



Cours 1 : Les limites du modèle relationnel

Riadh ZAAFRANI - Rim Samia KAABI

Septembre 2014

3ème Année Ingénieur

Spécialité Génie du Logiciel et des Systèmes d'Information (GLSI)

1

Plan

- Historique
- Avantages du modèle relationnel
- Limites du modèle relationnel
- Le modèle objet relationnel
- Le modèle Orienté Objet
- Le modèle déductif

2

Evolution des SGBD

- 1955-1960: SGF
 - 1960-1980:
 - SGBD hiérarchiques
 - SGBD réseaux
 - 1970-1990
 - SGBD relationnel
 - Langage SQL
 - 1985-1990 :SGBD objet
 - Depuis 1995
 - BD et internet
 - BD et applications décisionnelles
- Les modèles et les systèmes traditionnels :
- très performants dans le développement d'**applications classiques**.
 - présentent des limitations lorsqu'il s'agit d'implémenter des applications plus **complexes** :
 - ❑ Conception et fabrication en génie civil (CAO/FAO et FIO: Fabrication Intégrée par Ordinateur),
 - ❑ les expériences scientifiques,
 - ❑ les télécommunications,
 - ❑ les systèmes d'information géographiques
 - ❑ les applications multimédia, etc.

3

Evolution des SGBD

- Ces nouvelles applications ont des **exigences** et des **caractéristiques** qui diffèrent des applications de gestion traditionnelles :
 - des **objets de structure plus complexe**,
 - des **transactions de plus longue durée**,
 - de **nouveaux types de données** permettant de stocker des images ou des textes volumineux,
 - et la nécessité de définir des **opérations non standard** spécifiques aux applications.

4

Evolution des SGBD

- Les **bases de données orientées objet** ont vu le jour pour répondre aux besoins de ces applications plus complexes.
- Elles permettent au concepteur de spécifier :
 - la **structure d'objets complexes**
 - les **opérations applicables à ces objets**.
- Les fournisseurs de SGBD relationnels ont également reconnu la nécessité d'ajouter d'autres possibilités de modélisation des données :
 - Il en résulte des systèmes qualifiés de **SGBD relationnels-objet ou relationnels étendus (SQL 3)**.

5

ODMG

- À mesure que des SGBD orientés objet devenaient disponibles, la nécessité d'un modèle et d'un **langage standard** se faisait reconnaître.
- Comme la procédure formelle d'approbation d'une norme prend généralement un certain nombre d'années, un consortium de fournisseurs et d'utilisateurs de SGBD OO, l'ODMG (Objet Database Management Group), a proposé un standard connu sous le nom de standard ODMG-93, qui a été révisé depuis.
- www.odmg.org

6

Plan

- Historique
- **Avantages du modèle relationnel**
- Limites du modèle relationnel
- Le modèle objet relationnel
- Le modèle Orienté Objet
- Le modèle déductif

7

SGBDR : avantages

- E. Codd, 1970
- Basé sur la théorie des ensembles
- Indépendant de toute implémentation
- Basé sur la notion de **relation**
- Standard de fait
- Facile à comprendre : "relation = tableau"
- Indépendance entre les programmes d'applications et la représentation interne des données
- Fournit une base solide pour traiter les problèmes d'incohérence et de redondance
- langage déclaratif standard **SQL2**

8

Relationnel : Normalisation

- Processus récursif à partir de la notion de dépendance fonctionnelle permettant d'obtenir des tables qui seront "sans problèmes" lors de l'utilisation.
- L'objectif de cette décomposition est d'aboutir à un schéma conceptuel représentant les entités et les associations canoniques du monde réel.
- Les tables sont dites en 1^{ère}, 2^{ème} ou 3^{ème} FN
 - **Avantage majeur de l'approche relationnelle**

Relationnel : Formes Normales

- **1^{ère} FN** : Une relation est en 1FN si tous les attributs qui la composent sont non décomposables → sans attributs multivalués
- **2^{ème} FN** : Une relation est en 2FN si elle est en 1FN et si toutes les DFs entre la clé et les autres attributs sont élémentaires.
- **3^{ème} FN** : Une relation est en 3FN si elle est en 2FN et si toutes les DFs entre la clé et les autres attributs sont élémentaires et directes.

