Faculté de Sciences Économiques et de Gestion

Bases de données

Maîtrise de Sciences Économiques

Année 2001-2002

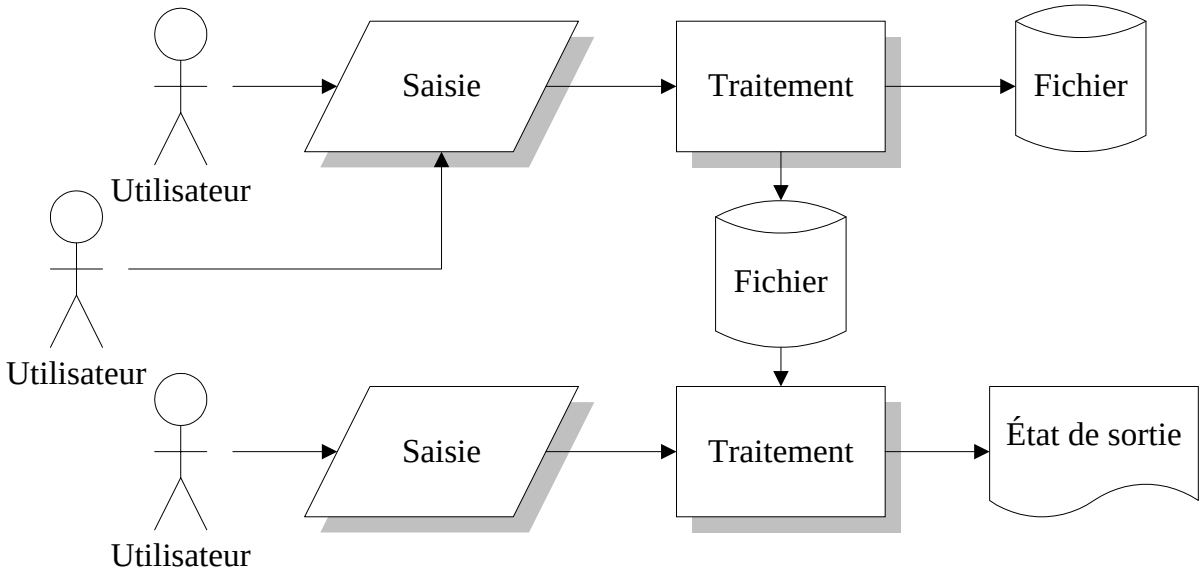
Jérôme Darmont

http://eric.univ-lyon2.fr/~jdarmont/

Plan du cours

Plan du cours

I.1. Limites des systèmes à fichiers

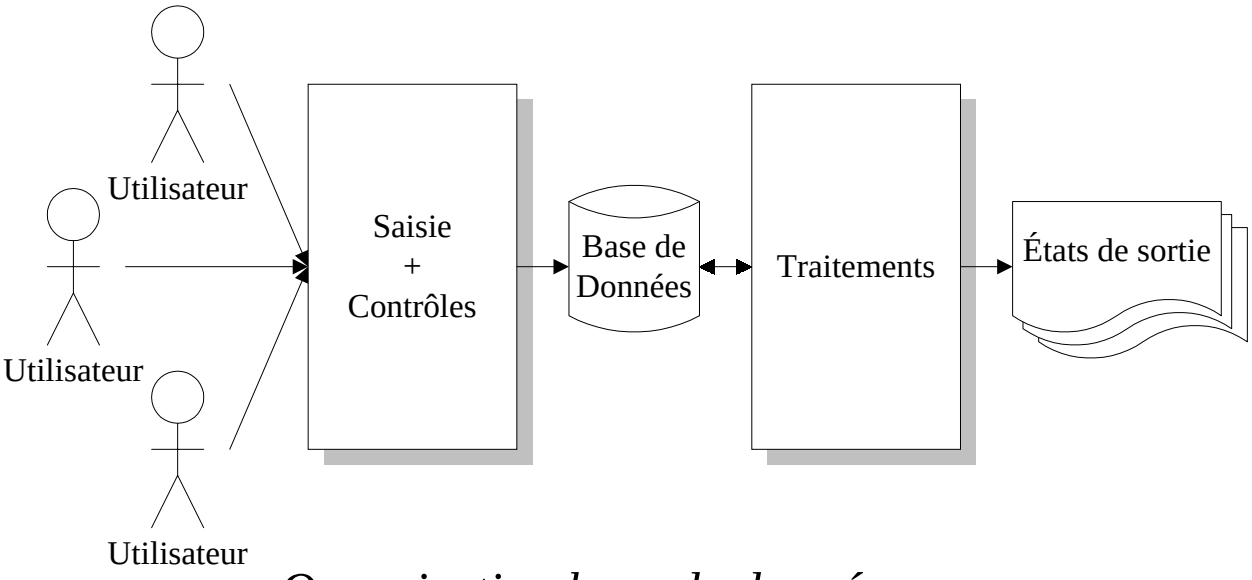


Organisation en fichiers

I.1. Limites des systèmes à fichiers

- Particularisation de la saisie et des traitements en fonction des fichiers ⇒ un ou plusieurs programmes par fichier
- Contrôle en différé des données
  ⇒ augmentation des délais et du risque d'erreur
- Particularisation des fichiers en fonction des traitements ⇒ grande redondance des données

