Systèmes de gestion des bases de données

1.0

DAAS Mohamed Skander

Légende

Table des matières

Objectifs et pré-requis

1

Objectifs

L'implémentation de systèmes complexes nécessite très souvent la création, l'utilisation ou la consolidation de données structurées dans l'optique de les sauvegarder, d'effectuer des recherches ..etc. Au delà des systèmes d'information, tous les domaines ont potentiellement un tel besoin. Ce module apporte un ensemble de compétences dans l'objectif de concevoir et d'interroger des bases données.

Le module vise à préparer les étudiants à la conception et l'administration de base de données relationnelle. L'accent est mis sur :

L'utilisation d'une méthodologie de conception de base de données.

La maîtrise des éléments d'architecture logique et physique d'une base de données relationnelle.

Pré-requis

Aucun prérequis n'est nécessaire à ce cours sauf quelques notions de bases sur la théorie des ensembles et les expressions logiques.

Pour pouvoir créer ou interroger une base de données, vous devez dans un premier temps installer un système de gestion de base de données (MySQL, PostgreSQL, Access ou autres). Il en existe beaucoup.

Introduction

II

Jusqu'en 1960 les informations étaient enregistrées dans des fichiers manipulées par les logiciels applicatifs. L'idée des bases de données a été lancée en 1960 dans le cadre du programme Apollo. Le but était de créer un dispositif informatique destiné à enregistrer les nombreuses informations en rapport avec le programme spatial, en vue de se poser sur la lune avant la fin de la décennie. C'est dans ce but que IBM, conjointement avec Rockwell met sur le marché le logiciel Information Management System (IMS). Avec ce SGBD, les informations sont enregistrées dans des bases de données organisées de manière hiérarchique.À la même époque, General Electric, avec l'aide de Charles Bachman met sur le marché le logiciel Integrated Data Store. Avec ce SGBD les informations sont enregistrées dans des bases de données organisées selon un modèle réseau, ce qui permet d'enregistrer des informations ayant une organisation plus complexe que le modèle hiérarchique.En 1965, Dick PICK développe le système d'exploitation Pick, qui comporte un SGBD et le langage Databasic de Charles Bachman. En 2002 la technologie de Pick est utilisée dans des produits contemporains tels que JBase.En 1967, le consortium CODASYL forme un groupe de travail, le database task group abr. DBTG, qui travaille à la normalisation de deux langages informatique en rapport avec les bases de données: le DML et le DDL.Les organisations hiérarchiques et réseau des années 1960 manquaient d'indépendance vis-à-vis du format des fichiers, ils rendaient complexe la manipulation des données et il leur manquait une base théorique. En 1970 Edgar Frank Codd, employé de IBM publie le livre A relational model of data for large shared data banks, un ouvrage qui présente les fondations théoriques de l'organisation relationnelle. Sur la base des travaux de E.F Codd, IBM développe le SGBD System R, qui sera mis sur le marché à la fin des années 1970. Il est destiné à démontrer la faisabilité d'un SGBD relationnel. Le langage informatique propre à ce SGBD est le Structured Query Language (abr. SQL), défini par IBM et destiné à la manipulation des bases de données relationnelles.Charles Bachman reçoit le prix Turing en 1973 pour ces contributions à la technologie des bases de données et Edgar Frank Codd reçoit le prix Turing en 1981 pour les mêmes raisons.En 1978, ANSI publie la description de l'architecture Ansi/Sparc qui sert de modèle de référence en rapport avec l'indépendance des données des SGBD.Les deux SGBD ténors du marché de 2010 que sont IBM DB2 et Oracle Database ont été mis sur le marché en 1979 et sont tous deux basés sur le modèle relationnel. La même année le langage SQL est normalisé par ISO.Les moteurs de recherche et les datawarehouse sont des applications informatiques apparues dans les années 1990, qui ont influencé le marché des SGBD. Les moteurs de recherche ont nécessité le traitement d'informations non structurées et écrites en langage naturel. Et les datawarehouse

ont nécessité la collecte et la consolidation de très grande quantités d'informations en vue de réaliser des tableaux de synthèse.Les modèles d'organisation orienté objet et objet-relationnel sont apparus dans les années 1990. Les premiers SGBD objet-relationnel ont été Postgres, Informix et Oracle Database en 1995. Le standard relatif au langage SQL a été modifié en 1999 pour pouvoir s'appliquer à ce type de SGBD.

A. Base de données

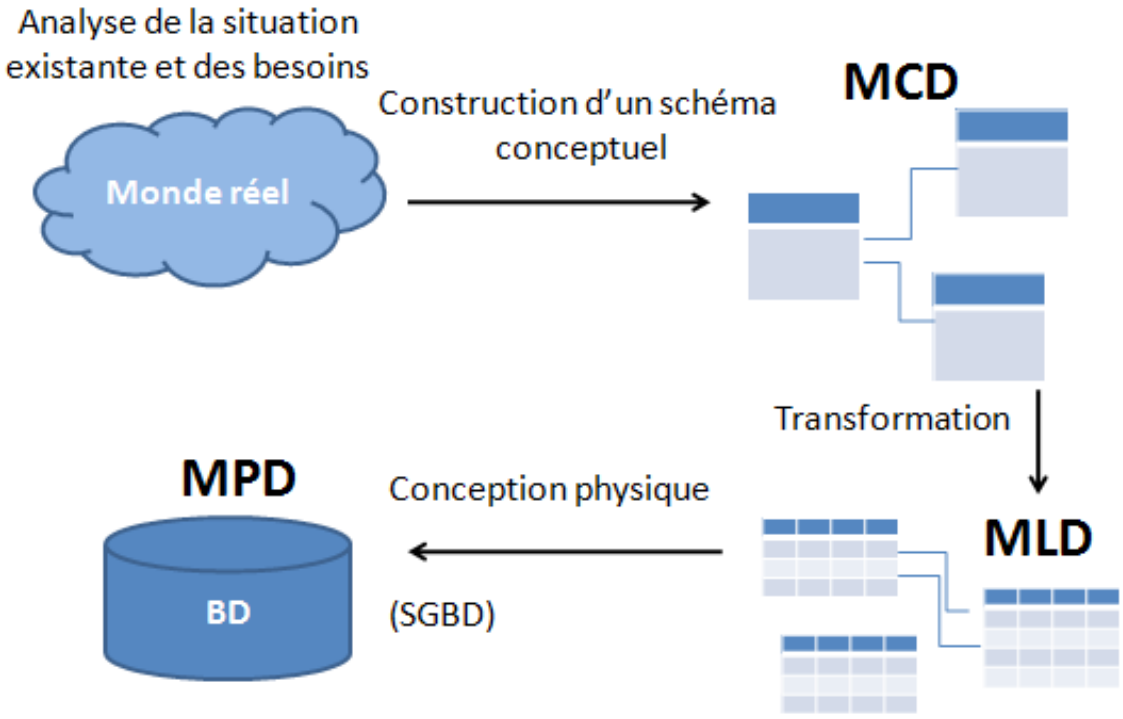
Une base de données est une collection de données opérationnelles enregistrées sur un support adressable, ces données sont utilisées par des systèmes d'application d'une organisation particulière, en outre la collection des données est structurée indépendamment d'une application particulière, une base de données est cohérente, de redondance minimale et accessible simultanément par plusieurs utilisateurs.Il existe plusieurs types de base de données : le modèle hiérarchique, le modèle réseaux sémantiques, le modèle objet et le modèle relationnel qui est le plus répondu.

B. Système de gestion de bases de données

C'est un ensemble de programmes assurant : la structuration, le stockage, la maintenance, la mise à jour, la recherche, la confidentialité et le contrôle de l'intégrité des données dans une base de données. Il existe plusieurs systèmes de gestion de base de données : Oracle Database, MySQL, PostgreSQL, Microsoft SQL Server, Microsoft access,DB2, … etc.

Etapes-type de construction d'une base de donnée :

1. Spécification et analyse: Analyse de la situation existante et des besoins.
2. Construction d'un schéma conceptuel (modèle conceptuel de données).
3. Transformation du schéma conceptuel en un schéma logique (modèle logique de données) : modéliser la structure selon laquelle les données seront stockées dans la future base de données.
4. Conception physique (modèle physique de données) : Implémenter la base de données dans un SGBD.



Etapes-type de construction d'une base de données

Chapitre 01 : Modèle conceptuel de données (Modèle Entité-Association)

Le Modèle Conceptuel de Données (MCD) a été introduit dans les années 70, c' est une représentation du système d'information du point de vue des données. Il a pour objectif de constituer une représentation claire et cohérente des données manipulées en décrivant leur sémantique et les rapports qui existent entre elles. le MCD est plus facile à lire pour la construction des bases de données. Les règles de construction du MCD permettent d'aboutir à une représentation graphique standard qui élimine les redondances et les ambiguïtés.

