Donner le nom et le prénom des professeurs qui ont suivi des stages de **toutes** les entreprises.   
*SELECT P.nom, P.prénom*

*FROM Professeur P, Stage S*

*WHERE P.id = S.idProfesseur*

*GROUP BY P.nom, P.prénom, P.id*

*HAVING COUNT(DISTINCT.idEnterprise) = (SELECT COUNT(id) FROM Entreprise);*

*----------  
ρ nom, prenom,idEntreprise (Πnom, prenom, idEntrepise(Stage) P.id=idProfesseur Professeur)) ÷ (Πid (Entreprise)*

Donner un arbre algébrique optimisé en utilisant l’heuristique du cours pour la requête SQL suivante qui permet de trouver le titre ainsi que l'année des stages effectués à Paris de l'étudiant Martin Dupont, suivi par des professeurs de département informatique.  
*SELECT titre, année*

*FROM Stage S, Etudiant E, Entreprise EE, Professeur P*

*WHERE E.id = S.idEtudiant*

*AND EE.is = S.idEntreprise*

*AND P.id = S.idProfesseur*

*AND E.prénom = 'Martin'*

*AND E.nom = 'Dupont'*

*AND P.spécialité = 'Informatique'*

*AND EE.adresse LIKE '% Paris%';*

Etudiant(id, nom, prénom, promotion)   
Professeur(id, nom, prénom, spécialité)   
Entreprise(id, nom, adresse,nombreStagesProposés)   
Stage(id, titre, niveau, durée, année, *idEnterprise, idEtudiant,idProfesseur*)

Donner les noms des entreprises dont le nombre

de stages proposés est supérieur à la moyenne; le résultat doit être trié selon le nombre de stages.

*SELECT nom*

*FROM Entreprise*

*WHERE nombreStagesProposés > (SELECT AVG(nombreStagesProposés) FROM Entreprise)*

*ORDER BY nombreStagesProposés;*

Donner le nom et le prénom des étudiants qui n'ont jamais effectué des stages suivis par un professeur en mathématique.

*SELECT E.nom, E.prénom*

*FROM Etudiant E*

*MINUS*

*SELECT E. nom, E.prénom*

*FROM Etudiant E, Professeur P, Stage S*

*WHERE P.id = S.idProfesseur  
AND E.id = S.idEtudiant*

*AND spécialité = 'Mathématique';*

Quel est le prix d'une chambre double de New York Palace le soir du réveillon 31/12/2012 ?

*SELECT prix*

*FROM Hotel H, Chambre C, Tarif*

*WHERE H.id = idHotel*

*AND idChambre = C.id*

*AND nom = ‘New York Palace’ AND tdate = to\_date(‘31/12/2012’,’DD/MM/YYYY’) AND type = ‘double’ ;*

Calculer la note moyenne de chaque hôtel. On veut afficher dans le résultat le nom, la ville et le pays de chaque hôtel ainsi que la note moyenne.

*SELECT nom, ville, pays, AVG(note)*

*FROM Hotel, Evaluation*

*WHERE id = idHotel*

*GROUP BY idHotel, nom, ville, pays ;*

*------------*

*R1 = idHotelFidHotel, AVG(note) (Evaluation)*

*R2 = R1 idHotel = id Hotel*

*Res = Пnom, ville, pays, AVG(note) (R2)*

Donner pour chaque professeur en informatique son nom et le nombre de stages suivis durant l'année 2011, y compris les professeurs qui n'ont pas encadré d'élèves.

*SELECTnom, COUNT(s.id)*

*FROM Professeur P LEFT JOIN (SELECT \* FROM Stage WHERE année = 2011) ON P.id = S.idProfesseur*

*WHERE spécialité = 'Informatique'*

*GROUPE BY nom, P.id;*

Compter pour chaque professeur le nombre de stages dont il est professeur référant. Nous nous intéressons seulement aux stages de niveau 'ING2' et un nombre de stages de ce niveau supérieur ou égal à 2.   
*SELECT nom, prénom, COUNT(s.id)*   
*FROM Professeur P, Stage S*   
*WHERE P.id = S.idProfesseur*   
*AND niveau = 'ing2'*   
*GROUPE BY nom, prénom, idProfesseur*   
*HAVING COUNT(S.id) >= 2;*

-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

Donner le nombre des élèves de promotion 2012 qui ont effectué des stages à Capgemini les années 2010 ou 2011, de durée supérieure à 5 mois.

*SELECT count(E.id)   
FROM Etudiant E, Stage S, Entreprise EE   
WHERE E.id = S.idEtudiant   
AND EE.id = S.idEntreprise   
AND promotion = 2012   
AND EE.nom = 'Capgemini'   
AND annee BETWEEN 2010 AND 2011  
AND durée > 5;  
  
R1 = nom = 'Capgemini' (Entreprise)   
R2 = (durée > 5) AND (année = 2010 OR année = 2011) (Stage)   
R3 = promotion = 2012 (Etudiant)   
R4 = R3 R2 R1   
R5 = FCOUNT(Etudiant.id)(R4)*

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.

*R1 = nom = 'Capgemini' (Entreprise)   
R2 = (durée > 5) AND (année = 2010 OR année = 2011) (Stage)   
R3 = promotion = 2012 (Etudiant)   
R4 = R3 R2 R1   
R5 = FCOUNT(Etudiant.id)(R4)*

Classer en ordre décroissant les hôtels selon le nombre de commentaires laissés par les clients, y compris les hôtels qui n'ont pas eu d'évaluation. On affichera le nom, la ville et le pays des hôtels ainsi que le nombre d’évaluations.

