INFORMATIQUE - CNAM, Paris
BASES DE DONNÉES Relationnelles
Équipe VERTIGO

Slide 1

Cours SGBD 1/2 valeur B7, 19786

M. Scholl, B. Amann, P. Rigaux et D. Vodislav

(scholl|amann|rigaux|vodislav)@cnam.fr

2000/2001

Table de Matières

Slide 4

INTRODUCTION

Slide 5

A QUI S'ADRESSE LE COURS?

Slide 6

Aux étudiants du cycle B du CNAM

OBJECTIF: Comprendre et Maitriser la technologie relationnelle

BIBLIOGRAPHIE

Ouvrages en français

1. Carrez C., *Des Structures aux Bases de Données*, Masson
2. Date C.J, *Introduction aux Bases de Données*, Vuibert, 970 Pages, Janvier 2001

Ouvrages en anglais

Slide 7

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

Le standard SQL

Slide 8

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

Slide 9

1. Introduction
2. Modèle et Algèbre Relationnels
3. Calcul Relationnel
4. SQL
5. Conception d'un Bon Schéma Relationnel
6. Organisation Physique, Index
7. Algorithmes de Jointure
8. Optimisation des requêtes; l'exemple d'ORACLE.
9. Concurrence d'accès et reprise sur pannes.

Exemples d'Applications

Slide 10

1. **CLASSIQUES**
   - Gestion (salaires, stocks, . . .)
   - Transactionnel (comptes, centrales d'achat, . . .)
   - Réservations (avions, trains, . . .)
2. **MULTIMÉDIA**
   - Librairie électronique (bibliothèques, journaux, web, . . .)
   - Documentation technique (nomenclature, plans, dessins,. . .)
   - Bureautique (formulaires, textes, images, son, . . .)
   - Géographique (cartes routières, thématiques, . . .)
   - Génie Logiciel (programmes, manuels, . . .)

Comment Stocker et Manipuler les Données?

DONNÉES → BASE DE DONNÉES (B.D.)

Slide 11

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

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

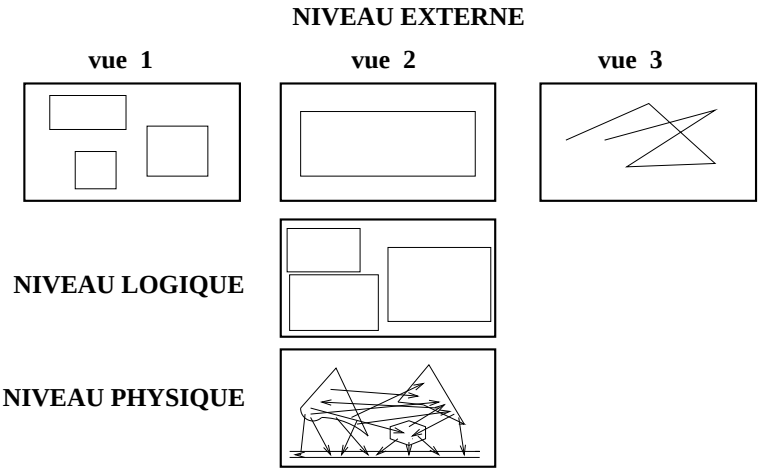
Diversité des utilisateurs, des interfaces et des Architectures:

Slide 12

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

ARCHITECTURE d'un SGBD : ANSI-SPARC (1975)

Slide 13



FONCTIONNALITÉS d'un SGBD

Slide 14

Chaque niveau du SGBD réalise un certain nombre de fonctions :

NIVEAU PHYSIQUE

- 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)

Slide 15

NIVEAU LOGIQUE

- 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)

Niveau Externe : Vues Utilisateurs

Slide 16

1. **Vue de la planification des salles :** pour chaque cours
   - Nom de Prof
   - Horaires et salles
2. **Vue de la paye :** un ensemble de Prof
   (nom, prénom, adresse, indice, nombre d'heures. . .)
3. **Vue du service de scolarité (suivi des élèves) :** . . .

Intégration de ces Vues

Slide 17

1. On laisse chaque usager avec sa vision du monde
2. PASSAGE DU NIVEAU EXTERNE AU NIVEAU LOGIQUE:
   On “intègre” l’ensemble de ces vues en une description **unique**:
   le SCHÉMA LOGIQUE

Slide 18

Fonctionnalités du SGBD au NIVEAU EXTERNE

- 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

Slide 19

un GROS VOLUME D'INFORMATIONS

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

Définition du schéma de données

Slide 20

Modèles de données

Slide 21

Un modèle de données est caractérisé par :

- Une structuration des objets
- Des opérations sur ces objets

Slide 22

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

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

$\Rightarrow$ ces différents modèles correspondent aux niveaux dans l'architecture d'un SGBD.

Modèle Conceptuel: Exemple Entité-Relation

Slide 23

- 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

Slide 24

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

Ces langages sont **abstraits** :

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

Slide 25

1. **Simplicité d'accès** Les structures et les langages sont plus simples, donc plus faciles pour l'usager 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.

HISTORIQUE DES SGBD

Slide 26

À chaque génération correspond un modèle logique

Les premiers étaient peu abstraits (navigationnels)

< 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	RELATIONNEL ETENDU	nouvelles applications
	DATALOG (SGBD déductifs)	pas encore de marché
	ORIENTÉ-OBJET	navig. + déclaratif

Slide 27

Opérations sur les données

Exemples de questions (requêtes) pos’ees à la base

Slide 28

Insérer un employé nommé Jean

Augmenter Jean de 10%

Détruire Jean

Chercher les employés cadres

Chercher les employés du département comptabilité

Salaire moyen des employés comptables, avec deux enfants, nés avant 1960 et travaillant à Paris

Quels types d'opérations ?

Slide 29

4 types d'opérations classiques (ou **requêtes**) :

1. La **création** (ou **insertion**).
2. La **modification** (ou **mise-à-jour**).
3. La **destruction**.
4. La **recherche**.

Ces opérations correspondent à des commandes du LMD. La plus complexe est la **recherche** en raison de la variété des critères.

Le Traitement d'une Requête

Slide 30

- **ANALYSE SYNTAXIQUE**
- **OPTIMISATION**
  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

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

Slide 31

- 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.
- Donner une image cohérente des données si l'un fait des requêtes et l'autre des mises-à-jour.

Le but : éviter les blocages, tout en empêchant des modifications anarchiques.

Le Facteur Humain

- **L'éditeur (le constructeur) du SGBD**
- **L'administrateur de la base**

Slide 32

Rôle de l'administrateur

- discute avec les différents utilisateurs
- conception d'un schéma logique (et des différentes vues)
- conception du schéma physique
- installation de la base et réglages fins (tuning)
- gère l'évolution de la base (nouveaux besoins, utilisateurs)

Outils à sa disposition fournis par l'éditeur du SGBD

Le Facteur Humain

Slide 33

- **Utilisateur expert:** informaticien connaissant langages programmation et langages BD
- **Concepteur et programmeur d'application**
  à partir : besoins des différents utilisateurs, écrit l'application pour des utilisateurs "naïfs"
- **Utilisateur naïf:** du non spécialiste des SGBD au non informaticien.

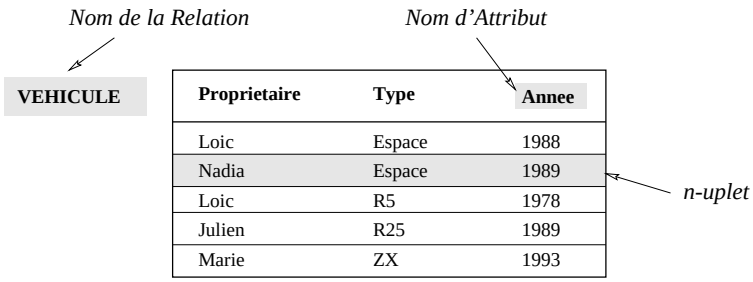
LE MODèLE RELATIONNEL

Slide 34

Présentation Générale

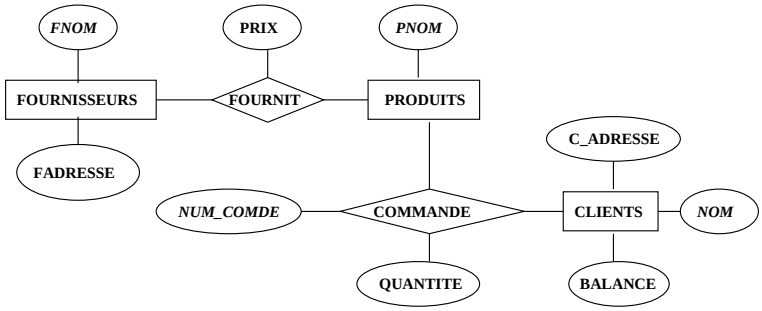
Exemple de Relation

Slide 35



Le Modèle Relationnel sur un Exemple

Slide 36



FOURNISSEUR

FNOM	FADRESSE
Abounayan	92190 Meudon
Cima	75010 Paris
Preblocs	92230 Gennevilliers
Sarnaco	75116 Paris

Slide 37

FOURNITURE

FNOM	PNOM	PRIX
Abounayan	sable	300
Abounayan	briques	1500
Preblocs	parpaing	1200
Sarnaco	parpaing	1150
Sarnaco	ciment	125

COMMANDES

NUM_COMDE	NOM	PNOM	QUANTITE
1	Jean	briques	5
2	Jean	ciment	10
3	Paul	briques	3
4	Paul	parpaing	9
5	Vincent	parpaing	7

Slide 38

CLIENTS

NOM	CADRESSE	BALANCE
Jean	75006 Paris	-12000
Paul	75003 Paris	0
Vincent	94200 Ivry	3000
Pierre	92400 Courbevoie	7000

Slide 39

Définitions

Définitions

Slide 40

- 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 $(v_1, \ldots, v_n)$ 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 $D_1 \times \ldots \times D_n$ entre des domaines $D_1, \ldots, D_n$ est l'ensemble de **tous** les nuplets $(v_1, \ldots, v_n)$ où $v_i \in D_i$.

Slide 41

- RELATION : soit $D_1, \ldots, D_n$ les domaines respectifs des attributs $A_1, \ldots, A_n$. Une relation R définie sur les attributs $A_1, \ldots, A_n$ est un sous-ensemble fini du produit cartésien $D_1 \times \ldots \times D_n$: 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 42

- 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 :

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

ou plus simplement :

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

Exemple :

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

Slide 43

- 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

Slide 44

SCHÉMA :

- 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

Slide 45

UNIVERS D'ATTRIBUTS :

- 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 et Langages

Slide 46

Opérations sur une Base de Données Relationnelle

Slide 47

- 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

Slide 48

- LANGAGE DE MANIPULATION DES DONNÉES
  - Saisie des nuplets d'une relation
  - Affichage d'une relation
  - 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

Slide 49

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

Les langages de requête relationnels utilisent deux approches :

- calcul relationnel
- algèbre relationnelle

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

Slide 50

ALGÈBRE RELATIONNELLE

Algèbre Relationnelle

Slide 51

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

Opérations de l'Algèbre Relationnelle

Slide 52

5 Opérations de base pour exprimer toutes les requêtes :

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

Projection

Slide 53

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

Notation :

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

Projection: Exemples

Slide 54

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

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

Slide 55

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

R

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

$\Rightarrow$ $\pi_{A,C}(R)$

A	C
a	c
d	b
c	d
a	e
e	a

Sélection

Slide 56

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

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

NOTATION :

$$\sigma_C(R)$$

Sélection: Exemples

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

Slide 57

R

A	B	C
a	b	1
d	a	2
c	b	3
a	b	4
e	e	5

$\Rightarrow \quad \sigma_{B=''b''}(R)$

A	B	C
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 58

R

A	B	C
a	b	1
d	a	2
c	b	3
a	b	4
e	e	5

$\Rightarrow \quad \sigma_{(A=''a'' \vee B=''a'') \wedge C \leq 3}(R)$

A	B	C
a	b	1
d	a	2

Sélection: Exemples

Slide 59

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

R

A	B	C
a	b	1
d	a	2
c	b	3
a	b	4
e	e	5

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

A	B	C
e	e	5

Condition de Sélection

Slide 60

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

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

Expressions de l'Algèbre Relationnelle

Slide 61

- 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 62

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

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

Slide 63

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

$$R(A_1, A_2, \ldots, A_n) \;\; S(B_1, B_2, \ldots, B_k)$$

- SCHÉMA DE $T = R \times S$: $T(A_1, A_2, \ldots, A_n, B_1, B_2, \ldots, B_k)$
- 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 64

R

	A	**B**
	1	1
$\mid R \mid$	1	2
	3	4

S

	C	**D**	**E**
	a	b	a
$\mid S \mid$	a	b	c
	b	a	a

$\Rightarrow$

Slide 65

$| R | \times | S |$

R × 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
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

Slide 66

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

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

  où $X_1, \ldots, X_k$ sont les attributs en commun.
- SCHÉMA DE $T = R \bowtie S$: $T(A_1, \ldots, A_m, B_1, \ldots, B_n, X_1, \ldots, X_k)$
- 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 67

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

Slide 68

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

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

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

Jointure Naturelle: Exemple

Slide 69

R

A	B
1	a
1	b
4	a

S

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

$\Rightarrow$

R × 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

$\Downarrow$

Slide 70

R ⋈ 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

Slide 71

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

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

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

θ-Jointure

- ARGUMENTS : 2 relations quelconques :

$$R(A_1, \ldots, A_m) \qquad S(B_1, \ldots, B_n)$$

Slide 72

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

θ-Jointure: Exemple

Slide 73

R

A	B
1	a
1	b
3	a

S

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

Slide 74

T := R × 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)**
= **R** $\bowtie_{A \leq C}$ **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	4	a	a

Équijointure: Exemple

Slide 75

R

A	B
1	a
1	b
3	a

S

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

Slide 76

T := R × S	**A**	**B**	**C**	**D**	**E**
$B \neq D \rightarrow$	1	**a**	1	**b**	a
$B \neq D \rightarrow$	1	**a**	2	**b**	c
	1	a	4	a	a
	1	b	1	b	a
	1	b	2	b	c
$B \neq D \rightarrow$	1	**b**	4	**a**	a
$B \neq D \rightarrow$	3	**a**	1	**b**	a
$B \neq D \rightarrow$	3	**a**	2	**b**	c
	3	a	4	a	a

$\Rightarrow$

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

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 77

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

$$\pi_{PROP}(\overbrace{IMMEUBLE \bowtie \sigma_{OCCUP=''DURAND''}(APPIM)}^{JointureNaturelle})$$

2. Appartements occupés par des propriétaires d'immeuble :

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

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

Slide 78

- Schéma Relationnel :

$$COMMANDES(PNOM, CNOM, NUM\_CMDE, QTE)$$

$$CLIENTS(CNOM, CADRESSE, BALANCE)$$

- Requête Relationnelle :

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

Union

Slide 79

- ARGUMENTS : 2 relations de même schéma :

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

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

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

Union: Exemple

Slide 80

R

A	B
a	b
a	c
d	e

S

A	B
a	b
a	e
d	e
f	g

$\Rightarrow$

R ∪ S

	A	B
→	a	b
	a	c
→	d	e
	a	e
	f	g

Différence

Slide 81

- ARGUMENTS : 2 relations de même schéma :

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

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

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

Différence: Exemple

Slide 82

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

S - R	A	B
	a	e
	f	g

Intersection

Slide 83

- ARGUMENTS : 2 relations de même schéma :

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

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

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

Intersection: Exemple

Slide 84

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 ∩ S = R - (R - S)

A	B
a	b
d	e

Semijointure

Slide 85

- ARGUMENTS : 2 relations quelconques :

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

  où $X_1, \ldots, X_k$ sont les attributs en commun.
- NOTATION : $R \rtimes S$
- SCHÉMA DE $T = R \rtimes S$: $T(A_1, \ldots, A_m, X_1, \ldots, X_k)$
- VALEUR DE $T = R \rtimes S$: Projection sur les attributs de R de la jointure naturelle entre R et S.

Semijointure

Slide 86

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

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

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

Semijointure: Exemple

Slide 87

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

$\Rightarrow \pi_{A,B,C}(R \bowtie S) \Rightarrow$

R $\rtimes$ S

A	**B**	**C**
a	b	c
d	b	c
c	a	d

Division: Exemple

Slide 88

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 89

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

R

NOM	PNOM
Jean	briques
Jean	ciment
Jean	parpaing
Paul	briques
Paul	parpaing
Vincent	parpaing

PROD

PNOM
briques
ciment
parpaing

$\Downarrow$

$R \div PROD$

NOM
Jean

Division: Exemple

Slide 90

R

A	B	C	D
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

S

C	D
x	m
y	n
z	o

R \ S

A	B
a	b
c	e

Slide 91

Division: Exemple

Slide 92

R

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

S

C	D
1	b
2	b
4	a

⇒

R ÷ S

A	B	E
1	a	a
3	c	b

Division

Slide 93

- ARGUMENTS : 2 relations :

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

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

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

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

Division

Slide 94 La division s'exprime en fonction du produit cartésien, de la projection et de la différence : $R \div S = R_1 - R_2$ où

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

Renommage

Slide 95

- NOTATION : ρ
- ARGUMENTS : 1 relation :
$$R(A_1, \ldots, A_n)$$
- SCHÉMA DE $T = \rho_{A_i \to B_i} R$: $T(A_1, \ldots, A_{i-1}, B_i, A_{i+1}, \ldots, A_n)$
- VALEUR DE $T = \rho_{A_i \to B_i} R$: $T = R$. La valeur de R est inchangée. Seul le nom de l'attribut A_i a été remplacé par B_i

Et si les attributs sont connus par leur rang?

