Bases de données Relationnelles

Hala Skaf-Molli

Loria

bureau B032

e-mail: skaf@loria.fr

www.loria.fr/~skaf

Plan de cours

- Introduction
- Fonctionnalités d'un SGBD
- Développement d'une base de données
- Modèle relationnel
  - Algèbre relationnel
  - Langage de requête SQL
- Modèle entité-association

BIBLIOGRAPHIE

- C. Date , "An Introduction to Data Base System", Addison Wesley, 1994.
- C. Delobel, M. Adiba "Bases de données et systèmes relationnels", Dunod Informatique, 1982 .
- G. Gardarin "Bases de données : les systèmes et leurs langages", Eyrolles, 1983
- J. Ullman "Principles of database systems", Computer Science Press, 1982
- *N. Boudjlida. Bases de données relationnelles et systèmes d'informations : Langages, systèmes et méthodes. Cours et exercices corrigés. Dunod, Paris, 1999, Séries Sciences-Sup.*
- *H.G. Molina, J. Ullman, J. Widom. Database Systems the complete Book, Prentice Hall, 2002.*
- *Ramez Elmasri et Shamkant Navathe, Conception et architecture des bases de données, Pearson Edition, 4éme édition, 2004.*
- et d'autres références sur le web.

Introduction

- Pourquoi un SGBD ?
  - limite de SGF à gérer une grande masse volume de données et les liens entre ces données
  - données reparties entre plusieurs fichiers
  - redondances de données et problème du maintien de la cohérence…
  - effort de programmation important pour exploiter les données (séquentiel, séquentielle indexé, ..)
  - programme dépend de données..
- Un SGBD permet de répondre à ces besoins..

Base de Données

• Un ensemble de données bien structuré relatives à un sujet global et accessible par **plusieurs utilisateurs** à la fois.

•Exemples

- Une banque stocke les informations sur les clients et leurs dépôts d'épargne dans une BD.
- Une BD de réservation de tickets du SNCF.
- Un service de scolarité stocke les informations relatives aux étudiants inscrits dans une BD...

ETUDIANT	Nom	NumeroEtudiant	Niveau	Dominante
	Schmidt	17	1	INFO
	Brun	8	2	INFO

COURS	NomCours	NumeroCours	HeuresCredit	Departement
	Introduction à l'informatique	CS1310	4	INFO
	Structures de données	CS3320	4	INFO
	Mathématiques discrètes	CS1310	3	MATH
	Bases de données	CS3380	3	INFO

UV	IdUV	NumeroCours	Trimestre	Annee	Enseignant
	85	MATH2410	4	98	Crémant
	92	CS1310	4	98	André
	102	CS3320	3	99	Queneau
	112	MATH2410	3	99	Chenet
	119	CS1310	3	99	André
	135	CS3380	3	99	Segonzac

NOTE	NumeroEtudiant	IdUV	Note
	85	112	B
	92	119	C
	102	85	A
	112	92	A
	119	102	B
	135	135	A

PREREQUIS	Numero Cours	Numero Prerequis
	CSS3380	INFO3320
	CSS3380	MATH2410
	CSS3380	INFO1310

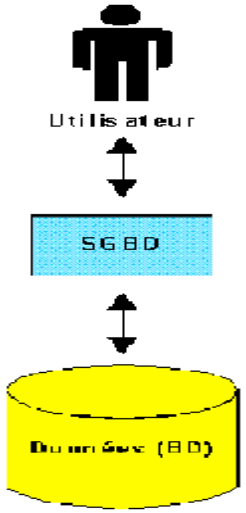
Base de données UNIVERSITE

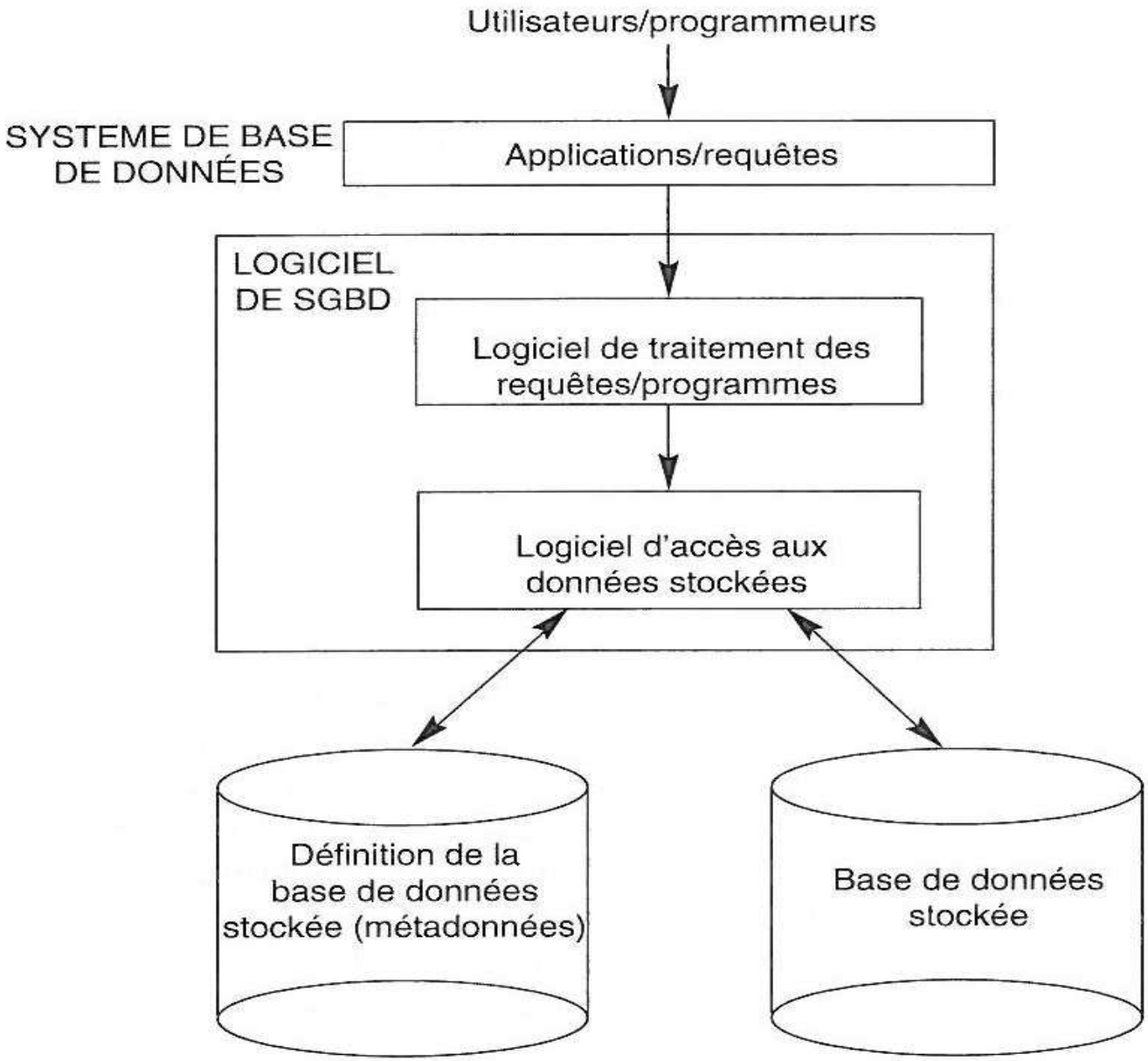
Figure 1.2 • Base de données des étudiants et des cours.

