Projection : A retenir

- La projection est un opérateur unaire qui produit une relation R'
- ayant les mêmes tuples/enregistrements que la relation R de départ,
- mais dont le schéma est réduit à un ensemble d'attributs déterminés

# Projection : A retenir

- La projection est un opérateur unaire qui produit une relation R'
- ayant les mêmes tuples/enregistrements que la relation R de départ,
- mais dont le schéma est réduit à un ensemble d'attributs déterminés
- La projection réduit la taille de la relation horizontalement

# Projection : A retenir

- La projection est un opérateur unaire qui produit une relation R'
- ayant les mêmes tuples/enregistrements que la relation R de départ,
- mais dont le schéma est réduit à un ensemble d'attributs déterminés
- La projection réduit la taille de la relation horizontalement
- Notation

$$R' = \pi_{A1,A2,...,An}(R)$$

# Projection : A retenir

- La projection est un opérateur unaire qui produit une relation R'
- ayant les mêmes tuples/enregistrements que la relation R de départ,
- mais dont le schéma est réduit à un ensemble d'attributs déterminés
- La projection réduit la taille de la relation horizontalement
- Notation

$$R' = \pi_{A1,A2,...,An}(R)$$

- Élimine les doublons

# Projection *RealisationSQL*

- `SELECT A1, A2,..., An FROM R`

# Projection *RealisationSQL*

- `SELECT A1, A2,..., An FROM R`
- SQL laisse les doublons a priori ! D'où

# Projection *RealisationSQL*

- `SELECT A1, A2,..., An FROM R`
- SQL laisse les doublons a priori ! D'où
- `SELECT DISTINCT A1, A2,..., An FROM R`

# Projection *RealisationSQL*

- `SELECT A1, A2,..., An FROM R`
- SQL laisse les doublons a priori ! D'où
- `SELECT DISTINCT A1, A2,..., An FROM R`
- Exemple : sans doublons
  `SELECT DISTINCT numero, auteur FROM documents`

# Projection *RealisationSQL*

- `SELECT A1, A2,..., An FROM R`
- SQL laisse les doublons a priori ! D'où
- `SELECT DISTINCT A1, A2,..., An FROM R`
- Exemple : sans doublons
  `SELECT DISTINCT numero, auteur FROM documents`
- Exemple : avec doublons
  `SELECT ALL numero, auteur FROM documents`

# Sélection(ou restriction)

**Pour $A$ un attribut de $S$ et un prédicat $P$ sur $D(A)$ :**

$$P : D(A) \rightarrow \{vrai, faux\}$$

**on définit la *sélection* $\sigma_{P(A)}$ par**

$$\sigma_{P(A)}(R) = \{e \in R \mid P(e.A)\}$$

Exemple : $\sigma_{1900<date<2000}(R_1) =$
$\{(2, \text{Algèbre I}, \text{Nicolas Bourbaki}, 1970),$
$(3, \text{Topologie générale}, \text{Nicolas Bourbaki}, 1971)\} =$

| numéro | titre | auteur | année |
|---|---|---|---|
| 2 | Algèbre I | Nicolas Bourbaki | 1970 |
| 3 | Topologie générale | Nicolas Bourbaki | 1971 |

# Sélection : A retenir

- La sélection (ou restriction) est un opérateur unaire qui produit une relation R'

# Sélection : A retenir

- La sélection (ou restriction) est un opérateur unaire qui produit une relation R'
- ayant les mêmes attributs que la relation R de départ,

# Sélection : A retenir

- La sélection (ou restriction) est un opérateur unaire qui produit une relation R'
- ayant les mêmes attributs que la relation R de départ,
- mais dont les tuples/enregistrements sont uniquement ceux de la relation de départ qui satisfont une condition P donnée sur les valeurs des attributs

# Sélection : A retenir

- La sélection (ou restriction) est un opérateur unaire qui produit une relation R'
- ayant les mêmes attributs que la relation R de départ,
- mais dont les tuples/enregistrements sont uniquement ceux de la relation de départ qui satisfont une condition P donnée sur les valeurs des attributs
- La restriction réduit la taille de la relation verticalement

# Sélection : A retenir

- La sélection (ou restriction) est un opérateur unaire qui produit une relation R'
- ayant les mêmes attributs que la relation R de départ,
- mais dont les tuples/enregistrements sont uniquement ceux de la relation de départ qui satisfont une condition P donnée sur les valeurs des attributs
- La restriction réduit la taille de la relation verticalement
- Notation

$$R' = \sigma_P(R)$$

# Sélection : A retenir

- La sélection (ou restriction) est un opérateur unaire qui produit une relation R'
- ayant les mêmes attributs que la relation R de départ,
- mais dont les tuples/enregistrements sont uniquement ceux de la relation de départ qui satisfont une condition P donnée sur les valeurs des attributs
- La restriction réduit la taille de la relation verticalement
- Notation

$$R' = \sigma_P(R)$$

- où P est la condition (prédicat), qui peut porter sur un ou plusieurs attributs de R.

