

**ENSTA, Paris**  
**BASES DE DONNÉES Relationnelles**  
P. Rigaux, M. Scholl et D. Gross-Amblard

Slide 1

(rigaux|scholl|dgram)@cnam.fr

2002/2003

**Slide 2**

## **INTRODUCTION**

### **Objectif**

**Slide 3** **OBJECTIF:**

Comprendre et Maitriser la technologie relationnelle

## BIBLIOGRAPHIE

### Ouvrages en français

1. P. Rigaux, *Cours bases de données*, cedric/cnam.fr/vertigo voir à support de cours.
2. Date C.J, *Introduction aux Bases de Données*, Vuibert, 970 Pages, Janvier 2001

### Ouvrages en anglais

#### Slide 4

1. R. Ramakrishnan et J. Gehrke, *DATABASE MANAGEMENT SYSTEMS*, MacGraw Hill
2. R. Elmasri, S.B. Navathe, *Fundamentals of database systems*, 3e édition, 1007 pages, 2000, Addison Wesley
3. Ullman J.D. and Widom J. *A First Course in Database Systems*, Prentice Hall, 1997
4. Garcia-Molina H., Ullman J. and Widom J., *Implementation of*

*Database Systems*, Prentice Hall, 1999

5. Ullman J.D., *Principles of Database and Knowledge-Base Systems*, 2 volumes, Computer Science Press
6. Abiteboul S., Hull R., Vianu V., *Foundations of Databases*, Addison-Wesley

#### Slide 5

### Le standard SQL

1. Date C.J., *A Guide to the SQL Standard*, Addison-Wesley

### Trois Systèmes

1. Date C.J., *A Guide to DB2*, Addison-Wesley
2. Date C.J., *A Guide to Ingres*, Addison-Wesley
3. *ORACLE version 7 Server Concepts Manual 1992 Oracle*

## Plan

Plan général du cours

1. Modèle et langages relationnels
2. Aspects pratiques des SGB relationnels
3. Aspects systèmes des modèles relationnels
4. TP

**Slide 6** Plan des trois premiers cours:

1. Introduction
2. Modèle relationnel
3. Algèbre Relationnelle
4. Langage de requête SQL
5. Techniques de stockage, organisation de fichiers
6. Indexation des fichiers

## Exemples d'Applications

### 1. CLASSIQUES

- Gestion (salaires, stocks, ...)
- Transactionnel (comptes, centrales d'achat, ...)
- Réservations (avions, trains, ...)

### 2. PLUS RECENTES

**Slide 7**

- Librairie et commerce électroniques (bibliothèques, journaux, web, ...)
- Documentation technique (nomenclature, plans, dessins, ...)
- Multimédia (textes, images, son, vidéo, ...)
- Géographique (cartes routières, thématiques, ...)
- Génie Logiciel (programmes, manuels, ...)

## Comment Stocker et Manipuler les Données?

### DONNÉES → BASE DE DONNÉES (B.D.)

- Une B.D. est un *GROS ENSEMBLE* d'informations *STRUCTURÉES* mémorisées sur un support *PERMANENT*.

Slide 8

### LOGICIEL → SGBD

- Un Système de Gestion de Bases de Données (SGBD) est un logiciel de *HAUT NIVEAU* qui permet de manipuler ces informations.

## Diversité -> Complexité

### Diversité des utilisateurs, des interfaces et des Architectures:

1. diversité des utilisateurs: administrateurs, programmeurs, non informaticiens, ...
2. diversité des interfaces: utilisateur final, langages BD, menus, saisies, rapports, ...
3. diversité des architectures : centralisé, distribué, accès à plusieurs bases hétérogènes accessibles par réseau, pair à pair

Slide 9

## **FONCTIONNALITÉS d'un SGBD**

Chaque niveau du SGBD réalise un certain nombre de fonctions :

### **NIVEAU PHYSIQUE**

**Slide 10**

- Accès aux données, gestion sur mémoire secondaire (fichiers) des données, des index
- Partage de données et gestion de la concurrence d'accès
- Reprise sur pannes (fiabilité)
- Distribution des données et interopérabilité (accès aux réseaux)

### **NIVEAU LOGIQUE**

**Slide 11**

- Définition de la structure de données : Langage de Description de Données (LDD)
- Consultation et Mise à Jour des données : Langages de Requêtes (LR) et Langage de Manipulation de Données (LMD)

### **Fonctionnalités du SGBD au NIVEAU EXTERNE**

**Slide 12**

- Gestion des Vues
- Environnement de programmation (intégration avec un langage de programmation)
- Interfaces conviviales et Langages de 4e Génération (L4G)
- Outils d'aides (e.g. conception de schémas)
- Outils de saisie, d'impression d'états
- Débogueurs
- Passerelles (réseaux, autres SGBD,etc...)

### **En Résumé, on Veut Gérer**

#### **un GROS VOLUME D'INFORMATIONS**

**Slide 13**

- Persistantes (années) et fiables (protection sur pannes)
- Partageables (utilisateurs, programmes)
- Manipulées indépendamment de leur organisation physique

Slide 14

### **Définition du schéma de données**

### **Modèles de données**

Slide 15

**Un modèle de données est caractérisé par :**

- Une structuration des objets
- Des opérations sur ces objets

**Dans un SGBD, il existe plusieurs modèles plus ou moins abstraits des mêmes objets, e.g. :**

**Slide 16**

- le modèle conceptuel : la description du système d'information
- le modèle logique : interface avec le SGBD
- le modèle physique : fichiers

⇒ ces différents modèles correspondent aux niveaux dans l'architecture d'un SGBD.

### **Modèle Conceptuel: Exemple Entité-Association**

**Slide 17**

- Modèle très abstrait, pratique pour :
  - l'analyse du monde réel
  - la conception du système d'information
  - la communication entre différents acteurs de l'entreprise
- **Mais n'est pas associé à un langage.**

DONC UNE STRUCTURE MAIS PAS D'OPÉRATIONS
--

### **Modèle logique**

1. **Langage de définition de données (LDD)** pour décrire la structure.
2. **Langage de manipulation de données (LMD)** pour appliquer des opérations aux données.

#### **Slide 18**

Ces langages sont **abstrait**s :

1. Le LDD est indépendant de la représentation physique des données.
2. Le LMD est indépendant de l'implantation des opérations.

### **Les avantages de l'abstraction**

1. **Simplicité d'accès**  
Les structures et les langages sont plus simples, donc plus faciles pour l'utilisateur non expert.
2. **INDÉPENDANCE PHYSIQUE.**  
On peut modifier l'implantation physique sans modifier les programmes d'application
3. **INDÉPENDANCE LOGIQUE.**  
On peut modifier les programmes d'application sans toucher à l'implantation.

#### **Slide 19**

## HISTORIQUE DES SGBD

À chaque génération correspond un modèle logique

Les premiers étaient peu abstraits (navigationnels)

<b>Slide 20</b>	< 60	S.G.F. (e.g. COBOL)	
	mi-60	HIÉRARCHIQUE IMS (IBM)	navigationnel
		RÉSEAU (CODASYL)	navigationnel
	73-80	RELATIONNEL	déclaratif
	mi-80	RELATIONNEL	explosion sur micro
	Fin 80	ORIENTÉ-OBJET	déclaratif + navigationnel
	Fin 90	Objet-relationnel	nouvelles appli
2003	XML	documents	

**Slide 21**

**Opérations sur les données**

### Exemples de questions (requêtes) posées à la base

*Insérer un employé nommé Jean*

*Augmenter Jean de 10%*

*Détruire Jean*

**Slide 22**

*Chercher les employés cadres*

*Chercher les employés du département comptabilité*

*Salaires moyens des employés comptables, avec deux enfants,  
nés avant 1960 et travaillant à Paris*

Les requêtes sont émises avec un *langage de requêtes* (SQL2, OQL, SQL3, XQUERY, etc.).

### Le Traitement d'une Requête

- ANALYSE SYNTAXIQUE
- OPTIMISATION

**Slide 23**

Génération (par le SGBD) d'un programme optimisé à partir de la connaissance de la structure des données, de l'existence d'index, de statistiques sur les données.

- EXÉCUTION POUR OBTENIR LE RÉSULTAT.

NB : on doit tenir compte du fait que d'autres utilisateurs sont peut-être en train de modifier les données qu'on interroge !

### **Concurrence d'accès et reprise sur pannes**

Plusieurs utilisateurs doivent pouvoir accéder en même temps aux mêmes données. Le SGBD doit savoir :

**Slide 24**

- Gérer les conflits si les deux font des mises-à-jour sur les mêmes données.
- Offrir un mécanisme de retour en arrière si on décide d'annuler des modifications en cours.
- Restaurer la base et revenir à un état cohérent en cas de panne.

**Slide 25**

## **LE MODÈLE RELATIONNEL**

### **Présentation Générale**

Slide 26

### Exemple de Relation

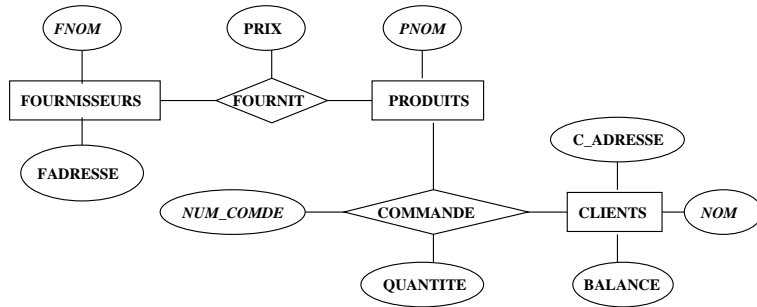
*Nom de la Relation*
*Nom d'Attribut*

Proprietaire	Type	Annee
Loic	Espace	1988
Nadia	Espace	1989
Loic	R5	1978
Julien	R25	1989
Marie	ZX	1993

*VEHICULE*
*n-uplet*

Slide 27

### un exemple de conception (schéma EA)



**Représentation par un ensemble de relations**

<b>FOURNISSEUR</b>	<b>FNOM</b>	<b>FADRESSE</b>
	Abounayan	92190 Meudon
	Cima	75010 Paris
	Preblocs	92230 Gennevilliers
	Sarnaco	75116 Paris

Slide 28

<b>FOURNITURE</b>	<b>FNOM</b>	<b>PNOM</b>	<b>PRIX</b>
	Abounayan	sable	300
	Abounayan	briques	1500
	Preblocs	parpaing	1200
	Sarnaco	parpaing	1150
	Sarnaco	ciment	125

<b>COMMANDES</b>	<b>NUM_COMDE</b>	<b>NOM</b>	<b>PNOM</b>	<b>QUANTITE</b>
	1	Jean	briques	5
	2	Jean	ciment	10
	3	Paul	briques	3
	4	Paul	parpaing	9
	5	Vincent	parpaing	7

Slide 29

<b>CLIENTS</b>	<b>NOM</b>	<b>CADRESSE</b>	<b>BALANCE</b>
	Jean	75006 Paris	-12000
	Paul	75003 Paris	0
	Vincent	94200 Ivry	3000
	Pierre	92400 Courbevoie	7000

### Retour sur le Modèle EA

1) **Entités**: un fournisseur est une entité

- Slide 30**
1. Type d'entités: tous les fournisseurs ont même type: *Fournisseurs*
  2. Attribut: représente une propriété (caractéristique) d'1 entité, exemple: FNOM
  3. concept: synonyme d'entité.

### Modèle EA

2) **Associations**: fournit est une association

- Slide 31**
1. relie deux (ou plus) entités, exemple: *fournit* relie Fournisseurs et Produits
  2. Attribut: Une association peut avoir des attributs propres
  3. role: synonyme d'association.
  4. *association 1,n* entre *E* et *F*: à 1 entité de type *E* on peut associer 1 ou plusieurs entités de type *F*: exemple rajoutons l'association *siège social* entre *Fournisseurs* et *Ville*
  5. *Association n,n*: à 1 entité *E* correspond 1 ou plusieurs entités *F* et réciproquement, exemple: *Fournit*.