A. Schéma Entité-Association

Un schéma Entité-Association Est un schéma conceptuel de données, ses concepts de base sont : l'« *entité* », l'« *association* », les « *attributs* » et les « *cardinalités* ».

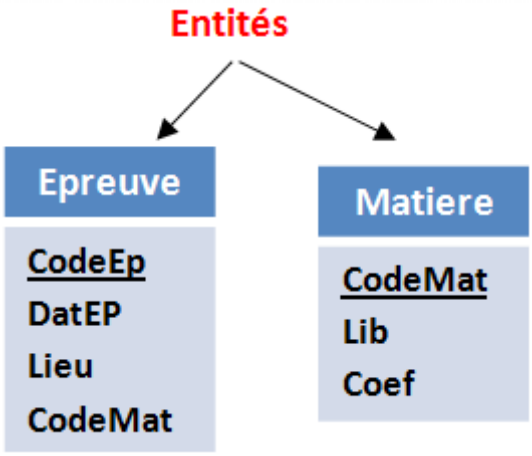
B. Entité

Définition : Entité

Une entité est la représentation d'un élément concret ou abstrait du monde réel perçu. Dans une entité, on met les informations nécessaires et suffisantes pour caractériser cette entité (son nom, ses propriétés et son identifiant).

Exemple

Entité



Entité Matière et entité Epreuve

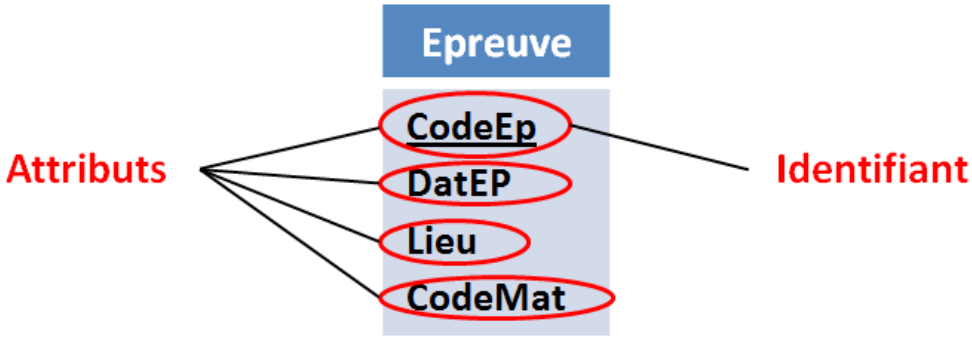
C. Attributs et valeurs

Définition : Attributs et valeurs

Les propriétés d'une entité sont appelées attributs, ces attributs prennent des valeurs pour chaque occurrence d'une entité.

Exemple

Attributs



Attributs

D. Identifiant

Définition : Identifiant

Un identifiant d'une entité est un ensemble de ses attributs (un ou plusieurs) permettant d'identifier de manière unique chaque occurrence de cette entité.

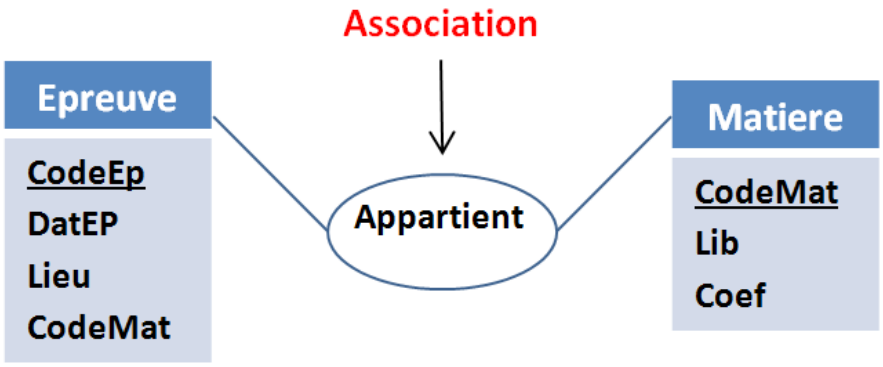
E. Association

Définition : Association

Une association (appelée aussi parfois relation) représente les liens sémantiques qui peuvent exister entre les différentes entités. Une association possède un nom (souvent un verbe) et parfois aussi des attributs.

Exemple

Association



Association Appartient

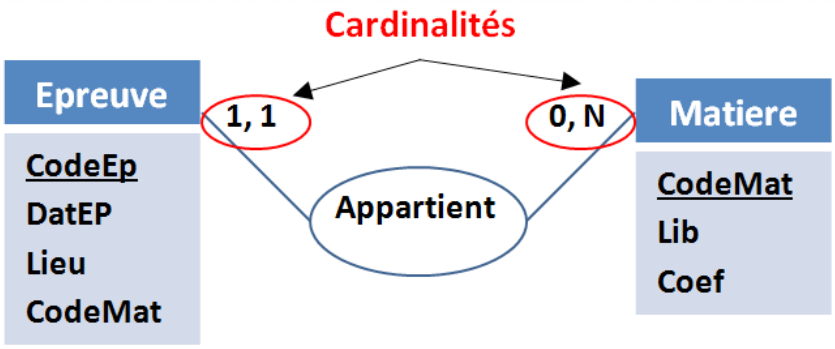
F. Cardinalités

Définition : Cardinalités

Les cardinalités d'une entité dans une association décrient le nombre de fois minimum et le nombre de fois maximum qu'une occurrence de cette entité est impliquée dans une occurrence de l'association.

Exemple

Cardinalités



Cardinalités de l'association 'Appartient' pour les entités 'Matiere' et 'Epreuve'

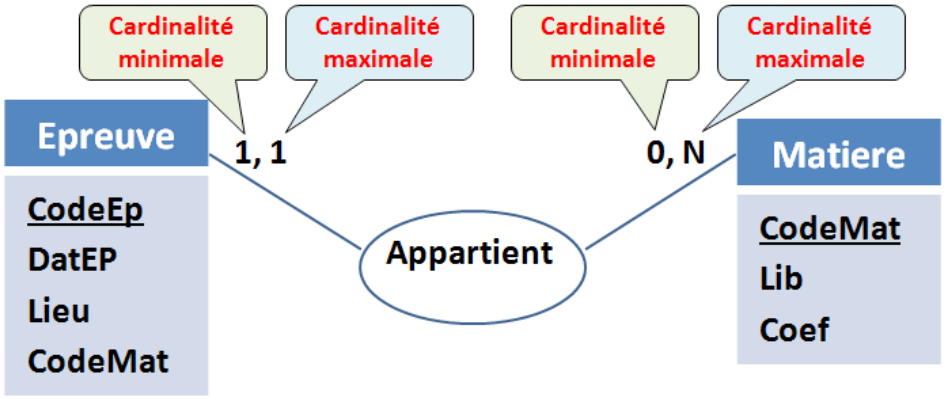
Complément

Pour chaque patte, poser les deux questions :

- Pour n'importe laquelle des occurrences de l'entité, peut-il y avoir 0 occurrences de la relation, ou doit-il y en avoir au moins une ?
- Pour n'importe laquelle des occurrences de l'entité, peut-il y avoir n occurrences de la relation, ou doit-il y en avoir au plus une ?

Les valeurs possibles des cardinalités sont (Cardinalité **minimale**, Cardinalité **maximale**): **(0,1)**, **(1,1)**, **(0,N)** et **(1,N)**.

Exemple



cardinaliteminmax

Cardinalités (Min-Max)
Une epreuve appartient à une et une seule matière

- cardinalités **1,1**

Une matière a pour épreuve aucune ou plusieurs épreuves

- cardinalités **0, n**

G. Les différents types d'association

A - Les associations binaires

Une association binaire est une association entre deux entités

A - Les associations réflexives

Une association réflexive est une association qui lie une entité à elle même.

A - Les associations n-aire

Une association N-aire est une association qui lie n entités.

H. Notations

On dit qu'une association binaire est de type :

- **1:1** : Si la valeur maximale des cardinalités des deux cotés est 1.
- **1:N** : Si la valeur maximale des cardinalités dans un coté est 1 et la valeur maximale des cardinalités dans l'autre coté est N.
- **M:N** : Si la valeur maximale des cardinalités des deux cotés est N

I. Exercice

Exercice

Voici les règles de gestion du système d'information d'un café : Les serveurs sont caractérisés par un numéro et un prénom. Chaque table est numérotée et est gérée par un serveur (il ya plus de tables que de serveurs). Les serveurs prennent les commandes. Une commande correspond à une table bien précise. Une commande est définie par un numéro, une date, une heure et un montant TTC. Une commande contient une ou plusieurs consommation(s). Une consommation quelconque peut être commandée en plusieurs exemplaires. Chaque consommation est définie par un numéro, un libellé et un prix unitaire.