Relationnel : Formes Normales

■ 1^{ère} FN :

Personne (numP, patronyme)

→ **Personne(numP, nom, prenom)**

→ 2^{ème} FN :

Employé(numE, numP, nomE, temps)

avec numE → nomE et (numE, numP) → temps

→ **Employé(numE, nomE),**

→ **TempsProjet(numE, numP, temps)**

■ 3^{ème} FN :

fournisseur(numF, nomF, NumP, prixP)

avec numF → nomF, NumP et numP → prixP

→ **fournisseur(numF, nomF, NumP), Produit(NumP, prixP)**

11

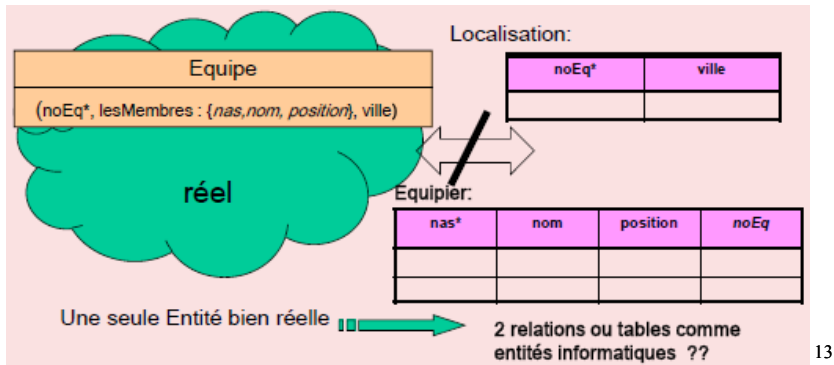
Plan

- Historique
- Avantages du modèle relationnel
- Limites du modèle relationnel
- Le modèle objet relationnel
- Le modèle Orienté Objet
- Le modèle déductif

12

Représentation complexe du réel avec le relationnel

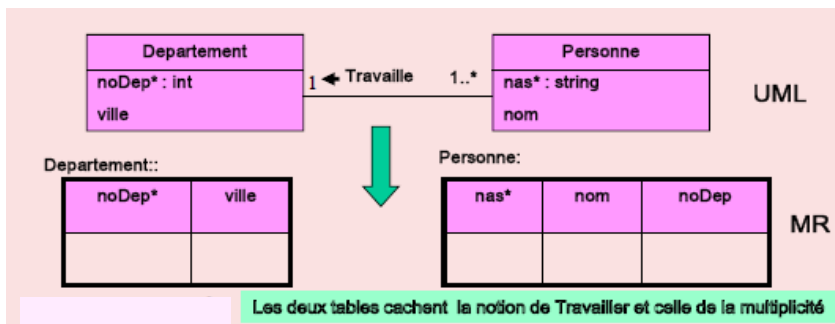
- Représentation conceptuelle distante du réel
- La modélisation fournit de nombreuses relations, en rupture avec le réel, qui est souvent plus simple



13

Surcharge sémantique

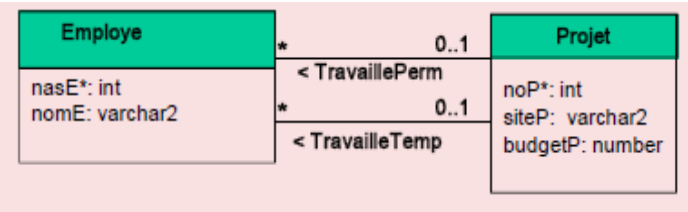
- Les éléments du MR sont souvent **polysémiques**: une table peut représenter à la fois une classe et/ou une association
- Tout le conceptuel est modélisé en relationnel par des relations → Perte du nom de la relation



14

Double association entre deux classes exige un renommage de certains attributs

- Création de nouveaux attributs dans le MR

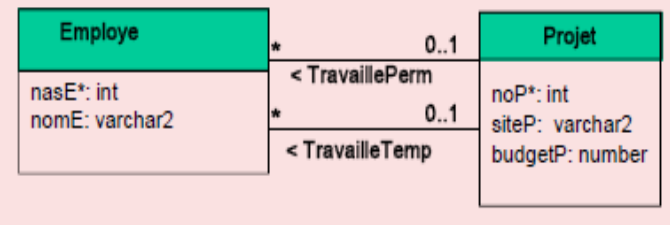


- Le passage en relationnel se fait en créant deux clés étrangères qui réfèrent au noP de la classe Projet mais qui sont absentes dans le DC UML

15

Double association entre deux classes exige un renommage de certains attributs

- Afin d’avoir deux clés étrangères distinctes dans la même table, on est obligé de renommer la clé principale
- → 2 clés étrangères avec un libellé différent
- Employe (nasE*, nomE, noPP, noPT)
- Projet (noP*, siteP, budget)