Système de gestion de bases de données

SGBD : un ensemble deprogrammes permettant à des utilisateurs de créer et d'utiliser de BDs. Les activités supportées sont:

- la définition d'une base de données (spécification des types de données, des structures et des contraintes)
- la construction d'une base de données, (stockage des données proprement dites)
- la manipulation des données (ajouter, supprimer, retrouver des données).
- Et le partage d'une base de données (par les utilisateurs et les programmes)





Environnement de système de base de données simplifié

SGBD

- Sépare la description de données, des données elles mêmes...
- Cette description (méta-données) est stockée dans un **catalogue (dictionnaire)** également géré par le SGBD et peut-être consultée par les utilisateurs...
- Indépendance programme-données (abstraction de données)
- Exemple:

Nom du champ	Position de début dans l'enrgistrement	Longueur en octets
Nom	1	30
NumeroEtudiant	31	4
Niveau	35	4
Dominante	39	4

SGBD

- Les SGBD commerciaux les plus connus sont oracle, sybase, ingres, informix, Access et DB2
- Les SGBD non commerciaux les plus connus sont postgres, mysql, sybase sous linux, oracle sous linux...

Fonctionnalités d'un SGBD

- Définition de données
- Manipulation des données
- Contrôler la redondance d'informations
- Intégrité des données
- Partage des données
- Confidentialité des données
- Sécurité de fonctionnement
- Support de persistance

Définition de données

- LDD permet de décrire:
  - des **objets** (personnes, étudiants, voitures..)
  - des **attributs** sur les objets (nom, numéro,..)
  - des **liens** entre les objets (une personne *possède* des voitures)
  - des **contraintes** sur les objets, les attributs, les liens (une voiture n'a qu'un propriétaire)

- **Schéma ..**

Manipulation de données

- LMD
  - Création, Recherche, Suppression, Modification des données
- Interfaces d'accès multiples :
  - Interfaces orientées utilisateur final
    - langages de requêtes déclaratifs comme SQL avec mise en œuvre graphique, interface de type formulaire, ...
  - Interfaces orientées programmeurs d'applications:
    - interface avec des langages de programmation classiques : C, Cobol, Pascal, C++, JAVA, PHP ou "embedded SQL"

Redondance de données

- Problèmes
  - coût en temps,
  - coût en volume
  - et risque d'incohérence entre les différentes copies…
- Supprimer la redondance

Contraintes d'intégrité

Un schéma BD se compose de:

- une **description des données**, de leurs **relations**, ainsi que d'un ensemble de **contraintes d'intégrité**.

Une CI est une propriété de l'application à modéliser.

Les données stockés dans une BD doivent vérifier ces CI..

- 2 types des contraintes:
  - Contraintes **structurelles** : *un employé a un et un seul chef*
  - Contraintes **dynamique**: *un salaire ne peut pas diminuer*

Les SGBD commerciaux supportent automatiquement un certain nombre de contraintes **structurelles**, mais ne prennent pas en compte les contraintes dynamiques (elles doivent être codées dans les programmes d'application)

Les accès concurrents

- Plusieurs utilisateurs peuvent accéder à la même information en même temps.
  - Contrôler les accès concurrents:
    - des techniques de **verrouillage** des données (pour éviter par exemple qu'on puisse lire une information qu'on est en train de mettre à jour).
  - Vue utilisateur

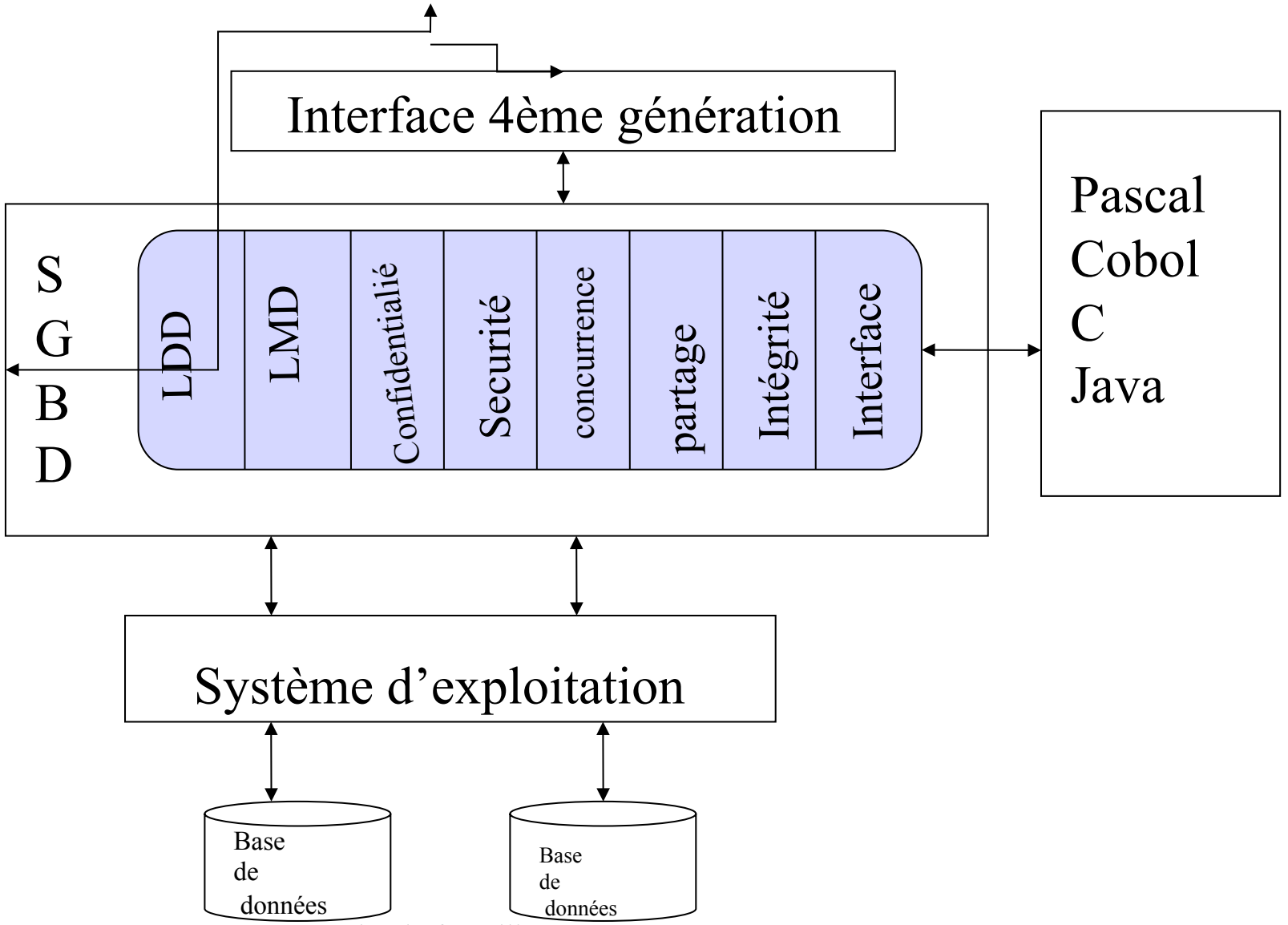
Confidentialité des données

- Multi-utilisateurs
- Problème de la confidentialité des données
- Gérer des droits d'accès sur les données :
  - droits de lecture,
  - de mise à jour,
  - mots de passe

Sécurité de fonctionnement

- Remettre rapidement une BD dans un état opérationnel après un incident hardware ou software
- Journalisation des opérations réalisées sur la BD;
- Ré-exécution automatique en cas de besoin.