# Sélection *RealisationSQL*

- `SELECT * FROM R WHERE P`

# Sélection *RealisationSQL*

- `SELECT * FROM R WHERE P`
- Exemple :
  si l'attribut de type **Integer** :
  `SELECT * FROM documents WHERE 1900<date and date <2000`

# Sélection *RealisationSQL*

- `SELECT * FROM R WHERE P`
- Exemple :
  si l'attribut de type **Integer** :
  `SELECT * FROM documents WHERE 1900<date and date <2000`

  ou encore
  `SELECT * FROM film WHERE date between 1900 and 2000`

# Sélection *RealisationSQL*

- `SELECT * FROM R WHERE P`
- Exemple :
  si l'attribut de type **Integer** :
  `SELECT * FROM documents WHERE 1900<date and date <2000`

  ou encore
  `SELECT * FROM film WHERE date between 1900 and 2000`
- si l'attribut de type **Date** :
  `SELECT * FROM personne WHERE date between '1900-01-01' and '2000-01-01'`

# Prédicats

- On peut utiliser de nombreux prédicats pour les sélections : $<, >, \leq, \geq, =$ sur les nombres,

# Prédicats

- On peut utiliser de nombreux prédicats pour les sélections : $<, >, \leq, \geq, =$ sur les nombres,
- mais aussi sur les chaînes (ordre alphabétique, sous-chaînes, motifs, etc),

# Prédicats

- On peut utiliser de nombreux prédicats pour les sélections : $<, >, \leq, \geq, =$ sur les nombres,
- mais aussi sur les chaînes (ordre alphabétique, sous-chaînes, motifs, etc),
- et les composer entre eux, avec les connecteurs logiques : et, ou, non.

# Prédicats

- On peut utiliser de nombreux prédicats pour les sélections : $<, >, \leq, \geq, =$ sur les nombres,
- mais aussi sur les chaînes (ordre alphabétique, sous-chaînes, motifs, etc),
- et les composer entre eux, avec les connecteurs logiques : et, ou, non.
- Détails : voir TP (langage SQL).

# Prédicats

- On peut utiliser de nombreux prédicats pour les sélections : $<, >, \leq, \geq, =$ sur les nombres,
- mais aussi sur les chaînes (ordre alphabétique, sous-chaînes, motifs, etc),
- et les composer entre eux, avec les connecteurs logiques : et, ou, non.
- Détails : voir TP (langage SQL).
- ex : $\pi_{\text{année}}(\sigma_{\text{De Falco}\in\text{auteur}}(R_1)) = \{(2013)\}$

# Prédicats

- On peut utiliser de nombreux prédicats pour les sélections : $<, >, \leq, \geq, =$ sur les nombres,
- mais aussi sur les chaînes (ordre alphabétique, sous-chaînes, motifs, etc),
- et les composer entre eux, avec les connecteurs logiques : et, ou, non.
- Détails : voir TP (langage SQL).
- ex : $\pi_{\text{année}}(\sigma_{\text{De Falco} \in \text{auteur}}(R_1)) = \{(2013)\}$
- `SELECT annee FROM documents WHERE auteur LIKE "%De Falco%"`

# Prédicats

- On peut utiliser de nombreux prédicats pour les sélections : $<, >, \leq, \geq, =$ sur les nombres,
- mais aussi sur les chaînes (ordre alphabétique, sous-chaînes, motifs, etc),
- et les composer entre eux, avec les connecteurs logiques : et, ou, non.
- Détails : voir TP (langage SQL).
- ex : $\pi_{\text{année}}(\sigma_{\text{De Falco} \in \text{auteur}}(R_1)) = \{(2013)\}$
- `SELECT annee FROM documents WHERE auteur LIKE "%De Falco%"`

Réponse

| année |
|---|
| 2013 |

# Renommage

- but : résoudre des problèmes de compatibilité entre noms d'attributs de deux relations opérandes d'une opération binaire

# Renommage

- but : résoudre des problèmes de compatibilité entre noms d'attributs de deux relations opérandes d'une opération binaire
- Le renommage est un opérateur unaire qui produit une relation R'

# Renommage

- but : résoudre des problèmes de compatibilité entre noms d'attributs de deux relations opérandes d'une opération binaire
- Le renommage est un opérateur unaire qui produit une relation R'
- ayant les mêmes tuples/enregistrements que la relation R de départ,

# Renommage

- but : résoudre des problèmes de compatibilité entre noms d'attributs de deux relations opérandes d'une opération binaire
- Le renommage est un opérateur unaire qui produit une relation R'
- ayant les mêmes tuples/enregistrements que la relation R de départ,
- ayant le même schéma que la relation R de départ,

# Renommage

- but : résoudre des problèmes de compatibilité entre noms d'attributs de deux relations opérandes d'une opération binaire
- Le renommage est un opérateur unaire qui produit une relation R'
- ayant les mêmes tuples/enregistrements que la relation R de départ,
- ayant le même schéma que la relation R de départ,
- mais dans lequel un attribut a été renommé.

