Bases de données relationnelles et SGBDR

Prof Edouard Ngor SARR
Enseignant-chercheur en Informatique
Université Assane SECK de Ziguinchor (UASZ)
Email : Edouard-ngor.sarr@univ-zig.sn
Site : www.edouardsarr.com
Oct 2025

Introduction

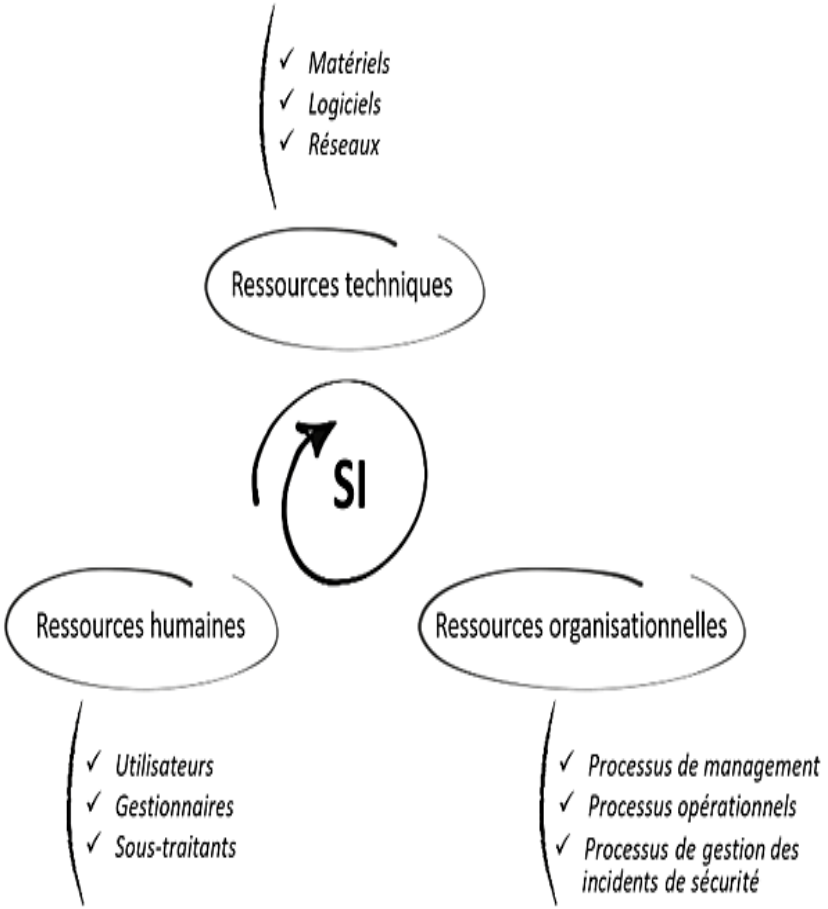
Données Vs Informations

- **UNE DONNEE (DATA)**
  - Un enregistrement dans un code bien déterminé issu de :
    - Observation
    - Objet de nature
    - Phénomène
    - Image, Vidéo ou Son
    - Micro-Texte ou Texte
  - Une matière première / un ingrédient
  - Pas sure d'en tiré de la valeur (Utilité)

- **UNE INFORMATION**
  - Provient de l'analyse et/ou de l'interprétation de (s) d'une (des) donnée (s)
  - Sa pertinence est relative et dépend :
    - Destinataire (Utilisateur)
    - Du Contexte : La situation
    - Du Moment : la Temporalité
  - Génère ou contribue à l'acquisition d'une connaissances
    - Les clients les plus réguliers
    - Les produits les plus vendus

Système d'information informatisé (SII)

- **Ensemble de Ressources**
  - Données & Information
  - Humaines
  - Matérielles
  - Logicielles & Technologique
  - …
  - **Utilisées pour :**
    - **Collecter** des données en interne ou en exterene
    - **Stocker** des données dans des BD
    - **Traiter / Analyser / Transformer** des données pour avoir de l'information
    - Diffuse de l'information dans via les formats et les canaux adequats
- Un SI c'est deux composant:
  - Les données : La matière première
  - Les traitement : Opération sur les données



`SII: l'outil informatique est utilisé pour les procès du SI.

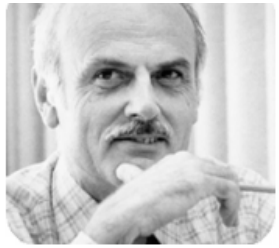
Bases de données

- **Au sens large**
  - Collection de données brutes
  - Ensemble de données
  - Grande quantité de données stockées
- **En informatique :**
  - **Ensemble de données**
    - **Numériques :** Stocker dans des supports électroniques
    - **S**tructurées et bien Organisées (Souvent en lignes et en colonnes)
    - Toujours gérées par un logiciel : Le système de gestion de base de données (SGBD).
  - **Entité cohérente**
    - **V**éhiculant une certaine sémantique (Un sens)
    - Données de la base sont inter-reliées (liaison entre les données de la bases de données)
    - Porte le plus souvent sur un unique domaine
      - Exemple : BD Comptablité, BD Finance, BD RH, BD Orientation etc
- **Attention : Faut faire une différence entre Base de données et SGBD**

Les bases de données relationnelles

Modèle relationnel : Historique

L'histoire du modèle relationnel



- 1970 « A relational Model of Data for Large Shared Data Banks »
- 1972 « Relational Completeness of Data Base Sublanguages » (Codd, IBM)

Edgar (Ted) Frank Codd

- **02 Propositions**
  - **Modèle relationnel :** Organisation des données permettant de minimiser la redondance, et maximiser la cohérence.
  - **Algèbre relationnel** (Langages de requêtes) qui sera utiliser dans la pratique grâce au SQL pour de définir, manipulerait et controler les données et les objets

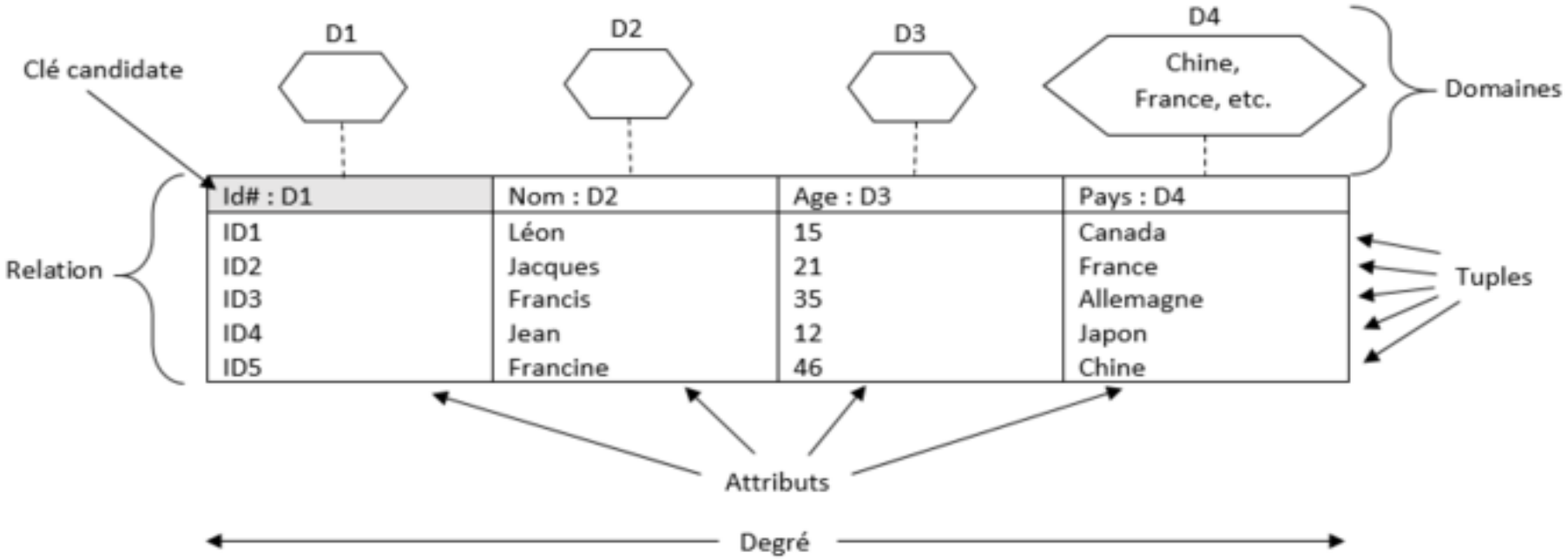
- **La base de données relationnelle**
  - Le plus répandu aujourd'hui.
  - Basé sur l'algèbre relationnel
  - Devenu la « norme »
  - BD simples à gérer et à faire évoluer, indépendamment de leur support.
  - Les données sont organisées en relations (tables)
  - Beaucoup plus pratiques en termes d'accès.

Modèle relationnel : Concepts

- **Les concepts de base**
  - **La relation :** Un structure à deux dimensions de N ligne et M colonnes
  - **Le domaine** : Un ensemble de valeurs définies que peut prendre un attribut.
    - **Simple** : es éléments sont atomiques ou non décomposables. Ex: Entier, réel, Chaine de caractères
    - **Composé** : si les éléments peuvent être décomposés. Ex : les dates sont décomposées d'un jour, un mois et une année.
  - **L'attribut :**
    - Une informations élémentaires de la relation appartenant toutes à un seul domaine
    - Il contient un ensemble des valeurs de son domaine.
    - Un attribut est simples et **mono-value**.
  - **t-uples**: Un membre, un enregistrement ou une ligne de la relation
  - **Schéma des relations :** Définition de l'Ensemble des table
  - **Schéma des attributs :** Définition de l'Ensemble des attributs
  - **Population** : Ensemble des t-uples de la relation (Table)

Modèle relationnel : Concepts

- **Les concepts de base**
  - **t-uples**: Un membre, un enregistrement ou une ligne de la relation
  - **Schéma des relations :** Définition de l'Ensemble des table
  - **Schéma des attributs :** Définition de l'Ensemble des attributs
  - **Population** : Ensemble des t-uples de la relation (Table)



Id# : D1	Nom : D2	Age : D3	Pays : D4
ID1	Léon	15	Canada
ID2	Jacques	21	France
ID3	Francis	35	Allemagne
ID4	Jean	12	Japon
ID5	Francine	46	Chine

Modèle relationnel : Contraintes

- C'est une condition à respecter lors des insertion, mises à jour ou suppression.
- **Les types de contraintes**
  - **Les identifiants**
    - UNICITE / **UNIQUE** : Pas de possibilité de doublons / Clé Candidate
    - IDENTIFIANT PRIMAIRE / **PRIMARY KEY** : Clé primaire de la table
    - IDENTIFIANT EXTERNE / **FOREIGH KEY** : Champ provenant d'une autre table où il était identifiant primaire
  - **Autres**
    - OBLIGATION / **NOT NULL** : Champs nécessitant une valeurs
    - VALEUR PAR DEFAUT / **DEFAULT** : Champ avec une valeur par défaut
    - VERIFICATION / **CHECK** : Champ qui doit respecte une condition
- Types de contraintes :
  - Contrainte de niveau Table : Concernant la table dans son ensemble.
    - PRIMARY KEY, FOREIGN KEY
  - Contrainte de niveau colonne : Concerne seulement une colonne

Modèle relationnel : Contraintes

- **UNIQUE**
  - Identifiant ou clé Candidate
  - Une colonne marquée à UNIQUE ne peut avoir de doublons (Redondance) c'est-à-dire deux valeurs identiques
  - Une relation peut avoir plusieurs identifiant UNIQUE
    - Exemple : TELEPHONE, Email, NIN, NINEA, INE peuvent être UNIQUE lors de l'inscription de l'étudiant
- **NULL & NOT NULL**
  - NOT NULL : Un champ Obligatoire
    - Une colonne marquée à NOT NULL doit forcer être renseigné lors d'une insertion c'est-à-dire qu'elle ne peut être NULL
      - Exemple : PRENOMS peut être obligatoire lors de l'inscription de l'étudiant
  - NULL : Champ facultatif
    - Une colonne marquée à NULL peut être omise lors d'une insertion c'est-à-dire qu'elle ne peut prendre la valeur NULL
      - Exemple : CODE POSTAL peut être facultatif lors de l'inscription de l'étudiant

Modèle relationnel : Contraintes

- **DEFAULT**
  - Le champ est optionnel pour l'utilisateur mais obligatoire pour le Système
  - Le Système propose une valeur par défaut
  - Si l'utilisateur ne propose pas de valeur alors celle par défaut est prise automatiquement
    - Exemple : PAYS peut avoir comme valeur par défaut SENEGAL

- **CHECK**
  - Une contrainte de vérification lors des insertions et des mises à jour
  - Permet de garantir la cohérence des données et de reduire les erreur
  - Utiliser pour les données booléennes (Avec seulement deux valeur) ou pour limiter un ensemble
  - Exemples :
    - SEXE dans les normes ne peut prendre que M ou F
    - ETAT_COMPTE ne prend que ACTIF ou NON ACTIF à la création du compte
    - AGE_ETUDINT doit être supérieur à 16 à l'inscription
    - QUANTITE_ACHAT doit être supérieure à 0 lors d'une commande

Modèle relationnel : Contraintes

- **PRIMARY KEY = UNIQUE + NOT NULL**
  - Une ou un ensemble de colonnes de la relation
    - Unique : Pas de redondance pour n-uplet de la relation
    - Obligatoire : Pas facultatif pour n-uplet de la relation
  - Exemple : INE pour la relation étudiant
  - Un identifiant peut-être composé de plusieurs attributs
    - On dira que la clé ou l'identifiant est **composite**.
    - Exemple : Numéro_Salle et Batimat peuvent contituer la clé primaire de la relation Salles
  - Attention
    - On peut avoir une relation sans identifiant explicite
    - Si la relation n'a qu'un seul identifiant candidat (Unique) alors celui-ci peut être choisi comme la clé primaire
    - Si la relation a plusieurs identifiants candidats (Unique) alors il faudra choisir parmi eux la clé primaire et "le meilleur » choix sera celui qui est Obligatoire.
      - Les autres identifiants seront déclarés avec la clause UNIQUE

Modèle relationnel : Contraintes

- **FOREIGN KEY : Clé étrangère**
  - Un colonne d'une relation dont les valeurs proviennent forcement d'une colonne d'une autre relation
  - Une relation peut avoir plusieurs clés etrangères
  - Certains attributs d'une relation font référencent à des tuples d'une autre relation (ou parfois de la même);
    - C'est-à-dire que leurs valeurs sont toujours issues de celles de l'identifiant d'un tuple existant d'une autre relation.
  - Explication
    - Soient deux relations R1(X, Y) et R2 (V, W), où X, Y, V, W, désignent des attributs ou des ensembles d'attributs, et où X est un identifiant de R1
    - On dit que W est un identifiant externe sur R1 (ou que W référence R1) si pour tout tuple de R2, la valeur prise par W est nécessairement la valeur de X pour un tuple existant de R1.
      - Autrement dit, à tout instant, l'ensemble des valeurs prises par W est compris dans l'ensemble des valeurs prises par X.