- Donner le modèle entité-association de ce système.

  Solution (cf. MCD p 53)

IV

Chapitre 02 : Modèle logique de données (Modèle relationnel)

A. Modèle logique de données (Modèle relationnel)

Le modèle relationnel est basé sur une organisation des données sous forme de tables. La manipulation des données se fait selon le concept mathématique de relation de la théorie des ensembles, c'est-à-dire l'algèbre relationnelle. L'algèbre relationnelle a été inventée en 1970 par E.F. Codd, le directeur de recherche du centre IBM de San José. Elle est constituée d'un ensemble d'opérations formelles sur les relations. Les opérations relationnelles permettent de créer une nouvelle relation (table) à partir d'opérations élémentaires sur d'autres tables (par exemple l'union, l'intersection, ou encore la différence).

B. Domaine

Définition

Un domaine représente l'ensemble des valeurs qui peuvent être prises, un domaine est défini par :

- Un nom
- Une description logique
- Un type de données

Exemple

- Age des étudiants : ensemble des entiers entre 18 et 35
- Le domaine des booléens : {0,1}
- Le domaine des doigts de la main : {pouce, index, majeur, annulaire, auriculaire

C. Relation (table)

Définition

Sous ensemble du produit cartésien(*) d'une liste de domaines caractérisé par un nom.

D. Attribut

Définition

Colonne d'une relation caractérisée par un nom.

Exemple

Attribut

CodeEp	DatEp	Lieu	CodeMat
11	01/02/2015	Amphi1	Mat
12	19/01/2015	Amphi1	Phy
13	18/01/2015		Chi
14	18/01/2015	Amphi1	Inf
16	21/01/2015	Amphi1	Phy

Attributs

Attributs : CodeEp, DatEp, Lieu, CodeMat

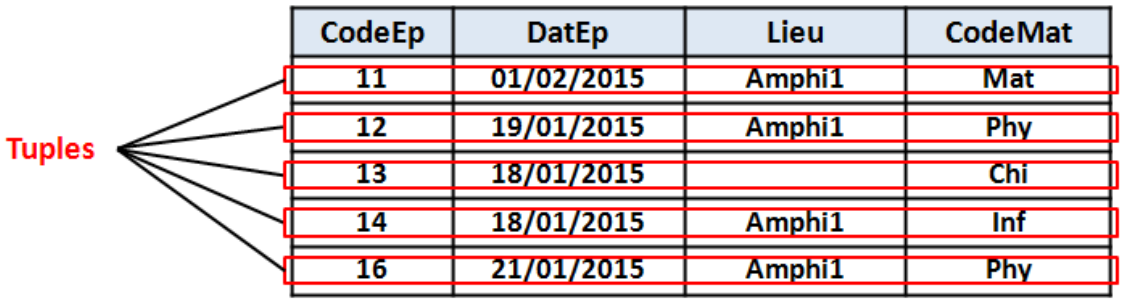
E. Tuple

Définition

Ligne d'une relation correspondant à un enregistrement.

Exemple

Tuples



Ensemble de tuples

F. Schéma relationnel

Définition

Un schéma relationnel R est le nom de la relation suivi de la liste d'attributs et la définition de leurs domaines. A1, ..., An

Syntaxe

R(A1:D1, ..., An:Dn)

Remarque

Le degré d'un schéma relationnel est le nombre de ses attributs

G. Concept de Relation

Définition

Une relation de schéma R(A1:D1, ..., An:Dn) est donc un ensemble de n-uplets

Exemple

Relation

CodeEp	DatEp	Lieu	CodeMat
11	01/02/2015	Amphi1	Mat
12	19/01/2015	Amphi1	Phy
13	18/01/2015		Chi
14	18/01/2015	Amphi1	Inf
16	21/01/2015	Amphi1	Phy

Relation

H. Notion de clé primaire

Définition

Il existe un ensemble d'attributs **K** de **R** tel que pour deux tuples **t1** et **t2** quelconques de **R**, on ait **t1[K]≠t2[K]** On appelle **K** une *« clé candidate »*

La *« clé candidate »* choisie pour identifiant est appelée **Clé primaire**.

Par convention, les attributs composant la clé primaire sont **soulignés** (Dans ce cours les clés primaires sont mises en italique)

I. Notion de clé étrangère

Définition

C'est une clé qui représente la valeur de la *« clé primaire »* d'une autre *« relation »*.

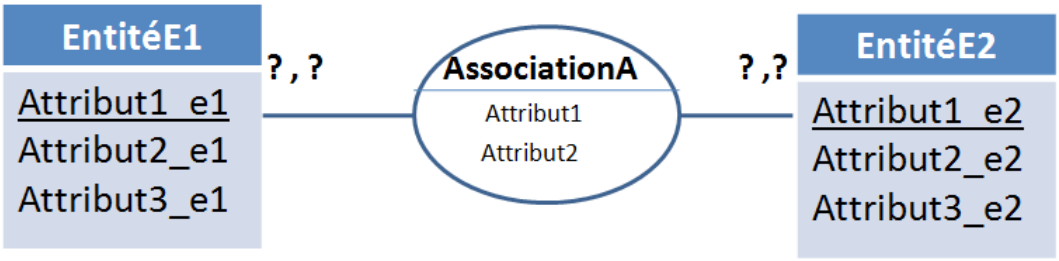
Par convention, la *« clé étrangère »* est précédée du caractère **#**

J. Règles de passage du modèle entité association au modèle relationnel

1. Règle 1 : (entité)

Chaque entité donne une nouvelle relation dont la clé primaire est l'identifiant de l'entité.

Exemple : Entité-association



Entité-association

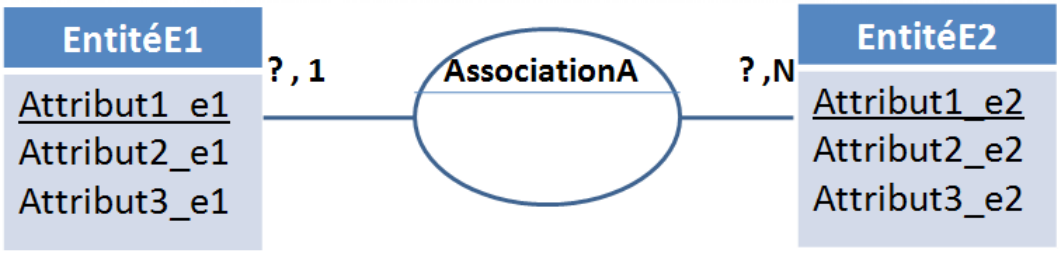
Modèle Relationnel:

- **EntiteE1**(*attribut1_e1*, attribut2_e1, attribut3_e1)
- **EntiteE2**(*attribut1_e2*, attribut2_e2, attribut3_e2)

2. Règle 2 : (Association de type 1-N)

La clé primaire de l'entité côté N est ajoutée à la relation de l'entité côté 1 où elle devient clé étrangère.

Exemple



Entité-association

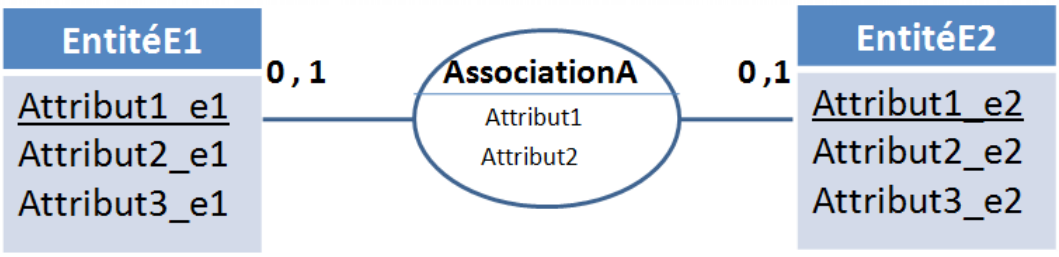
Modèle Relationnel:

- **EntiteE1**(*attribut1_e1*, attribut2_e1, attribut3_e1, **#attribut1_e2**)
- **EntiteE2**(*attribut1_e2*, attribut2_e2, attribut3_e2)

3. Règle 3 : (Association de type 1-1)

- Si les 2 cardinalités minimales sont à 0 : La clé primaire de l'entité d'un côté est ajoutée à l'autre côté où elle devient clé étrangère, tout éventuel attribut de l'association est aussi ajouté à l'autre coté.
- Si la cardinalité minimale dans un coté est 0 et la cardinalité minimale de l'autre coté est 1 : La clé primaire de l'entité côté cardinalité minimale 0 est ajoutée au côté cardinalité minimale 1 où elle devient clé étrangère.
- Si les 2 cardinalités minimales sont à 1 : les deux relations des deux entités seront fusionnées. La clé primaire de la nouvelle relation sera la clé primaire de l'une des deux relations.