16

Accès aux tuples par les ordres SQL

- Une table relationnelle correspond seulement à une structure de données. L'accès aux tuples se fait par des select, insert,...sans autres traitements que ceux qui se font au préalable. Les tuples sont traités directement par les applications.
- Exemple: lors d'une insertion, si le noDep d'une personne doit être augmenté de 1000 suite à une fusion de 2 systèmes de gestion distincts des RH (et cela pour éviter les doublons):
 - L'augmentation est faite par un trigger : ok
 - Si trigger absent; l'application doit le faire → source d'erreur

17

Accès aux tuples par les ordres SQL

Departement:	Personne:												
<table><tr><th>noDep*</th><th>ville</th></tr><tr><td></td><td></td></tr></table>	noDep*	ville			<table><tr><th>nas*</th><th>nom</th><th>age</th><th>noDep</th></tr><tr><td></td><td></td><td></td><td></td></tr></table>	nas*	nom	age	noDep				
noDep*	ville												
nas*	nom	age	noDep										

- Autre solution : une table a un type doté d'une interface (signature) adéquate capable de prendre en charge le calcul
- → approche objet

18

Contrainte d'intégrité: entité et référentielle des systèmes relationnels

- Contraintes d'entité et contraintes référentielles sont facultatives.
- Elles peuvent être implémentées dans le MR

```
Create table Personne (  
    nas char(9) not null, nom varchar(45) not null,  
    age number(3) null check (age between 0 and 100), noDep int,  
    noDep number (3) check (noDep is not null),  
    Constraint cp_Personne Primary Key (nas),  
    Constraint fk_PersonneDepartement Foreign Key (noDep) References  
        Departement (noDep) On Delete CASCADE) ;
```

19

Contrainte sémantique élémentaire uniquement

- Contrainte sémantique du domaine absente ou difficile dans le MR

Exemple : inscription seulement des personnes non étudiantes

```
Create table Personne (  
    nas char(9) not null,  
    nom varchar(45) not null Check (nom NOT IN  
        (Select nom From RegistreEtudiant Where statut = 'I')),  
    age number(3) null Check (age between 0 and 100),  
    noDep int,  
    Constraint cp_Personne Primary Key (nas),  
    Constraint fk_PersonneDepartement Foreign Key (noDep) References Departement  
        (noDep) On Delete CASCADE) ;
```

Clause souvent interdite

- Enregistrement des personnes non étudiantes difficile à contrôler! Le CHECK est parfois oublié ou la clause est interdite

20

Structure homogène et atomique du MR

- Le MR assume l'homogénéité horizontale et verticale des tuples d'une même table:
 - **Horizontale**: chaque tuple est composé d'une valeur atomique pour chaque attribut. Absence d'une variabilité du nombre de valeurs pour chaque attribut
 - **Verticale**: les valeurs d'une même colonne proviennent obligatoirement d'un même domaine atomique: syntaxique et/ou sémantique
- Or les attributs dans le monde réel sont plus complexes

Ex.: adresse peut signifier : no, rue, ville, ...
salaire peut représenter l'historique des salaires d'une personne ou le salaire actuel + les 3 précédents.

21

Parcours de l'association conceptuelle par jointure

- Le suivi d'une association se réalise par le calcul d'une jointure de tables → calcul lourd notamment si absence de cluster ou d'index
- Ce suivi est possible via les liens faits par les attributs



- Jointure exige 2 attributs partageant le même domaine sans avoir nécessairement le même libellé

22

Jointure lourde à calculer

- Avec 100 départements et 10000 employés
- Le calcul d'une jointure sans index exige 106 lectures sur un ou plusieurs disques et autant de comparaisons
- Avec 100 tuples par page, 104 lectures avec accès disque seront nécessaires
- Le calcul de la jointure avec au moins un index sur la table la plus volumineuse sera plus rapide
- Le calcul sera encore plus rapide avec deux index. Ces derniers gérant le placement des tuples au moment du stockage ou d'une réorganisation de la base: les valeurs similaires d'attributs qui définissent le cluster sont placées dans le voisinage

23

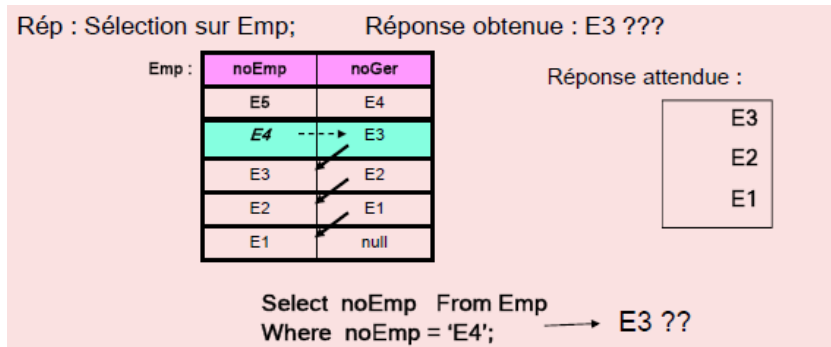
Opérations limitées de l'algèbre relationnelle