Architecture
Fonctionnelle d'un SGBD



Développement d'une base de données

- Méthodes de conception classiques pas adaptées
  - Nouvelles méthodes de conception (MERISE, REMORA,…).
  - Trois niveaux d'abstraction de représentation des données

Niveau conceptuel

Description abstraite et globale du monde réel

- aspect statique (données)
- aspects dynamique (traitements)

=> le schéma conceptuel

Le schéma conceptuel décrit la structure de la base

indépendamment de son implantation

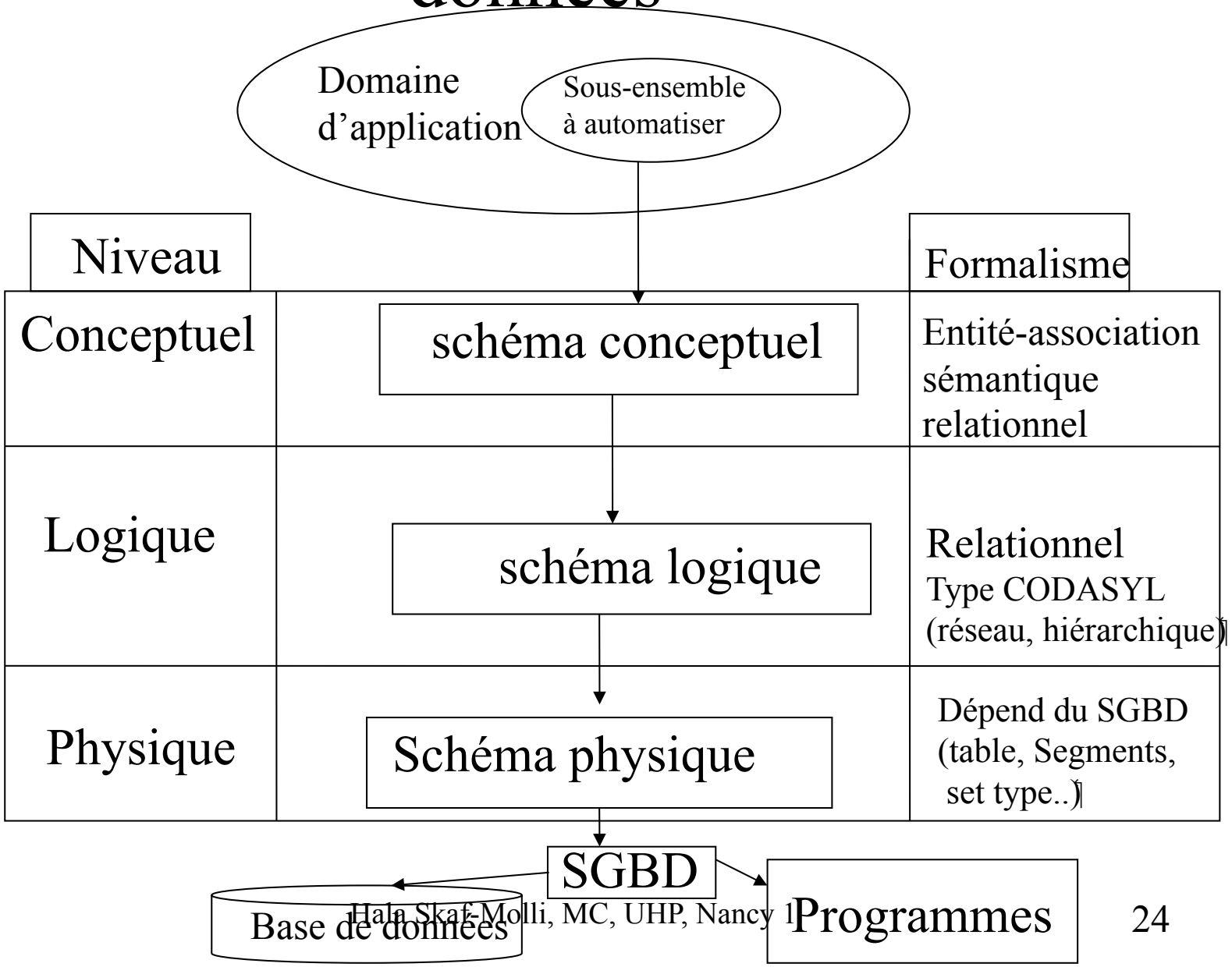
Niveau logique

- Prise en compte de facteurs quantitatifs
  - évaluer le volume de données
  - évaluer les coûts de mise en place de la BD
  - Le schéma logique est issu du schéma conceptuel + [transformations]

Niveau physique

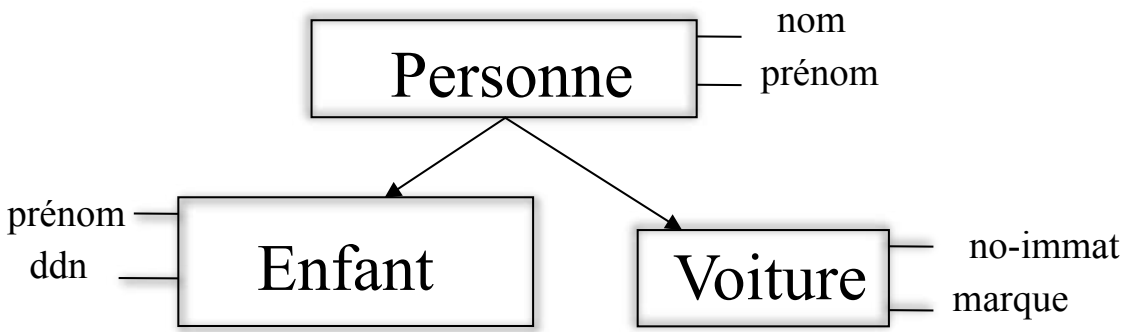
- Contraintes liées au matériel et logiciel
- Adaptation du schéma logique
- Structures de données décrites dans le LDD
- Traitements exprimés avec les outils de manipulation du SGBD

Cycle de développement des bases de données



Modèle de représentation de données: Typologie des SGBD

- Modèle hiérarchique:



- Modèle réseau
- **Modèle relationnel**
- Modèle entité-association
- Modèle Orienté objet

Métiers du domaine

- Utilisaturs de bases de données
  - Les utilisateurs **occasionnels** : technicité moyenne, LMD
  - Les utilisateurs **naïfs**: "presse-bouton "
  - Les utilisateurs **plus spécialisés**: mettre en oeuvre les différents outils du SGBD

Métiers du domaine

- Concepteurs et développeurs:
  - **Concepteur de bases de données**:
    elle identife et strusture les types de données,
    et traitements
    (compétence en BD+conception de logiciel)
  - **Les développeurs d'applications**:
    déterminer les besoin des utilisateurs, implanter les transactions
    et programmes nécessaires
    (compéténce algorithmique et programmation + LMD)

Métiers du domaine

- Administrateur des bases et des systèmes (personne ou équipe de personnes)
  - Droit d'accès à la base
  - Problème de performance
  - Sécurité de fonctionnement

Compétence au-delà de concepteurs de la base de données et de développeurs d'applications

Métiers du domaine

- Réalisateur de logiciel de gestion et de développement de bases de données
  - **Les développeur d'outils**: logiciels facilitant la conception, l'implémentation et l'utilisation de bases de données ou SGBD
  - **Les concepteurs et implémenteurs de SGBD:** compétence en BD, en compilation, en systèmes d'exploitation, en réseaux, ...

