Université Abou Bakr Belkaid– Tlemcen
Faculté des Sciences
Département d'Informatique

BASES DE DONNÉES

MATALLAH Houcine

L2 Informatique

2018-2019

INFORMATIONS PRATIQUES

- **Volume horaire**
  - Cours : (1 x 1,5 h *Dimanche 11H30-13H*)
  - TD : (1 x 1,5 h)
  - TP : (1 x 3 h)
- **Evaluation**
  - Contrôle
  - Tests TP
  - Examen final
  - Note Finale = (2 x Examen + Contrôle + Note TP) / 4

OBJECTIFS DE LA MATIERE

1. Concepts de base des BD
2. Modèle relationnel
3. Algèbre relationnelle
4. Langage SQL
5. Dépendances Fonctionnelles et Normalisation

PLAN DE LA MATIERE

- Concepts de base
- Modèle relationnel
- Algèbre relationnelle
- SQL : Définitions des schémas
- SQL : Mise à jour de données
- SQL : Interrogation simple
- SQL : Fonctions de groupe
- SQL : Extraction de données de plusieurs tables (Jointures)
- SQL : Opérateurs ensemblistes
- SQL : Sous-requêtes
- Dépendances fonctionnelles et Normalisation des relations

PLAN DE LA MATIERE

- **Concepts de base**
- Modèle relationnel
- Algèbre relationnelle
- SQL : Définitions des schémas
- SQL : Mise à jour de données
- SQL : Interrogation simple
- SQL : Fonctions de groupe
- SQL : Extraction de données de plusieurs tables (Jointures)
- SQL : Opérateurs ensemblistes
- SQL : Sous-requêtes
- Dépendances fonctionnelles et Normalisation des relations

RAPPEL DE COURS SI

Rappel : Conception SI **MCD → MLD**

Client (Num_Cl, Nom, Prenom, Adresse)

Commande (Num_Cde, Date_Cde, Num_Cl#)

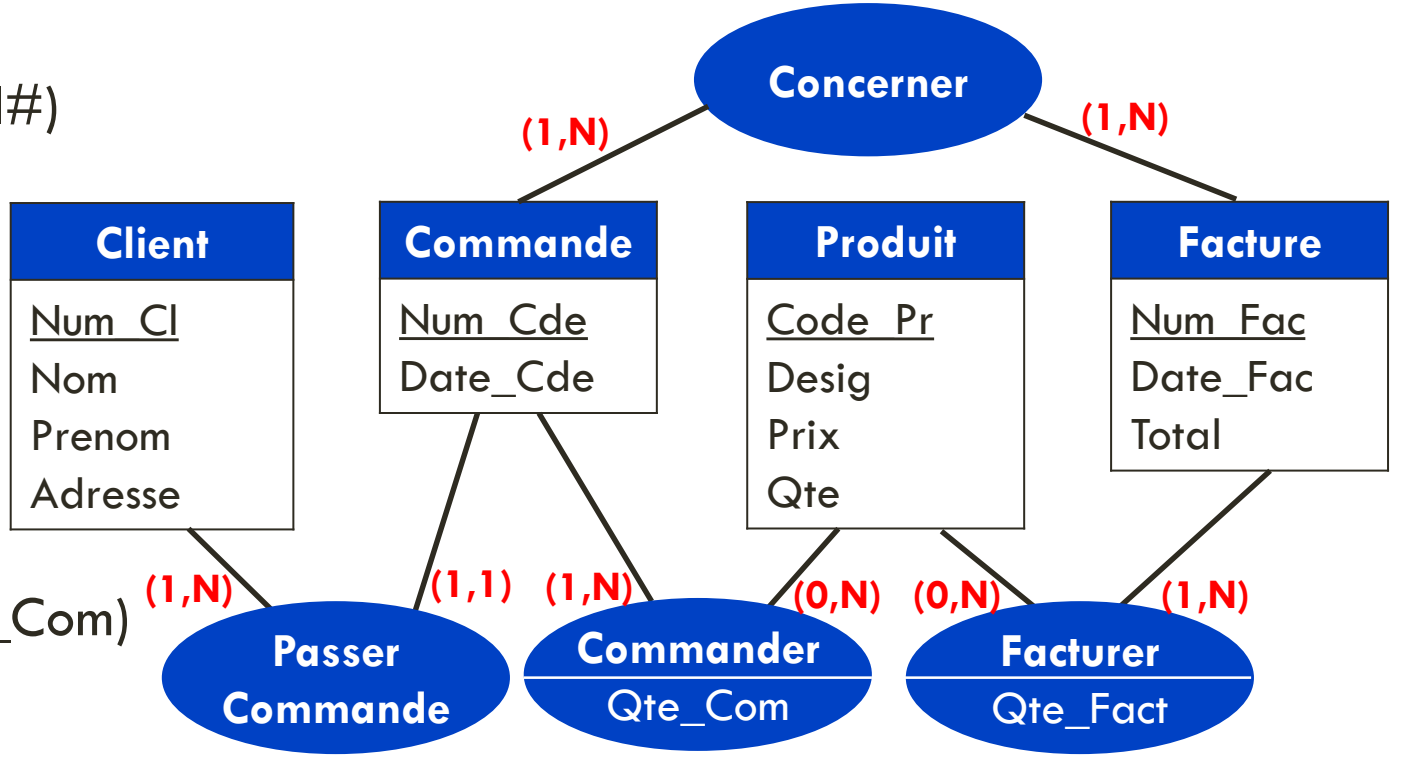
Produit (Code_Pr, Desig, Prix, Qte)

Facture (Num_Fac, Date_Fac, Total)

Facturer (Code_Pr#, Num_Fac#, Qte_Fact)

Concerner (Num_Cde#, Num_Fac#)

Commander (Num_Cde#, Code_Pr#, Qte_Com)



CONCEPTS DE BASE

Evolution des systèmes de gestion de données

- Fichiers plats (60)
- Bases de données (70)
- Entrepôts de données (90)
- Mouvance NoSQL et NewSQL (2009 et 2011)

CONCEPTS DE BASE

Fichiers plats et Systèmes de Gestion de fichiers (SGF)

- Un **fichier** est un **ensemble de données homogènes** gérées par un **SGF**
- Le **fichier plat** est la **1ére forme** des fichiers électroniques stockés sur mémoire secondaire
- Le fichier plat est une table sous la forme d'un simple fichier (.txt ou .ini ou .csv)
- Chaque ligne du tableau correspond à une ligne dans le fichier
- Dans une même ligne, on sépare les données appartenant à différentes colonnes à l'aide d'un caractère particulier :

Asma, BDD, 12
Youcef, GL, 14
Yassine, Algo, 10

CONCEPTS DE BASE

Caractéristiques des fichiers plats

- Dépendance Données - Programmes
- Modèle des données intégré dans les programmes
- N'est pas conçu pour gérer une masse importante de données et de liens
- **Exemples** : Fichier Produit, Fichier client, Fichier Abonné, Fichier employé, Fichier salaire

CONCEPTS DE BASE

Limites d'utilisation des fichiers

- **Programmation plus détaillée**
  - Programmes sensibles aux modifications physiques (organisation) ou logique (structure)
  - Toute modification de la structure des enregistrements (ajout d'un champ par exemple) entraîne la réécriture de tous les programmes qui manipulent ces fichiers
- **Connaissance technique approfondie :** L'utilisation de fichiers impose à l'utilisateur de connaître
  - Mode d'accès (séquentielle, indexée, …)
  - Structure physique des enregistrements
  - Localisation des fichiers qu'il utilise afin de pouvoir accéder aux informations dont il a besoin
- **Lourdeur d'accès aux données**
  - Un accès aux données = un programme

CONCEPTS DE BASE

Limites d'utilisation des fichiers

- Pour des **nouvelles applications,** l'utilisateur devra obligatoirement écrire de **nouveaux programmes** et il pourra être amené à créer de **nouveaux fichiers** qui contiendront peut-être des **informations déjà présentes dans d'autres fichiers**
- **Particularisation des fichiers en fonction des traitements** (Duplication de la même donnée sur plusieurs fichiers : Grande redondance)
- **Particularisation de la saisie et traitements en fonction des fichiers** : Un ou plusieurs programmes par fichier
- Chaque organisme a **ses propres applications travaillant sur ces propres fichiers**

CONCEPTS DE BASE

Limites d'utilisation des fichiers

- **Redondance des données et incohérences**
  - Effort pour le maintien de la cohérence (MAJ de la même donnée sur les différents fichiers)
  - Coût de maintien de la cohérence (Délais de MAJ supérieurs)
  - Multiplier les erreurs de MAJ
  - Sinon : Travailler sur des données contradictoires
- **Sécurité et protection des données**
- **Pas de contrôle de concurrence**

CONCEPTS DE BASE

Bases de données et systèmes de gestion de bases de données

- **Objectif** : Pallier les insuffisances des SGF

- **Solution** : Gestion centralisée des données
  - Chaque donnée n'est enregistrée qu'en un seul endroit de la base
  - Diminuer les risques d'erreurs et de MAJ
  - Eviter les informations contradictoires sur une même donnée dupliquée sur des fichiers différents

CONCEPTS DE BASE

Définitions : Donnée

- **Information** sur un objet, une personne, un événement sous sa forme brute que nous voulons conserver pour pouvoir la traiter
- **Renseignement** mise à la disposition de l'utilisateur (Fatima, 20M2, Peugeot, 3Kg, Algérie, 15000 Km, Bleu,17.5,..)
- **Relation** entre les informations : « *Ahmed enseigne les Bases de données* »

*(**Base** : Fondation, Fondement, Assiette, Socle, Semelle,..)*

CONCEPTS DE BASE

Définitions : Base de Données

- Collection de données **Cohérentes** et **Structurées**
  - Homogènes, Similaires
  - Organisées, Arrangées
- L'ensemble **cohérent, intégré, partagé** des informations nécessaires au fonctionnement d'une entreprise, utilisées par des programmes ou des utilisateurs
- C'est une entité dans laquelle il est possible de **stocker** les données de façon structurée et avec le **moins de redondances possibles**
  - Persistance
  - Minimiser les répétitions

CONCEPTS DE BASE

Définitions : Base de Données (BD)

- Ensemble de données modélisant les objets d'une partie du monde réel et servant de support à une application informatique

- **Exemples**
  - BD Gestion Personnel (Employé, Congé, Absence, Paie, ..)
  - BD Gestion Scolarité (Etudiant, Matière, Cours, Salle,..)
  - BD Gestion Commerciale (Produit, Client, Fournisseur, Commande, Facture,..)

CONCEPTS DE BASE

Caractéristiques d'une BD (Pq une BD ?)

- **Exhaustivité** : Elle doit regrouper toutes les données (BD Complète, Intégrale)
- **Persistance** : Conserver les données
- **Non redondance** : Les données ne doivent pas se répéter à l'intérieur de la base
- **Cohérence** : On veut pas entrer une facture pour le client 35 s'il n'y a pas de client 35
- **Partageabilité** : La BD doit assurer des accès simultanés

CONCEPTS DE BASE

Cycle de vie d'une BD

- Phase 1 : **Conception**
- Phase 2 : **Implantation ou Déploiement**
- Phase 3 : **Utilisation et Administration**

CONCEPTS DE BASE

Abstraction des données

Trois niveaux d'abstraction pour trois profils d'utilisateurs :

- **Niveau conceptuel/logique** (profil concepteur)
  - Quelle est la structure des données stockées
- **Niveau physique/interne** (profil administrateur)
  - Comment sont organisées des données sur le support physique
  - Comment elles sont stockées
  - Comment accéder rapidement aux données
- **Niveau vue/externe** (profil utilisateur)
  - Quelles sont les données manipulées

CONCEPTS DE BASE

Actions sur une BD

- **Définition de la structure de données** (Contenant)
- **Interrogation des données** (Opérations de lecture du contenu)
- **Mise à jour des données** (Opérations d'écriture du contenu)
  - Insertion
  - Modification
  - Suppression

CONCEPTS DE BASE

Définitions : Système de Gestion de Base de Données (SGBD – DBMS)

- Comment conserver ces données, les mette à la disposition de l'utilisateur, partager ces données, veiller à la cohésion et l'intégrité de la BD, assurer la sécurité des données ?
- Afin de pouvoir contrôler les données et les utilisateurs

- Besoin d'outils logiciels permettant d'assurer toutes ces fonctions

SGBD

CONCEPTS DE BASE

Définitions : Système de Gestion de Base de Données (SGBD – DBMS)

- **Déf 1 :** Ensemble de services ou logiciels informatiques qui permet de définir, modifier, interroger, partager et administrer une BD
- **Déf 2 :** Logiciel de haut niveau qui permet de manipuler les informations stockées dans une base de données
- Logiciel gérant une BD ou plusieurs BDs et peut aussi accéder aux BDs d'autres SGBD

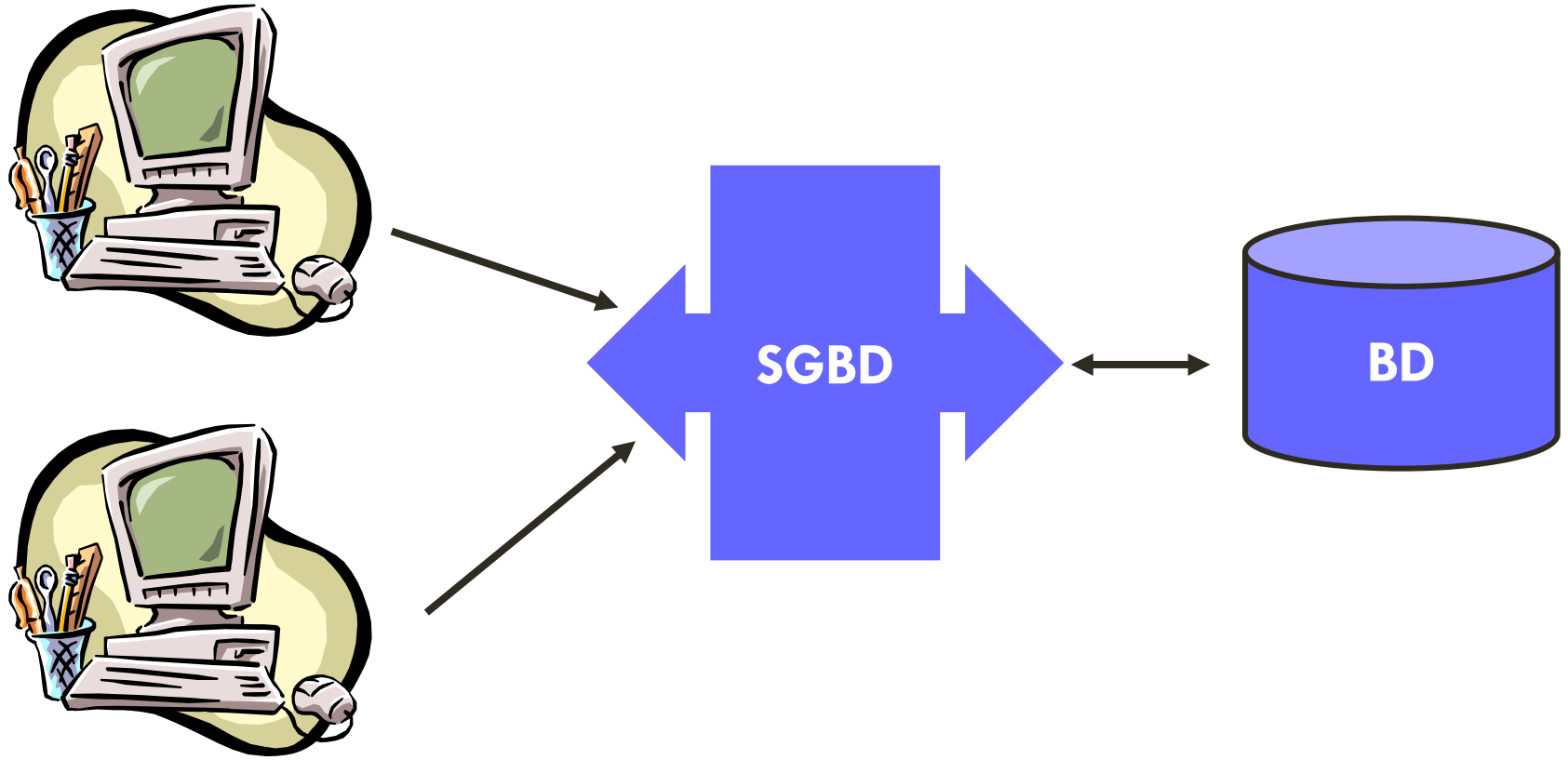
- **Micro systèmes :** MySQL, PostgreSQL, SQLite, MSAccess, Interbase, dBase, FireBird,...
- **Gros systèmes :** Oracle, MS SQL Server, DB2, Informix, Sybase, Teradata, Hive,...