- Certaines applications manipulent des types plus complexes et exigent des opérateurs plus appropriés
- Exemple d'opérateurs: distance entre deux points, intersection entre deux aires, inclusion totale/partielle d'une aire dans une autre
- → Solution: redéfinir de nouveaux opérateurs ou mieux des méthodes pour chaque type de données

24

Récurtivité/ déduction difficile voire impossible avec le MR

- Opération de déduction impossible dans le MR
- De quels gérants l'employé E4 peut-il recevoir des instructions?



25

Absence de BLOB

- Plusieurs SGBD relationnels ne gèrent pas les LOB: **BLOB** et **CLOB** (Character large object)
- BLOB : Binary Large Object : valeur en binaire représentant une image, une vidéo
- Le SGBD ignore tout sur le contenu ou sur la structure du BLOB
- Aucun opérateur disponible sur un tel attribut
- BLOB stocké dans :
 - Un fichier externe
 - Un fichier interne à la BD

26

Requête difficile voire impossible sur un BLOB

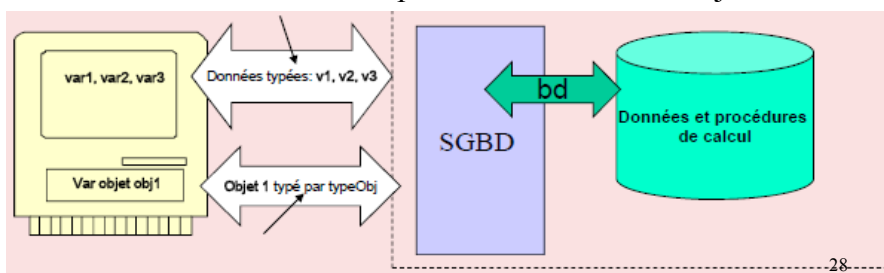
```
Create table Film (noFilm :integer, producteur: varchar (45), video: BLOB ) ;  
  
Select RechercheFrame (video, 350: debut, 6489 :fin)  
From Film  
Where noFilm = 2345;
```

- **RechercheFrame** ne peut être qu'une procédure capable de fouiller une vidéo et d'isoler une séquence de frames (images)
- **Encapsulation impossible**
- → RechercheFrame devrait être un opérateur intégré au langage SQL

27

Echange de données entre l'application et le SGBD

- Peu importe le nombre de tiers entre l'application et le serveur, le calcul d'une requête, l'insertion ou la modification sous-tend en but de course un échange de données :
1. un flot de données typées extraites d'un objet et mises en correspondance avec des variables adéquates et typées du L3G
 2. un objet complet mis en correspondance avec une variable objet du L3G dont la structure est compatible avec celle de l'objet



28

Structure de stockage de données fournie par le SGBD : Compatibilité des types avec L3G

- Toute valeur en provenance du SGBD doit avoir un type (sinon transformée) compatible avec l'un des types prédéfinis dans le L3G de l'application ou construit de toutes pièces dans le L3G
- Tout objet transféré à l'application doit être stocké dans une structure de données adéquate (classe) définie par une variable objet.
- Lorsqu'il y a une nécessité d'une telle transformation, on dit qu'il y a **impedence mismatch**

29

Langage impedence mismatch

- Types simples SQL du MR différents des LP
- Exemple:
 - Date de SQL, absent dans C ou ayant une structure différente dans C++
 - Number de SQL absent dans C
 - Types Long (max 2Go)

30

Absence d'attribut de structure ensembliste

- Personne (nasP, nomP, telE, nasE)
- Avec nasP pour identifier un parent d'un enfant et nasE est celui de l'enfant

Personne:

nasP*	nomP	telE	nasE
111	P. Dion	458-2345	544
112	J. Bois	677-6789	567
112	J. Bois	345-6789	568
567	A. Bois	234-6534	159
544	P. Dion	458-2345	987
568	J. Bois	677-6789	456
987	L. Dion	458-2345	544

Les DF:

nasP → nomP (1)

nasE,nasP → nomP

nasE → telE (2)

telE -> nasP, nasE

Données redondantes => normalisation vers FN3 où tout attribut non primaire ne dépend que d'une clé).

31

Normalisation avec multiplication des tables

EnfantDe :

nasP*	nasE*
111	544
112	567
112	568
544	987
567	159
568	456
987	544

Parent :

(1)

nasP*	nomP
111	P. Dion
112	J. Bois
567	A. Bois
544	P. Dion
568	J. Bois
987	R. Dion

TelEnfant :

(2)

nasE*	telE
544	458-2345
567	677-6789
568	345-6789
159	234-6534
456	677-6789
327	565-6777

Réduction des anomalies, mais

→ multiplication des tables!