Le modèle relationnel de données

- Introduction
- Définition formelle
- Caractéristiques des relations
- Contraintes d'intégrité
- Algèbre relationnelle

Introduction

- 1970 par Codd
- 1ères réalisations (System-R IBM, Ingres Berkeley), vers 1976.
- Les premiers systèmes commerciaux, au début des années 80.
- Le modèle relationnel est **simple**, facile à appréhender, même pour un non spécialiste.
- Solides bases théoriques: définir de façon formelle les langages de manipulation associés.
- Le **modèle relationnel** représente l'information dans une collection de relations.
  - Une **relation** comme une **table** à double entrée, voir même comme un fichier.
  - Chaque **ligne** de la table (appelée **nuplet** ou **tuple**) peut être vue comme un fait décrivant une entité du monde.
  - Une **colonne** de la table est appelée un **attribut**.

Relation et schéma de relation

- **Schéma de relation R**
  - R(A1, A2, ..., An) est un ensemble d'attributs R = {A1, A2, ..., An}; chaque **attribut** Ai est associé un domaine Di
- **Domaine** est un ensemble de valeurs atomiques.
- **Degré** d'une relation est le nombre d'attributs n de son schéma relationnel
- Exemple:

Etudiant(NoSS:chaîne, Nom:chaîne,Tel_dom:chaîne, Age:entier)

Etudiant(NoSS,Nom,Tel_dom,Age);

ETUDIANT

Nom	NumeroEtudiant	Classe	Dominante

COURS

NomCours	NumeroCours	HeuresCredit	Departement

PREREQUIS

NumeroCours	NumeroPrerequis

UV

IdUV	NumeroCours	Trimestre	Annee	Enseignant

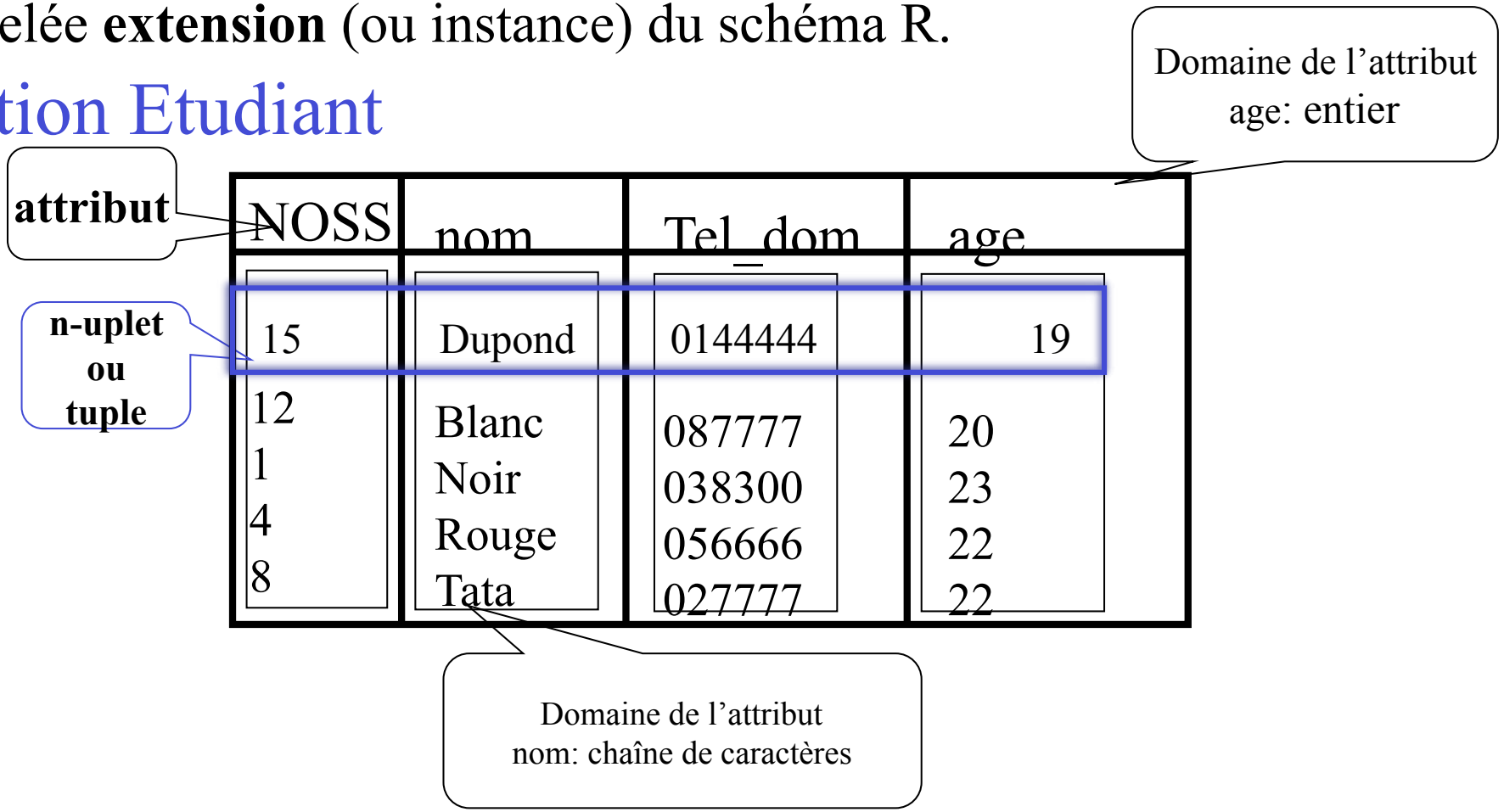
ETAT_NOTES

NumeroEtudiant	IdUV	Note

2.1 • Diagramme schématique de la figure 1.2.

- Une **relation** r sur le schéma de relation R(A1, A2, ..., An) est un ensemble de n-uplets r= {t1, t2, ..., tn}. r est souvent appelée **extension** (ou instance) du schéma R.

Relation Etudiant



Extension d'une relation (ou instance): ensemble de ses n-uplets

DB relationnelle: ensemble de relations (variables dans le temps)

Schéma relationnel: ensemble des schémas des relation de la BD

Caractéristiques des relations

- Une **relation** est un ensemble de n-uplets: pas d'ordre sur les n-uplets,
- Un **n-uplet** est un **séquence ordonnée** d'attributs...
- Une **valeur d'attribut** est **atomique** mais peut être éventuellement nulle (valeur particulière qui indique que la valeur est manquante).

Contraintes d'intégrité

- Schéma de relation R(U,P)
  - U: liste des attributs et leur domaines
  - P: liste des contraintes d'intégrité CI

- A chaque n-uplet est associé une **clé** qui l'identifie de manière unique.
  - Etudiant(**no,nom,prenom,age**), l'attribut **no** est une **clé primaire**
  - Unicité des valeurs de clé (un seul nuplet étudiant peut avoir une valeur de **no** donnée).
  - Une clé primaire ne peut contenir de valeur nulle.
- Les n-uplets d'une relation sont **distincts** deux à deux

Contraintes d'intégrité

- **CI référentielle entre deux relations**.

  – vérifier que l'information utilisée dans un n-uplet pour désigner un autre nuplet est valide, notamment si le n-uplet désigné existe bien.