Modèle relationnel : Contraintes

- **FOREIGN KEY : Clé étrangère**
- Une fois déclaré, l'identifiant externe W de R2 qui fait référence à X de R1, le SGBD vérifie automatiquement :
  - Lors d'une insertion dans R2 : Que la valeur mis dans W existe dans X de R1. Si ce n'est pas le cas l'insertion est refusée;
  - Lors d'une mise à jour dans R2 : Que la valeur de remplacement mis dans W existe dans X de R1. Si ce n'est pas le cas la modification est refusée;
  - Lors d'une Suppression dans R1 : Qu'il n'existe pas de tuple dans R2 référençant ce tuple de R1 à supprimer. S'il Oui, selon le SGBD :
    - Soit la suppression est refusée
    - Soit la suppression est propagée: les tuples de R2 qui référençaient cette valeur de X sont eux aussi supprimés (FORCING

- Il y a une situation **d'intégrité référentielle** lorsqu'à chaque valeur de la clé étrangère W correspond une valeur de la clé primaire référencée X.

Modèle relationnel : Contraintes

- **FOREIGN KEY : Clé étrangère**
- Une fois déclaré, l'identifiant externe W de R2 qui fait référence à X de R1, le SGBD vérifie automatiquement :
  - Lors d'une insertion dans R2 : Que la valeur mis dans W existe dans X de R1. Si ce n'est pas le cas l'insertion est refusée;
  - Lors d'une mise à jour dans R2 : Que la valeur de remplacement mis dans W existe dans X de R1. Si ce n'est pas le cas la modification est refusée;
  - Lors d'une Suppression dans R1 : Qu'il n'existe pas de tuple dans R2 référençant ce tuple de R1 à supprimer. S'il Oui, selon le SGBD :
    - Soit la suppression est refusée
    - Soit la suppression est propagée: les tuples de R2 qui référençaient cette valeur de X sont eux aussi supprimés (FORCING

- Il y a une situation **d'intégrité référentielle** lorsqu'à chaque valeur de la clé étrangère W correspond une valeur de la clé primaire référencée X.

Modèle relationnel : Contraintes

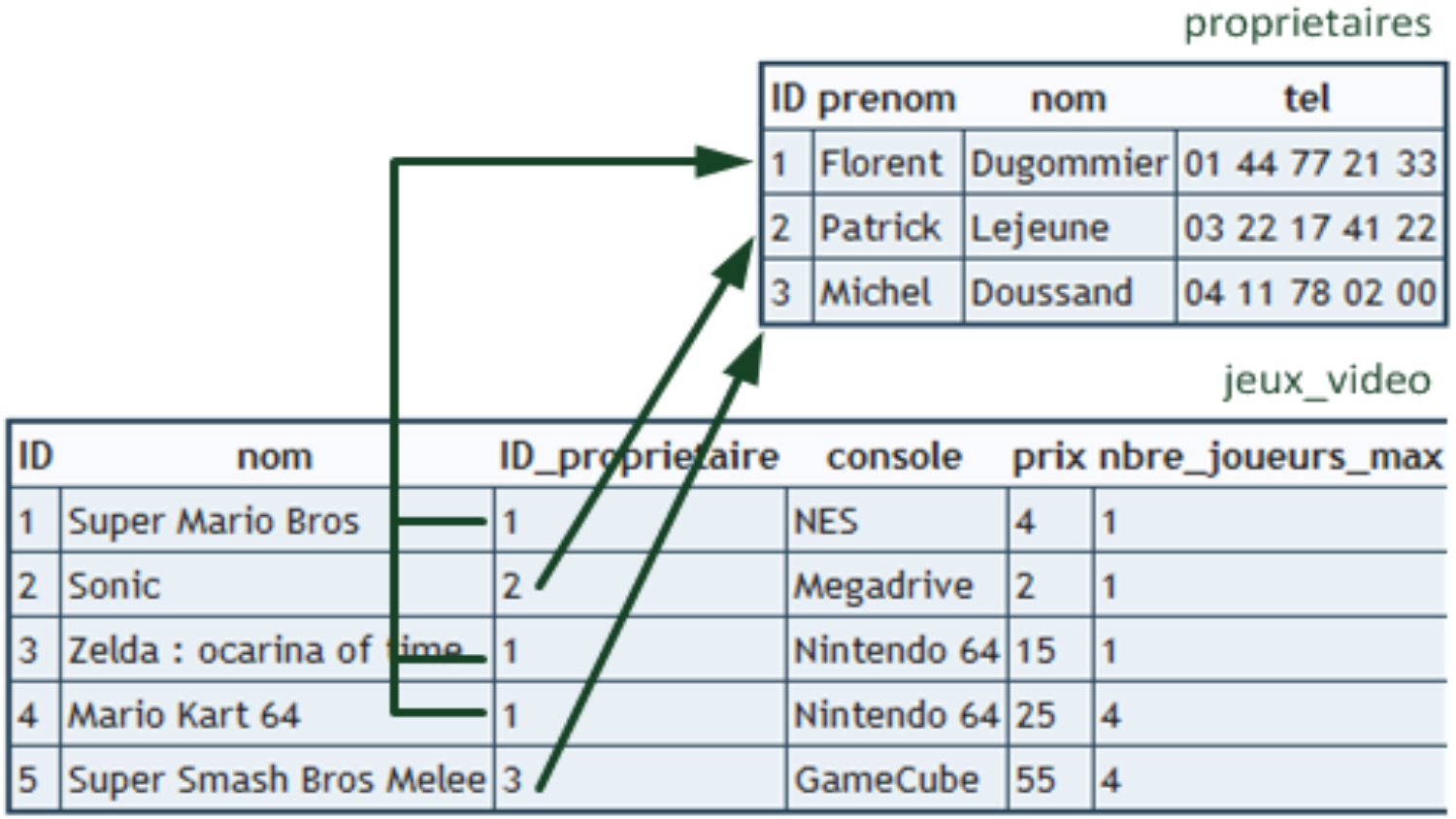
Exemple 1 : La relation CLIENTS

- Matricule est la clé primaire = PRIMARY KEY
- Prenom et nom sont obligatoire = NOT NULL
- Sexe est optionnel = NULL
- Ville à comme valeur par defaut DAKAR = DEFAULT
- Sexe est soit M soit F = CHECK

	MATRICULE	PRENOM	NOM	VILLE	SEXE
1	C007	Veronique	TEND...	Poffine ...	F
2	C008	Fatou	Dafe	Mbour ...	F
3	C001	Edouard	SARR	Mbour ...	M
4	C002	Marie Angelique	Senghor	Dakar ...	F
5	C003	Amadou Nar	DIOP	NGOHE ...	(null)
6	C004	Ngone	DIOP	Mbour ...	F
7	C005	Maxime	Diatta	Ngohe ...	M
8	C006	Fatou	FALL	Mbour ...	F
9	C009	Anissa	SAGNE	Dakar ...	F
10	C0010	Ousmane	SENE	Mbour ...	M

Modèle relationnel : Contraintes

Exemple 2 : Clé Etrangère



proprietaires

ID	prenom	nom	tel
1	Florent	Dugommier	01 44 77 21 33
2	Patrick	Lejeune	03 22 17 41 22
3	Michel	Doussand	04 11 78 02 00

jeux_video

ID	nom	ID_proprietaire	console	prix	nbre_joueurs_max
1	Super Mario Bros	1	NES	4	1
2	Sonic	2	Megadrive	2	1
3	Zelda : ocarina of time	1	Nintendo 64	15	1
4	Mario Kart 64	1	Nintendo 64	25	4
5	Super Smash Bros Melee	3	GameCube	55	4

Modèle relationnel : Contraintes

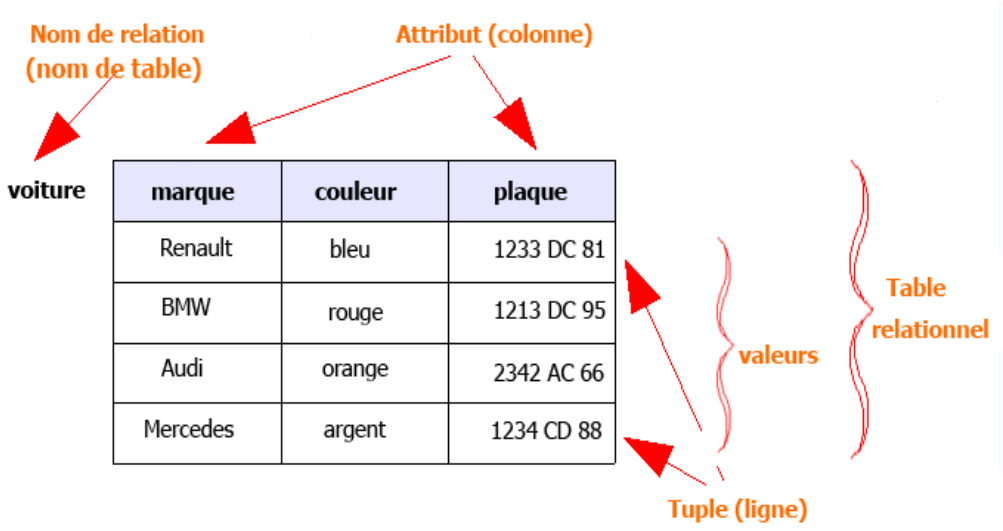
Exemple 3: Clé Etrangère, Unique et Not NULL

	ID_FOURNISSEUR	PRENOM	NOM
1	F0001	Awa	DIOUF
2	F0002	Alimata	DIOUF
3	F0004	Mamadou	KANDE
4	F0005	Cheikou Wade	GUEYE
5	F0003	Bernadeth	DIOME

	ID_PRODUIT	NOM_PRODUIT	QUANTITE	UNITE	TYPE	PRIX	ID_FOURNISSEUR
1	P0001	Vitalait	9234	Sachet ...	LAIT	1200	F0001
2	P0002	Baralait	1342	Sachet ...	LAIT	1500	F0001
3	P0003	Santex	734	Savon ...	SAVON	350	F0002
4	P0005	Tigo	233	Carte ...	CREDIT	500	F0005
5	P0004	Riz	54	Tonne ...	RIZ	80	F0003

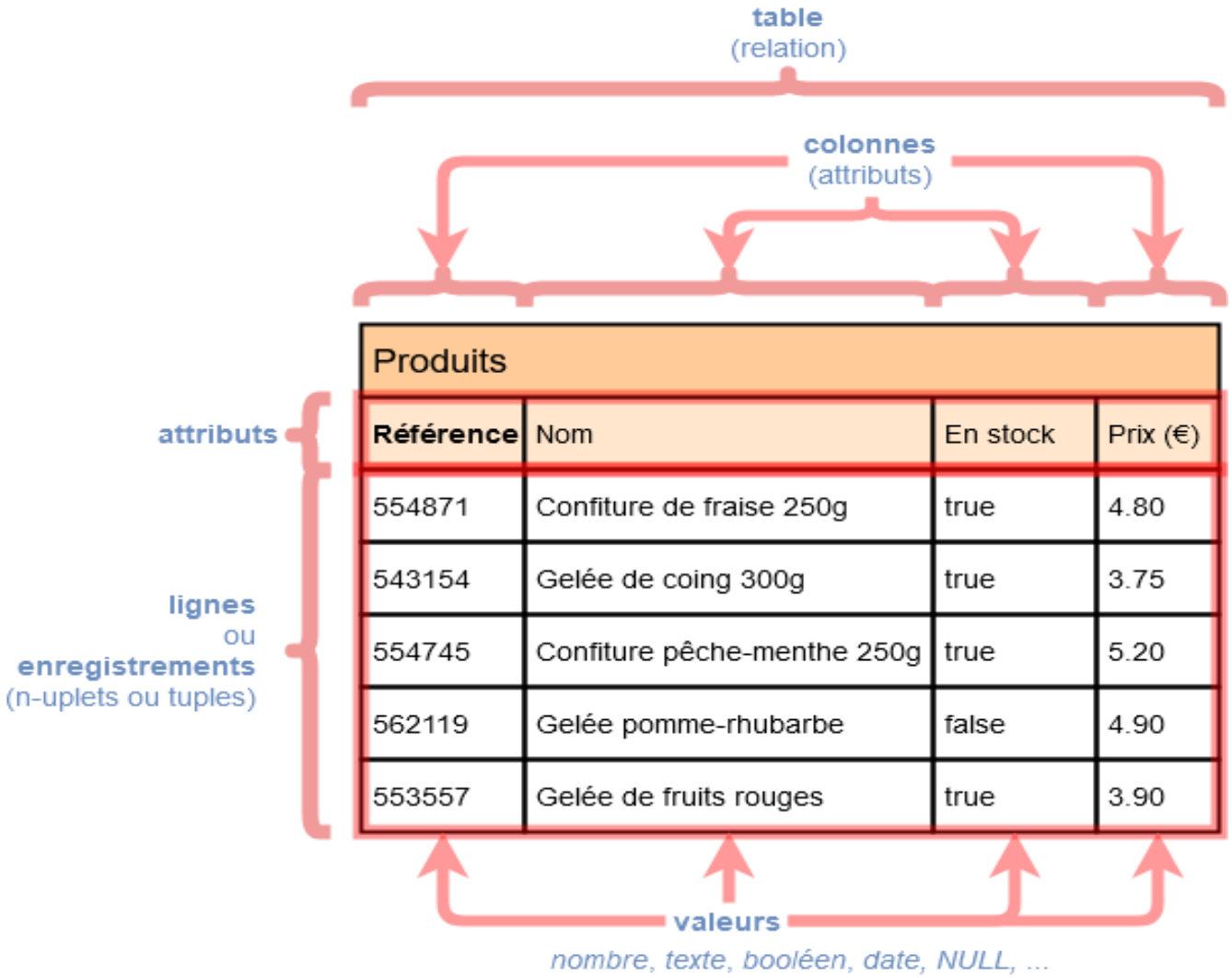
Les objets de la base : La table

- **Unité principale du model relationnel**
- Un ensemble de données organisées sous forme d'un tableau à deux dimension
- Constitué de :
  - Plusieurs lignes ou enregistrements : les membres de la table
  - Plusieurs colonnes ou champs : Les informations élémentaires de la table
    - Chaque colonne à un domaine (TYPE)
    - Les types primitifs sont
      - Entiers : int, Number, Long
      - Réels : Double, Float, Number
      - Chaines de caractères : String, Varchar, Text
      - Dates : DATE
      - Date & heure (h:m) : DATETIME
      - Date & heure (h:m:s:t)