Slide 96

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

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 $PNOM$

Slide 97

SQL

Principe

- SQL (Structured Query Language) est le Langage de Requêtes standard pour les SGBD relationnels

Slide 98

- Expression d'une requête par un bloc *SELECT FROM WHERE*

  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 99

SQL91 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 100

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

Slide 101

Expressions de Base

Projection

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

REQUÊTE: *Information sur toutes les commandes*

Slide 102

SQL:

```
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 103

```
SELECT PNOM
FROM   COMMANDES
```

NOTE: Contrairement à l'algèbre relationnelle, SQL n'élimine pas les dupliqués. 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 104

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 105

```
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:

Slide 106

1. *attribut comparateur valeur*
2. *attribut comparateur attribut*

où $comparateur$ est $=, <, >, <>, \ldots,$

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

REQUÊTE: *Produits de prix supérieur à 200F*

SQL:

Slide 107

```
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 108

SQL:

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

Conditions de sélection en SQL : Suite

Le $comparateur$ est BETWEEN, LIKE, IS NULL, IN

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

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

Slide 109

SQL:

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

NOTE: La condition y BETWEEN x AND z est équivalente à $y <= z$ AND $x <= y$.

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

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

SQL:

Slide 110

```
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.

La valeur NULL (*)

Slide 111

La valeur NULL est une valeur “spéciale” qui représente une *valeur (information) inconnue*.

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

Slide 112

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

REQUÊTE: *Les Fournisseurs de Paris.*

SQL:

```
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 113

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

NOTE: Le prédicat IS NULL (ou IS NOT NULL) n'est pas exprimable en 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 114

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

NOTE: La condition x IN $\{a, b, \ldots, z)$ est équivalente à $x = a$ OR $x = b$ OR ... OR $x = z$.

Jointure

Slide 115

Soit le schéma de relations

COMMANDES(NUM,CNOM,PNOM,QUANTITE)
FOURNITURE(PNOM,FNOM,PRIX)

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

ALGÈBRE :

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

Slide 116

SQL :

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

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:

Slide 117

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

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:

Slide 118

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

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

SQL:

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

Trois Valeurs de Vérité (*)

Slide 119

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

1. `vrai AND inconnu = inconnu`
2. `faux AND inconnu = faux`
3. `inconnu AND inconnu = inconnu`
4. `vrai OR inconnu = vrai`
5. `faux OR inconnu = inconnu`
6. `inconnu OR inconnu = inconnu`
7. `NOT inconnu = inconnu`

Exemple (*)

Slide 120

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

SQL:

```
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 ?

Slide 121

Jointures dans SQL2 (*)

Opérations de Jointure (*)

Slide 122

SQL2	opération	**Algèbre**
R1 CROSS JOIN R2	produit cartesien	$R1 \times R2$
R1 JOIN R2 ON R1.A < R2.B	théta-jointure	$R1 \bowtie_{R1.A<R2.B} R2$
R1 NATURAL JOIN R2	jointure naturelle	$R1 \bowtie R2$

Jointure Naturelle: Exemple (*)

SCHEMA: **EMPLOYE**(EMPNO,ENOM,DEPNO,SAL), **DEPT**(DEPNO,DNOM)

REQUÊTE: *Noms des départements avec les noms de leurs employés.*

Slide 123

SQL:

```
SELECT DNOM, ENOM
FROM   DEPT NATURAL JOIN EMP
```

Note : Comme dans la définition algébrique, l'expression `DEPT NATURAL JOIN EMP` fait la jointure naturelle (sur l'attribut `DEPNO`) et l'attribut `DEPNO` n'apparaît qu'une seule fois dans le schéma du résultat.

Theta-Jointure: Exemple (*)

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

Slide 124

REQUÊTE: *Nom et salaire des employés gagnant plus que l'employé 12546*

SQL:

```
SELECT E1.ENOM, E1.SAL
FROM   EMPLOYE E1 JOIN EMPLOYE E2 ON E1.SAL > E2.SAL
WHERE  E2.EMPNO = 12546
```

Jointure Externe (*)

Slide 125

EMP

EMPNO	*DEPNO*	SAL
Tom	1	10000
Jim	2	20000
Karin	3	15000

DEPT

DEPNO	DNOM
1	Comm.
2	Adm.
4	Tech.

Jointure : les n-uplets qui ne peuvent pas être joints sont éliminés :

EMP NATURAL JOIN DEPT

Tom	1	10000	Comm.
Jim	2	20000	Adm.

Jointure Externe (*)

Jointure externe : les n-uplets qui ne peuvent pas être joints *ne sont pas éliminés*.

- On garde tous les n-uplets des deux rélations :

Slide 126

EMP NATURAL *FULL* OUTER JOIN DEPT

Tom	1	10000	Comm.
Jim	2	20000	Adm.
Karin	3	15000	NULL
NULL	4	NULL	Tech.

Slide 127

- On garde tous les n-uplets de la première rélation (gauche) :

EMP NATURAL *LEFT* OUTER JOIN DEPT

Tom	1	10000	Comm.
Jim	2	20000	Adm.
Karin	3	15000	NULL

- On garde tous les n-uplets de la deuxième rélation (droite) :

EMP NATURAL *RIGHT* OUTER JOIN DEPT

Tom	1	10000	Comm.
Jim	2	20000	Adm.
NULL	4	NULL	Tech.

Jointures Externes dans SQL2 (*)

Slide 128

- R1 NATURAL FULL OUTER JOIN R2 : Remplir R1.* et R2.*
- R1 NATURAL LEFT OUTER JOIN R2 : Remplir R2.*
- R1 NATURAL RIGHT OUTER JOIN R2 : Remplir R1.*

avec NULL quand nécessaire.

D'une manière similaire on peut définir des théta-jointures externes :

- R1 (FULL|LEFT|RIGHT) OUTER JOIN R2 ON prédicat

Slide 129

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*

Slide 130

ALGÈBRE:

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

SQL:

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

Slide 131

NOTE: L'union élimine les dupliqués. Pour garder les dupliqués on utilise l'opération `UNION ALL` : le résultat contient chaque n-uplet $a + b$ fois, où a et b est 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*

Slide 132

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

SQL:

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

NOTE: La différence élimine les dupliqués. Pour garder les dupliqués on utilise

Slide 133 l'opération `EXCEPT ALL` : le résultat contient chaque n-uplet $a - b$ fois, où a et b est le nombre d'occurrences du n-uplet dans la première et la deuxième requête.

Intersection

L'intersection ne fait pas partie du standard.

EMPLOYE(EMPNO,ENOM,DEPTNO,SAL)
DEPARTEMENT(DEPTNO,DNOM,LOC)

Slide 134 REQUÊTE: *Départements ayant des employés qui gagnent plus que 20000F et qui se trouvent à Paris*

ALGÈBRE :

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

SQL:

Slide 135

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

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

Slide 136

Imbrication des Requêtes en SQL

Requêtes imbriquées simples

La Jointure s'exprime par deux blocs SFW imbriqués

Soit le schéma de relations

Slide 137 **COMMANDES**(NUM,CNOM,PNOM,QUANTITE)
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:

Slide 138

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

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 139

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 140

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 141

```
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 ...) est vraie ssi la comparaison $f \theta v$ est vraie au moins pour une valeur v du résultat du bloc (SELECT F FROM ...).

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 142

```
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 143

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

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

Soit le schéma de relations

EMPLOYE(EMPNO,ENOM,DEPNO,SAL)
DEPARTEMENT(DEPTNO,DNOM,LOC)

REQUÊTE: *Départements sans employés*

SQL:

Slide 144

```
SELECT DEPTNO
FROM   DEPARTEMENT
WHERE  DETPNO 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)

Slide 145

REQUÊTE: *Fournisseurs qui fournissent au moins un produit*

```
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 146

```
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 θ est un des opérateurs de comparaison $<, =, >, \ldots$

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

Slide 147

EXISTS (SELECT * FROM R1, . . . Rn WHERE p AND x θ Ri.y)

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

NOT EXISTS (SELECT * FROM R1, . . . Rn WHERE (p) AND NOT (x θ 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 148

```
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')
```

Slide 149

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:

```
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

Slide 150

Soit le schéma de relations

FOURNITURE(FNUM,PNUM,QUANTITE)
PRODUIT(PNUM,PNOM,PRIX)
FOURNISSEUR(FNUM,FNOM,STATUS,VILLE)

REQUÊTE: *Fournisseurs qui fournissent tous les produits*

ALGÈBRE:

$$\text{R1} := \pi_{FNUM,PNUM}(FOURNITURE) \div \pi_{PNUM}(PRODUIT)$$
$$\text{R2} := \pi_{FNOM}(FOURNISSEUR \bowtie R1)$$

SQL:

Slide 151

```
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 152

Fonctions de Calcul

COUNT, SUM, AVG, MIN, MAX

Slide 153

REQUÊTE: *Nombre de Fournisseurs de Paris*

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

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 dupliqués ni vérification des valeurs nulles. Dans le cas contraire on utilise la clause COUNT(UNIQUE ...).

Slide 154

REQUÊTE: *Quantité totale de Briques commandées*

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

REQUÊTE: *Coût moyen de Briques fournies*

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

ou

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

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

Slide 155

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

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

Slide 156

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

Slide 157

Opérations d'Agrégation

GROUP BY

Slide 158 REQUÊTE: *Nombre de fournisseurs par ville*

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

Slide 159

LA BASE ET LE RESULTAT :

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

VILLE	COUNT(FNOM)
PARIS	2
LYON	3

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

Slide 160

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

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

RÉSULTAT:

PNOM	AVG (PRIX)
BRIQUE	10.5
ARDOISE	9.8

NOTE: Les fonctions de calcul appliquées au résultat de régroupement 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 161

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

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

Slide 162

NOTE: La clause HAVING permet d'éliminer des partitionnements, comme la clause WHERE élimine des n-uplets du résultat d'une requête. Par exemple on garde les produits dont le nombre des fournisseurs est $>= 2$. De cette façon des conditions de sélection peuvent être appliquées avant le calcul d'agrégat (clause WHERE) mais aussi après (clause HAVING).

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*

Slide 163

```
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 164

```
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).

Slide 165

Récursion dans SQL3 (*)

Enfants (*)

Soit le schéma de relation **ENFANT**(NOMPAR,NOMENF)

Slide 166

REQUÊTE: *Les enfants de Charlemagne*

SQL:

```
SELECT NOMENF
  FROM ENFANT
 WHERE NOMPAR='Charlemagne';
```

Descendants (*)

Soit le schéma de relation **ENFANT**(NOMPAR,NOMENF).

REQUÊTE: *Les* **descendants** *de Charlemagne*

SQL:

Slide 167

```
WITH RECURSIVE DESCENDANT(NOMANC, NOMDESC) AS
      (SELECT NOMPAR, NOMENF FROM ENFANT)
  UNION
      (SELECT R1.NOMANC, R2.NOMDESC
         FROM DESCENDANT R1, DESCENDANT R2
        WHERE R1.NOMDESC=R2.NOMANC)
SELECT NOMDESC FROM DESCENDANT
 WHERE NOMANC='Charlemagne';
```

Slide 168

Mises à jour avec SQL

Création de Tables

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

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

Slide 169

```
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);
```

Destruction de Tables

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

Slide 170

```
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 $R(A_1, A_2, \ldots, A_n)$ **VALUES** $(v_1, v_2, \ldots v_n)$

Slide 171

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 172

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** $A_1 = v_1, A_2 = v_2, \ldots, A_n = v_n$
WHERE *condition*

Slide 173

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 174

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 175

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 fournit par le BHV :

Slide 176

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

Destruction du BHV :

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

Slide 177

CALCUL RELATIONNEL

Exemple : la base de données CINÉMA

Slide 178

Films

Titre	Metteur en Scene	Acteur
Jules et Jim	F. Truffaut	J. Moreau
Jules et Jim	F. Truffaut	O. Werner
Les Quatre Cent Coups	F. Truffaut	J.P. Leaud
Metropolis	F. Lang	B. Helm
Chimes at Midnight	O. Welles	J. Moreau

Lieu

Cinema	Adresse	No-Telephone
Rex	Bd Poissonniere	42 36 83 93
Champo	R. des Ecoles	43 54 51 60
Cinoche	R. de Conde	46 33 10 82

Slide 179

Pariscope

Cinema	Titre	Heure
Rex	Jules et Jim	18
Rex	Jules et Jim	20
Cinoche	Jules et Jim	20
Champo	Metropolis	18

Slide 180

Syntaxe

Symboles (c.a.d. l'alphabet du langage)

Slide 181

- Constantes: 'F. Truffaut', 'Jules et Jim', 18, 43 54 51 60,...
- Variables: x, y, z, ...
- Prédicats: *Movies*,*Location*, ...
- Comparateurs arithmétiques: =, $<$, $>$, $<=$, ...
- Connecteurs logiques : $\vee, \wedge, \neg$
- Quantificateurs : $\exists, \forall$
- Parenthèses : "(", ")"

Formules atomiques (atomes)

Slide 182

- p(x1,...,xn) est un atome où p est un symbole de prédicat, et xi, i $\in$ [1,n] est soit une variable, soit une constante.

  Exemples :
  - Films ('Jules et Jim', 'F.Truffaut', 'J. Moreau')
  - Films (x, 'Truffaut', y)
- xθy est un atome où x et y sont soit des variables soit des constantes et θ est l'un des 6 comparateurs arithmétiques.

  Exemples : 1 < 3, x < y, x < 8
- Toutes les occurrences de variables apparaissant dans une formule atomique sont dites **libres**

Formules bien formées

Slide 183

- si F est une formule avec x parmi ses variables libres, alors (∃x)F et (∀x)F sont des formules. Toutes les occurrences de x dans F sont alors dites *liées*.
  Exemples:
  - $(\exists x)(x < 3)$
  - $(\forall x)(x < 3)$
  - $(\exists x)(x < y)$
- si F1 et F2 sont des formules, alors F1 ∨ F2, F1 ∧ F2 et ¬ F1 sont des formules (les occurrences de variables de ces nouvelles formules sont libres ou liées si elles le sont dans F1, F2). Exemples:
  - $(x <= 3) \vee (x > 5)$
  - $(\exists x)(x < 3 \wedge x > 5)$
- si F est une formule, alors (F) est une formule;

Formes Abrégés

Slide 184

- Formes abrégées de quantificateurs:
  1. $\exists x1, x2, \ldots, xn$ est une abréviation de $\exists x1 \exists x2 \ldots \exists xn$,
  2. $\forall x1, x2, \ldots, xn$ est une abréviation de $\forall x1 \forall x2 \ldots \forall xn$.
- Ordre de précédence (priorité) entre les connecteurs et quantificateurs (du plus au moins prioritaire):
  1. $\neg, \forall, \exists$,
  2. $\wedge$,
  3. $\vee$.

  Par exemple, $(\forall x) \neg p(x, y) \vee q(y) \wedge r(z)$ est compris comme

$$((\forall x)(\neg p(x, y))) \vee (q(y) \wedge r(z)).$$

Requêtes

Requête : $\{x1, x2, \ldots, xn \mid F\}$ où F est une formule avec variables libres, x1,x2,. . . ,xn.

Slide 185 **Exemples de Requêtes** :

- $\{x \mid x < 3\}$
- $\{x \mid Film('Jules\ et\ Jim','Truffaut', x)\}$
- $\{x \mid (\exists y) Film(y,'Truffaut', x)\}$

Sémantique

Slide 186

Interprétation

Slide 187

- L'atome p(x1,...,xn) où p est un symbole de prédicat et où les xi i ∈ [1,n] sont des constantes, est vrai si [x1,...,xn] est un n-uplet de la relation p.
  Exemple : $Film('Jules\ et\ Jim','Truffaut','Moreau')$ est vrai ssi $['Jules\ et\ Jim','Truffaut','Moreau']$ est un n-uplet dans la table $Film$.
- L'interprétation de xθy est naturelle. Par exemple x<y est vrai ssi x est inférieur à y.

Interprétation

Slide 188

- (∃x)F est vraie s'il existe une instanciation de x (on remplace x par une constante) telle que si on substitue toutes les occurrences de x dans F par cette instanciation, alors F est vraie.
  Par exemple $(\exists x)Films(x,'Truffaut','Moreau')$ est vraie s'il existe un film dirigé par Truffaut dans lequel joue J. Moreau.
- L'interprétation de F1∨F2. ¬F,... est l'interprétation habituelle : F1∨F2 est vraie si F1 est vraie *ou* F2 est vraie.

