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 : Méthode d'Etude et de Réalisation Informatique pour les Systèmes d'Entreprise

Concept : séparation des données et des traitements

Niveau conceptuel :

-L’information vue de l’utilisateur : Quoi ? Regle de Gestion

Niveau logique :

-L’organisation logique des données : Qui, ou, quand ? Regle d’organisation

Niveau physique :

-L’organisation du stockage : Comment ?

Modele relationnel :

Principes de l'algèbre relationnelle

-Indépendance de la structure de la base de données avec sa mise en œuvre physique

-Intégrité des données

-Redondance des données

Modèle relationnel :

-Relations + attributs + clés

Base de données relationnelle :

-Tables + colonnes + clés

Méthode d’analyse

-Formaliser une réflexion.

-Garder une trace compréhensible de la réflexion grâce à :

Une démarche admise et utilisée par tous

Des modèles admis et utilisés par tous

-Tendre vers une réalisation adaptée aux besoins

-Fournir des programmes structurés et donc maintenables

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

-Une clé candidate d’une relation est un ensemble minimal des attributs de la relation dont les valeurs identifient à coup sûr une occurrence.

-La clé primaire d’une relation est une de ses clés candidates.Maniere unique d’enregistrement dans table

-Une clé étrangère dans une relation est formée d’un ou plusieurs attributs qui constituent une clé primaire dans une autre relation. Notion d’association

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.

Dépendance fonctionnelle (DF):

On dit que X détermine Y ou que Y dépend fonctionnellement de X ssi, des valeurs identiques de X impliquent des valeurs identiques de Y.On le note : X → Y

Une DF élémentaire est une DF de la forme X → A où A est un attribut unique n’appartenant pas à X et où il n’existe pas X' inclus au sens strict dans X tel que X' → A

Une DF X → A est une DF directe s’il n’existe aucun attribut B tel que l’on puisse avoir X → B et B → A

Normalisation pour éviter :

-La redondance de données

-La perte de données

-Les incohérences

-L'effondrement de performance des traitements

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).

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.

Transaction :ensemble ordonné d'opérations modifiant des données(LMD) qu'un SGBD effectuera complètement ou pas du tout. **(insert, update, delete)**

**Propriété ACID :**

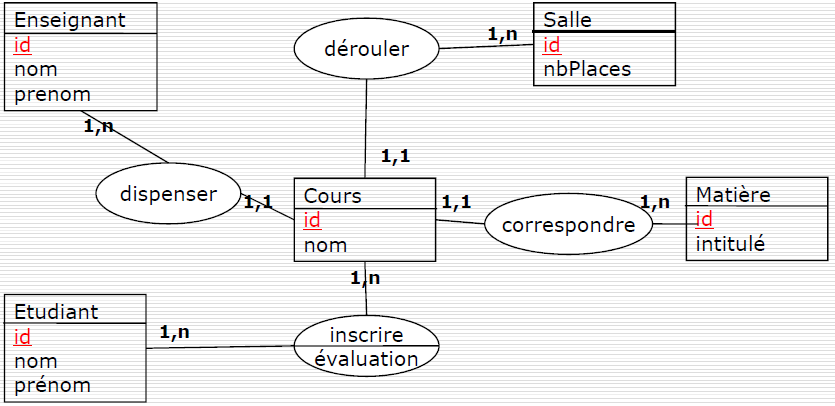
**A**tomicité : une transaction doit soit être complètement validée ou annulée.

**C**ohérence :transaction ne peut pas laisser la base de données dans un état incohérent.

**I**solation : transaction ne peut voir aucune autre transaction en cours d'exécution.

**D**urabilité : après que le client a été informé du succès de la transaction, les résultats de celle-ci ne disparaîtront pas.

|  |
| --- |
|  |



**CREATE TABLE** enseignant(

id NUMBER CONSTRAINT pk\_enseignant PRIMARY KEY,

nom VARCHAR2(30),

nbPlaces NUMBER CHECK (nbPlaces > 0));

**CREATE TABLE** cours(

id NUMBER CONSTRAINT pk\_cours PRIMARY KEY,

nom VARCHAR2(30),

idEnseignant CONSTRAINT fk\_cours1 REFERENCES enseignant(id) ;

**DROP TABLE** personne **;**

**ALTER TABLE** table1 **ADD** (col1 type1, col2 type2, …);

**INSERT INTO** employe **VALUES**('DERAY','ODILE',NULL);

**CREATE TABLE** etudiant **(...,**

classe CHAR(4) DEFAULT 'ING1');