Les objets de la base : La table

- **Unité principale du model relationnel**

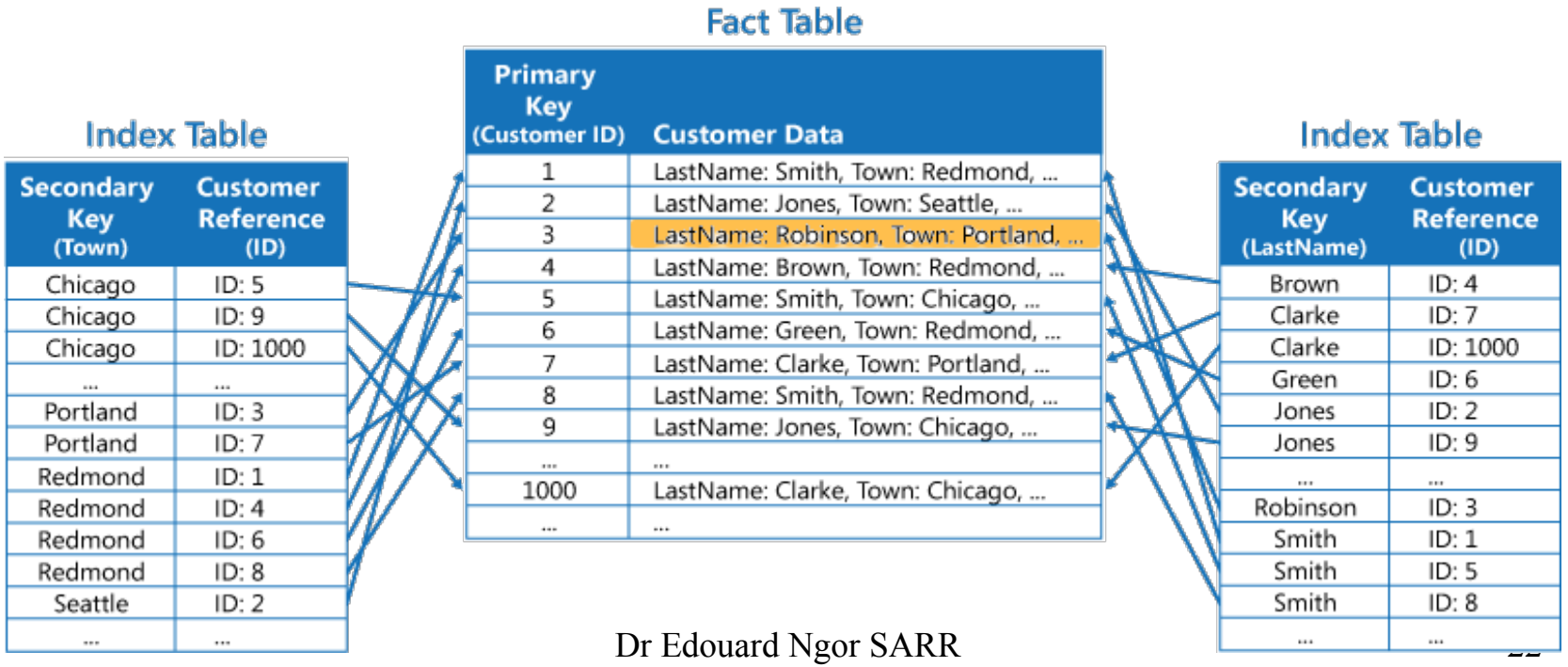


Produits

Référence	Nom	En stock	Prix (€)
554871	Confiture de fraise 250g	true	4.80
543154	Gelée de coing 300g	true	3.75
554745	Confiture pêche-menthe 250g	true	5.20
562119	Gelée pomme-rhubarbe	false	4.90
553557	Gelée de fruits rouges	true	3.90

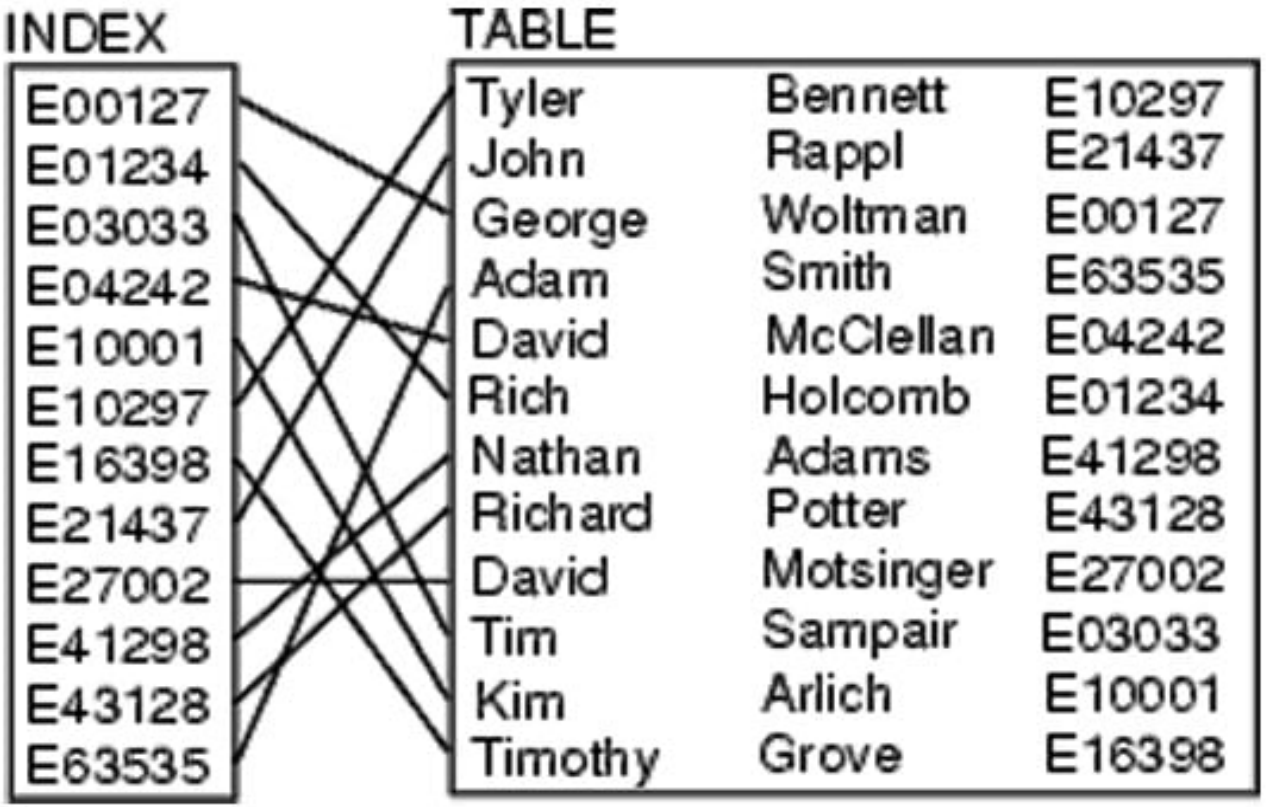
Les objets de la base : L'indexe

- Structure de données redondante Organisée de manière à accélérer la recherches.
- Consiste à découper les données à stocker en plus petites parties ordonnées et ce de manière récursive.
  - Ainsi retrouver une information indexée consiste à naviguer de branche en branche dans l'index en éliminant à chaque étape un grand nombre de cas.



Les objets de la base : L'indexe

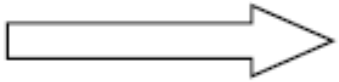
- Structure de données redondante Organisée de manière à accélérer la recherches.



Les objets de la base : La vue

- Représentation virtuelle d'un résultat d'une requête SELECT présenté sous forme d'une table virtuelle à laquelle on donne un nom.
- La définition de la vue est enregistrée dans la base de données, mais les lignes correspondant à la vue ne le sont pas
- La suppression d'une vue n'entraîne pas la suppression des données
- **Utilisation :** La vue est utilisé en consultation comme si elle était une table
- **Rôle de la vue :** Pour Montrer à un utilisateur que ce dont vous voulez qu'il voit.
- **Forme 1 : Vue réductrice**
  - C'est une photographie complète ou partielle d'une table.

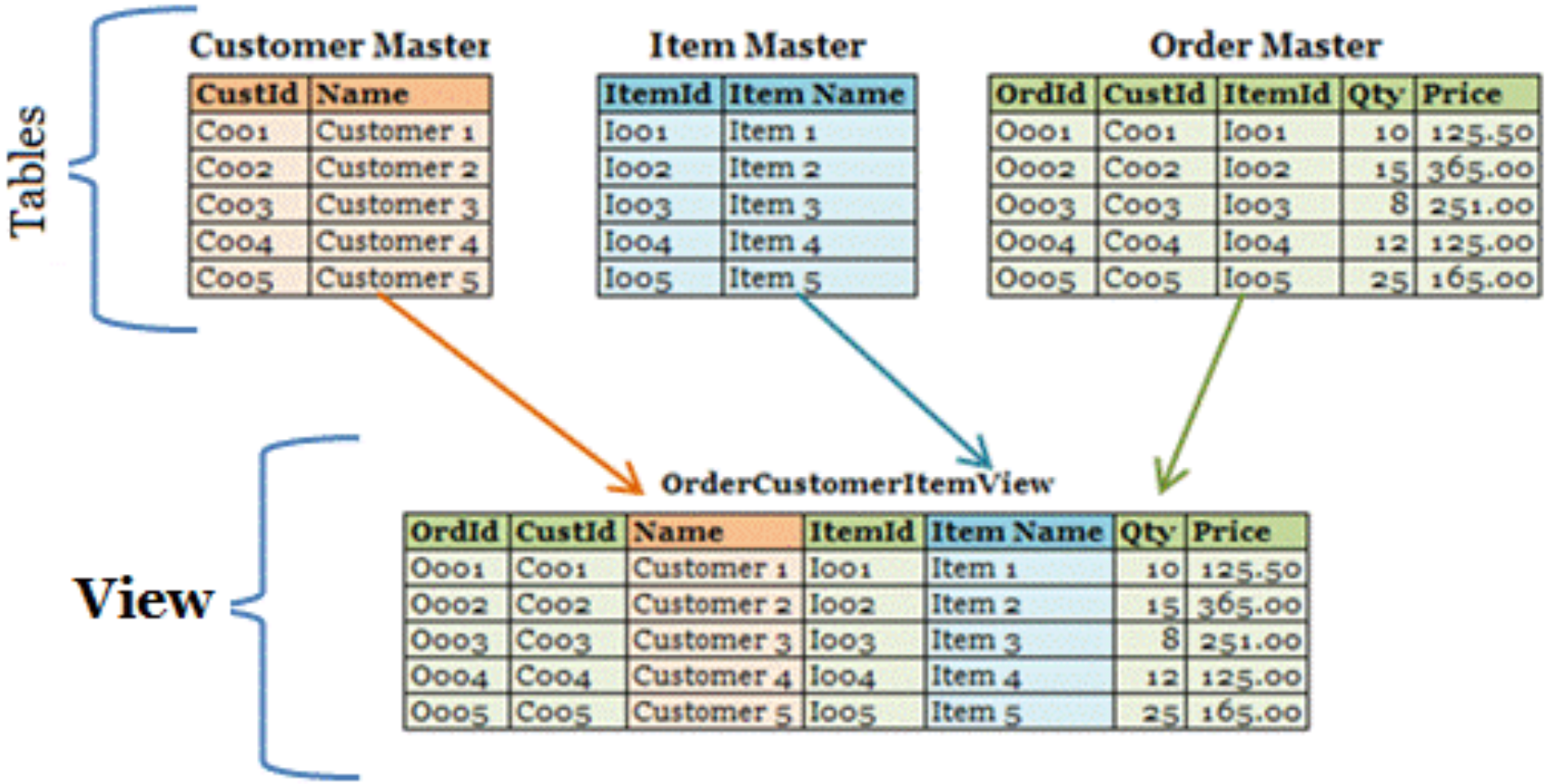
ID...	PRENOM	NOM	DATE_EMBAUCHE	SALAIRE
5	Malick	Diouf	09/12/13	3400000
4	Ngor	Sarr	12/11/13	220000
3	Marie Noel	Ciss	11/12/14	2500000
2	Fatou	Fall	01/05/12	300000
1	alioune	Faye	10/03/12	200000



PRENOM	NOM	SALAIRE
Malick	Diouf	3400000
Fatou	Fall	300000
Marie Noel	Ciss	2500000

Les objets de la base : La vue

- **Forme 1 : Vue large**
  - C'est une fusion de morceaux provenant de déférentes tables



Les objets de la base : Séquence et Synonyme

Une Séquence

- Une suite de nombres entiers évolutive.
- Peut permettre :
  - De générer des clés uniques dans des tables
  - Avoir un compteur à titre informatif, que l'on incrémente quand on veut
  - Exemple : AUTO INCREMENTATION

Un Synonyme

- Un autre nom donné a un Objet : Une table ou Une Vues
- Utiliser pour ramifier les noms et faciliter l'utilisation dans les requetes
- C'est Nom plus simple à Mémorise et à Utiliser
- Exemple :
  - Employes peut avoir comme synonyme Emp
  - Etudiant_licence peut avoir comme synonyme EtL