Note: La disjonction et la quantification universelle peuvent être exprimées en utilisant la negation:

1. F1 ∨ F2 ≡ ¬(¬ F1 ∧¬ F2),
2. (∀ x) F ≡ (¬∃ x) ¬ F.

Interprétation d'une Requête

Slide 189

Requête : $\{x1, x2, \ldots, xn \mid F\}$ où F est une formule. Interprétation :

C'est l'ensemble des n-uplets $[a1, a2, \ldots, an]$ tels que F(a1,a2,. . . ,an) est vraie.

- $\{x \mid Film('Jules\ et\ Jim', 'Truffaut', x)\}$ retourne tous les acteurs du film 'Jules et Jim' de Truffaut.
- $\{x \mid (\exists y) Film(y, 'Truffaut', x)\}$ retourne tous les acteurs qui ont tourné avec Truffaut.

Un Peu de Théorie

Slide 190

Algèbre relationnelle vs Calcul relationnel sain

Slide 191

L'algèbre relationnelle et le calcul relationnel sain ont le même pouvoir d'expression.

Théorême 1 : Toute requête exprimable dans l'algèbre relationnelle est exprimable dans le calcul relationnel.

Formule saine : Formule dont le résultat est fini (le nb de n-uplets qui satisfont la formule est fini).

Exemple de formule non saine : $\{x, y | \neg p(x, y)\}$. Le resultat contient tous les nuplets qui ne sont pas dans la relation p.

Théorême 2 : Toute requête du calcul relationnel sain est exprimable en algèbre relationnelle.

Calcul relationnel domaine/n-uplet

Slide 192

Le calcul relationnel précédent est appelé **calcul relationnel domaine**.

Pour obtenir le **calcul relationnel n-uplet**, on remplace :

- x1,x2,. . . ,xn (où xi est une variable domaine) par la variable n-uplet t.
- L'attribut A de rang i (xi dans le calcul domaine) est noté t.A (voir exemples ci-dessous).

Theorême 3 : Le calcul relationnel n-uplet sain a le même pouvoir d'expression que l'algèbre relationnelle.

Exemples

Slide 193

Quelques Requêtes

Slide 194

1. Qui dirige Metropolis?
2. Adresse et numéro de téléphone du Studio?
3. Salles où on peut voir un film de Truffaut?
4. Adresses des cinémas montrant un film de Truffaut?
5. Quels films de Truffaut ne passent pas en ce moment?

Requête 1: Qui a dirigé le film Metropolis ?

SQL

```
select Metteur-en-Scene
  from Films
 where Titre = 'Metropolis';
```

Slide 195

Calcul relationnel n-uplet

$$\{x.Metteur\_en\_Scene \mid Films(x) \wedge x.Titre =' Metropolis'\}$$

Calcul relationnel domaine

$$\{d \mid (\exists x) Films('Metropolis', d, x)\}$$

Requête 2: Adresse et numéro de téléphone du Studio ?

Calcul relationnel n-uplet

Slide 196

$$\{x.Adresse, x.Numero\_Tel \mid Lieu(x) \wedge x.Cinema =' Studio'\}$$

Calcul relationnel domaine

$$\{ad, ph \mid Lieu('Studio', ad, ph)\}$$

Requête 3: Salles où on peut voir un film de Truffaut ?

Slide 197

Calcul relationnel n-uplet

$$\{x.Cinema \mid (\exists y)(Pariscope(x) \wedge Films(y) \wedge x.Titre = y.Titre \wedge y.Metteur\_en\_Scene =' Truffaut')\}$$

Calcul relationnel domaine

$$\{s \mid (\exists hor, act, titre)(Pariscope(s, titre, hor) \wedge Films(titre,' Truffaut', act))\}$$

Requête 4: Adresses des cinémas montrant un Truffaut ?

Slide 198

SQL

```
select L.Adresse
  from Films F, Lieu L, Pariscope P
 where F.Metteur-en-Scene = 'Truffaut'
   and P.Titre = F.Titre
   and L.Cinema = P.Cinema;
```

Calcul relationnel n-uplet

$$\{z.Adresse \mid (\exists x, y)(Films(x) \wedge Pariscope(y) \wedge Lieu(z) \wedge x.Metteur\_en\_Scene =' Truffaut' \wedge x.Titre = y.Titre \wedge y.Cinema = z.Cinema)\}$$

Requête 4: Adresses des cinémas montrant un Truffaut ?

Slide 199

Calcul relationnel domaine

$$\{a \mid (\exists t, cine, s, teleph, act)(Films(t, \text{“}Truffaut\text{”}, act) \wedge Pariscope(cine, t, s) \wedge Lieu(cine, a, teleph))\}$$

Requête 5: Quels films de Truffaut ne passent pas?

Slide 200

SQL

```
select Titre
  from Films F
 where F.Metteur en Scene = 'Truffaut'
   and not exists ( select *
                      from Pariscope P
                     where  P.Titre = F.Titre )
```

Calcul relationnel n-uplet

$$\{x.Titre \mid (Films(x) \wedge x.Metteur\_en\_Scene =' Truffaut') \wedge (\neg\exists y)(Pariscope(y) \wedge y.Titre = x.Titre)\}$$

Requête 5: Quels films de Truffaut ne passent pas?

Calcul relationnel domaine

Slide 201

$$\{x \mid (\exists y) Films(x, \text{“}Truffaut\text{”}, y) \wedge (\neg \exists z, w) Pariscope(z, x, w)\}$$

ou

$$\{x \mid (\exists y) Films(x, \text{“}Truffaut\text{”}, y) \wedge (\forall z, w) \neg Pariscope(z, x, w)\}$$

Slide 202

DÉPENDANCES FONCTIONNELLES

Exemples : Relation PERSONNE

PERSONNE	NOSS	NOM	VILLE
	123	toto	Paris
	324	mimi	Paris
	574	toto	Marseilles

Slide 203

Qu'est-ce qu'on peut dire sur la table PERSONNE?

- "Connaissant le numéro de sécurité sociale `NOSS`, je connais le `NOM`"
- L'attribut `NOSS` *détermine* l'attribut `NOM`
- L'attribut `NOM` *dépend fonctionnellement* de l'attribut `NOSS`
- Attention : Connaissant le `NOM` on ne connaît pas le `NOSS` : $\texttt{NOM} \not\rightarrow \texttt{NOSS}$

NOTATION : $\texttt{NOSS} \rightarrow \texttt{NOM}$

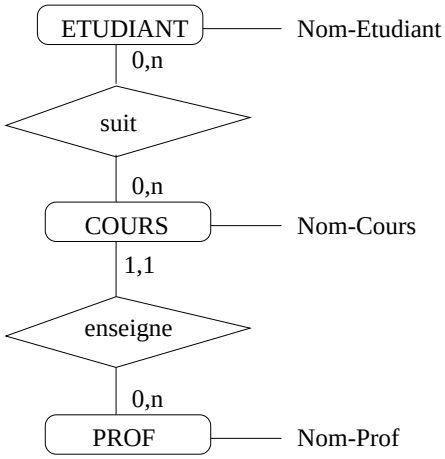
Relation COURS

COURS	NOM-COURS	PROF	ETUDIANT
	BDPI	MATHELOT	DUPONT
	BDPI	MATHELOT	DURAND
	BDPI	MATHELOT	LENORMAND
	IPA	HARDIN	DUPONT

Slide 204

- "Un Cours a un seul Professeur"
- $\texttt{NOM-COURS} \rightarrow \texttt{PROF}$
- 2 nuplets qui ont même valeur pour `NOM-COURS` ont même valeur pour `PROF`

Slide 205



Exemple de Clé

`PERSONNE (NOSS, NOM, PRENOM, ADRESSE)`

Slide 206

`NOSS` → `NOM`, `NOSS` → `PRENOM`, `NOSS` → `ADRESSE`

- Connaissant le numéro de sécurité sociale `NOSS`, on connaît tous les attributs
- 2 nuplets ayant le même `NOSS` sont *identiques*
- Le `NOSS` *identifie* le nuplet
- Le `NOSS` est une *clé*

`NOSS NOM` → `PRENOM`

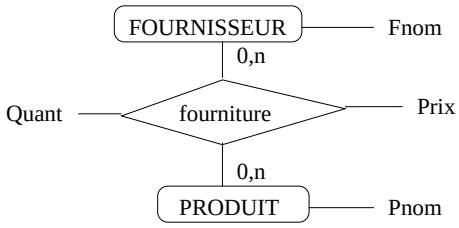
`NOSS NOM` est une *superclé* (`NOM` est redondant)

Relation FOURNITURE

`FOURNITURE (FNOM, PNOM, QUANT, PRIX)`

Slide 207

`FNOM PNOM` $\rightarrow$ `QUANT`
`FNOM PNOM` $\rightarrow$ `PRIX`
`FNOM PNOM` est une clé.



Relation ADRESSE

`ADRESSE (RUE, NUMERO, VILLE, CODE-P)`

Slide 208

Hypothèses simplificatrices

- Plusieurs codes pour une ville : Paris = 75001, 75143, ...
- Une seule ville pour un code : 75xxx = Paris
- Une adresse (rue + numéro + ville) n'appartient qu'à un seul code : 292, rue St Martin, Paris = 75143

`RUE NUMERO VILLE` $\rightarrow$ `CODE-P`
`CODE-P` $\rightarrow$ `VILLE`

Clés de la relation `ADRESSE` : `(RUE NUMERO CODE-P)` et `(RUE NUMERO VILLE)`

Définitions

a) Dépendance fonctionnelle

Soit R(U) un schéma de relation, r une relation de schéma R, $X \subset U$, $Y \subset U$ deux sous-ensembles d'attributs de R. La *dépendance fonctionnelle*

Slide 209

$$X \rightarrow Y$$

est vrai dans r, ssi (si et seulement si) tous les nuplets de r qui ont même valeur pour (tous) les attributs de X, ont même valeur pour (tous) les attributs de Y.

Exemple :

```
ADRESSE(RUE, NUMERO, VILLE, CODE-P)
X = CODE-P
Y = VILLE
```

b) Superclé

Soit R(U) un schéma et $X \subset U$ un sous ensemble d'attributs.
X est une *superclé* de r de schéma R, si $X \rightarrow U$.

Exemple :

NOSS NOM $\rightarrow$ NOSS NOM PRENOM ADRESSE

NOSS NOM est une superclé

Slide 210

c) Clé

X est une clé, si :

1. X est une superclé : $X \rightarrow U$
2. il n'existe pas $Y \subset X$, tel que $Y \rightarrow U$

Exemple :

RUE NUMERO VILLE $\rightarrow$ RUE NUMERO VILLE CODE-P

RUE NUMERO VILLE est une clé (pourquoi?)

Calcul d'une Clé: Exemple

COURS(Nomc,Heure,Salle,Prof)

$\mathcal{F} = \mathtt{N} \rightarrow \mathtt{HS},\ \mathtt{HS} \rightarrow \mathtt{P}$

Nous pouvons prouver à partir de $\mathcal{F}$ que si on connaît le nom de cours, on connaît aussi le nom du professeur (`Nomc` est une clé) :

Slide 211

1. $\mathtt{N} \rightarrow \mathtt{HS}$: deux nuplets qui partagent le nom de cours partagent également l'horaire et la salle
2. $\mathtt{HS} \rightarrow \mathtt{P}$: deux nuplets qui partagent l'horaire et la salle partagent également le nom du professeur
3. DF 1 et 2 impliquent, que deux nuplets qui partagent le nom de cours partagent également le nom du prof : $\mathtt{N} \rightarrow \mathtt{P}$

On peut définir des propriétés (règles) sur les DF qui permettent de déduire d'autres DF.

Propriétés des Dépendances Fonctionnelles

(Axiomes d'Armstrong)

1) Réflexivité

Slide 212

si $\mathtt{X} \subseteq \mathtt{Y}$ alors $\mathtt{Y} \rightarrow \mathtt{X}$ (pour tous les ensembles d'attributs X et Y)

Exemple :

$\mathtt{NOM\ VILLE} \rightarrow \mathtt{NOM}$

Trivial : "Deux personnes qui ont le même nom et habitent la même ville, ont le même nom."

2) Transitivité

si $X \rightarrow Y$ et $Y \rightarrow Z$, alors $X \rightarrow Z$

Exemple :

R(NOSS, CODE-P, VILLE)

{NOSS → CODE-P, CODE-P → VILLE} ⇒ NOSS → VILLE

Slide 213

"Si on connaît le code postale à partir du numéro de sécurité sociale et la ville à partir du code postale, on connaît la ville à partir du numéro de sécurité sociale."

3) Augmentation

$X \rightarrow Y \Rightarrow XZ \rightarrow YZ$

Exemple :

NOSS → CODE-P ⇒ NOSS VILLE → CODE-P VILLE

4) Union et décomposition

$\{ X \rightarrow A, X \rightarrow B \} \Leftrightarrow X \rightarrow AB$

Slide 214

5) Pseudotransitivité

$\{ X \rightarrow Y, WY \rightarrow Z \} \Rightarrow WX \rightarrow Z$

REMARQUE : Union, décomposition et pseudotransitivité peuvent être déduites des autres axiomes

Clôture d'un ensemble de dépendances fonctionnelles

De l'ensemble des dépendances fonctionnelles données par l'analyse du monde réel, en utilisant les propriétés ci-dessus, appelées *Axiomes d'Armstrong*, on peut en déduire d'autres :

Slide 215

Exemples :

1) R(A,B,C,D)

$\mathcal{F}$ = { A → B, B → C }

Par transitivité, on déduit A → C

Notation : $\mathcal{F} \models$ A → C

2) R(Cours, Prof, Heure, Salle, Eleve, Note)

$\mathcal{F}$ = { C → P, HS → C, HP → S, CE → N, HE → S }

Slide 216

$\mathcal{F} \models$ HS → P, HE → C,P,N

Donc HE est une **clé**. Pourquoi?

L'union de $\mathcal{F}$ et de toutes les dépendances ainsi déduites est appelée **clôture**, ou **couverture** de $\mathcal{F}$ et notée $\mathcal{F}^+$.

Couverture minimale

Slide 217

La couverture minimale $\mathcal{G}$ d'un ensemble $\mathcal{F}$ de dépendances fonctionnelles (DF) est un ensemble de DF tel que :

1. on peut déduire de $\mathcal{G}$ les mêmes DF que de $\mathcal{F}$: $\mathcal{G}^+ = \mathcal{F}^+$
2. Il n'y a *qu'un* attribut à droite dans toutes les DF de $\mathcal{G}$ (décomposition)
3. Toutes les dépendances sont utiles : si on enlève une quelconque, on ne peut obtenir $\mathcal{F}^+$.
4. Toutes les dépendances sont *élémentaires* : l'ensemble $\{\mathsf{A} \rightarrow \mathsf{C};\ \mathsf{AC} \rightarrow \mathsf{B}\}$ est redondant : on remplace $\mathsf{AC} \rightarrow \mathsf{B}$, qui n'est pas élémentaire, par $\mathsf{A} \rightarrow \mathsf{B}$.

Exemple

Slide 218

Dépendances fonctionnelles	**Clôture minimale**
$\mathsf{AB} \rightarrow \mathsf{C}$	$\mathsf{AB} \rightarrow \mathsf{C}$
$\mathsf{C} \rightarrow \mathsf{A}$	$\mathsf{C} \rightarrow \mathsf{A}$
$\mathsf{BC} \rightarrow \mathsf{D}$	$\mathsf{BC} \rightarrow \mathsf{D}$
$\mathsf{ACD} \rightarrow \mathsf{B}$	$\mathsf{CD} \rightarrow \mathsf{B}$
$\mathsf{D} \rightarrow \mathsf{EG}$	$\mathsf{D} \rightarrow \mathsf{E}$
$\mathsf{BE} \rightarrow \mathsf{C}$	$\mathsf{D} \rightarrow \mathsf{G}$
$\mathsf{CE} \rightarrow \mathsf{AG}$	$\mathsf{BE} \rightarrow \mathsf{C}$
	$\mathsf{CE} \rightarrow \mathsf{G}$

$\mathsf{ACD} \rightarrow \mathsf{B}$ n'est pas élémentaire, pourquoi?
$\mathsf{CE} \rightarrow \mathsf{A}$ a disparu, pourquoi?

Réponse à la question :

ACD → B n'est pas élémentaire, pourquoi?

C → C (réflexivité); C → C et C → C |= C → CC (union)

C → A |= CCD → ACD par *augmentation*

Slide 219

CCD → ACD |= CD→ ACD par pseudotransitivité avec C → CC

CD → ACD et ACD → B |= CD → B par *transitivité*

Autre démonstration

Par pseudotransitivité, { C → A, ACD → B} |= CCD → B ; CCD → B et C → CC |= CD → B (voir plus haut).