**INSERT INTO** etudiant **VALUES**('DUPONT','EMILE',DEFAULT);

**UPDATE** personne **SET** date\_naissance = DATE '1900-01-01’**WHERE** date\_naissance IS NULL;

**DELETE FROM** personne **WHERE** date\_naissance IS NULL**;**

**UPDATE UPDATE**

MLD

Enseignant(id, nom, prénom)

Etudiant(id, nom, prénom)

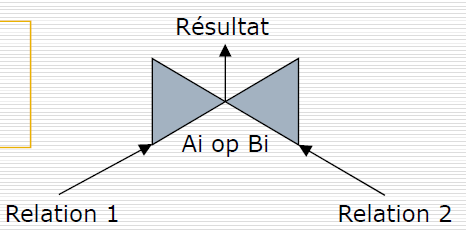
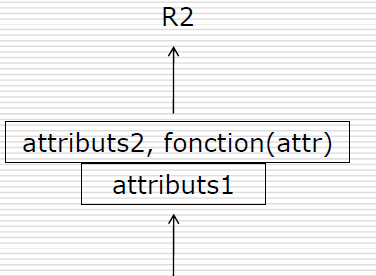
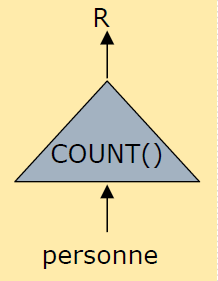
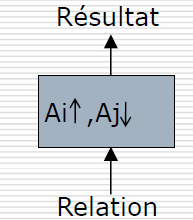
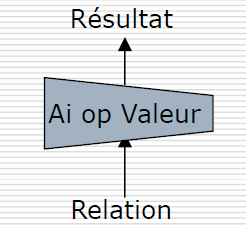
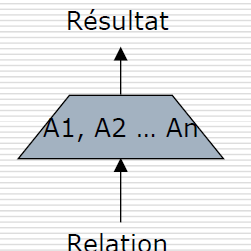
Salle(id, nbPlaces)

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

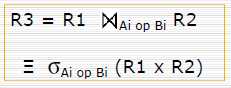
**INTERSECT**

SELECT nom, prenom FROM professeur;

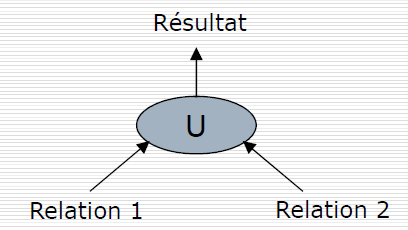
R3 = R1 x R2

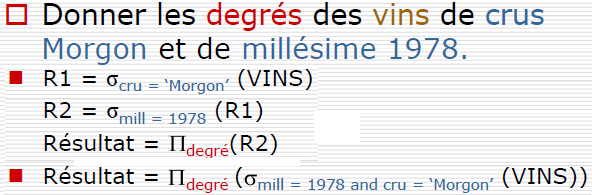
SELECT \* FROM table1, table2;





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 i  WHERE **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** individu  WHERE **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 au  critere de jointure, mais ***aussi*** *celles pour lesquelles il n'existe pas de correspondances* |

Where/Having

**Index**

Permettent une recherche rapide

lorsque le critère n’est pas une clef primaire.

Indépendants de la table. Sont mis à jour automatiquement.

Peuvent concerner plusieurs colonnes (index composé).

Ils ralentissent les mises à jours !

On met en place des indexes sur :

les colonnes utilisées comme critère de jointure, les colonnes servant de critères de sélection, sur une table de gros volume.

**CREATE INDEX** nom\_index **ON** nom\_table (nom\_col [ASC/DESC], nom\_col.. ) ;

**DROP INDEX** nom\_index ;

Index non utilisé dans ces cas :

SELECT \* FROM client ;

SELECT \* FROM client WHERE nom IS NULL ;

SELECT \* FROM client WHERE ca\*10 > 10000;

**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;

**ROLLBACK TO SAVEPOINT** nom\_point;

SELECT \* FROM film film\_lars JOIN individu i

ON film\_lars.num\_ind = i.num\_ind

AND nom = 'von Trier' AND prenom = 'Lars‘

WHERE NOT EXISTS (

SELECT \* FROM individu i JOIN jouer j

ON i.num\_ind = j.num\_ind

WHERE i.num\_ind = acteur\_tous\_lars.num\_ind

AND num\_film = film\_lars.num\_film));