### Clé

**Slide 32**

1. Attribut ou groupe d'attributs identifiant 1 entité: 2 entités de même type ne peuvent pas avoir la même valeur de clé, exemple *FNOM* pour Fournisseurs (hyp: 2 fournisseurs ne peuvent avoir le même nom)
2. *Clé primaire*: il peut y avoir plusieurs clés, exemple : si on rajoute le no de SS comme attribut à Fournisseurs, on a deux clés. Une est choisie comme clé primaire, e.g; NoSS

### Passage EA-Relationnel

A partir d'un schéma entité-association (niveau conceptuel) comment passer au niveau logique (comment choisir les relations)?

**Slide 33**

1. Une relation par type d'entité (e.g; Fournisseur). A une entité correspond un nuplet.
2. la clé primaire de la relation (attribut(s) de la relation qui identifie un nuplet) est celle du type d'entité
3. Une relation par association n,n: Les attributs sont les clés primaires des relations reliées par l'association, ainsi que les attributs propres de l'association
4. *Clé étrangère*: si l'association A entre R et S est 1,n, (n entités de type S pour une entité de type R), on ne crée pas de relation associée à cette association: on rajoute dans la table qui représente S la clé de R comme attribut. Celle-ci est appelée clé étrangère.
5. exercice: rajouter le type d'entité *VILLE* (clé: nom de ville) et l'association *siège social* entre *VILLE* et *FOURNISSEURS*. Comment est modifiée la relation *FOURNISSEURS*?

Slide 34

**Modèle relationnel: Définitions****Définitions**

- Un Domaine est un ensemble de valeurs. Exemples :  $\{0, 1\}$ ,  $N$ , l'ensemble des chaînes de caractères, l'ensemble des chaînes de caractères de longueur 10.
- Un ATTRIBUT prend ses valeurs dans un domaine. Plusieurs attributs peuvent avoir le même domaine.
- Un NUPLET est une liste de  $n$  valeurs  $(v_1, \dots, v_n)$  où chaque valeur  $v_i$  est la valeur d'un attribut  $A_i$  de domaine  $D_i : v_i \in D_i$
- Le PRODUIT CARTÉSIEN  $D_1 \times \dots \times D_n$  entre des domaines  $D_1, \dots, D_n$  est l'ensemble de **tous** les nuplets  $(v_1, \dots, v_n)$  où  $v_i \in D_i$ .

Slide 35

**Slide 36**

- RELATION : soit  $D_1, \dots, D_n$  les domaines respectifs des attributs  $A_1, \dots, A_n$ . Une relation  $R$  définie sur les attributs  $A_1, \dots, A_n$  est un sous-ensemble fini du produit cartésien  $D_1 \times \dots \times D_n$ :  $R$  est un ensemble de nuplets.
- Une relation  $R$  est représentée sous forme d'une **table**. L'ordre des colonnes ou des lignes n'a pas d'importance. Les colonnes sont distinguées par les noms d'attributs et chaque ligne représente un élément de l'ensemble  $R$  (un nuplet).
- Un attribut peut apparaître dans plusieurs relations.
- Une BASE DE DONNÉES est un ensemble de relations.

**Slide 37**

- L'UNIVERS D'ATTRIBUTS D'UNE BASE DE DONNÉES est l'ensemble de tous les attributs des relations de la base.
- Le SCHÉMA D'UNE RELATION  $R$  est défini par le nom de la relation et la liste des attributs avec pour chaque attribut son domaine. Notation :

$$R(A_1 : D_1, \dots, A_n : D_n)$$

ou plus simplement :

$$R(A_1, \dots, A_n)$$

Exemple :

VEHICULE(NOM:CHAR(20), TYPE:CHAR(10),  
ANNEE:ENTIER)

**Slide 38**

- Si la relation a  $n$  attributs ( $n$  colonnes),  $n$  est appelé ARITÉ de la relation. La relation *VEHICULE* est d'arité 3.
- Le SCHÉMA D'UNE BASE DE DONNÉES est l'ensemble des schémas de ses relations.

**Exemple de Base de Données**SCHÉMA :**Slide 39**

- FOURNISSEURS(FNOM:CHAR(20), FADRESSE:CHAR(30))
- FOURNITURE(FNOM:CHAR(20), PNOM:CHAR(10), PRIX:ENTIER))
- COMMANDES(NUM\_COMDE:ENTIER, NOM:CHAR(20),  
PNOM:CHAR(10), QUANTITE;ENTIER))
- CLIENTS(NOM: CHAR(20), CADRESSE:CHAR(30), BALANCE:RELATIF)

### Exemple de Base de Données

#### UNIVERS D'ATTRIBUTS :

- Slide 40**
- $U = \{FNOM, PNOM, NOM, FADRESSE, CADRESSE, PRIX, NUM\_CODE, QUANTITE, BALANCE\}$

#### RELATION UNIVERSELLE :

- $FPCC(FNOM, PNOM, NOM, FADRESSE, CADRESSE, PRIX, NUM\_CODE, QUANTITE, BALANCE)$

## Opérations sur une Base de Données Relationnelle

- Slide 42**
- LANGAGE DE DÉFINITION DES DONNÉES (définition et MAJ du schéma) :
    - Création et destruction d'une relation ou d'une base
    - Ajout, suppression d'un attribut

- LANGAGE DE MANIPULATION DES DONNÉES
  - Saisie des nuplets d'une relation
  - Affichage d'une relation
- **Slide 43**
  - Modification d'une relation : insertion, suppression et maj des nuplets
  - Requêtes : consultation d'une relation ou calcul d'une nouvelle relation
- GESTION DES TRANSACTIONS
- GESTION DES VUES

## Langages de Requêtes Relationnels

POUVOIR D'EXPRESSION : Qu'est-ce qu'on peut calculer ? Quelles opérations peut-on faire ?

**Slide 44** Les langages de requête relationnels utilisent deux approches :

- calcul relationnel
- algèbre relationnelle

Les deux approches ont **même pouvoir d'expression**.

**Slide 45**

**ALGÈBRE RELATIONNELLE**

## Algèbre Relationnelle

- Slide 46**
- une opération prend en entrée une ou deux relations
  - le résultat est toujours une relation

## Opérations de l'Algèbre Relationnelle

5 Opérations de base pour exprimer toutes les requêtes :

- Slide 47**
- Opérations unaires : sélection, projection
  - Opérations binaires : union, différence, produit cartésien
  - Autres opérations qui s'expriment en fonction des 5 opérations de base : jointure (naturelle,  $\theta$ -jointure), intersection, division

## Projection

LA PROJECTION “ÉLIMINE” UNE OU PLUSIEURS COLONNES D’UNE RELATION.

### Slide 48

Notation :

$$\pi_{A_1, A_2, \dots, A_k}(R)$$

## Projection: Exemples

a) On élimine la colonne  $C$  dans la relation  $R$  :

### Slide 49

R	A	B	C
→	a	b	c
	d	a	b
	c	b	d
→	a	b	e
	e	e	a

 $\Rightarrow$ 

$\pi_{A,B}(R)$		A	B
		a	b
		d	a
		c	b
		e	e

Le nuplet  $(a, b)$  n’apparaît qu’**une** fois dans la relation  $\pi_{A,B}(R)$ , bien qu’il existe **deux** nuplets  $(a, b, c)$  et  $(a, b, e)$  dans  $R$ .

**Projection: Exemples**

b) On élimine la colonne  $B$  dans la relation  $R$  (on garde  $A$  et  $C$ ) :

Slide 50

<b>R</b>	<b>A</b>	<b>B</b>	<b>C</b>
	a	b	c
	d	a	b
	c	b	d
	a	b	e
	e	e	a

 $\Rightarrow$ 

$\pi_{A,C}(R)$	<b>A</b>	<b>C</b>
	a	c
	d	b
	c	d
	a	e
	e	a

**Sélection**

Sélection sur la condition  $\mathcal{C}$  :

Slide 51

On garde les nuplets qui satisfont  $\mathcal{C}$ .

NOTATION :

$$\sigma_{\mathcal{C}}(R)$$

**Sélection: Exemples**

a) On sélectionne les nuplets dans la relation  $R$  tels que l'attribut  $B$  vaut "b" :

Slide 52

<b>R</b>	<b>A</b>	<b>B</b>	<b>C</b>
a	b	1	
d	a	2	
c	b	3	
a	b	4	
e	e	5	

 $\Rightarrow$ 

$\sigma_{B="b"}(R)$	<b>A</b>	<b>B</b>	<b>C</b>
a	b	1	
c	b	3	
a	b	4	

**Sélection: Exemples**

b) On sélectionne les nuplets tels que

$$(A = "a" \vee B = "a") \wedge C \leq 3 :$$

Slide 53

<b>R</b>	<b>A</b>	<b>B</b>	<b>C</b>
a	b	1	
d	a	2	
c	b	3	
a	b	4	
e	e	5	

 $\Rightarrow$ 

$\sigma_{(A="a" \vee B="a") \wedge C \leq 3}(R)$	<b>A</b>	<b>B</b>	<b>C</b>
a	b	1	
d	a	2	

### Sélection: Exemples

c) On sélectionne les nuplets tels que la 1re et la 2e colonne sont identiques :

Slide 54

<b>R</b>	<b>A</b>	<b>B</b>	<b>C</b>
	a	b	1
	d	a	2
	c	b	3
	a	b	4
	e	e	5

 $\Rightarrow \sigma_{A=B}(R)$ 

<b>A</b>	<b>B</b>	<b>C</b>
e	e	5

### Condition de Sélection

La condition  $\mathcal{C}$  d'une sélection peut être une **formule logique** quelconque avec des **et** ( $\wedge$ ) et des **ou** ( $\vee$ ) entre termes de la forme  $A_i \theta A_j$  et  $A_i \theta a$  où

Slide 55

- $A_i$  et  $A_j$  sont des attributs,
- $a$  est un élément (une valeur) du domaine de  $A_i$ ,
- $\theta$  est l'un de  $=, <, \leq, >, \geq, \neq$ .

### Expressions de l'Algèbre Relationnelle

Slide 56

- le résultat d'une opération est une **relation**
- sur cette relation, on peut faire une **autre opération** de l'algèbre

$\Rightarrow$  *Les opérations peuvent être composées pour former des expressions de l'algèbre relationnelle.*

### Expressions de l'Algèbre Relationnelle

EXEMPLE :  $COMMANDES(NOM, PNOM, NUM, QTE)$

Slide 57

$$R'' = \pi_{PNOM}(\overbrace{\sigma_{NOM='Jean'}}^{R'}(COMMANDES))$$

La relation  $R'(NOM, PNOM, NUM, QTE)$  contient les nuplets dont l'attribut  $NOM$  a la valeur "Jean". La relation  $R''(PNOM)$  contient tous les produits commandés par Jean.

### Produit Cartésien

- NOTATION :  $R \times S$
- ARGUMENTS : 2 relations quelconques :

$$R(A_1, A_2, \dots, A_n) \quad S(B_1, B_2, \dots, B_k)$$

#### Slide 58

- SCHÉMA DE  $T = R \times S$  :  $T(A_1, A_2, \dots, A_n, B_1, B_2, \dots, B_k)$
- VALEUR DE  $T = R \times S$  : ensemble de tous les nuplets ayant  $n + k$  composants (attributs)
  - dont les  $n$  premiers composants forment un nuplet de  $R$
  - et les  $k$  derniers composants forment un nuplet de  $S$

### Exemple de Produit Cartésien

#### Slide 59

<b>R</b>		A	B	
		1	1	
		1	2	
		3	4	

<b>S</b>		C	D	E	
		a	b	a	
		a	b	c	
		b	a	a	

$\Rightarrow$

Slide 60

$R \times S$	A	B	C	D	E
	1	1	a	b	a
	1	1	a	b	c
	1	1	b	a	a
	1	2	a	b	a
$ R  \times  S $	1	2	a	b	c
	1	2	b	a	a
	3	4	a	b	a
	3	4	a	b	c
	3	4	b	a	a

### Jointure Naturelle

- NOTATION :  $R \bowtie S$
- ARGUMENTS : 2 relations quelconques :

$$R(A_1, \dots, A_m, X_1, \dots, X_k) \quad S(B_1, \dots, B_n, X_1, \dots, X_k)$$

Slide 61

où  $X_1, \dots, X_k$  sont les attributs en commun.

