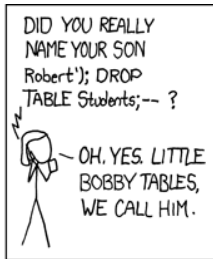
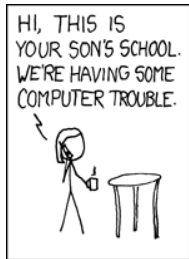


Bases de données 2: plusieurs tables

Informatique pour tous



Projection

Étant donné une table R , une **projection** de R revient à ne conserver que certains attributs:

R		
A	B	C
a_1	b_1	c_1
a_1	b_1	c_2
a_2	b_2	c_3

$\pi_{(A,B)}(R)$	
A	B
a_1	b_1
a_2	b_2

Projection

Étant donné une table R , une **projection** de R revient à ne conserver que certains attributs:

R		
A	B	C
a_1	b_1	c_1
a_1	b_1	c_2
a_2	b_2	c_3

$\pi_{(A,B)}(R)$	
A	B
a_1	b_1
a_2	b_2

En SQL: `SELECT DISTINCT A, B FROM R;`

Sélection

Étant donné une table R , une **sélection** $\sigma_{\text{condition}}(R)$ de R permet d'obtenir les enregistrements vérifiant une condition:

R		
A	B	C
a_1	b_1	c_1
a_1	b_1	c_2
a_2	b_2	c_3

$\sigma_{A=a_1}(R)$		
A	B	C
a_1	b_1	c_1
a_1	b_1	c_2

Étant donné une table R , une **sélection** $\sigma_{\text{condition}}(R)$ de R permet d'obtenir les enregistrements vérifiant une condition:

R		
A	B	C
a_1	b_1	c_1
a_1	b_1	c_2
a_2	b_2	c_3

$\sigma_{A=a_1}(R)$		
A	B	C
a_1	b_1	c_1
a_1	b_1	c_2

En SQL: `SELECT * FROM R WHERE A = a1;`

Exemple

Expliquer ce que donne l'expression suivante:

$$\pi_{nom, rayon}(\sigma_{etoile='Soleil'}(planete))$$

Dans la table: planete (nom, rayon, poids, etoile)

Union

Si R_1 et R_2 sont des tables ayant le **même schéma relationnel**, $R_1 \cup R_2$ contient les enregistrements dans R_1 ou R_2 :

R_1		
A	B	C
a_1	b_1	c_1
a_2	b_2	c_2

R_2		
A	B	C
a_1	b_1	c_1
a_3	b_3	c_3

$R_1 \cup R_2$		
A	B	C
a_1	b_1	c_1
a_2	b_2	c_2
a_3	b_3	c_3

En SQL: `SELECT * FROM R1 UNION SELECT * FROM R2;`

Exemple

Étant donné des tables `eleve_hugo`, `eleve_haag`, `eleve_pergaud`, il est possible d'obtenir tous les élèves de CPGE de Besançon:

```
SELECT * FROM eleve_hugo  
UNION SELECT * FROM eleve_haag  
UNION SELECT * FROM eleve_pergaud;
```


Différence

Si R_1 et R_2 sont des tables ayant le **même schéma relationnel**,
 $R_1 - R_2$ contient les enregistrements dans R_1 mais pas dans R_2 :

R_1		
A	B	C
a_1	b_1	c_1
a_2	b_2	c_2

R_2		
A	B	C
a_1	b_1	c_1
a_3	b_3	c_3

$R_1 - R_2$		
A	B	C
a_2	b_2	c_2

En SQL: `SELECT * FROM R1 MINUS SELECT * FROM R2;`

Intersection

Si R_1 et R_2 sont des tables ayant le **même schéma relationnel**, $R_1 \cap R_2$ contient les enregistrements à la fois dans R_1 et R_2 :

R_1		
A	B	C
a_1	b_1	c_1
a_2	b_2	c_2

R_2		
A	B	C
a_1	b_1	c_1
a_3	b_3	c_3

$R_1 \cap R_2$		
A	B	C
a_1	b_1	c_1

En SQL: `SELECT * FROM R1 INTERSECT SELECT * FROM R2;`

Produit cartésien

On peut réaliser le **produit cartésien** $R_1 \times R_2$ de deux tables:

R_1		
A	B	C
a_1	b_1	c_1
a_2	b_2	c_2

R_2	
D	E
d_1	e_1
d_2	e_2

$R_1 \times R_2$				
A	B	C	D	E
a_1	b_1	c_1	d_1	e_1
a_1	b_1	c_1	d_2	e_2
a_2	b_2	c_2	d_1	e_1
a_2	b_2	c_2	d_2	e_2