Exemple : Cas : 1



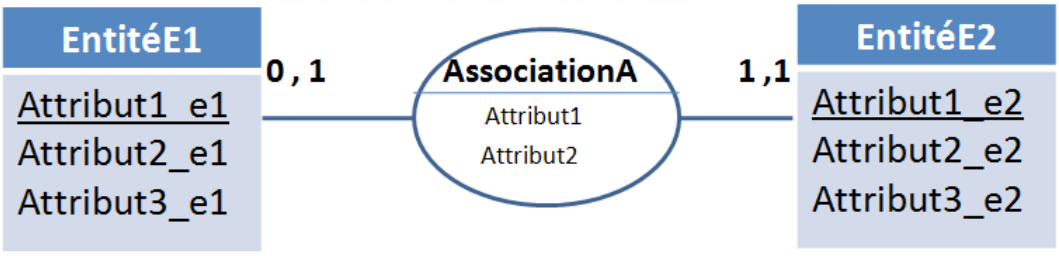
Entité-association

Modèle Relationnel:

- **EntiteE1**(*attribut1_e1*, attribut2_e1, attribut3_e1)
- **EntiteE2**(*attribut1_e2*, attribut2_e2, attribut3_e2, **#attribut1_e1**, attribut1, attribut2)

  ou :
- **EntiteE1**(*attribut1_e1*, attribut2_e1, attribut3_e1, **#attribut1_e2**, attribut1, attribut2)
- **EntiteE2**(*attribut1_e2*, attribut2_e2, attribut3_e2)

Exemple : Cas : 2

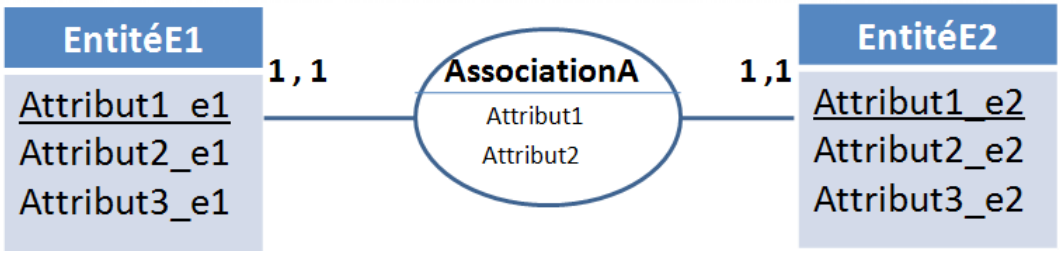


Entité-association

Modèle Relationnel:

- **EntiteE1**(*attribut1_e1*, attribut2_e1, attribut3_e1)
- **EntiteE2**(*attribut1_e2*, attribut2_e2, attribut3_e2, **#attribut1_e1**, attribut1, attribut2)

Exemple : Cas : 3



Entité-association

Modèle Relationnel:

- **EntiteE1E2**(*attribut1_e1*, attribut2_e1, attribut3_e1, attribut2_e2, attribut3_e2, attribut1, attribut2)

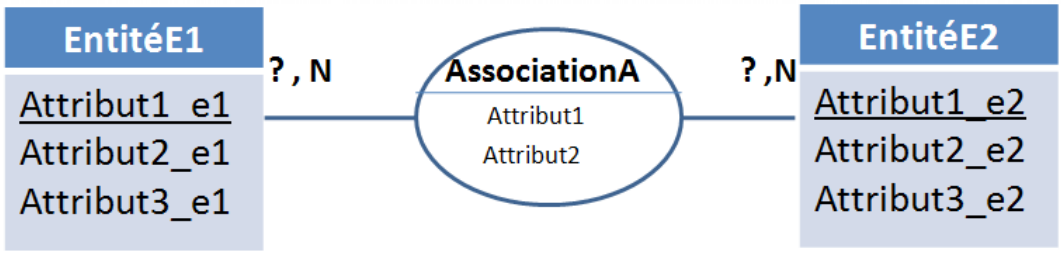
ou :

EntiteE1E2(*attribut1_e2*, attribut2_e1, attribut3_e1, attribut2_e2, attribut3_e2, attribut1, attribut2)

4. Règle 4: (Association de type N-M)

Création d'une nouvelle relation dont la clé est l'ensemble des clés primaires des deux entités concernées. Tout attribut de l'association devient attribut de la nouvelle table.

Exemple



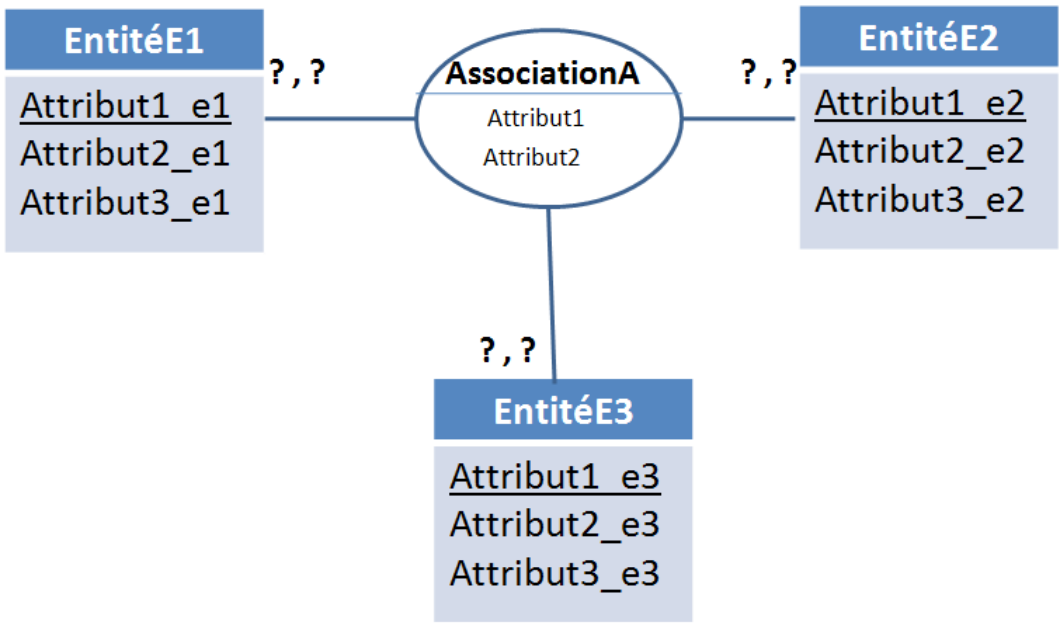
Entité-association

Modèle Relationnel:

- **EntiteE1**(*attribut1_e1*, attribut2_e1, attribut3_e1)
- **EntiteE2**(*attribut1_e2*, attribut2_e2, attribut3_e2))
- **AssociationA**(*#attribut1_e1*, *#attribut1_e2*, attribut1, attribut2)

5. Règle 5 : (Association non binaire)

Création d'une nouvelle relation dont la clé est l'ensemble des clés primaires des entités concernées. Tout attribut de l'association devient attribut de la nouvelle table.



Entité-association

Modèle Relationnel:

- **EntiteE1**(*attribut1_e1*, attribut2_e1, attribut3_e1)
- **EntiteE2**(*attribut1_e2*, attribut2_e2, attribut3_e2))
  EntiteE3(*attribut1_e3*, attribut2_e3, attribut3_e3)
- **AssociationA**(*#attribut1_e1*, *#attribut1_e2*, *#attribut1_e3*, attribut1, attribut2)

K. Exercice

Soit le modèle entité-association suivant :

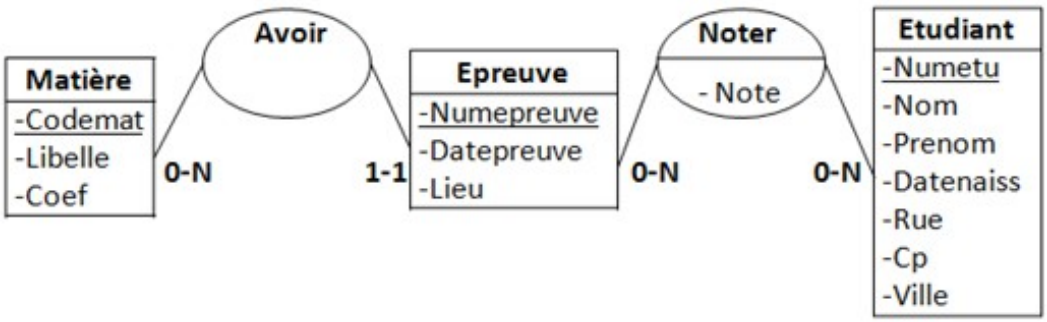


Schéma entité/association

- Donner le schéma relationnel de ce modèle.
  Solution (cf. MLD p 53)