- SCHÉMA DE  $T = R \bowtie S$  :  $T(A_1, \dots, A_m, B_1, \dots, B_n, X_1, \dots, X_k)$
- VALEUR DE  $T = R \bowtie S$  : ensemble de tous les nuplets ayant  $m + n + k$  attributs dont les  $m$  premiers et  $k$  derniers composants forment un nuplet de  $R$  et les  $n + k$  derniers composants forment un nuplet de  $S$ .

**Jointure Naturelle: Exemple**

Slide 62

R	A	B	C
	a	b	c
	d	b	c
	b	b	f
	c	a	d

S	B	C	D
	b	c	d
	b	c	e
	a	d	b

⇒

R ⋈ S	A	B	C	D
	a	b	c	d
	a	b	c	e
	d	b	c	d
	d	b	c	e
	c	a	d	b

**Jointure Naturelle**

Soit  $U = \{A_1, \dots, A_m, B_1, \dots, B_n, X_1, \dots, X_k\}$  l'ensemble des attributs des 2 relations et  $V = \{X_1, \dots, X_k\}$  l'ensemble des attributs en commun.

Slide 63

$$R \bowtie S = \pi_U(\sigma_{\forall A \in V: R.A=S.A}(R \times S))$$

NOTATION :  $R.A$  veut dire "l'attribut  $A$  de la relation  $R$ ".

**Jointure Naturelle: Exemple**

Slide 64

R	A	B
1	a	
1	b	
4	a	

S	A	B	D
1	a	b	
2	c	b	
4	a	a	

$$\Rightarrow$$

$R \times S$	R.A	R.B	S.A	S.B	D
	1	a	1	a	b
→	1	a	2	c	b
→	1	a	4	a	a
→	1	b	1	a	b
→	1	b	2	c	b
→	1	b	4	a	a
→	4	a	1	a	b
→	4	a	2	c	b
→	4	a	4	a	a

↓

Slide 65

$R \bowtie S$	A	B	D
	1	a	b
	4	a	a

$$\Leftarrow \pi_{R.A, R.B, D}(\sigma_{R.A=S.A \wedge R.B=S.B}(R \times S))$$

### Jointure Naturelle: Algorithme

Pour chaque nuplet  $a$  dans  $R$  et pour chaque nuplet  $b$  dans  $S$  :

1. on concatène  $a$  et  $b$  et on obtient un nuplet qui a pour attributs

$$\overbrace{A_1, \dots, A_m, X_1, \dots, X_k}^a, \overbrace{B_1, \dots, B_n, X_1, \dots, X_k}^b$$

Slide 66

2. on ne le garde que si chaque attribut  $X_i$  de  $a$  est égal à l'attribut  $X_i$  de  $b$  :

$$\forall_{i=1..k} a.X_i = b.X_i.$$

3. on élimine les valeurs (colonnes) dupliquées : on obtient un nuplet qui a pour attributs

$$\overbrace{A_1, \dots, A_m}^a, \overbrace{B_1, \dots, B_m}^b, \overbrace{X_1, \dots, X_k}^{a \text{ et } b}$$

### $\theta$ -Jointure

- ARGUMENTS : 2 relations quelconques :

$$R(A_1, \dots, A_m) \quad S(B_1, \dots, B_n)$$

Slide 67

- NOTATION :  $R \bowtie_{A_i \theta B_j} S, \theta \in \{=, \neq, <, \leq, >, \geq\}$
- SCHÉMA DE  $T = R \bowtie_{A_i \theta B_j} S : T(A_1, \dots, A_m, B_1, \dots, B_n)$
- VALEUR DE  $T = R \bowtie_{A_i \theta B_j} S : T = \sigma_{A_i \theta B_j} (R \times S)$
- ÉQUIJOINTURE :  $\theta$  est l'égalité.

**$\theta$ -Jointure: Exemple**

Slide 68

R	A	B
	1	a
	1	b
	3	a

S	C	D	E
	1	b	a
	2	b	c
	4	a	a

Slide 69

$T := R \times S$	A	B	C	D	E
	1	a	1	b	a
	1	a	2	b	c
	1	a	4	a	a
	1	b	1	b	a
	1	b	2	b	c
	1	b	4	a	a
	3	a	1	b	a
	3	a	2	b	c
	3	a	4	a	a

$A > C \rightarrow$   
 $A > C \rightarrow$

$\sigma_{A \leq C}(T)$	A	B	C	D	E
	1	a	1	b	a
	1	a	2	b	c
	1	a	4	a	a
	1	b	1	b	a
	1	b	2	b	c
	1	b	4	a	a
	3	a	4	a	a

$= R \bowtie_{A \leq C} S$

**Équijointure: Exemple**

Slide 70

**R**

A	B
1	a
1	b
3	a

**S**

C	D	E
1	b	a
2	b	c
4	a	a

Slide 71

**T := R × S**

A	B	C	D	E
1	<b>a</b>	1	<b>b</b>	a
1	<b>a</b>	2	<b>b</b>	c
1	a	4	a	a
1	b	1	b	a
1	b	2	b	c
1	<b>b</b>	4	<b>a</b>	a
3	<b>a</b>	1	<b>b</b>	a
3	<b>a</b>	2	<b>b</b>	c
3	a	4	a	a

$\sigma_{B=D}(T)$   
 $= R \bowtie_{B=D} S$

$\Rightarrow$

A	B	C	D	E
1	a	4	a	a
1	b	1	b	a
1	b	2	b	c
3	a	4	a	a

## Équijointure vs. Jointure Naturelle

$IMMEUBLE(ADI, NBETAGES, DATEC, PROP)$

$APPIM(ADI, NAP, OCCUP, ETAGE)$

Slide 72

1. Nom du propriétaire de l'immeuble où est situé l'appartement occupé par *Durand* :

$$\pi_{PROP}(\overbrace{IMMEUBLE \bowtie_{\sigma_{OCCUP='DURAND'}} APPIM}^{Jointure\ Naturelle})$$

2. Appartements occupés par des propriétaires d'immeuble :

$$\pi_{ADI, NAP, ETAGE}(\overbrace{APPIM \bowtie_{OCCUP=PROP} IMMEUBLE}^{équi\ jointure})$$

Autre Exemple de REQUÊTE : Nom et adresse des clients qui ont commandé des parpaings:

- Schéma Relationnel :

Slide 73

$COMMANDES(PNOM, CNOM, NUM_CMDE, QTE)$

$CLIENTS(CNOM, CADRESSE, BALANCE)$

- Requête Relationnelle :

$$\pi_{CNOM, CADRESSE}(CLIENTS \bowtie_{\sigma_{PNOM='PARPAING'}}(COMMANDES))$$

### Union

- ARGUMENTS : 2 relations de même schéma :

$$R(A_1, \dots, A_m) \quad S(A_1, \dots, A_m)$$

Slide 74

- NOTATION :  $R \cup S$
- SCHÉMA DE  $T = R \cup S$  :  $T(A_1, \dots, A_m)$
- VALEUR DE  $T$  : Union ensembliste sur  $D_1 \times \dots \times D_m$  :

$$T = \{t \mid t \in R \vee t \in S\}$$

### Union: Exemple

Slide 75

R	A	B
	a	b
	a	c
	d	e

S	A	B
	a	b
	a	e
	d	e
	f	g

$R \cup S$	A	B
→	a	b
	a	c
→	d	e
	a	e
	f	g

### Différence

- ARGUMENTS : 2 relations de même schéma :

$$R(A_1, \dots, A_m) \quad S(A_1, \dots, A_m)$$

Slide 76

- NOTATION :  $R - S$
- SCHÉMA DE  $T = R - S$  :  $T(A_1, \dots, A_m)$
- VALEUR DE  $T$  : Différence ensembliste sur  $D_1 \times \dots \times D_m$  :

$$T = \{t \mid t \in R \wedge t \notin S\}$$

### Différence: Exemple

<b>R</b>	<b>A</b>	<b>B</b>
→	a	b
	a	c
→	d	e

<b>S</b>	<b>A</b>	<b>B</b>
→	a	b
	a	e
→	d	e
	f	g

Slide 77

<b>R - S</b>	<b>A</b>	<b>B</b>
	a	c

<b>S - R</b>	<b>A</b>	<b>B</b>
	a	e
	f	g

### Intersection

- ARGUMENTS : 2 relations de même schéma :

$$R(A_1, \dots, A_m) \quad S(A_1, \dots, A_m)$$

Slide 78

- NOTATION :  $R \cap S$
- SCHÉMA DE  $T = R \cap S$  :  $T(A_1, \dots, A_m)$
- VALEUR DE  $T$  : Intersection ensembliste sur  $D_1 \times \dots \times D_m$  :

$$T = \{t \mid t \in R \wedge t \in S\}$$

### Intersection: Exemple

Slide 79

R	A	B
→	a	b
	a	c
→	d	e

S	A	B
→	a	b
	a	e
→	d	e
	f	g

R - S	A	B
	a	c

$R \cap S = R - (R - S)$	A	B
	a	b
	d	e

### Semijointure

- ARGUMENTS : 2 relations quelconques :

$$R(A_1, \dots, A_m, X_1, \dots, X_k) S(B_1, \dots, B_n, X_1, \dots, X_k)$$

Slide 80

où  $X_1, \dots, X_k$  sont les attributs en commun.

- NOTATION :  $R \bowtie S$
- SCHÉMA DE  $T = R \bowtie S : T(A_1, \dots, A_m, X_1, \dots, X_k)$
- VALEUR DE  $T = R \bowtie S$  : Projection sur les attributs de  $R$  de la jointure naturelle entre  $R$  et  $S$ .

### Semijointure

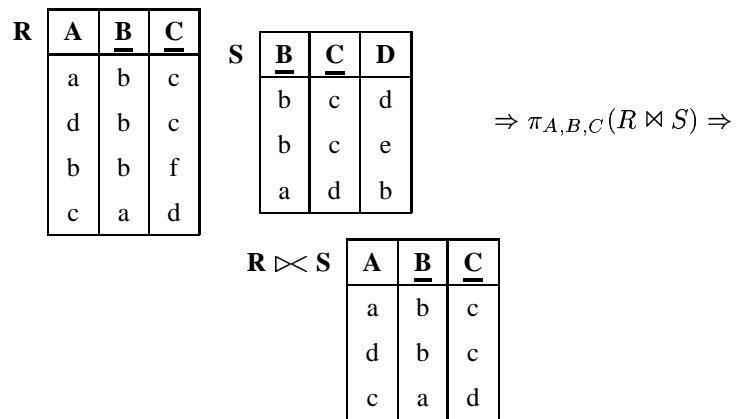
La semijointure correspond à une sélection où la condition de sélection est définie par le biais d'une autre relation.