Clôture minimale

Concrètement, la clôture minimale

Slide 220

- soit est donnée directement par l'analyse du monde réel
- soit est triviale à obtenir car en général
  - toutes le DF sont utiles et élémentaires
  - il suffit de décomposer les membres droits

Nouvelle définition d'une Superclé

Soit R(U) un schéma de relation, $\mathcal{F}$ un ensemble de dépendances fonctionnelles et X $\subseteq$ U un ensemble d'attributs. X est une superclé de R si pour tout A $\in$ U

Slide 221

$$\mathcal{F} \models X \rightarrow A$$

ou encore si:

$$X \rightarrow A \in \mathcal{F}^+$$

Calculer $\mathcal{F}^+$ à partir de $\mathcal{F}$ peut être très long.

Par contre montrer que $\mathcal{F} \models X \rightarrow A$ est facile et rapide.

Calcul de la clé d'une relation

Soit R(U) un schéma de relation et X $\subseteq$ U un ensemble d'attributs.

- On définit X^+ comme ensemble des attributs A tels que $\mathcal{F} \models X \rightarrow A$

Slide 222

- Si A appartient à X^+, alors par définition $\mathcal{F} \models X \rightarrow A$
- X^+ est l'ensemble des attributs fonctionnellement dépendants de X.

Pour calculer une clé on utilise l'algorithme suivant :

1. On cherche un X tel que $X^+ = U \Rightarrow$ X est une superclé
2. X est une clé, s'il n'existe pas Y $\subset$ X tel que $Y^+ = U$

Calcul de X^+

Étape 1: On part de X

Pour toute Y $\rightarrow$ A, telle que Y $\subseteq$ X, Y $\rightarrow$ A $\in \mathcal{F}$, on rajoute A à X.
On obtient X^1.

Slide 223

Étape i: On part de X^{i-1}

Pour toute Y $\rightarrow$ A, telle que Y $\subseteq$ X^{i-1}, Y $\rightarrow$ A $\in \mathcal{F}$, on rajoute A à X^{i-1}.
On obtient X^i.

On s'arrête quand on ne trouve plus de nouvelle DF :

$$X^+ = X^{i+1} = X^i$$

Exemples

1. Montrer que HE est une clé pour R(CHENSP) avec l'ensemble $\mathcal{F}$ de DF donné ci-dessus
2. Pour la relation ADRESSE(VILLE, RUE, NUMERO, CODE_P) ci-dessus, montrer que VILLE RUE NUMERO est une clé. Quelle est l'autre?

Slide 224

Slide 225

ANOMALIES DE MISE À JOUR

Exemple

Slide 226

Soit le schéma S1 :

```
FOURNISSEUR(FNOM, FADRESSE)
FOURNITURE(FNOM, PNOM, PRIX)
```

et l'ensemble de DF : $\mathcal{F} = \{\texttt{FNOM} \rightarrow \texttt{FADRESSE}, \texttt{(FNOM PNOM)} \rightarrow \texttt{PRIX}\}$

Supposons qu'on remplace S1 par le schéma S2 :

```
R(FNOM, FADRESSE, PNOM, PRIX)
```

Anomalies

R (FNOM, FADRESSE, PNOM, PRIX)

$\mathcal{F} = \{$FNOM $\rightarrow$ FADRESSE, (NOM PNOM) $\rightarrow$ PRIX$\}$

Quelle est la clé de R?

Slide 227

TOTO	LYON	BRIQUES	1000
DUPONT	ROUEN	BRIQUES	900
TOTO	LYON	BETON	400

1) REDONDANCE : l'adresse d'une personne apparaît plusieurs fois.

2) MAJ : si on modifie l'adresse dans un nuplet, il faut le faire dans les autres.

3) SUPPRESSION : si DUPONT ne fournit plus de BRIQUES, on supprime le 2e nuplet, on perd toute info sur DUPONT

4) INSERTION : on ne peut insérer un nouveau fournisseur et son adresse, si on ne connaît pas au moins un produit qu'il fournit

Slide 228

TOTO	PARIS	BRIQUES	1000
TOTO	LYON	BETON	400
DURAND	NICE		

$\Longrightarrow$ LE SCHÉMA INITIAL S1 EST "MEILLEUR"

Contraintes d'intégrité

La liste des attributs est insuffisante pour décrire la sémantique du monde réel.

Il existe plusieurs types de contraintes sur les nuplets :

Slide 229

1. dépendances (fonctionnelles, multivaluées, de jointure, etc.)
2. contraintes qui dépendent du domaine d'un attribut : Taille $<$ 2m10, année $<$ 2000
3. etc.

Ce sont les dépendances qui permettent la *conception d'un bon schéma*, c.a.d. la décomposition en "bonnes" relations.

Qualités d'un bon schéma

1. Éviter les anomalies $\Longrightarrow$ décomposition
2. La décomposition doit conserver la même information
   La jointure d'une relation
   `f1` *de schéma* `FOURNISSEUR(FNOM, FADRESSE)`
   Slide 230
   et d'une relation
   `f2` *de schéma* `FOURNITURE(FNOM, PNOM, PRIX)`
   obtenues par décomposition d'une relation
   `r` *de schéma* `R(FNOM, FADRESSE, PNOM, PRIX)`
   doit redonner `r`.
3. La décomposition doit conserver les mêmes contraintes (DF). La décomposition de `R` en `R1(FNOM, FADRESSE, PRIX)` et `R2(PNOM, PRIX)` ne préserve pas les DF. Pourquoi?

Slide 231

FORMES NORMALES ET DÉCOMPOSITION

Relation en première forme normale (1FN)

Slide 232

- Tous les attributs sont atomiques (*élémentaires*)
- Relations telle qu'on les connaît.
- **Relation non normalisée, non en 1e FN** :
  Certains attributs sont des ensembles de valeurs, des relations elles même

Relation 1FN

```
NOTES ( COURS     ETUDIANT    NOTE )
        BDB        Toto         15
        BDB        Lulu         17
        BDB        Lili          0
        ARCHI      Lili         20
        ARCHI      Toto          0
```

Slide 233

Relation N1FN

```
NOTES ( COURS     PERF (ETUDIANT  NOTE) )
        BDB             Toto        15
                        Lulu        17
                        Lili         0
        ARCHI           Lili        20
                        Toto         0
```

Relation en 3e forme normale

Slide 234

- **2e forme normale** :
  Purement historique
- **3e forme normale : 3FN**
  - évite la plupart des anomalies

  le but du jeu : décomposer une relation (1FN) en un ensemble de relations 3FN

3FN : Première Définition

Soit un schéma (R, F).

Slide 235

On suppose que $\mathcal{F}$ est une couverture minimale.

Définition : R est en 3FN, si quelle que soit la DF X $\rightarrow$ A de $\mathcal{F}$,

- soit X est une clé
- soit A appartient à l'une des clés

Exemples: 3e Forme Normale

1) Poste(Ville, Rue, Code)

$\mathcal{F}$ = { VR $\rightarrow$ C, C $\rightarrow$ V }

Clés : VR, RC

Slide 236

R est en 3FN.

2) FOURNITURE(NOMF, ADR, NOMP, PRIX)

$\mathcal{F}$ = { NOMF $\rightarrow$ ADR, NOMF,NOMP $\rightarrow$ PRIX }

Clé : (NOMF NOMP)

FOURNITURE n'est pas en 3FN.

3) PLANNING(Cours, Heure, Salle)

$\mathcal{F}$ = { SH $\rightarrow$ C, C $\rightarrow$ S }

Clés : SH, CH

PLANNING est en 3FN.

Slide 237

4) R(A, B, C, D)

$\mathcal{F}$ = { AB $\rightarrow$ C, B $\rightarrow$ D, BC $\rightarrow$ A }

Clés : AB, BC

R n'est pas en 3FN.

3FN : Deuxième Définition

Remarque : Il n'est pas nécessaire que $\mathcal{F}$ soit une couverture minimale.

Il suffit que pour toute DF X $\rightarrow$ A de $\mathcal{F}^+$,

- A soit un seul attribut,
- A ne soit pas l'un des attributs de X

Slide 238

La définition d'un schéma 3FN devient :

Définition : Une relation R est en 3FN, si quelle que soit la DF X $\rightarrow$ A de $\mathcal{F}^+$ qui satisfait les conditions précédentes,

- soit X est une *super*clé
- soit A appartient à l'une des clés

En fait, il n'est pas nécessaire de vérifier *toutes* les DF de $\mathcal{F}^+$.

Il suffit de vérifier celles de $\mathcal{F}$!

Remarque (Suite) :

Pourquoi *super*clé dans la condition 1 ?

Slide 239

Avec les conditions 1 et 2 plus faibles que celles d'une couverture minimale, il peut y avoir des conditions non *élémentaires* :

si $X \rightarrow A$ est une condition non élémentaire
et si $X^+ = U$, alors X est une *super*clé.

Remarque (fin) :

$R(A,B,C,D)$

Slide 240

$\mathcal{F} = \{ AB \rightarrow C, B \rightarrow D, D \rightarrow B, B \rightarrow A \}$

$\mathcal{F}$ n'est pas une couverture minimale, pourquoi ?

$(R, \mathcal{F})$ est en 3FN, pourquoi?

Décomposition sans perte d'information (SPI)

Slide 241

Exemple

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

et $\mathcal{F}$ = { A → B }

Décomposons en :

R1	A	B
	a	b
	c	b

R2	B	C
	b	c
	b	a
	b	d

Slide 242

R1 = $\pi_{A,B}$(R)
R2 = $\pi_{B,C}$(R)
R′ = R1 ⋈ R2 ≠ R :

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

La décomposition de R en R1 et R2 est avec perte d'informations.

La jointure crée des nuplets qui n'existaient pas dans R.

Décomposons R′ maintenant en :

```
R1′( A   B )        R2′ ( A    C )
     a   b                 a    c
     c   b                 a    a
                           c    d
```

R″ = R1′ ⋈ R2′ = R′ :

Slide 243

```
R′′  ( A   B   C )
       a   b   c
       a   b   a
       c   b   d
```

Cette décomposition est *sans perte d'information* (SPI)

Il faut qu'après la jointure, on retrouve la même information qu'avant la décomposition.

Définition

Une décomposition de R en R1, R2, ..., Rk par rapport à un ensemble de DF $\mathcal{F}$ est SPI (sans perte d'information), ssi quelle que soit r de schéma R *satisfaisant* $\mathcal{F}$, on a :

$$r = \pi_{R1}(r) \bowtie \pi_{R2}(r) \ldots \bowtie \pi_{Rk}(r)$$

Théorème :

Slide 244

Si (R1, R2) est une décomposition de R et $\mathcal{F}$ un ensemble de DF, alors (R1, R2) est SPI par rapport à $\mathcal{F}$, ssi :

$$R1 \cap R2 \rightarrow R1 - R2$$

ou

$$R1 \cap R2 \rightarrow R2 - R1$$

est une dépendance de $\mathcal{F}^+$.

Exemples

R (A, B, C)

$\mathcal{F} = \{ A \rightarrow B \}$

Slide 245

1) R1(A, B), R2(B, C)

$AB \cap BC = B$

$AB - BC = A$

$BC - AB = C$

Il n'y a ni la DF $B \rightarrow A$, ni la DF $B \rightarrow C$ dans $\mathcal{F}^+$

$\Rightarrow$ **Décomposition avec perte d'information**

2) R1(A, B), R3(A, C)

$AB \cap AC = A$

Slide 246

$AB - AC = B$

$A \rightarrow B$ est dans $\mathcal{F}$ ($\mathcal{F}^+$).

$\Rightarrow$ **Décomposition sans perte d'information**

Décomposition qui préserve les dépendances fonctionnelles

Définitions

Slide 247

1. Projection d'un ensemble de dépendances sur $\mathsf{Z} \subset \mathsf{U}$
   $\pi_Z(\mathcal{F}) = \{X \rightarrow Y \in \mathcal{F}^+ \mid XY \subseteq Z\}$
   Exemple : $\mathsf{R(A,B,C,D)}$, $\mathcal{F}$ = { $\mathsf{AB} \rightarrow \mathsf{C}$, $\mathsf{C} \rightarrow \mathsf{A}$, $\mathsf{A} \rightarrow \mathsf{D}$ }
   $\pi_{ABC}(\mathcal{F})$ = { $\mathsf{AB} \rightarrow \mathsf{C}$, $\mathsf{C} \rightarrow \mathsf{A}$ }
2. Décomposition qui préserve les DF de $\mathcal{F}$
   Soit Δ = $\mathsf{(R1,...,Rk)}$ une décomposition, et $\mathcal{F}$ un ensemble de DF.
   Δ préserve les DF de $\mathcal{F}$, si on peut retrouver toutes les DF de $\mathcal{F}^+$ à partir de l'union $\mathcal{G}$ de toutes les DF projetées de $\mathcal{F}$ dans $\pi_{R1}(\mathcal{F}), \ldots, \pi_{Rk}(\mathcal{F})$:
   $\mathcal{G}^+ = \mathcal{F}^+$

Exemples

R(A,B,C,D)

Slide 248

$\mathcal{F}$ = { $\mathsf{AB} \rightarrow \mathsf{C}$, $\mathsf{C} \rightarrow \mathsf{A}$, $\mathsf{A} \rightarrow \mathsf{D}$ }

Δ = (ABC, BD) ne préserve pas les DF de $\mathcal{F}$

Δ = (ABC, AD) préserve les DF de $\mathcal{F}$

R(A,B,C)

$\mathcal{F}$ = { $\mathsf{A} \rightarrow \mathsf{B}$, $\mathsf{B} \rightarrow \mathsf{A}$, $\mathsf{A} \rightarrow \mathsf{C}$ }

Δ = (AB, BC) préserve les DF de $\mathcal{F}$

R(A, B, C, D)

$\mathcal{F} = \{ \mathtt{A} \rightarrow \mathtt{B}, \mathtt{B} \rightarrow \mathtt{C}, \mathtt{AB} \rightarrow \mathtt{D} \}$

La décomposition

Slide 249

```
R1(AC)
R2(AB)
R3(CD)
```

ne préserve pas les DF de $\mathcal{F}$. Pourquoi?

Décomposition d'une relation en relations 3FN

Slide 250

Étant donné un schéma `(R, F)` non en 3FN, i.e. avec des anomalies, on veut une décomposition de `R` :

1. en relations 3FN
2. qui soit SPI
3. qui préserve les DF de $\mathcal{F}$

Remarque :

- une décomposition SPI ne préserve pas forcément les DF et inversement

Slide 251

- le résultat ne donne pas forcément des relations 3FN

Théorème : Toute relation en 1FN possède une décomposition en relations 3FN qui soit SPI et préserve les dépendances fonctionnelles.

Algorithme de décomposition

On suppose que $\mathcal{F}$ est une couverture minimale

1. Pour chaque $\mathtt{X} \rightarrow \mathtt{A} \in \mathcal{F}$, créer une relation de schéma `(XA)`.

Slide 252

2. Si aucune des clés n'est contenue dans l'un des schémas créés dans l'étape 1, rajouter une relation de schéma `(Y)`, où `Y` est une clé.
3. Si après l'étape 1, il existe une relation `R1` dont le schéma `(X1A1)` est contenu dans le schéma `(X2A2)` d'une autre relation `R2`, supprimer la relation `R1`.
4. Remplacer les relations `(XA1),...,(XAk)` (correspondant à des dépendances ayant même membre gauche) par une relation unique : `(XA1...Ak)`.

Exemples

1) R(A,B,C,D)

$\mathcal{F}$ = { AB → C, B → D, C → A }

Clés : AB, BC

Slide 253

- Étape 1 : R1(ABC) R2(BD) R3(CA)
- Étape 2 : Pas la peine de rajouter une relation de schéma la clé AB :

  AB est contenue dans R1
- Étape 3 : Supprimer R3 : CA ⊂ ABC

Bonne décomposition : R1(ABC) R2(BD)

On peut vérifier que R1 et R2 sont en 3eFN.

2) R(A,B,C,D,E)

$\mathcal{F}$ = { AB → C, C → D, C → A } **Clés** : ABE, BCE

- Étape 1 : R1(ABC) R2(CD) R3(CA)
- Étape 2 : On rajoute une relation de schéma pour la clé ABE : R4(ABE)
- Étape 3 : Supprimer R3 : CA ⊂ ABC

Slide 254

Bonne décomposition : R1(ABC) R2(CD) R4(ABE)

R3 n'a pas de dépendance. Quelles sont les dépendances des autres?

Autre solution :

- Étape 4 : On remplace R2 et R3 de l'étape 1 par une relation de schéma (CAD)

Autre bonne décomposition : R1(ABC) R2′(CAD) R4(ABE)

Que se passe-t-il si on avait choisi la clé CBE?

3) R(A,B,C,D)

$\mathcal{F}$ = { AB → C, C → D, C → A, AB → D }

Clés : BA, BC

La relation n'est pas en 3e Forme Normale. Pourquoi?

Slide 255

- Étape 1 : R1(ABC) R2(CD) R3(CA) R4(ABD)
- Étape 2 : on ne rajoute pas de relation : clé AB ⊆ R1(ABC)
- Étape 3 : Supprimer R3 : CA ⊆ ABC
- Étape 4 : On remplace R1 et R4 par R5(ABCD) ⇒ on peut supprimer R2.