Chapitre 03 : Algèbre relationnelle

V

A. Algèbre relationnelle

L'algèbre relationnelle a été inventée en 1970 par Edgar Frank Codd, le directeur de recherche du centre IBM de San José.Elle est constituée d'un ensemble d'opérations formelles sur les données représentées par des relations (représentées graphiquement par des tableaux ou "tables").Les opérations relationnelles permettent de créer une nouvelle relation (table) à partir d'opérations élémentaires sur d'autres relations (par exemple l'union, l'intersection, ou encore la différence). Ces opérateurs sont implémentés dans des langages de requêtes de haut niveau des SGBD tel SQL (Structured Query Langage), QUEL (Ingres) ou QBE (Paradox).L'algèbre relationnelle est une "partie" de la théorie des ensembles (mathématique).En d'autres termes - et au sens littéral - l'algèbre relationnelle est un ensemble d'opérations appliquée à un même type d'objet mathématique, chaque opération produisant à nouveau le même type d'objet en sortie. En algèbre relationnelle, l'objet est une "relation" et les opérations effectuées sur les relations en paramètre de l'opération donnent comme résultat une nouvelle relation en sortie.Par conséquent, l'algèbre relationnelle est une fermeture au sens mathématique du terme. Cependant, l'algèbre relationnelle n'est pas une fermeture transitive. En effet l'algèbre relationnelle ne permet pas de procéder à des opérations récursives, seule solution pour les problématiques de parcours de graphe.

B. Opérateurs ensemblistes

Définition : Opérateur de l'union

L'union des deux relations R1 et R2 est la relation contenant les tuples appartenant à R1 ou à R2

Syntaxe

R1∪R2

Définition : Opérateur de l'intersection

L'intersection des deux relation R1 et R2 est la relation contenant les tuples appartenant à R1 et à R2

Syntaxe

R1∩R2

Définition : Opérateur de la différence

La différence des deux relation R1 et R2 (R1-R2) est la relation contenant les tuples de R1 n'appartenant pas à R2

Syntaxe

R1-R2

Exemple

R1

CodeMat	Lib	Coef
Inf	Informatique	3
Fran	Français	1
Ang	Anglais	1
His	Histoire	2

Relation R1

R2

CodeMat	Lib	Coef
Phy	Physique	5
Ang	Anglais	1
Inf	Informatique	3

Relation R2

CodeMat	Lib	Coef
Inf	Informatique	3
Fran	Français	1
Ang	Anglais	1
His	Histoire	2
Phy	Physique	5

R1 Union R2

CodeMat	Lib	Coef
Ang	Anglais	1
Inf	Informatique	3

R1 Intersection R2

CodeMat	Lib	Coef
Fran	Français	1
His	Histoire	2

R1 - R2

C. Produit cartésien

Définition

Le produit cartésien de **R1** et de **R2** est défini par la relation Q(A1, ..., An, B1, ..., Bp) telle que (a1, ..., an, b1, ..., bp) ∈Q ssi (a1, ..., an)∈R1 et (b1, ..., bp) ∈R2Son résultat est une relation regroupant **exclusivement toutes les possibilités de combinaison** des occurrences des relations R1 et R2.

Syntaxe

R1 x R2

Exemple

CodeMat	Lib	Coef
Ang	Anglais	1
Fran	Français	1

Relation R1

CodeEp	DatEp	Lieu	CodeMat
6	06/02/15	Amphi1	Mat
7	28/01/15	Amphi2	Chi
8	29/01/15	Amphi1	Ang

Relation R2

CodeMat	Lib	Coef	NEp	DatEp	Lieu	CodeMat
Ang	Anglais	1	6	06/02/15	Amphi1	Mat
Fran	Français	1	6	06/02/15	Amphi1	Mat
Ang	Anglais	1	7	28/01/15	Amphi2	Chi
Fran	Français	1	7	28/01/15	Amphi2	Chi
Ang	Anglais	1	8	29/01/15	Amphi1	Ang
Fran	Français	1	8	29/01/15	Amphi1	Ang

R1 x R2

D. Sélection

Définition

La sélection : opérateur σ utilisé pour sélectionner un sous-ensemble de *« tuples »* d'une relation qui satisfont l'*« expression logique »*.

Syntaxe

σ **condition** (**relation**)

Exemple

CodeMat	Lib	Coef
Mat	Mathématiques	7
Phy	Physique	5
Chi	Chimie	5
Inf	Informatique	3
Ang	Anglais	1
Fran	Français	1
Méc	Mécanique	4
Bio	Biologie	3
His	Histoire	2
Des	Dessin	1

Relation : Matiere

σ Coef=1 (Matiere)

CodeMat	Lib	Coef
Ang	Anglais	1
Fran	Français	1
Des	Dessin	1

Relation résultante d'une relation qui satisfait une expression logique

E. Projection

Définition

La projection est utilisé pour sélectionner certaines colonnes d'une relation.

Syntaxe

π Attribut1, Attribut2, ...(Relation)

Exemple

CodeMat	Lib	Coef
Mat	Mathématiques	7
Phy	Physique	5
Chi	Chimie	5
Inf	Informatique	3
Ang	Anglais	1
Fran	Français	1
Méc	Mécanique	4
Bio	Biologie	3
His	Histoire	2
Des	Dessin	1

Relation : Matiere

π CodeMat, Coef (Matiere)

CodeMat	Coef
Mat	7
Phy	5
Chi	5
Inf	3
Ang	1
Fran	1
Méc	4
Bio	3
His	2
Des	1

Relation résultante d'une projection

F. Jointure

Définition

La jointure est une opération entre deux relations R1 et R2, son résultat est une nouvelle relation regroupant exclusivement toutes les possibilités de combinaison des occurrences des relations R1 et R2 qui satisfont l'expression logique (Condition).En fait, la jointure n'est qu'un **produit cartésien** suivi d'une **sélection**.

Syntaxe

Jointure

Exemple

CodeMat	Lib	Coef
Mat	Mathématiques	7
Phy	Physique	5
Chi	Chimie	5
Inf	Informatique	3
Ang	Anglais	1
Fran	Français	1
Méc	Mécanique	4
Bio	Biologie	3
His	Histoire	2
Des	Dessin	1

Relation : Matiere

CodeEp	DatEp	Lieu	CodeMat
1	26/01/15	Amphi1	Phy
2	22/01/15	Amphi2	Inf
3	23/01/15	Amphi1	Fran
4	23/01/15	Amphi1	Bio
5	23/01/15	Amphi2	Mat
6	06/02/15	Amphi1	Mat
7	28/01/15	Amphi2	Chi
8	29/01/15	Amphi1	Ang
9	30/01/15	Amphi2	Méc
10	20/01/15	Amphi1	His
11	01/02/15	Amphi1	Mat
12	19/01/15	Amphi1	Phy
13	18/01/15		Chi
14	18/01/15	Amphi1	Inf
15	01/01/15	Amphi2	Mat
16	21/01/15	Amphi1	Phy

Relation : Epreuve

Matiere.CodeMat= Epreuve.CodeEp

Exemple

CodeMat	Lib	Coef	CodeEp	DatEp	Lieu	CodeMat
Mat	Mathématiques	7	5	23/01/15	Amphi2	Mat
Mat	Mathématiques	7	6	06/02/15	Amphi1	Mat
Mat	Mathématiques	7	11	01/02/15	Amphi1	Mat
Mat	Mathématiques	7	15	01/01/15	Amphi2	Mat
Phy	Physique	5	1	26/01/15	Amphi1	Phy
Phy	Physique	5	12	19/01/15	Amphi1	Phy
Phy	Physique	5	16	21/01/15	Amphi1	Phy
Chi	Chimie	5	7	28/01/15	Amphi2	Chi
Chi	Chimie	5	13	18/01/15		Chi
Inf	Informatique	3	2	22/01/15	Amphi2	Inf
Inf	Informatique	3	14	18/01/15	Amphi1	Inf
Ang	Anglais	1	8	29/01/15	Amphi1	Ang
Fran	Français	1	3	23/01/15	Amphi1	Fran
Méc	Mécanique	4	9	30/01/15	Amphi2	Méc
Bio	Biologie	3	4	23/01/15	Amphi1	Bio
His	Histoire	2	10	20/01/15	Amphi1	His

Relation résultante d'une jointure

Complément

- **Thêta-jointure** : Dans la Thêta-jointure, l'*« expression logique »* E est une **simple comparaison** entre un *« attribut »* de la relation R1 et un *« attribut »* de la relation R2.
- **Équi-jointure** : Dans l'Équi-jointure, l'*« expression logique »* E est un **test d'égalité** entre un *« attribut »* de la relation R1 et un *« attribut »* de la relation R2.
- **Jointure naturelle** : Dans la jointure naturelle, l'*« expression logique »* E est un **test d'égalité entre** les *« attributs »* qui portent le **même nom** dans les relations R1 et R2 (ces *« attributs »* ne seront pas dupliqués dans la relation résultante).