I.2. Organisation base de données



Organisation base de données

I.2. Organisation base de données

- Uniformisation de la saisie et standardisation des traitements (ex. tous les résultats de consultation sous forme de listes et de tableaux)
- Contrôle immédiat de la validité des données
- Partage de données entre plusieurs traitements
  $\Rightarrow$ limitation de la redondance des données

I.3. Définitions

- **Base de données (BD) :** Collection de données cohérentes et structurées
- **Système de Gestion de Bases de Données (SGBD) :** Logiciel(s) assurant structuration, stockage, maintenance, mise à jour et consultation des données d'une BD

I.4. Objectifs des SGBD

- **Indépendance physique :** un remaniement de l'organisation physique des données n'entraîne pas de modification dans les programmes d'application (traitements)
- **Indépendance logique :** un remaniement de l'organisation logique des fichiers (ex. nouvelle rubrique) n'entraîne pas de modification dans les programmes d'application non concernés

I.4. Objectifs des SGBD

- **Manipulation facile des données :** un utilisateur non-informaticien doit pouvoir manipuler simplement les données (interrogation et mise à jour)
- **Administration facile des données :** un SGBD doit fournir des outils pour décrire les données, permettre le suivi de ces structures et autoriser leur évolution (tâche de l'*administrateur de BD*)

I.4. Objectifs des SGBD

- **Efficacité des accès aux données :** garantie d'un bon *débit* (nombre de transactions exécutées par seconde) et d'un bon *temps de réponse* (temps d'attente moyen pour une transaction)
- **Redondance contrôlée des données :** diminution du volume de stockage, pas de mise à jour multiple ni d'incohérence

I.4. Objectifs des SGBD

- **Cohérence des données :** ex. L'âge d'une personne doit être un nombre entier positif. Le SGBD doit veiller à ce que les applications respectent cette règle (*contrainte d'intégrité*).
- **Partage des données :** utilisation simultanée des données par différentes applications
- **Sécurité des données :** les données doivent être protégées contre les accès non-autorisés ou en cas de panne

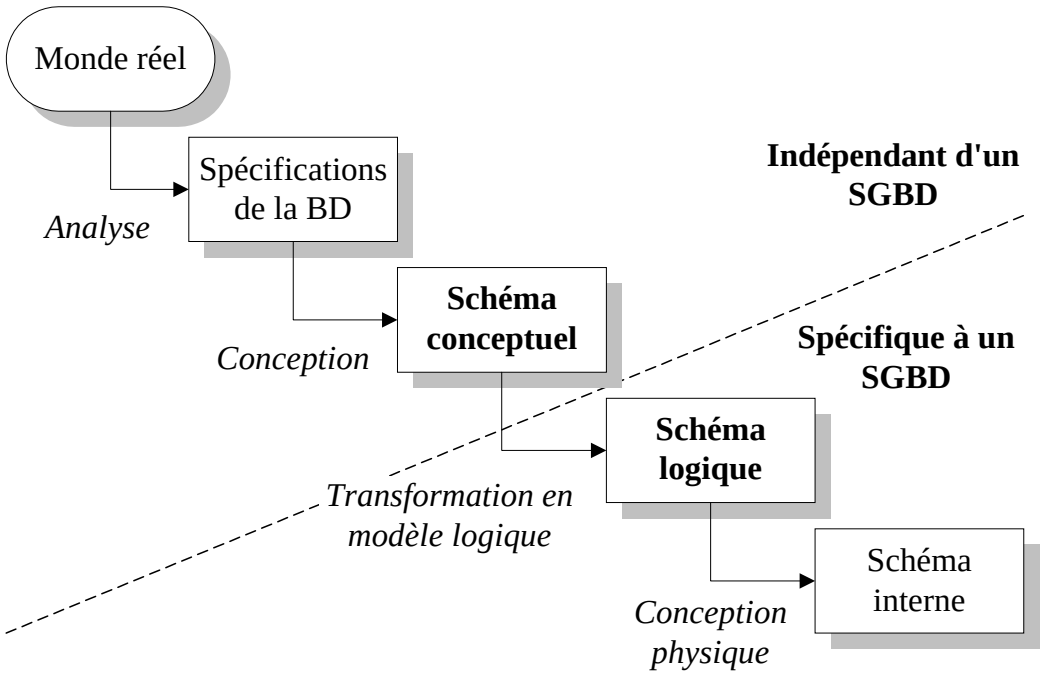
I.5. Fonctions des SGBD

- Description des données : *Langage de Définition de Données* (LDD)
- Recherche des données
- Mise à jour des données
- Transformation des données

} *Langage de Manipulation de Données* (LMD)

- Contrôle de l'intégrité des données
- Gestion de transactions et sécurité

I.6. Processus de conception d'une base de données



Plan du cours

I. Introduction

☞ **II. Le modèle E/A**

III. Le modèle relationnel

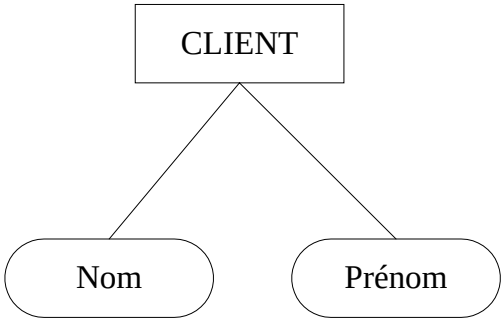
IV. Le langage SQL

II.1. Généralités

- E/A signifie *Entité-Association*, traduction de E/R (*Entity-Relationship*).
- Le modèle E/A est un *modèle conceptuel* conçu dans les années 1970.
- Il utilise une *représentation graphique*.