Slide 81

Soit  $U = \{A_1, \dots, A_m\}$  l'ensemble des attributs de  $R$ .

$$R \bowtie S = \pi_U(R \bowtie S)$$

**Semijointure: Exemple**



Slide 82

**Division: Exemple**

REQUÊTE : Clients qui commandent tous les produits:

COMM	NUM	NOM	PNOM	QTE
	1	Jean	briques	100
	2	Jean	ciment	2
	3	Jean	parpaing	2
	4	Paul	briques	200
	5	Paul	parpaing	3
	6	Vincent	parpaing	3

Slide 83

Slide 84

$$R = \pi_{NOM,PNOM}(COMM) :$$

<b>R</b>	<b>NOM</b>	<b>PNOM</b>
	Jean	briques
	Jean	ciment
	Jean	parpaing
	Paul	briques
	Paul	parpaing
	Vincent	parpaing

<b>PROD</b>	<b>PNOM</b>
	briques
	ciment
	parpaing

↓

$$R \div PROD$$

<b>NOM</b>
Jean

**Division: Exemple**

Slide 85

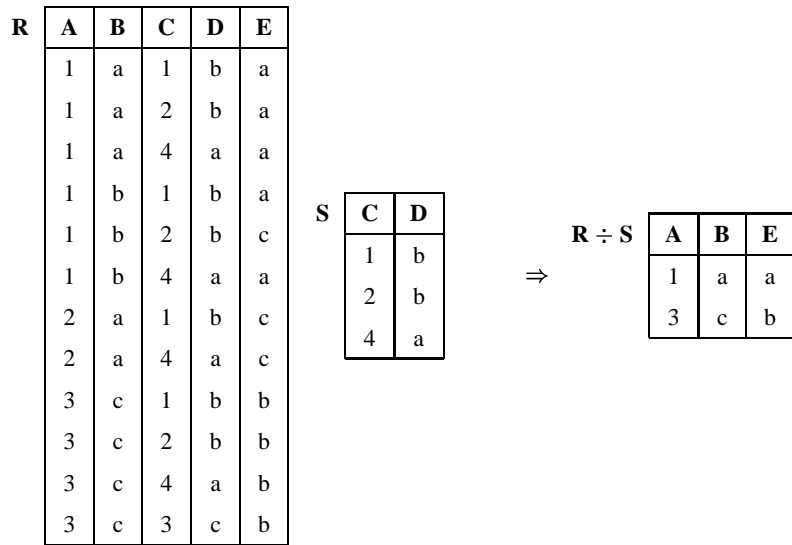
<b>R</b>	<b>A</b>	<b>B</b>	<b>C</b>	<b>D</b>
	a	b	x	m
	a	b	y	n
	a	b	z	o
	b	c	x	o
	b	d	x	m
	c	e	x	m
	c	e	y	n
	c	e	z	o
	d	a	z	p
	d	a	y	m

<b>S</b>	<b>C</b>	<b>D</b>
	x	m
	y	n
	z	o

<b>R \ S</b>	<b>A</b>	<b>B</b>
	a	b
	c	e

**Division: Exemple**

Slide 86



**Division**

- ARGUMENTS : 2 relations :

$$R(A_1, \dots, A_m, X_1, \dots, X_k) \quad S(X_1, \dots, X_k)$$

où **tous** les attributs de  $S$  sont des attributs de  $R$ .

Slide 87

- NOTATION :  $R \div S$
- SCHÉMA DE  $T = R \div S$  :  $T(A_1, \dots, A_m)$
- VALEUR DE  $T = R \div S$  :

$$R \div S = \{(a_1, \dots, a_m) \mid \forall (x_1, \dots, x_k) \in S : (a_1, \dots, a_m, x_1, \dots, x_k) \in R\}$$

### Division

**Slide 88** La division s'exprime en fonction du produit cartésien, de la projection et de la différence :  $R \div S = R_1 - R_2$  où

$$R_1 = \pi_{A_1, \dots, A_m}(R) \text{ et } R_2 = \pi_{A_1, \dots, A_m}((R_1 \times S) - R)$$

### Renommage

- NOTATION :  $\rho$
  - ARGUMENTS : 1 relation :  

$$R(A_1, \dots, A_n)$$
  - SCHÉMA DE  $T = \rho_{A_i \rightarrow B_i} R : T(A_1, \dots, A_{i-1}, B_i, A_{i+1}, \dots, A_n)$
  - VALEUR DE  $T = \rho_{A_i \rightarrow B_i} R : T = R$ . La valeur de R est inchangée. Seul le nom de l'attribut  $A_i$  a été remplacé par  $B_i$
- Slide 89**

### Et si les attributs sont connus par leur rang?

On peut dans les opérations de l'algèbre indiquer un attribut par son rang au lieu de son nom.

#### Slide 90

Dans ce cas l'ordre des attributs dans une relation a de l'importance.

L'exemple de la sélection :

$\sigma_{\&1='PARPAING'} COMMANDES$  au lieu de

$\sigma_{PNOM='PARPAING'} COMMANDES$

Le premier attribut  $\&1$  correspond a l'attribut *PNOM*

## Principe

- SQL (Structured Query Language) est le Langage de Requêtes standard pour les SGBD relationnels
- Expression d'une requête par un bloc *SELECT FROM WHERE*

### Slide 92

```
SELECT <liste des attributs a projeter>  
FROM   <liste des relations arguments>  
WHERE  <conditions sur un ou plusieurs attributs>
```

- Dans les requêtes simples, la correspondance avec l'algèbre relationnelle est facile à mettre en évidence.

## Historique\*

**SQL86 - SQL89 ou SQL1** La référence de base:

- Requêtes compilées puis exécutées depuis un programme d'application.
- Types de données simples (entiers, réels, chaînes de caractères de taille fixe)
- Opérations ensemblistes restreintes (UNION).

### Slide 93

**SQL92 ou SQL2** Standard actuel:

- Requêtes dynamiques: exécution différée ou immédiate
- Types de données plus riches (intervalles, dates, chaînes de caractères de taille variable)
- Différents types de jointures: jointure naturelle, jointure externe
- Opérations ensemblistes: différence (EXCEPT), intersection (INTERSECT)
- Renommage des attributs dans la clause SELECT

**SQL3 (en cours)** : SQL devient un langage de programmation :

**Slide 94**

- Extensions orientées-objet
- Opérateur de fermeture transitive (recursion)
- ...

**Slide 95**

**Expressions de Base**

### Projection

Soit le schéma de relation **COMMANDES**(NUM,CNOM,PNOM,QUANTITE)

REQUÊTE: *Information sur toutes les commandes*

SQL:

**Slide 96**

```
SELECT NUM , CNOM , PNOM , QUANTITE
FROM   COMMANDES
```

ou

```
SELECT *
FROM   COMMANDES
```

### Projection : Distinct

Soit le schéma de relation **COMMANDES**(NUM,CNOM,PNOM,QUANTITE)

REQUÊTE: *Produits commandés*

**Slide 97**

```
SELECT PNOM
FROM   COMMANDES
```

**NOTE:** Contrairement à l'algèbre relationnelle, SQL n'élimine pas les doublés. Pour les éliminer on utilise DISTINCT :

```
SELECT DISTINCT PNOM
FROM   COMMANDES
```

Le DISTINCT peut être remplacé par la clause UNIQUE dans certains systèmes

### Sélection

Soit le schéma de relation **COMMANDES**(NUM,CNOM,PNOM,QUANTITE)

REQUÊTE: *Produits commandés par Jean*

**Slide 98** ALGÈBRE:  $\pi_{PNOM}(\sigma_{CNOM='JEAN'}(COMMANDES))$

SQL:

```
SELECT PNOM
FROM   COMMANDES
WHERE  CNOM = 'JEAN'
```

REQUÊTE: *Produits commandés par Jean en quantité supérieure à 100*

SQL:

**Slide 99**

```
SELECT PNOM
FROM   COMMANDES
WHERE  CNOM = 'JEAN'
AND    QUANTITE > 100
```

### Conditions de sélection en SQL : Conditions simples

Les conditions de base sont exprimées de deux façons:

1. *attribut comparateur valeur*

**Slide 100** 2. *attribut comparateur attribut*

où *comparateur* est =, <, >, <>, ... ,

Soit le schéma de relation **FOURNITURE**(PNOM, FNOM, PRIX)

REQUÊTE: *Produits de prix supérieur à 200F*

SQL:

**Slide 101**

```
SELECT PNOM
FROM FOURNITURE
WHERE PRIX > 2000
```

Soit le schéma de relation **FOURNITURE**(PNOM, FNOM, PRIX)

REQUÊTE: *Produits dont le nom est celui du fournisseur*

**Slide 102** SQL:

```
SELECT PNOM
FROM   FOURNITURE
WHERE  PNOM = FNOM
```

### Conditions de sélection en SQL : Suite

Le *comparateur* est BETWEEN, LIKE, IN

Soit le schéma de relation **FOURNITURE**(PNOM, FNOM, PRIX)

REQUÊTE: *Produits avec un coût entre 1000F et 2000F*

**Slide 103** SQL:

```
SELECT PNOM
FROM   FOURNITURE
WHERE  PRIX BETWEEN 1000 AND 2000
```

**NOTE:** La condition  $y$  BETWEEN  $x$  AND  $z$  est équivalente à  $y \leq z$  AND  $x \leq y$ .

Soit le schéma de relation **COMMANDES**(NUM,CNOM,PNOM,QUANTITE)

REQUÊTE: *Clients dont le nom commence par "C"*

SQL:

**Slide 104**

```
SELECT CNOM
FROM   COMMANDES
WHERE  CNOM LIKE 'C%'
```

**NOTE:** Le littéral qui suit LIKE doit être une chaîne de caractères éventuellement avec des caractères jokers (\_, %). Pas exprimable avec l'algèbre relationnelle.

Soit le schéma de relation **FOURNITURE**(PNOM, FNOM, PRIX)

REQUÊTE: *Produits avec un coût de 100F, de 200F ou de 300F*

SQL:

**Slide 105**

```
SELECT PNOM
FROM   FOURNITURE
WHERE  PRIX IN {100, 200, 300}
```

**NOTE:** La condition  $x \text{ IN } \{a, b, \dots, z\}$  est équivalente à  $x = a \text{ OR } x = b \text{ OR } \dots \text{ OR } x = z$ .

### Jointure

Soit le schéma de relations

**COMMANDES**(NUM,CNOM,PNOM,QUANTITE)

**FOURNITURE**(PNOM, FNOM, PRIX)

**Slide 106**

REQUÊTE: *Nom, Coût, Fournisseur des Produits commandés par Jean*

ALGÈBRE :

$$\pi_{PNOM, PRIX, FNOM}(\sigma_{CNOM='JEAN'}(COMMANDES) \bowtie (FOURNITURE))$$

SQL :

```
SELECT COMMANDES.PNOM, PRIX, FNOM
FROM   COMMANDES, FOURNITURE
WHERE  CNOM = 'JEAN' AND
       COMMANDES.PNOM = FOURNITURE.PNOM
```

**Slide 107**

**NOTE:** Cette requête est équivalente à une jointure naturelle. Noter qu'il faut toujours expliciter les attributs de jointure.

**NOTE:** SELECT COMMANDES.PNOM, PRIX, FNOM FROM COMMANDES, FOURNITURE équivaut à un produit cartésien des deux relations, suivi d'une projection.

Soit le schéma de relation **FOURNISSEUR**(FNOM,STATUT,VILLE)

REQUÊTE: *Fournisseurs qui habitent deux à deux dans la même ville*

SQL:

```
SELECT PREM.FNOM, SECOND.FNOM
FROM   FOURNISSEUR PREM, FOURNISSEUR SECOND
WHERE  PREM.VILLE = SECOND.VILLE AND
       PREM.FNOM < SECOND.FNOM
```

### Slide 108

La deuxième condition permet

1. l'élimination des paires (x,x)
2. d'éviter d'obtenir au résultat à la fois (x,y) et (y,x)

**NOTE:** PREM représente une instance de FOURNISSEUR, SECOND une autre instance de FOURNISSEUR.

Soit le schéma de relation **EMPLOYE**(EMPNO,ENOM,DEPNO,SAL)

REQUÊTE: *Nom et Salaire des Employés gagnant plus que l'employé de numéro 12546*

ALGÈBRE:

$$R1 := \pi_{SAL}(\sigma_{EMPNO=12546}(EMPLOYEE))$$

$$R2 := \pi_{ENOM,EMPLOYEE.SAL}((EMPLOYEE) \bowtie_{EMPLOYEE.SAL > R1.SAL} (R1))$$

### Slide 109

SQL:

```
SELECT E1.ENOM, E1.SAL
FROM   EMPLOYEE E1, EMPLOYEE E2
WHERE  E2.EMPNO = 12546 AND
       E1.SAL > E2.SAL
```

Slide 110

**Expressions Ensemblistes****Union****COMMANDES**(NUM,CNOM,PNOM,QUANTITE)**FOURNITURE**(PNOM, FNOM, PRIX)REQUÊTE: *Produits qui coûtent plus que 1000F ou ceux qui sont commandés par Jean*

ALGÈBRE:

Slide 111

$$\begin{aligned} & \pi_{PNOM}(\sigma_{PRIX > 1000}(FOURNITURE)) \\ & \cup \\ & \pi_{PNOM}(\sigma_{CNOM='Jean'}(COMMANDES)) \end{aligned}$$

SQL:

Slide 112

```

SELECT PNOM
FROM FOURNITURE
WHERE PRIX >= 1000
UNION
SELECT PNOM
FROM COMMANDES
WHERE CNOM = 'Jean'

```

**NOTE:** L'union élimine les doublés. Pour garder les doublés on utilise l'opération UNION ALL : le résultat contient chaque n-uplet  $a + b$  fois, où  $a$  et  $b$  sont le nombre d'occurrences du n-uplet dans la première et la deuxième requête.