# Renommage

- but : résoudre des problèmes de compatibilité entre noms d'attributs de deux relations opérandes d'une opération binaire
- Le renommage est un opérateur unaire qui produit une relation R'
- ayant les mêmes tuples/enregistrements que la relation R de départ,
- ayant le même schéma que la relation R de départ,
- mais dans lequel un attribut a été renommé.
- Notation

$$R' = \alpha[nomattribut : nouveaunom](R)$$

# Renommage

- but : résoudre des problèmes de compatibilité entre noms d'attributs de deux relations opérandes d'une opération binaire
- Le renommage est un opérateur unaire qui produit une relation R'
- ayant les mêmes tuples/enregistrements que la relation R de départ,
- ayant le même schéma que la relation R de départ,
- mais dans lequel un attribut a été renommé.
- Notation

$$R' = \alpha[nomattribut : nouveaunom](R)$$

- précondition : le nouveau nom d'attribut n'est pas déjà le nom d'un attribut de R

# Renommage *RealisationSQL*

- `SELECT A1 AS A'1 FROM R`

# Renommage *RealisationSQL*

- `SELECT A1 AS A'1 FROM R`
- Intérêt : permet de résoudre des problèmes de compatibilité entre noms d'attributs de 2 tables opérandes d'une opération binaire (vues plus tard)

# Renommage *RealisationSQL*

- `SELECT A1 AS A'1 FROM R`
- Intérêt : permet de résoudre des problèmes de compatibilité entre noms d'attributs de 2 tables opérandes d'une opération binaire (vues plus tard)
- Exemple :
  Renommer l'attribut 'auteur' par 'auteur_livre'

# Renommage *RealisationSQL*

- `SELECT A1 AS A'1 FROM R`
- Intérêt : permet de résoudre des problèmes de compatibilité entre noms d'attributs de 2 tables opérandes d'une opération binaire (vues plus tard)
- Exemple :
  Renommer l'attribut 'auteur' par 'auteur_livre'
- solution :
  `SELECT auteur AS auteur_livre FROM Livre`
  Le résultat est une nouvelle table temporaire : Une seule colonne.

# Renommage *RealisationSQL*

- `SELECT A1 AS A'1 FROM R`
- Intérêt : permet de résoudre des problèmes de compatibilité entre noms d'attributs de 2 tables opérandes d'une opération binaire (vues plus tard)
- Exemple :
  Renommer l'attribut 'auteur' par 'auteur_livre'
- solution :
  `SELECT auteur AS auteur_livre FROM Livre`
  Le résultat est une nouvelle table temporaire : Une seule colonne.

# Renommage *RealisationSQL*

- `SELECT A1 AS A'1 FROM R`
- Intérêt : permet de résoudre des problèmes de compatibilité entre noms d'attributs de 2 tables opérandes d'une opération binaire (vues plus tard)
- Exemple :
  Renommer l'attribut 'auteur' par 'auteur_livre'
- solution :
  `SELECT auteur AS auteur_livre FROM Livre`
  Le résultat est une nouvelle table temporaire : Une seule colonne.
  `SELECT numero, titre, auteur AS auteur_livre , année FROM Livre`

- Passons aux opérateurs binaires.

# Union

- L'union est un opérateur binaire qui produit une relation T

# Union

- L'union est un opérateur binaire qui produit une relation T
- à partir de deux relations R et S,

# Union

- L'union est un opérateur binaire qui produit une relation T
- à partir de deux relations R et S,
- dont le schéma est le schéma commun à R et S,

# Union

- L'union est un opérateur binaire qui produit une relation T
- à partir de deux relations R et S,
- dont le schéma est le schéma commun à R et S,
- et dont les tuples/enregistrements sont ceux qui appartiennent à R ou à S.

# Union

- L'union est un opérateur binaire qui produit une relation T
- à partir de deux relations R et S,
- dont le schéma est le schéma commun à R et S,
- et dont les tuples/enregistrements sont ceux qui appartiennent à R ou à S.
- Élimination des doublons.

# Union

- L'union est un opérateur binaire qui produit une relation T
- à partir de deux relations R et S,
- dont le schéma est le schéma commun à R et S,
- et dont les tuples/enregistrements sont ceux qui appartiennent à R ou à S.
- Élimination des doublons.
- Notation

$$T = R \cup S$$

# Union

- L'union est un opérateur binaire qui produit une relation T
- à partir de deux relations R et S,
- dont le schéma est le schéma commun à R et S,
- et dont les tuples/enregistrements sont ceux qui appartiennent à R ou à S.
- Élimination des doublons.
- Notation

$$T = R \cup S$$

- Exemple :

# Union : Exemple

- Table : Enseignant

| nom |
|---|
| A |
| B |
| C |

- Table : Etudiant

| nom |
|---|
| D |
| C |
| E |

- Noms des personnes à l'institut ?
- Résultat = Enseignant $\cup$ Etudiant (Table sans doublons)

| nom |
|---|
| A |
| B |
| C |
| D |
| E |

# Union : *RealisationSQL*

- SELECT ...
  UNION
  SELECT ...