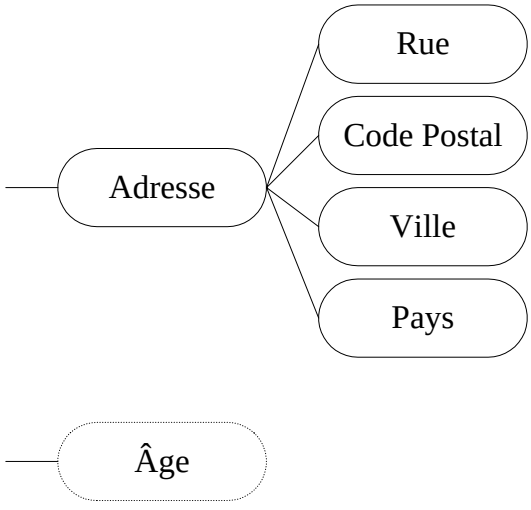
II.2. Entités et attributs

- **Entité :** objet de l'univers du discours (ex. CLIENT)
- **Attribut :** propriété de l'entité (ex. Nom et Prénom du client)
  - Attribut *simple* : non divisible (ex. Nom)



II.2. Entités et attributs

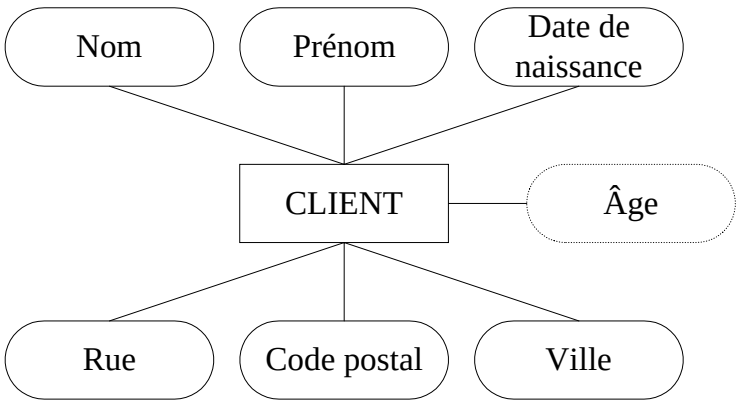
- Attribut *composé* : subdivisé en attributs simples (ex. Adresse)
- Attribut *dérivé* : dont la valeur est calculée (ex. Âge d'après la date de naissance)



II.2. Entités et attributs

- Valeur **nulle** (NULL) pour un attribut : la valeur n'est pas connue
  ex. Un client dont on ne connaît pas la date de naissance.
- **Domaine :** ensemble des valeurs que peut prendre un attribut
  ex. Prix des produit de 1 à 10.000 F

II.2. Entités et attributs



Exemple d'entité avec ses attributs

II.3. Types et occurrences

- **Type d'entité :** ex. CLIENT
- **Occurrences du type** CLIENT : les clients

  Albert Dupont
  James West
  Marie Martin
  Gaston Durand
  ...

II.3. Types et occurrences

- **Type d'attribut**
  - Nom et Prénom sont des *chaînes de caractères*
  - Date de naissance est une *date*
  - Âge est un *nombre entier*
- **Occurrences d'attribut :** valeur particulière d'un attribut
  ex. Bleu, Rouge sont des occurrences d'un attribut Couleur.

II.4. Identifiants

Liste des clients

Nom	*Prénom*	*Date de Naissance*	*Etc.*
Dupont	Albert	01/06/70	...
West	James	03/09/63	...
Martin	Marie	05/06/78	...
Durand	Gaston	15/11/80	...
Titgoutte	Justine	28/02/75	...
Dupont	Noémie	18/09/57	...
Dupont	Albert	23/05/33	...

Problème : Comment distinguer les Dupont ?

II.4. Identifiants

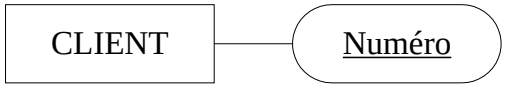
Solution : Ajouter un attribut *Numéro de client*

Numéro	*Nom*	*Prénom*	*Date de Naissance*
1	Dupont	Albert	01/06/70
2	West	James	03/09/63
3	Martin	Marie	05/06/78
4	Durand	Gaston	05/11/80
5	Titgoutte	Justine	28/02/75
6	Dupont	Noémie	18/09/57
7	Dupont	Albert	23/05/33

II.4. Identifiants

- Le numéro de client est un **identifiant**. Un identifiant caractérise *de façon unique* les occurrences d'un type d'entité.

- Notation graphique :



II.4. Associations et cardinalités

Association : liaison perçue entre des entités

ex. Les clients commandent des produits.