Décomposition : R5(ABCD)

Cette décomposition n'est pas en 3e Forme Normale. Où est le problème?

Forme Normale Boyce-Codd (BCNF)

Des anomalies subsistent en 3FN.

Exemple : Poste(Ville,Rue,Code), $\mathcal{F}$ = { VR → C, C → V }

Slide 256

Clés : VR, RC

```
Poste  ( Ville       Rue            Code  )
         Paris       St Michel      75005
         Paris       Champollion    75005
```

⇒ Redondance entre le code et la ville.

Définition : Une relation est en forme normale de *Boyce-Codd* (BCNF), si quelle que soit la dépendance de $\mathcal{F}$, le membre de gauche est une clé.

Intérêt : On a éliminé toutes les anomalies

Slide 257

Remarque : Toute relation BCNF est en 3FN

Malheureusement, il n'existe pas toujours une décomposition en relations BCNF

- qui soit SPI
- qui préserve les DF

L'exemple de la poste

`Poste(Ville,Rue,Code)`, $\mathcal{F}$ = { `VR` $\rightarrow$ `C`, `C` $\rightarrow$ `V` }

Slide 258

Clés : `VR, RC`

`R` est 3FN mais n'est pas BCNF (dans `C` $\rightarrow$ `V`, `C` n'est pas une clé)

```
Poste  ( Ville       Rue           Code  )
         Sevres      de Gaulle     92310
         Chaville    de Gaulle     92370
```

La décomposition `R1(Ville,Code), R2(Rue,Code)` évite la redondance `Ville,Code`, elle est SPI, mais elle ne préserve pas la dépendance `VR` → `C`

```
R1 ( Ville     Code  )     R2 ( Rue       Code  )
     Sevres    92310            de Gaulle 92310
     Chaville  92370            de Gaulle 92370
```

Slide 259

L'insertion `Sevres de Gaulle 92190`, c.a.d. `Sevres 92190` et `de Gaulle 92190` respecte `C` → `V` mais ne respecte plus `VR` → `C`

```
R1 ( Ville     Code  )     R2 ( Rue       Code  )
     Sevres    92310           de Gaulle  92310
     Chaville  92370           de Gaulle  92370
     Sevres    92190           de Gaulle  92190
```

Slide 260

ORGANISATION PHYSIQUE DES DONNEES

Organisation des données en mémoire secondaire

Slide 261

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

Les fichiers

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

Slide 262

- 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**.

Les articles

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

Slide 263

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

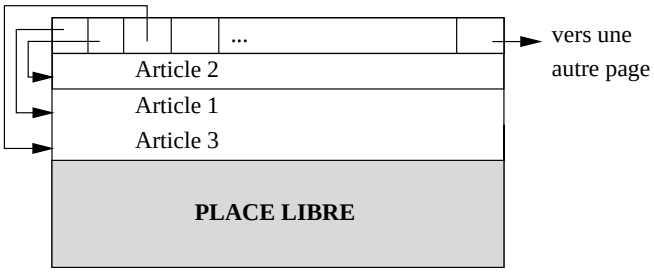
Slide 264

L'adresse d'un article est constituée de

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.

Structure interne d'une page

Slide 265



Opérations sur les fichiers

Slide 266

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.

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 267

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 268

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

Organisation séquentielle

Slide 269

- **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

Slide 270

Nombre moyen de lectures/écritures sur disque,
Fichier de n pages.

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

$\Rightarrow$ organisation utilisée pour les fichiers de petite taille.

Fichiers séquentiels triés

Une première amélioration consiste à **trier** le fichier sur sa clé d'accès. On peut alors effectuer une recherche par **dichotomie** :

Slide 271

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**. Le(s) champ(s) sur le(s)quel(s) le fichier est trié est appelé **clé**.

Slide 272

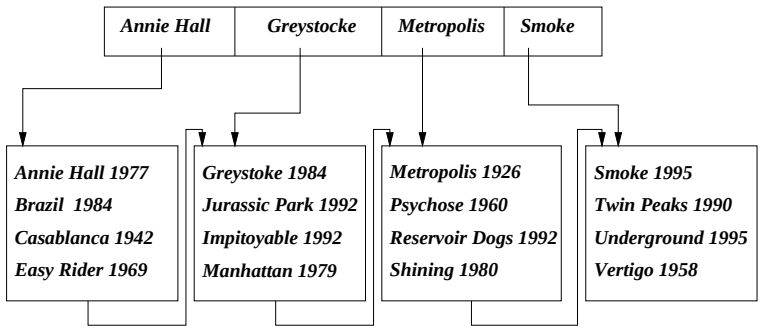
Un index est un second fichier possédant les caractéristiques suivantes :

1. Les articles sont des couples (Valeur de clé, Adresse de page)
2. Une occurence (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.

Exemple

Slide 273



Recherche avec un index

Slide 274

Un index permet d'accélérer les recherches : 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 $v_2 < v_1$.
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

Slide 275

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.

- 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**

Autres opérations : insertion

Slide 276

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 :

1. Il y 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. Plusieurs solutions, notamment : créer une **page de débordement**.

NB : il faut éventuellement réorganiser l'index.

Autres opérations : destructions et mises-à-jour

Slide 277

Relativement facile en général :

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 couteux.

Séquentiel indexé : définition

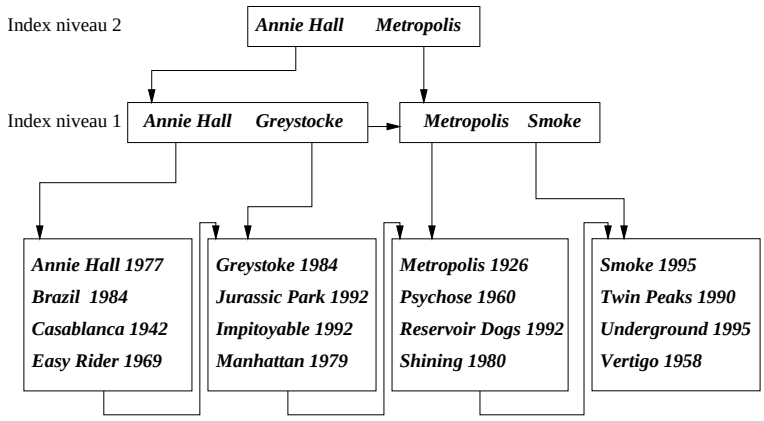
Slide 278

Un index est un fichier : on peut lui-même l'indexer :

1. On trie le fichier sur la clé.
2. On répartit les articles triés dans n pages, en laissant de la place libre dans chaque page.
3. On constitue un index à plusieurs niveaux sur la clé.

⇒ on obtient un **arbre** dont les feuilles constituent le fichier et les noeuds internes l'index.

Slide 279



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.

Slide 280

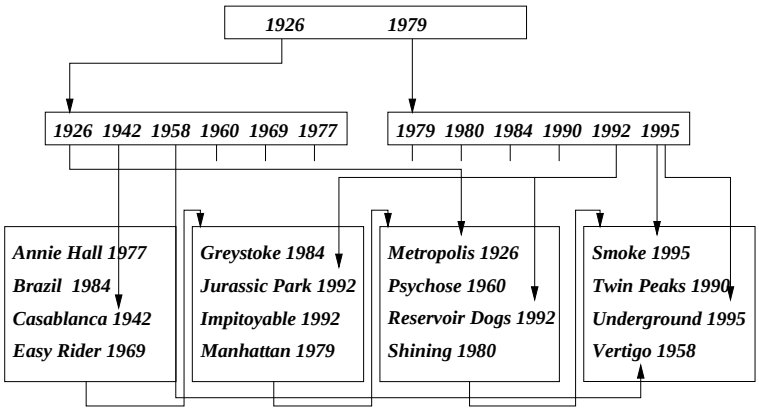
Un index est **dense** ssi il existe une valeur de clé dans l'index pour chaque article dans le fichier F.

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.

Exemple d'index dense

Slide 281



Inconvénients du séquentiel indexé

Slide 282

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

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

Slide 283

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.

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

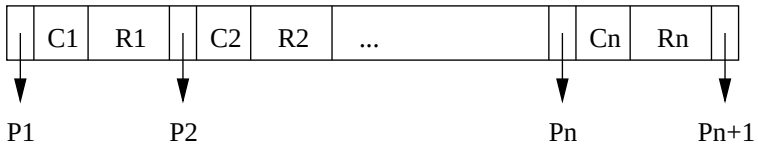
Slide 284

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

1. Chaque noeud est une page contenant au moins k et au plus $2k$ articles, $k \in N$.
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.

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

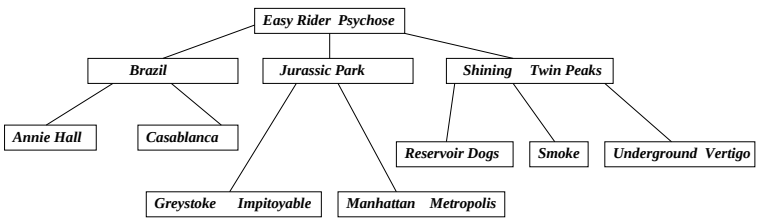
Slide 285



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'un arbre-B

Slide 286



Recherche dans un arbre-B

Rechercher les articles de clé C. A partir de la racine, appliquer récursivement l'algorithme suivant :

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

Slide 287

1. Chercher C dans la page courante. Si C y est, accéder aux valeurs des autres champs, Fin.
2. Sinon, Si $C < C_1$ (ou $C > C_n$), on continue la recherche avec le noeud référencé par P_1 (ou P_{n+1}).
3. Sinon, il existe $i \in [1, k[$ tel que $C_i < C < C_{i+1}$, 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 l'article doit prendre place et on l'y insère. Si la page p déborde (elle contient $2k + 1$ éléments) :

Slide 288

1. On alloue une nouvelle page p'.
2. On place les k 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.

Destruction dans un arbre-B d'ordre k

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

Slide 289

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 $ReservoirDogs$

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

Slide 290

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 $Psychose$, remplaçé par $Metropolis$
3. On vient de retirer un article dans une feuille : si elle contient moins de k éléments, on procède comme indiqué précédemment.

$\Rightarrow$ toute destruction a donc un effet seulement **local**.

Quelques mesures pour l'arbre-B

Slide 291

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

$$\log_{2k+1}(n+1) \leq h \leq \log_{k+1}(\tfrac{n+1}{2})$$

Exemple pour $k = 100$:

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

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

Variante de l'arbre-B

Slide 292

1. l'arbre B contient à la fois l'index et le fichier indexé.
2. Si la taille d'un article est grande, chaque noeud en contient peu, ce qui augmente la hauteur de l'arbre
3. on peut alors séparer l'index (Arbre B) du fichier : stocker les articles dans un fichier F, remplacer l'article R_i dans les pages de l'arbre par un pointeur vers l'article dans F.

L’arbre B+

Inconvénient de l’arbre B (et de ses variantes:

Slide 293

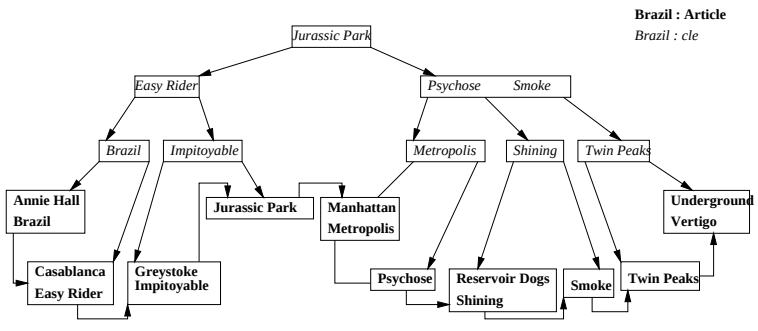
Les recherches sur des intervalles de valeurs sont complexes.

D’où l’arbre-B+ : seules les feuilles de l’arbre pointent sur les articles du fichier. De plus ces feuilles sont chaînées entre elles.

Variante: les feuilles contiennent les articles.

Exemple d’un arbre-B+

Slide 294



Hachage

Accès direct à la page contenant l'article recherché :

Slide 295

1. On estime le nombre N de pages qu'il faut allouer au fichier.
2. **fonction de hachage** H : à toute valeur de la clé de domaine V associe un nombre entre 0 et $N-1$.
$$H : V \rightarrow \{0, 1, \ldots, N-1\}$$
3. On range dans la page de numéro i tous les articles dont la clé c est telle que $H(c) = i$.

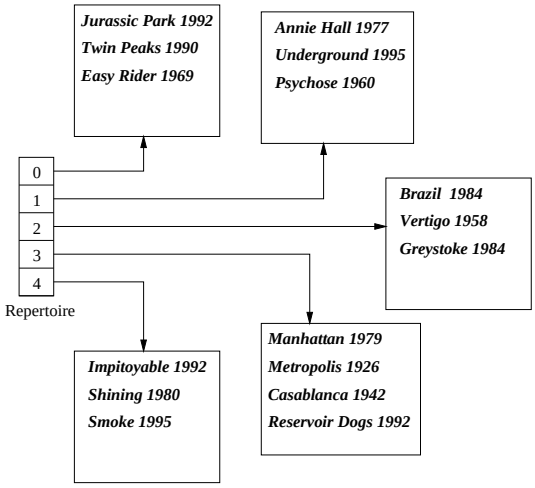
Exemple : hachage sur le fichier $Films$

On suppose qu'une page contient 4 articles :

Slide 296

1. On alloue 5 pages au fichier.
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 297



Remarques

1. Le nombre $H(c) = i$ n'est pas une adresse de page, mais l'indice d'1 table ou "répertoire" R. $R(i)$ contient l'adresse de la page associée à i

Slide 298

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 $\{0, \ldots N-1\}$
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.

Hachage : recherche

Etant donné une valeur de clé v :

Slide 299

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.

$\Rightarrow$ **donc une recherche ne coûte qu'une seule lecture**.

Hachage : insertion

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

Si la page p est pleine, il faut :

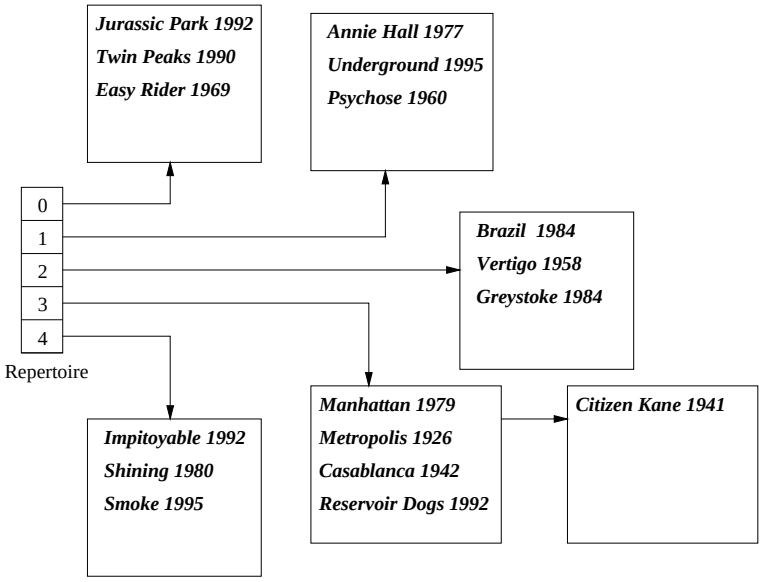
Slide 300

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

$\Rightarrow$ 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 301



Hachage : avantages et inconvénients

Slide 302

Intérêt du hachage :

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

En revanche :

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

Comparatif

Slide 303

Organisation	Coût	Avantages	Inconvénients
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 304

OPTIMISATION

Pourquoi l'optimisation ?

Slide 305

Les langages de requêtes de haut niveau comme SQL sont **déclaratifs**. L'utilisateur :

1. indique ce qu'il veut obtenir.
2. n'indique pas **comment** l'obtenir.

Donc le système doit faire le reste :

1. Déterminer le (ou les) chemin(s) d'accès aux données, les stratégies d'évaluation de la requête
2. **Choisir la meilleure**. Ou une des meilleures ...

L'optimisation sur un exemple

Slide 306

Considérons le schéma :

$CINEMA(Cinéma, Adresse, Gérant)$
$SALLE(Cinéma, NoSalle, Capacité)$

avec les hypothèses :

1. Il y a 300 n-uplets dans CINEMA, occupant 30 pages.
2. Il y a 1200 n-uplets dans SALLE, occupant 120 pages.

Expression d'une requête

On considère la requête :

$$Adresse\ des\ cinémas\ ayant\ des\ salles\ de\ plus\ de\ 150\ places$$

Slide 307

En SQL, cette requête s'exprime de la manière suivante :

```
SELECT Adresse
FROM   CINEMA, SALLE
WHERE  capacité > 150
AND    CINEMA.cinéma = Salle.cinéma
```

En algèbre relationnelle

Slide 308

Traduit en algèbre, on a plusieurs possibilités. En voici deux :

1. $\pi_{Cinéma}(\sigma_{Capacité>150}(CINEMA \bowtie SALLE))$
2. $\pi_{Cinéma}(CINEMA \bowtie \sigma_{Capacité>150}(SALLE))$

Soit une jointure suivie d'une sélection, ou l'inverse.