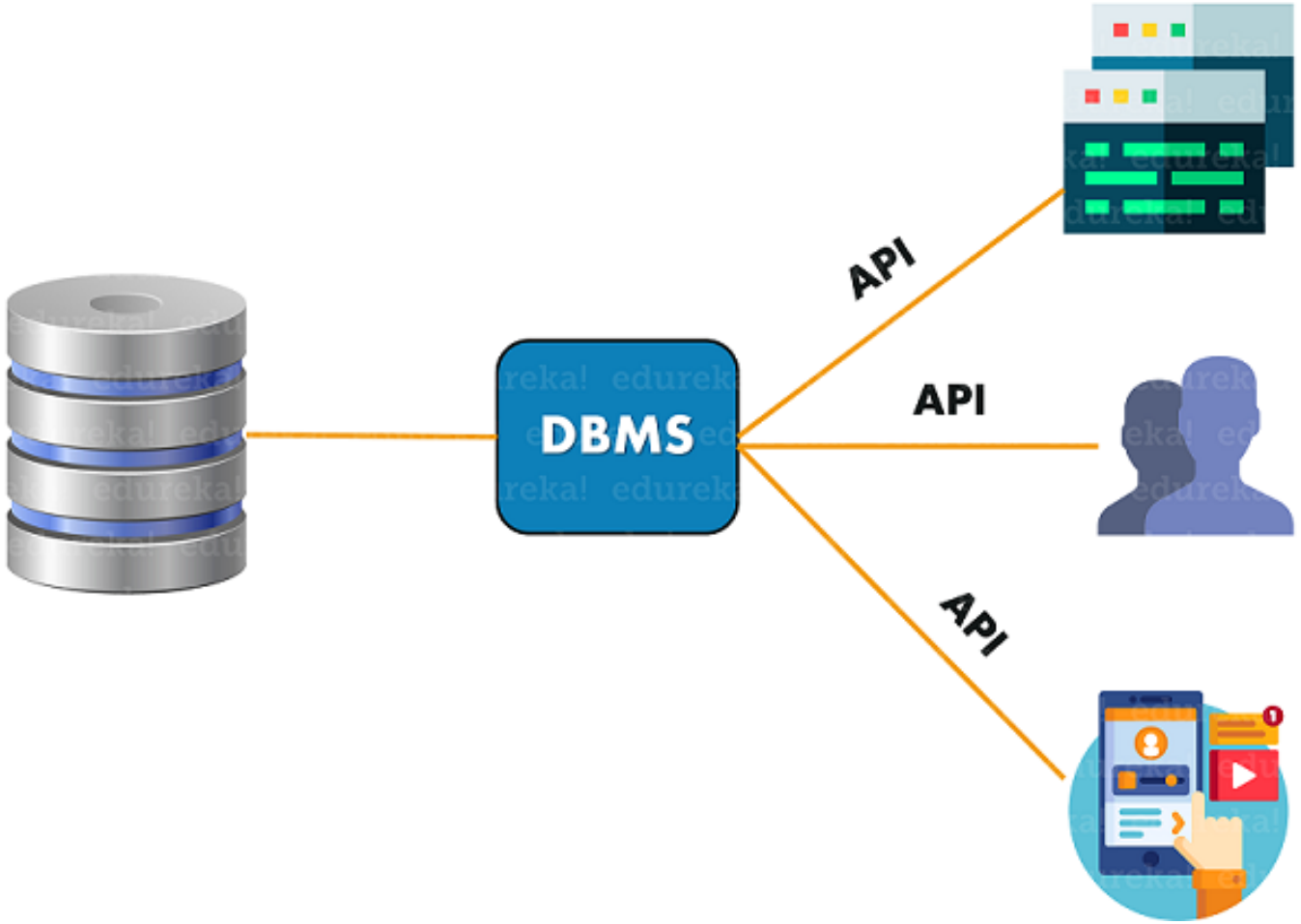
Le SGBDR

SGBDR : Définitions

SGBD : Système de gestion des bases de données

- Un logiciel qui fait office d'interface entre les utilisateurs et les données stockées dans la base
- Les SGBDR Sont
  - ***tous basés sur le model relationnel de Edgar Codd***
  - ***Respectent la regle ACID***
  - ***Utilise Tous le SQL comme langage d'interrogation***
- **Fonctions fondamentales**
  - **Stockage** : Stocker l'information de façon structurée et fiable
  - **Massification** : Traiter de grands volumes de données
  - **Optimisation** : optimisation Traiter rapidement les données
  - **Sécuriser** : Sécuriser les accès aux données
  - **Cohérence** : Contrôler la qualité des données
  - **Partage** : Partage les données entre applications dédiées et utilisateurs
  - **Cohérence** : Rendre accessible les données en réseau

SGBDR : Définitions



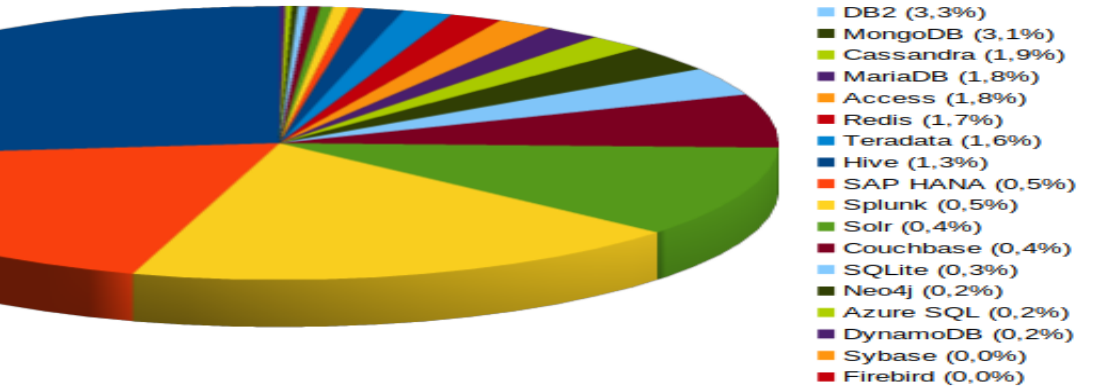
SGBDR : le marché

Microsoft SQL Server

ORACLE

PostgreSQL

MySQL



SGBDR

Faiblesses des SGBDR

- Pas adaptées à la gestion de données
  - Non-structurées
  - Volumineux (Big Data)
- Lenteurs face au Big Data user à cause des règles ACID à respecter.
- L'évolutivité (Scalabité) horizontale est en général plus rapide et plus économique que l'évolutivité verticale
- Difficile de faire partitionnage (qui consiste à partitionner et distribuer les données sur un ensemble de machines) poser des problèmes et mettre en danger le conformité ACID.

Forces des SGBDR

- Il permet de gérer des BDD respectant les 4 critères ACID :
- Garantit donc la sécurité des transactions.
- Les données sont stockées et manipulées facilement grâce aux requêtes SQL.
- Les BDD relationnelles sont Scalables.
  - L'augmentation du volume de données n'a pas d'incidence sur les données existantes et l'organisation de la base.
- Proposent une bonne gestion des accès et des droits d'utilisation est optimale.

SGBDR

Gestion des transactions

- Une transaction un ensemble atomique de requêtes SQL dont le traitement ne peut être fait séparément..
- Une transaction est une suite d'opérations interrogeant et/ou modifiant la BD, pour laquelle l'ensemble des opérations doit être soit validé, soit annulé.
- **La règle ACID**
  - **Atomicité:** l'ensemble des opérations d'une transaction est soit exécuté en bloc, soit annulé en bloc
  - **Cohérence :** Une transaction fait passer une BD d'un état cohérent à un autre état cohérent. Un état cohérent est un état dans lequel les contraintes d'intégrité sont vérifiées.
  - **Isolation**: Une transaction se déroule sans être perturbée par les transactions concurrentes : tout se passe comme si elle se déroulait seule.
  - **Durabilité:** une fois une transaction confirmée, les données correspondantes restent durablement dans la base, même en cas de panne

Introduction au SQL

Introduction au SQL

- **SQL (Structured Query Langage)**
- **Un langage de données**
  - est un langage de base de données relationnelle.
  - Un langage informatique permettant de décrire et de manipuler les schémas et les données d'une BD
  - Commun à tous les SGBDR

Introduction au SQL

Un langage de données

- Il permet:
  - La création de base et des tables.
  - L'ajout d'enregistrements sous forme de lignes.
  - L'interrogation de la base.
  - La mise à jour.
  - Le changement de structure de la table:
    - Ajout
    - MAJ
    - Suppression de colonnes.
  - La gestion de droits d'utilisateurs de la base.
-

Introduction au SQL

Langage de Définition de Données LDD ou DDL :

- Permet la description de la structure de la base de données (tables, vues, attributs, index). Cette ordre est composé des verbes suivants :
  - CREATE pour ajouter un objet dans la base
  - DROP pour supprimer un objet de la base
  - ALTER pour modifier le comportement d'un objet de la base
  - RENAME pour renommer un objet de la base
  - TRUNCATE pour vider une table

Introduction au SQL

Langage de Manipulation de Données LMD ou DML :

- Permet la manipulation des tables et des vues avec les quatre commandes :
  - SELECT : Rechercher et afficher des informations des tables
  - UPDATE : Modifier les enregistrements ou lignes des tables
  - INSERT : pour insérer de nouvelles lignes dans nos tables
  - DELETE : pour supprimer des lignes
  - MERGE : pour synchroniser des données
  -

Introduction au SQL

Langage de Contrôle de Données LCD ou DCL :

- Comprend les primitives de gestion des transactions:
- GRANT pour affecter des privilèges et rôles sur des données à des utilisateurs.
- REVOKE pour récupérer ces privilèges et rôles.

Langage de contrôle transactionnel LCT ou TCL

- Pour l'élaboration et contrôle de transactions. Ils sont au nombre de trois :
- COMMIT pour valider la transaction
- ROLLBACK pour annuler une transaction
- SAVEPOINT pour créer un point de retour ou de reprise.
-

DDL

DDL : Langage de définition des données

La DDL ou LDD langage de définition de données.

- Ce sont des requêtes à commutation directe ou automatique:
  - Une fois lancé, l'opération se valide automatiquement sans un COMMIT.
- Il permet de créer, supprimer, modifier, renommer, ou tronquer les objets de la base de données.
- **Un Objet de la base peut être :**
  - TABLE: unité logique composée de plusieurs lignes
  - VIEW: représentation logique d'un ensemble de lignes provenant d'une table ou plusieurs tables.
    - SEQUENCE: permet de générer des valeurs numériques
    - INDEX: permet d'augmenter la performance des recherches de données
    - SYNONYM: pour donner des synonymes à mos tables

DML: Langage de manipulation des données

CREATION D'UN OBJET: CREATE

- Le CREATE sert à créer des objets comme des tables, des vues ou des indexes

Ainsi nous avons :

- CREATE TABLE
- CREATE VIEW
- CREATE INDEX
- CREATE SYNONYM
- CREATE SEQUENCE

DDL: Langage de définition des données

CREATION D'UN OBJET: CREATE TABLE

```
CREATE TABLE  NOMTABLE (
      COLONNE1  TYPE2  CONTRAINTES,
      COLONNE2  TYPE2  CONTRAINTES,
      .... ;
      CONSTRAINT  NOM_CONTRAINTE1 TYPE_CONTRAINTE1,
      CONSTRAINT  NOM_CONTRAINTE2 FOREIGN KEY  REFERENCES NOMTABLE
) ;
```

DDL: Langage de définition des données

CREATION D'UN OBJET: CREATE TABLE

- Nous avons deux types de contrainte en SQL Oracle:
- **Les Contraintes niveau Colonne**
- Indiquée juste après la spécification de la colonne dans CREATE :
    - Prenom Varchar2 NOT NULL,

- **Les Contraintes niveau Table**
- Indiquée après la spécification de toutes les colonnes de table dans CREATE. On utilisera le mot CONSTRAINT.
    - CONSTRAINT primary Key (NumCin),

NOT NULL, DEFAULT et CHECK sont le plus souvent utilisés avec une contrainte de niveau Colonne alors que PRIMARY KEY et FOREGH KEY sont eux utilisés avec une contrainte de niveau table.

DDL: Langage de définition des données

CREATION D'UN OBJET: CREATE TABLE

- Les contraintes

NULL ou NOT NULL	Accepter ou non les valeurs NULL
UNIQUE	N'accepte pas de doublons
CHECK	Condition à vérifier. Elle prend entre les parenthèses toutes les conditions de la close WHER exceptée les sous interrogations : < > <= >= <> BETWEEN END OR NOT
PRIMARY KEY	Clé primaire : Elle est obligatoire et non redondante.
FOREIGN KEY	Clé étrangère

- Faire préfixer le nom de la contrainte par PK ou FK n'est pas une obligation mais ca nous permet de retrouver le plus facilement une contrainte et son type.

DDL: Langage de définition des données

CREATION D'UN OBJET: CREATE TABLE

- Exemples

```
CREATE TABLE CLIENTS (
Matricula VARCHAR2(45)  CONSTRAINT   PK_CLIENTS  PRIMARY KEY,
Prenom   Varchar2(57)  NOT NULL,
Nom   Varchar2(40)  NOT NULL,
Ville    Char (65)  DEFAULT   'NGOHE',
Sexe   Char (1) CHECK (Sexe IN ('M', 'F'))  ) ;
```

```
CREATE TABLE FOURNISSEURS (
    id_fournisseur      VARCHAR2(45) ,
    Prenom      Varchar2(57)      NOT NULL,
     Nom       Varchar2(40)     NOT NULL,
    Adresse Char (65)   DEFAULT 'Castor',
    Telephone    Char (9) ,
    CONSTRAINT    PK_FOURNISSEUR PRIMARY KEY (id_fournisseur)
) ;
```

DDL: Langage de définition des données

CREATION D'UN OBJET: CREATE TABLE

- Exemple

```
CREATE  TABLE  PRODUITS(
 id_produit  VARCHAR2(45) ,
 Nom_Produit Varchar2(57) NOT NULL,
 Quantite  NUMBER(5,0) NOT NULL,
 Unite    Char(40) NOT NULL ,
 Type varchar2(27) NOT NULL,
 Prix NUMBER(7) NOT NULL,
 Id_Fournisseur Varchar2(45),
 CONSTRAINT PK_Produits PRIMARY KEY (id_Produit),
 CONSTRAINT FK_Produit FOREIGN KEY (id_fournisseur) REFERENCES FOURNISSEURS
) ;
```

DDL: Langage de définition des données

CREATION D'UN OBJET: CREATE TABLE

- Créer une table avec la même structure et les mêmes enregistrements qu'une autre table

Ceci revient à la copie d'une table dans une autre.

```
CREATE   TABLE T2 AS (SELECT * FROM T1 );
```

Avec cette syntaxe les deux tables seront identiques alors que

```
CREATE TABLE T2 AS (SELECT COL1, COL2, COL3 FROM T1  );
```

Ici la table T2 aura le même nombre de ligne que la table T1 mais seulement avec 3 colonnes de T1.

Cas pratique 1 : Créons une table CLIENTS_COPY qui est une copie identique de la table CLIENTS.

```
CREATE TABLE CLIENTS_COPY AS
(SELECT * FROM CLIENTS);
```

DDL: Langage de définition des données

CREATION D'UN OBJET: CREATE TABLE

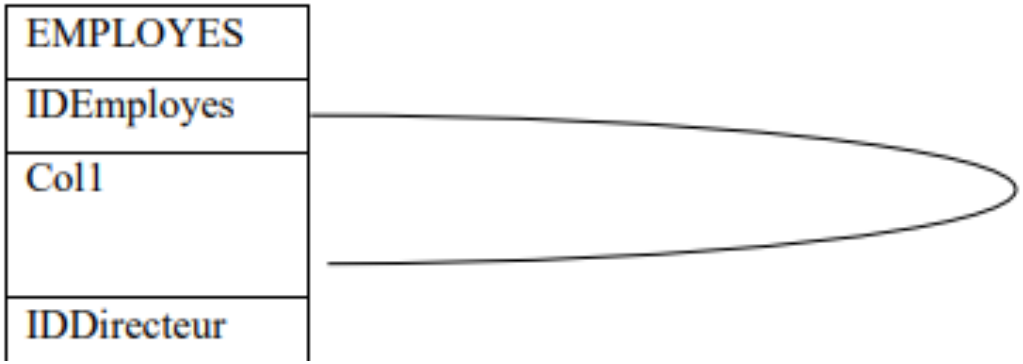
- Créer une table Vide avec la même structure qu'une autre.
  - Pour ce faire, nous savons qu'une requête n'est traitée que lorsque la condition qui l'accompagne est Vraie.
  - En prenant la syntaxe vue précédemment, il suffit simplement de lui ajouter une condition non vérifiée pour limiter l'exécution à l'ordre CREATE TABLE.
  - Exemple de condition non vraie : 1=2 ou 3=0 ou même a=b.

```
CREATE TABLE T2 AS
    ( SELECT * FROM T1
      WHERE 1=2 );
```

DDL: Langage de définition des données

CREATION D'UN OBJET: CREATE TABLE

Création de table réflexive :



- Pour créer c'est genre de table il faut procéder en deux étapes :
  - **Créer la table T1 avec seulement sa clé primaire.**
  - **Ajouter une colonne avec une contrainte clé étrangère avec ALTER TABLE**
    - **Syntaxe :**
      - ALTER TABLE T1 ADD (nom_Colonne Type CONSTRAINT Nom_Constraint Type_Constraint).