# Union : *RealisationSQL*

- ```
  SELECT ...
  UNION
  SELECT ...
  ```

- Exemple :

  ```
  SELECT * FROM Enseignant
  UNION
  SELECT * FROM Eutudiant
  ```
  ```

# Union : *RealisationSQL*

- ```
  SELECT ...
  UNION
  SELECT ...
  ```

- Exemple :

  ```
  SELECT * FROM Enseignant
  UNION
  SELECT * FROM Eutudiant
  ```

- Pour l'utiliser il est nécessaire que chacune des requêtes à concaténer retournes le même nombre de colonnes, avec les mêmes types de données et dans le même ordre.
  ```

# Union : *RealisationSQL*

- par défaut, les enregistrements exactement identiques ne seront pas répétés dans les résultats. Pour effectuer une union dans laquelle même les lignes dupliquées sont affichées il faut plutôt utiliser la commande `UNION ALL`.

# Union : *RealisationSQL*

- par défaut, les enregistrements exactement identiques ne seront pas répétés dans les résultats. Pour effectuer une union dans laquelle même les lignes dupliquées sont affichées il faut plutôt utiliser la commande `UNION ALL`.

- Exemple :

```
SELECT * FROM Enseignant
UNION ALL
SELECT * FROM Eutudiant
```

# Produit cartésien

Soient $S = (A_1, ..., A_n)$ et $S' = (B_1, ..., B_m)$, on définit $S \times S' = (A_1, ..., A_n, B_1, ..., B_m)$.
Soient $R$ de schéma $S$ et $R'$ de schéma $S'$, on définit

$$R \times R' = \{(e_1, ..., e_n, e'_1, ..., e'_m) \mid (e_1, ..., e_n) \in R, (e'_1, ..., e'_m) \in R'\}$$

$R \times R'$ est alors de schéma $S \oplus S'$.
On a $\mathbf{card}(R \times R') = \mathbf{card}R \times \mathbf{card}R'$.

# Produit cartésien : A retenir

- Le produit cartésien est un opérateur binaire qui produit une relation T

# Produit cartésien : A retenir

- Le produit cartésien est un opérateur binaire qui produit une relation T
- à partir de deux relations R et S,

# Produit cartésien : A retenir

- Le produit cartésien est un opérateur binaire qui produit une relation T
- à partir de deux relations R et S,
- donc le schéma est la concaténation de ceux de R et S,

# Produit cartésien : A retenir

- Le produit cartésien est un opérateur binaire qui produit une relation T
- à partir de deux relations R et S,
- donc le schéma est la concaténation de ceux de R et S,
- donc le résultat est l'ensemble de toutes les combinaisons possibles des tuples obtenus par concaténation de chaque tuple de R avec chaque tuple de S

# Produit cartésien : A retenir

- Le produit cartésien est un opérateur binaire qui produit une relation T
- à partir de deux relations R et S,
- donc le schéma est la concaténation de ceux de R et S,
- donc le résultat est l'ensemble de toutes les combinaisons possibles des tuples obtenus par concaténation de chaque tuple de R avec chaque tuple de S
- Notation

$$T = R \times S$$

# Produit cartésien : A retenir

- Le produit cartésien est un opérateur binaire qui produit une relation T
- à partir de deux relations R et S,
- donc le schéma est la concaténation de ceux de R et S,
- donc le résultat est l'ensemble de toutes les combinaisons possibles des tuples obtenus par concaténation de chaque tuple de R avec chaque tuple de S
- Notation

$$T = R \times S$$

- Exemple :

# Produit cartésien : Exemple 1

- Le produit cartésien $T = R \times S$ est une nouvelle relation où chaque tuple de R est associé à chaque tuple de S.

R :

| A | B |
|---|---|
| a | b |
| x | y |

S :

| C | D |
|---|---|
| c | d |
| u | v |
| x | y |

T :

| A | B | C | D |
|---|---|---|---|
| a | b | c | d |
| a | b | u | v |
| a | b | x | y |
| x | y | c | d |
| x | y | u | v |
| x | y | x | y |

# Produit cartésien : Exemple 2

- Soit $A$ l'ensemble { A, R, D, V, 10, 9, 8, 7, 6, 5, 4, 3, 2 }.
- Soit $B$ l'ensemble { pique, cœur, carreau, trèfle }.

