

Projet : Cours - Base de données

Etude et mise en œuvre pratique : Projet BookFlow (Gestion de Librairie)

Présenté par :
Yasser Sidane

Encadrement :
Sous la direction de : M.Lakhdissi

GRP-3B

Sommaire

Chapitre 1 : Modélisation des Données	7.
Chapitre 2 : SQL Avancé	14.
Chapitre 3 : Optimisation BDD	27.
Chapitre 4 : NoSQL	41.
Chapitre 5 : Transactions ACID	55.
Chapitre 6 : Sécurité BDD	68.
Chapitre 7 : Data Warehouse	79.
Chapitre 8 : Projet Intégrateur	88.

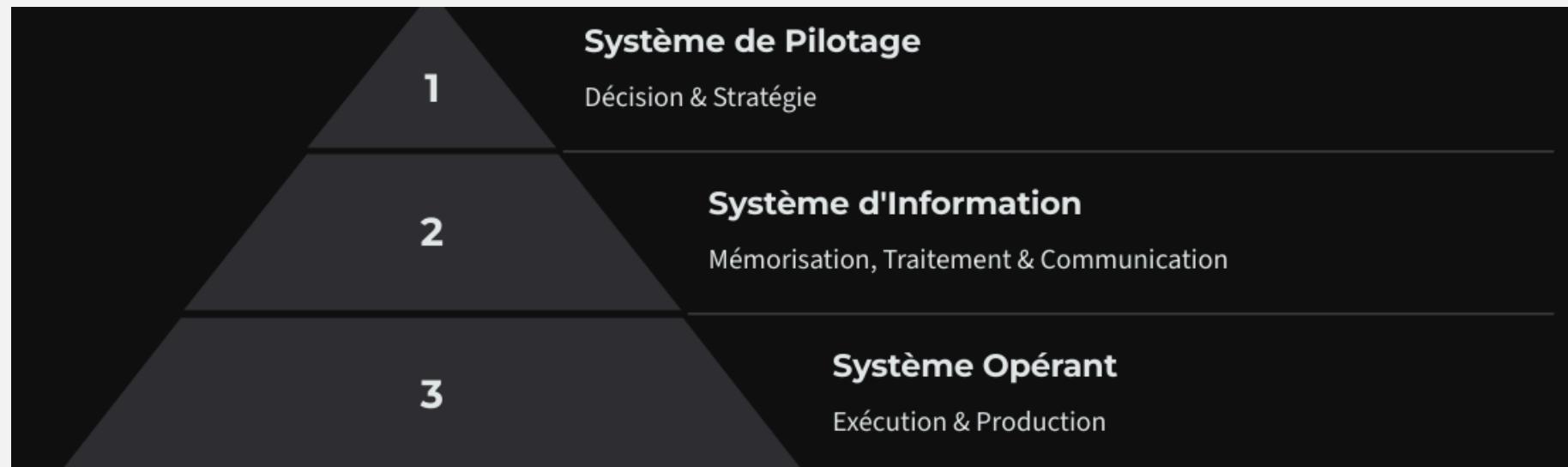
Introduction Générale

1. Le Contexte : La donnée comme actif stratégique Dans l'ère de la transformation digitale, la donnée n'est plus un simple sous-produit de l'activité, mais le cœur battant de l'entreprise. Maîtriser les Systèmes d'Information, c'est être capable de structurer, sécuriser et exploiter ce capital pour créer de la valeur.
2. L'Ambition de ce Dossier Ce travail n'est pas une simple compilation de commandes SQL. C'est une exploration complète de la chaîne de valeur de l'information :
 - Concevoir des architectures robustes et évolutives.
 - Garantir l'intégrité et la haute disponibilité des données (Concepts ACID).
 - Protéger les accès sensibles (Sécurité & Administration).
 - Transformer la donnée brute en intelligence stratégique (Business Intelligence).
3. La Méthodologie L'approche adoptée ici allie une rigueur théorique stricte (concepts fondamentaux) à une mise en œuvre concrète via le Projet Intégrateur, prouvant ainsi que la théorie des bases de données est le socle indispensable du développement logiciel moderne.

3.

Introduction & Définitions

- Donnée vs Information : La donnée est un fait brut. L'information est une donnée interprétée et utile pour la décision.
- Définition B.D. : "Un grand ensemble d'informations structurées, mémorisées sur un support permanent, pour satisfaire plusieurs utilisateurs simultanément."
- Définition SGBD : Logiciel système servant d'interface entre les utilisateurs, les programmes et la base de données.



Limites de l'Approche Fichiers (SGF)

- Redondance des données : L'information est dupliquée à plusieurs endroits (Gaspillage d'espace).
- Incohérence (Anomalie de mise à jour) : Si on modifie une adresse dans un fichier, elle reste inchangée dans les autres.
- Dépendance Données/Programmes : Toute modification de la structure du fichier oblige à réécrire le code de l'application.

5.