32

Normalisation avec multiplication des tables

- Quel est le téléphone des petits enfants de P. Dion?

(A) `Personne (nasP: string, nomP:string, telE :string, nasE : string)`

`Select telE`

`From Personne`

← 1 sélection et 1 table

`Where nomP = "P. Dion " ;`

(B) `Parent(nasP*, nomP) EnfantDe (nasP*, nasE*) TelE (nasE*, telE)`

`Select T.telE`

`From Parent p, EnfantDe e, TelE t`

`Where p.nasP = e.nasP and enasE = t.nasE and p.nomP = "P. Dion";`

2 jointures et 3 tables + 1 sélection

33

Pourquoi cette complexité ?

- Absence d'un attribut de type ensembliste ou structure dans le modèle relationnel
- `Personne (nasP, nomP, enfants{nasE, telE})`
- → attribut complexe qui simplifie la requête

Le téléphone des petits-enfants de P. Dion est fourni par cette simple requête:

réponse:

`Select telE`
`From Personne`
`Where nasP = "P. Dion";`

telE
677-6789
345-6789

34

Relationnel : Résumé des faiblesses

- Absence de pointeurs visibles
 - Jointure par valeurs à éviter: opération lourde et coûteuse
 - Chaînage direct des données: à considérer
- Non support de domaines composés
 - 1FN inadaptée aux objets complexes
 - Introduction des BLOB et CLOB est insuffisante
- Non intégration des opérations

35

-
- L'objet-relationnel et l'orienté-objet
constituent une tentative de réponse
à (certains de) ces besoins
nouveaux

36

L'objet-Relationnel et l'orienté-objet

- C'est un système qui intègre les fonctionnalités d'un SGBDR et les caractéristiques d'un langage objet
- **Modélisation :**
 - intégration des concepts des modèles de données objets complexes
 - conception homogène des données et des programmes concept de classe et notions d'héritage

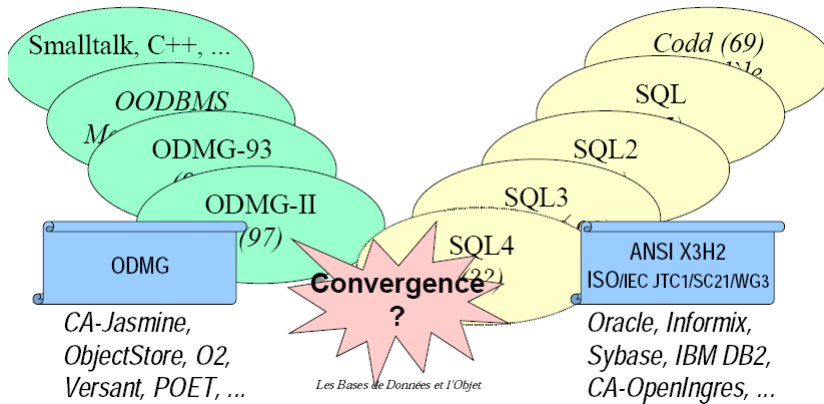
37

-
- **Approches**
 - SGBD Objet
 - SGBD relationnel-objet

38

Deux approches en BD

Le modèle Orienté Objet (OO) Le modèle Objet-Relationnel (OR)



39

Plan

- Historique
- Avantages du modèle relationnel
- Limites du modèle relationnel
- Le modèle objet relationnel
- Le modèle Orienté Objet
- Le modèle déductif

40

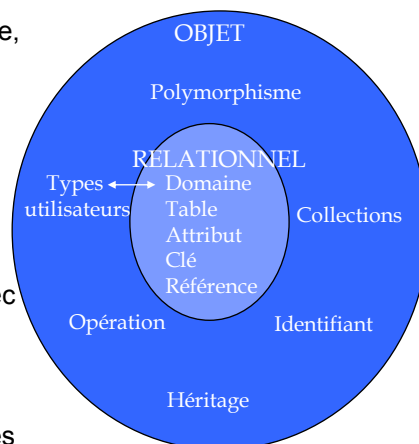
Modèle objet relationnel

- Le modèle objet-relationnel (OR) reprend le modèle relationnel en ajoutant quelques notions qui peuvent être très utiles dans certaines circonstances
- La compatibilité est **ascendante** : les anciennes applications relationnelles fonctionnent dans le monde OR
- La norme SQL99 (**SQL3**) reprend beaucoup d'idées du modèle OR

41

L'objet-relationnel

- **Extension du modèle relationnel**
 - attributs multivalués : structure, liste, tableau, ensemble, ...
 - héritage sur relations et types
 - domaine type abstrait de données (structure cachée + méthodes)
 - identité d'objets
- **Extension de SQL**
 - définition des types complexes avec héritage
 - appels de méthodes en résultat et qualification
 - imbrication des appels de méthodes
 - surcharge d'opérateurs



42

Pourquoi étendre le modèle relationnel ?