DDL: Langage de définition des données

CREATION D'UN OBJET: CREATE VIEW

- Nous allons utiliser CREATE OR REPLACE VIEW qui est plus pratique que CREATE VIEW pour deux raisons : Car ALTER VIEWS n'existe pas
- Afin d'évité de supprimer et de recréer une vue lorsque celle-ci existe, on la remplace sinon elle est créée.

```
CREATE OR REPLACE VIEW NOM_VUE
AS (SOUS REQUETE SELECT );
```

Exemple :

```
CREATE OR REPLACE VIEW VUE_CLIENT
AS (SELECT * FROM CLIENTS WHERE ville = 'MBOUR' );
```

DDL: Langage de définition des données

CREATION D'UN OBJET: CREATE INDEX

- Permet de jouer sur les performances des requêtes c'est comme un pointeur sur une valeur ou une information sur la base de données.
- Une indexe est indépendante de la table où elle est liée.
- Sa création peut se faire de manière implicite donc faite derrière par le système ou de manière explicite, entamée par l'utilisateur

```
CREATE [UNIQUE] [BITMAT] INDEX Nom_Idexe
ON NomTable ( Colonne1, Colonne2 …) ;
```

Exemple :

```
CREATE INDEX Clients_prenom_index
On Clients (Prenom);
```

DDL: Langage de définition des données

CREATION D'UN OBJET: CREATE SYNONYM

- Un synonyme est un autre nom donné à un objet d'une de ma base de données. Pour donner des synonymes aux objets de la base
- Le mot clé Public permet montrer que ce synonyme est accessible par tous les utilisateurs ayant les privilèges.

Un synonyme pour la table CLIENTS : CREATE SYNONYM LESCLIENTS FOR CLIENTS ;
Un synonyme public pour la table CLIENTS : CREATE PUBLIC SYNONYM NOSCLIENTS FOR CLIENTS;
Un synonyme pour la vue VUE_CLIENTS: CREATE SYNONYM LAVUECLIENT FOR VUE_CLIENT;
Un synonyme pour l'index PK_CLIENT: CREATE SYNONYM IndexTclient FOR PK_CLIENT;

DDL: Langage de définition des données

SUPPRESSION D'UN OBJET: DROP TABLE

- DROP est une requête de la DDL qui nous permet de supprimer un objet de ma base de données.
- **Supprimer définitivement avec DROP … PURGE**

```
DROP  Nom_Objet PURGE ;
```

PURGE permet de supprimer définitivement un objet. Donc plus de possibilité de récupération avec FLASHBACK.

Exemple :

```
DROP  CLIENTS_COPY  PURGE ;
```

- DROP TABLE EMPLOYES ;
- DROP TABLE EMPLOYES CASCADE
- DROP SYNONYM LES EMPLOYES ;
- DROP VIEW LES CLIENTS ;
- DROP INDEX Clients_prenom_index;
- DROP SEQUENCE SeqMatricule;

DDL: Langage de définition des données

MODIFICATION D'UN OBJET: ALTER TABLE

- ALTER TABLE est très complet. Il nous nous permettra entre autres de:
  - Ajouter des colonnes ADD
  - Ajouter une Contrainte ADD CONSTRAINT …. USING INDEX ….
  - Modifier des colonnes MODIFY
  - supprimer des colonnes DROP
  - Renommer une colonne RENAME
  - Renommer une table RENAME COLUMN
  - Marquer une colonne à suppression SET UNUSED
  - Suppression de contrainte
  - Activer et désactiver une contrainte
  - Renommer une contrainte
  - Mettre une table en lecture seule READ ONLY
  - Mettre une table en lecture ecriture READ WRITE

DDL: Langage de définition des données

MODIFICATION D'UN OBJET: ALTER TABLE

```
ALTER   TABLE   NOM_TABLE
   ADD  (  Nom_Colonne1    Type_Colonne1    Constraintes1,
           Nom_Colonne2    Type_Colonne2    Constraintes2 ) ;
```

```
ALTER   TABLE   CLIENTS ADD CONSTRAINT C_S CHECK (VILLE <> 'Ngohe') ;
```

```
Alter Table CLIENTS MODIFY (
  PAYS Varchar2(349) default 'Senegal');
```

```
Alter Table   CLIENTS   DROP  ( PAYS );
```

```
Alter Table CLIENTS RENAME TO CLIENTS2;
```

```
Alter Table  CLIENTS  RENAME COLUMN VILLE TO  VILLES_CLIENT;
```

```
ALTER TABLE  CLIENT2   READ   ONLY ;
```

DDL: Langage de définition des données

VIDER UNE TABLE : TRUNCATE

- TRUNCATE permet de vider les lignes d'une table.
- Il a les mêmes fonctionnalités que DELETE mais contrairement à DELETE.
- Il est à commutation directe. Faire un TRUNCATE NOMTABLE revient à faire DELETE FROM NOMTABLE.
- TRUNCATE appartient au DDL donc une transaction à validation automatique c'est à dire n'a pas besoin de COMMIT pour être valider. Et dans ce cas
- Aucune possibilité de retour en arrière ou d'annulation avec ROOLBACK
- Aucune possibilité de récupération avec un FLASHBACK

DDL: Langage de définition des données

VIDER UNE TABLE : TRUNCATE

- TRUNCATE permet de vider les lignes d'une table.
- Faire un TRUNCATE NOMTABLE revient à faire DELETE FROM NOMTABLE.
- TRUNCATE appartient au DDL donc une transaction à validation automatique c'est à dire n'a pas besoin de COMMIT pour être valider. Et dans ce cas
- Aucune possibilité de récupération avec un FLASHBACK

```
TRUNCATE  NomTable ;
```

Exemple :

```
TRUNCATE  PRODUITS
```

Cette requête va vider le contenu de la table PRODUITS.

DDL: Langage de définition des données

RONOMMER UN OBJET: RENAME

- TRUNCATE permet de vider les lignes d'une table.
- **Rename** est une instruction qui permet de renommer le nom d'un objet par exemple le nom d'une tale TABLE.

```
RENAME TABLE   NOM_TABLE    TO   NOUVEAU_NOM_TABLE ;
```

Exemple :

```
RENAME   TABLE  CLIENTS2  TO  CLIENTS3;
```

DML

DML: Langage de manipulation des données

Langage de Manipulation de Données LMD ou DML :

- Permet la manipulation des tables et des vues avec les quatre commandes :
  - SELECT : Rechercher et afficher des informations des tables
  - UPDATE : Modifier les enregistrements ou lignes des tables
  - INSERT : pour insérer de nouvelles lignes dans nos tables
  - DELETE : pour supprimer des lignes
  - MERGE : pour synchroniser des données
  -

DML: Langage de manipulation des données

Insertion de données avec INSERT

- Permet d'ajouter des données dans mes objets et plus précisément dans nos tables.

```
INSERT  INTO  NomTable  ( Colonne1, Colonne2 ,…..  ColonneN)
                VALUES  ( valeur1, Valeur2,  …… ValeurN) ;
```

- Cas d'exemple:

Voici la structure de la table CLIENTS avec la commande DESC ou DESCRIBE

```
DESC CLIENTS ;
Nom                 NULL          Type
--------- -------- ------------ ----------------------
MATRICULE    NOT NULL      VARCHAR2(45)
PRENOM         NOT NULL     VARCHAR2(57)
NOM              NOT NULL     VARCHAR2(40)
VILLE             CHAR(65)
SEXE              CHAR(1)
```

DML: Langage de manipulation des données

Insertion de données avec INSERT

- REGLES :
  - *Autant de colonnes listées que de valeurs soumises.*
  - *Donner respectivement les types de valeurs correspondants à ceux des colonnes.*
  - *Lister les colonnes de la table est optionnel*
  - *L'ordre des valeurs de correspondance avec les colonnes est très important*
  - *Les valeurs de type date et de caractères sont toujours mis entre un code ' ' ou double code '' ''*
  - *Il est possible d'insérer quelques colonnes et de laisser les autres dans ce cas les autres colonnes sont remplit a NULL ou par la valeur par défaut spécifiée lors de la création de la table*
  - *Les colonnes NOT NULL doivent obligatoirement être remplis.*
  - *Respecter les contraintes PRIMARY KEY, FOREIGN KEY et CHECK*

DML: Langage de manipulation des données

Insertion de données avec INSERT

- **CAS 1** : Insérons une première ligne dans cette table en respectant la syntaxe de base consistant à lister les colonnes avant de soumettre les valeurs:

```
INSERT INTO CLIENTS ( MATRICULE, PRENOM, NOM, VILLE, SEXE)
VALUES ('C001','Edouard','SARR','Mbour','M');
```

- **CAS 2** : Insérons une deuxième ligne dans cette table sans respecter la syntaxe de base :

```
INSERT INTO CLIENTS VALUES ('C002','Agnes','SAGNE','Dakar','F');
```

- CAS 3 : Insertion d'une ligne en omettant les colonnes à valeur NULL

```
INSERT INTO CLIENTS ( MATRICULE, PRENOM, NOM) VALUES ('C003','Amadou Nar','DIOP');
```

- **CAS 4** : Utilisation de FONCTION dans le INSERT INTO

```
INSERT INTO CLIENTS VALUES ('C004', 'Ngone ', UPPER('diop'), 'Mbour','F');
```

-

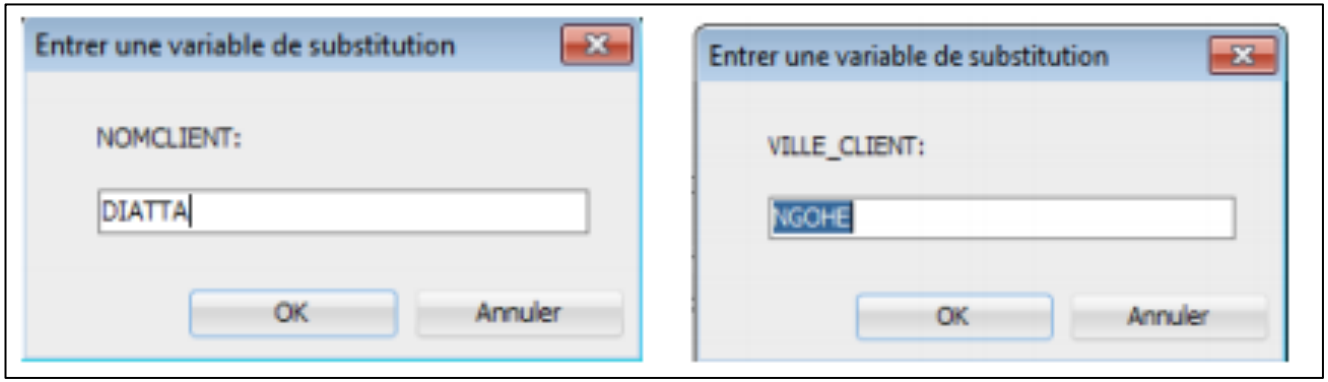
DML: Langage de manipulation des données

Insertion de données avec INSERT

Insertion avec variable de substitution.

- Utilisation de variables de substitution avec ESPARLUETTE (&).
- Il est aussi possible d'utiliser des variables de substitution.

```
INSERT INTO CLIENTS VALUES ('C005','Maxime ','&NomClient','&Ville_Client','M');
```



- NB : &&Val garde en mémoire la première valeur de Val. Elle devienne ainsi une variable Statique.

DML: Langage de manipulation des données

Insertion de données avec INSERT

Insertion multiple

- L'insertion multiple se fait Par copy : Insertion de données suite à une requête SELECT.
- Exemple 1 : Créer une table nommées CLIENTS_COPY avec la même structure que Clients mais sans valeur:

```
Create table CLIENTS_COPY
As
 SELECT * FROM CLIENTS   WHERE 1=2;
```

- Exemple 2 : Créer une table nommées CLIENTS_COPY avec la même structure que Clients mais avec valeur:

```
INSERT  INTO CLIENTS_COPY  AS
 SELECT   *  FROM CLIENTS ;
```

DML: Langage de manipulation des données

Modification de données avec UPDATE

- UPDATE est un verbe de la DML qui permet de modifier des enregistrements dans une table de la base de données.

```
UPDATE   NomTable
SET    NomColonne1= NewValeur1,   NomColonne2=Newvaleur2 , …..  NomColonneN= NewValeurN
WHERE   Condition ;
```

- REGLES :

- La clause WHERE n'est utilisée que pour nous permet spécifier la ou les lignes à modifier. Sans elle, toutes les enregistrements vont subir la modification.
- De préférence le WHERE porte sur la clé primaire pour plus de précision.

DML: Langage de manipulation des données

Modification de données avec UPDATE

- Exemple 1: Modifions le client COO3, sa nouvelle ville est Tivaoune et le sexe M :

```
UPDATE CLIENTS SET VILLE='Tivaoune' , SEXE='M' Where MATRICULE='C003' ;
```

- Exemple 2: Mise à jour multiple

```
UPDATE CLIENTS SET VILLE='Poffine'
Where NOM ='SARR' ;
```

- Exemple 3: Update avec les sous interrogations

```
UPDATE CLIENTS
SET VILLE= ( SELECT VILLE
        FROM CLIENTS
        WHERE MATRICULE='C004'
        )
WHERE VILLE=( SELECT VILLE
        FROM CLIENTS
        WHERE MATRICULE='C003'
        );
```

DML: Langage de manipulation des données

Suppression de données avec DELETE

- Le DELETE permet de supprimer des données dans la base.
- Si une ligne est référée par une clé étrangère dans une autre table alors le DELETE est refusé par ORACLE.
- Ce ci permet de garantir l'intégrité des données.

```
DELETE FROM NomTable
WHERE Conditions ;
```

- Toutes les lignes qui répondent à la clause WHERE seront supprimées.
- Mais, Attention à la violation de contrainte d'intégrité
- On ne peut pas supprimer une ligne qui est référencer dans une autre table sauf dans certaine conditions que nous allons voir par la suite.

DML: Langage de manipulation des données

Suppression de données avec DELETE

- Exemple

Cas 1: Supprimer une ligne

Toute suppression basée sur la clé primaire affecte un et un seul enregistrement de la table.

```
DELETE FROM CLIENTS WHERE Matricule ='C002' ;
```

Cas 2 : Suppression multiple:

```
DELETE FROM CLIENTS WHERE NOM ='sarr' ;
```

Cas 2 : Vider une table:

Sans la clause WHERE, le DELETE vide toute la table.

```
DELETE FROM CLIENTS;
```

Nous pouvons utiliser des fonctions dans le DELETE de même que des sous interrogations.