Employe(no_emp, no_emp, adresse_emp, rôle, no_dept)

Departement(no_dept, nom_dep)

  – un n-uplet de Employe référence un n-uplet de Departement via l'attribut no-dept (numéro de département)

Les langages relationnels

- LDD et LMD
- Deux classes de langage:
  - **langage algébrique**
  - **langage prédicatifs**
- Sur le plan industriel:
  - langage algébrique : **SQL** (structured query language)
  - langage prédicatifs : QBE (Query By Example)

Langage algébrique ou algèbre relationnelle

- Un ensemble d'opérateurs sur des **relations** et produisant de nouvelles **relations**.
- Construire de nouvelles informations à partir des relations de départ et d'une composition séquentielle d'opérateurs.

Algèbre relationnelle

Opérateus unaires	Opérateurs binaires (schémas différents)	Opérateurs binairires (relations de même schéma)
• projection	• produit cartésien	• union
• sélection	• thêta-jointure	• intersection
	• division	• différence

Opérateurs unaires

- Projection

Signature: relation x liste d'attributs → relation

Notation: proj(R,Y) ou $\prod y(R) = S(y)$ tel que:

Définition informelle:

La relation S est formée par les colonnes Y de la relation R.

Base de données utilisées dans les exemples

Produit(prod_id,nom,pu)

Depot(dep_id, adr,volume)

Stock(prod_id,dep_id,qte)

Produit

prod_id	nom	pu
p1	A3	10.0
p2	crayon	9
p3	stylo	15
p4	A4	10.0

Depot

dep_id	adr	volume
1	Nancy	100
2	Laxou	200
3	Vandoeuvre	115
5	Nancy	220
6	Nancy	1000

Stock

prod_id	dep_id	qte
p1	1	0
p3	2	9
p1	3	15
p2	5	20
p3	6	0
p1	5	5
p2	6	2
p3	3	30
p1	2	10

Exemple: Projection

- Donner les noms des produits ?

Sélection

- Signature: relation x expression logique → relation
- Notation: *select (R,E) ou* $\sigma_E(R)$ *ou* $\sigma(R,E)$
- Le résultat a le même schéma que la relation R et chaque n-uplets du résultat vérifie l'expression E.
- E : **Condition booléenne, simple ou composé**
  - **Simple** *nom_attribut op nom_attribut* | *nom_attribut op constante*
  - Op est un opérateur de comparaison (>,>,=,<>,>=,<=)
  - **Composé** si E1, E2 sont des expressions logiques et est un connecteur logique *and or et not* est une expression logique

Exemple: Sélection

- Les produits dont le pu est inférieur à 10?

Opérateurs binaires (schémas différents)

- Produit cartésien x

Signature: relation x relation → relation

R(X) x S(Y) = T(X U Y)

X et Y: ensemble disjoints d'attributs

Définition informelle (extension T)

ensemble des tuples obtenus par concaténation de chaque tuple de R avec chaque tuple de S

- Attention le nombre d'éléments du produit cartésien est le produit des cardinalités des relations R et S!

Exemple : Produit cartésien

- Produit x Stock

Opérateurs binaires (schémas différents)

θ-jointure

Signature: relation x relation x -θ−exp → relation

join $_{\theta E\text{-exp}}$(R,S), join(R,S, θ-exp), R $\theta_{\text{-exp}}$ S

θ-exp porte sur les attributs de R et S **(attributs de jointure)**

θ-jointure : est équivalente à un produit cartésien suivi d'une opération de sélection.

R join $_{\theta E}$ S = $\sigma\theta_{E}$ (R X S)

Exemple : theta-jointure

- Relation qui associe à chaque produit le dépôt dans lequel il est stocké ...

Variantes de jointure

- EQUI-JOINTURE
  - θE-exp porte sur les attributs de jointure est ‘=‘
- jointure naturelle *:
  - Équi-jointure suivie de la suppression des attributs superflus

Opérateurs binaires (schémas différents)

- Division

Signature: relation x relation → relation

div(R,S), $R \div S$

$R(X,Y) \div S(Y) = T(X)$

- Définition formelle:

$T(X) = \{<x> \mid \forall y, <y> \in S \Rightarrow <x,y> \in R\}$

- Définition informelle

T est la projection de R sur X restreinte aux tuples en liaison avec tous les tuples de S

Exemple : Division

- -

Opérateurs binaires (relations de même schéma)

- union $\cup$, intersection $\cap$, différence -

Signature: relation x relation $\rightarrow$ relation

- $R \cup S$, $R \cap S$, $R - S$
- Sémantique identique à celles des opérations ensemblistes
- R, S **ont le même schéma** (même ensemble d'attributs)..
- Exemple:

Exemple

- -

Propriétés des opérateurs

- Union et intersection:

commutatives: $R \cap S = S \cap R$

associatives, $(R \cap S) \cap T = R \cap (S \cap T)$

idempotentes $R \cap R = R$

- Jointure: commutative, associative
- Optimisation de requêtes ...

Résumons

- 8 opérateurs:
  - sélection, projection
  - jointure, produit cartésien, division
  - union, intersection, différence
- Ensemble minimal d'opérateurs:
    - projection, sélection, produit cartésien, union, différence
- La prochaine épisode, le langage SQL !!

Plan

- Les langages des SGBDR
- De l'algèbre relationnelle à SQL

select … from… where

Les langages des SGBDR

- SQUARE --->SEQUEL ---> SQL (Structured Query Language)
- SQL est un standard international (ANSI et ISO).
- Il est connu par tous les SGBDR.
- Désormais la présence de standards internationaux tels que

SQL-86, SQL-89, SQL-92 (SQL2), la dernière SQL-99 (SQL3), chaque SGBD sur le marché utilise un peu son propre dialecte du langage SQL.

Les langages des SGBDR
SQL

- Des primitives de base :
  - Recherche de données, création des relations temporaires..
  - Fonctions d'agrégation (min, max,….)
- Définition et manipulation de schéma de relations (create table, update, insert..)
- Extensions procédurales (create proc,…)

Exemple

Produit(prod_id,nom,pu)

Depot(dep_id, adr,volume)

Stock(prod_id,dep_id,qte)

Produit

prod_id	nom	pu
p1	A3	10.0
p2	crayon	9
p3	stylo	15
p4	A4	10.0
P5	A566	10

Depot

dep_id	adr	volume
1	Nancy	100
2	Laxou	200
3	Vandoeuvre	
115		
5	Nancy	220
6	Nancy	1000

Stock

prod_id	dep_id	qte
p1	1	0
p3	2	9
p1	3	15
p2	5	20
p3	6	0
p1	5	5
p2	6	2
p3	3	30
p1	2	10

Algèbre relationnelle

- Projection

$\Pi_{\text{liste d'attributs}}(\text{nom de relation})$

SQL

select liste d'attributs

from nom de relation

Exemple:

SQL

Attention: pas d'élimination des doubles sauf mention explicite:

select **distinct** pu
from Produit

ou

select **unique** pu
from Produit

AR

Sélection

$\sigma_{\text{condition}}$**(nom de relation)**

SQL

select *
from nom de relation
where condition

- * dénote toutes les colonnes de la relation.
- La condition peut comporter:
  - opérateurs de comparaison: =,!=, >,>=,<,<=, like
  - connecteurs logiques: and, or, not
  - caractères génériques

% peut se substituer à toute **chaîne** de caractères

_ peut se substituer à tout caractère

Exemple: produit dont le pu est < = 10

- Quels sont les produits dont le nom commence par un ‘A’ et dont le prix unitaire est inférieur ou égal à 10 ?
- Réponse (en SQL):

AR

- projection-sélection

Π **l_attributs** (σ **cond**(**nom de relation**))

σ **cond** (Π **l_attributs** (**nom de relation**))

SQL

select l_attributs

from nom de relation

where cond

- Ex: nom et pu des produit dont 9<pu <10 ?