G. Division

Définition

L'opération la division de la relation R1 par R2 génère une troisième relation regroupant les occurrences de la relation R1 qui ont une **correspondance** à **toutes** les occurrences de la relation R2 dans R1.

(le schéma de R2 doit être strictement inclus dans celui de R1).

Syntaxe

R1÷ R2

Exemple

X	Y
a1	b1
a2	b2
a3	b1
a1	b2
a2	b1

Relation R1

Z
b1
b2

Relation R2

R1÷ R2

X
a1
a2

Relation résultante de la division

Chapitre 04 : Interrogation des bases de données (SQL)

VI

SQL (sigle de Structured Query Language, en français langage de requête structurée) est un langage informatique normalisé servant à exploiter des bases de données relationnelles. La partie langage de manipulation des données de SQL permet de rechercher, d'ajouter, de modifier ou de supprimer des données dans les bases de données relationnelles.Outre le langage de manipulation des données, la partie langage de définition des données permet de créer et de modifier l'organisation des données dans la base de données, la partie langage de contrôle de transaction permet de commencer et de terminer des transactions, et la partie langage de contrôle des données permet d'autoriser ou d'interdire l'accès à certaines données à certaines personnes.Créé en 1974, normalisé depuis 1986, le langage est reconnu par la grande majorité des systèmes de gestion de bases de données relationnelles (abrégé SGBDR) du marché.SQL fait partie de la même famille que les langages SEQUEL (dont il est le descendant), QUEL ou QBE (Zloof).

A. Commande SELECT

Définition

L'utilisation la plus courante de SQL consiste à lire des données issues de la base de données. Cela s'effectue grâce à la commande **SELECT**, qui retourne des enregistrements dans un tableau de résultat. Cette commande peut sélectionner

une ou plusieurs colonnes d'une table.

Syntaxe

SELECT « *liste_d_attributs_recherchés* » ou « * » **FROM** « *liste_des_relations* »[**WHERE** « *condition* »] ;

- Clause **SELECT** pour spécifier une projection
- Clause **FROM** pour identifier les relations
- Clause **WHERE** pour spécifier une sélection

B. Requêtes sur une table

1. Projection

Il est possible de lire plusieurs colonnes à la fois depuis une table. Il suffit de séparer les noms des attributs souhaités par une virgule.

Exemple

Afficher le nom des matières et leurs coefficient.

*Pour accéder à la table **Matiere** ou **Epreuve** cliquez ici* (cf. Les tables : Matiere et Epreuve)

```
SELECT Lib, coef
FROM Matiere ;
```

Lib	coef
Anglais	1
Biologie	3
Chimie	5
Dessin	1
Français	1
Histoire	2
Informatique	3
Mathématiques	7
Mécanique	4
Physique	5

Table résultante d'une projection SQL

2. Projection

Il est possible de retourner toutes les colonnes d'une table sans citer le nom de tous les attributs. On doit simplement utiliser le caractère *

Exemple

Afficher toutes les informations sur les Matières.

*Pour accéder à la table **Matiere** ou **Epreuve** cliquez ici* (cf. Les tables : Matiere et Epreuve)

```
SELECT *
FROM Matiere ;
```

CodeMat	Lib	Coef
Ang	Anglais	1
Bio	Biologie	3
Chi	Chimie	5
Des	Dessin	1
Fran	Français	1
His	Histoire	2
Inf	Informatique	3
Mat	Mathématiques	7
Méc	Mécanique	4
Phy	Physique	5

*Relation résultante de l'opérateur **

3. Sélection sans doublons (Mot clé DISTINCT)

Définition

L'utilisation de la commande « *SELECT* » en SQL permet de lire toutes les données d'une ou plusieurs colonnes. Cette commande peut potentiellement afficher des lignes en doubles. Pour éviter des redondances dans les résultats il faut simplement ajouter **DISTINCT** après le mot « *SELECT* ».

Exemple

Afficher toutes les coefficients des matières sans répétition.

*Pour accéder à la table **Matiere** ou **Epreuve** cliquez ici* (cf. Les tables : Matiere et Epreuve)

```
SELECT DISTINCT coef
FROM Matiere;
```

coef
1
2
3
4
5
7

Relation résultante de l'opérateur DISTINCT

4. Condition de sélection

Une expression logique sur les attributs qui utilise les opérateurs de comparaison **= != < > <= >=** et des opérateurs spécifiques qui peut être complexe et utiliser les connecteurs **AND**, **OR** et **NOT**.

5. Opérateurs spécifiques

a) Opérateur BETWEEN

Définition

L'opérateur **BETWEEN** est utilisé dans une requête SQL pour sélectionner un intervalle de données dans une requête utilisant « *WHERE* ». L'intervalle peut être constitué de chaînes de caractères, de nombres ou de dates. L'exemple le plus concret consiste par exemple à récupérer uniquement les enregistrements entre 2 dates définies.

Syntaxe

```
SELECT *FROM tableWHERE attribut BETWEEN 'valeur1' AND
'valeur2'
```

Exemple

Quels sont les matières qui ont le coefficient 7 ou un coefficient entre 2 et 4?

*Pour accéder à la table **Matiere** ou **Epreuve** cliquez ici* (cf. Les tables : Matiere et Epreuve)

```
SELECT *
FROM Matiere
WHERE coef =7 OR coef BETWEEN 2 and 4;
```

CodeMat	Lib	Coef
Mat	Mathématiques	7
Inf	Informatique	3
Méc	Mécanique	4
Bio	Biologie	3
His	Histoire	2

Relation résultante d'une requête utilisant une condition contenant l'opérateur BETWEEN

b) Opérateur IN

Définition

L'opérateur logique **IN** dans SQL s'utilise avec la commande « *WHERE* » pour vérifier si une colonne est égale à une des valeurs comprise dans set de valeurs déterminés. C'est une méthode simple pour vérifier si une colonne est égale à une valeur OU une autre valeur OU une autre valeur et ainsi de suite, sans avoir à utiliser de multiple fois l'opérateur « *OR* ».

Syntaxe

```
SELECT attribut1,...FROM tableWHERE attribut IN ( valeur1,
valeur2, valeur3, ... )
```

Exemple

Quels sont les matieres dont le coefficient est soit 3, soit 4, soit 6, soit 8?

*Pour accéder à la table **Matiere** ou **Epreuve** cliquez ici* (cf. Les tables : Matiere et Epreuve)

```
SELECT *
FROM Matiere
WHERE Coef IN (3, 4, 6, 8);
```

CodeMat	Lib	Coef
Inf	Informatique	3
Méc	Mécanique	4
Bio	Biologie	3

Relation résultante d'une requête utilisant une condition contenant l'opérateur IN

c) Opérateur LIKE

Définition

L'opérateur **LIKE** est utilisé dans la clause « *WHERE* » des requêtes SQL. Ce mot-clé permet d'effectuer une recherche sur un modèle particulier. Il est par exemple possible de rechercher les enregistrements dont la valeur d'une colonne commence par telle ou telle lettre. Les modèles de recherches sont multiple.

Syntaxe

```
SELECT attribut1,...FROM tableWHERE attribut LIKE 'modele'
```

Exemple

Quels sont les matieres dont le nom commence par M et un c pour 3ième lettre?

*Pour accéder à la table **Matiere** ou **Epreuve** cliquez ici* (cf. Les tables : Matiere et Epreuve)

```
SELECT *
FROM Matiere
WHERE Lib LIKE 'M_c%';
```

CodeMat	Lib	Coef
Méc	Mécanique	4

Relation résultante d'une requête utilisant une condition contenant l'opérateur LIKE

d) Opérateur IS NULL

Définition

Dans le langage SQL, l'opérateur **IS** permet de filtrer les résultats qui contiennent la valeur **NULL**. Cet opérateur est indispensable car la valeur NULL est une valeur inconnue et ne peut par conséquent pas être filtrée par les opérateurs de comparaison (cf. égal, inférieur, supérieur ou différent).

Syntaxe

```
SELECT attribut1,...
FROM table
WHERE attribut IS NULL
```

Exemple

Quels sont les epreuves dont le lieu est inconnu?

*Pour accéder à la table **Matiere** ou **Epreuve** cliquez ici* (cf. Les tables : Matiere et Epreuve)

```
SELECT *
FROM Epreuve
WHERE Lieu IS NULL;
```

CodeEp	DatEp	Lieu	CodeMat
13	18/01/15		Chi

Relation résultante d'une requête utilisant une condition contenant l'opérateur IS NULL

6. La commande ORDER BY (Tri)

Définition

La commande **ORDER BY** permet de trier les lignes dans un résultat d'une requête SQL. Il est possible de trier les données sur une ou plusieurs colonnes, par ordre ascendant ou descendant.