Historique des SGBD : http://fadace.developpez.com/sgbdcmp/story/

CONCEPTS DE BASE

Définitions : Système de Gestion de Base de Données (SGBD – DBMS)

Intermédiaire entre les utilisateurs et les fichiers physiques



CONCEPTS DE BASE

Objectifs des SGBD

1. Indépendance physique
2. Indépendance logique
3. Manipulation facile des données
4. Efficacité des accès aux données
5. Redondance contrôlée des données
6. Cohérence des données
7. Partage des données
8. Sécurité des données
9. Administration centralisée des données

PLAN DE LA MATIERE

- Concepts de base
- **Modèle relationnel**
- Algèbre relationnelle
- SQL : Définitions des schémas
- SQL : Mise à jour de données
- SQL : Interrogation simple
- SQL : Fonctions de groupe
- SQL : Extraction de données de plusieurs tables (Jointures)
- SQL : Opérateurs ensemblistes
- SQL : Sous-requêtes
- Dépendances fonctionnelles et Normalisation des relations

MODÈLE RELATIONNEL

Modèles de données

- Un modèle de données est un ensemble de concepts qui permet de décrire les données, les liens entre les données, la sémantique des données, les contraintes d'intégrités sur les données

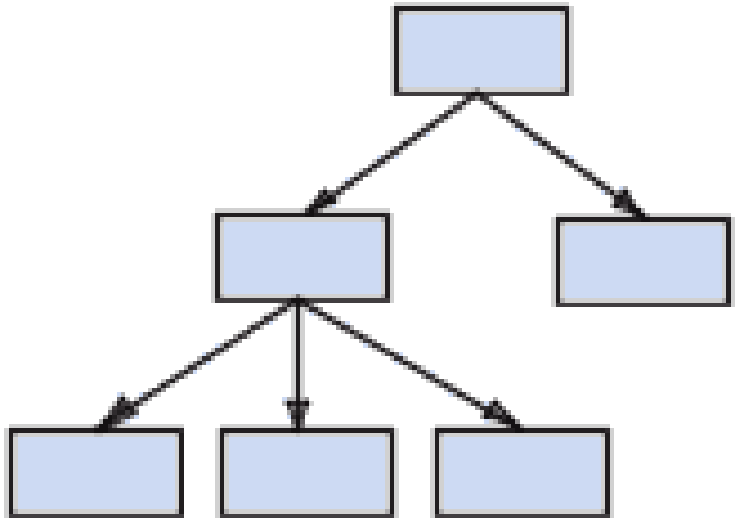
- Plusieurs modèles proposés pour modéliser une BD, chacun ses propres concepts, schémas et règles :

    1. **Hiérarchique**
    2. **Réseau**
    3. **Relationnel**

MODÈLE RELATIONNEL

Modèle Hiérarchique

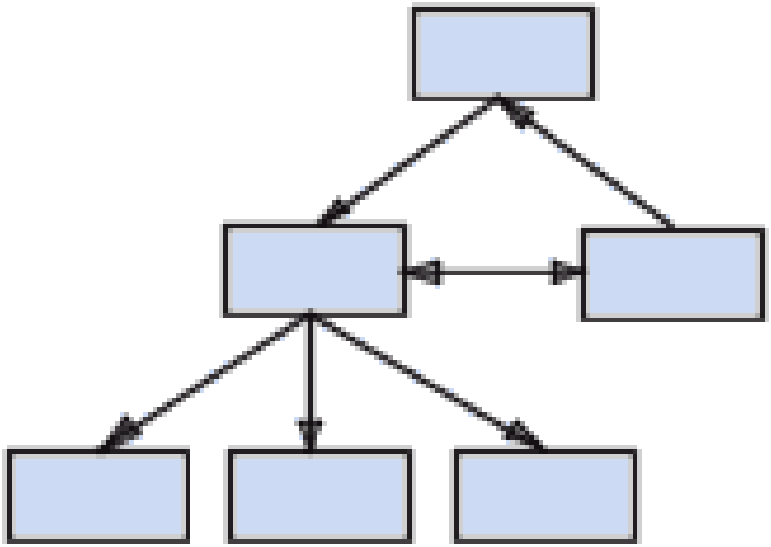
- Les données sont classées hiérarchiquement, selon une arborescence descendante
- Ce modèle utilise des pointeurs entre les différents enregistrements



MODÈLE RELATIONNEL

Modèle Réseau

- Pointeurs entre les entités formant un graphe, offrant plusieurs voies d'accès possible à une information
- La structure n'est plus forcément arborescente dans le sens descendant



MODÈLE RELATIONNEL

Définitions

- Modèle introduit par ***Edgar Codd*** *(IBM 1970)*, Fondé sur des **concepts mathématiques**
- Les entités et associations du mode réel sont représentées par un concept unique « **Relation** »
- Le modèle relationnel représente la BD comme un ensemble de relations
- Les relations sont des tables à 2 dimensions (**lignes, colonnes**)

MODÈLE RELATIONNEL

Définitions

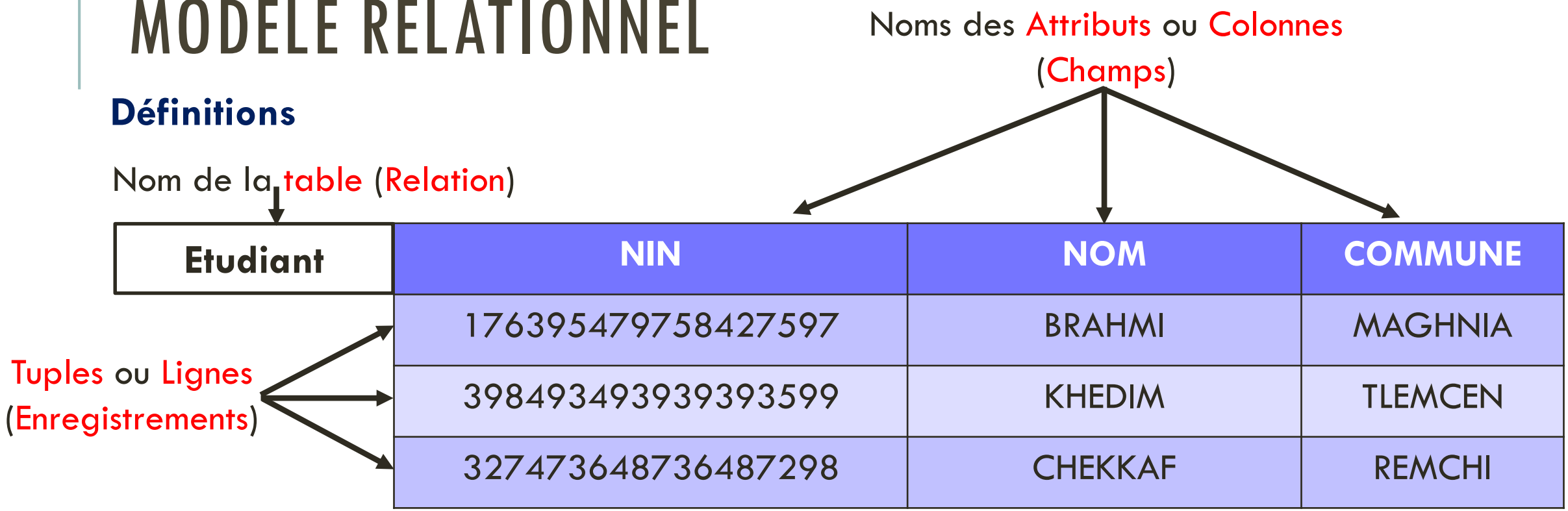
- Chaque ligne de la table est un enregistrement représenté par un ensemble de valeurs reliées correspondant à une réalisation d'entité ou d'association dans le monde réel
- Chaque colonne de la table représente un champ de la BD
- Représentation des données issue du modèle Entité/Association

  Propriété→ Champ ou Attribut ou Colonne

  Occurrence→ Enregistrement ou Tuple ou Ligne
- L'ensemble de ces tables constituent une base de données relationnelle ayant un schéma

MODÈLE RELATIONNEL

Définitions



Etudiant	NIN	NOM	COMMUNE
	1763954797584275 97	BRAHMI	MAGHNIA
	3984934939393599	KHEDIM	TLEMCEN
	3274736487364872 98	CHEKKAF	REMCHI

- Toutes les valeurs d'une colonne sont de même type
- Le type de données introduites dans chaque colonne est représenté par un domaine de valeurs possibles
- Le domaine est l'ensemble fini ou infini des valeurs possibles des données : Entier {0,…., infini}, Booléen {0, 1}, Sit fam {C, M, D, V}, Note [0,20], Date, {Admis, Ajourné}, Sexe {M, F}, Noms {Chaine de car}, ..

MODÈLE RELATIONNEL

Définitions

- **Degré de relation** : Nombre d'attributs
- **Cardinalité de relation** : Nombre de tuples ou instances
- La gestion, la lecture, le stockage, la mise à jour, le partage, la cohérence et la sécurité des données sont assurés par un SGBD Relationnel
- Ce modèle est à la base de nombreux systèmes : Oracle, MS SQL Server, MySQL, SQLite,..

MODÈLE RELATIONNEL

Définitions

- Le Schéma d'une BD est la description de la BD, obtenue en employant un modèle de données
- Le Schéma est la structure de la relation caractérisée par les trois concepts : **Relation, Attribut** et **Domaine**
  - Schéma d'une relation : R (A1 : D1, A2 : D2, ……., An : Dn)
  - **Exemple** : Etudiant (NIN : Integer, Nom : Varchar, Commune : Varchar)
    Format plus simple : Etudiant (NIN , Nom, Commune)
- Schéma d'une BD relationnelle est un ensemble de schémas de tables relationnelles
  - **Exemple** : Etudiant (NumE, Nom, Age)
    Cours (DesC, Horaire, Lieu)
    Suivre (NumE#, DesC#)

MODÈLE RELATIONNEL

Clé primaire

- L'identifiant de l'entité du modèle conceptuel devient une clé primaire de la table relationnelle
- C'est un **ensemble minimum d'attributs** tel qu'il n'existe jamais 2 tuples **ayant mêmes valeurs** pour tous ces attributs regroupés
- En général, la clé est un seul attribut, comme elle peut contenir l'ensemble des attributs de la relation
- La valeur de la clé sert à **identifier un** et **un seul tuple**
- Toute relation doit avoir **une** et **une seule clé primaire** (**Primary Key : PK**)

MODÈLE RELATIONNEL

Clé primaire : Unicité et Présence de valeur

- La clé primaire doit être unique : **Contrainte d'unicité** (UNIQUE)
- La clé primaire doit être renseignée : **Contrainte de présence de valeur** (NOT NULL)

2 Pbs !!

Pb d'unicité

Pb d'absence de valeur

MATRICULE	NOM	COMMUNE
I45	BRAHMI	MAGHNIA
M64	KHEDIM	TLEMCEN
S12	CHEKKAF	REMCHI
M64	ABBACI	NEDROMA
	ARBI	SEBDOU

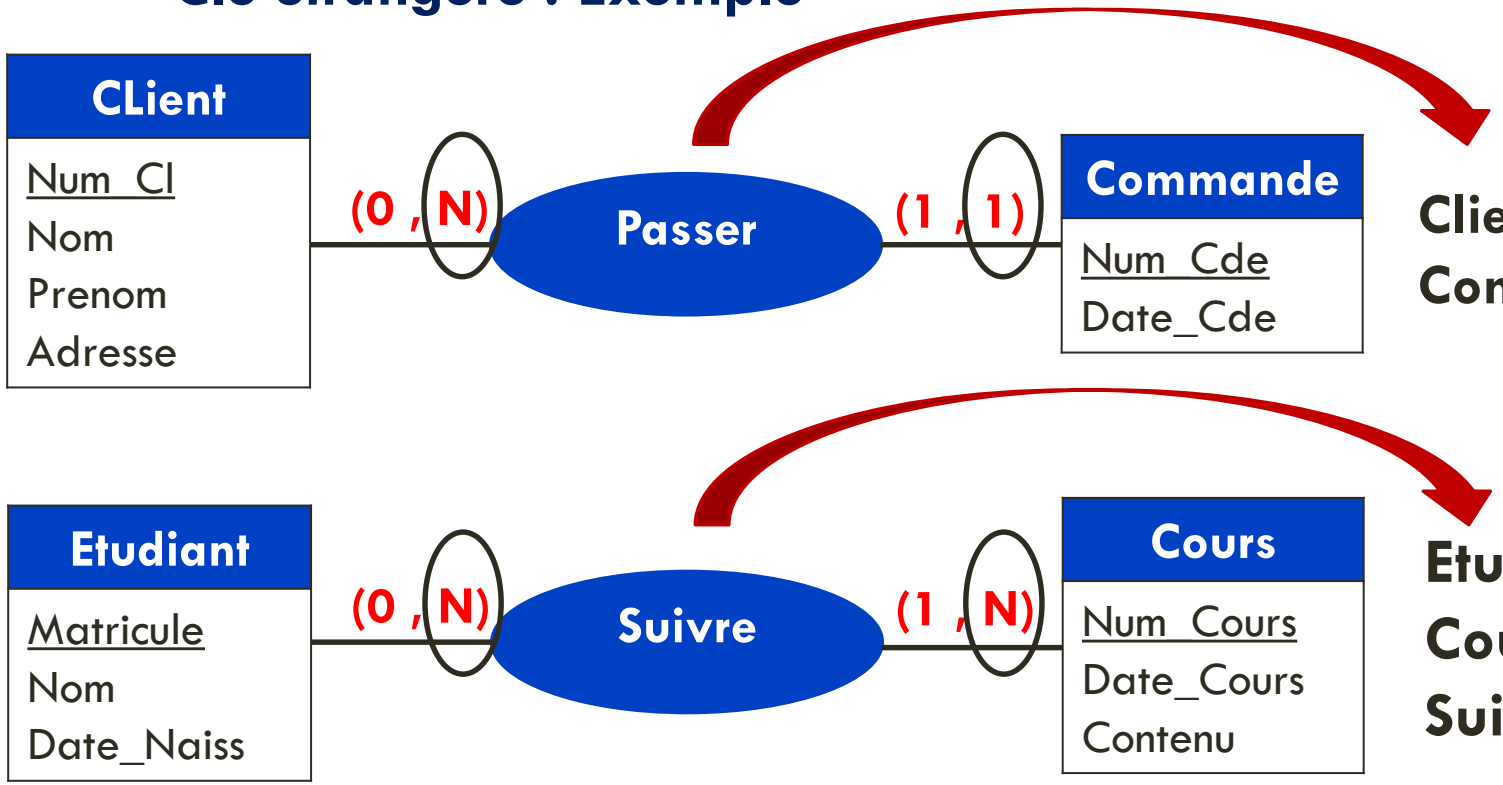
MODÈLE RELATIONNEL

Clé étrangère

- Une clé étrangère est un attribut d'une relation qui fait référence à une clé primaire déclarée dans une autre relation de la même BD (**Foreign Key : FK**)
- La table contenant la clé primaire est dite **Père,** celle qui contient la clé étrangère est dite **Fils**
- FK implémente le lien conceptuel entre deux entités, **utile pour joindre les deux tables**
- La PK et la FK doivent partager le même domaine (Même type et même taille)
- La définition d'une FK impose une contrainte d'intégrité référentielle aux enregistrements des tables père et fils. **SI B fait référence à A, alors A doit exister** :
  - Interdire de saisir une note pour un étudiant qui n'existe pas
  - Interdire de supprimer un client ayant des factures

MODÈLE RELATIONNEL

Clé étrangère : Exemple



Client (Num_Cl, Nom, Prenom, Adresse)
Commande (Num_Cde, Date_Cde, Num_Cl#)

Etudiant (Matricule, Nom, Dat_Naiss)
Cours (Num_Cours, Date_Cours, Contenu)
Suivre (Matricule#, Num_Cours#)

MODÈLE RELATIONNEL

Règles de modélisation : Cas d'un attribut facultatif

- « NULL » représente l'absence de valeur pour tous les types de données

  Ce n'est pas une valeur

MATRICULE	NOM	AGE	ADRESSE
I45	BRAHMI	21	NULL
M64	KHEDIM	NULL	N°5 Rue de l'indépendance
S12	CHEKKAF	20	N°11 Bt A Imama
M64	ABBACI	22	N°78 Cité Nord Ouest

MODÈLE RELATIONNEL

Règles de modélisation : Cas d'un attribut composé

- **Exemple** : Adresse composé de Num_Résidence, Nom_Rue, Ville et Code_Postal

- 2 Solutions possibles :

1. **Un seul attribut Adresse est défini** : si on veut faire des recherches sur la ville, on devra lire l'adresse et chercher dans la chaîne de caractères le nom de la ville

2. **On définit les 4 attributs** (Num_Résidence, Nom_Rue, Ville, Code_Postal) : on peut appliquer des opérations sur chaque attribut mais pour avoir l'adresse, il faut la composer

MODÈLE RELATIONNEL

Règles de modélisation : Cas d'un attribut multivalué

- **Exemple** : Etudiant (Matricule, Nom, Prénom1, Prénom2, Prénom3,…..)
- 2 Solutions possibles :

1. **Définir plusieurs attributs** (Prénom1, Prénom2,..)
   - Il faut savoir le nombre de prénoms maximum à définir
   - Pré-réserver des colonnes même pour les étudiants qui ont un seul prénom
   - Pour rechercher un prénom, il faut parcourir tous les attributs des prénoms

   Mauvaise modélisation

2. **Créer une nouvelle table pour l'attribut multivalué**

   Etudiant (N°Etud, Nom)

   Prénoms_Etudiant (N°Etud#, Prénom)

MODÈLE RELATIONNEL

Langages de définition et manipulation de données

- **Comment** définir le schéma d'une BD relationnelle ?
- **Comment** ajouter de nouvelles données dans une BD ?
- **Comment** modifier et supprimer les données d'une BD ?
- **Comment** chercher, afficher, imprimer les données d'une BD ?