AR

Produit cartésien

jointure sans condition

nom_rel1 x **nom_rel2** x **[…]**

Ex:

Produit x Depot

SQL

select *
from **nom_rel1**, **nom_rel2**, **[…]**

select *
from Produit, Depot

AR

Jointure

nom_rel1 θ**-exp** **nom_rel2** **[...]**

SQL

Forme 1

sans sous-requête

select *

from **nom_rel1**, **nom_rel2**, **[…]**

where θ**-exp**

Produit θ **Produit.prod_id= Stocke.prod_id** **Stock**

SQL

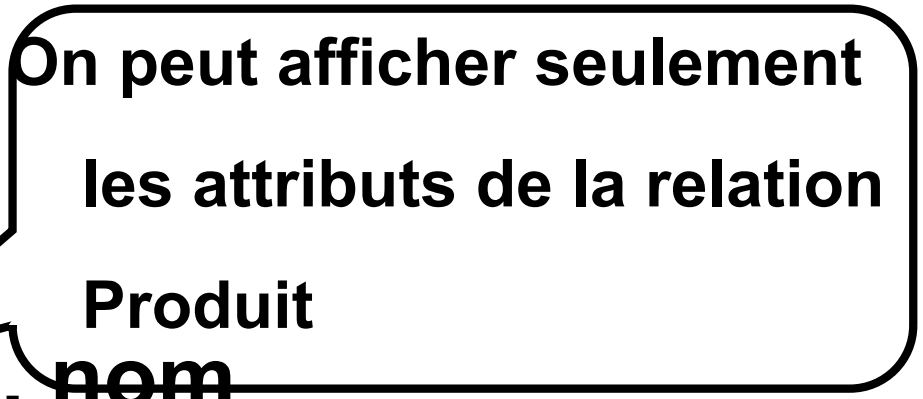
Forme 2

Imbrication de select (ou sous-requête)

Composition jointure, sélection, projection

- Numéros et noms des produits en rupture de stock ?

- En utilisant sous-requête:




```
select prod_id, nom
from Produit
where prod_id in ( select prod_id
                   from Stock
                   where qte <= 0)
```

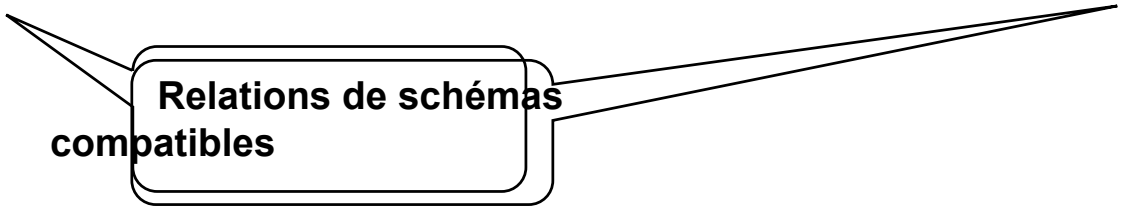
Composition jointure, sélection, projection (suite): *utilisation des alias*

Quels sont les produits en rupture de stock (sortir prod_id, nom, dep_id et adr) ?

Union, Intersection, Différence

Syntaxe:

<clause select> **union** | **intersect** | **minus** <clause select>



Exemple

(select prod_id from Produit where pu > 100)

union

(select prod_id from Produit where pu <50)

Ordonnancement des résultat (order by)

```
select dep_id,prod_id, qte
from Stock
order by prod_id ASC
```

```
select dep_id,prod_id, qte
from Stock
order by prod_id ASC, qte DESC
```

Fonctions d'agrégation

- count: cardinal ...
- sum : la somme des valeurs..
- avg: moyenne...
- max: la valeur maximale
- min: la valeur minimum

- **Ces fonctions peuvent être utilisées dans une clause select**
- **Le résultat est une valeur (pas un ensemble)**
- **Toutes les fonctions ignorent les valeurs nulles (sauf count(*) voir l 'exemple)**

Fonctions d'agrégation: exemples

```
select count(prod_id)
from Produit
-----
3
select count(nom)
from Produit
-----
2
select count(distinct nom)
from Produit
```

Produit

prod_id	nom
p1	n2
p2	NULL
p3	n2

```
    select count(*)   /* donne le nombre des lignes dans une table*/
from  Produit
```

Fonctions d'agrégation: exemples

- Moyenne des pu des produits stockés dans le dépôt 4?

- Le numéro et l 'adresse du dépôt de plus grande capacité ?

  – sinon on peut définir une relation temporaire

Les relations temporaires

- Dans le SGBD Sybase
  - Create table #toto (name type..)
  - **select** liste_attributs **into nom_relation_temporaire**

  from liste_relations

  where condition
- Durée de vie d'une relation temporaire
  - Durée de la session de travail où elle a été créée
- Durée de vie d'une relation
  - Temps écoulé entre sa création et sa destruction
- Toute relation dont le nom débute par # est temporaire: elle est supprimée systématiquement à la fin de session

Les relations temporaires

- Oracle:
  - Table temporaire d'une session:
    - Create global temporary table (nom_col type …..)
  - Table temporaire d'une transaction:
    - Create global temporary table (nom_col type …..) on commit delete rows

Exemples des relations temporaires

- Le numéro et l'adresse du dépôt de plus grande capacité ?

Avec une relation temporaire:

Synthèse: syntaxe générale de recherche

select <liste d'attributs projetés>

from <liste de relations>

where <liste de critères de restriction et de jointure>

Syntaxe générale de recherche

- La partie from décrit les relations qui sont utilisables dans la requête (c'est à dire l'ensemble des attributs que l'on peut utiliser).
- La partie where exprime la (les conditions) que doivent respecter les attributs d'un tuple pour pouvoir être dans la réponse. Cette partie est optionnelle.
- La partie select indique le sous-ensemble des attributs qui doivent apparaître dans la réponse (c'est le schéma de la relation résultat). Bien entendu ces attributs doivent appartenir aux relations indiquées dans la partie from.

Partitionnement d'une relation (ou groupement)
Clause group by

- ***Syntaxe***

group by <liste d'attributs>

- ***Principe***
  - partitionnement horizontal d'une relation, selon les valeurs d'un attribut ou d'un groupe d'attributs qui est spécifié dans la clause **group by**
  - la relation est (logiquement) fragmentée en groupes de tuples, où tous les tuples de chaque groupe ont la même valeur pour l'attribut (ou le groupe d'attributs) de partitionnement

Exemple

- Quelles sont les quantités stockées de chaque produit?

Attention en **SQL standard** chaque expression du **select** doit être **monovaluée** par groupe:

- soit l'attribut de groupement
- soit une fonction sur cet attribut
- soit une fonction qui réduit les valeurs d'un groupe à une valeur unique..

Exemple (suite)

- **Attention en SQL standard:**
  - les attributs après le **groupe by** doivent apparaître dans la clause **select**
  - et les attributs dans **select** doivent apparaître dans **group by** ou bien ils doivent être un argument d'une fonction d'agrégation
- Ce n'est pas le cas de Transact-SQL de Sybase, par exemple...