Syntaxe

```
SELECT attribut1, fonction(attribut2)FROM tableGROUP BY
attribut1
```

Exemple

Donner le nom et le coefficient des matières dont le coefficient est supérieur à 3 selon l'ordre croissant des coefficients.

*Pour accéder à la table **Matiere** ou **Epreuve** cliquez ici* (cf. Les tables : Matiere et Epreuve)

```
SELECT Lib, coef
FROM Matiere
WHERE coef>3
ORDER BY Coef ASC;
```

Lib	coef
Mécanique	4
Chimie	5
Physique	5
Mathématiques	7

Relation résultante d'une requête utilisant la commande ORDER BY

C. Requêtes sur plusieurs tables

1. Jointure

Définition

Les jointures en SQL permettent d'associer plusieurs tables dans une même requête. Cela permet d'exploiter la puissance des bases de données relationnelles pour obtenir des résultats qui combinent les données de plusieurs tables de manière efficace.

Exemple

Donner les codes des épreuves ainsi que le nom de leurs matières?

*Pour accéder à la table **Matiere** ou **Epreuve** cliquez ici* (cf. Les tables : Matiere et Epreuve)

```
SELECT CodeEp, Lib
FROM Matiere ,Epreuve
WHERE Matiere.CodeMat=Epreuve.CodeMat;
```

CodeEp	Lib
5	Mathématiques
6	Mathématiques
11	Mathématiques
15	Mathématiques
1	Physique
12	Physique
16	Physique
7	Chimie
13	Chimie
2	Informatique
14	Informatique
8	Anglais
3	Français
9	Mécanique
4	Biologie
10	Histoire

Relation résultante d'une jointure

2. Noms absolus – Renommage

Les alias SQL sont utilisés pour renommer temporairement une table ou un entête de colonne.

Exemple

Exemple 1

Donner le nom des matières qui ont un coefficient supérieur à celui de la matière Mécanique.

*Pour accéder à la table **Matiere** ou **Epreuve** cliquez ici* (cf. Les tables : Matiere et Epreuve)

```
SELECT M1.Lib
FROM Matiere M1, Matiere M2
WHERE M2.Lib='Mécanique' and M1.Coef>M2.Coef;
```

Lib
Mathématiques
Physique
Chimie

Relation résultante d'une requête utilisant le renommage de tables

Exemple 2

Donner le nom des matières qui ont le même coefficient que la matière Anglais.

*Pour accéder à la table **Matiere** ou **Epreuve** cliquez ici* (cf. Les tables : Matiere et Epreuve)

```
SELECT M1.Lib
FROM Matiere M1, Matiere M2
WHERE M2.Lib='Anglais' and M1.Coef=M2.Coef;
```

Lib
Dessin
Français
Anglais

Relation résultante d'une requête utilisant le renommage de tables

D. Opérateurs ensemblistes

1. Opérateur UNION

Définition

La commande **UNION** de SQL permet de mettre bout-à-bout les résultats de plusieurs requêtes utilisant elles-même la commande « *SELECT* ». C'est donc une commande qui permet de concaténer les résultats de 2 requêtes ou plus. Pour l'utiliser il est nécessaire que chacune des requêtes à concaténer retournes le même nombre de colonnes, avec les mêmes types de données et dans le même ordre.

Complément

Par défaut, les enregistrements exactement identiques ne seront pas répétés dans les résultats. Pour effectuer une union dans laquelle même les lignes dupliquées sont affichées il faut plutôt utiliser la commande **UNION ALL**.

Syntaxe

```
SELECT attributx,... FROM table1UNIONSELECT attributy,...
FROM table2
```

Exemple

Donner la liste des codes des matières qui ont eu des épreuves avant le 19/01/2015 ou qui ont un coefficient supérieur à 4 ?

```
SELECT CodeMat
FROM Epreuve
WHERE DatEp &lt; '19/01/2015'
UNION
SELECT CodeMat
FROM Matiere
WHERE Coef &gt;4 ;
```

CodeMat
Chi
Inf
Mat
Phy

Relation résultante de l'opérateur UNION

2. Opérateur INTERSECT

Définition

La commande SQL **INTERSECT** permet d'obtenir l'intersection des résultats de 2 requêtes. Cette commande permet donc de récupérer les enregistrements communs à 2 requêtes. Cela peut s'avérer utile lorsqu'il faut trouver s'il y a des données similaires sur 2 tables distinctes.

Complément

Pour l'utiliser convenablement il faut que les 2 requêtes retourne le même nombre de colonnes, avec les mêmes types et dans le même ordre.

Syntaxe

```
SELECT attributx,... FROM table1
UNION
SELECT attributy,... FROM table2
```

Exemple

Donner la liste des codes des matières qui ont eu des épreuves avant le 19/01/2015 et qui ont un coefficient supérieur à 4 ?

```
SELECT CodeMat
FROM Epreuve
WHERE DatEp <'19/01/2015'
INTERSECT
SELECT CodeMat
FROM Matiere
WHERE Coef >4;
```

CodeMat
Chi
Mat

Relation résultante de l'opérateur INTERSECT

3. Opérateur EXCEPT

Définition

Dans le langage SQL la commande **EXCEPT** s'utilise entre 2 instructions pour récupérer les enregistrements de la première instruction sans inclure les résultats de la seconde requête. Si un même enregistrement devait être présent dans les résultats des 2 syntaxes, ils ne seront pas présent dans le résultat final.

Complément

Cette commande peut être appellée différemment selon les SGBD (l'autre alternative est **MINUS**)

Syntaxe

```
SELECT attributx,... FROM table1
```

```
EXCEPT
SELECT attributy,... FROM table2
```

Exemple

Donner la liste des codes des matières qui ont eu des épreuves avant le 19/01/2015 et qui n'ont pas un coefficient supérieur à 4 ?

```
SELECT CodeMat
FROM Epreuve
WHERE DatEp <'19/01/2015'EXCEPTSELECT CodeMat
FROM Matiere
WHERE Coef >4;
```

CodeMat
Inf

Relation résultante de l'opérateur EXCEPT

E. Requêtes de regroupement et d'agrégation

1. Requêtes de regroupement et d'agrégation

Le regroupement regroupe les tuples résultats selon des valeurs de certains attributs

Une fonction d'agrégation permet de calculer une seule valeur numérique à partir de plusieurs tuples qui correspondent à un attribut donné. Une fonction d'agrégation peut être utilisée dans la clause « *SELECT* » pour effectuer un calcul et peut etre utilisée aussi dans la clause « *WHERE* » pour participer dans une expression logique.

Syntaxe

```
SELECT attribut1,..., fonctions_statistiques FROM table1,
tables2,...
WHERE condition
GROUP BY attributx,... [HAVING condition ] ;
```

Syntaxe : Fonctions statistiques

- **SUM**(« *nom_d_attribut* »)
- **COUNT**(*)
- **COUNT**(**DISTINCT** « *nom_d_attribut* »)
- **MAX**(« *nom_d_attribut* »)
- **MIN**(« *nom_d_attribut* »)
- **AVG**(« *nom_d_attribu* »« *t* »)
- **AVG**(**DISTINCT** « *nom_d_attribut* »)

2. Commande GROUP BY

Définition

La commande **GROUP BY** est utilisée en SQL pour grouper plusieurs résultats et utiliser une fonction de totaux sur un groupe de résultat. Sur une table qui contient toutes les ventes d'un magasin, il est par exemple possible de liste regrouper les ventes par clients identiques et d'obtenir le coût total des achats pour chaque client.

Exemple

Quel est le nombre de matières pour chaque coefficient ?

```
SELECT coef, COUNT(CodeMat)
FROM Matiere
GROUP BY coef;
```

regroupement des matières selon leur coefficient. Pour chaque groupe, calcul du nombre d'éléments et affichage du coefficient et du nombre de matières

Coef	
1	3
2	1
3	2
4	1
5	2
7	1

Relation résultante d'une requête utilisant la commande GROUP BY

Exemple : En utilisant la clause HAVING

Donner le nombre de matières pour chaque coefficient (toute fois on prend pas les coefficients qui contribuent à une seule matière)?

```
SELECT coef, COUNT(CodeMat)
FROM Matiere
GROUP BY coef
HAVING COUNT(CodeMat)>1 ;
```

pour ajouter une condition sur les groupes à considérer.

coef	
1	3
3	2
5	2

Relation résultante d'une requête utilisant la condition HAVING de GROUP BY

F. Requêtes imbriquées

1. Sous-requêtes indépendantes

a) Utilisant l'opérateur IN

Exemple

Quels sont les codes des épreuves de la matière 'Mathématiques?