- Besoin de langage pour manipulation et interrogation des données
- Les données d'une BD Rel sont manipulées par le langage algébrique « **Algèbre Relationnelle** »
- Les langages algébriques « **théoriques** » ont dérivé le langage informatique « **SQL** »

PLAN DE LA MATIERE

- Concepts de base
- Modèle relationnel
- **Algèbre relationnelle**
- SQL : Définitions des schémas
- SQL : Mise à jour de données
- SQL : Interrogation simple
- SQL : Fonctions de groupe
- SQL : Extraction de données de plusieurs tables (Jointures)
- SQL : Opérateurs ensemblistes
- SQL : Sous-requêtes
- Dépendances fonctionnelles et Normalisation des relations

ALGÈBRE RELATIONNELLE

Présentation

- Ensemble d'opérateurs qui s'appliquent aux relations pour donner une nouvelle relation
- Ce langage permet l'interrogation de la BD et non pas la définition et la mise à jour

1. **Opérateurs ensemblistes**
   a. Union
   b. Intersection
   c. Différence
   d. Produit cartésien
   e. Division

2. **Opérateurs relationnels**
   a. Projection
   b. Sélection ou Restriction
   c. Jointure

Les opérateurs essentiels et les plus utilisés

ALGÈBRE RELATIONNELLE

Présentation

- Deux sortes d'opérations :

1. **Opérations binaires ou n-aires**
    a. Union
    b. Intersection
    c. Différence
    d. Produit cartésien
    e. Division
    f. Jointure
2. **Opérations unaires**
    a. Projection
    b. Sélection ou Restriction

ALGÈBRE RELATIONNELLE

Union

- L'union de deux relations de même schéma est une nouvelle relation de même schéma contenant l'ensemble des tuples de R1 et de ceux de R2 avec élimination des doublons
- **Notation :** R3 = R1 U R2 R3 = UNION (R1 , R2)

  « = » ou « ← » peuvent être utilisés : R3 ← R1 U R2 ou R3 ← UNION (R1 , R2)
- **Exemple**

ETUDIANT1

CODE	NOM	ADRESSE
I45	BRAHMI	MAGHNIA
M64	KHEDIM	TLEMCEN
S12	CHEKKAF	REMCHI
M27	ABBACI	NEDROMA

ETUDIANT2

CODE	NOM	ADRESSE
M64	KHEDIM	TLEMCEN
M27	ABBACI	NEDROMA
I09	ARBI	SEBDOU

ETUDIANT1 U ETUDIANT2

CODE	NOM	ADRESSE
I45	BRAHMI	MAGHNIA
M64	KHEDIM	TLEMCEN
S12	CHEKKAF	REMCHI
M27	ABBACI	NEDROMA
I09	ARBI	SEBDOU

ALGÈBRE RELATIONNELLE

Intersection

- L'intersection de deux relations de même schéma est une nouvelle relation de même schéma contenant les tuples appartenant à la fois à R1 et à R2
- **Notation :** R3 = R1 ∩ R2 R3 = INTERSECT (R1 , R2)
- **Exemple**

ETUDIANT1

CODE	NOM	ADRESSE
I45	BRAHMI	MAGHNIA
M64	KHEDIM	TLEMCEN
S12	CHEKKAF	REMCHI
M27	ABBACI	NEDROMA

ETUDIANT2

CODE	NOM	ADRESSE
M64	KHEDIM	TLEMCEN
M27	ABBACI	NEDROMA
I09	ARBI	SEBDOU

ETUDIANT1 ∩ ETUDIANT2

CODE	NOM	ADRESSE
M64	KHEDIM	TLEMCEN
M27	ABBACI	NEDROMA

ALGÈBRE RELATIONNELLE

Différence

- La différence de deux relations de même schéma est une nouvelle relation de même schéma contenant les tuples appartenant à R1 et n'appartenant pas à R2
- **Notation :** R3 = R1 – R2 R3 = MINUS (R1 , R2)
- **Exemple**

ETUDIANT1

CODE	NOM	ADRESSE
I45	BRAHMI	MAGHNIA
M64	KHEDIM	TLEMCEN
S12	CHEKKAF	REMCHI
M27	ABBACI	NEDROMA

ETUDIANT2

CODE	NOM	ADRESSE
M64	KHEDIM	TLEMCEN
M27	ABBACI	NEDROMA
I09	ARBI	SEBDOU

ETUDIANT1 – ETUDIANT2

CODE	NOM	ADRESSE
I45	BRAHMI	MAGHNIA
S12	CHEKKAF	REMCHI

- **Rmq 1 :** La différence n'est pas commutative (R1 – R2 <> R2 – R1)
- **Rmq 2 :** R1 ∩ R2 = R1 – (R1 – R2) = R2 – (R2 – R1)

ALGÈBRE RELATIONNELLE

Produit Cartésien

- Le produit cartésien de deux relations est une nouvelle relation, ayant pour attributs la concaténation des attributs de R1 et R2, contenant toutes les possibilités de combinaison des tuples des 2 relations
- Degré R3=Degré R1+ Degré R2 Cardinalité R3=Cardinalité R1 * Cardinalité R2
- **Notation :** R3 = R1 X R2 R3 = R1 ⊗ R2 R3 = PRODUCT (R1 , R2)
- **Exemple**

ETUDIANT

CODE	NOM	ADRESSE
I45	BRAHMI	MAGHNIA
S12	CHEKKAF	REMCHI
M27	ABBACI	NEDROMA

MATIERE

DES	NIVEAU
BD	L2
GL	M1

ETUDIANT X MATIERE

CODE	NOM	ADRESSE	DES	NIVEAU
I45	BRAHMI	MAGHNIA	BD	L2
S12	CHEKKAF	REMCHI	BD	L2
M27	ABBACI	NEDROMA	BD	L2
I45	BRAHMI	MAGHNIA	GL	M1
S12	CHEKKAF	REMCHI	GL	M1
M27	ABBACI	NEDROMA	GL	M1

ALGÈBRE RELATIONNELLE

Division

- La division de R1 sur R2, où R1(A, A2, …, Ap,… , An) et R2(Ap+1, …An), crée une nouvelle relation R3(A1, A2, …, Ap) contenant tous les tuples tels que leur concaténation à chacun des tuples de R2 donne toujours un tuple de R1
- Le schéma de R2 doit être inclut dans R1 et R1 doit posséder au moins un attribut de plus que R2
- La division réduit le degré et la cardinalité de la relation résultat
- Permet de répondre à des questions de la forme « Quelque soit x, Trouver y » ou « Trouver Tous »
- **Notation :** R3 = R1 / R2 R3 = R1 ÷ R2 R3 = **DIVISION** (R1 , R2)
  - **Rmq1** : La division n'est pas commutative (R1 ÷ R2 <> R2 ÷ R1)
  - **Rmq2** : R3 X R2 ⊆ R1

ALGÈBRE RELATIONNELLE

Division

- **Exemple**

ENSEIGNEMENT

CODE_ENS	DES_MATIERE
01	Bases de données
01	Système d'information
01	Système d'exploitation
02	Bases de données
02	Système d'exploitation
03	Bases de données
03	Système d'information
04	Système d'exploitation

MATIERE1

DES_MATIERE
Bases de données
Système d'information

MATIERE2

DES_MATIERE
Bases de données
Système d'exploitation

MATIERE3

DES_MATIERE
Bases de données

ENSEIGNEMENT / MATIERE1

CODE_ENS
01
03

ENSEIGNEMENT / MATIERE2

CODE_ENS
01
02

ENSEIGNEMENT / MATIERE3

CODE_ENS
01
02
03

- **Requête :** Quels sont les enseignants qui enseignent toutes les matières ?

ALGÈBRE RELATIONNELLE

Projection

- La projection consiste à retenir seulement certains attributs (colonnes) d'une relation R1 pour donner une nouvelle relation R2 (Réduction du degré de la relation) avec élimination des doublons
- **Notation :** $R2 = \Pi_{Att1, Att2, ..., AttN} (R1)$ R2 = PROJECT (R1/Att1, Att2, …, AttN)
- **Exemple :** Afficher les noms des étudiants

ETUDIANT

CODE	NOM	ADRESSE
I45	BEKHTAOUI	SEBRA
M64	BERREZOUG	TLEMCEN
S12	BOUALI	GHAZAOUET
M27	BOUALI	SEBDOU

Π_{NOM}(ETUDIANT)

NOM
BEKHTAOUI
BERREZOUG
BOUALI

ALGÈBRE RELATIONNELLE

Sélection (Restriction)

- La sélection appliquée sur R1, crée une nouvelle relation de même schéma R2, contenant seulement les tuples (lignes) qui vérifient une condition Q (Réduction de la cardinalité de la relation)
- Les opérateurs de comparaison utilisés : $=,\neq,<,\leq,>,\geq$ et les opérateurs logiques utilisés : $\neg$, $\wedge$, $\vee$
- **Notation** : $R2 = \sigma_Q (R1)$ $R2 = RESTRICT\ (R1/Q)$ $R2 = SELECT(R1/Q)$
- **Exemple** : Afficher la liste des étudiants qui habitent à Tlemcen

ETUDIANT

CODE	NOM	ADRESSE
I45	BEKHTAOUI	BENSEKRANE
M64	BERREZOUG	TLEMCEN
S12	BOUALI	HENNAYA
M27	GHERNAOUT	TLEMCEN

$\sigma_{ADRESSE=TLEMCEN}$ **(ETUDIANT)**

CODE	NOM	ADRESSE
M64	BERREZOUG	TLEMCEN
M27	GHERNAOUT	TLEMCEN

ALGÈBRE RELATIONNELLE

Combinaison de Projection et Sélection (Exercice)

- **Exemple 1** : Liste des noms des étudiants qui ont pour prénom Mohamed ou Hamza
- Formulation de la requête en AR : Π_{NOM} ($\sigma_{[PRENOM=MOHAMED \vee PRENOM=HAMZA]}$ (Etudiant))

ETUDIANT

CODE	NOM	PRENOM	AGE	ADRESSE
I45	KHERROUS	RAHMA	20	AIN YOUCEF
M64	MOUMENI	MOHAMED	20	TLEMCEN
S12	MEHADJI	MOHAMED	21	REMCHI
M27	MANSRI	SIHEM	19	TLEMCEN
I09	RAHMANI	HAMZA	22	TLEMCEN

RESULTAT

NOM
MOUMENI
MEHADJI
RAHMANI

ALGÈBRE RELATIONNELLE

Combinaison de Projection et Sélection (Exercice)

- **Exemple 2:** Liste des noms et prénoms des étudiants résidant à Tlemcen ayant un âge moins de 21 ans
- Formulation de la requête en AR : $\Pi_{\text{NOM, PRENOM}}$ ($\sigma_{\text{[ADRESSE=TLEMCEN} \wedge \text{ AGE < 21]}}$ (Etudiant))

ETUDIANT

CODE	NOM	PRENOM	AGE	ADRESSE
I45	MOHAMMEDI	MERIEM	20	MAGHNIA
M64	ILES	WALID	19	TLEMCEN
S12	ANITER	HICHAME	21	REMCHI
M27	BELARABI	ASMAA	21	TLEMCEN
I09	BOUARFA	AMEL	20	TLEMCEN

RESULTAT

NOM	PRENOM
ILES	WALID
BOUARFA	AMEL

ALGÈBRE RELATIONNELLE

Jointure

- La jointure de deux relations est une nouvelle relation, ayant pour attributs la concaténation des attributs de R1 et R2, contenant toutes les possibilités de combinaison des tuples des 2 relations satisfaisant un critère de comparaison
- La jointure de deux relations R1 et R2 selon une qualification Q est l'ensemble des tuples du produit cartésien R1 X R2 satisfaisant la qualification Q : $\sigma_Q (R \times S)$
- La jointure est l'opération inverse de la projection (Augmentation du degré de la relation)
- Permet l'utilisation raisonnable du produit catésien

ALGÈBRE RELATIONNELLE

Jointure

- Le critère de sélection ou de comparaison s'applique à un attribut de R1 et à un attribut de R2 définis sur le même domaine

<Attribut1> <Opérateur> <Attribut2>

- **Notation :** $R3 = R1 \underset{Q}{\bowtie} R2$ R3 = JOIN (R1,R2/Q)

  Q : Condition de jointure

Selon le type d'opérateur :

- **Thêta-jointure** ou **Inéqui-jointure** (≠,<,≤,>,≥)
- **Equi-jointure** (=)

ALGÈBRE RELATIONNELLE

Thêta-jointure

- **Exemple :** Jointure des deux tables NOTE-ETUD et AGE-ETUD avec Note < Age

NOTE-ETUD

CODE	NOM	NOTE
I45	MOHAMMEDI	15
M64	MOUMENI	10
S12	BOUARFA	19
M27	BELARABI	20

AGE-ETUD

CODE	AGE
S58	20
M44	19
I26	17

JOIN (NOTE-ETUD,AGE-ETUD / Note<Age)

CODE	NOM	NOTE	CODE	AGE
I45	MOHAMMEDI	15	S58	20
I45	MOHAMMEDI	15	M44	19
I45	MOHAMMEDI	15	I26	17
M64	MOUMENI	10	S58	20
M64	MOUMENI	10	M44	19
M64	MOUMENI	10	I26	17
S12	BOUARFA	19	S58	20

ALGÈBRE RELATIONNELLE

Equi-jointure

- **Exemple 1 :** Jointure des deux tables NOTE-ETUD et AGE-ETUD avec Note = Age

NOTE-ETUD

CODE	NOM	NOTE
I45	SAHEL	10
M64	OMARI	17
S12	SEDDAR	18
M27	MOULAI	20

AGE-ETUD

CODE	AGE
S58	20
M44	19
I45	17

JOIN (NOTE-ETUD,AGE-ETUD / Note=Age)

CODE	NOM	NOTE	CODE	AGE
M64	OMARI	17	I45	17
M27	MOULAI	20	S58	20

- **Exemple 2 :** Jointure des deux tables NOTE-ETUD et AGE-ETUD avec Code = Matr

NOTE-ETUD

CODE	NOM	NOTE
I45	SARI	17
M64	CHIALI	15
S12	OUAHRANI	18
M27	RADJA	12

AGE-ETUD

MATR	AGE
S12	20
M44	21
I45	19

JOIN (NOTE-ETUD,AGE-ETUD / Code=Matr)

CODE	NOM	NOTE	MATR	AGE
I45	SARI	17	I45	19
S12	OUAHRANI	18	S12	20

ALGÈBRE RELATIONNELLE

Equi-jointure

NOTE-ETUD

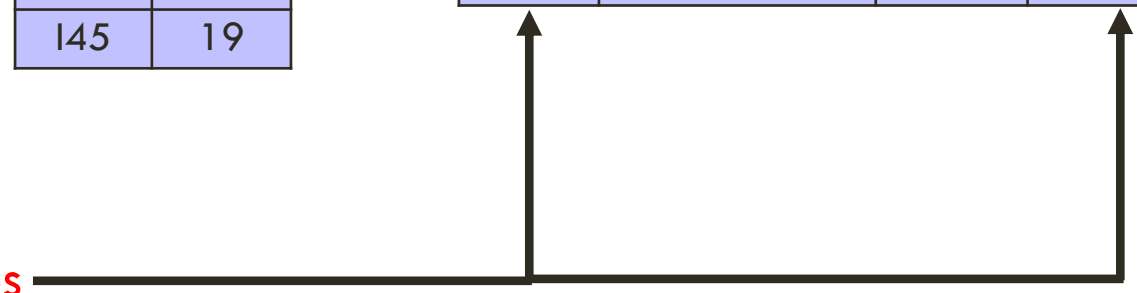
CODE	NOM	NOTE
I45	SARI	17
M64	CHIALI	15
S12	OUAHRANI	18
M27	RADJA	12

AGE-ETUD

MATR	AGE
S12	20
M44	21
I45	19

JOIN (NOTE-ETUD,AGE-ETUD / Code=Matr)

CODE	NOM	NOTE	MATR	AGE
I45	SARI	17	I45	19
S12	OUAHRANI	18	S12	20



- Deux colonnes redondantes
- Pourquoi les garder toutes les deux ?