- La reconstitution d'objets complexes éclatés sur plusieurs tables relationnelles est coûteuse car elle occasionne de nombreuses **jointures**
- Pour échapper aux éclatements jointures, l'OR réhabilite :
 - les **références** qui permettent d'implanter des structures complexes
 - les **attributs multivalués** (tableaux, ensembles ou listes)

43

Pourquoi étendre le modèle relationnel ?

- SQL92 ne permet pas de créer de **nouveaux types**
→ manque de souplesse et une interface difficile avec les applications orientées objet
- L'OR (et SQL99) permet de définir de nouveaux types utilisateur simples ou complexes (User data type), avec des fonctions ou procédures associées comme dans les classes des langages objet
- L'OR supporte l'héritage de type pour profiter du polymorphisme et faciliter la réutilisation

44

Pourquoi ne pas passer directement aux SGBD OO ?

- Le relationnel a ses avantages, en particulier :
 - sa grande facilité et efficacité pour effectuer des recherches complexes dans des grandes bases de données
 - la facilité de spécifier des contraintes d'intégrité sans programmation
 - une théorie solide et des normes reconnues

45

Pourquoi ne pas passer directement aux SGBD OO ?

- **Inertie de l'existant** : de très nombreuses bases relationnelles en fonctionnement
- Manque de normalisation pour les SGBDOO ;
- trop de solutions propriétaires
- SGBDOO moins souple que le relationnel pour s'adapter à plusieurs applications
- Peu d'informaticiens formés aux SGBDOO
- Le modèle OR peut permettre un passage en douceur

46

Nouvelles possibilités de l'OR

- Définir de nouveaux types complexes avec des fonctions pour les manipuler : **NF2 (non first normal form)**.
- Une colonne peut contenir une collection (ensemble, sac, liste)
- Ligne considérée comme un objet avec un identificateur d'objet (Object Identifier OID)
- Utilisation de références aux objets
- Extensions du langage SQL (SQL3) pour la recherche et la modification des données

47

Exemple

- Type **REGION** = [NOMR : Chaîne,
POPUR : Entier,
DPTS : {
[NOMD : Chaîne,
SUPER : Entier,
POPUD : Entier] }]
=> une relation est définie comme
une variable REGIONS de type {REGION}

48

Les problèmes de l'OR

- Ne s'appuie pas sur une théorie solide comme le modèle relationnel
- Manque de standard de fait : implantations différentes, et encore partielles, dans les divers SGBDs







49

SQL99 (SQL3)

- SQL3 étend le SQL aux concepts Objet.
- Cependant la « **relation** » reste fondamentale dans la manipulation des données
- Les SGBD basés sur SQL3 sont appelés **Objet-Relationnels**
 - Informix, Oracle, Sybase, IBM/DB2, CA-OpenIngres, PostGres ...

50

Exemple de table et objet (Oracle8)

Num	Nom	Adresse	Conducteurs		Accidents		
24	Paul	Paris	Conducteur	Age	Accident	Rapport	Photo
			Paul	45	134		
			Robert	17	219		
					037		

Objet Police

51

Principe

■ Typage fort

Type = Données + Méthodes

- La création de type ne crée pas d'objets
- Les objets d'un type ne sont persistants que lorsqu'ils sont insérés dans des tables déclarées (**CREATE TABLE**)

52

Vue d'ensemble de SQL3

- De multiples facettes :
 - Un langage de définition de types
 - Un langage de programmation
 - Un langage de requêtes
 - Un langage temporel
 - ...
- Pour gérer des données complexes dans le cadre de système objet-relationnel
 - Nouveaux
 - Illustra, UniSQL, ODB II, Versant
 - Relationnels étendus ("universels")
 - Ingres, Oracle, DB2 UDB, Informix

53

Concepts

- **Extensibilité des types de données**
 - Définition de types abstraits
 - Possibilité de types avec ou sans OID
- **Support d'objets complexes**
 - Constructeurs de types (tuples, set, list, ...)
- **Héritage**
 - Définition de sous-types
 - Définition de sous-tables