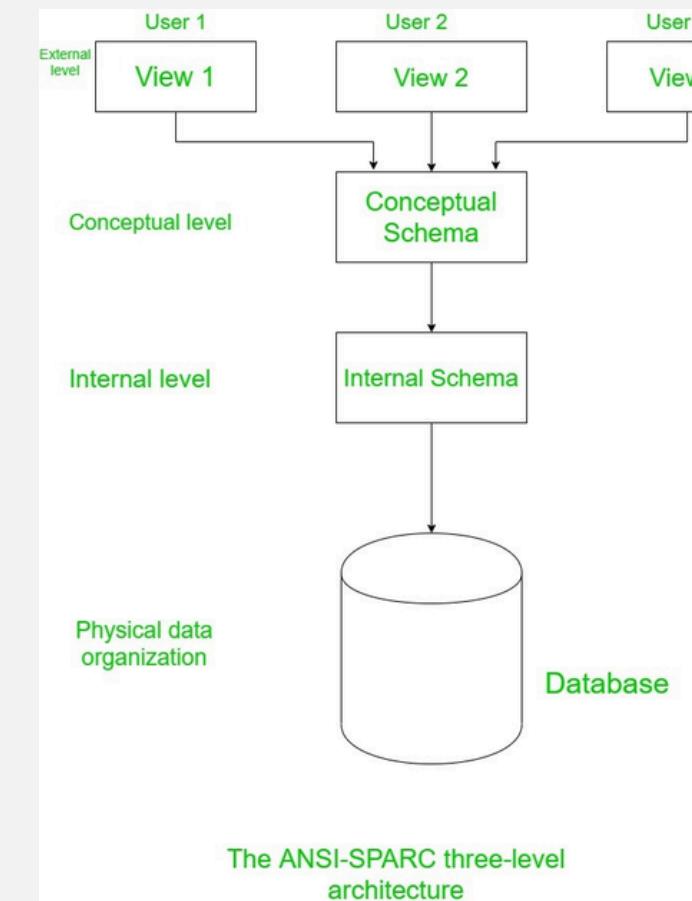
Modifications anarchiques

noCompte	nomClient	adresseClient	noTéléphone	dateOuverture	solde
100	Hugh Paycheck	Ottawa	(999)999-9999	5/05/1999	1000.00
200	Dollard Cash	Montréal	(888)888-8888	10/10/1999	2000.00
300	Hugh Paycheck	Québec	(555)555-5555	10/10/1999	1000.00
400	Ye San Le Su	Montréal	(777)777-7777	20/7/2000	5.00
600	Ye San Le Su	Montréal	(777)777-7777	15/10/2000	10.00

noPrêt	nomClient	adresse Client	noTéléphone	montant Prêt	dateDébut	taux Intérêt	fréquence Paiement
1000	Hugh Paycheck	Alma	(444)444-4444	10000.00	10/6/2000	10	12
2000	Ye San Le Su	Montréal	(777)777-7777	20000.00	20/7/2000	12	52
3000	Hugh Paycheck	Ottawa	(999)999-9999	5000.00	15/8/2000	12	12

L'Architecture ANSI-SPARC

- Pour garantir l'indépendance entre les données et les programmes, l'architecture d'un SGBD est découpée en 3 niveaux :
- 1. Niveau Externe (Vues) : Ce que voit l'utilisateur final (ex: Interface web, formulaires).
- 2. Niveau Conceptuel (Logique) : La structure globale des données, indépendante du matériel (C'est ici qu'on fait le MCD).
- 3. Niveau Interne (Physique) : Comment les données sont réellement stockées sur le disque (Fichiers, Index, Pointeurs).