DML: Langage de manipulation des données

Suppression de données avec DELETE

- Exemple

```
DELETE FROM CLIENTS WHERE Matricule IN ( 'C001', 'C003', 'C005');
```

Supprimer tous les clients portant le même nom que le client de matricule C004.

```
DELETE FROM CLIENTS
 WHERE Nom = ( SELECT Nom FROM Clients WHERE Matricule ='C004' );
```

NB : TRUNCATE permet aussi de supprimer l'ensemble des lignes de la table. Mais n'oublier pas que TRUNCATE appartient à la DDL.

Syntaxe:

```
TRUNCATE TABLE NomTABLE ;
```

DML: Langage de manipulation des données

Synchroniser de données avec MERGE

- Permet de fusionner et synchronisant des données de la base.
- Exemple : Soient les tables suivantes

- CLIENTS

	MATRICULE	PRENOM	NOM	VILLE		SEXE
1	C007	Veronique	TENDENGUE	Poffine	...	F
2	C008	Fatou	Dafe	Mbour	...	F
3	C001	Edouard	SARR	Mbour	...	M
4	C002	Marie Angelique	Senghor	Dakar	...	F
5	C003	Amadou Nar	DIOP	NGOHE	...	(null)
6	C004	Ngone	DIOP	Mbour	...	F
7	C005	Maxime	Diatta	Ngohe	...	M
8	C006	Fatou	FALL	Mbour	...	F
9	C009	Anissa	SAGNE	Dakar	...	F
10	C0010	Ousmane	SENE	Mbour	...	M

- CLIENTS_DAKAR

	MATRICULE	PRENOM	NOM	VILLE		SEXE
1	C002	Marie Angelique	Senghor	Dakar	...	F
2	C009	Anissa	SAGNE	Dakar	...	F

DML: Langage de manipulation des données

Synchroniser de données avec MERGE

- Permet de fusionner et synchronisant des données de la base.
- Atelier : Soient les tables suivantes
  - - Modifions le prénom de C009 dans CLIENTS_DAKAR en mettant AWA

- Avant

	MATRICULE	PRENOM	NOM	VILLE		SEXE
1	C002	Marie Angelique	Senghor	Dakar	...	F
2	C009	Anissa	SAGNE	Dakar	...	F

- Apres

	MATR...	PRENOM	NOM	VILLE		SEXE
1	C002	Marie Angelique	Senghor	Dakar	...	F
2	C009	AWA	SAGNE	Dakar	...	F

DML: Langage de manipulation des données

Synchroniser de données avec MERGE

- Permet de fusionner et synchronisant des données de la base.
- Atelier : Soient les tables suivantes
  - Faisons la fusion avec un MERGE pour Synchroniser les deux tables CLIENTS_DAKAR et CLIENTS

```
MERGE  INTO  CLIENTSDAKAR  cliDak
USING  COPYCLIENTS  cli
On  (cliDak.MATRICULE = cli.MATRICULE)
WHEN  MATCHED  THEN
UPDATE SET
  cliDak.PRENOM= cli.PRENOM,
  cliDak.NOM= cli.NOM,
  cliDak.VILLE= cli.VILLE,
  cliDak.SEXE= cli.SEXE
 WHEN  NOT MATCHED THEN
 INSERT  VALUES(cli.MATRICULE, cli.PRENOM,  cli.NOM,  cli.VILLE, cli.SEXE);
```

DML: Langage de manipulation des données

Synchroniser de données avec MERGE

- Explication

Explication:

MERGE INTO CLIENTSDAKAR cliDak

USING COPYCLIENTS cli On (cliDak.MATRICULE = cli.MATRICULE)

- Fusionner la table CLIENTSDAKAR renommée à cliDak avec COPYCLIENTS renommée cli. LA source c'est COPYCLIENTS et la destination est CLIENTSDAKAR. La condition de fusion est la colonne MATRICULE

WHEN MATCHED THEN

UPDATE SET

cliDak.PRENOM= cli.PRENOM,

cliDak.NOM= cli.NOM,

cliDak.VILLE= cli.VILLE,

cliDak.SEXE= cli.SEXE

- lorsqu'une donnée existe différemment de part et d'autre des deux table WHEN MATCHED , alors faudra les modifier UPDATE SET en utilisant les informations de la table source.

WHEN NOT MATCHED THEN

INSERT VALUES(cli.MATRICULE, cli.PRENOM, cli.NOM, cli.VILLE, cli.SEXE);

- Lorsqu'une donnée figure dans la source et non dans la destination alors faut l'ajouter

DML: Langage de manipulation des données

Recherche de données avec SELECT

- C'est la requête la plus utilisée mais sans doute la plus complexe.
- Elle permet de faire une sélection (avec ou sans tri, avec ou sans ordre, avec ou sans groupement, …) sur des objets notamment les tables et les vues.
- La requête SELECT est composée de 2 parties : les informations à afficher et la ou les tables cibles
- Elle est très puissante dés lors que nous maitrisons son fonctionnement.
- Le SELECT nous donne trois possibilités
  - Projection
  - Sélection
  - Mixte: Projection + Sélection

DML: Langage de manipulation des données

Recherche de données avec SELECT

- **La projection**

MATRICULE	PRENOM	NOM	VILLE		SEXE
C001	Edouard	SARR	Mbour	...	M
C0010	Ousmane	SENE	Mbour	...	M
C002	Marie Angelique	Senghor	Dakar	...	F
C003	Amadou Nar	DIOP	NGOHE	...	(null)
C004	Ngone	DIOP	Mbour	...	F
C005	Maxime	Diatta	Ngohe	...	M
C006	Fatou	FALL	Mbour	...	F
C007	Veronique	TEND...	Poffine	...	F
C008	Fatou	Dafe	Mbour	...	F
C009	Anissa	SAGNE	Dakar	...	F

DML: Langage de manipulation des données

Recherche de données avec SELECT

- **La selection**

MATRICULE	PRENOM	NOM	VILLE		SEXE
C001	Edouard	SARR	Mbour	...	M
C0010	Ousmane	SENE	Mbour	...	M
C002	Marie Angelique	Senghor	Dakar	...	F
C003	Amadou Nar	DIOP	NGOHE	...	(null)
C004	Ngone	DIOP	Mbour	...	F
C005	Maxime	Diatta	Ngohe	...	M
C006	Fatou	FALL	Mbour	...	F
C007	Veronique	TEND...	Poffine	...	F
C008	Fatou	Dafe	Mbour	...	F
C009	Anissa	SAGNE	Dakar	...	F

DML: Langage de manipulation des données

Recherche de données avec SELECT

- **Mixte: Projection + selection**

MATRICULE	PRENOM	NOM	VILLE	SEXE
C001	Edouard	SARR	Mbour ...	M
C0010	Ousmane	SENE	Mbour ...	M
C002	Marie Angelique	Senghor	Dakar ...	F
C003	Amadou Nar	DIOP	NGOHE ...	(null)
C004	Ngone	DIOP	Mbour ...	F
C005	Maxime	Diatta	Ngohe ...	M
C006	Fatou	FALL	Mbour ...	F
C007	Veronique	TEND...	Poffine ...	F
C008	Fatou	Dafe	Mbour ...	F
C009	Anissa	SAGNE	Dakar ...	F

DML: Langage de manipulation des données

Recherche de données avec SELECT

- **Syntaxe générale**

```
SELECT * | DISTINCT | COLONNE1…COLONNE n | Expression | ALIAS , …
FROM TABLE | TABLE2 …
WHERE CONDITION (Operation | sous interrogation)
AND | OR CONDITION (Operation | sous interrogation)
AND …
GROUP BY Criteres
ORDER BY critere ASC / DESC
;
```

DML: Langage de manipulation des données

Recherche de données avec SELECT

- **Syntaxe générale : Explication**
  - ** permet de sélectionner toutes les colonnes*
  - *DISTINCT pour éviter les doublons*
  - *COLONNE 1… COLONNE n liste des colonnes à sélectionner*
  - *Expression pour mettre des Operations*
  - *Alias pour changer les noms des colonnes a l'affichage.*
  - *AND ajouter une condition à vérifier en paire avec celle du WHERE*
  - *OR ajouter une condition à satisferai autre que celle du WHERE*
  - *GROUP BY pour regrouper en fonction d'un critère*
  - *ORDER BY trier suivant un critère (croissant ou décroissant)*

DML: Langage de manipulation des données

Recherche de données avec SELECT

- **Gestion du NULL**
- Null est différent Zero 0 ou espace. 0 est un chiffre et espace est un caractère.
- La valeur NULL absorbe tout opérateur, l'addition, la multiplication, la soustraction, …. sauf la concaténation ||.

- SELECT 10 + NULL from DUAL;

	10+NULL
1	(null)

- SELECT 10 * NULL from DUAL;

10*NULL
(null)

- SELECT 10 / NULL from DUAL;

10/NULL
(null)

-

- SELECT 10 - NULL from DUAL;

10-NULL
(null)

DML: Langage de manipulation des données

Recherche de données avec SELECT

- **Les ALIAS**
- Elle permet de renommer les entrées par un nom plus explicite. Elles sont très utilisées dans les calcules. Alias suit directement le nom de la colonne à remplacer.
- On utilise soit :
  - le mot clé AS
  - En mettant simplement un espace suivit du nom de l'alias juste apres la colonne Exemple : Afficher les prénoms des clients avec comme alias PRENOM_CLIENTS

```
SELECT PRENOM AS PRENOM_CLIENTS   FROM CLIENTS;

Ou

SELECT PRENOM    PRENOM_CLIENTS   FROM CLIENTS;
```

DML: Langage de manipulation des données

Recherche de données avec SELECT

- **La concatenation**
- Elle permet de mettre côte à côte deux chaines de caractères. Pour faire la concaténation on utilise ||

Exemple 1: sélectionner le nom et le prénom concaténé et afficher suivant un alias PRENOM ET NOM CLIENTS.

```
SELECT PRENOM || ' ' || NOM AS "PRENOM ET NOM DU CLIENTS" FROM CLIENTS;
```

- NB: en concaténant un String avec NULL donne le String lui même.
- Dons NULL absorbe toute operateur sauf la concaténation ||.

DML: Langage de manipulation des données

Recherche de données avec SELECT

- **Operateur QUOTE**
- Operateur QUOTE ou q est utilisé pour gérer les apostrophes avec [ ].
- A la place [ ] pour quote nous pouvons utiliser {} ou ( ), ou < >

```
SELECT q'[l'adresse du client]' || prenom || ' ' || nom || ' est ' || ville from CLIENTS;

Ou

SELECT  q' <l'adresse du client> '  || prenom || ' ' || nom || ' est ' || ville from CLIENTS;
```

DML: Langage de manipulation des données

Recherche de données avec SELECT

- **Operateur DISTINCT**
- Il permet de d'enlever les duplicatas sur le résultat.
- Exemple : lister les différents noms des clients.

```
SELECT  DISTINCT (nom) from Clients;
```

	NOM
1	Senghor
2	SAGNE
3	Dafe
4	Diatta
5	SENE
6	TENDENGUE
7	DIOP
8	SARR
9	FALL

DML: Langage de manipulation des données

Recherche de données avec SELECT

- **La clause Where**
- Permet de limiter les résultats de nos requêtes en fonction d'une condition.
- Dans le WHERE nous fixons une condition pour faire une SELECTION.

Exemple : Lister les clients qui habitent à DAKAR.

```
SELECT * FROM CLIENTS where Ville='Mbour';
```

	MATRICULE	PRENOM	NOM	VILLE		SEXE
1	C008	Fatou	Dafe	Mbour	...	F
2	C001	Edouard	SARR	Mbour	...	M
3	C004	Ngone	DIOP	Mbour	...	F
4	C006	Fatou	FALL	Mbour	...	F
5	C0010	Ousmane	SENE	Mbour	...	M

DML: Langage de manipulation des données

Recherche de données avec SELECT

- **La clause Where**
- Permet de limiter les résultats de nos requêtes en fonction d'une condition.
- Dans le WHERE nous fixons une condition pour faire une SELECTION.

REGLES:

- *la clause WHERE suit directement la clause FROM.*
- *Les chaine de caractères et les date doivent toujours être mise en code ' '.*
- *Dans la close WHERE nous ne pouvons pas faire des ALIAS.*
- *Oracle respect la Casse donc oracle tient aussi compte des minuscules et des majuscules.*

DML: Langage de manipulation des données

Recherche de données avec SELECT

- **La clause Where**

Exemple : La requête suivante ne marchera pas car MBOUR en majuscule n'existe pas dans la base.

```
SELECT  *  FROM  CLIENTS where Ville='MBOUR';
```

Cette requête non plus:

```
SELECT  *  FROM  CLIENTS where Ville='MBOUr';
```

- Lister les employés embauchés avant le 10 octobre 2013

```
Select *  from employes where date_embauche< '10/10/2013';
```

NB: Oracle comprend aussi les dates avec le –

```
Select *  from employes where date_embauche< '10-10-2013';
```

DML: Langage de manipulation des données

Recherche de données avec SELECT

- **Les operateurs arithmétique**

Dans ce tableau vous retrouverez les operateurs arithmétiques de base.

OPERATEURS	DESCRIPTIONS
+	addition
-	soustraction
*	multiplication
/	Division

Dans une Operations l'ordre de priorité des operateurs est important.

Exemple 1 : Lister les noms des produits avec le prix fois 100/

```
SELECT nom_produit,prix*100 FROM produits ;
```

DML: Langage de manipulation des données

Recherche de données avec SELECT

- **Les operateurs de comparaison**

=	Egalité
>	Supérieur
<	Inferieur
>=	Supérieur ou égale
<=	Inferieur ou égale
<>	Différent
BETWEEN …. AND	Compris entre deux valeurs
IN (val1, val2…. valN)	Parmi ces valeurs
LIKE	Egalité de chaine de caractères
IS NULL	Est Null

NB : Pour l'operateur différent nous avons trois possibilités:

- ! =
- ^=
- <>

DML: Langage de manipulation des données

Recherche de données avec SELECT

- **Les operateurs de comparaison : Exemple**

- Afficher tous les employés dont le prénom commence par a.

```
SELECT *  FROM employes  WHERE Prenom like 'a%' ;
```

Afficher les employés dont les salaires sont 100000, 200000 ou 300000.

```
SELECT  * FROM employes  WHERE  salaire IN (100000, 200000,300000) ;
```

Afficher tous les employés dont le salaire est comprit entre 250000 et 350000.

```
SELECT  * FROM employes  WHERE  salaire  BETWEEN 250000 AND  350000;
```

- lister les employés qui on comme troisième lettre un i dans leurs prenoms :

```
SELECT  *  FROM employes  WHERE  Prenom  like '__i%' ;
```

DML: Langage de manipulation des données

Recherche de données avec SELECT

- **Les operateurs logique**

NB: les operateurs logique sont au nombre de 3

- AND en français ET
- OR en français OU
- NOT en français NON

NOT IN nous permet de chercher tous ce qui ne figure pas dans la liste.