*SELECT nom, ville, pays, COUNT(commentaire)*

*FROM Hotel LEFT JOIN Evaluation ON id = idHotel*

*GROUP BY id, nom, ville, pays*

*ORDER BY COUNT(commentaire) DESC ;*

*--------*

*R1 = HotelJOINTURE idHotel = id Evaluation*

*R2 = id, nom, ville, paysFnom, ville, pays, COUNT(commentaire) (R1)*

*Res = Tri(R2, COUNT(commentaire) DESC)*

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.

|  |
| --- |
|  |

Quel est le client qui a évalué tous les hôtels de la ville Durbuy ? Donner son nom et son prénom.

*R1 = ρidHotel (Пid (σville = ‘Durbuy’ (Hotel)))*

*R2 = Evaluation ÷ R1*

*Res = Пnom,prenom(R2 idClient = id Client)*

*SELECT C.nom, prénom*

*FROM Client C, Hotel H, Evaluation*

*WHERE C.id = idClient*

*AND H.id = idHotel*

*AND ville = ‘Durbuy’*

*GROUP BY idClient, C.nom, prénom*

*HAVING COUNT(idHotel) = (SELECT COUNT(id)*

*FROM Hotel*

*WHERE ville = ‘Durbuy’);*

*SELECT nom, prénom*

*FROM Client C*

*WHERE NOT EXISTS*

*(SELECT \**

*FROM Hotel H*

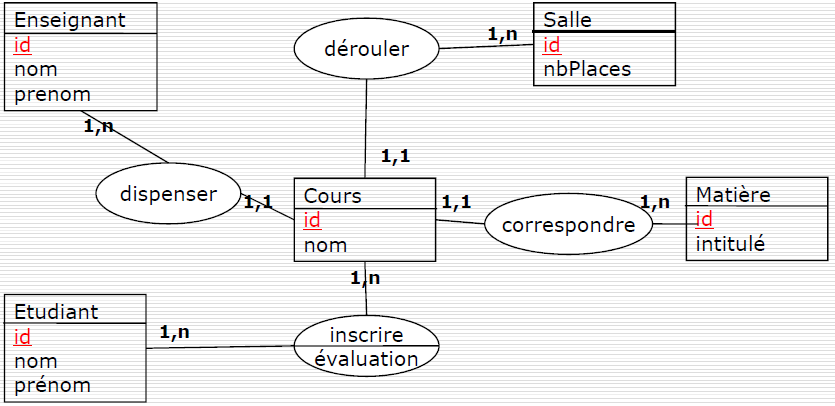
*WHERE NOT EXISTS*

*(SELECT \* FROM Evaluation*

*WHERE C.id = idClient*

*AND H.id = idHotel*

*AND H.ville = ‘Durbuy’));*



.

**UPDATE UPDATE**

MLD

Enseignant(id, nom, prénom)

Etudiant(id, nom, prénom)

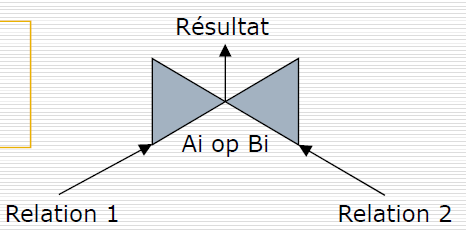
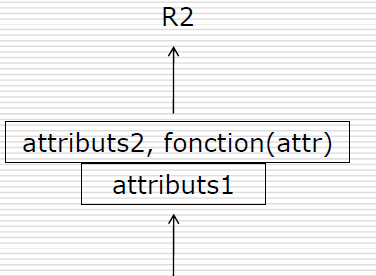
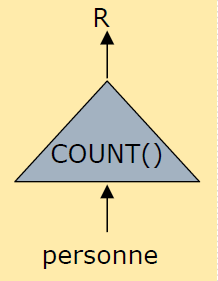
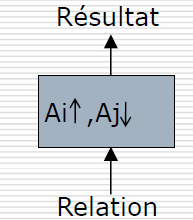
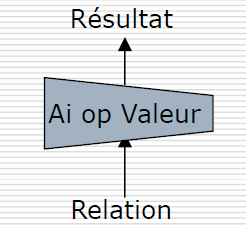
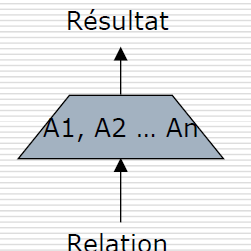
Salle(id, nbPlaces)

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

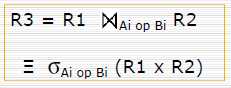
**INTERSECT**

SELECT nom, prenom FROM professeur;

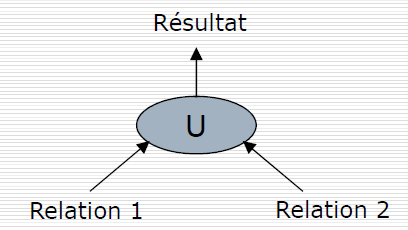
R3 = R1 x R2

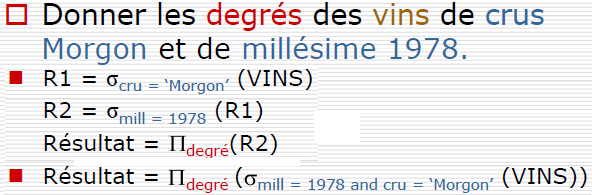
SELECT \* FROM table1, table2;





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));