Evaluation des coûts

Slide 309

On suppose qu'il n'y a que 5 % de salles de plus de 150 places.

1. Jointure : on lit 3 600 pages (120x30); comme on projète sur tous les attributs de Cinéma, on obtient 5 % de 120 pages, soit 6 pages. Nombre d'E/S : 3 600 + 120x2 + 6 = 3 846.
2. Sélection : on lit 120 pages et on obtient 6 pages. Jointure : on lit 180 pages (6x30) et on obtient 6 pages. Nombre d'E/S : 120 + 6 + 180 + 6 = 312.

$\Rightarrow$ la deuxième stratégie est de loin la meilleure !

Optimisation de requêtes : premières conclusions

Slide 310

1. Il faut **traduire** une requête exprimée avec un langage déclaratif en une suite d'opérations (typiquement les opérateurs de l'algèbre relationnelle).
2. En fonction (i) des coûts de chaque opération (ii) des caractéristiques de la base, (iii) des algorithmes utilisés, on cherche à estimer la meilleure stratégie.
3. On obtient le **plan d'exécution** de la requête. Il n'y a plus qu'à le traiter au niveau physique.

Les paramètres de l'optimisation

Slide 311

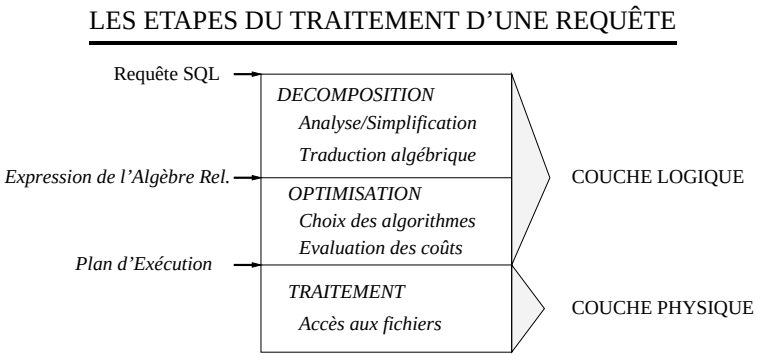
Comme on l'a vu sur l'exemple, l'optimisation s'appuie sur :

1. Des **règles de réécriture** des expressions de l'algèbre.
2. Des connaissances sur **l'organisation physique** de la base (index, hachage,. . .)
3. Des **statistiques** sur les caractéristiques de la base (taille des relations par exemple).

Un **modèle de coût** permet de classer les différentes stratégies envisagées

Architecture d'un SGBD et Optimisation

Slide 312



Slide 313

Décomposition de requêtes

Analyse syntaxique

Slide 314

On vérifie la validité (syntaxique) de la requête.

1. Contrôle de la correction grammaticale.
2. Vérification de l'existence des relations et des noms d'attributs.

$\Rightarrow$ on utilise le "dictionnaire" de la base.

Simplification

D'autres types de transformations avant optimisation. Citons :

Slide 315

1. **L'analyse sémantique** détecte les incohérences ou les contradictions (exemple : "$NoSalle = 11\ AND\ NoSalle = 12$").
2. **Simplification** de clauses inutilement complexes. Exemple : $(A\ OR\ NOT\ B)\ AND\ B$ est équivalent à $A\ AND\ B$.
3. Enfin la requête est normalisée pour faciliter la traduction algébrique.

Traduction algébrique

Déterminer l'expression algébrique équivalente à la requête :

Slide 316

1. arguments du SELECT : projections.
2. arguments du WHERE : $NomAttr1 = NomAttr2$ correspond en général à une jointure, $NomAttr = constante$ à une sélection.

On obtient une expression algébrique qui peut être représentée par un **arbre de requête**.

Traduction algébrique : exemple

Considérons l'exemple suivant :

$$Quels\ films\ passent\ au\ REX\ à\ 20\ heures\ ?$$

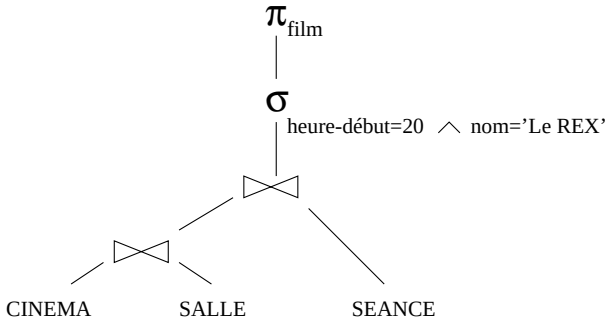
Slide 317

```
SELECT film
FROM   Cinéma, Salle, Séance
WHERE  Cinéma.nom-cinéma = 'Le Rex'
AND    Séance.heure-début = 20
AND    Cinéma.nom-cinéma = Salle.nom-cinéma
AND    Salle.salle = Séance.salle
```

Expression algébrique et arbre de requête

$$\pi_{film}(\sigma_{Nom='Le\ REX' \wedge heure-début=20}((CINEMA \bowtie SALLE) \bowtie SEANCE))$$

Slide 318



Restructuration

Il y a plusieurs expressions **équivalentes** pour une même requête.

Slide 319 ROLE DE L'OPTIMISEUR

1. Trouver les expressions équivalentes à une requête.
2. Les évaluer et choisir la meilleure.

On convertit une expression en une expression équivalente en employant des **règles de réécriture**.

Règles de réécriture

Il en existe beaucoup : en voici 8 parmi les plus importantes.

1. **Commutativité des jointures** :
   $R \bowtie S \equiv S \bowtie R$
2. **Associativité des jointures** :
   $(R \bowtie S) \bowtie T \equiv R \bowtie (S \bowtie T)$

Slide 320

3. **Regroupement des sélections** :
   $\sigma_{A=' a' \wedge B=' b'}(R) \equiv \sigma_{A=' a'}(\sigma_{B=' b'}(R))$
4. **Commutativité de la sélection et de la projection**
   $\pi_{A_1,A_2,\ldots A_p}(\sigma_{A_i=' a}(R)) \equiv \sigma_{A_i=' a}(\pi_{A_1,A_2,\ldots A_p}(R)), i \in \{1,\ldots,p\}$
5. **Commutativité de la sélection et de la jointure**.
   $\sigma_{A=' a'}(R(\ldots A \ldots) \bowtie S) \equiv \sigma_{A=' a'}(R) \bowtie S$

6. **Distributivité de la sélection sur l'union**.
   $\sigma_{A='a'}(R \cup S) \equiv \sigma_{A='a'}(R) \cup \sigma_{A='a'}(S)$
   NB : valable aussi pour la différence.

Slide 321

7. **Commutativité de la projection et de la jointure**
   $\pi_{A_1 \ldots A_p B_1 \ldots B_q}(R \bowtie_{A_i = B_j} S) \equiv$
   $\pi_{A_1 \ldots A_p}(R) \bowtie_{A_i = B_j} \pi_{B_1 \ldots B_q}(S))\ i \in \{1, \ldots, p\}, j \in \{1, \ldots, q\}$
8. **Distributivité de la projection sur l'union**
   $\pi_{A_1 A_2 \ldots A_p}(R \cup S) \equiv \pi_{A_1 A_2 \ldots A_p}(R) \cup \pi_{A_1 A_2 \ldots A_p}(S)$

Exemple d'un algorithme de restructuration

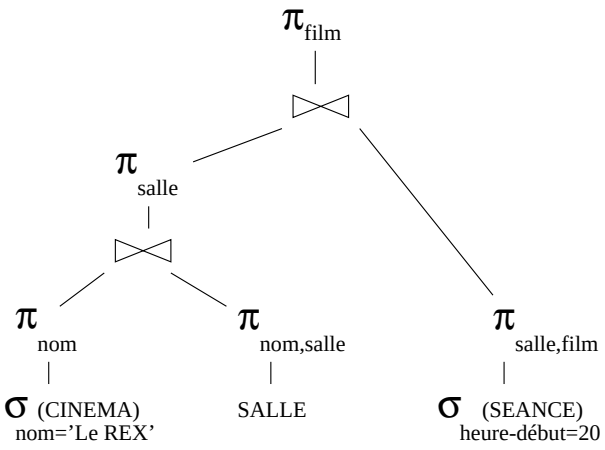
Voici un algorithme basé sur les propriétés précédentes.

Slide 322

1. Séparer les sélections avec plusieurs prédicats en plusieurs sélections à un prédicat (règle 3).
2. Descendre les sélections le plus bas possible dans l'arbre (règles 4, 5, 6).
3. Regrouper les sélections sur une même relation (règle 3).
4. Descendre les projections le plus bas possible (règles 7 et 8).
5. Regrouper les projections sur une même relation.

Arbre de requête après restructuration

Slide 323



Quelques remarques sur l'algorithme précédent

L'idée de base est de réduire le plus rapidement possible la taille des relations manipulées. Donc :

Slide 324

1. On effectue les sélections d'abord car on considère que c'est l'opérateur le plus "réducteur".
2. On élimine dès que possible les attributs inutiles par projection.
3. Enfin on effectue les jointures.

Le plan obtenu est-il toujours optimal ? NON !!!

Un contre-exemple

Quels sont les films visibles entre 14h et 22h?

Slide 325

Voici deux expressions de l'algèbre, dont l'une "optimisée" :

1. $\pi_{film}(\sigma_{h-début>14 \wedge h-début<22}(FILM \bowtie SEANCE))$
2. $\pi_{film}(FILM \bowtie \sigma_{h-début>14 \wedge h-début<22}(SEANCE))$

La relation FILM occupe 8 pages, la relation SEANCE 50.

Evaluation des coûts

Hypothèses : (i) 90 % des séances ont lieu entre 14 et 22 heures, (ii) seulement 20 % des films sont à l'affiche.

Slide 326

1. Jointure : on lit 400 pages et on aboutit à 10 pages (20% de 50 pages). Sélection : on se ramène à 9 pages.
   Nombre d'E/S : 400 + 10x2 = 420.
2. Sélection : on lit 50 pages et on aboutit à 45 pages. Jointure : on lit 360 pages et on aboutit à 9 pages.
   Nombre d'E/S : 50 + 45x2 + 360 = 500.

$\Rightarrow$ la première stratégie est la meilleure ! Ici la jointure est plus réductrice que la sélection (cas rare).

Traduction algébrique : conclusion

Slide 327

La réécriture algébrique est nécessaire mais pas suffisante. L'optimiseur tient également compte :

1. Des chemins d'accès aux données.
2. Des différents algorithmes implantant une même operation algébrique.
3. De propriétés statistiques sur la base.

Les chemins d'accès

Slide 328

Ils dépendent des organisations de fichiers existantes :

1. Balayage séquentiel
2. Parcours d'index
3. Accès par hachage

Attention ! Dans certains cas un balayage peut être préférable à un parcours d'index.

Algorithmes pour les opérations algébriques

Slide 329

Il existe souvent plusieurs algorithmes pour implanter une opération. Exemple : la jointure.

1. Boucles imbriquées simple
2. Boucles imbriquées avec accès à une des relations par index ou par hachage.
3. Tri-fusion
4. Jointure par hachage
5. etc.

Le choix dépend essentiellement - mais pas totalement - du chemin d'accès disponible.

Algorithmes de jointure

Slide 330

En l'absence d'index, les principaux algorithmes sont :

1. Boucles imbriquées.
2. Tri-fusion.
3. Jointure par hachage.

Jointure par boucles imbriquées

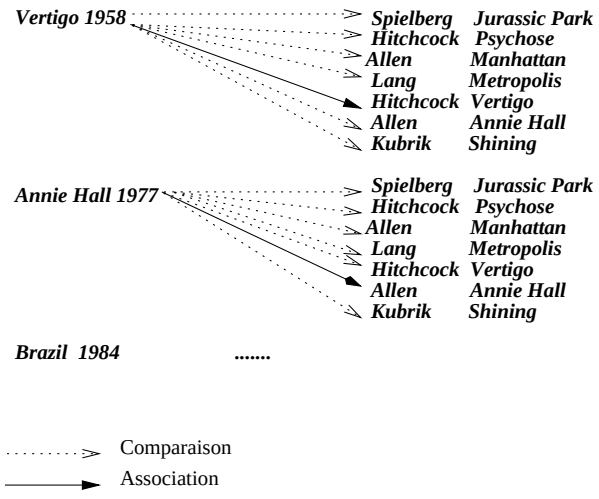
A utiliser quand les tailles des relations sont petites.
Soit les deux relations R et S :

Slide 331

ALGORITHME **boucles-imbriquées**
begin
J:=$\emptyset$
for each r **in** R
  for each s **in** S
    if *r et s sont joignables* **then** $J := J + \{r \bowtie s\}$
end

Exemple de jointure par boucles imbriquées

Slide 332



Analyse

La boucle s'effectue en fait à deux niveaux :

1. Au niveau des **pages** pour les charger en mémoire.

Slide 333

2. Au niveau des **articles** des pages chargées en mémoire.

Du point de vue E/S, c'est la première phase qui compte. Si T_R et T_S représentent le nombre de pages de R et S respectivement, le coût de la jointure est :

$$T_R \times T_S$$

On ne tient pas compte dans l'évaluation du coût des algorithmes de jointure, du coût d'écriture du résultat sur disque, lequel dépend de la taille du résultat.

Jointure par tri-fusion

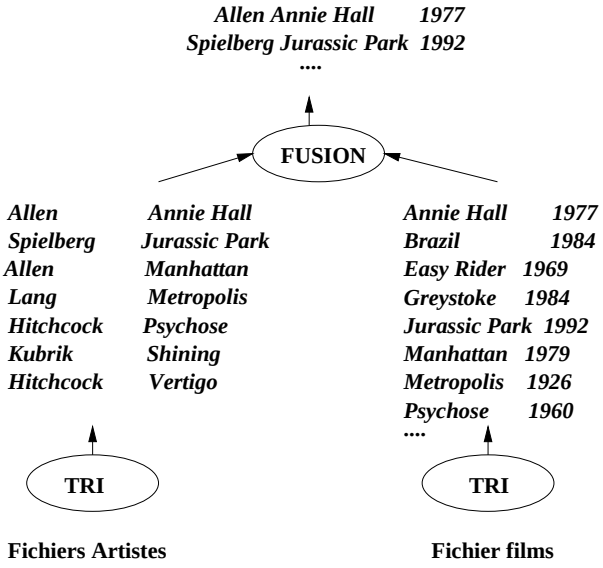
Soit l'expression $\pi_{R.Ap,S.Bq}(R \bowtie_{A_i=B_j} S)$.

Slide 334

Projeter R sur $\{A_p, A_i\}$
Trier R sur A_i
Projeter S sur $\{B_q, B_j\}$
Trier S sur B_j
Fusionner les deux listes triées.

On les parcourt en parallèle en joignant les n-uplets ayant même valeur pour A_i et B_j.

Slide 335



Tri-fusion : performance

Slide 336

Le coût est dominé par la phase de tri :

$$O(card(R) \times \log(card(R)) + card(S) \times \log(card(S))).$$

Dans la seconde phase, un simple parcours en parallèle suffit.

Cet algorithme est particulièrement intéressant quand les données sont déjà triées en entrée.

Slide 337

Pour des grandes relations et en l'absence d'index, la jointure par tri-fusion présente les avantages suivants :

1. **Efficacité** : bien meilleure que les boucles imbriquées.
2. **Manipulation de données triées** : facilite l'élimination de dupliqués ou l'affichage ordonné.
3. **Très général** : permet de traiter tous les types de θ-jointure

Jointure par hachage

Slide 338

Comme le tri-fusion, la jointure par hachage permet de limiter le nombre de comparaisons entre n-uplets.

1. Une des relations, $R1$, est hachée sur l'attribut de jointure avec une fonction H.
2. La deuxième relation est parcourue séquentiellement. Pour chaque n-uplet, on consulte la page indiquée par application de la fonction H et on regarde si elle contient des n-uplets de $R1$. Si oui on fait la jointure limitée à ces n-uplets.

Jointure par hachage : algorithme

Pour une jointure $R \bowtie_{A=B} S$.

Slide 339

Pour chaque n-uplet r de R **faire**
placer r dans la page indiqu ee par $H(r.A)$
Pour chaque n-uplet s de S **faire**
calculer $H(s.B)$
lire la page p indiqu ee par $H(s.B)$
effectuer la jointure entre $\{s\}$ et les n-uplets de p

Jointure par hachage : discussion

Coût (en E/S), en supposant k articles par page et un tampon de 2 pages en memoire centrale (dans le pire des cas):

Slide 340

1. Phase 1 : Coût du hachage de $R1$: $T_{R1} + 2 \times k \times T_{R1}$.
2. Phase 2 : Lecture de $R2$. Pour chaque page, on lit k pages de la relation hachée $R1$.

$$\text{Coût} = ((1 + 2k) \times T_{R1}) + ((1 + k) \times T_{R2})$$

Si $R1$ tient en mémoire centrale, le coût se réduit à $T_{R1} + T_{R2}$.