# Produit cartésien : Exemple 2

- Soit $A$ l'ensemble { A, R, D, V, 10, 9, 8, 7, 6, 5, 4, 3, 2 }.
- Soit $B$ l'ensemble { pique, cœur, carreau, trèfle }.
- Alors le produit cartésien $A \times B$ de ces deux ensembles est un jeu classique de 52 cartes,
- c'est-à-dire l'ensemble :
  { (A, pique) ... (2, pique) , (A, cœur) ... (2, cœur) , (A, carreau) ... (2, carreau) , (A, trèfle) ... (2, trèfle) }.

# Produit cartésien : Exemple 2

| Numero |
|---|
| A |
| R |
| D |
| V |
| 10 |
| 9 |
| 8 |
| 7 |
| 6 |
| 5 |
| 4 |
| 3 |
| 2 |

| Type |
|---|
| pique |
| cœur |
| carreau |
| trèfle |

| Numero | Type |
|---|---|
| A | pique |
| ... | ... |
| 2 | pique |
| A | cœur |
| ... | ... |
| 2 | cœur |
| A | carreau |
| ... | ... |
| 2 | carreau |
| A | trèfle |
| ... | ... |
| 2 | trèfle |

Résultat : 52 cartes

# Produit cartésien : *RealisationSQL*

- `SELECT * FROM R,S`

# Produit cartésien : *RealisationSQL*

- `SELECT * FROM R,S`
- Exemple :

# Produit cartésien : *RealisationSQL*

- `SELECT * FROM R,S`
- Exemple :
- `SELECT * FROM NumCartes, CouleurCartes`

# Jointure

## Fusionner deux tables avec des valeurs d'attributs communes

$S, S'$ deux schémas , $A \in S, A' \in S'$ tels que $D(A) = D(A')$, $R(S)$ et $R'(S')$ :
on définit la *jointure* de $R$ et $R'$ selon $A$ et $A'$ par

$$R \underset{\text{A=A'}}{\bowtie} R' =$$

# Jointure

## Fusionner deux tables avec des valeurs d'attributs communes

$S, S'$ deux schémas , $A \in S, A' \in S'$ tels que $D(A) = D(A')$, $R(S)$ et $R'(S')$ :
on définit la *jointure* de $R$ et $R'$ selon $A$ et $A'$ par

$$R \underset{\text{A=A'}}{\bowtie} R' = R[A = A']R' \quad =$$

# Jointure

## Fusionner deux tables avec des valeurs d'attributs communes

$S, S'$ deux schémas , $A \in S, A' \in S'$ tels que $D(A) = D(A')$, $R(S)$ et $R'(S')$ :
on définit la *jointure* de $R$ et $R'$ selon $A$ et $A'$ par

$$R \underset{\text{A=A'}}{\bowtie} R' = R[A = A']R' = \{e \in R \times R' \mid e.A = e.A'\}$$
$$=$$

# Jointure

## Fusionner deux tables avec des valeurs d'attributs communes

$S, S'$ deux schémas , $A \in S, A' \in S'$ tels que $D(A) = D(A')$, $R(S)$ et $R'(S')$ :
on définit la *jointure* de $R$ et $R'$ selon $A$ et $A'$ par

$$R \underset{\text{A=A'}}{\bowtie} R' = R[A = A']R' = \{e \in R \times R' \mid e.A = e.A'\}$$
$$= \sigma_{A=A'}(R \times R')$$

# Jointure

## Fusionner deux tables avec des valeurs d'attributs communes

$S, S'$ deux schémas , $A \in S, A' \in S'$ tels que $D(A) = D(A')$, $R(S)$ et $R'(S')$ :
on définit la *jointure* de $R$ et $R'$ selon $A$ et $A'$ par

$$R \underset{\text{A=A'}}{\bowtie} R' = R[A = A']R' = \{e \in R \times R' \mid e.A = e.A'\}$$
$$= \sigma_{A=A'}(R \times R')$$

Remarque : si $S \cap S' \neq \emptyset$, on renommera auparavant les attributs de $S'$.

# Jointure

## Fusionner deux tables avec des valeurs d'attributs communes

$S, S'$ deux schémas , $A \in S, A' \in S'$ tels que $D(A) = D(A')$, $R(S)$ et $R'(S')$ :
on définit la *jointure* de $R$ et $R'$ selon $A$ et $A'$ par

$$R \underset{\text{A=A'}}{\bowtie} R' = R[A = A']R' = \{e \in R \times R' \mid e.A = e.A'\}$$
$$= \sigma_{A=A'}(R \times R')$$

Remarque : si $S \cap S' \neq \emptyset$, on renommera auparavant les attributs de $S'$.
Opération de renommage : $R' = \alpha[nomattribut : nouveaunom](R)$.