Les entités CLIENT et PRODUIT sont dites *participantes* à la relation COMMANDE.

II.4. Associations et cardinalités

Association 1-1

ex. Un client donné ne commande qu'un seul produit. Un produit donné n'est commandé que par un seul client.

X,Y - X : *cardinalité* mini - Y : *cardinalité* maxi

Lire : Un client commande X à Y produits.

II.4. Associations et cardinalités

Association 1-N

ex. Un client donné commande plusieurs produits. Un produit donné n'est commandé que par un seul client.

La cardinalité « un à plusieurs » (1-N) peut aussi être « zéro à plusieurs » (0-N).

II.4. Associations et cardinalités

Association 0 ou 1-N

ex. Un client donné commande plusieurs produits. Un produit donné est commandé au maximum par un client, mais peut ne pas être commandé.

La cardinalité « un à plusieurs » (1-N) peut aussi être « zéro à plusieurs » (0-N).

II.4. Associations et cardinalités

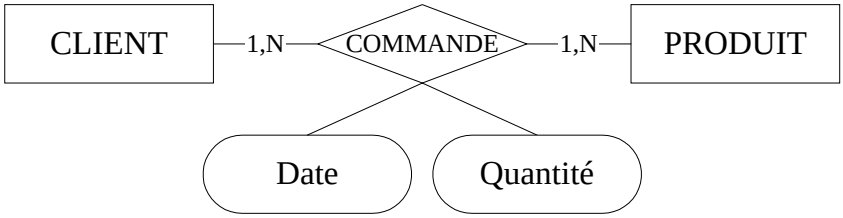
Association M-N

ex. Un client donné commande plusieurs produits. Un produit donné est commandé par plusieurs clients.

Les cardinalités « un à plusieurs » (1-N) peuvent aussi être « zéro à plusieurs » (0-N).

II.4. Associations et cardinalités

Attribut d'association : Dans une *association M-N*, il est possible de caractériser l'association par des attributs.
ex. Une commande est passée à une Date donnée et concerne une Quantité de produit fixée.



II.5. Exemple VPC complet

Spécifications

- Les clients sont caractérisés par un numéro de client, leur nom, prénom, date de naissance, rue, code postal et ville.
- Ils commandent des produits à une date donnée et dans une quantité donnée.

II.5. Exemple VPC complet

Spécifications (suite)

- Les produits sont caractérisés par un numéro de produit, leur désignation et leur prix unitaire.
- Chaque produit est fourni par un fournisseur unique (mais un fournisseur peut fournir plusieurs produits).
- Les fournisseurs sont caractérisés par un numéro de fournisseur et leur raison sociale.

II.5. Exemple VPC complet

Marche à suivre pour produire un schéma E/A

1) Identifier les entités.

2) Identifier les associations entre les entités.

3) Identifier les attributs de chaque entité et de chaque association.

4) Evaluer les cardinalités des associations.

II.5. Exemple VPC complet

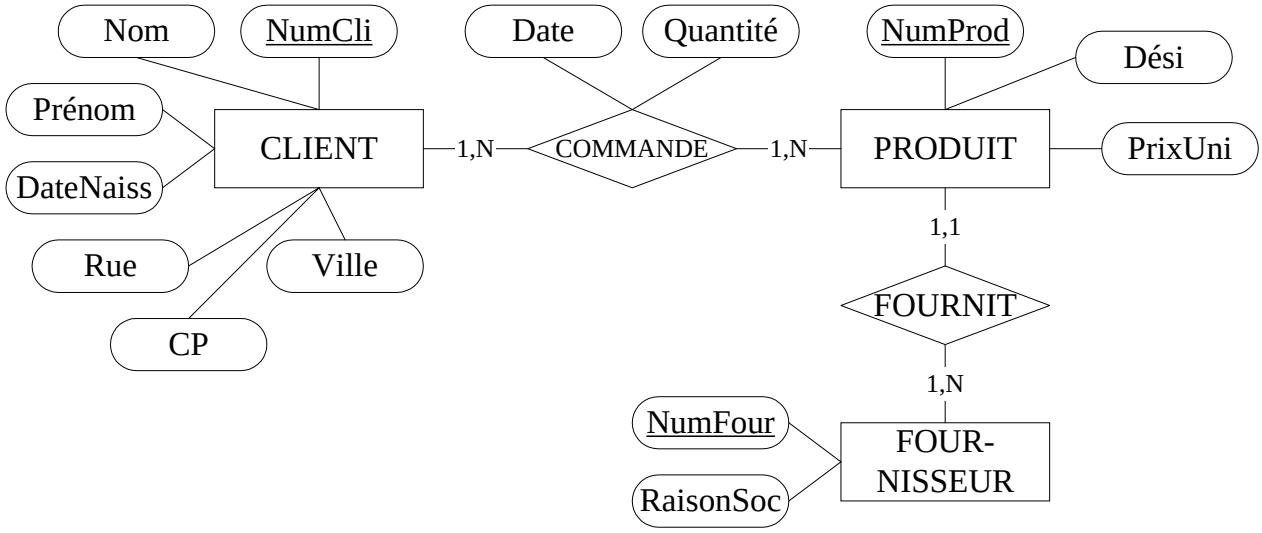


Schéma E/A de l'exemple VPC

Plan du cours

III.1. Généralités

- Le modèle relationnel est un *modèle logique* associé aux SGBD relationnels (ex. Oracle, DB2, SQLServer, Access, Paradox, dBase…).
- **Objectifs du modèle relationnel :**
  - indépendance physique
  - traitement du problème de redondance des données
  - LMD non procéduraux (faciles à utiliser)
  - devenir un standard