ALGÈBRE RELATIONNELLE

Jointure Naturelle

- La jointure naturelle de R1 et R2 est une équi-jointure sur les attributs équivalents suivie de la projection qui permet de conserver un seul de ces attributs égaux
- Il faut que les schémas de R1 et R2 possèdent un attribut en commun
- En pratique, c'est la jointure la plus utilisée
- La jointure la plus logique est la jointure entre la clé primaire de la table père (PK) et la clé étrangère de la table fils (FK)
- **Notation :** $R3 = R1 \bowtie R2$ $\qquad$ $R3 = \text{JOIN}(R1,R2)$

$$= \Pi\ (\ \sigma_{Q}\ (R1 \times R2)\ )$$

ALGÈBRE RELATIONNELLE

Jointure Naturelle

- **Exemple :** Liste des étudiants avec leurs notes

ETUDIANT

CODE	NOM	AGE
I45	SARI	20
M64	CHIALI	19
S12	SAHEL	22
I51	OMARI	20
S88	SEDDAR	19
M27	MOULAI	21

NOTE

CODE	NOTE
S12	11
M44	15
I45	9
S88	17
I51	6

ETUDIANT ⋈ NOTE

CODE	NOM	AGE	NOTE
I45	SARI	20	9
S12	SAHEL	22	11
I51	OMARI	20	6
S88	SEDDAR	19	17

ALGÈBRE RELATIONNELLE

Requête en Algèbre Relationnelle

- Plusieurs opérateurs d'Algèbre relationnelle peuvent être combinés pour exprimer une requête
- **Requête** : Composition ou enchainement d'opérateurs d'Algèbre relationnelle
- Opérateurs fréquemment utilisés :
  - **Projection** Π
  - **Sélection** σ
  - **Jointure naturelle** $\bowtie$

ALGÈBRE RELATIONNELLE

Requête en Algèbre Relationnelle (Exercice)

- **Exemple :** Liste des noms des étudiants admis ne résidant pas à Tlemcen et ayant un âge plus de 19 ans, avec leurs moyennes

- Formulation de la requête en AR : $\Pi_{\text{NOM, MOYENNE}} (\sigma_{[\text{ADRESSE}<>\text{TLEMCEN} \wedge \text{AGE} > 19 \wedge \text{Moyenne} \geq 10]} (\text{Etudiant} \bowtie \text{NOTE}))$

ETUDIANT

CODE	NOM	Ville	AGE
I51	OMARI	SEBDOU	20
M64	CHIALI	TLEMCEN	19
S12	SAHEL	BENISAF	22
I45	SARI	TLEMCEN	20
S88	SEDDAR	HONAINE	21
M27	MOULAI	OULED MIMOUN	19

NOTE

CODE	MOYENNE
S12	11
M44	15
I45	9
S88	13
I51	6

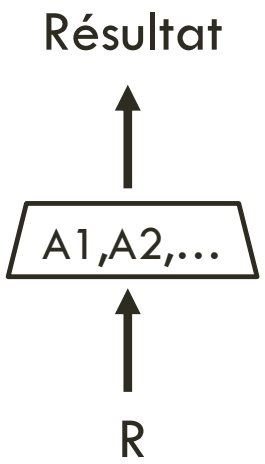
RESULTAT

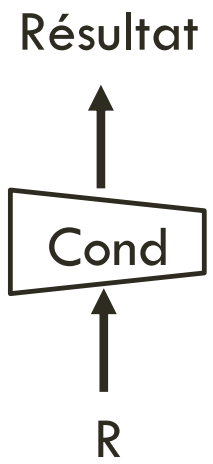
NOM	MOYENNE
SAHEL	11
SEDDAR	13

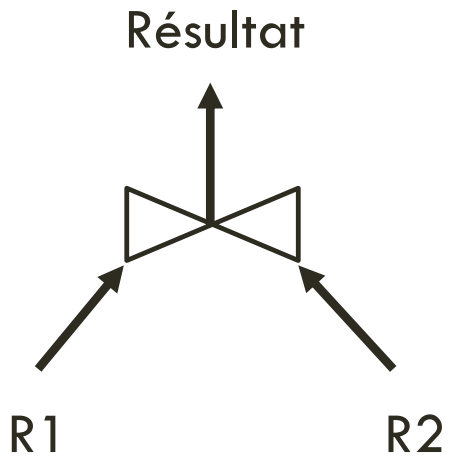
ALGÈBRE RELATIONNELLE

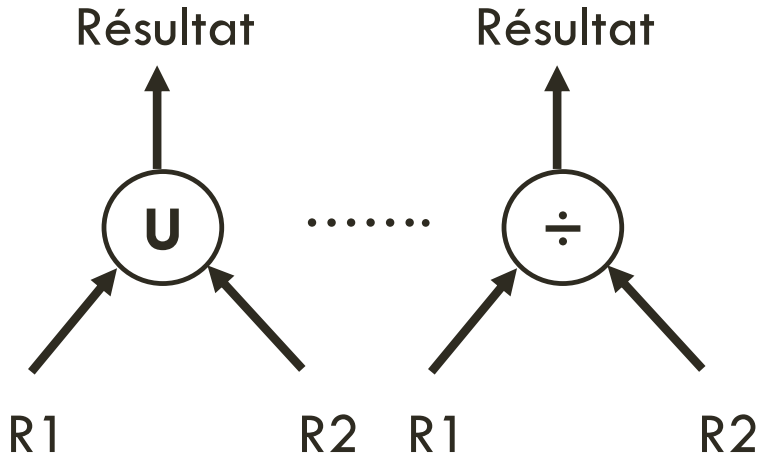
Représentation graphique des opérateurs

- Projection
- Sélection
- Jointure
- 5 Autres opérateurs





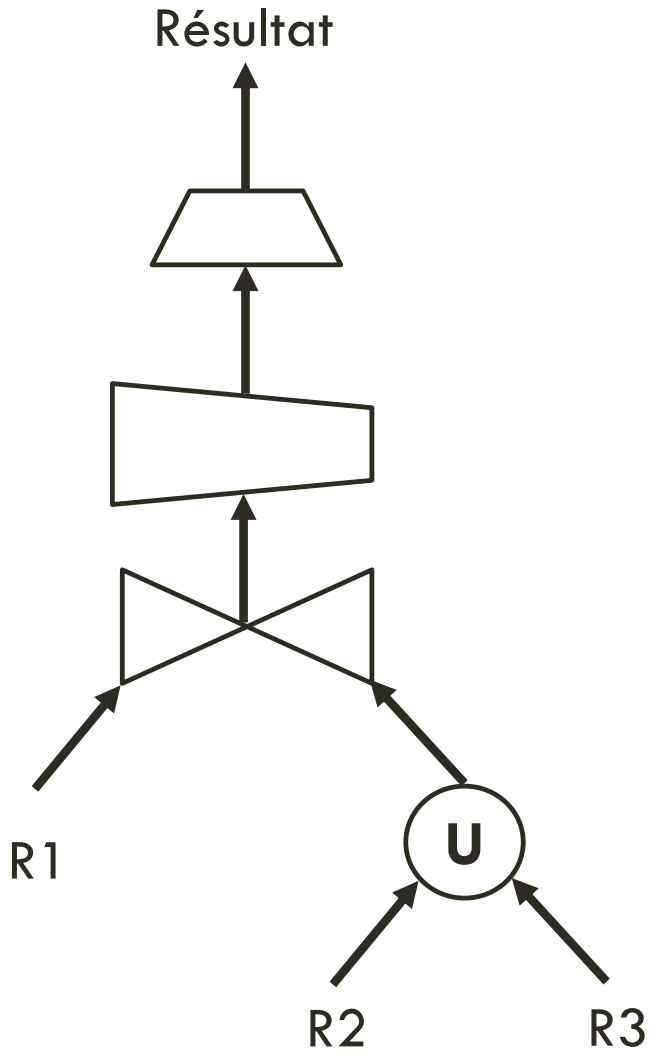




ALGÈBRE RELATIONNELLE

Arbre algébrique

- L'arbre algébrique ou l'arbre de requête est une description graphique plus lisible à traduire
- Un arbre de requête est une structure de données arborescente qui correspond à une expression de l'Algèbre relationnelle :
  - Les relations fournies en entrée à la requête sont présentées sous forme de nœuds feuilles dans l'arbre
  - Les opérateurs de l'Algèbre relationnelle sont présentées sous forme de nœuds internes
- L'arbre algébrique illustre l'ordre d'exécution des opérateurs



PLAN DE LA MATIERE

- Concepts de base
- Modèle relationnel
- Algèbre relationnelle
- **SQL : Présentation**
- SQL : Définitions des schémas
- SQL : Mise à jour de données
- SQL : Interrogation simple
- SQL : Fonctions de groupe
- SQL : Extraction de données de plusieurs tables (Jointures)
- SQL : Opérateurs ensemblistes
- SQL : Sous-requêtes
- Dépendances fonctionnelles et Normalisation des relations

SQL

Présentation

- **S**tructured **Q**uery **L**anguage est un langage complet de gestion de BD relationnelles
- **Langage d'Interrogation Structuré** qui consiste à **définir , manipuler, contrôler** les BDs
- Un **standard** des **BD R**elationnelles (ISO et ANSI)
- Chaque éditeur tend de développer son propre **dialecte** : rajouter des éléments hors normes qui sont fonctionnellement identiques mais de syntaxes différentes

SQL

Présentation

- SQL est un langage de type « **Déclaratif** » : on spécifie ce qu'on veut et c'est la machine qui décide comment elle doit l'exécuter
- Une commande s'appelle **un ordre,** une instruction s'appelle **une requête**
- Chaque requête doit terminer par un « **;** »
- Une requête peut être utilisée de manière **interactive** ou incluse dans **un programme**
- Les systèmes relationnels sont dénommés dans plusieurs travaux récents par les **systèmes SQL**

SQL

Présentation

Date de première version	1974
Auteur	Donald Chamberlin et Raymond Boyce
Développeur	IBM
Dernière version stable	SQL 2011
Paradigme	Déclaratif
Dialectes	SQL 86, SQL 89(SQL1), SQL 92(SQL2), SQL 99(SQL3), SQL 2003, SQL 2008, SQL 2011
Système d'exploitation	Multi-Plateforme

SQL

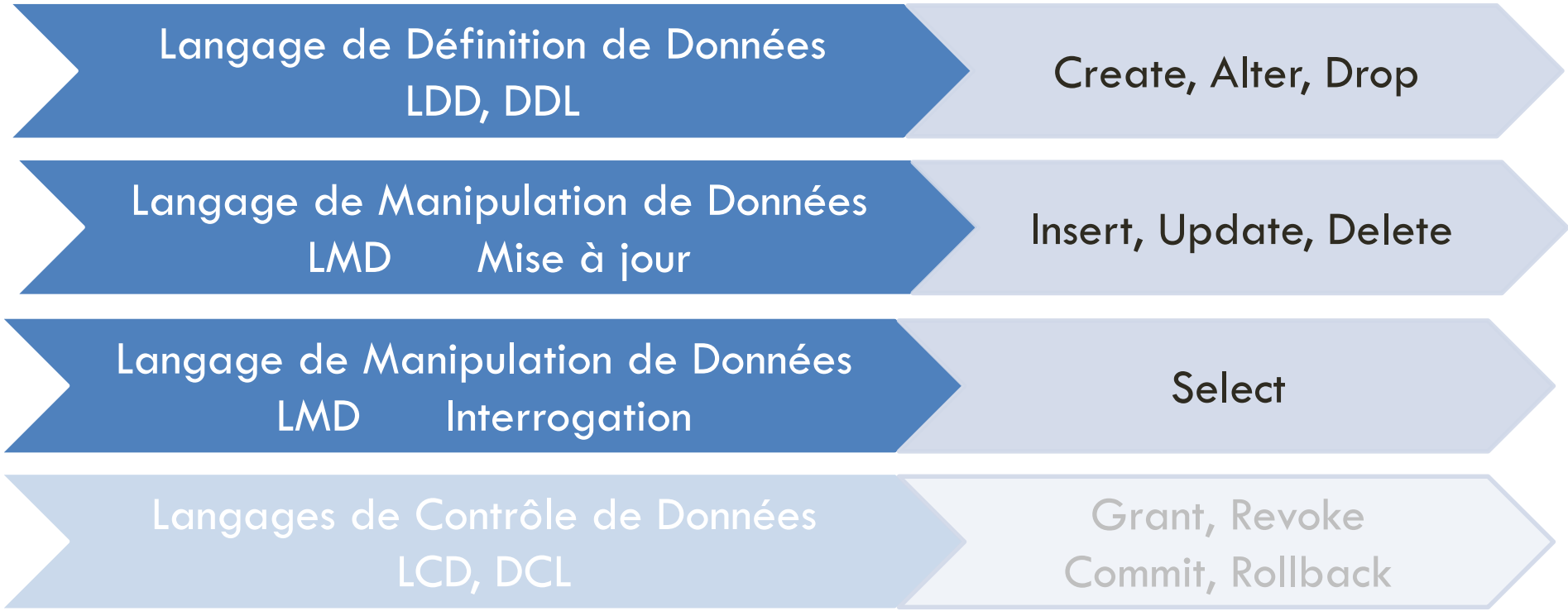
Présentation

- Composé de 3 Sous langages :

Langage de Définition de Données LDD, DDL	Create, Alter, Drop
Langage de Manipulation de Données LMD, DML	Insert, Update, Delete Select
Langages de Contrôle de Données LCD, DCL	Grant, Revoke Commit, Rollback

SQL

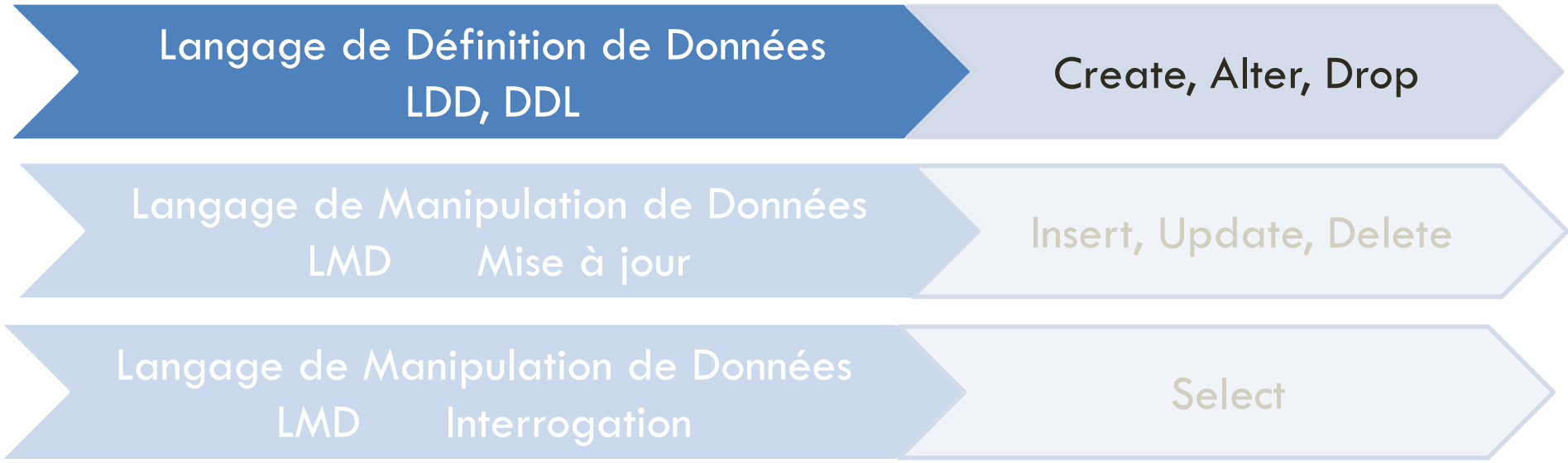
Programme L2



PLAN DE LA MATIERE

- Concepts de base
- Modèle relationnel
- Algèbre relationnelle
- SQL : Présentation
- **SQL : Définitions des schémas**
- SQL : Mise à jour de données
- SQL : Interrogation simple
- SQL : Fonctions de groupe
- SQL : Extraction de données de plusieurs tables (Jointures)
- SQL : Opérateurs ensemblistes
- SQL : Sous-requêtes
- Dépendances fonctionnelles et Normalisation des relations

SQL - Langage de Définition de Données



SQL - Langage de Définition de Données

Création de table

- **CREATE TABLE <Nom Table>** (Attribut1 Type [Contrainte_integrité],
  Attribut2 Type [Contrainte_integrité],
  ……..…,
  ……..……,
  [Contrainte_integrité], …) ;

  - La requête peut être exprimée dans une ligne ou étalée sur plusieurs lignes
  - Toutes les valeurs d'une colonne sont de même type
  - « Describe » ou « Desc » permet d'afficher la structure de la table

SQL - Langage de Définition de Données

Création de table

Types colonnes

- **SMALLINT** : Entiers courts codés sur 2 octets
- **INTEGER** : Entiers codés sur 4 octets
- **BIGINT** : Entiers longs codés sur 8 octets
- **REAL** : Réel comportant 6 chiffres significatifs codés sur 4 octets
- **DOUBLE PRECISION** : Permet de stocker des réels comportant 15 chiffres significatifs codés sur 8 octets
- **NUMERIC (p, q)** : Nombre de décimaux de p chiffres dont q après la virgule
- **NUMBER** : nombre en virgule flottante avec jusqu'à 38 chiffres significatifs
- **CHAR (p)** : Chaîne de caractères de longueur fixe de p caractères
- **VARCHAR (p)** : Chaîne de caractère de longueur variable au maximum p caractères
- **DATE** : Date **TIME** : Heure, Minute, Seconde
- **TIMESTAMP** : Temps précis (Date , Heure, Minute, Seconde)
- **BOOLEAN** : Valeur Booléenne

- Chaque SGBD ajoute des types spécifiques

SQL - Langage de Définition de Données

Création de table

- Contraintes d'intégrité
  - NOT NULL
  - DEFAULT
  - UNIQUE
  - CHECK
  - PRIMARY KEY
  - FOREIGN KEY