# Jointure : exemple 1

- Client=

| code | nom | téléphone |
|---|---|---|
| 1 | Wael | 20236521 |
| 2 | Firas | 52126520 |
| 3 | Loujaen | 94365874 |
| 4 | Manel | 25814569 |

Frs=

| gérant | société |
|---|---|
| Wael | la Force |
| Firas | Jadida |
| Ramzi | Tornado |
| Tahar | YZR-M1 |

# Jointure : exemple 1

- Client=

| code | nom | téléphone |
|---|---|---|
| 1 | Wael | 20236521 |
| 2 | Firas | 52126520 |
| 3 | Loujaen | 94365874 |
| 4 | Manel | 25814569 |

Frs=

| gérant | société |
|---|---|
| Wael | la Force |
| Firas | Jadida |
| Ramzi | Tornado |
| Tahar | YZR-M1 |

- Chercher les téléphones des clients qui sont aussi parmi nos fournisseurs ?

# Jointure : exemple 1

- Client=

| code | nom | téléphone |
|---|---|---|
| 1 | Wael | 20236521 |
| 2 | Firas | 52126520 |
| 3 | Loujaen | 94365874 |
| 4 | Manel | 25814569 |

Frs=

| gérant | société |
|---|---|
| Wael | la Force |
| Firas | Jadida |
| Ramzi | Tornado |
| Tahar | YZR-M1 |

- Chercher les téléphones des clients qui sont aussi parmi nos fournisseurs ?
- Client[nom = gérant]Frs

# Jointure : exemple 1

- Client=

| code | nom | téléphone |
|---|---|---|
| 1 | Wael | 20236521 |
| 2 | Firas | 52126520 |
| 3 | Loujaen | 94365874 |
| 4 | Manel | 25814569 |

Frs=

| gérant | société |
|---|---|
| Wael | la Force |
| Firas | Jadida |
| Ramzi | Tornado |
| Tahar | YZR-M1 |

- Chercher les téléphones des clients qui sont aussi parmi nos fournisseurs ?
- Client[nom = gérant]Frs =

| code | nom | téléphone | gérant | société |
|---|---|---|---|---|
| 1 | Wael | 20236521 | Wael | la Force |
| 2 | Firas | 52126520 | Firas | Jadida |

- $\pi_{\{nom,tél,société\}}(\text{Client}[\text{nom} = \text{gérant}]\text{Frs})$

# Jointure : exemple 1

- Client=

| code | nom | téléphone |
|---|---|---|
| 1 | Wael | 20236521 |
| 2 | Firas | 52126520 |
| 3 | Loujaen | 94365874 |
| 4 | Manel | 25814569 |

Frs=

| gérant | société |
|---|---|
| Wael | la Force |
| Firas | Jadida |
| Ramzi | Tornado |
| Tahar | YZR-M1 |

- Chercher les téléphones des clients qui sont aussi parmi nos fournisseurs ?
- Client[nom = gérant]Frs =

| code | nom | téléphone | gérant | société |
|---|---|---|---|---|
| 1 | Wael | 20236521 | Wael | la Force |
| 2 | Firas | 52126520 | Firas | Jadida |

- $\pi_{\{nom,tél,société\}}(\text{Client}[\text{nom} = \text{gérant}]\text{Frs})$=

| nom | tél | société |
|---|---|---|
| Wael | 20236521 | la Force |
| Firas | 52126520 | Jadida |

# Jointure : A retenir

- Permet d'établir un lien sémantique entre les relations

# Jointure : A retenir

- Permet d'établir un lien sémantique entre les relations
- But : créer toutes les combinaisons significatives entre tuples de deux relations

# Jointure : A retenir

- Permet d'établir un lien sémantique entre les relations
- But : créer toutes les combinaisons significatives entre tuples de deux relations
- significatives = portent la même valeur pour les attributs de même domaine !

# Jointure : A retenir

- Permet d'établir un lien sémantique entre les relations
- But : créer toutes les combinaisons significatives entre tuples de deux relations
- significatives = portent la même valeur pour les attributs de même domaine !
- **précondition :** les deux relations ont au moins un attribut de même domaine

# Jointure *RealisationSQL*

- `SELECT * FROM R JOIN S ON P`

# Jointure *RealisationSQL*

- `SELECT * FROM R JOIN S ON P`
- `SELECT * FROM R JOIN S ON R.attribut = S.Attribut WHERE ...`

# Jointure *RealisationSQL*

- `SELECT * FROM R JOIN S ON P`
- `SELECT * FROM R JOIN S ON R.attribut = S.Attribut WHERE ...`
- Trop d'informations...On projette pour ne garder que quelques attributs

# Jointure *RealisationSQL*

- `SELECT * FROM R JOIN S ON P`
- `SELECT * FROM R JOIN S ON R.attribut = S.Attribut WHERE ...`
- Trop d'informations...On projette pour ne garder que quelques attributs
- Exemple SQL :
  `SELECT nom, téléphone, société FROM Client JOIN Frs ON Client.nom = Frs.gérant`