### Différence

La différence ne fait pas partie du standard.

**EMPLOYE**(EMPNO,ENOM,DEPTNO,SAL)

**DEPARTEMENT**(DEPTNO,DNOM,LOC)

REQUÊTE: *Départements sans employés*

ALGÈBRE:  $\pi_{DEPTNO}(DEPARTEMENT) - \pi_{DEPTNO}(EMPLOYE)$

Slide 113 SQL:

```

SELECT DEPTNO
FROM DEPARTEMENT
EXCEPT
SELECT DEPTNO
FROM EMPLOYE

```

**NOTE:** La différence élimine les doublés. Pour garder les doublés on utilise l'opération EXCEPT ALL :

### Intersection

L'intersection ne fait pas partie du standard.

**EMPLOYE**(EMPNO,ENOM,DEPTNO,SAL)

**DEPARTEMENT**(DEPTNO,DNOM,LOC)

REQUÊTE: *Départements ayant des employés qui gagnent plus que 20000F et qui se trouvent à Paris*

**Slide 114**

ALGÈBRE :

$$\pi_{DEPTNO}(\sigma_{LOC="Paris"}(DEPARTEMENT)) \cap \pi_{DEPTNO}(\sigma_{SAL>20000}(EMPLOYE))$$

SQL:

```
SELECT DEPTNO
FROM DEPARTEMENT
WHERE LOC = 'Paris'
INTERSECT
SELECT DEPTNO
FROM EMPLOYE
WHERE SAL > 20000
```

**Slide 115**

**NOTE:** L'intersection élimine les doublés. Pour garder les doublés on utilise l'opération INTERSECT ALL : le résultat contient chaque n-uplet  $\min(a, b)$  fois, où  $a$  et  $b$  sont le nombre d'occurrences du n-uplet dans la première et la deuxième requête.

Slide 116

**Imbrication des Requêtes en SQL****Requêtes imbriquées simples**

La Jointure s'exprime par deux blocs SFW imbriqués

Soit le schéma de relations

**COMMANDES**(NUM,CNOM,PNOM,QUANTITE)

Slide 117

**FOURNITURE**(PNOM, FNOM, PRIX)

REQUÊTE: *Nom, prix et fournisseurs des Produits commandés par Jean*

ALGÈBRE:

$\pi_{PNOM, PRIX, FNOM}(\sigma_{CNOM='JEAN'}(COMMANDES) \bowtie (FOURNITURE))$

SQL:

```
SELECT PNOM, PRIX, FNOM
FROM FOURNITURE
WHERE PNOM IN (SELECT PNOM
                FROM COMMANDES
                WHERE CNOM = 'JEAN')
```

**Slide 118**

ou

```
SELECT FOURNITURE.PNOM, PRIX, FNOM
FROM FOURNITURE, COMMANDES
WHERE FOURNITURE.PNOM = COMMANDES.PNOM
AND CNOM = 'JEAN'
```

La Différence s'exprime aussi par deux blocs SFW imbriqués

Soit le schéma de relations

**EMPLOYE**(EMPNO, ENOM, DEPNO, SAL)

**DEPARTEMENT**(DEPTNO, DNOM, LOC)

**Slide 119**

REQUÊTE: *Départements sans employés*

ALGÈBRE :

$$\pi_{DEPTNO}(DEPARTEMENT) - \pi_{DEPTNO}(EMPLOYE)$$

SQL:

```
SELECT DEPTNO
FROM DEPARTEMENT
WHERE DETPNO NOT IN (SELECT DISTINCT DEPTNO
                     FROM EMPLOYE)
```

**Slide 120** ou

```
SELECT DEPTNO
FROM DEPARTEMENT
EXCEPT
SELECT DISTINCT DEPTNO
FROM EMPLOYE
```

### Requêtes imbriquées plus complexes : ANY - ALL

Soit le schéma de relation **FOURNITURE**(PNOM,FNOM,PRIX)

REQUÊTE: *Fournisseurs des Briques à un coût inférieur au coût maximum des Ardoises*

**Slide 121**

```
SQL :   SELECT FNOM
        FROM   FOURNITURE
        WHERE  PNOM = 'Brique'
        AND    PRIX < ANY (SELECT PRIX
                          FROM   FOURNITURE
                          WHERE  PNOM = 'Ardoise')
```

**NOTE:** La condition  $f \theta ANY (SELECT F FROM \dots)$  est vraie ssi la comparaison  $f \theta v$  est vraie au moins pour une valeur  $v$  du résultat du bloc  $(SELECT F FROM \dots)$ .

Soit le schéma de relations

**COMMANDE**(NUM,CNOM,PNOM,QUANTITE)

**FOURNITURE**(PNOM, FNOM, PRIX)

REQUÊTE: *Nom, Coût et Fournisseur des Produits commandés par Jean*

SQL:

**Slide 122**

```
SELECT PNOM, PRIX, FNOM
FROM   FOURNITURE
WHERE  PNOM = ANY (SELECT PNOM
                   FROM   COMMANDE
                   WHERE  CNOM = 'JEAN')
```

**NOTE:** Les prédicats IN et = ANY sont utilisés de façon équivalente.

Soit le schéma de relation **COMMANDE**(NUM,CNOM,PNOM,QUANTITE)

REQUÊTE: *Client ayant commandé la plus petite quantité de Briques*

SQL:

**Slide 123**

```
SELECT CNOM
FROM   COMMANDE
WHERE  PNOM = 'Brique' AND
       QUANTITE <= ALL (SELECT QUANTITE
                       FROM   COMMANDE
                       WHERE  PNOM = 'Brique')
```

**NOTE:** La condition  $f \theta \text{ ALL } (\text{SELECT } F \text{ FROM } \dots)$  est vraie ssi la comparaison  $f \theta v$  est vraie pour toutes les valeurs  $v$  du résultat du bloc  $(\text{SELECT } F \text{ FROM } \dots)$ .

Soit le schéma de relations

**EMPLOYE**(EMPNO,ENOM,DEPNO,SAL)

**DEPARTEMENT**(DEPTNO,DNOM,LOC)

REQUÊTE: *Départements sans employés*

**Slide 124**

SQL:

```
SELECT DEPTNO
FROM DEPARTEMENT
WHERE DEPTNO NOT = ALL ( SELECT DISTINCT DEPTNO
                        FROM EMPLOYE )
```

**NOTE:** Les prédicats NOT IN et NOT = ALL sont utilisés de façon équivalente.

### Requêtes imbriquées plus complexes : EXISTS

Soit le schéma de relations

**FOURNISSEUR**(FNOM,STATUS,VILLE)

**FOURNITURE**(PNOM,FNOM,PRIX)

REQUÊTE: *Fournisseurs qui fournissent au moins un produit*

**Slide 125**

```
SQL : SELECT FNOM
      FROM FOURNISSEUR
      WHERE EXISTS ( SELECT *
                    FROM FOURNITURE
                    WHERE FNOM = FOURNISSEUR.FNOM )
```

**NOTE:** La condition EXISTS (SELECT \* FROM ...) est vraie ssi le résultat du bloc (SELECT F FROM ...) n'est pas vide.

Soit le schéma de relations

**FOURNISSEUR**(FNOM,STATUS,VILLE)

**FOURNITURE**(PNOM,FNOM,PRIX)

REQUÊTE: *Fournisseurs qui ne fournissent aucun produit*

SQL:

**Slide 126**

```
SELECT FNOM
FROM   FOURNISSEUR
WHERE  NOT EXISTS (SELECT *
                   FROM   FOURNITURE
                   WHERE  FNOM = FOURNISSEUR.FNOM)
```

**NOTE:** La condition NOT EXISTS (SELECT \* FROM ...) est vraie ssi le résultat du bloc (SELECT F FROM ...) est vide.

### Formes Équivalentes de Quantification

Si  $\theta$  est un des opérateurs de comparaison  $<, =, >, \dots$

- La condition  $x \theta$  ANY (SELECT Ri.y FROM R1, ... Rn WHERE p) est équivalente à

**Slide 127**

```
EXISTS (SELECT * FROM R1, ... Rn WHERE p AND x  $\theta$  Ri.y)
```

- La condition  $x \theta$  ALL (SELECT Ri.y FROM R1, ... Rn WHERE p) est équivalente à

```
NOT EXISTS (SELECT * FROM R1, ... Rn WHERE (p) AND NOT (x  $\theta$  Ri.y))
```

Soit le schéma de relations

**COMMANDE**(NUM,CNOM,PNOM,QUANTITE)

**FOURNITURE**(PNOM, FNOM, PRIX)

REQUÊTE: *Nom, prix et fournisseur des produits commandés par Jean*

**Slide 128** SELECT PNOM, PRIX, FNOM FROM FOURNITURE  
 WHERE EXISTS (SELECT \* FROM COMMANDE  
                   WHERE CNOM = 'JEAN'  
                   AND PNOM = FOURNITURE.PNOM)

SELECT PNOM, PRIX, FNOM FROM FOURNITURE  
 WHERE PNOM = ANY (SELECT PNOM FROM COMMANDE  
                   WHERE CNOM = 'JEAN')

Soit le schéma de relation **FOURNITURE**(PNOM, FNOM, PRIX)

REQUÊTE: *Fournisseurs qui fournissent au moins un produit avec un coût supérieur au coût des produits fournis par Jean*

SQL:

**Slide 129** SELECT DISTINCT P1.FNOM  
 FROM FOURNITURE P1  
 WHERE NOT EXISTS (SELECT \* FROM FOURNITURE P2  
                   WHERE P2.FNOM = 'JEAN'  
                   AND P1.PRIX <= P2.PRIX)

SELECT DISTINCT FNOM FROM FOURNITURE  
 WHERE PRIX > ALL (SELECT PRIX FROM FOURNITURE  
                   WHERE FNOM = 'JEAN')

### Division

Soit le schéma de relations

**Slide 130** **FOURNITURE**(FNUM,PNUM,QUANTITE)  
**PRODUIT**(PNUM,PNOM,PRIX)  
**FOURNISSEUR**(FNUM, FNOM, STATUS, VILLE)  
 REQUÊTE: *Fournisseurs qui fournissent tous les produits*

ALGÈBRE:

$$R1 := \pi_{FNUM,PNUM}(FOURNITURE) \div \pi_{PNUM}(PRODUIT)$$

$$R2 := \pi_{FNOM}(FOURNISSEUR \bowtie R1)$$

SQL:

**Slide 131**

```

SELECT FNOM
FROM FOURNISSEUR
WHERE NOT EXISTS
  (SELECT *
   FROM PRODUIT
   WHERE NOT EXISTS
    (SELECT *
     FROM FOURNITURE
     WHERE FOURNITURE.FNUM = FOURNISSEUR.FNUM
     AND FOURNITURE.PNUM = PRODUIT.PNUM) )
  
```

Slide 132

## Fonctions de Calcul

### COUNT, SUM, AVG, MIN, MAX

REQUÊTE: *Nombre de Fournisseurs de Paris*

```
SELECT COUNT(*) FROM FOURNISSEUR
WHERE VILLE = 'Paris'
```

Slide 133

REQUÊTE: *Nombre de Fournisseurs qui fournissent actuellement des produits*

```
SELECT COUNT(DISTINCT FNOM) FROM FOURNITURE
```

**NOTE:** La fonction COUNT(\*) compte le nombre des  $n$ -uplets du résultat d'une requête sans élimination des doublés ni vérification des valeurs nulles. Dans le cas contraire on utilise la clause COUNT(UNIQUE ...).