Exemple de requête erronée en SQL standard

```
select  prod_id, dep_id, avg(qte)
from  Stock
group by prod_id
```

Résultat "attendu"

prod_id	dep_id	Avg(qte)
p1	{1,3,5,2}	10.0
p2	{5,6}	11.0
p3	{2,6,3}	13.0
p4	{2}	0

Requête non valide en SQL standard !!!

L'attribut **dep_id** est multivalué par rapport à l'attribut de partitionnement prod_id

Le résultat n'est pas en première forme normale

Exemple (suite)

Select prod_id, dep_id, avg(qte)

from Stock

group by prod_id

Résultat	prod_id	dep_id	Avg(qte)
p1	1	10.0	
p1	3	10.0	
p1	5	10.0	
p1	2	10.0	
p2	5	11.0	
p2	6	11.0	
p3	2	13.0	
p3	6	13.0	
p3	3	13.0	

Requête valide en Transact-SQL de sybase !!!

Exemple (suite)

Quelles sont les quantités stockées de chaque produit dans tous les dépôts sauf le dépôt numéro 5?

donc, si la requête comporte une clause where: les tuples ne vérifiant pas la condition sont exclus AVANT d'opérer le groupement..

Exemple (suite)

- Quelles sont les quantités en stock de chaque produit dans tous les dépôts de "Nancy"? Trier le résultat sur prod_id?

*Donc, si le résultat doit être trié, la clause **order by** doit apparaître APRES le group by.*

Partitionnement d'une relation (ou groupement) Clause group by

- **Restriction sur les groupes**
  - Application possible d'un critère de restriction sur les groupes obtenus
  - clause **having**
  - l'expression suivant le **having** doit être monovaluée pour chaque groupe

Exemple

- *Quels sont les produits stockés dans plus de 2 dépôts ?*

Exemple

Quels sont les dépôts ayant plus de trois produits en rupture de stock ?

```
 select dep_id, adr
from Depot
where dep_id in ( select  dep_id
                  from Stock
                  where qte <= 0
                  group by dep_id
                  having count(*) >3)
```

➔*éliminer des dépôts ayant qte >0*

➔*groupement sur dep_id*

➔*comptage des membres de chaque groupe*

➔*éliminer des groupes ayant moins de 4 membres*

Restriction

Clause where restriction sur les tuples d'une relation (pas de fonction d'agrégation)

Clause having restriction sur les groupes d'une relation obtenus par la clause group by (on peut utiliser une fonction d'agrégation)

Donner les produits et les sommes des quantités associés, si aucun produit considéré n'a pas de quantité supérieur ou égal à 12" ?

Exemple (suite)

- On peut comparer à la requête suivante :

```
SELECT prod_id, sum(qte)
FROM Stock
WHERE qte <12
GROUP BY prod_id
```

- qui donne pour chaque **produit** la somme des quantités si la quantité est inférieur à 12 (elle correspond à la requête initiale mais travaillant non pas sur la relation **Stock** toute entière, mais seulement les tuples de **quantité** inférieur à 12).

Prédicat d'existence " exists "

- Syntaxe **exists** <sous-requête>
  - Tester si la réponse à une sous-question est *true* ou *false*:
    - *false* si la réponse est l'ensemble **vide**, et *true* sinon
- Q: *donner l'adresse des depots où est stocké le produit «p1» ?*

- Cette requête peut également s'écrire sans utiliser de quantificateur :

Prédicat d’existence "exists "

Adresse des dépôts où n’est pas est stocké le produit «p1» ?

Traduction de la division

- *"Quels sont les numéros et les adresses des dépôts où sont entreposés tous les produits (ceux connus de la base de données)?"*
- PARAPHRASE EN FRANCAIS

"Un dépôt est sélectionné s'il n'existe aucun produit qui n'ait pas été stocké par ce dépôt"

=> DOUBLE NEGATION

Exemple

- -

Exemple (suite)

- Pour ceux n'ayant aucun attrait particulier pour la logique,

voici une solution :

```
SELECT  d. dep_id, d.adr
FROM    Depot d
WHERE   d.dep_id  IN
            (SELECT dep_id FROM Stock
             GROUP BY dep_id
             HAVING COUNT (*)=
                  (SELECT COUNT(*) FROM Produit)
)
```

- Attention, cette écriture n'est correcte que parce que l'on est sûr (par définition) que l'ensemble des produits d'un entrepôt est inclus (ou égal) dans l'ensemble des produits de la base de données.

Synthèse

(6) **select** <liste attributs Aj et/ou expressions sur attributs Ap>
(1) **from** <liste relations Ri>
(2) **where** <condition sur tuples C1>
(3) **group by** <liste attributs Ak>
(4) **having** <condition sur groupes C2>
(5) **order by** <liste attributs Al>

1) Produit cartésien des relations Ri

(2) Sélection des tuples de (1) vérifiant C1

(3) Partitionnement de l'ensemble obtenu en (2) suivant les valeurs des Ak

(4) Sélection des groupes de (3) vérifiant C2

(5) Tri des groupes obtenus en (4) suivant les valeurs des Al

(6) Projection de l'ensemble obtenu en (5) sur les attributs Aj,

avec calcul des fonctions appliquées aux groupes (s'il y en a)

Un exemple complet

- *"Donnez par ordre croissant le nom et la somme des quantités des produits, uniquement si chaque quantité stockée est strictement supérieure à 20?"*

- Requête SQL

```
SELECT p.prod_id, p.nom, Sum(s.qte)
FROM Produit p, Stock s
WHERE p.prod_id = s.prod_id
GROUP BY p.prod_id,p.nom
HAVING  min(s.qte) > 20
ORDER BY p.nom
```

Dictionnaire

Une caractéristiques des SGBD est qu'ils gèrent des «méta-données»

i.e. des données qui décrivent les objets de la base: relation, attributs, vues, utilisateurs connus, mots de passe.

Les métas-données sont elles-mêmes gérées comme une base de données relationnelles, de schéma prédéfini et parfois appelée dictionnaire de SGBD ou catalogue

Dictionnaire de Sybase

Dictionnaire de données

contient les informations relatives à une base des données:

nom de la base, nom des relations, nom et types des attributs..

Les SGDB sybase (comme les autres SGBDR) structure le dictionnaire comme une base de données: on peut interroger cette BD

Dictionnaire de Sybase

Sybase gère un dictionnaire fractionné en
deux parties:
tables communes à toutes les BD (base **master**) :
sysservers, sysdatabases, syslogins…)
tables spécifiques à chaque BD (base **Model**):
sysusers, sysobjects, syscolumns, sysprocedures,..

Dictionnaire d'oracle

- Vue utilisateur: USER_xxx
  - Informations sur les objets dont l'utilisateur (courant) est propriétaire
    - USER_TABLES
    - USER_TAB_COLUMNS
    - USER_VIEWS
    - USER_CONSTRAINTS
    - USER_TRIGGER
- Exemples:

```
Select Table_Name  from USER_TABLES;
Select Column_Name, Data_Type
    from USER_TAB_COLUMNS
    where Table_Name='PRODUIT';
```

Dictionnaire d'oracle

- Vue de la forme ALL_xxx:
  - Informations sur les objets auxquels a accès l'utilisateur.
    - ALL_TABLES
    - ALL_VIEW
- Vues de la forme BDA_xxx
  - Informations dont seul un administrateur peut bénéficier (tous les objets de tous les utilisateurs, config de la base, droits des différents utilisateurs)
- V$: les vues dynamiques de la base
- GV$: les vues de la base en cluster

Dictionnaire d'oracle

- La vue DICTIONARY
  - Ensemble des vues BDA,USER, ALL, V$ du dictionnaire (avant Oracle9i)
  - SQL> desc dictionnary

Nom	NULL?	Type
Table_NAME		varchar(30)
Comments		varchar2(4000)

- Select table_name, comments from dictionnary
- Where table_name like '%FILE%';
- Select * from Dict[IONARY]
  - Toutes les tables et vues accessibles par l'utilisateur.
  - Select * from TAB;
  - Select * from COL;

Nommage des objets

Notion de propriétaire: l'utilisateur qui crée l'objet.