# Intersection

- L'intersection est un opérateur binaire qui produit une relation T
- à partir de deux relations R et S,
- dont le schéma est le schéma commun à R et S,
- et dont les tuples/enregistrements sont ceux qui sont à la fois dans R et à S.
- Élimination des doublons.
- Notation

$$T = R \cap S$$

- Commutatif : $[R1 \cap R2] = [R2 \cap R1]$
- Associatif : $[(R1 \cap R2) \cap R3] = [R2 \cap (R1 \cap R3)]$
- Exemple :

# Intersection : Exemple

- Soit deux relations $R1$ et $R2$ **de même schéma**
- $R1 \cap R2$ est la relation contenant les tuples appartenant à $R1$ et à $R2$
- R1=

| A1 | A2 | A3 |
|---|---|---|
| a1 | a2 | a3 |
| b1 | b2 | b3 |
| c1 | c2 | c3 |
| d1 | d2 | d3 |

R2=

| A1 | A2 | A3 |
|---|---|---|
| a1 | a2 | a3 |
| e1 | e2 | e3 |
| b1 | b2 | b3 |

# Intersection : Exemple

- Soit deux relations $R1$ et $R2$ **de même schéma**
- $R1 \cap R2$ est la relation contenant les tuples appartenant à $R1$ et à $R2$
- R1=

| A1 | A2 | A3 |
|---|---|---|
| a1 | a2 | a3 |
| b1 | b2 | b3 |
| c1 | c2 | c3 |
| d1 | d2 | d3 |

R2=

| A1 | A2 | A3 |
|---|---|---|
| a1 | a2 | a3 |
| e1 | e2 | e3 |
| b1 | b2 | b3 |

- Intersection $R1 \cap R2$ (Relation temporaire)

# Intersection : Exemple

- Soit deux relations $R1$ et $R2$ **de même schéma**
- $R1 \cap R2$ est la relation contenant les tuples appartenant à $R1$ et à $R2$
- R1=

| A1 | A2 | A3 |
|---|---|---|
| a1 | a2 | a3 |
| b1 | b2 | b3 |
| c1 | c2 | c3 |
| d1 | d2 | d3 |

R2=

| A1 | A2 | A3 |
|---|---|---|
| a1 | a2 | a3 |
| e1 | e2 | e3 |
| b1 | b2 | b3 |

- Intersection $R1 \cap R2$ (Relation temporaire)=

| A1 | A2 | A3 |
|---|---|---|
| a1 | a2 | a3 |
| b1 | b2 | b3 |

- On garde que les lignes identiques

# Intersection : *RealisationSQL*

- SELECT ...
  INTERSECT
  SELECT ...

# Intersection : *RealisationSQL*

- SELECT ...
  INTERSECT
  SELECT ...

- Exemple :
  SELECT * FROM R1
  INTERSECT
  SELECT * FROM R2

# Différence

- La différence est un opérateur binaire qui produit une relation T

# Différence

- La différence est un opérateur binaire qui produit une relation T
- à partir de deux relations R et S,

# Différence

- La différence est un opérateur binaire qui produit une relation T
- à partir de deux relations R et S,
- dont le schéma est le schéma commun à R et S,

# Différence

- La différence est un opérateur binaire qui produit une relation T
- à partir de deux relations R et S,
- dont le schéma est le schéma commun à R et S,
- et dont les tuples/enregistrements sont ceux qui appartiennent à R mais pas à S.

# Différence

- La différence est un opérateur binaire qui produit une relation T
- à partir de deux relations R et S,
- dont le schéma est le schéma commun à R et S,
- et dont les tuples/enregistrements sont ceux qui appartiennent à R mais pas à S.
- Notation

$$T = R - S$$

# Différence

- La différence est un opérateur binaire qui produit une relation T
- à partir de deux relations R et S,
- dont le schéma est le schéma commun à R et S,
- et dont les tuples/enregistrements sont ceux qui appartiennent à R mais pas à S.
- Notation

$$T = R - S$$

- Non commutatif : $[R1 - R2] \neq [R2 - R1]$

# Différence

- La différence est un opérateur binaire qui produit une relation T
- à partir de deux relations R et S,
- dont le schéma est le schéma commun à R et S,
- et dont les tuples/enregistrements sont ceux qui appartiennent à R mais pas à S.
- Notation

$$T = R - S$$

- Non commutatif : $[R1 - R2] \neq [R2 - R1]$
- Non associatif : $[(R1 - R2) - R3] \neq [R2 - (R1 - R3)]$

# Différence

- La différence est un opérateur binaire qui produit une relation T
- à partir de deux relations R et S,
- dont le schéma est le schéma commun à R et S,
- et dont les tuples/enregistrements sont ceux qui appartiennent à R mais pas à S.
- Notation

$$T = R - S$$

- Non commutatif : $[R1 - R2] \neq [R2 - R1]$
- Non associatif : $[(R1 - R2) - R3] \neq [R2 - (R1 - R3)]$
- Exemple :