NB : contrairement au tri-fusion, la jointure par hachage n'est pas adaptée aux jointures avec inégalités.

Jointure avec une table indexée

Slide 341

1. On parcourt séquentiellement la relation sans index.
2. Pour chaque n-uplet, on recherche par l'index les n-uplets de la seconde relation qui satisfont la condition de jointure (traversée de l'index et accès aux nuplets de la seconde relation par adresse)

Boucles imbriquées avec une table indexée

Slide 342

ALGORITHME **boucles-imbriquées-index**
begin
J:=$\emptyset$
for each r **in** R
  for each s **in** $Index_{S_B}(r.A)$
    $J := J + \{r \bowtie s\}$
end

La fonction $Index_{S_B}(r.A)$ donne les nuplets de S dont l'attribut B a pour valeur $r.A$ en traversant l'index de S sur B

Coût : $O(card(R) \times log(card(S)))$.

Jointure avec deux tables indexées

Si les deux tables sont indexées, on peut utiliser une variante de l'algorithme de tri-fusion :

Slide 343

1. On fusionne les deux index (déjà triés) pour constituer une liste (Rid, Sid) de couples d'adresses pour les articles satisfaisant la condition de jointure.
2. On parcourt la liste en accédant aux tables pour constituer le résultat.

Inconvénient : on risque de lire plusieurs fois la même page. En pratique, on préfère utilise une boucle imbriquée en prenant la plus petite table comme table directrice.

Statistiques

Permettent d'ajuster le choix de l'algorithme. Par exemple :

Slide 344

1. Boucles imbriquées simples si les relations sont petites.
2. Balayage séquentiel au lieu de parcours d'index si la sélectivité est faible.

Suppose

1. Soit l'existence d'un module récoltant périodiquement des statistiques sur la base
2. Soit l'estimation en temps réel des statistiques par échantillonnage.

Plans d'exécution

Le résultat de l'optimisation est un **plan d'exécution**: c'est un ensemble d'opérations de niveau intermédiaire, dit **algèbre "physique"** constituée :

Slide 345

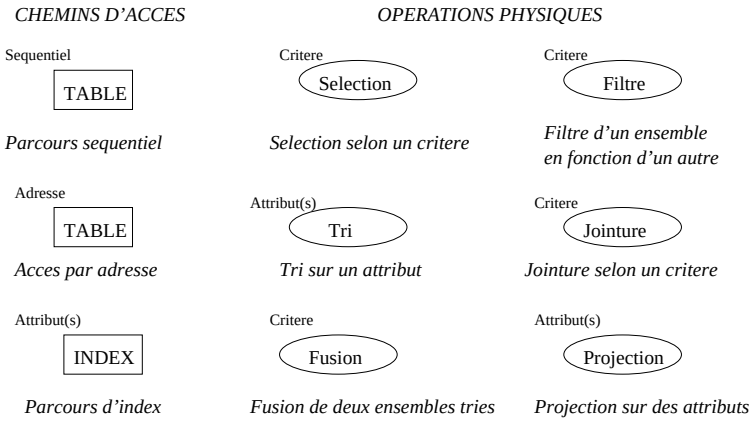
1. De chemins d'accès aux données
2. D'opérations manipulant les données, (correspondant aux noeuds internes de l'arbre de requête).

Plans d'exécution sur la requête :

$$Quels\ films\ passent\ au\ REX\ à\ 20\ heures\ ?$$

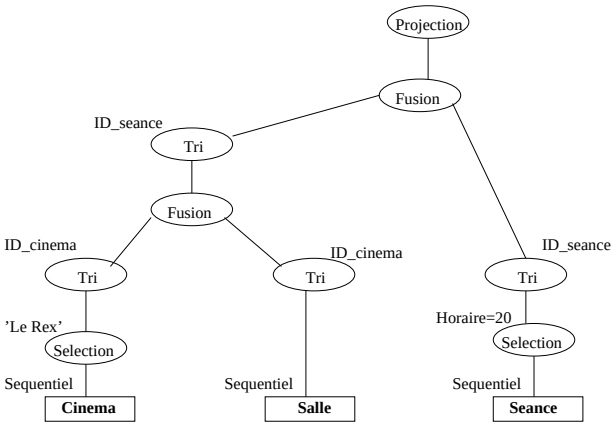
Algèbre physique

Slide 346



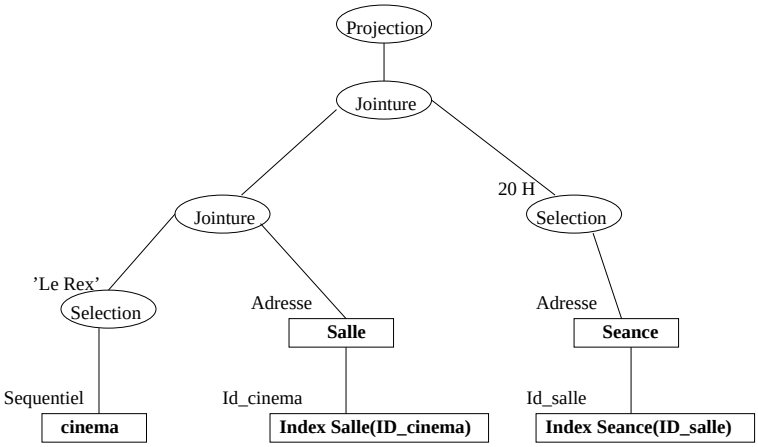
Sans index ni hachage

Slide 347



Avec un index sur toutes les clés

Slide 348



Slide 349

Evaluation de requêtes

En quoi consiste l’évaluation d’une requête

Slide 350

Le résultat de l’optimisation est un plan d’exécution, i.e. une séquence d’opérations à exécuter. On doit maintenant :

1. Appliquer les techniques d’accès appropriées pour chaque opérateur.
2. Gérer les flots de données entre chaque opération

Les algorithmes, techniques d’accès et heuristiques mise en oeuvre relèvent de **l’évaluation de requêtes**.

Objectifs de l'évaluation de requêtes

Slide 351

Dans l'hypothèse d'une base centralisée, on cherche essentiellement à limiter le nombre d'entrées/sorties. La stratégie employée dépend cependant fortement du point de vue adopté :

1. Soit on cherche à obtenir le premier enregistrement le plus vite possible (exemple d'une application interactive).
2. Soit on cherche à minimiser le temps global d'exécution (exemple d'une application batch).

Selon le cas, le choix des algorithmes peut varier.

Plan de la présentation

Slide 352

On va étudier successivement les points suivants :

1. **Techniques d'accès** : parcours séquentiels, parcours d'index, gestion de la mémoire centrale.
2. **Algorithmes de base** : tri, hachage, fusion.
3. **Implantation des opérateurs algébriques** : sélection, jointure, agrégation.
4. **Flots de données** entre opérateurs.

Exemple de référence

Slide 353

Vertigo	*1958*	*Metropolis*	*1926*
Annie Hall	*1977*	*Psychose*	*1960*
Brazil	*1984*	*Greystoke*	*1984*
Twin Peaks	*1990*	*Shining*	*1980*
Jurassic Park	*1992*	*Manhattan*	*1979*
Underground	*1995*	*Easy Rider*	*1969*

Les films

Spielberg	*Jurassic Park*
Hitchcock	*Psychose*
Allen	*Manhattan*
Lang	*Metropolis*
Hitchcock	*Vertigo*
Allen	*Annie Hall*
Kubrik	*Shining*

Les metteurs en scène

Techniques d’accès

Slide 354

Techniques d'accès : parcours séquentiel

Slide 355

Systématiquement optimisé dans les SGBD en utilisant les techniques suivantes :

1. **Regroupement** des pages disques sur des espaces contigus (nommés **segments** ou **extensions**).
2. **Lecture à l'avance** : quand on lit une page, on prend également les n (typiquement $n = 7$ ou $n = 15$) suivantes dans le segment.

$\Rightarrow$ l'unité d'E/S dans un SGBD est donc souvent un multiple de celle du gestionnaire de fichier sous-jacent.

Techniques d'accès : parcours d'index

Slide 356

La plupart des SGBD utilisent une des variantes de l'arbre B. Outre les recherches par clés et par intervalle, ils permettent :

1. D'éviter l'accès aux données quand la valeur recherchée est dans l'index.
2. De faire directement des comptages ou des tests d'existence.
3. Enfin on peut optimiser des opérations de sélection ou de jointure en manipulant les adresses stockées dans l'index.

On va prendre un exemple de ce dernier type d'optimisation.

Soit la requête suivante, en supposant un index sur Année :

select titre **from** FILM
whereannée **IN** (1956, 1934, 1992, 1997)

Slide 357

Implantation simple : on recherche dans l'index les adresses pour chaque valeur du **IN** et on lit l'enregistrement.

Implantation optimisée :

1. On recherche dans l'index **toutes** les adresses pour **toutes** les valeurs du **IN**.
2. On regroupe l'ensemble d'adresses par numéro de page.
3. On lit les pages et on extrait les enregistrements

L'optimisation précédente a permis de ne pas lire deux fois la même page. **Mais** elle impose d'attendre qu'on ait lu toutes les adresses avant d'afficher le résultat.

On peut appliquer une technique intermédiaire utilisant une zone tampon T :

Slide 358

1. On lit les adresses et on les place (triées) dans T.
2. Dès qu'il faut fournir une donnée, ou que T est plein, on lit la page la plus référencée.

Cette technique peut être utilisée pour des jointures.

Techniques d'accès : gestion de la mémoire

Problème très complexe : essayer de conserver en mémoire une page susceptible d'être réutilisée prochainement.

Slide 359

Le programme exécutant la requête ne demande pas lui-même la lecture, mais s'adresse au $buffer\ manager$ du SGBD. Les concepts essentiels sont :

1. **Page statique** : la page reste en mémoire jusqu'à ce qu'on demande au $buffer\ manager$ de la libérer.
2. **Page volatile** : la page est à la disposition du $buffer\ manager$.

Algorithmes de base

Slide 360

Le tri

Le tri est l'opération la plus fréquente. On l'utilise par exemple :

Slide 361

1. Pour afficher des données ordonnées (clause ORDER BY).
2. Pour éliminer des doublons ou faire des agrégats.
3. Pour certains algorithmes de jointure.

L'algorithme utilisé dans les SGBD est le **tri par fusion** ($MERGE\ SORT$).

Rappels sur l'algorithme de tri par fusion

On applique la stratégie dite "diviser pour régner". Elle consiste à :

Slide 362

1. **Diviser** récursivement de problème jusqu'à obtenir des sous-problèmes trivialement résolubles.
2. **Fusionner** récursivement les solutions des sous-problèmes.

NB : quand on peut faire le tri en mémoire centrale, on utilise plutôt le **tri rapide** ($Quicksort$).

Fusion de deux listes triées

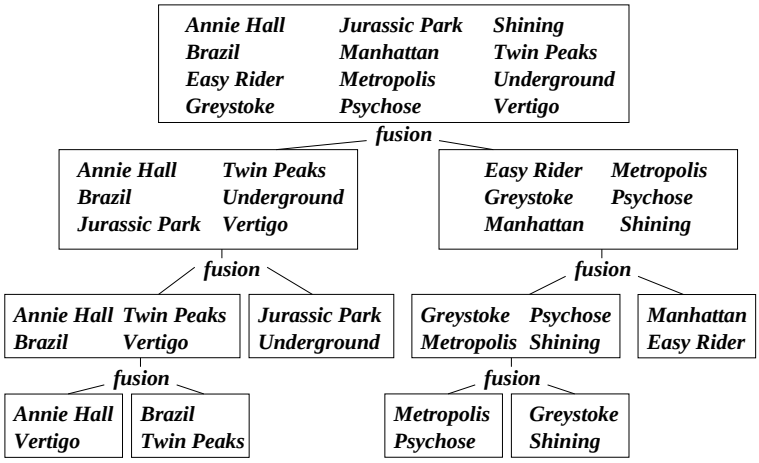
Soit deux listes triées A et B. On les fusionne en effectuant un parcours parallèle.

Slide 363

```
Algorithme Fusionner(A,B)
begin
  a := premier élément de A; b := premier élément de B
  while il reste un élément dans A ou dans B
        if a est avant b then
          Ecrire a; a := élément suivant de A
        else
          Ecrire b; b := élément suivant de B
end
```

Exemple de tri-fusion sur le fichier des films

Slide 364



Tri-fusion dans un contexte SGBD

Slide 365

Dans un contexte SGBD, on fixe les paramètres du tri-fusion de la manière suivante :

1. On s'arrête de diviser quand on peut trier en mémoire centrale. Donc la taille M de la mémoire disponible détermine la taille des sous-problème triviaux.
2. Dans la phase de fusion, le facteur de division est $F = (M/B) - 1$ où B est la taille d'un buffer. On a F buffers en lecture et 1 en écriture.

Algorithme de tri-fusion dans un SGBD

Slide 366

Soit M la mémoire disponible en unités E/S, B la taille d'un buffer et E la taille de l'entrée.

1. Diviser récursivement l'entrée en fichiers de taille $T < M$.
   NB : on obtient $N = \lceil E/M \rceil$ fichiers.
2. Trier chaque fichier en mémoire centrale avec $Quicksort$.
3. Fusionner. A chaque étape, on fusionne $F = \lfloor (M/B) - 1 \rfloor$ fichiers simultanément.
   NB : il y a $log_F(N)$ étapes de fusion.

Exemple

Slide 367

Supposons qu'un article de $Films$ occupe $8KB$. Le fichier de 12 films occupe $E = 96KB$. On dispose de $16KB$ en mémoire et l'unité d'E/S est de $4KB$. Donc il y a 4 unités d'E/S :

1. On doit diviser le fichier en sous-fichiers de taille $T < 16KB$. On obtient 6 sous-fichiers de taille $T = 16KB$.
2. On trie avec $Quicksort$ chacun de ces 6 fichiers.
3. On dispose de $3 = (16/4) - 1$ buffers. Donc on peut faire une fusion en deux étapes.

Coût d'une opération de tri

Slide 368

L'analyse donne :

1. $N = \lceil E/M \rceil$ unités d'E/S.
2. Chaque unité est sujette à $L = log_{\lfloor (M-1)/B \rfloor}(N)$ étapes de fusion.
3. Chaque étape consiste en une lecture **et** une écriture.

D'où un coût global de $2 \times log_{\lfloor (M-1)/B \rfloor} \times N$.

NB : ce type de formule est utilisé dans un modèle de coût.

Hachage

Slide 369

On se place dans le cas où le fichier à hacher ne tient pas en mémoire centrale. On dispose de deux techniques extrêmes:

1. **Hachage en mémoire secondaire:** on détermine dès le départ le nombre de partitions nécessaires, et on écrit chaque partition sur disque.
2. **Hachage en mémoire centrale**: on partitionne le fichier en mémoire centrale. Si un débordement survient, on écrit les partitions sur disque.

En pratique, on utilise un technique intermédiaire, le **hachage hybride**.

Principes du hachage hybride

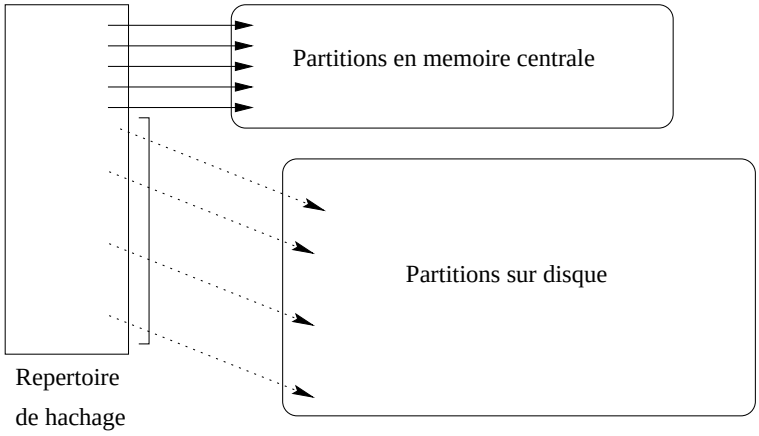
Slide 370

On procède de la manière suivante:

1. On commence par hacher en mémoire centrale.
2. Quand un débordement survient:
   (a) On divise la partie déjà hachée en $F = \lfloor M/B - 1 \rfloor$ partitions, où M est la taille mémoire, et B la taille d'un buffer d'E/S.
   (b) On écrit une des partitions sur disque, et on continue le hachage.

Donc on optimise, à chaque instant, l'occupation de la mémoire.

Slide 371



Quelques détails

Le hachage hybride n'est valable que pour des entrées dont la taille E est telle que $M \leq E \leq F \times M$. On sait alors qu'on peut écrire une partie seulement du fichier sur disque.

Slide 372 Pour déterminer une partition, on suit les principes suivants ;

1. On place plusieurs entrées de la table de hachage dans une même partition.
2. A la fin du hachage, la taille d'une partition devrait être inférieure à celle de la mémoire disponible.

Il existe plusieurs techniques (sophistiquées) pour décider quelles entrées sont placées dans une partition.

