Niveaux de description des données ANSI/SPARC

**Niveau externe** Vue utilisateur

**Niveau conceptuel** Description des entités et des relations

**Niveau interne** Description de la façon dont les données sont stockées

Système de gestion de bases de

données (SGBD)

Programme gérant :

-La définition des données

-La manipulation des données

-L’intégrité des données

-La sécurité des données

-La concurrence d’accès

-La résistance aux pannes

-L'indépendance physique

-L’indépendance logique

Modélisation : processus de description permettant de représenter l’information de manière compréhensible.

On utilise un modèle de données : modèle relationnel

Concept de base de données :

-Ensemble **structuré d'information**

-Logiciel **système de gestion de bases de données:**

utilisation de données simultanés parplusieurs utilisateurs et programmes





MERISE : Modèle entités-associations

-Entités

-Attributs (propriétés)

-Identifiants (clefs)

-Associations

-Cardinalités

Dictionnaire de Données

Recense les infos utiles au système

Libellé de la propriété/ Nom du champ/ Type/Dimension/Contrainte

Modèle Conceptuel des Données (MCD)

**-Recherche** des propriétés à gérer

**-Regroupement** des propriétés par entité

**-Construction** des entités

**-Recherche** des associations

**-Recherche** des cardinalités

**-Vérification et validation** du modèle

MCD à MLD

Règle 1 : Toute entité est représentée par une relation. Chaque attribut de l'entité devient un attribut de la relation. L'identifiant est conservé en tant que clé de la relation.

Règle 2 : Toute association qui associe plus de deux entités (ternaire et au-delà) est représentée par une relation.

Règle 3 : Toute association binaire dont les cardinalités maximales sont n de chaque côté est une relation (relation dont les attributs sont les attributs clefs des entités qu'elle relie ainsi que les éventuels attributs propres à l’association).

Règle 4 : Une association de type père - fils, cardinalité maximum à n d’un côté et à 1 de l’autre, n’est pas représentée par une relation.

Normalisation pour éviter :

-La redondance de données

-La perte de données

-Les incohérences

-L'effondrement de performance des traitements

Base de données: les objets

Une table enregistre des enregistrements qui décrivent une instance d’une entité

Les vues sont des résultats d’exploration de données que l’on fait apparaître comme une table.

Un index est une table d’encodage qui optimise l’accès aux données

Un objet User représente un utilisateur des données.

SQL: Structured Query Langage

-LDD : langage de définition de données : CREATE, DROP, ALTER

-LMD : langage de manipulation des données (CRUDE enregistrement)

SELECT, UPDATE, INSERT, DELETE

-LCD : langage de contrôle de données

GRANT, REVOKE

-LCT : langage de contrôle de transactions

COMMIT, SAVEPOINT, ROLLBACK

Une relation est en 1FN si, et seulement si, tout attribut contient une valeur **atomique** (non multiple, non composée).

Une relation est en 2FN si, et seulement si elle est en 1FN et si toutes les dépendances fonctionnelles entre la clé et les autres attributs sont **élémentaires**.

Une relation est en 3FN si, et seulement si elle est en 2FN et si toutes les dépendances fonctionnelles entre la clé et les autres attributs sont **directes**.

Une relation est en BCNF (Forme normale de Boyce-Codd) si, et seulement si elle est en 3FN et si les seules dépendances fonctionnelles élémentaires sont celles dans lesquelles une clé détermine un attribut (et non l’inverse).

|  |
| --- |
|  |



**UPDATE UPDATE**

MLD

Enseignant(id, nom, prénom)

Etudiant(id, nom, prénom)

Salle(id, nbPlaces)

Matière(id, intitulé)

Cours(id, nom, *#idEnseignant,#idMatière, #idSalle*)

Inscription(*#idEtudiant, #idCours*, évaluation)

 Projection Restriction Tri Fonction Agrégat Jointure



R3=R1UR2

SELECT nom, prenom FROM etudiant

**UNION**

SELECT nom, prenom FROM professeur

R3=R1∩R2

SELECT nom, prenom FROM etudiant

**INTERSECT**

SELECT nom, prenom FROM professeur;

R3 = R1 x R2

SELECT \* FROM table1, table2;





 Exemple algèbre relationnelle :

RENOMMAGE (algèbre relationnelle):



Arbre relationnel :

|  |  |  |  |
| --- | --- | --- | --- |
| Jointure avect Pivot | Jointure Interne | Jointure Naturelle | Jointure Externe : LEFT/RIGHT/FULL |
| SELECT DISTINCT **c.nom**FROM cinema c, projection p, film f, jouer j, individu iWHERE **c.num\_cine = p.num\_cine**AND **p.num\_film = f.num\_film**AND **f.num\_film = j.num\_film**AND **j.num\_ind = i.num\_ind**AND **i.nom = 'Travolta'**; | SELECT DISTINCT **c.nom**FROM cinema c**JOIN** projection p **ON c.num\_cine = p.num\_cine****JOIN** film f **ON p.num\_film = f.num\_film****JOIN** jouer j **ON f.num\_film = j.num\_film****JOIN** individu i **ON j.num\_ind = i.num\_ind**WHERE **i.nom = 'Travolta'**; | SELECT DISTINCT **nom**FROM cinema **NATURAL JOIN** projection**NATURAL JOIN** film**NATURAL JOIN** jouer**NATURAL JOIN** individuWHERE **nom = 'Travolta'**; | **SELECT \* FROM tab1****LEFT OUTER JOIN tab2 ON tab1.col11 = tab2.col21;**La jointure externe permet de recuperer les lignes des tables correspondant aucritere de jointure, mais ***aussi*** *celles pour lesquelles il n'existe pas de correspondances* |

Where/Having



**Les vues**

Une vue est une **table virtuelle** de la base de données dont le contenu est défini par une requête SELECT.

-Les données de la vue ne sont pas stockées physiquement.

-Seule la requête décrivant la vue est stockée.

**Avantage** :Sécurité/Simplicité des requêtes

 Simplicité structurelle/ Isolation des modifications

 Intégrité des données

**Inconvénients** : perf. : traduction de requêtes longues/ mises à jours que sur vues simples

Contrôle d’intégrité : **WITH CHECK OPTION**

SELECT num\_cine, COUNT(\*)

FROM Projection

**WHERE** pdate< DATE '2000-01-01'

GROUP BY num\_cine;

SELECT num\_cine, COUNT(\*)

FROM Projection

GROUP BY num\_cine

**HAVING** COUNT(\*) >= 4;

**Les transactions**

**GRANT** <privilege: select, insert, update, delete,

alter, all> **ON** <table> **TO** <utilisateur>

**REVOKE** <privilege> **ON** <table> **FROM** <utilisateur2>

Jalons: **SAVEPOINT** nom\_point;

** **