SQL - Langage de Définition de Données

Création de table

Contraintes d'intégrité

- **NOT NULL** : Force la saisie d'une colonne (spécifiée implicitement pour les attributs qui font partie de la clé primaire)
- **DEFAULT** <Valeur> : Précise une valeur par défaut qui est incluse dans tout nouveau tuple qui n'a pas été renseigné
- **Ex : CREATE TABLE** Etudiant

```
(Matricule VARCHAR(3),
Nom VARCHAR(20)  NOT NULL,
Prenom VARCHAR(15),
Date_Naiss DATE     NOT NULL,
Annee_Insc Integer   DEFAULT 2017 );
```

SQL - Langage de Définition de Données

Création de table

Contraintes d'intégrité

- **UNIQUE [(Attribut)]** : Toutes les valeurs contenues dans la colonne doivent être uniques au sein de la table (spécifiée implicitement pour la clé primaire)
- **CHECK** (Condition) : Spécifie les valeurs acceptables pour une colonne
- **Ex : CREATE TABLE** Etudiant

```
(Matricule VARCHAR(3),
Nom VARCHAR(20)          NOT NULL,
Prenom VARCHAR(15)       UNIQUE,
Age INTEGER              CHECK (Age BETWEEN 17 AND 25),
Année_Insc INTEGER       DEFAULT 2017,
Ville VARCHAR(15)        CHECK (Ville IN ('Tlemcen', 'Ain Temouchent') );
```

SQL - Langage de Définition de Données

Création de table

- **Contraintes d'intégrité**
  - **PRIMARY KEY** : Définition d'une clé primaire simple

    CREATE TABLE <Nom Table> (Attribut1 Type **PRIMARY KEY**,
    ……..);
  - **Ex : CREATE TABLE** Etudiant

```
(Matricule VARCHAR(3)     PRIMARY KEY,
 Nom VARCHAR(20)          NOT NULL,
 Prénom VARCHAR(15)       UNIQUE,
 Age INTEGER              CHECK (Age BETWEEN 17 AND 25),
 Année_Insc INTEGER       DEFAULT 2017,
 Ville VARCHAR(15)        CHECK (Ville IN ('Tlemcen', 'Ain Temouchent') );
```

SQL - Langage de Définition de Données

Création de table

- **Contraintes d'intégrité**
  - **PRIMARY KEY** : Définition d'une clé primaire composée

    CREATE TABLE <Nom Table> (Attribut1 Type,
    ……,
    PRIMARY KEY (Attribut1, Attribut2));
  - **Ex : CREATE TABLE** Etudiant
    (Nom VARCHAR(20),
    Prénom VARCHAR(15),
    Age INTEGER NOT NULL,
    Année_Insc INTEGER DEFAULT 2017,
    PRIMARY KEY (Nom, Prénom));

SQL - Langage de Définition de Données

Création de table

- **Contraintes d'intégrité**
  - **FOREIGN KEY** : Définition d'une clé étrangère

```
CREATE TABLE <Nom Table> (Attribut1 Type,
                          Attribut2 Type,
                          Att_Clé_Etr Type REFERENCES TablePère (Clé_Primaire),
                          ……,
                          FOREIGN KEY (Att_Cé_Etr) REFERENCES TablePére (attribut) );
```

SQL - Langage de Définition de Données

Création de table

- **Contraintes d'intégrité**
  - **FOREIGN KEY** : Définition d'une clé étrangère
  - **Ex : CREATE TABLE** Etudiant (Matricule …)

    CREATE TABLE Cours (CodeCours …)

    CREATE TABLE Suivre

    (Matricule VARCHAR(3) REFERENCES Etudiant (Matricule),
    CodeCours NUMBER,
    PRIMARY KEY (Matricule, CodeCours),
    FOREIGN KEY (CodeCours) REFERENCES Cours (CodeCours));

SQL - Langage de Définition de Données

Modification de structure

- **ALTER TABLE <Nom_Table>**

  ADD Attribut Type [Contrainte_integrité] |

  MODIFY/CHANGE Attribut Type [Contrainte_integrité] |

  RENAME TO Nouveau_Nom_Table |

  RENAME COLUMN Old_Name TO New_Name |

  DROP COLUMN Nom-Col ;

  - **COLUMN** : Facultatif dans plusieurs SGBD
  - **MODIFY** ou **CHANGE** dans quelques SGBD permet aussi de renommer les colonnes

SQL - Langage de Définition de Données

Modification de structure

- **Ajouter une colonne**

```
ALTER TABLE Etudiant ADD Telephone INTEGER NOT NULL;
```

- **Modifier une colonne**

```
ALTER TABLE Etudiant MODIFY Telephone VACHAR (12);
```

- **Supprimer une colonne**

```
ALTER TABLE Etudiant DROP Année_Insc;
```

- **Renommer une colonne**

```
ALTER TABLE Etudiant RENAME COLUMN Ville TO Adresse;
```

- **Renommer une table**

```
ALTER TABLE Etudiant RENAME TO Enseignant;
```

SQL - Langage de Définition de Données

Suppression de table

- **DROP TABLE <Nom_Table>** [CASCADE CONSTRAINTS]
  - Supprime la table avec son contenu
  - CASCADE CONSTRAINTS : Supprime toutes les contraintes de clé externe référençant cette table
  - Si on tente à détruire une table dont certains attributs sont référencés sans spécifier CASCADE CONSTRAINTS → Le SGBD va refuser

```
DROP TABLE Etudiant ;
```

PLAN DE LA MATIERE

- Concepts de base
- Modèle relationnel
- Algèbre relationnelle
- SQL : Définitions des schémas
- **SQL : Mise à jour de données**
- SQL : Interrogation simple
- SQL : Fonctions de groupe
- SQL : Extraction de données de plusieurs tables (Jointures)
- SQL : Opérateurs ensemblistes
- SQL : Sous-requêtes
- Dépendances fonctionnelles et Normalisation des relations

SQL - Langage de Manipulation de Données

SQL - Langage de Manipulation de Données

Insertion de données

- **INSERT INTO <Nom Table>** [(Attribut1, Attribut2, Attribut3,….)]
  VALUES (Valeur1, Valeur2, Valeur3,…), (Valeur1, Valeur2, Valeur3,…), … ;
  - Les noms de colonnes sont facultatifs si on respecte **l'ordre de définition** et **toutes les valeurs de colonnes sont fournies**
  - Les attributs non spécifiés seront NULL ou à la valeur par défaut
  - **Ex :** INSERT INTO Etudiant (Nom , Prénom, Age, Année_Insc) VALUES ('SARI', 'RIHAM', 20, 2016);
    INSERT INTO Etudiant VALUES ('BENSAYAH', 'FATIMA', 21, 2015);
    INSERT INTO Etudiant (Age, Prénom, Année_Insc, Nom) VALUES (20, 'AMAL', 2016, 'MAHI');
    INSERT INTO Etudiant (Prenom, Age) VALUES ('MERYEM', 20);
    INSERT INTO Etudiant VALUES ('BENSOUNA', Null, 20, Null);

SQL - Langage de Manipulation de Données

Insertion de données (Exercice)

- **Etudiant (Nom, Prénom, Age, Année_Insc)** : Parmi ces 7 requêtes, quelles qui sont justes ou fausses ?

1. INSERT INTO Etudiant VALUES ('AMARBENSABEUR', 'NADJWA', '2015'); Fausse
2. INSERT INTO Etudiant VALUES ('ANITER', 'HICHAME', 21, 2016), ('ARICHI', '111', 20, 2022), ('ATTAR', '222', 19, 2010), ('BAGHLI', '333', 20, 1920); Juste
3. INSERT INTO Etudiant (Prénom, Age) VALUES ('AMINA', 20), ('ISLAM', 20), ('BAHAR', 'AYMEN', 20); Fausse
4. INSERT INTO Etudiant VALUES (19, 'ROMAISSA', 2016, 'BELDJILALI'); Fausse
5. INSERT INTO Etudiant (Age) VALUES (20); Juste
6. INSERT INTO Etudiant VALUES (Null, 'SOUFYANE', Null, Null); Juste
7. INSERT INTO Etudiant VALUES (Nom , Prénom, Age, Année_Insc) ('BENOMARI', Null, Null); Fausse

SQL - Langage de Manipulation de Données

Modification de données

- **UPDATE** <Nom_Table>
  SET Col1=Exp1, Col2=Exp2, …..
  [**WHERE** Condition] ;
  - **SET** : Spécifie les colonnes à modifier
  - **WHERE** : Clause facultative qui spécifie les lignes concernées par cette modification
  - **Exp** : peut être une valeur, une fonction ou une formule
  - **Ex : Etudiant (Nom, Prénom, Age, Ville, Note)**

```
UPDATE  Etudiant SET Ville = 'MAGHNIA' WHERE Nom='BENOSMAN' ;
UPDATE  Etudiant SET Age= MAX (Age) ;
UPDATE  Etudiant SET Note = Note+2 WHERE Note < 8 ;
```

SQL - Langage de Manipulation de Données

Modification de données (Exercice)

- **Etudiant (Nom, Prenom, Age, Ville, Note)**

1. Remplacer la ville de tous les étudiants résidant à 'Relizane' par 'Mostaganem'

   UPDATE Etudiant SET Ville = 'Mostaganem' WHERE Ville = 'Relizane' ;

2. Augmenter la note de tous les étudiants ayant un âge moins de 20 ans de 20 %

   UPDATE Etudiant SET Note = Note * 1.2 WHERE Age < 20 ;

3. Initialiser tous les âges des étudiants

   UPDATE Etudiant SET Age = 0 ;

4. Enlever tous les prénoms des étudiants habitant à 'Remchi'

   UPDATE Etudiant SET Prenom = NULL WHERE Ville = 'Remchi' ;

SQL - Langage de Manipulation de Données

Suppression de données

- **DELETE FROM** <Nom_Table>
  [**WHERE** Condition] ;

  - Supprime une ou plusieurs lignes d'une table vérifiant une certaine condition
  - La condition est optionnelle : Si elle n'est pas précisée, **toutes !!** les lignes de la table sont supprimées
  - Cette opération n'affecte pas le schéma de la table relationnelle

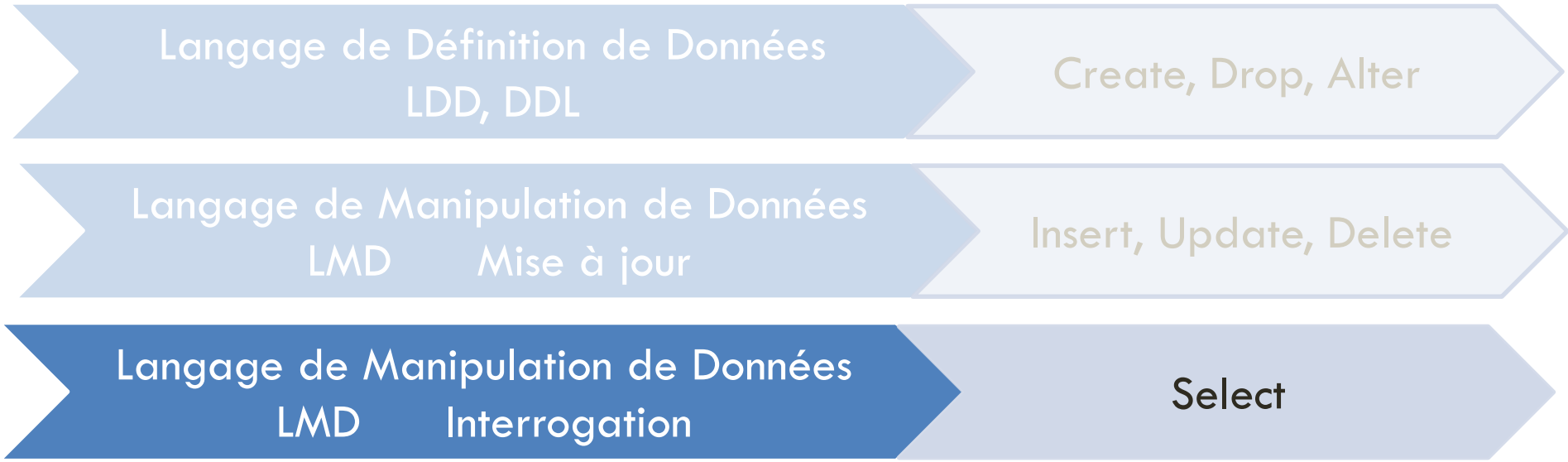
  - **Ex :** Supprimer les étudiants qui ont une note inférieure à 4

    DELETE FROM Etudiant WHERE Note < 4 ;

    Vider la table Etudiant

    DELETE FROM Etudiant ;

SQL - Langage de Manipulation de Données



PLAN DE LA MATIERE

- Concepts de base
- Modèle relationnel
- Algèbre relationnelle
- SQL : Définitions des schémas
- SQL : Mise à jour de données
- **SQL : Interrogation simple**
- SQL : Fonctions de groupe
- SQL : Extraction de données de plusieurs tables (Jointures)
- SQL : Opérateurs ensemblistes
- SQL : Sous-requêtes
- Dépendances fonctionnelles et Normalisation des relations

SQL - Langage d'Interrogation de Données

Schéma de la BD de Vente de Voitures d'Occasions

- **Voiture** (MatV, Marque, Type, Couleur, Km)
- **Client** (CodeC, Nom, Age, Ville, Sexe)
- **Vente** (Num, DateVente, Prix, MatV#, CodeC#)

VOITURE

MatV	Marque	Type	Couleur	Km(10^3)
11	PEUGEOT	307	Marron	150
22	CITROEN	C3	Noir	210
33	PEUGEOT	206	Gris	270
44	RENAULT	Clio		180
55	FIAT	Punto	Rouge	120
66	RENAULT	Megane	Noir	90

CLIENT

CodeC	Nom	Ville	Age	Sexe
1	BOUAYED	TLEMCEN	38	F
2	HEBRI	MAGHNIA	60	M
3	CHERIF	REMCHI	52	M
4	BOUDEHARI	SEBRA	45	F
5	DEMMOUCHE	TLEMCEN	75	M
6	MOUMENI	MAGHNIA	29	F

VENTE

Num	DateVente	Prix (*10^4)	MatV	CodeC
01	03/12/2015	165	11	1
02	30/03/2016	110	22	4
03	14/06/2014	75	44	1
04	02/04/2017	94	55	2
05	05/01/2018	138	66	5

SQL - Langage d'Interrogation de Données

Projection

- **SELECT** < * | Liste_Expressions >
  FROM Nom_Table1 [, Nom_Table2] ... ;

  - **SELECT** : l'ordre le plus important du langage SQL, permettant d'extraire d'une BD, les données répondant à une question particulière et son exécution produit une nouvelle table
  - * : Toutes les colonnes de la table
  - **Liste_Expressions** : Une **colonne**, sinon une **constante** ou une **fonction** ou même une **formule**
  - **Col** : Le nom de la colonne peut ou doit être préfixé par le nom de la table pour éviter les ambiguïtés si les tables ont des colonnes de nom identiques (Table.Col)
  - **FROM** : La table ou les tables contenant les données recherchées

SQL - Langage d'Interrogation de Données

Projection

- **Projection sur toutes les colonnes de la table VENTE**
  - Etat des ventes

```
SELECT * FROM VENTE ;
```

VENTE

Num	DateVente	Prix (Million)	MatV	CodeC
01	03/12/2015	165	11	1
02	30/03/2016	110	22	4
03	14/06/2014	75	44	1
04	02/04/2017	94	55	2
05	05/01/2018	138	66	5

- **Projection sur quelques colonnes de la table CLIENT**
  - Liste des noms des clients avec leurs âges correspondants

```
SELECT Nom, Age FROM CLient ;      ou
SELECT Client.Nom, Client.Age FROM CLient ;
```

RESULTAT

Nom	Age
BOUAYED	38
HEBRI	60
CHERIF	52
BOUDEHARI	45
DEMMOUCHE	75
MOUMENI	29

SQL - Langage d'Interrogation de Données

Projection

- **DISTINCT** : Elimine les éventuelles données dupliquées
  - Afficher les marques des voitures

    SELECT Marque FROM Voiture ;

RESULTAT

Marque
PEUGEOT
CITROEN
PEUGEOT
RENAULT
FIAT
RENAULT

  - Afficher les différentes marques de voitures

    SELECT DISTINCT Marque FROM Voiture ; ou

    SELECT DISTINCT (Marque) FROM Voiture ;

RESULTAT

Marque
PEUGEOT
CITROEN
RENAULT
FIAT

SQL - Langage d'Interrogation de Données

Projection

- **Alias ou Synonymes** : Permet de renommer la colonne à l'affichage et la table dans la requête

1. SELECT MatV AS 'Matricule Voiture', KM AS 'Kilométrage' FROM Voiture ;
   - Utile pour un affichage compréhensible de colonnes

RESULTAT

Matricule Voiture	Kilométrage
11	150
22	210
33	270
44	180
55	120
66	90

1. SELECT **V**.Num, **V**.Prix FROM Vente AS **V** ;
   - Utile pour abréger les noms des tables
   - Utile pour donner un nom au résultat d'une requête
   - AS est optionnel

RESULTAT

Num	Prix
01	165
02	110
03	75
04	94
05	138

SQL - Langage d'Interrogation de Données

Projection

- **Utilisation d'expressions calculées à la place de colonne**

Addition	+
Soustraction	-
Multiplication	*
Division	/
Modulo	%

- Afficher les marques, types des voitures avec le kilométrage en Km

```
SELECT Marque, Type, Km * 10^3 FROM Voiture ;
```