Analyse du hachage hybride

Slide 373

Soit K le nombre de partitions écrites sur disque ($0 \leq K \leq F$).

1. Il faut 1 buffer pour la lecture, et K pour écrire sur disque.
2. Il reste $M - (K + 1) \times B$ buffers en mémoire centrale.
3. Pour un fichier de taille E, K est le plus petit nombre tel que $K \times M + (M - (K + 1) \times B) \geq E$. Soit $K = \lceil (E - M + B)/(M - B) \rceil$.

On obtient le nombre (minimal) d'E/S: $2 \times (E - (M - (K + 1) \times B))/B$.

Exemple

Slide 374

Soit les valeurs suivantes: $E = 96KB, M = 48KB, B = 4KB$.

1. Nombre de partitions sur disque: $K = \lceil (96 - 48 + 4)/(48 - 4) = 2$. Chaque partition peut contenir $48KB$.
2. On utilise 8 pages en mémoires pour écrire, 4 pour lire.
3. Le reste de la mémoire, de taille $96 - 8 - 4 = 84$, contient la table de hachage.

Remarque: pour $M = 16KB$, le nombre de buffers est $F = 16/4 - 1 = 3$. On a $E = 96 > F \times M = 38$. Donc le hachage hybride n'est pas utile.

Slide 375

REPRESENTATION PHYSIQUE DANS ORACLE

Représentation physique dans ORACLE V7

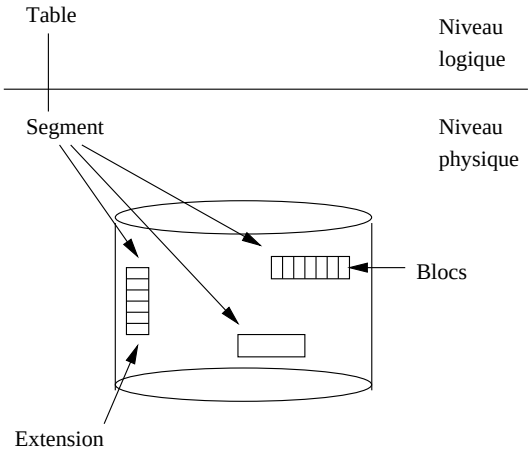
Slide 376

Les principales structures physiques utilisées dans ORACLE sont :

1. Le **bloc** est l'unité physique d'E/S. La taille d'un bloc ORACLE est un multiple de la taille des blocs du système sous-jacent.
2. L'**extension** est un ensemble de blocs contigus contenant un même type d'information.
3. Le **segment** est un ensemble d'extensions stockant un objet logique (une table, un index ...).

Tables, segments, extensions et blocs

Slide 377



Le segment ORACLE

Slide 378

Le segment est la zone physique contenant un objet logique. Il existe quatre types de segments :

1. Le segment de données (pour une table ou un $cluster$).
2. Le segment d'index.
3. Le $rollback\ segment$ utilisé pour les transactions.
4. Le segment temporaire (utilisé pour les tris par exemple).

Base ORACLE, fichiers et $TABLESPACE$

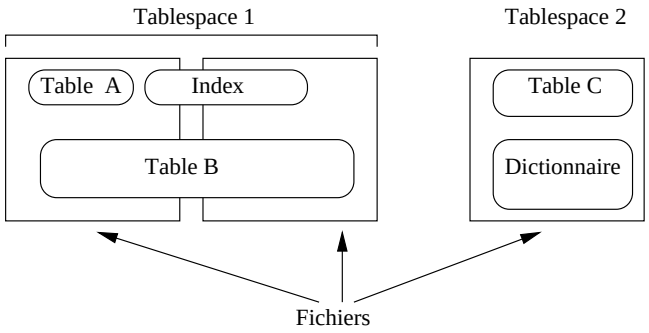
Slide 379

1. **Physiquement**, une base ORACLE est un ensemble de fichiers.
2. **Logiquement**, une base est divisée par l'administrateur en $tablespace$. Chaque $tablespace$ consiste en un (au moins) ou plusieurs fichiers.

La notion de $tablespace$ permet :

1. De contrôler l'emplacement physique des données. (par ex. : le dictionnaire sur un disque, les données utilisateur sur un autre).
2. de faciliter la gestion (sauvegarde, protection, etc).

Slide 380



Stockage des données

Slide 381

Il existe deux manières de stocker une table :

1. **Indépendamment**. Un segment est alors automatiquement alloué à la table. Il est possible de spécifier des paramètres pour ce segment :
   (a) Sa taille initiale
   (b) Le pourcentage d'espace libre dans chaque bloc.
   (c) La taille des extensions.
2. **Dans un** $cluster$.

Stockage des n-uplets

Slide 382

En règle générale un n-uplet est stocké dans un seul bloc. L'adresse physique d'un n-uplet est le $ROWID$ qui se décompose en trois parties :

1. Le numéro du n-uplet dans la page.
2. Le numéro de la page, relatif au **fichier** dans lequel se trouve le n-uplet.
3. Le numéro du fichier.

Exemple : 00000DD5.000.001 est l'adresse du premier n-uplet du bloc DD5 dans le premier fichier.

Structures de données pour l'optimisation

Slide 383

ORACLE 7 propose trois structures pour l'optimisation de requêtes :

1. Les index
2. Les "regroupements" de tables ($Cluster$).
3. Le hachage.

Les index ORACLE

Slide 384

On peut créer des index sur tout attribut (ou tout ensemble d'attributs) d'une table. ORACLE utilise l'arbre B+.

1. Les noeuds contiennent les valeurs de l'attribut (ou des attributs) clé(s).
2. Les feuilles contiennent chaque valeur indexée et le $ROWID$ correspondant.

Un index est stocké dans un segment qui lui est propre. On peut le placer par exemple sur un autre disque que celui contenant la table.

Les $cluster$

Slide 385

Le $cluster$ (regroupement) est un structure permettant d'optimiser les jointures. Par exemple, pour les tables $CINEMA$ et $SALLE$ qui sont fréquemment jointes sur l'attribut $ID - Cinema$:

1. On groupe les n-uplets de $CINEMA$ et de $SALLE$ ayant même valeur pour l'attribut $ID - cinema$.
2. On stocke ces groupes de n-uplets dans les pages d'un segment spécial de type $cluster$.
3. On crée un index sur $ID - cinema$.

Slide 386

Clé de regroupement (ID-cinéma)	**Nom-cinéma**	**Adresse**	
1209	Le Rex	2 Bd Italiens	
	ID-salle	**Nom-salle**	**Capacité**
	1098	Grande Salle	450
	298	Salle 2	200
	198	Salle 3	120
1210	**Nom-cinéma**	**Adresse**	
	Kino	243 Bd Raspail	
	ID-salle	**Nom-salle**	**Capacité**
	980	Salle 1	340
	...	...	...

Le hachage

On définit un $hash\ cluster$ décrivant les caractéristiques physiques de la table :

Slide 387

```
CREATE CLUSTER hash-cinéma (ID-cinéma NUMBER(10))
   HASH IS ID-cinéma HASHKEYS 1 000 SIZE 2K
```

$HASH\ IS$ (optionnel) spécifie la clé à hacher.
$HASHKEYS$ est le nombre de valeurs de la clé de hachage.
$SIZE$ est la taille d'un n-uplet. ORACLE détermine le nombre de pages allouées, ainsi que la fonction de hachage. On fait référence au $hash\ cluster$ en créant une table.

Slide 388

OPTIMISATION DANS ORACLE

Optimisation : l’exemple de ORACLE Version 7

Slide 389 Plan de la présentation :

1. Optimisation : principes et outils d’analyse.
2. Présentation sur des exemples.

Slide 390

Optimisation - principes généraux et outils d’analyse

L’optimiseur

Slide 391

L’optimiseur ORACLE suit une approche classique :

1. Génération de plusieurs plans d’exécution.
2. Estimation du coût de chaque plan généré.
3. Choix du meilleur et exécution.

Estimation du coût d’un plan d’exécution

Slide 392

Beaucoup de paramètres entrent dans l’estimation du coût :

1. Les chemins d’accès disponibles.
2. Les opérations physiques de traitement des résultats intermédiaires.
3. Des statistiques sur les tables concernées (taille, sélectivité). Les statistiques sont calculées par appel explicite à l’outil ANALYSE.
4. Les ressources disponibles.

Les chemins d'accès

Slide 393

1. **Parcours séquentiel** ($FULL\ TABLE\ SCAN$).
2. **Par adresse** ($ACCESS\ BY\ ROWID$).
3. **Parcours de regroupement** ($CLUSTER\ SCAN$). On récupère alors dans une même lecture les n-uplets des 2 tables du $cluster$.
4. **Recherche par hachage** ($HASH\ SCAN$).
5. **Parcours d'index** ($INDEX\ SCAN$).

Opérations physiques

Slide 394

Voici les principales :

1. $INTERSECTION$: intersection de deux ensembles de n-uplets.
2. $CONCATENATION$: union de deux ensembles.
3. $FILTER$: élimination de n-uplets (sélection).
4. $PROJECTION$: opération de l'algèbre relationnelle.

D'autres opérations sont liées aux algorithmes de jointures.

Algorithmes de jointure sous ORACLE

Slide 395

ORACLE utilise trois algorithmes de jointure :

1. **boucles imbriquées** quand il y a au moins un index.
   Opération $NESTED\ LOOP$.
2. **Tri/fusion** quand il n'y a pas d'index.
   Opération $SORT$ et $MERGE$.
3. Enfin, en présence d'un $join\ cluster$, on fait une recherche avec l'index du $cluster$, puis un accès au $cluster$ lui-même.

L'outil EXPLAIN

Slide 396

L'outil EXPLAIN donne le plan d'exécution d'une requête. La description comprend :

1. Le chemin d'accès utilisé.
2. Les opérations physiques (tri, fusion, intersection, ...).
3. L'ordre des opérations. Il est représentable par un arbre.

EXPLAIN par l'exemple : schéma de la base

Slide 397

```
CINEMA          SALLE           FILM
  (ID-cinéma*,    (ID-salle*,     (ID-film,
   Nom,            Nom,            Titre,
   Adresse);       Capacité,       Année,
                   ID-cinéma+);    ID-réalisateur+)

  SEANCE (ID-séance*,           ARTISTE (ID-artiste*,
          Heure-début, Heure-fin,          Nom,
          ID-salle+,ID-film                Date-naissance)
```

Attributs avec une * : index unique.
Attributs avec une + : index non unique.

Interprétation d'une requête par EXPLAIN

Reprenons l'exemple : Quels films passent aux Rex à 20 heures ?

Slide 398

```
EXPLAIN PLAN SET statement-id = 'cin'
FOR     SELECT ID-film
        FROM   Cinéma, Salle, Séance
        WHERE  Cinéma.ID-cinéma = Salle.ID-cinéma
        AND    Salle.ID-salle = Séance.ID-salle
        AND    Cinéma.nom = 'Le Rex'
        AND    Séance.heure-début = '20H'
```

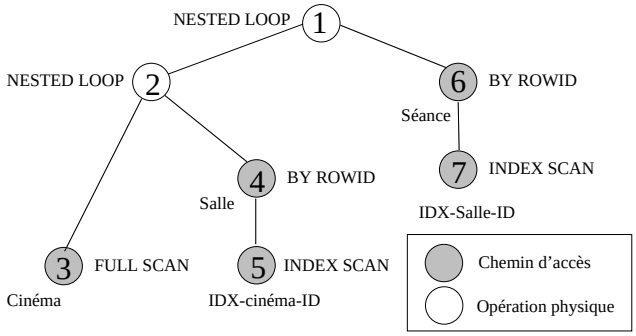
Plan d’exécution donné par EXPLAIN

Slide 399

```
0 SELECT STATEMENT
  1 NESTED LOOP
    2 NESTED LOOPS
      3 TABLE ACCESS FULL CINEMA
      4 TABLE ACCESS BY ROWID SALLE
        5 INDEX RANGE SCAN IDX-CINEMA-ID
    6 TABLE ACCESS BY ROWID SEANCE
      7 INDEX RANGE SCAN IDX-SALLE-ID
```

Représentation arborescente du plan d’exécution

Slide 400



Quelques remarques sur EXPLAIN

Slide 401

EXPLAIN utilise un ensemble de primitives que nous avons appelé "algèbre physique: des opérations comme le tri n'existent pas au niveau relationnel. D'autres opérations de l'algèbre relationnelle sont regroupées en une seule opération physique.

- Par exemple, la sélection sur l'horaire des séances est effectuée en même temps que la recherche par ROWID (étape 6).

Slide 402

Optimisation dans ORACLE - exemples

Sélection sans index

Slide 403

```
SELECT * FROM cinéma WHERE  nom = 'Le Rex'
```

Plan d'exécution :

```
0 SELECT STATEMENT
  1 TABLE ACCESS FULL CINEMA
```

Sélection avec index

Slide 404

```
SELECT * FROM cinéma WHERE  ID-cinéma = 1908
```

Plan d'exécution :

```
0 SELECT STATEMENT
  1 TABLE ACCESS BY ROWID CINEMA
    2 INDEX UNIQUE SCAN IDX-CINEMA-ID
```

Sélection conjonctive avec un index

Slide 405

```
SELECT capacité FROM Salle
WHERE  ID-cinéma =187 AND nom = 'Salle 1'
```

Plan d'exécution :

```
0 SELECT STATEMENT
  1 TABLE ACCESS BY ROWID SALLE
    2 INDEX RANGE SCAN IDX-SALLE-CINEMA-ID
```

Sélection conjonctive avec deux index

Slide 406

```
SELECT  nom FROM   Salle
 WHERE ID-cinéma = 1098 AND capacité = 150
```

Plan d'exécution :

```
0 SELECT STATEMENT
  1 TABLE ACCESS BY ROWID SALLE
    2 AND-EQUAL
      3 INDEX RANGE SCAN IDX-SALLE-CINEMA-ID
      4 INDEX RANGE SCAN IDX-CAPACITE
```

Sélection disjonctive avec index

SELECT nom FROM Salle
WHERE ID-cinéma = 1098 OR capacité > 150

Slide 407

Plan d'exécution :

```
0 SELECT STATEMENT
  1 CONCATENATION
    2 TABLE ACCESS BY ROWID SALLE
      3 INDEX RANGE SCAN IDX-CAPACITE
    4 TABLE ACCESS BY ROWID SALLE
      5 INDEX RANGE SCAN IDX-SALLE-CINEMA-ID
```

Sélection disjonctive avec et sans index

Slide 408

```
SELECT  nom FROM   Salle
WHERE   ID-cinema = 1098 OR nom = 'Salle 1'
```

Plan d'exécution :

```
0 SELECT STATEMENT
  1 TABLE ACCESS FULL SALLE
```

Jointure avec index

```
SELECT Cinéma.nom,capacité FROM cinéma, salle
WHERE  Cinéma.ID-cinéma = salle.ID-cinéma
```

Slide 409 Plan d'exécution :

```
0 SELECT STATEMENT
  1 NESTED LOOPS
    2 TABLE ACCESS FULL SALLE
    3 TABLE ACCESS BY ROWID CINEMA
      4 INDEX UNIQUE SCAN IDX-CINEMA-ID
```

Jointure et sélection avec index

SELECT Cinéma.nom,capacité FROM Cinéma, Salle
WHERE Cinema.ID-cinéma = salle.ID-cinéma
AND capacité $>$ 150

Slide 410 Plan d'exécution :

```
0 SELECT STATEMENT
  1 NESTED LOOPS
    2 TABLE ACCESS BY ROWID SALLE
      3 INDEX RANGE SCAN IDX-CAPACITE
    4 TABLE ACCESS BY ROWID CINEMA
      5 INDEX UNIQUE SCAN IDX-CINEMA-ID
```

Jointure sans index

```
SELECT titre
FROM Film, Séance
WHERE  Film.ID-film  = Séance.ID-film
AND     heure-début = '14H00'
```

Slide 411

Plan d'exécution :

```
0 SELECT STATEMENT
  1 MERGE JOIN
    2 SORT JOIN
      3 TABLE ACCESS FULL SEANCE
    4 SORT JOIN
      5 TABLE ACCESS FULL FILM
```

Différence

Dans quel cinéma ne peut-on voir de film après 23H ?

Slide 412

```
SELECT Cinéma.nom
FROM   Cinéma, Salle
WHERE  Cinéma.ID-cinéma = Salle.ID-cinéma
AND    NOT EXISTS (SELECT * FROM séance
                   WHERE Salle.ID-salle = Séance.ID-salle
                   AND heure-fin > '23H00')
```

Plan d’exécution donné par EXPLAIN

Slide 413

```
0 SELECT STATEMENT
  1 FILTER
    2 NESTED LOOPS
      3 TABLE ACCESS FULL SALLE
      4 TABLE ACCESS BY ROWID CINEMA
        5 INDEX UNIQUE SCAN IDX-CINEMA-ID
    6 TABLE ACCESS BY ROWID SEANCE
      7 INDEX RANGE SCAN IDX-SEANCE-SALLE-ID
```