REQUÊTE: *Quantité totale de Briques commandées*

```
SELECT SUM (QUANTITE)
FROM   COMMANDES
WHERE  PNOM = 'Brique'
```

**Slide 134**

REQUÊTE: *Coût moyen de Briques fournies*

```
SELECT AVG (PRIX)           SELECT SUM (PRIX)/COUNT(PRIX)
FROM   FOURNITURE           FROM FOURNITURE
WHERE  PNOM = 'Brique'     WHERE PNOM = 'Brique'
```

REQUÊTE: *Le prix des briques qui sont le plus chères.*

**Slide 135**

```
SELECT MAX (PRIX)
FROM   FOURNITURE
WHERE  PNOM = 'Briques';
```

REQUÊTE: *Fournisseurs des Briques au coût moyen des Briques*

**Slide 136**

```
SELECT FNOM
FROM FOURNITURE
WHERE PNOM = 'Brique' AND
      PRIX < (SELECT AVG(PRIX)
              FROM FOURNITURE
              WHERE PNOM = 'Brique')
```

**Slide 137**

**Opérations d'Agrégation**

**GROUP BY**REQUÊTE: *Nombre de fournisseurs par ville***Slide 138**

```

SELECT VILLE , COUNT( FNOM )
FROM   FOURNISSEUR
GROUP BY VILLE

```

LA BASE ET LE RESULTAT :

VILLE	FNOM
PARIS	TOTO
PARIS	DUPOND
LYON	DURAND
LYON	LUCIEN
LYON	REMI

**Slide 139**

VILLE	COUNT(FNOM)
PARIS	2
LYON	3

**NOTE:** La clause GROUP BY permet de préciser les attributs de partitionnement des relations déclarées dans la clause FROM. Par exemple on regroupe les fournisseurs par ville.

REQUÊTE: *Donner pour chaque produit fourni son coût moyen*

```
SELECT PNOM, AVG ( PRIX )
FROM   FOURNITURE
GROUP BY PNOM
```

RÉSULTAT:

**Slide 140**

PNOM	AVG (PRIX)
BRIQUE	10.5
ARDOISE	9.8

**NOTE:** Les fonctions de calcul appliquées au résultat de regroupement sont directement indiquées dans la clause SELECT. Par exemple le calcul de la moyenne se fait par produit obtenu au résultat après le regroupement.

## HAVING

REQUÊTE: *Produits fournis par deux ou plusieurs fournisseurs avec un coût supérieur de 100*

**Slide 141**

```
SELECT PNOM
FROM   FOURNITURE
WHERE  PRIX > 100
GROUP BY PNOM
HAVING COUNT(*) >= 2
```

Slide 142

AVANT LA CLAUSE HAVING

PNOM	FNOM	PRIX
BRIQUE	TOTO	105
ARDOISE	LUCIEN	110
ARDOISE	DURAND	120

APRÈS LA CLAUSE HAVING

PNOM	FNOM	PRIX
ARDOISE	LUCIEN	110
ARDOISE	DURAND	120

**NOTE:** La clause HAVING permet d'éliminer des partitionnements, comme la clause WHERE élimine des  $n$ -uplets du résultat d'une requête.

Slide 143

**REQUÊTE:** *Produits fournis et leur coût moyen pour les fournisseurs dont le siège est à Paris seulement si le coût minimum du produit est supérieur à 1000F*

```

SELECT PNOM, AVG(PRIX)
FROM FOURNITURE, FOURNISSEUR
WHERE VILLE = 'Paris' AND
      FOURNITURE.FNOM = FOURNISSEUR.FNOM
GROUP BY PNOM
HAVING MIN(PRIX) > 1000

```

### **ORDER BY**

En général, le résultat d'une requête SQL n'est pas trié. Pour trier le résultat par rapport aux valeurs d'un ou de plusieurs attributs, on utilise la clause `ORDER BY` :

**Slide 144**

```
SELECT VILLE, FNOM, PNOM
   FROM FOURNITURE, FOURNISSEUR
  WHERE FOURNITURE.FNOM = FOURNISSEUR.FNOM
 ORDER BY VILLE, FNOM DESC
```

Le résultat est trié par les villes (ASC) et le noms des fournisseur dans l'ordre inverse (DESC).

### Valeurs NULL, par défaut (\*)

#### Slide 146

- **Valeur NULL pour un attribut:** valeur spéciale qui représente une information inconnue (manquante, non fournie)
- attention ce n'est ni zéro, ni chaîne vide
- pour forcer un attribut à toujours avoir une valeur, option *default*, exemple: default "manquant": si l'attribut n'a pas de valeur fournie, on la force à "manquant"

### La valeur NULL (\*)

#### Slide 147

1.  $A \theta B$  est inconnu (ni vrai, ni faux) si la valeur de  $A$  ou/et  $B$  est NULL ( $\theta$  est l'un de  $=, <, \leq, >, \geq, \neq$ ).
2.  $A \text{ op } B$  est NULL si la valeur de  $A$  ou/et  $B$  est NULL ( $\text{op}$  est l'un de  $+, -, *, /$ ).

### Trois Valeurs de Vérité (\*)

Trois valeurs de vérité: vrai, faux et **inconnu**

- 1. vrai AND inconnu = inconnu
- 2. faux AND inconnu = faux
- Slide 148** 3. inconnu AND inconnu = inconnu
- 4. vrai OR inconnu = vrai
- 5. faux OR inconnu = inconnu
- 6. inconnu OR inconnu = inconnu
- 7. NOT inconnu = inconnu

Soit le schéma de relation **FOURNISSEUR**(FNOM,STATUT,VILLE)

REQUÊTE: *Les Fournisseurs de Paris.*

SQL:

**Slide 149**

```
SELECT FNOM
FROM   FOURNISSEUR
WHERE  VILLE = 'Paris'
```

On ne trouve pas les fournisseurs avec VILLE = NULL !

Soit le schéma de relation **FOURNISSEUR**(FNOM,STATUT,VILLE)

REQUÊTE: *Fournisseurs dont l'adresse est inconnu.*

SQL:

**Slide 150**

```
SELECT FNOM
FROM   FOURNISSEUR
WHERE  VILLE IS NULL
```

**NOTE:** Le prédicat IS NULL (ou IS NOT NULL) n'est pas exprimable en algèbre relationnelle.

### Exemple (\*)

Soit le schéma de relation **EMPLOYE**(EMPNO,ENOM,DEPNO,SAL)

SQL:

**Slide 151**

```
SELECT E1.ENOM
FROM   EMPLOYE E1, EMPLOYE E2
WHERE  E1.SAL > 20000 OR
       E1.SAL <= 20000
```

*Est-ce qu'on trouve les noms de tous les employés s'il y a des employés avec un salaire inconnu ?*

## Contraintes

Contraintes que l'on peut exprimer lors de la création de tables et que le système peut maintenir:

1. **NOT NULL**: un attribut doit toujours avoir une valeur
2. **PRIMARY KEY**: un (groupe d') attribut(s) constitue(nt) la clé primaire
3. **UNIQUE** un (groupe d') attribut(s) constitue(nt) une clé (secondaire).
4. **FOREIGN KEY (A) REFERENCES R** L'attribut A est la clé primaire de la table R (clé étrangère).
5. **CHECK**: contrainte sur la valeur

Slide 152

## Création de Tables

On crée une table avec la commande **CREATE TABLE** :

```
CREATE TABLE Produit(pnom VARCHAR(20),
                    prix INTEGER,
                    PRIMARY KEY (pnom));
```

Slide 153

```
CREATE TABLE Fournisseur(fnom VARCHAR(20) PRIMARY KEY,
                        ville VARCHAR(16));
```

```
CREATE TABLE Fourniture (pnom VARCHAR(20) NOT NULL,
                        fnom VARCHAR(20) NOT NULL,
                        FOREIGN KEY (pnom) REFERENCES Produit,
                        FOREIGN KEY (fnom) REFERENCES Fournisseur);
```

### Création de Tables

Slide 154

```
CREATE TABLE ARTISTE (id INTEGER NOT NULL,  
    nom VARCHAR(30) NOT NULL,  
    anneeNaiss INTEGER, default '1900',  
    CHECK (anneeNaiss BETWEEN 1900 AND 2000),  
    film VARCHAR (30),  
    PRIMARY KEY (id),  
    UNIQUE(nom),  
    FOREIGN KEY (film) REFERENCES FILM)  
  
CREATE TABLE FILM (idfilm VARCHAR (30) NOT NULL,...)
```

### Intégrité référentielle

Le système vérifie les contraintes d'intégrité référentielle. Ses réactions dépendent des options choisies lors de la création des tables, par exemple:

Slide 155

1. si on insère un artiste avec l'attribut film renseigné, le film doit exister dans la table Film
2. si un film est supprimé dans la relation FILM, les nuplets qui réfèrent ce film dans la table ARTISTE ont l'attribut film mis à **NULL** (option *ON DELETE SET NULL*).
3. répercute une maj de l'id faite dans FILM, dans les nuplets qui réfèrent ce film dans ARTISTE (option *ON UPDATE CASCADE*).

### Destruction de Tables

On détruit une table avec la commande **DROP TABLE** :

**Slide 156**

```
DROP TABLE Fourniture;  
DROP TABLE Produit;  
DROP TABLE Fournisseur;
```

### Insertion de n-uplets

On insère dans une table avec la commande **INSERT** dont voici la syntaxe.

```
INSERT INTO  $R(A_1, A_2, \dots, A_n)$  VALUES ( $v_1, v_2, \dots, v_n$ )
```

**Slide 157**

Donc on donne deux listes : celles des attributs (les  $A_i$ ) de la table et celle des valeurs respectives de chaque attribut (les  $v_i$ ).

1. Bien entendu, chaque  $A_i$  doit être un attribut de  $R$
2. Les attributs non-indiqués restent à **NULL** ou à leur valeur par défaut.
3. On doit toujours indiquer une valeur pour un attribut déclaré **NOT NULL**

### Insertion : exemples

Insertion d'une ligne dans *Produit* :

```
INSERT INTO Produit (pnom, prix)  
VALUES ('Ojax', 15)
```

#### Slide 158

Insertion de deux fournisseurs :

```
INSERT INTO Fournisseur (fnom, ville)  
VALUES ('BHV', 'Paris'), ('Casto', 'Paris')
```

Il est possible d'insérer plusieurs lignes en utilisant **SELECT**

```
INSERT INTO NomsProd (pnom)  
SELECT DISTINCT pnom FROM Produit
```

### Modification

On modifie une table avec la commande **UPDATE** dont voici la syntaxe.

```
UPDATE R SET  $A_1 = v_1, A_2 = v_2, \dots, A_n = v_n$   
WHERE condition
```

#### Slide 159

Contrairement à **INSERT**, **UPDATE** s'applique à un ensemble de lignes.

1. On énumère les attributs que l'on veut modifier.
2. On indique à chaque fois la nouvelle valeur.
3. La clause **WHERE** *condition* permet de spécifier les lignes auxquelles s'applique la mise à jour. Elle est identique au **WHERE** du **SELECT**

Bien entendu, on ne peut pas violer les contraintes sur la table.