III.2. Relations, attributs et tuples

- Une *relation* R est un ensemble d'*attributs* $\{A_1, A_2, \ldots, A_n\}$.
  ex. La relation PRODUIT est l'ensemble des attributs {NumProd, Dési, PrixUni}
- Chaque attribut A_i prend ses valeurs dans un *domaine* $dom(A_i)$.
  ex. PrixUni $\in$]0, 10.000]

III.2. Relations, attributs et tuples

- Un *tuple* t est un ensemble de valeurs
  $t = <V_1, V_2, \ldots, V_n>$ où $V_i \in dom(A_i)$ ou V_i est la valeur nulle (NULL).
  ex. <112, 'Raquette de tennis', 300> est un tuple de la relation PRODUIT.
- **Notation :** $R (A_1, A_2, \ldots, A_n)$
  ex. PRODUIT (NumProd, Dési, PrixUni)

III.3. Contraintes d'intégrité

- **Clé primaire :** Ensemble d'attributs dont les valeurs permettent de distinguer les tuples les uns des autres (*identifiant*).
  ex. NumProd clé primaire de la relation PRODUIT
- **Clé étrangère :** Attribut qui est clé primaire d'une autre relation.
  ex. Connaître le fournisseur de chaque produit ⇒ ajout de l'attribut NumFour à la relation PRODUIT

III.3. Contraintes d'intégrité

Notations : clés primaires soulignées, clés étrangères *en italiques*
ex. PRODUIT (NumProd, Dési, PrixUni, *NumFour*)

- **Contraintes de domaine :** les attributs doivent respecter une condition logique
  ex. PrixUni > 0 ET PrixUni ≤ 10000

III.4. Traduction E/A-Relationnel

1) Chaque entité devient une relation. Les attributs de l'entité deviennent attributs de la relation. Seuls les attributs simples des attributs composés sont inclus. L'identifiant de l'entité devient clé primaire de la relation.

ex. CLIENT (NumCli, Nom, Prénom, DateNaiss, Rue, CP, Ville)

III.4. Traduction E/A-Relationnel

2) Chaque association 1-1 est prise en compte en incluant la clé primaire d'une des relations comme clé étrangère dans l'autre relation.

ex. Si un client peut posséder un compte, on aura :

COMPTE (NumCom, Solde)
CLIENT (NumCli, Nom, Prénom, DateNaiss, Rue, CP, Ville, *NumCom*)

III.4. Traduction E/A-Relationnel

3) Chaque association 1-N est prise en compte en incluant la clé primaire de la relation dont la cardinalité maximale est N comme clé étrangère dans l'autre relation.

ex.

PRODUIT (NumProd, Dési, PrixUni, *NumFour*)
FOURNISSEUR (NumFour, RaisonSoc)

III.4. Traduction E/A-Relationnel

4) Chaque association M-N est prise en compte en créant une nouvelle relation dont la clé primaire et la concaténation des clés primaires des relations participantes. Les attributs de l'association sont insérés dans cette nouvelle relation.

ex. COMMANDE (*NumCli*, *NumProd*, Date, Quantité)

III.4. Traduction E/A-Relationnel

Schéma relationnel complet de l'exemple VPC

CLIENT (NumCli, Nom, Prénom, DateNaiss, Rue, CP, Ville)
PRODUIT (NumProd, Dési, PrixUni, *NumFour*)
FOURNISSEUR (NumFour, RaisonSoc)

COMMANDE (*NumCli, NumProd*, Date, Quantité)

III.5. Problème de la redondance

[En dehors des clés étrangères]

ex. Soit la relation COMMANDE_PRODUIT.

NumProd	*Quantité*	*NumFour*	*Adresse*
101	300	901	Quai des brumes
104	1000	902	Quai Claude Bernard
112	78	904	Quai des Marans
103	250	901	Quai des brumes

Cette relation présente différentes anomalies.

III.5. Problème de la redondance

- **Anomalies de modification :** Si l'on souhaite mettre à jour l'adresse d'un fournisseur, il faut le faire pour tous les tuples concernés.
- **Anomalies d'insertion :** Pour ajouter un fournisseur nouveau, il faut obligatoirement fournir des valeurs pour *NumProd* et *Quantité*.
- **Anomalies de suppression :** ex. La suppression du produit 104 fait perdre toutes les informations concernant le fournisseur 902.

III.6. Normalisation

Objectifs de la normalisation :

- Suppression des problèmes de mise à jour
- Minimisation de l'espace de stockage (élimination des redondances)

III.6. Normalisation

Les dépendances fonctionnelles (DF)

Soit R (X, Y, Z) une relation où X, Y, et Z sont des ensembles d'attributs. Z peut être vide.