Exemple :

- Lister les Client de Dakar nommés SARR

```
Select * from clients
Where ville='Mbour'
And nom='SARR';
```

DML: Langage de manipulation des données

Recherche de données avec SELECT

- **ORDER BY**
- Il permet de faire le tri suivant l'ordre croissant ou décroissant

- Afficher les clients par ordre croissant des prénoms

```
SELECT * FROM Clients ORDER BY prenom ASC ;
```

MATRICULE	PRENOM	NOM	VILLE		SEXE
C003	Amadou Nar	DIOP	NGOHE	...	(null)
C009	Anissa	SAGNE	Dakar	...	F
C001	Edouard	SARR	Mbour	...	M
C008	Fatou	Dafe	Mbour	...	F
C006	Fatou	FALL	Mbour	...	F
C002	Marie Angelique	Senghor	Dakar	...	F
C005	Maxime	Diatta	Ngohe	...	M
C004	Ngone	DIOP	Mbour	...	F
C0010	Ousmane	SENE	Mbour	...	M
C007	Veronique	TENDENGUE	Poffine	...	F

DML: Langage de manipulation des données

Recherche de données avec SELECT

- **ORDER BY**
- Il permet de faire le tri suivant l'ordre croissant ou décroissant

- Afficher les clients par ordre décroissant des prénoms

```
SELECT * FROM Clients ORDER BY prenom DESC ;
```

MATRICULE	PRENOM	NOM	VILLE		SEXE
C007	Veronique	TENDENGUE	Poffine	...	F
C0010	Ousmane	SENE	Mbour	...	M
C004	Ngone	DIOP	Mbour	...	F
C005	Maxime	Diatta	Ngohe	...	M
C002	Marie Angelique	Senghor	Dakar	...	F
C006	Fatou	FALL	Mbour	...	F
C008	Fatou	Dafe	Mbour	...	F
C001	Edouard	SARR	Mbour	...	M
C009	Anissa	SAGNE	Dakar	...	F
C003	Amadou Nar	DIOP	NGOHE	...	(null)

DML: Langage de manipulation des données

Recherche de données avec SELECT

- **ORDER BY**
- Il permet de faire le tri suivant l'ordre croissant ou décroissant

Regle:

- *ORDER BY affiche par défaut dans l'ordre ascendant ASC.*
- *ORDER BY est toujours la dernière instruction*
- *ORDER BY accepte les ALIAS.*

```
SELECT prenom AS PRE,Nom,Ville
FROM Clients
ORDER BY PRE ASC ;
```

DML: Langage de manipulation des données

Recherche de données avec SELECT

- **Substitution de variable**
- Nous permet d'utiliser des variables à la place des constantes. On utilise & pour demander une variable.
- Si nous voulons définir une variable STATIC qui ne change pas en cours du programme nous utilisant &&.
- Peut être l'utiliser dans :
  - La condition WHERE
  - Le ORDER BY
  - Les colonnes
  - Les expressions
  - Nom des tables
  - Dans le select

```
SELECT * FROM Clients
WHERE NOM like '&Donner_le_Nom';
```

```
SELECT &nom_Colonne
FROM &Nom_Table;
```

```
SELECT &&nom_Colonne
FROM employes;
```

DML: Langage de manipulation des données

Recherche de données avec SELECT

- **Substitution de variable : DEFINE**
- Une substitution en SQL peut s'accompagner de DEFINE et UNDEFINE
  - DEFINE permet de définir une variable avant la requête
  - UNDEFINE permet de la détruire. Exemple : Nous avons ici trois instruction à exécuter une à une

```
Define  Salaire_Employe=200000

SELECT *  FROM employes   WHERE  SALAIRE = &Salaire_Employe ;
```

```
DEFINE MaVille =Mbour

SELECT * FROM   clients Where  ville = '&MaVille';
```

DML: Langage de manipulation des données

Recherche de données avec SELECT

- **La jointure**
- C'est le fait de sélectionner des informations issues de tables différentes mais liées entre elles.
- Dans une jointure on peut y ajouter une sélection en n'affichant que les lignes désirées avec la clause WHERE.
- Pour qu'une jointure puisse se faire, il est impérative d'avoir une clause de jointure c'est-à-dire une colonne en commun pouvant nous permettre de lié ces deux table.
- Donc elle est retrouvée de part et d'autre et contienne la même information.
- Retenez cependant que la plupart des jointures entre tables s'effectuent en imposant l'égalité des valeurs d'une colonne d'une table à une colonne d'une autre table.

DML: Langage de manipulation des données

Recherche de données avec SELECT

- **La jointure**
- C'est le fait de sélectionner des informations issues de tables différentes mais liées entre elles.
- Dans une jointure on peut y ajouter une sélection en n'affichant que les lignes désirées avec la clause WHERE.
- Pour qu'une jointure puisse se faire, il est impérative d'avoir une clause de jointure c'est-à-dire une colonne en commun pouvant nous permettre de lié ces deux table.
- Donc elle est retrouvée de part et d'autre et contienne la même information.
- Retenez cependant que la plupart des jointures entre tables s'effectuent en imposant l'égalité des valeurs d'une colonne d'une table à une colonne d'une autre table.

DML: Langage de manipulation des données

Recherche de données avec SELECT

- **La jointure**
  - **Exemple**

Table CLIENTS

MATRICULE	PRENOM	NOM	VILLE	SEXE
C007	Veronique	TEND...	Poffine ...	F
C008	Fatou	Dafe	Mbour ...	F
C001	Edouard	SARR	Mbour ...	M
C002	Marie Angelique	Senghor	Dakar ...	F
C003	Amadou Nar	DIOP	NGOHE ...	(null)
C004	Ngone	DIOP	Mbour ...	F
C005	Maxime	Diatta	Ngohe ...	M
C006	Fatou	FALL	Mbour ...	F
C009	Anissa	SAGNE	Dakar ...	F
C0010	Ousmane	SENE	Mbour ...	M

Table VENTES

ID_VENTE	ID_CLIENT	ID_PRODUIT	QUANTITE
V00001	C001	P0005	12
V00002	C003	P0003	2
V00003	C001	P0002	54
V00004	C004	P0001	23
V00005	C001	P0005	12

DML: Langage de manipulation des données

Recherche de données avec SELECT

- **La jointure**
  - **Exemple**
  - En y regardant de plus prêt nous nous apercevons que les deux tables CLIENTS et VENTES sont liées par Id_Client de VENTES et Matricule de CLIENTS qui contiennent la même information.
  - Ainsi, les jointures permettent d'exploiter pleinement le modèle relationnel des tables d'une base de données.
  - Une jointure permet donc de combiner les colonnes de plusieurs tables.

DML: Langage de manipulation des données

Recherche de données avec SELECT

- **La jointure**

Pour faire une telle jointure, Il nous faudra déterminer :

- La Liste des colonnes à afficher dans la clause SELECT
  Exemple : SELECT prenom, nom , quantite
- Les deux tables avec leur ALIAS dans la clause FROM
  Exemple : FROM VENTES V, CLIENTS C
- la clause de jointure dans la clause WHERE en utilisant de préférence les ALIAS ; ici c'est la colonne id_clients de VENTES et celle Matricule de CLIENTS. Il est donc nécessaire d'indiquer au compilateur la provenance de chacune des colonnes et donc d'opérer une distinction entre l'une et l'autre colonne. Ainsi, chaque colonne devra être précédée du nom de la table ou de son Alias, suivi d'un point.
  Exemple : WHERE V.id_client=C.matricule;

DML: Langage de manipulation des données

Recherche de données avec SELECT

- **La jointure**
- Exemple 1 : Nous allons faire l'exemple précédent à savoir lister les prénoms, nom et quantités des ventes effectuées par les clients

```
SELECT   prenom, nom , quantite
From VENTES V, CLIENTS C
Where V.id_client=C.matricule;
```

- Exemple 2 : Faisons la même jointure mais cette fois si nous souhaitons sélectionner que les clients de Dakar ayant effectué une vente.

```
SELECt prenom, nom , quantite
From VENTES V, CLIENTS C
Where V.id_client=C.matricule
AND  C.Ville='Mbour';
```

DCL
Langage de contrôle de données

DCL: Langage de contrôle des données

GRANT & REVOKE

- Tout DBA a besoin de savoir ceux qui accèdent à sa base de données et ceux qu'ils y effectuent surtout lorsqu'il y a perte de données suspecte.
- Pour cela il faudra qu'il fasse un bon contrôle des accès à ses objets.

- Les contrôles d'accès d'une base de données ne sont possibles que si au préalable une bonne gestion des utilisateurs ait été bien élaboré depuis leur création, selon leur profile et de par des privilèges et rôles qui leurs étaient attribué

DCL: Langage de contrôle des données

GRANT & REVOKE

- Un privilège est un droit affecté à un utilisateur sur un objet de la base de données.
- L'Objectif des privilèges est de pouvoir limiter au strict nécessaire ce qu'un utilisateur peut faire sur la base de données.
- Sous Oracle le créateur d'un objet (table, vue …) est le propriétaire et initialement seul lui peut le manipuler à part le DBA.
- Afin qu'un autre utilisateur puisse manipuler ces objets il faut que le propriétaire lui accorde directement des privilèges (ou indirectement avec l'option Grant option). s.

La commande GRANT ajoute un privilège à un user ou un rôle
La commande REVOKE supprime les privilèges

DCL: Langage de contrôle des données

GRANT & REVOKE

- les privilèges système :
  - Ils permettent de modifier la structure de la base en créant ou détruisant des objets. Privilège système: permet aux utilisateurs d'effectuer des opérations particulières dans la BD.
  - Ces opérations comprennent la création, la suppression et la modification de tables, de vues, de procédures etc.
  - Exemple de privilèges système sous ORACLE :
    - CREATE USER
    - DROP USER
    - CREATE ANY TABLE
    - DROP ANY TABLE
    - BACKUP ANY TABLE

DCL: Langage de contrôle des données

GRANT & REVOKE

- les privilèges système
  - Exemple sous Oracle

Syntaxe d'affectation de privilèges système :

```
GRANT {priv_système|rôle} [, {priv_système|rôle} ]
TO {username|rôle| PUBLIC} [, {username|rôle| PUBLIC} ]
[WITH ADMIN OPTION];
```

PUBLIC veut que je le donne à tout le monde

WITH ADMIN OPTION: permet au bénéficiaire d'accorder à son tour le privilège à d'autres users

Exemple :

```
GRANT Connect TO ousmane ;
GRANT CREATE TABLE TO ousmane WITH ADMIN OPTION;
```

DCL: Langage de contrôle des données

GRANT & REVOKE

- les privilèges objet:
- Ils permettent de manipuler des objets existant : consultation et modification d'une table ou vue, exécution d'un sous-programme stocké
- Ils permettent de consulter ou modifier l'état d'un objet particulier, ou de l'exécuter, mais sans pouvoir le détruire ou en créer de nouveaux (ces opérations correspondent à des privilèges système).
- Chacun de ces privilèges fait référence à un objet particulier de la base :

DCL: Langage de contrôle des données

GRANT & REVOKE

- les privilèges objet:
  - Exemple sous Oracle

```
GRANT {priv_objet [(liste_colonne)] [, priv_objet [(liste_colonne)]] | ALL [PRIVILEGES]}
ON [schéma.]objet
TO {username|rôle| PUBLIC} [, {username|rôle| PUBLIC} ]
[WITH GRANT OPTION];
```

WITH GRANT OPTION: permet au bénéficiaire d'accorder à son tour les privilèges sur l'objet à d'autres users ou rôles

Exemple :

```
GRANT UPDATE (prenom, nom, salaire) ON Employes TO ousmane;
GRANT INSERT On Employes TO rose;
GRANT DELETE ON Employes TO amadou;
```

DCL: Langage de contrôle des données

GRANT & REVOKE

- les privilèges système
  - Exemple sous Oracle

La commande REVOKE supprime les privilèges

Syntaxe:

```
REVOKE {priv_système|rôle} [, {priv_système|rôle} ]
FROM {username|rôle| PUBLIC} [, {username|rôle| PUBLIC} ]
```

Exemple :

```
Revoke create table from ousmane;
```

```
REVOKE { priv_objet [, priv_objet] | ALL [PRIVILEGES]}
ON [schema.]objet
FROM {username| rôle|PUBLIC} [, {username| rôle|PUBLIC} ]
```

SQL AVANCE 1

LES FONCTIONS EN SQL

SQL Avancé : Les Fonctions SQL

Les fonctions en SQL

- Une fonction est une méthode des valeurs en entré et en sortie.
- Très utilisées et s'avèrent, très pratique et permet Economiser du temps et du code. Il existe deux types de fonction en SQL :
  - Les fonctions implicites : proposes par le système. Une fonction implicite peut être :
    - **monoligne** prend en entrée une ligne et retourne une ligne comme résultat.
    - **multiligne** s'appui en entrée comme en sortie sur plusieurs lignes.
  - Les fonctions explicites : éditées par l'utilisateur lui-même.

SQL Avancé : Les Fonctions SQL

Les fonctions monolignes

- Ce sont des fonctions qui retournent une et une seule ligne. Elles peuvent être :
  - Les fonctions sur les caractères
  - Les fonctions sur les numériques
  - Les fonctions sur les dates
  - Les fonctions de conversion
  - Les fonctions généralistes

```
FUNCTION_NAME (arg1, arg2,....)
```

- Elles prennent en argument des variables, des constantes, des noms de colonnes et des expressions

SQL Avancé : Les Fonctions SQL

Les fonctions monolignes

- Ce sont des fonctions qui retournent une et une seule ligne. Elles peuvent être :
  - Les fonctions sur les caractères
  - Les fonctions sur les numériques
  - Les fonctions sur les dates
  - Les fonctions de conversion
  - Les fonctions généralistes

```
FUNCTION_NAME (arg1, arg2,....)
```

- Elles prennent en argument des variables, des constantes, des noms de colonnes et des expressions

SQL Avancé : Les Fonctions SQL

Les fonctions monolignes : Fonctions de caractères

- LOWER : Minuscule
- UPPER: Majuscule
- INITCAP : mettant les premiers caractères en MAJUSCULE.

```
SELECT UPPER (prenom), LOWER (nom), INITCAP(Ville) FROM CLIENTS;
```

- CONCAT: Concaténation
- SUBSTR: découpe une expression en une position bien déterminée et avec un nombre définie

```
SUBSTR (expression, Position, Nombre_Caractere)
```

- LENGTH: nombre de caractères

SQL Avancé : Les Fonctions SQL

Les fonctions monolignes : Fonctions de caractères

- INSTR : Elle cherche la position d'une expression dans une autre expression

```
INSTR (expression, 'W')
```

- LPAD: Remplissage vers la gauche avec un nombre de caractères spécifié en paramètre.
- RPAD: Remplissage vers la gauche avec un nombre de caractères spécifié en paramètre.

```
LPAD (expression, Nombre, 'caractère')
```

- TRIM: ôte un caractère de toute autre expression commençant par ce premier.

```
SELECT TRIM ('A'   FROM   prenom) from Clients;
```

- REPLACE : Remplace permet de remplacer une expression par une autre dans une chaine de caractères

```
SELECT REPLACE (prenom, 'a', 'A') from Clients;
```