### Modification : exemples

Mise à jour du prix d'Ojax :

```
UPDATE Produit SET prix=17
WHERE pnom = 'Ojax'
```

Slide 160

Augmenter les prix de tous les produits fournis par BHV par 20% :

```
UPDATE Produit SET prix = prix*1.2
WHERE pnom in (SELECT pnom
FROM Fourniture
WHERE fnom = 'BHV')
```

### Destruction

On détruit une ou plusieurs lignes dans une table avec la commande **DELETE**

Slide 161

```
DELETE FROM R
WHERE condition
```

C'est la plus simple des commandes de mise-à-jour puisque elle s'applique à des lignes et pas à des attributs. Comme précédemment, la clause **WHERE condition** est indentique au **WHERE** du **SELECT**

### **Destruction : exemples**

Destruction des produits fournis par le BHV :

**Slide 162**

```
DELETE FROM Produit  
WHERE pnom in (SELECT pnom  
FROM Fourniture  
WHERE fnom = 'BHV')
```

Destruction du BHV :

```
DELETE FROM Fournisseur  
WHERE fnom = 'BHV'
```

### Organisation des données en mémoire secondaire

1. Le disque est divisé en **blocs physiques** (ou **pages**) de tailles égales.
2. Accès à un bloc par son adresse (par exemple le numéro de cylindre + le numéro de secteur + le numéro de face).
3. Le bloc est l'unité d'échange entre la mémoire secondaire et la mémoire principale

**Slide 164**

### Les fichiers

Les données sont stockées dans des **fichiers** :

- Un fichier occupe un ou plusieurs blocs sur un disque.
- L'accès aux fichiers est géré par un logiciel spécifique : le Système de Gestion de Fichiers (SGF).
- Un fichier est caractérisé par son nom.
- Enfin un fichier est un ensemble **d'articles**.

**Slide 165**

## Les articles

Un article est une séquence de **champs**.

**Slide 166**

### 1. Articles en format fixe.

- (a) La taille de chaque champ est fixée
- (b) Taille et nom des champs dans le **descripteur** de fichier.

### 2. Articles en format variable.

- (a) La taille de chaque champ est variable.
- (b) L'en tête du champ donne la taille réelle

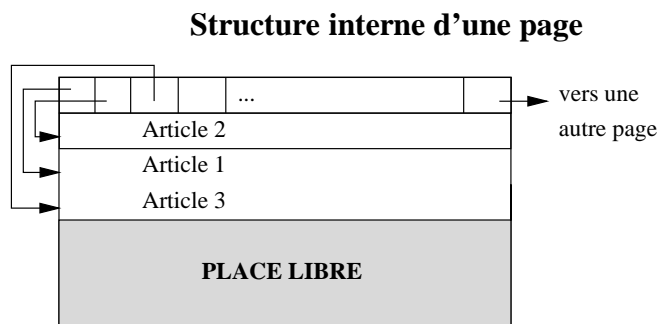
## Articles et pages

L'adresse d'un article est constituée de

**Slide 167**

- 1. L'adresse de la page dans laquelle il se trouve.
- 2. Un entier : indice d'une table placée en début de page qui contient l'adresse réelle de l'article dans la page.

Slide 168



### Opérations sur les fichiers

1. **Insérer un article.**
2. **Modifier un article**
3. **Détruire un article**
4. **Rechercher un ou plusieurs article(s)**
  - Par adresse
  - Par valeur d'un ou plusieurs champs.

Slide 169

**Hypothèse:** Le **coût** d'une opération est surtout fonction du **nombre d'E/S** (nb de pages)

## Organisation de fichiers

L'organisation d'un fichier est caractérisée  
par le mode de répartition des articles dans les pages

**Slide 170** Il existe trois organisations principales :

1. Fichiers séquentiels
2. Fichiers indexés (séquentiels indexés et arbres-B)
3. Hachage

## Exemple de référence

Organisation d'un fichier contenant les articles suivants :

**Slide 171**

<i>Vertigo 1958</i>	<i>Annie Hall 1977</i>
<i>Brazil 1984</i>	<i>Jurassic Park 1992</i>
<i>Twin Peaks 1990</i>	<i>Metropolis 1926</i>
<i>Underground 1995</i>	<i>Manhattan 1979</i>
<i>Easy Rider 1969</i>	<i>Reservoir Dogs 1992</i>
<i>Psychose 1960</i>	<i>Impitoyable 1992</i>
<i>Greystoke 1984</i>	<i>Casablanca 1942</i>
<i>Shining 1980</i>	<i>Smoke 1995</i>

### Organisation séquentielle

Slide 172

- **Insertion** : les articles sont stockés séquentiellement dans les pages au fur et à mesure de leur création.
- **Recherche** : le fichier est parcouru séquentiellement.
- **Destruction** : recherche, puis destruction (par marquage d'un bit par exemple).
- **Modification** : recherche, puis réécriture.

### Coût des opérations

Nombre moyen de lectures/écritures sur disque,

Fichier de  $n$  pages.

Slide 173

- **Recherche** :  $\frac{n}{2}$ . On parcourt en moyenne la moitié du fichier.
- **Insertion** :  $1 + 1 = 2$  ( $n + 1$  si on vérifie que l'article n'existe pas avant d'écrire).
- **Destruction et mises-à-jour** :  
 $\frac{n}{2} + 1$ .

⇒ organisation utilisée pour les fichiers de petite taille.

### Fichiers séquentiels triés

Lorsque la recherche est faite par valeur d'un champ, celui-ci est appelé **clé d'accès**.

Si le fichier est **trié** sur sa clé d'accès, on peut effectuer une recherche par **dichotomie** :

#### Slide 174

1. On lit la page qui est "au milieu" du fichier.
2. Selon la valeur de la clé du premier enregistrement de cette page, on sait si l'article cherché est "avant" ou "après".
3. On recommence avec le demi-fichier où se trouve l'article recherché.

Coût de l'opération :  $\log_2(n)$ .

### Ajout d'un index

L'opération de recherche peut encore être améliorée en utilisant un **index** sur un fichier **trié**.

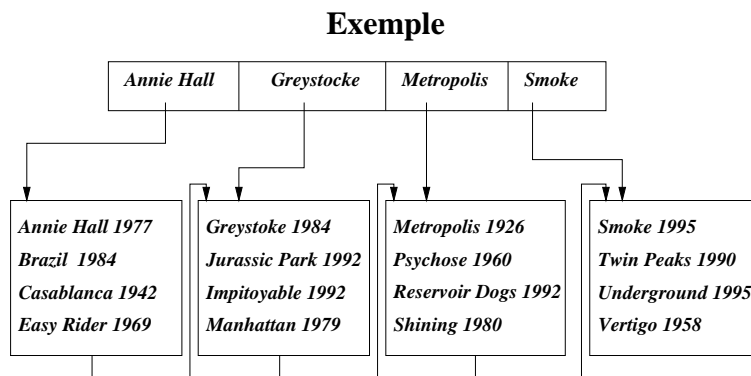
Un index est un second fichier possédant les caractéristiques suivantes :

#### Slide 175

1. Les articles sont des couples (Valeur de clé, Adresse de page)
2. Une occurrence  $(v, b)$  dans un index signifie que le premier article dans la page  $b$  du fichier trié a pour valeur de clé  $v$ .

L'index est lui-même **trié** sur la valeur de clé.

Slide 176



### Recherche avec un index

Chercher l'article dont la valeur de clé est  $v_1$ .

1. On recherche dans l'index (séquentiellement ou - mieux - par dichotomie) la plus grande valeur  $v_2$  telle que

Slide 177

$$v_2 < v_1.$$

2. On lit la page désignée par l'adresse associée à  $v_2$  dans l'index.
3. On cherche séquentiellement les articles de clé  $v_1$  dans cette page.

### Coût d'une recherche avec ou sans index

Soit un fichier  $F$  contenant 1000 pages. On suppose qu'une page d'index contient 100 entrées, et que l'index occupe donc 10 pages.

#### Slide 178

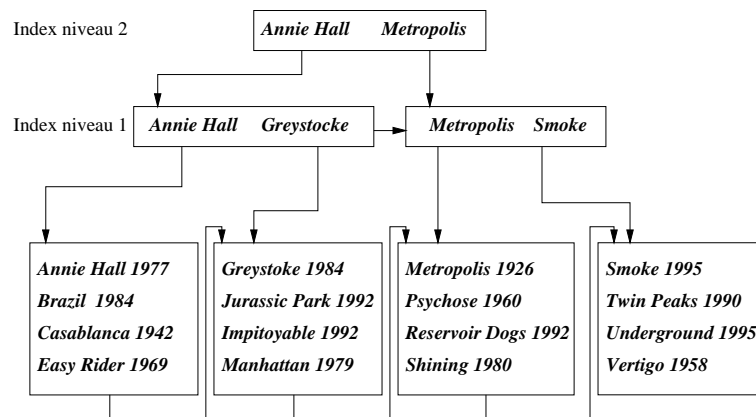
- $F$  non trié et non indexé. Recherche séquentielle : **500 pages**.
- $F$  trié et non indexé. Recherche dichotomique :  $\log_2(1000)=$ **10 pages**
- $F$  trié et indexé. Recherche dichotomique sur l'index, puis lecture d'une page :  $\log_2(10) + 1=$ **5 pages**

### Séquentiel indexé

#### Slide 179

1. Un index est un fichier : on peut lui-même l'indexer :
2. On obtient un index à plusieurs niveaux sur la clé.
3. C'est un **arbre** dont les feuilles constituent le fichier et les noeuds internes l'index.

## Slide 180



### Index dense et index non dense

L'index ci-dessus est **non dense** : une seule valeur de clé dans l'index pour l'ensemble des articles du fichier indexé  $F$  situés dans une même page.

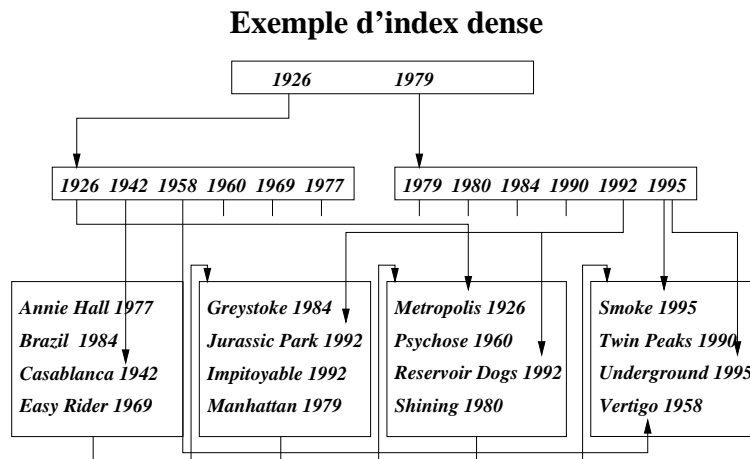
Un index est **dense** ssi il existe une valeur de clé dans l'index pour chaque article dans le fichier  $F$ .

## Slide 181

Remarques :

1. On ne peut créer un index non-dense que sur un fichier trié (et un seul index non-dense par fichier).
2. Un index non-dense est beaucoup moins volumineux qu'un index dense.

Slide 182



### Autres opérations : insertion

Etant donné un article  $A$  de clé  $v_1$ , on effectue d'abord une recherche pour savoir dans quelle page  $p$  il doit être placé. Deux cas de figure :

Slide 183

1. Il y a une place libre dans  $p$ . Dans ce cas on réorganise le contenu de  $p$  pour placer  $A$  à la bonne place.
2. Il n'y a plus de place dans  $p$ . On crée une **page de débordement**.

Exercice: montrer l'index non dense précédent après l'insertion de [Insomnia, 2002], [L'homme sans passé, 2002], [Le Destin fabuleux..., 2001], [L'auberge espagnole, 2002], [Jonathan aura 25 ans en l'an 2000, 1975].

### **Autres opérations : destructions et mises-à-jour**

Relativement facile en général :

**Slide 184**

1. On recherche l'article.
2. On applique l'opération.

⇒ on peut avoir à réorganiser le fichier et/ou l'index, ce qui peut être coûteux.