RESULTAT

Marque	Type	Km
PEUGEOT	307	150 000
CITROEN	C3	210 000
PEUGEOT	206	270 000
RENAULT	Clio	180 000
FIAT	Punto	120 000
RENAULT	Megane	90 000

- Afficher les matricules de voitures avec leurs prix de vente en € (1€ = 140 DA)

```
SELECT MatV AS 'Matricule' , (Prix*10^4)/140 AS 'Prix €' FROM Vente ;
```

RESULTAT

Matricule	Prix €
11	11 785,7
22	7 857,1
44	5 357,1
55	6 714,3
66	9 857,1

SQL - Langage d'Interrogation de Données

Projection

- **Autres fonctions intégrées (Numériques, Caractères, Dates)**
  - Calculer la racine carré du kilométrage arrondi

```
SELECT ROUND (SQRT (Km), 2) AS 'Racine Km' FROM Voiture ;
```

  - Transformer la colonne marque en majuscules de la table Voiture

```
SELECT UPPER (Marque) AS 'Marque Majuscule' FROM Voiture ;
```

  - Extraction du mois de la colonne DateVente de la table Vente

```
SELECT MONTH (DateVente) AS 'Mois de Vente' FROM Vente ;
```

RESULTAT

Racine Km
12,25
14,49
16,43
13,42
10,95
9,49

RESULTAT

Marque Majuscule
PEUGEOT
CITROEN
PEUGEOT
RENAULT
FIAT
RENAULT

RESULTAT

Mois de Vente
12
3
6
4
1

SQL - Langage d'Interrogation de Données

Projection

- **Fonctions statistiques de traitement des données d'une colonne (fonctions de groupe)**

MAX	Maximum des valeurs d'une colonne
MIN	Minimum des valeurs d'une colonne
AVG	Moyenne des valeurs d'une colonne
SUM	Somme des valeurs d'une colonne
COUNT	Comptage du nombre de lignes de la table (Cardinalité)

- Calcul de la moyenne et la somme des prix de vente

  SELECT AVG (Prix) AS 'Prix Moyen', SUM(Prix) FROM Vente ;

Prix Moyen	SUM(Prix)
116.4	582

- Donner le plus jeune et le plus âgé Client

  SELECT MIN(Age), MAX(Age) FROM Client ;

MIN(Age)	MAX(Age)
29	75

SQL - Langage d'Interrogation de Données

Projection

- **Fonctions statistiques de traitement des données d'une colonne (Count)**

  - COUNT (*) — Nombre d'enregistrements de la table

    SELECT COUNT (*) FROM Client

Nombre
6

  - COUNT (PK) a changer position — Nombre de valeurs de la clé (= COUNT(*))

    SELECT COUNT (CodeC) FROM Client

Nombre
6

  - COUNT (Col) — Nombre de valeurs renseignées de la colonne

    SELECT COUNT (Couleur) FROM Voiture

Nombre
5

  - COUNT (DISTINCT Col) — Nombre de valeurs distinctes et renseignées

    SELECT COUNT (DISTINCT Sexe) FROM Client

Nombre
2

SQL - Langage d'Interrogation de Données

Projection

- **Affichage de constantes au lieu de colonnes**
  - Lister les codes de clients avec leurs dates de ventes correspondantes et les afficher à un prix de 120

```
SELECT DateVente, CodeC, 120 FROM Vente ;
```

VENTE

DateVente	CodeC	Prix
03/12/2015	1	120
30/03/2016	4	120
14/06/2014	1	120
02/04/2017	2	120
05/01/2018	5	120

SQL - Langage d'Interrogation de Données

Restriction

- **SELECT** < * | Liste_Expressions >
  FROM Nom_Table1 [, Nom_Table2]...
  [**WHERE** Condition] ;

  - **WHERE** : L'opération de sélection (ou restriction) est une **Clause facultative** qui spécifie les critères que doivent satisfaire les lignes retournées dans le résultat
  - Le critère de sélection utilise le contenu des valeurs des colonnes de la table
  - Il est constitué d'expressions de conditions composées à l'aide d'opérateurs de comparaison et combinés à l'aide de connecteurs logiques
  - **N'importe quoi** : WHERE 2=2 (Toutes les lignes sont sélectionnées)
    WHERE 4=5 (Aucune ligne n'est sélectionnée)

SQL - Langage d'Interrogation de Données

Restriction

- **Opérateurs de comparaison**

=	Egal
<>	Différent
<	Inférieur
>	Supérieur
<=	Inférieur ou égal
>=	Supérieur ou égal

- Extraction des ventes dont le prix est supérieur à 100

```
SELECT * FROM Vente WHERE Prix > 100 ;
```

VENTE

Num	DateVente	Prix (*10^4)	MatV	CodeC
01	03/12/2015	165	11	1
02	30/03/2016	110	22	4
05	05/01/2018	138	66	5

SQL - Langage d'Interrogation de Données

Restriction

- **Opérateurs de comparaison spécifiques à SQL**

BETWEEN <valeur> **AND** <valeur>	Appartient à un intervalle
IN <liste de valeurs>	Appartient à un ensemble de valeurs
IS NULL	Teste si la colonne n'est pas renseignée
LIKE	Compare des chaînes de caractères

- Extraction des voitures marron, grises ou rouges

SELECT * FROM Voiture WHERE Couleur IN ('Marron', 'Gris', 'Rouge') ;

VOITURE

MatV	Marque	Type	Couleur	Km(10^3)
11	PEUGEOT	307	Marron	150
33	PEUGEOT	206	Gris	270
55	FIAT	Punto	Rouge	120

SQL - Langage d'Interrogation de Données

Restriction

- **Opérateurs de comparaison spécifiques à SQL**

BETWEEN <valeur> **AND** <valeur>	Appartient à un intervalle
IN <liste de valeurs>	Appartient à un ensemble de valeurs
IS NULL	Teste si la colonne n'est pas renseignée
LIKE	Compare des chaînes de caractères

- Recherche des clients dont l'âge est compris entre 50 et 65 ans

SELECT * FROM Client WHERE Age BETWEEN 50 AND 65 ;

CLIENT

CodeC	Nom	Ville	Age	Sexe
2	HEBRI	MAGHNIA	60	M
3	CHERIF	REMCHI	52	M

SQL - Langage d'Interrogation de Données

Restriction

- **Opérateurs de comparaison spécifiques à SQL**

BETWEEN <valeur> **AND** <valeur>	Appartient à un intervalle
IN <liste de valeurs>	Appartient à un ensemble de valeurs
IS NULL	Teste si la colonne n'est pas renseignée
LIKE	Compare des chaînes de caractères

- Recherche des voitures dont la couleur est inconnue

SELECT * FROM Voiture WHERE Couleur IS NULL ;

VOITURE

MatV	Marque	Type	Couleur	Km(10^3)
44	RENAULT	Clio		180

SQL - Langage d'Interrogation de Données

Restriction

- **Opérateurs de comparaison spécifiques à SQL**

BETWEEN <valeur> **AND** <valeur>	Appartient à un intervalle
IN <liste de valeurs>	Appartient à un ensemble de valeurs
IS NULL	Teste si la colonne n'est pas renseignée
LIKE	Compare des chaînes de caractères « % » Remplace rien ou plusieurs caractères « _ » Remplace un et un seul caractère

- Recherche des clients dont le nom contient la lettre « B »

SELECT * FROM Client WHERE Nom LIKE '%B%' ;

CLIENT

CodeC	Nom	Ville	Age	Sexe
1	BOUAYED	TLEMCEN	38	F
2	HEBRI	MAGHNIA	60	M
4	BOUDEHARI	SEBRA	45	F

SQL - Langage d'Interrogation de Données

Restriction

- **Opérateurs et connecteurs logiques**

AND	Et : les deux conditions sont vraies simultanément
OR	Ou : l'une des deux conditions est vraie
NOT	Inversion de la condition

- Recherche des voitures de marque PEUGEOT dépassant les 200 (10^3) Km

SELECT * FROM Voiture WHERE Marque='PEUGEOT' AND Km > 200 ;

VOITURE

MatV	Marque	Type	Couleur	Km(10^3)
33	PEUGEOT	206	Gris	270

SQL - Langage d'Interrogation de Données

Restriction

- **Opérateurs et connecteurs logiques**

AND	Et : les deux conditions sont vraies simultanément
OR	Ou : l'une des deux conditions est vraie
NOT	Inversion de la condition

- Recherche des clients résidants à Remchi et les clients résidants à Maghnia

SELECT * FROM Client WHERE (Ville='Remchi' OR Ville='Maghnia') ;

CLIENT

CodeC	Nom	Ville	Age	Sexe
2	HEBRI	MAGHNIA	60	M
3	CHERIF	REMCHI	52	M
6	MOUMENI	MAGHNIA	29	F

SQL - Langage d'Interrogation de Données

Restriction

- **Opérateurs et connecteurs logiques**
  - Liste des voitures qui n'ont pas les couleurs marron, grises ou rouges

    SELECT * FROM Voiture WHERE Couleur NOT IN ('Marron', 'Gris', 'Rouge') ;
  - Liste des clients dont l'âge n'est pas compris entre 50 et 65 ans

    SELECT * FROM Client WHERE Age NOT BETWEEN 50 AND 65 ;
  - Liste des voitures dont la couleur est renseignée

    SELECT * FROM Voiture WHERE Couleur IS NOT NULL ;
  - Recherche des clients dont le nom ne contient pas la lettre « B »

    SELECT * FROM Client WHERE Nom NOT LIKE '%B%' ;

SQL - Langage d'Interrogation de Données

Combinaison de Projection et Restriction (Exercice)

- Les noms et la moitié des âges des clientes femmes résidantes à Tlemcen, dont l'âge n'est pas compris entre 35 et 55

SELECT Nom, Age/2 FROM Client

WHERE Sexe='F' AND Ville='Tlemcen' AND Age NOT BETWEEN 35 AND 55 ;

CLIENT

Nom	Age
MOUMENI	14.5

- Les différentes marques de voitures dont la couleur est renseignée ou la marque se termine par la lettre « T »

SELECT DISTINCT Marque FROM Voiture

WHERE Couleur IS NOT NULL OR Marque LIKE '%T' ;

VOITURE

Marque
PEUGEOT
FIAT
RENAULT

PLAN DE LA MATIERE

- Concepts de base
- Modèle relationnel
- Algèbre relationnelle
- SQL : Définitions des schémas
- SQL : Mise à jour de données
- SQL : Interrogation simple
- **SQL : Extraction de données de plusieurs tables (Jointures)**
- SQL : Fonctions de groupe
- SQL : Opérateurs ensemblistes
- SQL : Sous-requêtes
- Dépendances fonctionnelles et Normalisation des relations

SQL - Langage d'Interrogation de Données

Extraction de données de plusieurs tables (Jointures)

- Les jointures permettent d'extraire des données issues de **plusieurs tables**
- Le processus de normalisation du MR est basé sur la décomposition qui va augmenter le nombre de tables d'un schéma
- La majorité des requêtes utilisent les jointures nécessaires pour pouvoir **extraire des données** de **tables distinctes**
- Une jointure met en relation **deux tables** sur la base d'une clause de jointure (comparaison de colonnes)

SQL - Langage d'Interrogation de Données

Jointure

- **SELECT** Col1, Col2, …

  FROM Nom_Table1, Nom_Table2,…

  WHERE Condition de jointure ;

  - Afin d'éviter les ambiguïtés concernant les noms de colonnes, on utilise les alias pour suffixer les tables dans la clause FROM et préfixer les colonnes dans les clauses SELECT et WHERE

- **SELECT** [Alias1.]Col1, [Alias2.]Col2, …

  FROM Nom_Table1 [Alias1] , Nom_Table2 [Alias2],…

  WHERE Condition de jointure ;

SQL - Langage d'Interrogation de Données

Jointure

- En fonction de la nature de l'opérateur utilisé et les tables concernées, on distingue :
  - **Thêta-jointure ou Inéqui-jointure** (<>, <, ≤, >, ≥, BETWEEN, LIKE, IN)
  - **Equi-jointure** (=) (Equi Join)
  - **Jointure naturelle** (Natural Join)
  - **Auto-jointure** (Self Join)

SQL - Langage d'Interrogation de Données

Inéqui-jointure

- Marques, Types de Voitures, Km et Prix de vente ou le Prix de vente est supérieur au Kilométrage

  - **SELECT** Marque, Type
    FROM Vente, Voiture
    WHERE Prix > Km ;

RESULTAT

Marque	Type	Km(10^3)	Prix
PEUGEOT	307	150	165
FIAT	Punto	120	165
FIAT	Punto	120	138
RENAULT	Megane	90	165
RENAULT	Megane	90	110
RENAULT	Megane	90	94
RENAULT	Megane	90	138

SQL - Langage d'Interrogation de Données

Equi-jointure

- Noms, Ages de Clients, Dates de ventes ou le Matricule de la voiture est égal à l'âge Client

```
SELECT Nom, Age, DateVente
FROM Vente, Client
WHERE MatV = Age ;
```

RESULTAT

Nom	Age	DateVente
HEBRI	66	05/01/2018

SQL - Langage d'Interrogation de Données

Jointure Naturelle

- Est une équi-jointure sur les attributs équivalents suivie de la projection qui permet de conserver **un seul** de ces attributs égaux
- La comparaison fait intervenir la **clé étrangère** d'une table avec la **clé primaire** d'une autre table
- En pratique, c'est la jointure la plus utilisée

VOITURE

MatV	Marque	Type	Couleur	Km(10^3)
11	PEUGEOT	307	Marron	150
22	CITROEN	C3	Noir	210
33	PEUGEOT	206	Gris	270
44	RENAULT	Clio	Blanche	180
55	FIAT	Punto	Rouge	120
66	RENAULT	Megane	Noir	90

CLIENT

CodeC	Nom	Ville	Age	Sexe
1	BOUAYED	TLEMCEN	38	F
2	HEBRI	MAGHNIA	66	M
3	CHERIF	REMCHI	52	M
4	BOUDEHARI	SEBRA	45	F
5	DEMMOUCHE	TLEMCEN	75	M
6	MOUMENI	MAGHNIA	29	F

VENTE

Num	DateVente	Prix (*10^4)	MatV	CodeC
01	03/12/2015	165	11	1
02	30/03/2016	110	22	4
03	14/06/2014	75	44	1
04	02/04/2017	94	55	2
05	05/01/2018	138	66	5

SQL - Langage d'Interrogation de Données

Jointure Naturelle

- Noms, Villes, Ages des Clients avec les Matricules de Voitures achetées

  - **SELECT** Nom, Ville, Age, MatV
    FROM Vente, Client
    WHERE Vente.CodeC = Client.CodeC ;
    ou
  - **SELECT** Client.Nom, Client.Ville, Client.Age, Vente.MatV
    FROM Vente, Client
    WHERE Vente.CodeC = Client.CodeC ;

RESULTAT

Nom	Ville	Age	MatV
BOUAYED	TLEMCEN	38	11
BOUAYED	TLEMCEN	38	44
HEBRI	MAGHNIA	66	55
BOUDEHARI	SEBRA	45	22
DEMMOUCHE	TLEMCEN	75	66

SQL - Langage d'Interrogation de Données

Jointure Naturelle

- Noms et Villes des Clients avec les Marques, Types et Couleurs des Voitures achetés

```
SELECT Nom, Ville, Marque, Type, Couleur
FROM Vente, Client, Voiture
WHERE Vente.CodeC = Client.CodeC AND Vente.MatV = Voiture.MatV ;
```

RESULTAT

Nom	Ville	Marque	Type	Couleur
BOUAYED	TLEMCEN	PEUGEOT	307	Marron
BOUAYED	TLEMCEN	RENAULT	Clio	Blanche
HEBRI	MAGHNIA	FIAT	Punto	Rouge
BOUDEHARI	SEBRA	CITROEN	C3	Noir
DEMMOUCHE	TLEMCEN	RENAULT	Megane	Noir

- Une jointure de trois tables est exprimée en deux jointures de deux tables

SQL - Langage d'Interrogation de Données

Jointure Naturelle

- Informations des Clients qui ont acquis des Voitures de Marque RENAULT

```
SELECT CodeC, Nom, Ville, Age, Sexe
FROM Vente, Client, Voiture
WHERE Vente.CodeC = Client.CodeC AND Vente.MatV = Voiture.MatV AND
      Voiture.Marque = 'RENAULT' ;
```

RESULTAT

CodeC	Nom	Ville	Age	Sexe
1	BOUAYED	TLEMCEN	38	F
5	DEMMOUCHE	TLEMCEN	75	M

SQL - Langage d'Interrogation de Données

Auto-jointure (Cas particulier de jointure naturelle, reliant une table avec elle-même)

- Couples de Marques et Types de Voitures ayant la même Couleur
  - **SELECT** V1.Marque, V1.Type, V2.Marque, V2.Type
    FROM Voiture V1, Voiture V2
    WHERE V1.Couleur = V2.Couleur ;
- Pour éliminer les combinaisons inutiles
  - **SELECT** V1.Marque, V1.Type, V2.Marque, V2.Type
    FROM Voiture V1, Voiture V2
    WHERE V1.Couleur = V2.Couleur AND V1.MatV <> V2.MatV ;
- Améliorant davantage les résultats
  - **SELECT** V1.Marque, V1.Type, V2.Marque, V2.Type
    FROM Voiture V1, Voiture V2
    WHERE V1.Couleur = V2.Couleur AND V1.MatV > V2.MatV ;