Le propriétaire d 'un objet peut donner des droits (la commande « grant » sous Sybase..)

Nommage

Relation (table)

[[database.]owner.].table_name

(ex: iup2.skaf.Produit)

table_name si elle m'appartient (ex: Produit)

owner.table_name sinon (ex: lui.Produit)

Nommage des objets (suite)

Attribut (colonne)

si pas d'ambiguïté: nom de l ’attribut donné lors de la création de la relation (ex: prod_id)

sinon ***table_name***.*column_name* (ex: Produit.prod_id)

ou *aliasTable.* *column_name* (ex: p.prod_id)

Nom des attributs dans la relation résultat par défaut, ceux de la relation sinon, renommage

select prod_id ``Numero produit ’’

from Produit

La définition de schémas de relations en SQL

- **create table**: création de schéma de relation
- **alter table**: modification de schéma de relation
- **drop table**: suppression de schéma de relation

Rappel:

schéma de relation R(U,P)

U: {<attribut, domaine>}

domaine: type de données

P: {contraintes d'intégrité}

ce qu'on peut exprimer dépend du SGBD

Création d'un schéma de relation

create table nom-table (nom-attr1 type-attr1 [……],

nom-attr2 type-attr1 [……])

- Le type d'un attribut

Chaque SGBD possède des domaines qui lui sont propres

En sybase 11.0.x, (voir sybooks, pour plus de détail..)

- Type de base
  - **int** (entier signé sur 4 bytes),
  - **smallint** (entier signé sur 2bytes),
  - **tinyint** entier non signé sur 1byte)
  - **float**(précision) (sur 8 bytes, la précision dépendant de la machine sur laquelle est installé le système)
  - **real** (sur 8 bytes..)
  - **char**(longueur) (chaine de caractères de longueur fixe, au plus 255 caractères)
  - **varchar**(longueur) (chaine de caractères de longueur variable, au plus 255 caractères)
  - **datetime** (la date de base est 1er janvier 1990….)
  - **money….**

Contraintes d’intégrité

- Une contrainte d'intégrité est une règle qui définit la cohérence d'une donnée ou d'un ensemble de données de la BD

- Quelles contraintes
  - **non nullité** des valeurs d'un attribut
  - **unicité** de la valeur d'un attribut ou d'un groupe d'attributs
  - **check ou validité** restreint les valeurs à insérer dans une table
    - Sur un attribut
    - Sur un ensemble d’attributs …

Contraintes d'intégrité

- **clé primaire** (un attribut ou un groupe)
  - quand : insert or update
  - Quoi faire : rejet de l'opération
- **intégrité référentielle**
  - Dans la 2ème table on référence: unique ou primar key
  - Quand: insert, delete, update
  - Quoi faire:
    - Default policy: rejet des modifications
    - Cascade policy: propager les modifications (delete, update)
    - Set-Null Policy: mettre une valeur nulle lors de (delete, update)
  - On peut choisir entre les différentes stratégies

intégrité référentielle

CONSTRAINT nomContr

FORIGN KEY listeAttributs REFERENCES nomTable(listeAttributs)

[ON DELETE {NO ACTION | CASCADE | SET NULL | SET DEFAULT}]

[ON UPDATE {NO ACTION | CASCADE | SET NULL | SET DEFAULT}]

Modifications des constraintes

- Alter table Produit drop constraint Idestcle;
- Alter table Produit add constraint Idestcle primary key(prodid);

Exemple de création de schémas de relation

- Contraintes d'intégrité:
  - pu est obligatoire dans Produit
  - les numéros de dépôts sont compris entre 1 et 100
  - les numéros des produits commencent par p..
  - CI référentielles existent entre Produit, Stock, Depot

Modification de schémas de relations

- Commande: alter table
  - ajouter de nouveaux attributs, contraintes
  - modifier de définition d ’attributs
- Possibilité de:
  - ajouter d’attributs, de contraintes (add)
  - supprimer de contraintes (drop constraint)
  - modifier de valeurs par défaut d’attributs (replace)
- Restriction:
  - On ne peut pas supprimer un attribut..
  - Un attribut ajouté à l’aide d’alter table n’est pas visible dans les procédures qui font (select *)
    - drop procedure, puis recréer la procédure

Exemples

alter table ***Produit***

add couleur varchar(20) NULL

- Ajout d'une colonne à la relation Produit
- Pour les lignes existantes, on affecte la valeur **NULL**

alter table ***Stock***

add st_id numeric(5,0) identity

- ajout d'une colonne identité à la relation Stock
- Pour les lignes existantes déjà dans la table, le serveur affecte des valeurs en séquence.

alter table ***Stock***

drop constraint stock-cons

- suppression d'une contrainte de la relation Stock

alter table ***Stock***

replace qte defaut null

- modification de la valeur par défaut de l'attribut qte

Suppression d'un schéma de relation

- drop table nom-relation [,nom-relation,…]

- Exemple:

drop table Stock

drop table Produit, Depot

  - suppression du **schéma** de la relation
  - suppression des **tuples** de la relation
  - suppression des **indexes** de la relation

Mise à jour d'une relation

- Insertion d'un ensemble de tuples
  - un seul tuple
  - plusieurs tuples
- Suppression d'un ensemble de tuples
- Modification d'un ensemble de tuples

Insertion de tuples dans une relation

- Insertion d'un seul tuple

insert into Produit values (100, 'vis', 23)

insert into Produit (prod_id, nom) VALUES (100, 'vis')

- Insertion d'un ensemble de tuples

Ex: **create table** Vis
(prod# int, pu float)

insert into Vis
select prod#, pu
from produit
where nom = 'vis '

➡

on peut insérer dans une table le résultat d'une clause **select**

Modification de tuples dans une relation

```
update  Produit
set  pu = 88
where prod_id =150

update  Produit
set  pu = pu * 1.1
where prod_id =150


update Stock
set qte = qte + 10
where prod_id in
      (select prod_id
       from Produit
     where nom = ‘ vis ’)
```

Suppression de tuples dans une relation

Delete [from] *nom-relation [,nom-relation,..]*

[where] *qualification*

- *Supprimer tous les tuples de Produit?*

delete Produit
on supprime tous les tuples mais le schéma de la relation existe toujours..

- *Supprimer le produit de numéro 150?*

delete from Produit
where prod_id= 150

- *Supprimer les produits de numéro <9 ou >12?*

delete from Produit
where prod_id < 9 or prod_id > 12

- *Supprimer les produits stockés au dépôt numéro 5 ?*

Synthèse

- SQL est un langage standard de définition de données LDD (create table,..)
- SQL est un langage standard de manipulation de données LMD (select…)
- Chaque SGBD implante sa propre «version» de SQL
- à suivre !!!