54

Le modèle SQL3

- Extension objet du relationnel, inspiré de C++
 - type = type abstrait de données avec fonctions
 - fonction
- associée à une base, une table ou un type
- écrite en SQL, SQL3 PSM (Persistent Stored Module) ou langage externe (C, C++, Java, etc.)
 - collection = constructeur de type
table, tuple, set, list, bag, array
 - sous-typage et héritage
 - objet = instance de type référencée par un OID (défaut)
 - association = contrainte d'intégrité inter-classe

55

SQL3 - Les procédures (SQL/PSM) (SQL/Persistent Stored Modules)

- Langage de programmation de procédures
 - déclaration de variables
 - assignation
 - conditionnels CASE, IF
 - boucles LOOP, FOR
 - exceptions SIGNAL, RESIGNAL
 - possibilité de procédures et fonctions externes
- Possibilité de structuration en modules

56

SQL3 - Les Objets

- Extensibilité des types de données
 - Définition de types abstraits
 - Possibilité de types avec ou sans OID
- Support d'objets complexes
 - Constructeurs de types (tuples, set, list, ...)
 - Utilisation de référence (OID)
- Héritage
 - Définition de sous-types
 - Définition de sous-tables

57

SQL3 - Les types abstraits (ADT : Abstract Data Type)

- **CREATE TYPE <nom ADT> <corps de l'ADT>**
- <corps de l'ADT>
 - <OID option> ::= WITH OID VISIBLE
 - objets sans OID par défaut
 - <subtype clause> ::= UNDER <supertype clause>
 - possibilité d'héritage multiple avec résolution explicite
 - <member list>
 - <column definition> : attributs publics ou privés
 - <function declaration> : opérations publiques
 - <operator name list> : opérateurs surchargés
 - <equals clause>, <less-than clause> : définition des ordres
 - <cast clause> : fonction de conversion de types

58

Quelques exemples

■ Un type avec référence

- CREATE TYPE WITH OID phone (country VARCHAR, area VARCHAR, number int, description CHAR(20))

■ Un type sans référence

- CREATE TYPE person (ncin INT, nom VARCHAR, tel phone)

■ Un sous-type

- CREATE TYPE student UNDER person (major VARCHAR, year INT)

59

Les constructeurs de types

■ Les types paramétrés

- possibilité de types paramétrés (TEMPLATE)
- généricité assurée par le compilateur ...

■ Les constructeurs de base

- collections SET(T), MULTISSET(T), LIST(T)
- CREATE TYPE person (cin INT, nom VARCHAR, prénoms LIST(varchar), tel SET(phone))

■ Les références

- possibilité de référencer un objet créé “without OID”
- CREATE TYPE car (number CHAR(9), color VARCHAR, owner REF(person))

■ Les constructeurs additionnels

- stack, queue, array, etc
- non intégrés dans le langage

60

SQL3 - Les fonctions

■ Définition des fonctions

- [<function type>] : CONSTRUCTOR, DESTRUCTOR
- FUNCTION <function name> <parameter list> RETURNS <function results> <SQL procedure> | <file name> END FUNCTION

■ Peuvent être associées à une base, un type, une table, ...

■ Exemple

```
CREATE FUNCTION sell (c Ref(Constructor), amount  
    MONEY) UPDATE Constructor SET total = total +  
    amount WHERE Ref(Constructor) = c  
END FUNCTION
```

61

SQL3 - Les tables

■ Caractéristiques

- une table peut posséder des attributs d'un type abstrait
- un tuple contient des références ou des valeurs complexes
- un attribut peut être de type référence (REF <type> ou with OID)

■ Possibilité d'utiliser un type prédéfini

- CREATE TABLE cars OF car ;

■ Possibilité de définir un nouveau type

- Le type est celui des tuples de la table
- CREATE TABLE Constructors OF NEW TYPE Constructor (name VARCHAR, total MONEY) ;

■ Possibilité de définir des sous-tables

- CREATE TABLE FrenchConstructors UNDER constructors(taxe MONEY)

62

L'appel de fonctions et opérateurs

■ Appel de fonctions

```
SELECT r.name  
FROM emp j, emp r  
WHERE j.name = 'Joe' and distance (j.location,r.location) < 1 ;
```

■ Appel d'opérateurs

```
SELECT r.name  
FROM emp, emp r  
WHERE emp.name = 'Joe' and contained(r.location,  
    circle(emp.location,1)) ;
```

63

Le parcours de référence

■ Possibilité d'appliquer les fonctions Ref et DeRef (implicite)

```
CREATE TABLE cars OF car  
SELECT c.Owner.name FROM cars c WHERE c.color= 'red'
```

■ Possibilité de cascader la notation pointée

```
SELECT dname FROM dept WHERE 1985 IN auto.years
```