# Différence : Exemple

- Soit deux relations $R1$ et $R2$ **de même schéma**
- $R1 - R2$ est la relation contenant les tuples de $R1$ n'appartenant pas à $R2$.
- R1=

| A1 | A2 | A3 |
|---|---|---|
| a1 | a2 | a3 |
| b1 | b2 | b3 |
| c1 | c2 | c3 |
| d1 | d2 | d3 |

R2=

| A1 | A2 | A3 |
|---|---|---|
| a1 | a2 | a3 |
| e1 | e2 | e3 |
| b1 | b2 | b3 |

# Différence : Exemple

- Soit deux relations $R1$ et $R2$ **de même schéma**
- $R1 - R2$ est la relation contenant les tuples de $R1$ n'appartenant pas à $R2$.
- R1=

| A1 | A2 | A3 |
|---|---|---|
| a1 | a2 | a3 |
| b1 | b2 | b3 |
| c1 | c2 | c3 |
| d1 | d2 | d3 |

R2=

| A1 | A2 | A3 |
|---|---|---|
| a1 | a2 | a3 |
| e1 | e2 | e3 |
| b1 | b2 | b3 |

- Différence $R1 - R2$ (Relation temporaire)

# Différence : Exemple

- Soit deux relations $R1$ et $R2$ **de même schéma**
- $R1 - R2$ est la relation contenant les tuples de $R1$ n'appartenant pas à $R2$.
- R1=

| A1 | A2 | A3 |
|---|---|---|
| a1 | a2 | a3 |
| b1 | b2 | b3 |
| c1 | c2 | c3 |
| d1 | d2 | d3 |

R2=

| A1 | A2 | A3 |
|---|---|---|
| a1 | a2 | a3 |
| e1 | e2 | e3 |
| b1 | b2 | b3 |

- Différence $R1 - R2$ (Relation temporaire)=

| A1 | A2 | A3 |
|---|---|---|
| c1 | c2 | c3 |
| d1 | d2 | d3 |

# Différence *RealisationSQL*

- SELECT ...
  EXCEPT
  SELECT ...
- La différence est supportée par d'autres SGBD parfois sous le nom MINUS. (à vous de tester...)

- Exemple :

```
SELECT * FROM R1
EXCEPT
SELECT * FROM R2
```

# Bilan : Opérateurs et contraintes

| **Opérateur** | **Notation** | **Schémas** |
|---|---|---|
| Renommage | $T = \alpha[A : A'](R)$ | $S(T) = S(R)$ |
| Restriction | $T = \sigma_P(R)$ | $S(T) = S(R)$ |
| Projection | $T = \pi_{Attributs}(R)$ | $S(T) \subseteq S(R)$ |
| Intersection | $T = R \cap S$ | $S(R) = S(S) = S(T)$ |
| Différence | $T = R - S$ | $S(R) = S(S) = S(T)$ |
| Union | $T = R \cup S$ | $S(R) = S(S) = S(T)$ |
| Produit cartésien | $T = R \times S$ | $S(T) = S(S) \cup S(R)$ |
| Jointure | $T = R \bowtie S$ | $S(T) = S(S) \cup S(R)$ |

# Equivalences

- $R \cap S = S \cap R$
- $R \cup S = S \cup R$
- $\sigma_{[P1]}(\sigma_{[P2]}(R)) = \sigma_{[P2]}(\sigma_{[P1]}(R)) = \sigma_{[P1 et P2]}(R) = \sigma_{[P1]}(R) \cap \sigma_{[P2]}(R)$
- $\pi_{[a]}(\sigma_{[P]}(R)) = \sigma_{[P]}(\pi_{[a]}(R))$

# Reste à voir

- la traduction de l'algèbre relationnelle vers SQL (Plus de détails)

# Reste à voir

- la traduction de l'algèbre relationnelle vers SQL (Plus de détails)
- comment on utilise SQL depuis un langage de programmation (par exemple Python) ;

# Reste à voir

- la traduction de l'algèbre relationnelle vers SQL (Plus de détails)
- comment on utilise SQL depuis un langage de programmation (par exemple Python) ;
- quelles architectures ont les systèmes utilisant des bases de données SQL ;

# Bibliographie


- https://fr.wikipedia.org/wiki/Base_de_donnees_relationnelle

# Bibliographie


- https://fr.wikipedia.org/wiki/Base_de_donnees_relationnelle

  **SQL :**

# Bibliographie


- https://fr.wikipedia.org/wiki/Base_de_donnees_relationnelle

  **SQL :**
- http://www.w3schools.com/sql/default.asp

# Bibliographie


- https://fr.wikipedia.org/wiki/Base_de_donnees_relationnelle

  **SQL :**
- http://www.w3schools.com/sql/default.asp
- http://sql.sh/cours