### **Inconvénients du séquentiel indexé**

Organisation bien adaptée aux fichiers qui évoluent peu. En cas de grossissement :

**Slide 185**

1. Une page est trop pleine → on crée une page de débordement.
2. On peut aboutir à des chaînes de débordement importantes pour certaines pages.
3. Le temps de réponse peut se dégrader et dépend de l'article recherché

⇒ on a besoin d'une structure permettant une réorganisation dynamique sans dégradation de performances.

## Arbres-B

Un arbre-B (pour *balanced tree* ou **arbre équilibré**) est une structure arborescente dans laquelle tous les chemins de la racine aux feuilles ont même longueur.

### Slide 186

Si le fichier grossit : la hiérarchie grossit **par le haut**.  
L'arbre-B est utilisé dans **tous** les SGBD relationnels (avec des variantes).

## Arbre-B : définition

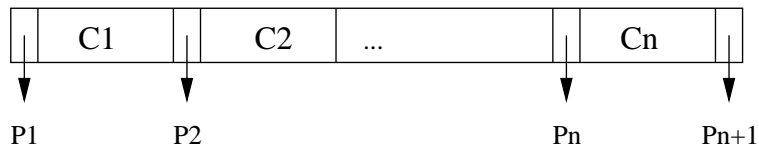
Un arbre-B **d'ordre  $k$**  est un arbre équilibré tel que :

### Slide 187

1. Chaque noeud est une page contenant au moins  $k$  et au plus  $2k$  articles,  $k \in \mathbb{N}$ .
2. Les articles dans un noeud sont triés sur la clé.
3. Chaque "père" est un index pour l'ensemble de ses fils.
4. Chaque noeud contenant  $n$  articles a  $n + 1$  fils.
5. La racine a 0 ou au moins deux fils.

On décrit la variante appelée **arbre B+**

**Structure d'un noeud dans un arbre-B d'ordre  $k$**



**Slide 188**

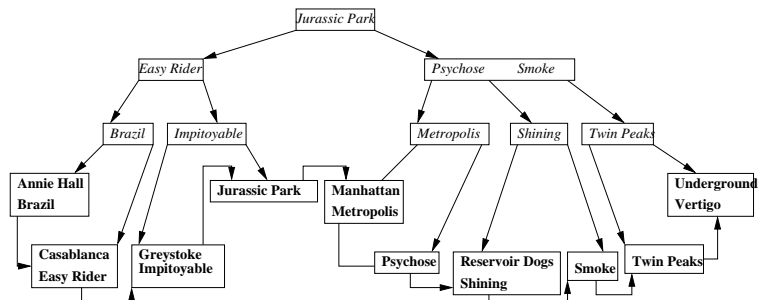
Les  $C_i$  sont les clés des articles, les  $R_i$  représentent

le reste des attributs d'un article de clé  $C_i$ . Les

$P_i$  sont les pointeurs vers les noeuds fils dans l'index. NB :

$k \leq n \leq 2k$ .

**Exemple d' arbre-B**



**Slide 189**

### L'arbre B

**Slide 190**

- Les feuilles de l'arbre contiennent des couples [valeur de clé, adresse d'article dans le fichier indexé par l'arbre]
- Les feuilles sont chaînées entre elles

### Recherche dans un arbre-B

Rechercher les articles de clé  $C$ .

A partir de la racine, appliquer récursivement

l'algorithme suivant :

**Slide 191**

Soit  $C_1, \dots, C_n$  les valeurs de clés de la page courante.

1. Si  $C \leq C_1$  (ou  $C > C_n$ ), on continue la recherche avec le noeud référencé par  $P_1$  (ou  $P_{n+1}$ ).
2. Sinon, il existe  $i \in [1, k[$  tel que  $C_i < C \leq C_{i+1}$ , on continue avec la page référencée par le pointeur  $P_{i+1}$ .

### Insertion dans un arbre-B d'ordre $k$

On recherche la feuille de l'arbre où le couple (clé, adresse) doit prendre place:

- Slide 192**
1. On l'y insère. Si la page  $p$  déborde (elle contient  $2k + 1$  éléments): on alloue une nouvelle page  $p'$ .
  2. On place les  $k + 1$  premiers articles (ordonnés selon la clé) dans  $p$  et les  $k$  derniers dans  $p'$ .
  3. On insère le  $k + 1^e$  article dans le père de  $p$ . Son pointeur gauche référence  $p$ , et son pointeur droit référence  $p'$ .
  4. Si le père déborde à son tour, on continue comme en 1 (sauf qu'en 2 on ne place que les  $k$  premiers articles dans  $p$  et non les  $k+1$ ).

Exercice: insérer Amélie Poulain et Citizen Kane

### Destruction dans un arbre-B d'ordre $k$

Chercher la page  $p$  contenant l'article. Si c'est une feuille :

- Slide 193**
1. On détruit l'article.
  2. S'il reste au moins  $k$  articles dans  $p$ , c'est fini. Sinon :
    - (a) Si une feuille "soeur" contient plus de  $k$  articles, on effectue une permutation pour rééquilibrer les feuilles. Ex : destruction de *Smoke*.
    - (b) Sinon on "descend" un article du père dans  $p$ , et on réorganise le père.

Ex : destruction de *Reservoir Dogs*

Supposons maintenant qu'on détruit un article dans un noeud interne. Il faut réorganiser :

**Slide 194**

1. On détruit l'article
2. On le remplace par l'article qui a la plus grande valeur de clé dans le sous-arbre gauche. Ex : destruction de *Psychose*, remplacé par *Metropolis*
3. On vient de retirer un article dans une feuille : si elle contient moins de  $k$  éléments, on procède comme indiqué précédemment.

⇒ toute destruction a donc un effet seulement **local**.

### Quelques mesures pour l'arbre-B

Hauteur  $h$  d'un arbre-B d'ordre  $k$  contenant  $n$  articles :

$$\log_{2k+1}(n+1) \leq h \leq \log_{k+1}\left(\frac{n+1}{2}\right)$$

**Slide 195**

Exemple pour  $k = 100$  :

1. si  $h = 3$ ,  $n \leq 8 \times 10^6$
2. si  $h = 4$ ,  $n \leq 1,6 \times 10^9$

Les opérations d'accès coûtent au maximum  $h$  E/S.

## Hachage

Accès direct à la page contenant l'article recherché :

1. On estime le nombre  $N$  de pages qu'il faut allouer au fichier.
- Slide 196** 2. **fonction de hachage**  $H$  : à toute valeur de la clé de domaine  $V$  associe un nombre entre 0 et  $N - 1$ .

$$H : V \rightarrow \{0, 1, \dots, N - 1\}$$

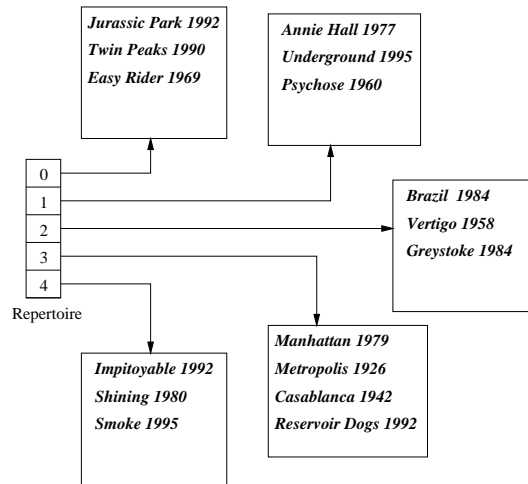
3. On range dans la page de numéro  $i$  tous les articles dont la clé  $c$  est telle que  $H(c) = i$ .

### Exemple : hachage sur le fichier *Films*

On suppose qu'une page contient 4 articles :

1. On alloue 5 pages au fichier.
- Slide 197** 2. On utilise une fonction de hachage  $H$  définie comme suit :
  - (a) clé : nom d'un film, on ne s'intéresse qu'à l'initiale de ce nom.
  - (b) On numérote les lettres de l'alphabet de 1 à 26 :  
 $No('a') = 1, No('m') = 13$ , etc.
  - (c) Si  $l$  est une lettre de l'alphabet,  $H(l) = MODULO(No(l), 5)$ .

## Slide 198



## Remarques

1. Le nombre  $H(c) = i$  n'est pas une adresse de page, mais l'indice d'une table ou "répertoire"  $R$ .  $R(i)$  contient l'adresse de la page associée à  $i$
2. si ce répertoire ne tient pas en mémoire centrale, la recherche coûte plus cher.
3. Une propriété essentielle de  $H$  est que la distribution des valeurs obtenues soit uniforme dans  $\{0, \dots, N - 1\}$
4. Quand on alloue un nombre  $N$  de pages, il est préférable de prévoir un remplissage partiel (non uniformité, grossissement du fichier). On a choisi 5 pages alors que 4 (16 articles / 4) auraient suffi.

## Slide 199

### Hachage : recherche

Etant donné une valeur de clé  $v$  :

**Slide 200**

1. On calcule  $i = H(v)$ .
2. On consulte dans la case  $i$  du répertoire l'adresse de la page  $p$ .
3. On lit la page  $p$  et on y recherche l'article.

⇒ **donc une recherche ne coûte qu'une seule lecture.**

### Hachage : insertion

Recherche par  $H(c)$  la page  $p$  où placer  $A$  et l'y insérer.

Si la page  $p$  est pleine, il faut :

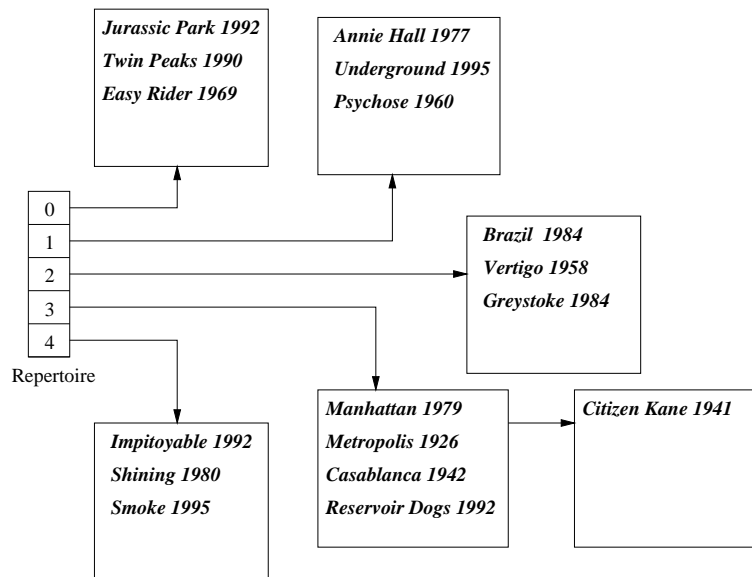
**Slide 201**

1. Allouer une nouvelle page  $p'$  (de débordement).
2. Chaîner  $p'$  à  $p$ .
3. Insérer  $A$  dans  $p'$ .

⇒ lors d'une recherche, il faut donc en fait parcourir la liste des pages chaînées correspondant à une valeur de  $H(v)$ .

Moins la répartition est uniforme, plus il y aura de débordements

## Slide 202



### Hachage : avantages et inconvénients

Intérêt du hachage :

1. **Très rapide.** Une seule E/S dans le meilleur des cas pour une recherche (répertoire résidant en mémoire)
2. Le hachage, contrairement à un index, **n'occupe aucune place disque.**

## Slide 203

En revanche :

1. Il faut penser à réorganiser les fichiers qui évoluent beaucoup.
2. **Les recherches par intervalle sont impossibles.**

**Comparatif**

<b>Organisation</b>	<b>Coût</b>	<b>Avantages</b>	<b>Inconvénients</b>
Sequentiel	$\frac{n}{2}$	Simple	Très coûteux !
Indexé	$\log_2(n)$	Efficace Intervalles	Peu évolutive
Arbre-B	$\log_k(n)$	Efficace Intervalles	Traversée
Hachage	1+	Le plus efficace	Intervalles impossibles

Slide 204