SQL Avancé : Les Fonctions SQL

Les fonctions monolignes : Fonctions numériques

- ROUND: Round permet d'arrondir une valeur numérique avec un nombre de valeur bien déterminé après la virgule.

```
SELECT ROUND (201774.762667, 2);
```

- TRUNC pour couper une chaine a partir d'un certain niveau.

```
SELECT TRUNC (201774.762667, 3)
```

- MOD c'est-à-dire MODULO est le reste de la division entière entre deux nombres

```
SELECT MOD (20, 6)
```

SQL Avancé : Les Fonctions SQL

Les fonctions monolignes : Fonctions de Date

- Ceux sont des fonctions qui manipule les dates. La date est definie en SIECLE, ANNEE, MOIS, JOUR, HEURE, MINIUTE et SECONDE
- Le format par défaut d'affichage est soit JJ/MM/RR ou JJ/MM/YY où JJ est le jour, MM le mois et YY ou RR l'année.
- Operations sur les dates:
  - DATE + Nombre= DATE à la quel on ajoute Nombre de jour
  - DATE - Nombre= DATE à la quel on enlève Nombre de jour
  - DATE + DATE= Nombre de jour
  - DATE + Nombre /24= DATE à la quel on ajoute Nombre d'heures

SQL Avancé : Les Fonctions SQL

Les fonctions monolignes : Fonctions de Date

- MONTHS_BETWEEN

```
MONTHS_BETWEEN (DATE1, NombreMois)
```

- ADD_MONTHS : Ajouter un nombre de mois

```
SELECT prenom, nom, ADD_MONTHS
(date_embauche,5) As Duree_travail FROM
employes;
```

- NEXT_DAY: Cette fonction donne la date du prochain jour d'une date
- LAST DAY

```
LAST_DAY (Date)
```

```
NEXT_DAY (DATE)
```

SQL Avancé : Les Fonctions SQL

Les fonctions monolignes : Fonctions de conversion

- Elles permettent de convertir une donnée d'un type à un autre. Les plus celebres sont : To_Char, To_Number, To_Date
- NB: la différence entre CHAR et VARCHAR réside dans le mode de reservation en mémoire.
- Avec CHAR la déclaration est STATIC alors que pour VARCAHR elle est DYNAMIQUE.
  Exemple :
- NOM CHAR(55) , dans ce cas, meme pour le nom SARR qui est de 4 caractères, il occupera un espace pour 55 caractères.
- Alors que pour NOM VARCHAR (55), le nom SARR n'occupera qu'une place de 4 caractères.

SQL Avancé : Les Fonctions SQL

Les fonctions monolignes : Fonctions de conversion

- **TO_CHAR**
- Elle transforme une DATE ou un NUMERIQUE en CHAINE DE CARACTERES.

```
TO_CHAR (DATE, 'format model');
```

- Exemple: Afficher le mois et l'année d'embauche de chaque employé dans le format 11/98

```
SELECT prenom, nom, TO_CHAR (date_embauche,'MM/YY') FROM employes;
```

- lister les employés embauches avant 2013

```
SELECT prenom, nom, TO_CHAR (date_embauche,'YYYY') FROM employes
WHERE date_embauche < '01/01/2013';
```

SQL Avancé : Les Fonctions SQL

Les fonctions monolignes : Fonctions de conversion

- **TO_NUMBER**
- Permet de transformer une chaine en Numérique.

```
TO_NUMBER ( CHAR) ;
```

- **TO_DATE**
- Permet de transformer une chaine en Date.
- NB: Il n'existe pas de fonction de conversion de passage directe entre DATE et NUMBER pour cela il faut passer obligatoirement par CHAR.

```
TO_DATE (chaine)
```

SQL Avancé : Les Fonctions SQL

Les fonctions monolignes : fonctions générales

- Elles sont génériques dont s'applique sur les DATE, les CHAR et les NUMBER. Elles sont :
- **NVL**: permet de comparer une valeur avec la valeur NULL, c'est a dire de tester si la valeur est nulle ou pas. Si exp1 est Null retourner la VALEUR sinon retourner EXPRESSION.

```
NVL (Expression, Valeur)
```

```
SELECT prenom, nom, NVL(sexe, 'Sexe Non Définie') from clients;
```

- **NVL2**: Elle fait la même chose que NVL mais autrement. Si EXPRESSION = NULL alors retourner VAL3 Sinon retourner VAL2

```
SELECT prenom, nom, NVL2(sexe, 'Sexe définie', 'sexe non définie') from clients;
```

SQL Avancé : Les Fonctions SQL

Les fonctions monolignes : fonctions générales

- **NULLIF** : NULLIF compare deux expressions si elles sont égales, elle retourne NULL sinon elle retourne la première expression.

```
NULLIF (expression 1, expression 2)
```

- **COALESCE**: Elle affiche la première expression NOT NULL rencontrée.

```
COALESCE (exp1, exp2, exp3, …. expN)
```

SQL Avancé : Les Fonctions SQL

Les fonctions monolignes : fonctions de condition

- **CASE** : C'est SELON en français. A la place de If THEN ELSE on peut utiliser CASE WHEN END.

```
SELECT prenom, nom,

CASE ville WHEN 'Poffine' THEN 'Region de Fatick'

        WHEN 'Ngohe'  THEN 'Region de Fatick'

        WHEN 'NGohe'  THEN 'Region de Fatick'

        WHEN 'Dakar'  THEN 'Region de Dakar'

        WHEN 'Mbour'  THEN 'Region de THIES'

        Else 'Region non connue'

END

 AS  "Les regions"   FROM CLIENTS;
```

SQL Avancé : Les Fonctions SQL

Les fonctions multilignes :

- Ce sont des fonctions qui s'opèrent sur plusieurs lignes et retourne une seule ligne.

AVG	**retourne la moyenne d'une série de valeurs**
MIN	retourne le minimum d'une série de valeurs
MAX	retourne le maximum d'une série de valeurs
SUM	retourne la somme d'une série de valeurs
COUNT	retourne nombre d'éléments d'une série de valeurs
DISTINCT	retourne les valeurs distinctes d'une série de valeurs
GROUP BY	Grouper par
HAVING	permet d'exclure des valeurs ne remplissant pas une certaine condition group by
STDDEV	Retourne l'écart type d'une série de valeurs

SQL Avancé : Les Fonctions SQL

Les fonctions multilignes :

- Dans un SELECT devant contenir un GROUP BY, toutes les colonnes spécifiées dans la close SELECT et non spécifiées avec des fonctions de Groupe, doivent obligatoirement se retrouver au niveau de la close GROUP BY sinon le système retourne une erreur.
- Dans GROUP BY nous pouvons aussi retrouver des colonnes ne figurant pas sur le SELECT.
- le GROUP BY de même que WHERE accepte les Alias contrairement à ORDER BY
- Dans le WHERE nous ne pouvons pas avoir de fonction de groupe
- les fonctions de group sont toujours utiliser avec group BY dés l'instant qu'il existe dans le select une colonne libre sans fonction de groupe.

SQL AVANCE 2

LES SOUS-INTERROGATIONS

SQL Avancé : Les sous interrogations

- Sous interrogation est une requête incluse dans une autre requête.
- Son résultat constitue une entrée pour la requête principale. Les sous interrogations sont de deux type :
  - Mono lignes font appelle aux operateurs de comparaisons mono lignes < <= > >= = <>
  - multi lignes font appelle aux operateurs de comparaisons multi lignes
    - IN
    - ANY
    - ALL

SQL Avancé : Les sous interrogations

SOUS INTERROGATION AVEC CREATE TABLE

- Il est possible de créer une Table avec les Sous Interrogations
- Syntaxe:
  - CREATE TABLE T2 AS (SELECT *|Liste des colonnes FROM NomTable WHERE … AND/OR);
- SYNTAXE:
  - Créer une table copy
    - CREATE TABLE EMPLOYES_COPY AS (SELECT * FROM EMPLOYES);
  - Créer une table Copy avec des restrictions sur les ligne
    - CREATE TABLE EMPLOYES_COPY AS (SELECT * FROM EMPLOYES WHERE ADRESSE LIKE 'MBOUR');
  - Créer Table Copy mais sans enregistrements
    - CREATE TABLE EMPLOYES_COPY AS (SELECT * FROM EMPLOYES WHERE 1=2);

SQL Avancé : Les sous interrogations

SOUS INTERROGATION AVEC UPDATE

- Nous pouvons aussi utiliser les sous interrogation dans les UPDATE. La sous interrogation peut être au niveau du SET ou au niveau du WHERE.

```
UPDATE    NomTable
SET colonne1 = (SELECT column
                FROM table
                WHERE condition),
    colonne2 =  (SELECT column
                FROM table
                WHERE condition)
WHERE  condition ;
```

```
UPDATE   CLIENTS
SET VILLE= (SELECT VILLE
        FROM CLIENTS
        WHERE MATRICULE='C004'
        )
WHERE VILLE= (SELECT VILLE
        FROM CLIENTS
        WHERE MATRICULE='C003'
```

SQL Avancé : Les sous interrogations

SOUS INTERROGATION AVEC DELETE

- Nous pouvons utiliser des fonctions dans le DELETE de même que des sous interrogations.

```
DELETE FROM CLIENTS
 WHERE   Nom = (SELECT Nom
                      FROM Clients
                       WHERE Matricule ='C004'
                      );
```

SQL Avancé : Les sous interrogations

SOUS INTERROGATION AVEC SELECT

- Le traitement d'une sous interrogation se fait en exécutant en premier lieu la Sous requête avant la Requête principale:
- *Règles*
  - *la sous interrogation s'exécute en première lieu*
  - *elle est toujours mise entre parenthèse.*
  - *Toujours décaler vers la droite pour une bonne lisibilité*
  - *La valeur retourner doit être de même structure (Type) que celle devant la recevoir.*
  - *Utilise les operateurs mono lignes comme = < <= > >= … pour avoir une seule valeur de retour*

SQL Avancé : Les sous interrogations

SOUS INTERROGATION AVEC SELECT

- Exemple 1: lister les employés embauchés avant l'employé Ngor.

```
SELECT Prenom, Nom, DATE_EMBAUCHE

FROM Employes

WHERE DATE_EMBAUCHE < (

        SELECT DATE_EMBAUCHE

        FROM Employes

        Where Prenom= 'Ngor');
```

SQL Avancé : Les sous interrogations

SOUS INTERROGATION AVEC SELECT

- Exemple 2: lister les employés de la même ville que l'employé C008 et de même sexe que C009.

```
SELECT Prenom, Nom, Ville, Sexe
FROM Clients
WHERE Ville = (SELECT ville
        FROM Clients
        Where Matricule='C008')
AND sexe = (SELECT sexe
        FROM Clients
        Where Matricule='C009');
```

SQL Avancé : Les sous interrogations

SOUS INTERROGATION AVEC SELECT

- Exemple 3: Lister les employés ayant un salaire supérieur à celui de NGOR
- Revient à lister CHAQUE (ANY) les employés SAUF NGOR dont le salaire est supérieur a celui de NGOR
  - > ALL compare à la valeur minimale de la liste
  - < ALL compare à la valeur maximale de la liste
  - NOT IN : ne figurant pas dans

```
SELECT Prenom, Nom, Salaire
FROM Employes
WHERE salaire > ANY (SELECT salaire
                     FROM employes
                     Where prenom='Ngor')
AND Prenom <> 'Ngor';
```

SQL Avancé : Les sous interrogations

SOUS INTERROGATION AVEC SELECT

- On peut utiliser dans les sous interrogations des fonctions de groupe :
  - AVG
  - MIN,
  - MAX,
  - COUNT,
  - STRDEV
  - …

```
SELECT prenom, nom, salaire
FROM Employes
WHERE Salaire > (Select AVG (salaire)
                 From Employes);
```

SQL Avancé : Les sous interrogations

SOUS INTERROGATION AVEC INSERT

- Lors de l'insertion avec une sous interrogation
  - VALUES est omise.
  - Autant de colonnes spécifiées dans le INSERT que dans le SELECT
  - Les types doivent être identique respectivement

```
INSERT INTO LES_CLIENTS (
                SELECT *
                 FROM CLIENTS
) ;
```

SQL Avancé : Les sous interrogations

SOUS INTERROGATION AVEC INSERT ALL

- **INSERTIONS MULTIPLE DANS DES TABLES MULTIPLES**
- Les TABLE représentent les cibles ou tables destinataires.
- Utiliser le plus dans les bases de données décisionnelles pour faire la répartition dans les entrepôts de données.
- On utilise INSERT ... SELECT pour insérer une ou plusieurs lignes dans une ou plusieurs tables.
- On utilise les IF THEN pour faire la les conditions.

SQL Avancé : Les sous interrogations

SOUS INTERROGATION AVEC INSERT ALL

- **INSERTIONS MULTIPLE DANS DES CIBLES MULTIPLES**
- Syntaxe:

```
INSERT ALL
INTO TABLE1 VALUES (Col1, Col2)
INTO TABLE2 VALUES (Col3, Col4)
INTO TABLE3 VALUES (Col1, Col3)
SELECT Col1, Col2, Col3, Col4, Col5
FROM   TableSource
WHERE Condition;
```

SQL Avancé : Les sous interrogations

SOUS INTERROGATION AVEC INSERT ALL

- **INSERTIONS MULTIPLE DANS DES CIBLES MULTIPLES**
- Exemple: Segmenter la table CLIENTS en deux autres tables CLI1 et CLI2

```
INSERT ALL
INTO CLI1 VALUES (Matricule, prenom, nom)
INTO CLI2 VALUES (Matricule, ville)
SELECT Matricule, prenom, nom, ville
FROM  CLIENTS ;
```

Références

Références

- Date, C. J., & Darwen, H. (1993). *A guide to SQL standard* (Vol. 3). Reading: Addison-Wesley.
- Gabillaud, J. (2009). *Oracle 11g: SQL, PL/SQL, SQL* Plus*. Editions ENI.
- Brouard, F., Bruchez, R., & Soutou, C. (2012). *SQL*. Pearson Education France.
- Audibert, L. (2007). Base de Données et langage SQL. *Developpez. com.[En ligne]. Disponible sur: http://laurent-audibert. developpez. com/Cours-BD/.[Consulté le: 06-août-2018]*.
- Site Oracle et PL/SQL : http://www.techonthenet.com/oracle/
- Référence PL/SQL Oracle : http://download.oracle.com/docs/cd/B19306 01/appdev.102/b14261/toc.htm
- Cours 5 : PL/SQL http://deptinfo.unice.fr/~grin/messupports/plsql.pdf http://sheikyerbouti.developpez.com/pl_sql/
- Anne Vilnat , Cours 4 : PL/SQL : ou comment faire plus avec ORACLE 2_eme partie
- Alexandre Mesle Introduction au PL/SQL Oracle, 2011

FIN PARTIE 2

Bases de données relationnelles et SGBDR

Prof Edouard Ngor SARR
Enseignant-chercheur en Informatique
Université Assane SECK de Ziguinchor (UASZ)
Email : Edouard-ngor.sarr@univ-zig.sn
Site : www.edouardsarr.com
Oct 2025