En SQL: `SELECT * FROM R1, R2;`

Produit cartésien

On peut réaliser le **produit cartésien** $R_1 \times R_2$ de deux tables:

R_1		
A	B	C
a_1	b_1	c_1
a_2	b_2	c_2

R_2	
D	E
d_1	e_1
d_2	e_2

$R_1 \times R_2$				
A	B	C	D	E
a_1	b_1	c_1	d_1	e_1
a_1	b_1	c_1	d_2	e_2
a_2	b_2	c_2	d_1	e_1
a_2	b_2	c_2	d_2	e_2

En SQL: `SELECT * FROM R1, R2;`

On peut aussi sélectionner seulement certaines colonnes de $R_1 \times R_2$ en écrivant, par exemple, `SELECT A, B FROM R1, R2;`

Exemple

Considérons une base de donnée bibliotheque avec les tables:

- ❶ livre (titre : CHAR(50), auteur : CHAR(50), pages : INT)
- ❷ emprunteur (id : INT, nom : CHAR(50))
- ❸ emprunt (id_emprunteur : INT, titre_livre: CHAR(50))

Exemple

Considérons une base de donnée bibliotheque avec les tables:

- ① livre (titre : CHAR(50), auteur : CHAR(50), pages : INT)
- ② emprunteur (id : INT, nom : CHAR(50))
- ③ emprunt (id_emprunteur : INT, titre_livre: CHAR(50))

Comment obtenir les emprunteurs qui sont aussi auteurs?

Exemple

Considérons une base de donnée bibliotheque avec les tables:

- ① livre (titre : CHAR(50), auteur : CHAR(50), pages : INT)
- ② emprunteur (id : INT, nom : CHAR(50))
- ③ emprunt (id_emprunteur : INT, titre_livre: CHAR(50))

Comment obtenir les emprunteurs qui sont aussi auteurs?

```
SELECT nom FROM emprunteur, livre  
WHERE nom = auteur;
```

Exemple

- ❶ livre (titre : CHAR(50), auteur : CHAR(50), pages : INT)
- ❷ emprunteur (id : INT, nom : CHAR(50))
- ❸ emprunt (id_emprunteur : INT, titre_livre: CHAR(50))

Comment obtenir les noms des personnes qui ont emprunté le livre dont le titre est Le Banquet?

Exemple

- ❶ livre (titre : CHAR(50), auteur : CHAR(50), pages : INT)
- ❷ emprunteur (id : INT, nom : CHAR(50))
- ❸ emprunt (id_emprunteur : INT, titre_livre: CHAR(50))

Comment obtenir les noms des personnes qui ont emprunté le livre dont le titre est Le Banquet?

```
SELECT nom FROM emprunteur, emprunt  
WHERE id = id_emprunteur  
AND titre_livre = 'Le Banquet';
```

JOIN

La jointure $R_1 \bowtie_{A=D} R_2$ de deux tables R_1 et R_2 revient à combiner les enregistrements de R_1 et R_2 en identifiant les colonnes A et D :

JOIN

La jointure $R_1 \bowtie_{A=D} R_2$ de deux tables R_1 et R_2 revient à combiner les enregistrements de R_1 et R_2 en identifiant les colonnes A et D :

R_1		
A	B	C
a_1	b_1	c_1
a_2	b_2	c_2
a_3	b_3	c_3

R_2	
D	E
a_1	e_1
a_2	e_2

$R_1 \bowtie_{A=D} R_2$			
A	B	C	E
a_1	b_1	c_1	e_1
a_2	b_2	c_2	e_2

En SQL: `SELECT ... FROM R1 JOIN R2 ON A = D;`

Exemple

- ❶ livre (titre : CHAR(50), auteur : CHAR(50), pages : INT)
- ❷ emprunteur (id : INT, nom : CHAR(50))
- ❸ emprunt (id_emprunteur : INT, titre_livre: CHAR(50))

Comment obtenir les noms des personnes qui ont emprunté le livre Le Banquet?

Exemple

- ❶ livre (titre : CHAR(50), auteur : CHAR(50), pages : INT)
- ❷ emprunteur (id : INT, nom : CHAR(50))
- ❸ emprunt (id_emprunteur : INT, titre_livre: CHAR(50))

Comment obtenir les noms des personnes qui ont emprunté le livre Le Banquet?

On peut aussi utiliser une jointure:

```
SELECT nom FROM emprunteur  
JOIN emprunt ON id = id_emprunteur  
WHERE titre_livre = 'Le Banquet';
```

Exemple

- ❶ livre (titre : CHAR(50), auteur : CHAR(50), pages : INT)
- ❷ emprunteur (id : INT, nom : CHAR(50))
- ❸ emprunt (id_emprunteur : INT, titre_livre: CHAR(50))

Comment obtenir les titres des livres empruntés par M. Machin?