Définition : *Y dépend fonctionnellement de X* ($X \rightarrow Y$) si c'est toujours la même valeur de Y qui est associée à X dans la relation R.

ex. PRODUIT (NumProd, Dési, PrixUni)
NumProd $\rightarrow$ Dési, Dési $\rightarrow$ PrixUni

III.6. Normalisation

Propriétés des dépendances fonctionnelles

- *Réflexivité :* Si $Y \subseteq X$ alors $X \rightarrow Y$.
- *Augmentation :* Si $W \subseteq Z$ et $X \rightarrow Y$ alors $X, Z \rightarrow Y, W$.
- *Transitivité :* Si $X \rightarrow Y$ et $Y \rightarrow Z$ alors $X \rightarrow Z$.

(Règles d'inférence d'Armstrong)

III.6. Normalisation

Propriétés des dépendances fonctionnelles

- *Pseudo-transitivité :* Si $X \rightarrow Y$ et $Y, Z \rightarrow T$ alors $X, Z \rightarrow T$.
- *Union :* Si $X \rightarrow Y$ et $X \rightarrow Z$ alors $X \rightarrow Y, Z$.
- *Décomposition :* Si $Z \subseteq Y$ et $X \rightarrow Y$ alors $X \rightarrow Z$.

NB : La notation X, Y signifie $X \cup Y$.

III.6. Normalisation

Première forme normale (1FN)

Une relation est en 1FN si tout attribut n'est pas décomposable.

ex. Les relations PERSONNE (Nom, Prénoms, Age) et DEPARTEMENT (Nom, Adresse, Tel) ne sont pas en 1FN si les attributs *Prénoms* et *Adresse* peuvent être du type [Jean, Paul] ou [Rue de Marseille, Lyon].

III.6. Normalisation

Deuxième forme normale (2FN)

Une relation est en 2FN si :
- elle est en 1FN ;
- tout attribut non clé primaire est dépendant de la clé primaire entière.

ex. La relation CLIENT (NumCli, Nom, Prénom, DateNaiss, Rue, CP, Ville) est en 2FN.

III.6. Normalisation

Deuxième forme normale (2FN)

ex. La relation COMMANDE_PRODUIT (NumProd, Quantité, NumFour, Ville) n'est pas en 2FN car on a NumProd, NumFour → Quantité et NumFour → Ville.

La décomposition suivante donne deux relations en 2FN : COMMANDE (NumProd, NumFour, Quantité) ; FOURNISSEUR (NumFour, Ville).

III.6. Normalisation

Troisième forme normale (3FN)

Une relation est en 3FN si :
- *elle est en 2FN ;*
- *il n'existe aucune DF entre deux attributs non clé primaire.*

ex. La relation COMPAGNIE (Vol, Avion, Pilote) avec les DF Vol → Avion, Avion → Pilote et Vol → Pilote est en 2FN, mais pas en 3FN.

III.6. Normalisation

Troisième forme normale (3FN)

Anomalies de mise à jour sur la relation COMPAGNIE : il n'est pas possible d'introduire un nouvel avion sur un nouveau vol sans préciser le pilote correspondant.

La décomposition suivante donne deux relations en 3FN qui permettent de retrouver (par transitivité) toutes les DF : R1 (Vol, Avion) ;
R2 (Avion, Pilote).

III.7. Algèbre relationnelle

- Ensemble d'opérateurs qui s'appliquent aux relations
- Résultat : nouvelle relation qui peut à son tour être manipulée

⇒ L'algèbre relationnelle permet d'effectuer des recherches dans les relations.

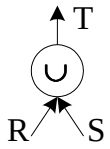
III.7. Algèbre relationnelle

Opérateurs ensemblistes

- *Union :* T = R ∪ S ou T = UNION (R, S)
  R et S doivent avoir même schéma.

  ex. R et S sont les relations PRODUIT de deux sociétés qui fusionnent et veulent unifier leur catalogue.

  Notation graphique :



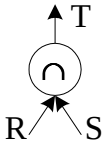
III.7. Algèbre relationnelle

Opérateurs ensemblistes

- *Intersection :* T = R ∩ S ou
  T = INTERSECT (R, S)
  R et S doivent avoir même schéma.

  ex. Permet de trouver les produits communs aux catalogues de deux sociétés.

  Notation graphique :



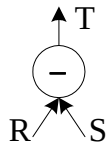
III.7. Algèbre relationnelle

Opérateurs ensemblistes

- *Différence :* T = R - S ou T = MINUS (R, S)
  R et S doivent avoir même schéma.

  ex. Permet de retirer les produits de la relation S existant dans la relation R.

  Notation graphique :



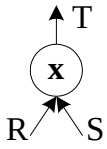
III.7. Algèbre relationnelle

Opérateurs ensemblistes

- *Produit cartésien :* T = R x S
  ou T = PRODUCT (R, S)

  Associe chaque tuple de R à chaque tuple de S.

  Notation graphique :



III.7. Algèbre relationnelle

ex.

NumProd	*Dési*
0	P1
1	P2

X

NumFour	*RaisonSoc*
10	F1
20	F2
30	F3

=

NumProd	*Dési*	*NumFour*	*RaisonSoc*
0	P1	10	F1
1	P2	10	F1
0	P1	20	F2
1	P2	20	F2
0	P1	30	F3
1	P2	30	F3

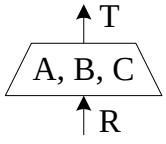
III.7. Algèbre relationnelle

Opérateurs spécifiques

- *Projection :* T = Π <A, B, C> (R)
  ou T = PROJECT (R / A, B, C)

  T ne contient que les attributs A, B et C de R.
  ex. Noms et prénoms des clients.