*Pour accéder à la table **Matiere** ou **Epreuve** cliquez ici* (cf. Les tables : Matiere et Epreuve)

```
SELECT CodeEp
FROM Epreuve
WHERE CodeMat IN (SELECT CodeMat
FROM Matiere
WHERE Lib='Mathématiques') ;
```

CodeEp
5
6
11
15

Relation résultante d'une requête utilisant la commande IN

b) Utilisant l'opérateur NOT IN

Exemple

Quels sont les codes des épreuves qui ne sont pas de la matière 'Mécanique'?

*Pour accéder à la table **Matiere** ou **Epreuve** cliquez ici* (cf. Les tables : Matiere et Epreuve)

```
SELECT CodeEp
FROM Epreuve
WHERE CodeMat NOT IN (SELECT CodeMat
FROM Matiere
WHERE Lib='Mécanique') ;
```

CodeEp
1
2
3
4
5
6
7
8
10
11
12
13
14
15
16

Relation résultante d'une requête utilisant la commande NOT IN

c) Utilisant l'opérateur ALL

Définition : Opérateur ALL

Dans le langage SQL, la commande **ALL** permet de comparer une valeur dans l'ensemble de valeurs d'une sous-requête. En d'autres mots, cette commande permet de s'assurer qu'une condition est « **égale** », « **différente** », « **supérieure** », « **inférieure** », « **supérieure ou égale** » ou « **inférieure ou égale** » pour tous les résultats retourné par une sous-requête.

Exemple

Quels sont les noms matières qui ont le coefficient le plus élevé ?

*Pour accéder à la table **Matiere** ou **Epreuve** cliquez ici* (cf. Les tables : Matiere et Epreuve)

```
SELECT Lib
FROM Matiere
WHERE Coef>=ALL(SELECT Coef FROM Matiere);
```

Lib
Mathématiques

Relation résultante d'une requête utilisant l'opérateur ALL

d) Utilisant l'opérateur ANY

Définition : Opérateur ANY

Dans le langage SQL, la commande **ANY** permet de comparer une valeur avec le résultat d'une sous-requête. Il est ainsi possible de vérifier si une valeur est « **égale** », « **différente** », « **supérieur** », « **supérieur ou égale** », « **inférieur** » ou « **inférieur ou égale** » pour au moins une des valeurs de la sous-requête.

Complément

Le mot-clé **SOME** est un alias de **ANY**, l'un et l'autre des termes peut être utilisé.

Exemple

Quels sont les noms matières qui n'ont pas le coefficient le plus bas ?

*Pour accéder à la table **Matiere** ou **Epreuve** cliquez ici* (cf. Les tables : Matiere et Epreuve)

```
SELECT Lib
FROM Matiere
WHERE Coef>ANY(SELECT Coef FROM Matiere);
```

Lib
Mathématiques
Physique
Chimie
Informatique
Mécanique
Biologie
Histoire

Relation résultante d'une requête utilisant l'opérateur ANY

2. Sous-requêtes corrélées

a) Utilisant l'opérateur IN

Exemple

Quels sont les noms des matières qui ont un coefficient en commun avec une autre matière?

*Pour accéder à la table **Matiere** ou **Epreuve** cliquez ici* (cf. Les tables : Matiere et Epreuve)

```
SELECT Lib
FROM Matiere M1
WHERE Coef IN (SELECT Coef
FROM Matiere M2
WHERE M1.CodeMat <> M2.CodeMat) ;
```

Lib
Physique
Chimie
Informatique
Anglais
Français
Biologie
Dessin

Relation résultante d'une requête utilisant l'opérateur IN (corrélation)

b) Utilisant l'opérateur EXISTS

Définition : Opérateur EXISTS

Dans le langage SQL, la commande **EXISTS** s'utilise dans une clause conditionnelle pour savoir s'il y a une présence ou non de lignes lors de l'utilisation d'une sous-requête.

Exemple

Quels sont les codes des épreuves de coefficient 1?

*Pour accéder à la table **Matiere** ou **Epreuve** cliquez ici* (cf. Les tables : Matiere et Epreuve)

```
SELECT CodeEp
FROM Epreuve E
WHERE EXISTS (SELECT *
FROM Matiere M
WHERE M.Coef=1 and E.CodeMat = M.CodeMat) ;
```

CodeEp
3
8

Relation résultante d'une requête utilisant l'opérateur EXISTS

c) Utilisant l'opérateur NOT EXISTS

Exemple

Quels sont les codes des épreuves dont le coefficient n'est pas 1?

*Pour accéder à la table **Matiere** ou **Epreuve** cliquez ici* (cf. Les tables : Matiere et Epreuve)

```
SELECT CodeEp FROM Epreuve E WHERE NOT EXISTS (SELECT *
FROM Matiere M WHERE M.Coef =1 and E.CodeMat = M.CodeMat) ;
```

CodeEp
1
2
4
5
6
7
9
10
11
12
13
14
15
16

Relation résultante d'une requête utilisant l'opérateur NOT EXISTS

G. Exercice

Exercice

Soit une base de données relationnelle décrite par les relations suivantes : **Fournisseur** (« ***nf*** », nomf, catf, vilf) **Pièce** (« ***np*** », nomp, clrp, pdsp) **Livraison** (#« ***nf*** », #« ***np*** », qte) Cette base de données contient des informations concernant des fournisseurs, des pièces et l'ensemble des livraisons effectuées. Le dictionnaire de données contient les informations suivantes :

Attribut	Signification
NF	numéro du fournisseur
NOMF	nom du fournisseur
CATF	catégorie du fournisseur
VILF	ville du fournisseur
NP	numéro de la pièce
NOMP	nom de la pièce
CLRP	couleur de la pièce
PDSP	poids de la pièce
QTE	quantité livrée

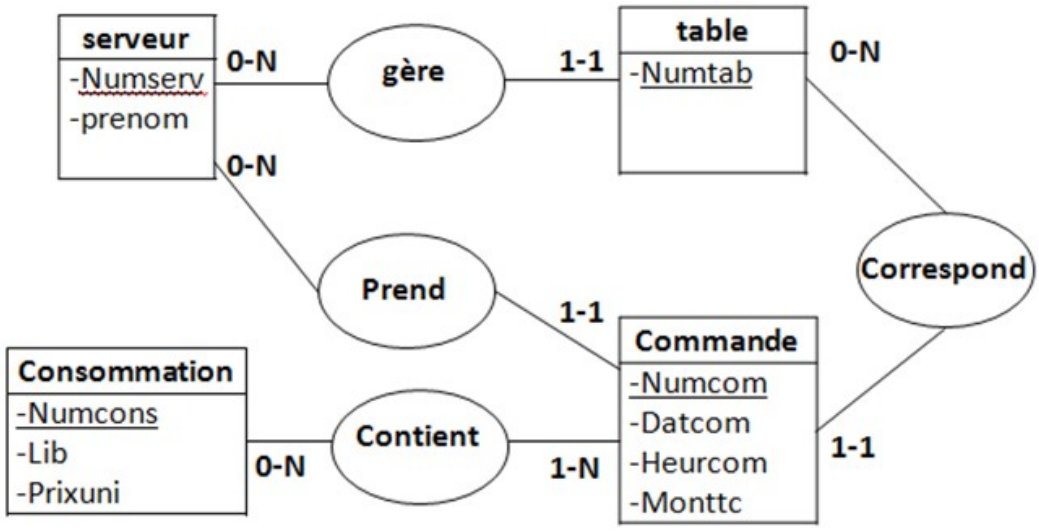
Dictionnaire de données

Exprimez les requêtes suivantes en SQL :

1. Donnez les noms de tous les fournisseurs.
2. Donnez les numéros et les catégories de tous les fournisseurs.
3. Donnez l'information complète sur les fournisseurs dont la catégorie est 10.
4. Donnez les numéros des fournisseurs qui habitent Constantine et dont la catégorie est supérieure à 20.
5. Donnez les numéros et les noms de pièces qui sont de couleur rouge et dont le poids est supérieur à 100.
6. Donnez les numéros des fournisseurs qui livrent la pièce nommée "P4" en quantité inférieure à 50.
7. Donnez les numéros de pièces livrées en quantité supérieure à 100.
8. Donnez les numéros des fournisseurs qui ne livrent pas de pièces.
9. Donnez les numéros des fournisseurs qui livrent à la fois des pièces numéro 6 et 8.
10. Donnez les numéros des fournisseurs qui livrent à la fois les pièces nommées "P1" et "P2".
11. Donnez les numéros des fournisseurs qui livrent les pièces nommées "P1" ou "P2".
12. Donnez les numéros des fournisseurs qui livrent toutes les pièces

Ressources annexes

- MCD



- MLD

Schéma relationnel :

Matière (Codemat, libelle, coef)

Epreuve (Numepreuve, Datepreuve, lieu, # Codemat)

Etudiant (Numetu, nom, prenom, Datenaiss, Rue, Cp, Ville)

Notation (# Numetu, # Numepreuve, Note)