RESULTAT

Marque	Type	Marque	Type
PEUGEOT	307	PEUGEOT	307
CITROEN	C3	CITROEN	C3
CITROEN	C3	RENAULT	Megane
PEUGEOT	206	PEUGEOT	206
RENAULT	Clio	RENAULT	Clio
FIAT	Punto	FIAT	Punto
RENAULT	Megane	RENAULT	Megane
RENAULT	Megane	CITROEN	C3

RESULTAT

Marque	Type	Marque	Type
CITROEN	C3	RENAULT	Megane
RENAULT	Megane	CITROEN	C3

RESULTAT

Marque	Type	Marque	Type
CITROEN	C3	RENAULT	Megane

PLAN DE LA MATIERE

- Concepts de base
- Modèle relationnel
- Algèbre relationnelle
- SQL : Définitions des schémas
- SQL : Mise à jour de données
- SQL : Interrogation simple
- SQL : Extraction de données de plusieurs tables (Jointures)
- **SQL : Tri du résultat d'une requête**
- SQL : Opérateurs ensemblistes
- SQL : Fonctions de groupe
- SQL : Sous-requêtes
- SQL : Sous interrogations synchronisées
- Dépendances fonctionnelles et Normalisation des relations

SQL - Langage d'Interrogation de Données

Tri du Résultat d'une Requête

- **SELECT** < Expressions >
  FROM Nom_Table1 [, Nom_Table2]...
  [**WHERE** Condition]
  [**ORDER BY** Col1 [ASC|DESC], ...] ;
  - **ORDER BY** : Spécifier la (les) colonne(s) (ou leurs alias) sur laquelle (lesquelles) on souhaite trier le résultat
  - On peut préciser l'ordre de tri par les mots clés **ASC** (Croissant par défaut) ou **DESC** (Décroissant)
- **Afficher les Marques et Types de Voitures triées par Marque**
  - SELECT Marque, Type
    FROM Voiture
    ORDER BY Marque ASC ; (ou ORDER BY Marque)

VOITURE

MatV	Marque	Type	Couleur	Km(10^3)
11	PEUGEOT	307	Marron	150
22	CITROEN	C3	Noir	210
33	PEUGEOT	206	Gris	270
44	RENAULT	Clio		180
55	FIAT	Punto	Rouge	120
66	RENAULT	Megane	Noir	90

RESULTAT

Marque	Type
CITROEN	C3
FIAT	Punto
PEUGEOT	307
PEUGEOT	206
RENAULT	Clio
RENAULT	Megane

SQL - Langage d'Interrogation de Données

Tri du Résultat d'une Requête

- **SELECT** < Expressions >
  FROM Nom_Table1 [, Nom_Table2]...
  [**WHERE** Condition]
  [**ORDER BY** Col1 [ASC|DESC], ...] ;

VENTE

Num	DateVente	Prix (*10^4)	MatV	CodeC
01	03/12/2015	165	11	1
02	30/03/2016	110	22	4
03	14/06/2014	75	44	1
04	02/04/2017	94	55	2
05	05/01/2018	138	66	5

- **Afficher les Dates de ventes, Matricules de Voiture et les Prix du plus cher au moins cher**
  - SELECT DateVente, MatV, Prix
    FROM Vente
    ORDER BY Prix DESC ;

RESULTAT

DateVente	MatV	Prix (*10^4)
03/12/2015	11	165
05/01/2018	66	138
30/03/2016	22	110
02/04/2017	55	94
14/06/2014	44	75

SQL - Langage d'Interrogation de Données

Tri du Résultat d'une Requête

- **SELECT** < Expressions >
  FROM Nom_Table1 [, Nom_Table2]...
  [**WHERE** Condition]
  [**ORDER BY** Col1 [ASC|DESC], ...] ;

CLIENT

CodeC	Nom	Ville	Age	Sexe
1	BOUAYED	TLEMCEN	38	F
2	HEBRI	MAGHNIA	60	M
3	CHERIF	REMCHI	52	M
4	BOUDEHARI	SEBRA	45	F
5	DEMMOUCHE	TLEMCEN	75	M
6	MOUMENI	MAGHNIA	29	F

  - On peut indiquer plusieurs critères de tri, qui sont traités par priorité de gauche à droite

- **Liste des Noms, Ville et Ages des Clients triés par Ville et par Age (plus âgé au plus jeune)**
  - SELECT Nom, Ville, Age
    FROM Client
    ORDER BY Ville ASC, Age DESC ;

RESULTAT

Nom	Ville	Age
HEBRI	MAGHNIA	60
MOUMENI	MAGHNIA	29
CHERIF	REMCHI	52
BOUDEHARI	SEBRA	45
DEMMOUCHE	TLEMCEN	75
BOUAYED	TLEMCEN	38

  - Si deux lignes sont identiques suivant l'expression la plus à gauche, elles sont comparées avec l'expression suivante

SQL - Langage d'Interrogation de Données

Tri du Résultat d'une Requête

- **Utiliser la position de chaque colonne dans l'ordre spécifié dans la clause SELECT**

```
SELECT Nom, Ville, Age
FROM Client
ORDER BY 1, 3 DESC;
```

- **Utiliser des expressions**

```
SELECT Ville, Count (*)
FROM Client
GROUP BY Ville
ORDER BY Count (*) ASC ;
```

- La clause ORDER BY est la dernière clause de toute requête SQL et ne doit figurer qu'une seule fois dans le SELECT, même s'il existe des requêtes imbriquées ou un jeu de requêtes ensemblistes

PLAN DE LA MATIERE

- Concepts de base
- Modèle relationnel
- Algèbre relationnelle
- SQL : Définitions des schémas
- SQL : Mise à jour de données
- SQL : Interrogation simple
- SQL : Extraction de données de plusieurs tables (Jointures)
- SQL : Tri du Résultat d'une Requête
- **SQL : Opérateurs ensemblistes**
- SQL : Fonctions de groupe
- SQL : Sous-requêtes
- SQL : Sous interrogations synchronisées
- Dépendances fonctionnelles et Normalisation des relations

SQL - Langage d'Interrogation de Données

Opérateurs ensemblistes

- Parmi les atouts du MR qu'il est fondé sur une base mathématique (théorie des ensembles)
- SQL prend en charge les opérations ensemblistes binaires suivantes :
  - **Union** (UNION élimine les redondances , UNION ALL n'élimine pas les redondances)
  - **Intersection** (INTERSECT)
  - **Différence** (EXCEPT ou MINUS)
  - **Produit cartésien** (2 tables dans la clause FROM)
- Dans les 3 premiers, les 2 tables doivent avoir le même schéma (Initialement ou après projection)
- Les trois opérateurs Union, Intersection et Produit cartésien sont commutatifs

SQL - Langage d'Interrogation de Données

Schéma de la BD de Vente de Voitures d'Occasions

- **Voiture** (MatV, Marque, Type, Couleur, Km)
- **Client** (CodeC, Nom, Age, Ville, Sexe)
- **Vente** (Num, DateVente, Prix, MatV#, CodeC#)

VOITURE

MatV	Marque	Type	Couleur	Km(10^3)
11	PEUGEOT	307	Marron	150
22	CITROEN	C3	Noir	210
33	PEUGEOT	206	Gris	270
44	RENAULT	Clio	Blanche	180
55	FIAT	Punto	Rouge	120
66	RENAULT	Megane	Noir	90

CLIENT

CodeC	Nom	Ville	Age	Sexe
1	BOUAYED	TLEMCEN	38	F
2	HEBRI	MAGHNIA	60	M
3	CHERIF	REMCHI	52	M
4	BOUDEHARI	SEBRA	45	F
5	DEMMOUCHE	TLEMCEN	75	M
6	MOUMENI	MAGHNIA	29	F

VENTE

Num	DateVente	Prix (*10^4)	MatV	CodeC
01	03/02/2018	165	11	1
02	30/03/2016	110	22	6
03	14/06/2017	75	44	1
04	02/04/2016	94	55	2
05	05/01/2018	138	66	6

SQL - Langage d'Interrogation de Données

Union

- Matricules des Voitures achetées par le Client N°1 ou qui ont un Kilométrage > 160

  - SELECT MatV FROM Vente WHERE CodeC=1
    UNION
    SELECT MatV FROM Voiture WHERE Km > 160 ;

  - SELECT MatV FROM Vente WHERE CodeC=1
    UNION ALL
    SELECT MatV FROM Voiture WHERE Km > 160 ;

RESULTAT

MatV
11
44
22
33

RESULTAT

MatV
11
44
22
33
44

SQL - Langage d'Interrogation de Données

Intersection1

- Codes des Clients qui ont acquis des Voitures en 2016 et en 2018

  - SELECT CodeC FROM Vente WHERE YEAR (DateVente)= '2016'
    INTERSECT
    SELECT CodeC FROM Vente WHERE YEAR (DateVente)= '2018' ;

RESULTAT

CodeC
6

SQL - Langage d'Interrogation de Données

Intersection2

- Codes des Clients qui ont acquis des Voitures en 2016 et en 2018

  - SELECT **DISTINCT** CodeC FROM Vente
    WHERE YEAR (DateVente)= '2016') AND
    CodeC **IN** (SELECT CodeC FROM Vente WHERE YEAR (DateVente)= '2018')

RESULTAT

CodeC
6

SQL - Langage d'Interrogation de Données

Différence1

- Matricules des Voitures non vendues

  - SELECT MatV FROM Voiture
    EXCEPT ou **MINUS**
    SELECT MatV FROM Vente ;

RESULTAT

MatV
33

SQL - Langage d'Interrogation de Données

Différence2

- Matricules des Voitures non vendues

  - SELECT MatV FROM Voiture
    WHERE MatV **NOT IN** (SELECT MatV FROM Vente) ;

RESULTAT

MatV
33

SQL - Langage d'Interrogation de Données

Produit cartésien

- Le produit cartésien est la combinaison de toutes les lignes d'une table avec toutes les lignes d'une autre table sans tenir aucun compte du « sens » associé aux données

SELECT * FROM Client, Voiture ;

CodeC	Nom	Ville	Age	Sexe	MatV	Marque	Type	Couleur	Km(10^3)
1	BOUAYED	TLEMCEN	38	F	11	PEUGEOT	307	Marron	150
2	HEBRI	MAGHNIA	66	M	11	PEUGEOT	307	Marron	150
3	CHERIF	REMCHI	52	M	11	PEUGEOT	307	Marron	150
4	BOUDEHARI	SEBRA	45	F	11	PEUGEOT	307	Marron	150
5	DEMMOUCHE	TLEMCEN	75	M	11	PEUGEOT	307	Marron	150
6	MOUMENI	MAGHNIA	29	F	11	PEUGEOT	307	Marron	150
1	BOUAYED	TLEMCEN	38	F	22	CITROEN	C3	Noir	210
……..……					……..……				

- Le nombre de lignes de la table « résultat » est égal au produit du nombre de lignes des deux tables
- Les colonnes sont celles des deux tables simplement juxtaposées

SQL - Langage d'Interrogation de Données

Traduction des opérateurs de l'Algèbre relationnelle

- **Projection**
  - SELECT Nom, Ville FROM Client ;
- **Restriction**
  - SELECT * FROM Client WHERE Age < 50 ;
- **Produit Cartésien**
  - SELECT * FROM Voiture, Vente ;

SQL - Langage d'Interrogation de Données

Traduction des Opérateurs de l'Algèbre relationnelle

- **Inéqui-jointure**
  - SELECT Marque, Type FROM Vente, Voiture WHERE Prix > Km ;
- **Equi-jointure**
  - SELECT Nom, Age, DateVente FROM Vente, Client WHERE MatV = Age ;
- **Jointure Naturelle**
  - SELECT Nom, Marque, Type FROM Vente, Voiture WHERE Vente.MatV = Voiture.MatV ;

SQL - Langage d'Interrogation de Données

Traduction des Opérateurs de l'Algèbre relationnelle

- **Union**
  - SELECT MatV FROM Vente WHERE CodeC=1
    UNION
    SELECT MatV FROM Voiture WHERE Km > 160 ;
- **Intersection**
  - SELECT CodeC FROM Vente WHERE YEAR (DateVente)= '2016'
    INTERSECT
    SELECT CodeC FROM Vente WHERE YEAR (DateVente)= '2018' ;
- **Différence**
  - SELECT MatV FROM Voiture
    EXCEPT ou MINUS
    SELECT MatV FROM Vente ;

SQL - Langage d'Interrogation de Données

Division

- **Numéros des Voitures qui ont été achetés par tous les Clients (Vente ÷ Client)**

  Décomposition en 2 sous questions :

  a. Le nombre de Clients qui ont acheté chaque Voiture

  - **SELECT MatV, COUNT(DISTINCT (CodeC))**
    FROM Vente
    GROUP BY MatV

  a. Pour avoir les Voitures qui ont été acheté par tous les Clients, on doit ajouter une condition sur le groupe qui va vérifier que ce nombre de Clients est égal au nombre total des Clients existant dans la table Client (**SELECT COUNT(*) FROM Client**)

  - **SELECT MatV**
    FROM Vente
    GROUP BY MatV
    HAVING COUNT(DISTINCT (CodeC)) = (SELECT COUNT(*) FROM Client)

PLAN DE LA MATIERE

- Concepts de base
- Modèle relationnel
- Algèbre relationnelle
- SQL : Définitions des schémas
- SQL : Mise à jour de données
- SQL : Interrogation simple
- SQL : Extraction de données de plusieurs tables (Jointures)
- SQL : Tri du Résultat d'une Requête
- SQL : Opérateurs ensemblistes
- **SQL : Fonctions de groupe**
- SQL : Sous-requêtes
- SQL : Sous interrogations synchronisées
- Dépendances fonctionnelles et Normalisation des relations

SQL - Langage d'Interrogation de Données

Groupage

- **SELECT** < Expressions >

  FROM Nom_Table1 [, Nom_Table2]...

  [**WHERE** Condition]

  [**GROUP BY** Col1 [,Col2], ..] ;

  - **GROUP BY** : Regroupe les lignes d'une table par valeurs contenues dans une colonne, dans des groupes sous forme de sous-tables
  - Chaque ligne du résultat est un groupe de lignes de la table
  - On applique des opérations de type statistique sur les « sous-tables » créées
  - Expressions : Peut inclure des **colonnes présentes dans le Group BY** ou des **fonctions de groupe** (MAX, MIN, AVG, SUM, COUNT)

SQL - Langage d'Interrogation de Données

Groupage

- **Calcul du nombre de Voitures des différentes Marques**

```
SELECT Marque, COUNT(*) AS 'Nombre'
FROM Voiture
GROUP BY Marque ;
```

RESULTAT

Marque	Nombre
PEUGEOT	2
CITROEN	1
RENAULT	2
FIAT	1

- **Calcul de la moyenne d'âge des Clients par Ville**

```
SELECT Ville, AVG (Age) AS 'Moyenne d'âge'
FROM Client
GROUP BY Ville ;
```

RESULTAT

Ville	Moyenne d'âge
TLEMCEN	56.5
MAGHNIA	44.5
REMCHI	52
SEBRA	45

SQL - Langage d'Interrogation de Données

Groupage

SELECT < Expressions >

FROM Nom_Table1 [, Nom_Table2]...

[**WHERE** Condition]

[**GROUP BY** Col1 [,Col2], ..]