  Notation graphique :



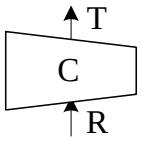
III.7. Algèbre relationnelle

Opérateurs spécifiques

- *Restriction :* T = σ <C> (R)
  ou T = RESTRICT (R / C)

  T ne contient que les attributs de R qui satisfont la condition C.
  ex. C = Clients qui habitent Lyon.

  Notation graphique :



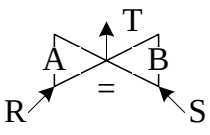
III.7. Algèbre relationnelle

Opérateurs spécifiques

- *Jointure naturelle :* T = R >< S
  ou T = JOIN (R, S)

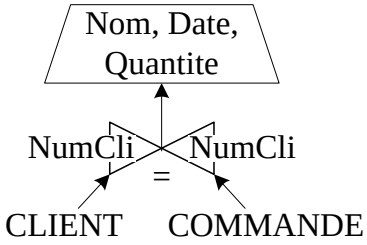
  Produit cartésien R x S et restriction A = B sur les attributs A ∈ R et B ∈ S.

  Notation graphique :



III.7. Algèbre relationnelle

ex. Commandes avec le nom du client et pas seulement son numéro



NB : *Requête* = enchaînement d'opérations

III.7. Algèbre relationnelle

Décomposition des opérations

CL

NumCli	*Nom*
1	C1
2	C2
3	C3

X

CO

NumCli	*Date*	*Quantité*
1	22/09/99	1
3	22/09/99	5
3	22/09/99	2

III.7. Algèbre relationnelle

=

CL.NumCli	*Nom*	*CO.NumCli*	*Date*	*Quantité*
1	C1	1	22/09/99	1
2	C2	1	22/09/99	1
3	C3	1	22/09/99	1
1	C1	3	22/09/99	5
2	C2	3	22/09/99	5
3	C3	3	22/09/99	5
1	C1	3	22/09/99	2
2	C2	3	22/09/99	2
3	C3	3	22/09/99	2

III.7. Algèbre relationnelle

CL >< CO

CL.NumCli	*Nom*	*CO.NumCli*	*Date*	*Quantité*
1	C1	1	22/09/99	1
3	C3	3	22/09/99	5
3	C3	3	22/09/99	2

Nom	*Date*	*Quantité*
C1	22/09/99	1
C3	22/09/99	5
C3	22/09/99	2

Π <Nom, Date, Quantité> (CL >< CO)

(Projection sur les attributs Nom, Date, Quantité)

Plan du cours

I. Introduction

II. Le modèle E/A

III. Le modèle relationnel

☞ **IV. Le langage SQL**

IV.1. Généralités

- SQL = *Structured Query Language*
- SQL permet la définition, la manipulation et le contrôle d'une base de données relationnelle. Il se base sur l'algèbre relationnelle.
- SQL est un standard depuis 1986.
- Nous adoptons dans ce chapitre la syntaxe de SQL*Plus (Oracle).

IV.2. Définition des données

ex. CREATE TABLE CLIENT (NumCli NUMBER(3),
Nom CHAR(30),
DateNaiss DATE,
Salaire NUMBER(8,2),
NumEmp NUMBER(3),

CONSTRAINT cle_pri PRIMARY KEY (NumCli),
CONSTRAINT cle_etr FOREIGN KEY (NumEmp)
REFERENCES EMPLOYEUR(NumEmp),
CONSTRAINT date_ok CHECK (DateNaiss < SYSDATE));

IV.2. Définition des données

Types de données

- NUMBER(n) : nombre entier à n chiffres
- NUMBER(n, m) : nombre réel à n chiffres au total (virgule comprise) et m chiffres après la virgule
- CHAR(n) : chaîne de caractères de taille n
- DATE : date au format ‘JJ-MM-AAAA’

IV.2. Définition des données

Contraintes d'intégrité

- Mot clé CONSTRAINT
- Identification par un nom de contrainte
- Clé primaire ⇒ PRIMARY KEY (clé)
- Clé étrangère ⇒ (clé) REFERENCES table(attribut)
- Contrainte de domaine ⇒ CHECK (condition)

IV.3. Mise à jour des données

- **Ajout d'un tuple**
  ex. INSERT INTO PRODUIT
  VALUES (400, 'Nouveau produit', 78.90);
- **Mise à jour d'un attribut**
  ex. UPDATE CLIENT SET Nom='Dudule'
  WHERE NumCli = 3;
- **Suppression de tuples**
  ex. DELETE FROM CLIENT WHERE Ville = 'Lyon';

IV.4. Interrogation des données

- **Tous les tuples d'une table**
  ex. SELECT * FROM CLIENT;
- **Tri du résultat**
  ex. *Par ordre alphabétique [inverse] de nom*
  SELECT * FROM CLIENT
  ORDER BY Nom [DESC];
- **Calculs** ex. *Calcul de prix TTC*
  SELECT PrixUni+PrixUni*0.206 FROM PRODUIT;