Exemple

- ❶ livre (titre : CHAR(50), auteur : CHAR(50), pages : INT)
- ❷ emprunteur (id : INT, nom : CHAR(50))
- ❸ emprunt (id_emprunteur : INT, titre_livre: CHAR(50))

Comment obtenir les titres des livres empruntés par M. Machin?

```
SELECT titre_livre FROM emprunteur  
JOIN emprunt ON id = id_emprunteur  
WHERE nom = 'Machin';
```

Exemple

- ❶ livre (titre : CHAR(50), auteur : CHAR(50), pages : INT)
- ❷ emprunteur (id : INT, nom : CHAR(50))
- ❸ emprunt (id_emprunteur : INT, titre_livre: CHAR(50))

Comment obtenir les noms des personnes ayant emprunté un livre écrit par Stephen King?

Exemple

- ❶ livre (titre : CHAR(50), auteur : CHAR(50), pages : INT)
- ❷ emprunteur (id : INT, nom : CHAR(50))
- ❸ emprunt (id_emprunteur : INT, titre_livre: CHAR(50))

Comment obtenir les noms des personnes ayant emprunté un livre écrit par Stephen King?

```
SELECT nom FROM emprunteur  
JOIN emprunt ON id = id_emprunteur  
JOIN livre ON titre_livre = titre  
WHERE auteur = 'Stephen King';
```

Exemple

- ❶ livre (titre : CHAR(50), auteur : CHAR(50), pages : INT)
- ❷ emprunteur (id : INT, nom : CHAR(50))
- ❸ emprunt (id_emprunteur : INT, titre_livre: CHAR(50))

Comment obtenir les plus gros livres empruntés avec leur nombre de pages?

Exemple

- ❶ livre (titre : CHAR(50), auteur : CHAR(50), pages : INT)
- ❷ emprunteur (id : INT, nom : CHAR(50))
- ❸ emprunt (id_emprunteur : INT, titre_livre: CHAR(50))

Comment obtenir les plus gros livres empruntés avec leur nombre de pages?

```
SELECT titre, pages FROM livre  
JOIN emprunt ON titre_livre = titre  
ORDER BY pages DESC;
```

- ❶ livre (id : INT, titre : CHAR(50), auteur : CHAR(50), pages : INT)
- ❷ emprunteur (id : INT, nom : CHAR(50))
- ❸ emprunt (id_emprunteur : INT, titre_livre : CHAR(50))

Problème: comment savoir, dans livre \times emprunteur, à quelle table **id** fait référence?

Ambiguïté

- ❶ livre (id INT, titre CHAR(50), auteur CHAR(50), pages INT)
- ❷ emprunteur (id INT, nom CHAR(50))
- ❸ emprunt (id_emprunteur INT, titre_livre CHAR(50))

```
SELECT id FROM livre, emprunteur;
```

Résultat:

ERROR 1052 (23000): Column 'id' in field list is ambiguous

- ❶ livre (id INT, titre CHAR(50), auteur CHAR(50), pages INT)
- ❷ emprunteur (id INT, nom CHAR(50))
- ❸ emprunt (id_emprunteur INT, titre_livre CHAR(50))

Solution:

```
SELECT livre.id FROM livre, emprunteur;
```

- ❶ livre (id INT, titre CHAR(50), auteur CHAR(50), pages INT)
- ❷ emprunteur (id INT, nom CHAR(50))
- ❸ emprunt (id_emprunteur INT, titre_livre CHAR(50))

Problème 2: afficher tous les couples de livres ayant le même nombre de pages.

- ❶ livre (id INT, titre CHAR(50), auteur CHAR(50), pages INT)
- ❷ emprunteur (id INT, nom CHAR(50))
- ❸ emprunt (id_emprunteur INT, titre_livre CHAR(50))

Problème 2: afficher tous les couples de livres ayant le même nombre de pages.

```
SELECT titre, titre FROM livre, livre WHERE pages = pages;  
Ne marche pas du tout!
```


- ❶ livre (id INT, titre CHAR(50), auteur CHAR(50), pages INT)
- ❷ emprunteur (id INT, nom CHAR(50))
- ❸ emprunt (id_emprunteur INT, titre_livre CHAR(50))

Solution: renommer les tables (temporairement).

- ❶ livre (id INT, titre CHAR(50), auteur CHAR(50), pages INT)
- ❷ emprunteur (id INT, nom CHAR(50))
- ❸ emprunt (id_emprunteur INT, titre_livre CHAR(50))

Solution: renommer les tables (temporairement).

```
SELECT liv1.titre, liv2.titre  
FROM livre AS liv1, livre AS liv2  
WHERE liv1.pages = liv2.pages;
```