■ Généralisation possible aux chemins multiples

```
SELECT dname FROM dept  
WHERE autos.(year=1985 and name = 'Ford')
```

64

Exemple de tables imbriquées

Services							
N°	Chef	Adresse	Employés		Dépenses		
24	Paul	Versailles	Nom	Age	NDep	Montant	Motif
			Pierre	45	1	2600	Livres
			Marie	37	2	8700	Mission
					3	15400	Portable
25	Patrick	Paris	Nom	Age	NDep	Montant	Motif
			Eric	42	5	3000	Livres
			Julie	51	7	4000	Mission

65

Comparaison avec le relationnel

■ Accès en relationnel

```
select effdate, name, vehicleyr
from policy, customers, vehicles
where policy.custno = customers.custno
and policy.vehicleno = vehicles.vehicleno
and model = 'ferrari'
```

■ Accès en objet-relationnel

```
select p.effdate, p.name, p.vehicleyr
from policy p
where p.carmodel.make = 'ferrari'
```

66

Plan

- Historique
- Avantages du modèle relationnel
- Limites du modèle relationnel
- Le modèle objet relationnel
- **Le modèle Orienté Objet**
- Le modèle déductif

67

Introduction

- Un SGBDOO doit d'abord supporter les **fonctionnalités d'un SGBD** :
 - Persistance des objets
 - Concurrence d'accès
 - Fiabilité des objets
 - Facilité d'interrogation
- + un ensemble de **fonctionnalités OO**:
 - Support d'objets atomiques et complexes
 - Identité d'objets
 - Héritage simple
 - polymorphisme

68

Les SGBDOO

- Modèle objet «pur»
 - Persistance
 - langages : C++, Smalltalk, Java / OQL
 - Produits: O2, ObjectStore, Ontos, Objectivity, Jasmine
 - Niches technologiques: CAO, SIG, Gestion de Données Techniques, ...

69

Concept du modèle objet ODMG

- Objet, attribut, association
- Opérations (méthodes), exceptions
- Héritage Multiple
- Clé
- Identité, nommage et durée de vie des objets
- Valeurs (littéraux) atomiques, structurées, collection
- Collections list, set, bag, array
- Transactions, Contrôle de Concurrency, Verrouillage

70

Les standards ODMG

- **Spécification**
 - **ODL** - Object Definition Language
 - **OIF** - Object Interchange Format
- **Manipulation**
 - **Non-Procédural**
 - **OQL** - Object Query Language
 - **Procédural**
 - **OML** - Object Manipulation Language

71

OQL- Object Query Language

- **Langage déclaratif d'interrogation**
 - compatibilité avec l'ordre SELECT de SQL-92
 - proposé par O2 Technology
- **Fonctionnalités générales**
 - tous les types ODMG et identité d'objet
 - requête select ... from ... where ...
 - polymorphisme de collections sources
 - imbrication de requêtes, requêtes nommées
 - appel d'opérateurs et de méthodes
 - opérateurs ensemblistes (union, intersect, except)
 - opérateur universel (for all)
 - quantifieur existentialiste (exists)
 - order by, group by, fonctions d'agrégat (count, sum, min, max, avg)

72

Exercice

```
■ select cl.name, paid
  from ( select x from Companies c, c.clients x
        where c.address.city = "New York"
        ) cl
 where count (cl.orders) > 2
 order by paid: sum ( select o.price from cl.orders o )
```

A quelle question répond cette requête ?

73

Plan

- Historique
- Avantages du modèle relationnel
- Limites du modèle relationnel
- Le modèle objet relationnel
- Le modèle Orienté Objet
- **Le modèle déductif**

74

Concept et motivation

- L'idée est de coupler une base de données à un ensemble de règles logiques qui permettent d'en déduire de l'information.
- Cela pourrait se faire dans le contexte d'un langage de programmation, mais on souhaite un couplage plus direct qui permet de manipuler les données au niveau de la base de données.
- Les questions qui se posent sont les suivantes:
 - ✓ Quelle est la forme des règles que l'on souhaite utiliser?
 - ✓ Comment les interpréter ?
 - ✓ Les requêtes exprimées en SQL peuvent-elles être exprimées par des règles ?
 - ✓ Les règles peuvent-elles exprimer plus ?

75

Définition

- Une base de données déductive (BDD) est constituée :
 - d'un ensemble de **prédicats** (relations), dits de base ou extensionnels, dont l'extension est conservée explicitement dans la base de données: la base de données **extensionnelle**
 - d'un ensemble de prédicats (relations), dits dérivés ou intentionnels, dont l'extension est définie par des règles déductives : la base de données **intentionnelle**

76