IV.4. Interrogation des données

- **Projection**
  ex. *Noms et Prénoms des clients, uniquement*
  SELECT Nom, Prenom FROM CLIENT;
- **Restriction**
  ex. *Clients qui habitent à Lyon*
  SELECT * FROM CLIENT
  WHERE Ville = 'Lyon';

IV.4. Interrogation des données

ex. *Commandes en quantité au moins égale à 3*

```
SELECT * FROM COMMANDE
WHERE Quantite >= 3;
```

ex. *Produits dont le prix est compris entre 50 et 100 F*

```
SELECT * FROM PRODUIT
WHERE PrixUni BETWEEN 50 AND 100;
```

ex. *Commandes en quantité indéterminée*

```
SELECT * FROM Commande
WHERE Quantite IS NULL;
```


IV.4. Interrogation des données

ex. *Clients habitant une ville dont le nom se termine par* sur-Saône

```
SELECT * FROM CLIENT
WHERE Ville LIKE '%sur-Saône';
```

'sur-Saône%' ⇒ commence par *sur-Saône*
'%sur%' ⇒ contient le mot *sur*

IV.4. Interrogation des données

ex. *Prénoms des clients dont le nom est Dupont, Durand ou Martin*

SELECT Prenom FROM CLIENT
WHERE Nom IN (‘Dupont’, ‘Durand’, ’Martin’);

NB : Possibilité d’utiliser la négation pour tous ces prédicats : NOT BETWEEN, NOT NULL, NOT LIKE, NOT IN.

IV.4. Interrogation des données

- **Fonctions d’agrégat**
  Elles opèrent sur un ensemble de valeurs.
  - AVG() : moyenne des valeurs
  - SUM() : somme des valeurs
  - MIN(), MAX() : valeur minimum, valeur maximum
  - COUNT() : nombre de valeurs

ex. *Moyenne des prix des produits*

SELECT AVG(PrixUni) FROM PRODUIT;

IV.4. Interrogation des données

Opérateur DISTINCT

ex. *Nombre total de commandes*

```
SELECT COUNT(*) FROM COMMANDE;
SELECT COUNT(NumCli) FROM COMMANDE;
```

ex. *Nombre de clients ayant passé commande*

```
SELECT COUNT( DISTINCT NumCli)
FROM COMMANDE;
```


IV.4. Interrogation des données

Table COMMANDE (simplifiée)

NumCli	*Date*	*Quantite*
1	22/09/99	1
3	22/09/99	5
3	22/09/99	2

COUNT(NumCli) ⇒ Résultat = 3
COUNT(DISTINCT NumCli) ⇒ Résultat = 2

IV.4. Interrogation des données

- **Jointure**

ex. *Liste des commandes avec le nom des clients*

```
SELECT Nom, Date, Quantite
FROM CLIENT, COMMANDE
WHERE   CLIENT.NumCli =
        COMMANDE.NumCli;
```


IV.4. Interrogation des données

ex. *Idem avec le numéro de client en plus*

```
SELECT C1.NumCli, Nom, Date, Quantite
FROM CLIENT C1, COMMANDE C2
WHERE C1.NumCli = C2.NumCli
ORDER BY Nom;
```

NB : Utilisation d'*alias* (C1 et C2) pour alléger l'écriture + tri par nom.

IV.4. Interrogation des données

Jointure exprimée avec le prédicat IN

ex. *Nom des clients qui ont commandé le 23/09*

```
SELECT Nom FROM CLIENT
WHERE NumCli IN (
    SELECT NumCli FROM COMMANDE
    WHERE Date = '23-09-1999'     );
```

NB : Il est possible d'imbriquer des requêtes.

IV.4. Interrogation des données

- **Prédicats EXISTS / NOT EXISTS**

ex. *Clients qui ont passé au moins une commande [n'ont passé aucune commande]*

```
SELECT * FROM CLIENT C1
WHERE [NOT] EXISTS (
    SELECT * FROM COMMANDE C2
    WHERE C1.NumCli = C2.NumCli     );
```

IV.4. Interrogation des données

- **Prédicats ALL / ANY**

  ex. *Numéros des clients qui ont commandé au moins un produit en quantité supérieure à chacune [à au moins une] des quantités commandées par le client n° 1.*

```
SELECT DISTINCT NumCli FROM COMMANDE
WHERE QUANTITE > ALL [ANY] (
    SELECT QUANTITE FROM COMMANDE
    WHERE NumCli = 1      );
```


IV.4. Interrogation des données

- **Groupement**

  ex. *Quantité totale commandée par chaque client*

```
SELECT NumCli, SUM(Quantite)
FROM COMMANDE
GROUP BY NumCli;
```

  ex. *Nombre de produits différents commandés...*

```
SELECT NumCli, COUNT(DISTINCT NumProd)
FROM COMMANDE
GROUP BY NumCli;
```

IV.4. Interrogation des données

ex. *Quantité moyenne commandée pour les produits faisant l'objet de plus de 3 commandes*

```
SELECT NumProd, AVG(Quantite)
FROM Commande
GROUP BY NumProd
HAVING COUNT(*)>3;
```

Attention : La clause HAVING ne s'utilise qu'avec GROUP BY.