[**HAVING** Condition] ;

- **HAVING** : Détermine une condition de sélection de groupe(même option que la condition du WHERE)
- Le résultat de l'opération de groupage peut lui-même être filtré : Il est possible d'éliminer des groupes de la solution obtenue par une requête avec regroupement
- Les conditions de la clause HAVING doivent porter soit sur des champs de la clause SELECT, soit sur une fonction d'agrégation appliquée aux sous-ensembles définis par le groupage

SQL - Langage d'Interrogation de Données

Groupage

- **Affichage du nombre de Voitures de Marque RENAULT dont le nombre est supérieur à 1**

```
SELECT Marque, COUNT(*) AS 'Nombre'
FROM Voiture
GROUP BY Marque
HAVING Marque='RENAULT'  AND  Nombre > 1 ;
```

RESULTAT

Marque	Nombre
RENAULT	2

- **Affichage de la moyenne d'âge des Clients par Ville comprise entre 44 et 54 ans**

```
SELECT Ville, AVG(Age) AS 'Moyenne Age'
FROM Client
GROUP BY Ville
HAVING Moyenne Age BETWEEN 44 AND 54 ;
```

RESULTAT

Ville	Moyenne Age
REMCHI	52
SEBRA	45

SQL - Langage d'Interrogation de Données

Groupage

- **HAVING vs WHERE**
  - HAVING permet d'effectuer une sélection sur le résultat de l'opération de groupage. WHERE opère une sélection sur les lignes de la table avant l'opération de groupage

- **Supposons que l'on veut éliminer les Clientes femmes de notre calcul de la moyenne d'âge**
  - Calcul de la moyenne d'âge des Clients hommes par Ville

```
SELECT Ville, AVG(Age) AS 'Moyenne Age'
FROM Client
WHERE Sexe='M'
GROUP BY Ville
```

RESULTAT

Ville	Moyenne d'âge
TLEMCEN	75
MAGHNIA	60
REMCHI	52

SQL - Langage d'Interrogation de Données

Groupage

- **HAVING vs WHERE**
  - Si la condition de sélection porte sur une des colonnes de la clause SELECT, on peut alors indifféremment l'exprimer soit dans la clause WHERE ou dans la clause HAVING
- **Affichage du nombre de Voitures de Marque PEUGEOT**
  - SELECT Marque, COUNT(*) AS 'Nombre'
    FROM Voiture
    WHERE Marque='PEUGEOT'
    GROUP BY Marque ;
  - SELECT Marque, COUNT(*) AS 'Nombre'
    FROM Voiture
    GROUP BY Marque
    HAVING Marque='PEUGEOT' ;

RESULTAT

Marque	Nombre
PEUGEOT	2

Même résultat

RESULTAT

Marque	Nombre
PEUGEOT	2

SQL - Langage d'Interrogation de Données

Groupage (Exercice)

- **Parmi ces 8 requêtes, quelles qui sont justes ou fausses, Justifier ?**

1. SELECT AVG(Km), COUNT(*) FROM Voiture GROUP BY Couleur ; Juste
2. SELECT MatV, COUNT(Type) FROM Voiture GROUP BY Couleur HAVING COUNT(*) < 2 ; Fausse
3. SELECT Couleur, MAX(MatV) FROM Voiture GROUP BY Couleur HAVING Couleur = 'Blanche' ; Juste
4. SELECT CodeC FROM Vente WHERE DateVente > '31/12/2016' GROUP BY MatV ; Fausse
5. SELECT CodeC, COUNT(Num) FROM Vente GROUP BY CodeC HAVING MIN(Prix) < 90 ; Juste
6. SELECT COUNT(Num) FROM Vente WHERE Prix>100 GROUP BY CodeC HAVING MAX(Prix) <90; Fausse
7. SELECT Nom, Age FROM Client GROUP BY Age WHERE Ville = 'Tlemcen' ; Fausse
8. SELECT Ville, SUM(Age) FROM Client GROUP BY Ville HAVING Age < 30 ; Fausse

SQL - Langage d'Interrogation de Données

Recap

- **SELECT** [**DISTINCT**] Col1, Col2,… — **Quoi ?** Noms des colonnes à afficher
  FROM Table1, Table2,… — **Où ?** Noms des tables ou se trouvent les colonnes
  [**WHERE** Condition] — **Quelle condition ?** Conditions à remplir par les lignes
  [**GROUP BY** Col1,Col2,… — Regroupements des lignes en sous tables
  [**HAVING** Condition] — Condition à remplir par le groupe
  [**ORDER BY** Col1 [ASC|DESC], …] ; — **Comment ?** Ordre d'affichage

PLAN DE LA MATIERE

- Concepts de base
- Modèle relationnel
- Algèbre relationnelle
- SQL : Définitions des schémas
- SQL : Mise à jour de données
- SQL : Interrogation simple
- SQL : Extraction de données de plusieurs tables (Jointures)
- SQL : Tri du Résultat d'une Requête
- SQL : Opérateurs ensemblistes
- SQL : Fonctions de groupe
- **SQL : Sous-requêtes**
- SQL : Sous interrogations synchronisées
- Dépendances fonctionnelles et Normalisation des relations

SQL - Langage d'Interrogation de Données

Sous-requêtes

- SQL utilise les sous-requêtes ou les requêtes imbriquées afin de décrire des requêtes complexes permettant d'effectuer des opérations dépendant d'autres requêtes
- La sous-interrogation doit être placée entre parenthèses et ne doit pas comporter « Order By »
- La sous-interrogation est toujours placée dans la clause « Where » ou « Having » au lieu d'une constante ou une fonction
- La sous-requête est exécutée avant la requête principale
- Le résultat d'une sous-requête est utilisé par la requête de niveau supérieur
- En cas de nécessité, on peut aller jusqu'à plusieurs niveaux d'imbrication

SQL - Langage d'Interrogation de Données

Sous-requêtes

- Une sous-requête peut ramener une **valeur unique** (une seule ligne) :
  - Ces requêtes sont appelées **sous-interrogations monolignes**
  - Les opérateurs classiques : **=, <>, <, >, <=, >=** peuvent être utilisés

- Une sous-requête peut ramener **plusieurs valeurs** (plusieurs lignes) :
  - Ces requêtes sont appelées **sous-interrogations multilignes**
  - Les opérateurs : **IN, NOT IN, ALL, ANY, EXISTS, NOT EXISTS** peuvent être utilisés

SQL - Langage d'Interrogation de Données

Sous-requêtes

- La sous-requête est toujours placée à droite d'un opérateur de comparaison :

Sous-interrogations monolignes

- SELECT …… FROM Vente WHERE **Prix > 100** ;

La constante « **100** » peut être remplacée par une sous-requête qui renvoie une seule valeur

- SELECT …… FROM Vente WHERE **Prix > (SELECT Prix FROM …..)**

Sous-requête

SQL - Langage d'Interrogation de Données

Sous-requêtes

Sous-interrogations multilignes

- SELECT …… FROM Voiture WHERE **Couleur IN ('Marron', 'Gris', 'Rouge')** ;

L'ensemble des couleurs peut être remplacé par une sous-requête qui renvoie plusieurs valeurs

- SELECT …… FROM Voiture WHERE **Couleur IN (SELECT Couleur FROM …..)**

Sous-requête

- Pour que le prédicat soit vrai, la valeur testée doit être présente dans le résultat de sous-requête

SQL - Langage d'Interrogation de Données

Sous-requêtes (Exemple)

COMPAGNIE

Comp	Ville	Pays	NomComp
AA	Alger	ALGERIE	Air Algerie
AZ	Paris	FRANCE	Aigle Azur
AF	Paris	FRANCE	Air France
EJ	Londres	ANGLETERRE	Easy Jet
TA	Istambul	TURQUIE	Turkish Airlines

PILOTE

Brevet	Nom	NbHVol	Compa	ChefPil
PL-1	Amine	450	AA	PL-4
PL-2	Mohamed	900	AF	PL-4
PL-3	Smail	1000	TA	PL-5
PL-4	Youcef	3400	AA	
PL-5	Oussama	2100	AZ	

AVION

Matr	TypeAv	NbHVol	Compa
A1	A320	1000	AA
A2	A330	1900	AF
A3	B737	550	AZ
A4	A340	1800	AA
A5	A320	200	TA
A6	B727	100	EJ

SQL - Langage d'Interrogation de Données

Sous-interrogation monoligne

- Brevet, Nom des pilotes de la compagnie « Air Algerie » ayant plus de 500 heures de vol

  - SELECT Brevet, Nom FROM **Pilote**
    WHERE **Compa = (SELECT Comp FROM Compagnie WHERE NomComp = 'Air Algerie'**
    AND NbHVol > 500) ;

Equi-Jointure

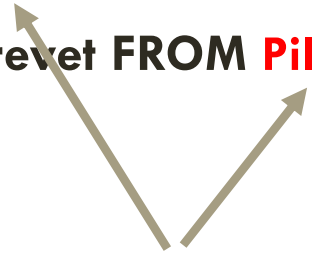
RESULTAT

Brevet	Nom
PL-4	Youcef

SQL - Langage d'Interrogation de Données

Sous-interrogation monoligne

- Brevet, Nom des pilotes placés sous la responsabilité du pilote « Youcef »
  - SELECT Brevet, Nom FROM **Pilote**
    WHERE **ChefPil = (SELECT Brevet FROM Pilote WHERE Nom = 'Youcef') ;**



Auto-Jointure

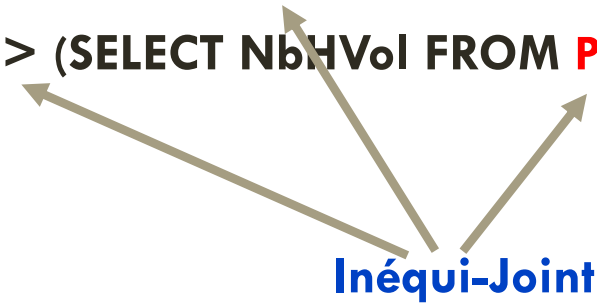
RESULTAT

Brevet	Nom
PL-1	Amine
PL-2	Mohamed

SQL - Langage d'Interrogation de Données

Sous-interrogation monoligne

- Nom, NbHVol des pilotes ayant plus d'expérience que le pilote de brevet « PL-2 »
  - SELECT Nom, NbHVol FROM **Pilote**
    WHERE **NbHVol > (SELECT NbHVol FROM Pilote WHERE Brevet = 'Pl-2') ;**



RESULTAT

Brevet	Nom
PL-1	Amine
PL-2	Mohamed

SQL - Langage d'Interrogation de Données

Sous-interrogation multilignes (IN, NOT IN)

- **IN** compare un élément à une donnée quelconque retournée par la sous requête

- **Exemple** : NomComp, Ville des compagnies qui embauchent des pilotes de moins de 1000 heures de vol

  - SELECT NomComp, Ville FROM Compagnie
    WHERE **Comp IN (SELECT Compa FROM Pilote WHERE NbHVol < 1000) ;**

RESULTAT

NomComp	Ville
Air Algerie	Alger
Air France	Paris

SQL - Langage d'Interrogation de Données

Sous-interrogation multilignes (IN, NOT IN)

- Les compagnies n'ayant pas de pilote
  - SELECT * FROM Compagnie
    WHERE **Comp NOT IN (SELECT Compa FROM Pilote WHERE Compa IS NOT NULL) ;**

RESULTAT

Comp	Ville	Pays	NomComp
EJ	Londres	ANGLETERRE	Easy Jet

SQL - Langage d'Interrogation de Données

Sous-interrogation multilignes (ANY, ALL)

- **ANY** compare l'élément à chaque donnée ramenée par la sous-requête :
  - L'opérateur « **= ANY** » est équivaut à **IN**
  - L'opérateur « **< ANY** » signifie « Inférieur à au moins une des valeurs »
  - L'opérateur « **> ANY** » signifie « Supérieur à au moins une des valeurs »
- **ALL** compare l'élément à toux ceux ramenés par la sous requête :
  - L'opérateur « **< ALL** » signifie « Inférieur au minimum »
  - L'opérateur « > **ALL** » signifie « Supérieur au maximum »

SQL - Langage d'Interrogation de Données

Sous-interrogation multilignes (ANY)

- Les avions dont le nombre d'heures de vol est inférieur à celui de **n'importe quel** avion de type « A320 »

  - SELECT * FROM Avion
    WHERE **NbHVol < ANY (SELECT NbHVol FROM Avion WHERE TypeAv = 'A320') ;**

AVION

Matr	TypeAv	NbHVol	Compa
A1	A320	1000	AA
A2	A330	1900	AF
A3	B737	550	AZ
A4	A340	1800	AA
A5	A320	200	TA
A6	B727	100	EJ

RESULTAT

Matr	TypeAv	NbHVol	Compa
A3	B737	550	AZ
A5	A320	200	TA
A6	B727	100	EJ

SQL - Langage d'Interrogation de Données

Sous-interrogation multilignes (ANY)

- Les avions dont le nombre d'heures de vol est supérieur à celui de **n'importe quel** avion de la compagnie «AA»

  - SELECT * FROM Avion

    WHERE **NbHVol > ANY (SELECT NbHVol FROM Avion WHERE Compa = 'AA') ;**

AVION

Matr	TypeAv	NbHVol	Compa
A1	A320	1000	AA
A2	A330	1900	AF
A3	B737	550	AZ
A4	A340	1800	AA
A5	A320	200	TA
A6	B727	100	EJ

RESULTAT

Matr	TypeAv	NbHVol	Compa
A2	A330	1900	AF
A4	A340	1800	AA

SQL - Langage d'Interrogation de Données

Sous-interrogation multilignes (ALL)

- Les avions dont le nombre d'heures de vol est inférieur **à tous** les avions de type « A320 »
  - SELECT * FROM Avion
    WHERE **NbHVol < ALL (SELECT NbHVol FROM Avion WHERE TypeAv = 'A320') ;**

AVION

Matr	TypeAv	NbHVol	Compa
A1	A320	1000	AA
A2	A330	1900	AF
A3	B737	550	AZ
A4	A340	1800	AA
A5	A320	200	TA
A6	B727	100	EJ

RESULTAT

Matr	TypeAv	NbHVol	Compa
A6	B727	100	EJ

SQL - Langage d'Interrogation de Données

Sous-interrogation multilignes (ALL)

- Les avions dont le nombre d'heures de vol est supérieur **à tous** les avions de la compagnie « AA »

  - SELECT * FROM Avion
    WHERE **NbHVol > ALL (SELECT NbHVol FROM Avion WHERE Compa = 'AA') ;**

AVION

Matr	TypeAv	NbHVol	Compa
A1	A320	1000	AA
A2	A330	1900	AF
A3	B737	550	AZ
A4	A340	1800	AA
A5	A320	200	TA
A6	B727	100	EJ

RESULTAT

Matr	TypeAv	NbHVol	Compa
A2	A330	1900	AF

PLAN DE LA MATIERE

- Concepts de base
- Modèle relationnel
- Algèbre relationnelle
- SQL : Définitions des schémas
- SQL : Mise à jour de données
- SQL : Interrogation simple
- SQL : Extraction de données de plusieurs tables (Jointures)
- SQL : Tri du Résultat d'une Requête
- SQL : Opérateurs ensemblistes
- SQL : Fonctions de groupe
- SQL : Sous-requêtes
- **SQL : Sous interrogations synchronisées**
- Dépendances fonctionnelles et Normalisation des relations

SQL - Langage d'Interrogation de Données

Sous-interrogation synchronisée

- Une sous-requête est **synchronisée** si elle manipule des colonnes d'une table du niveau supérieur
- Elle est exécutée une fois pour chaque ligne extraite par la requête de niveau supérieur
- Les alias sont utiles pour pouvoir manipuler les colonnes de tables de différents niveaux

```
SELECT Alias1.X  FROM Table1  Alias1
WHERE Col1  Opérateur (SELECT Alias2.Y FROM Table2 Alias2
                       WHERE Alias1.X Opérateur Alias2.Y) ;
```

- Une sous-requête synchronisée peut ramener une ou plusieurs lignes
- Différents opérateurs peuvent être employés (=, <>, <, >, <=, >=, **EXISTS, NOT EXISTS**)

SQL - Langage d'Interrogation de Données

Sous-interrogation synchronisée (EXISTS)

- **EXISTS** permet d'interrompre la sous-interrogation dés la première ligne trouvée
- La valeur FALSE est retournée si aucune ligne n'est retournée par la sous-requête
- **Exemple** : Brevet, Nom des pilotes ayant au moins un pilote sous leur responsabilité
  - SELECT P1.Brevet, P1.Nom FROM Pilote P1

    WHERE **EXISTS** **(SELECT P2.* FROM Pilote P2 WHERE P1.Brevet=P2.ChefPil) ;**

PILOTE

Brevet	Nom	NbHVol	Compa	ChefPil
PL-1	Amine	450	AA	PL-4
PL-2	Mohamed	900	AF	PL-4
PL-3	Smail	1000	TA	PL-5
PL-4	Youcef	3400	AA	
PL-5	Oussama	2100	AZ	

RESULTAT

Brevet	Nom
PL-4	Youcef
PL-5	Oussama

SQL - Langage d'Interrogation de Données

Sous-interrogation synchronisée (NOT EXISTS)

- **NOT EXISTS** retourne la valeur TRUE si aucune ligne n'est retournée par la sous-requête
- **Exemple** : Liste des compagnies n'ayant pas de pilotes
  - SELECT C.* FROM Compagnie C
    WHERE **NOT EXISTS (SELECT Compa FROM Pilote WHERE Compa = C.Comp) ;**

COMPAGNIE

Comp	Ville	Pays	NomComp
AA	Alger	ALGERIE	Air Algerie
AZ	Paris	FRANCE	Aigle Azur
AF	Paris	FRANCE	Air France
EJ	Londres	ANGLETERRE	Easy Jet
TA	Istambul	TURQUIE	Turkish Airlines

RESULTAT

Comp	Ville	Pays	NomComp
EJ	Londres	ANGLETERRE	Easy Jet

SQL - Langage d'Interrogation de Données

Jointure procédurale

- Les jointures procédurales sont écrites par des requêtes qui contiennent des sous-requêtes
- Chaque clause FROM ne contient qu'une seule table
- Cette forme d'écriture n'est pas la plus utilisée mais elle permet de mieux visualiser certaines jointures
- Elle est plus complexe à écrire, car l'ordre d'apparition des tables dans les clauses FROM a son importance
- Seules les colonnes de la table qui se trouve au niveau du premier SELECT peuvent être extraites

SQL - Langage d'Interrogation de Données

Jointure procédurale

- Noms de compagnies qui ont au moins un avion de type A320
- **Jointure classique**
  - SELECT NomComp FROM **Compagnie, Avion**
    WHERE Compagnie.Comp = Avion.Comp AND TypeAv = 'A320';

RESULTAT

NomComp
Air Algerie
Turkish Airlines

- **Jointure procédurale `**
  - SELECT NomComp FROM **Compagnie**
    WHERE **Comp = (SELECT Comp FROM Avion WHERE TypeAv = 'A320');**

Université Abou Bakr Belkaid– Tlemcen
Faculté des Sciences
Département d'Informatique

BASES DE DONNÉES

FIN

MATALLAH Houcine

L2 Informatique

2018-2019