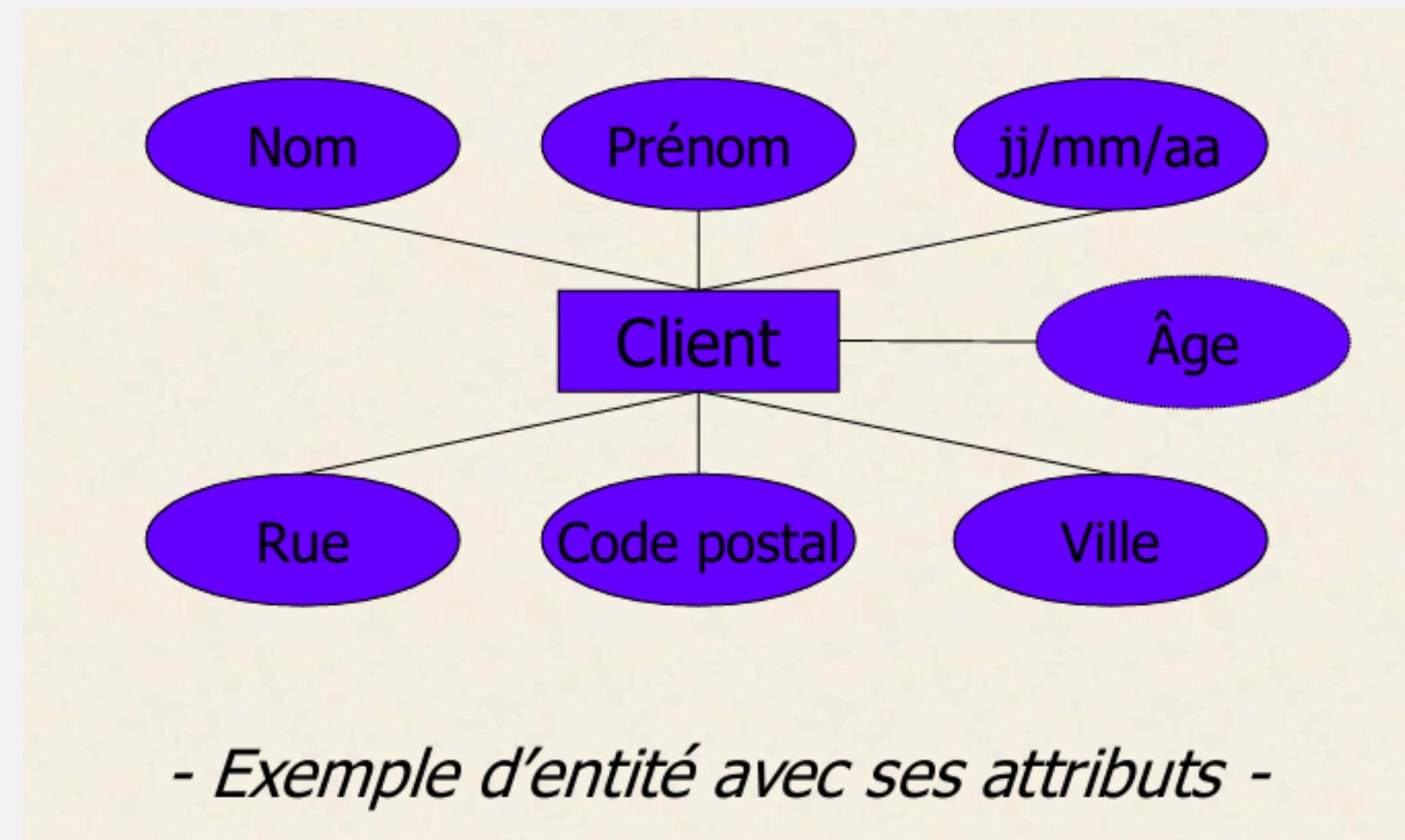


Chapitre 1 : Modélisation des Données

*Conception Conceptuelle
avec le Formalisme Merise*

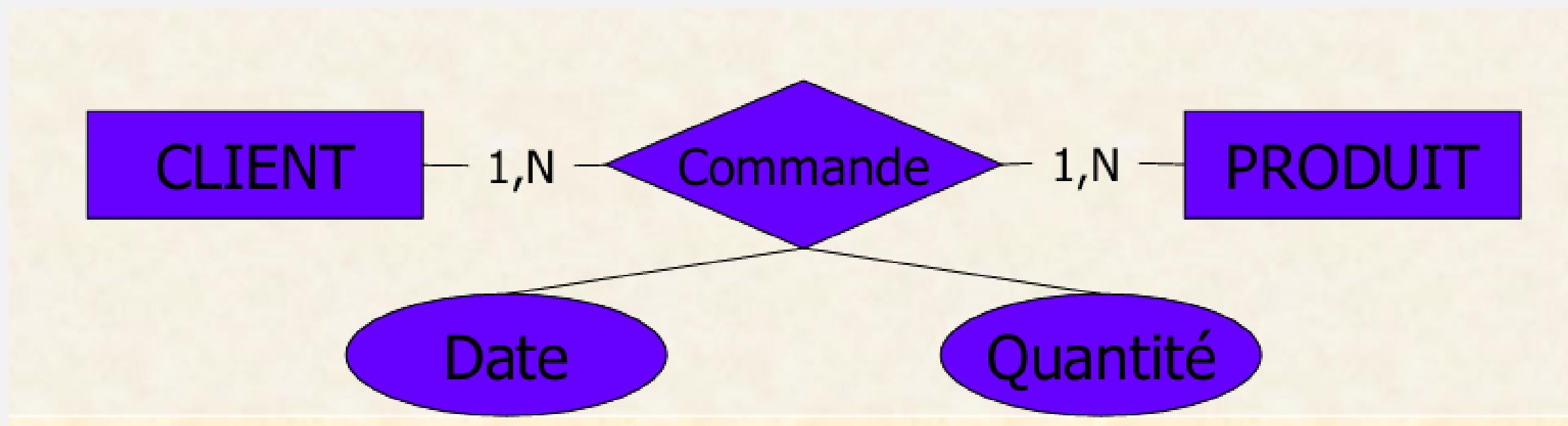
Concepts Fondamentaux (MCD)

- Entité : Représentation d'un objet de gestion (Ex : Client, Produit).
- Attribut : Propriété décrivant l'entité (Ex : Nom, Prix).
- Identifiant : Attribut unique (souligné) qui distingue chaque occurrence.



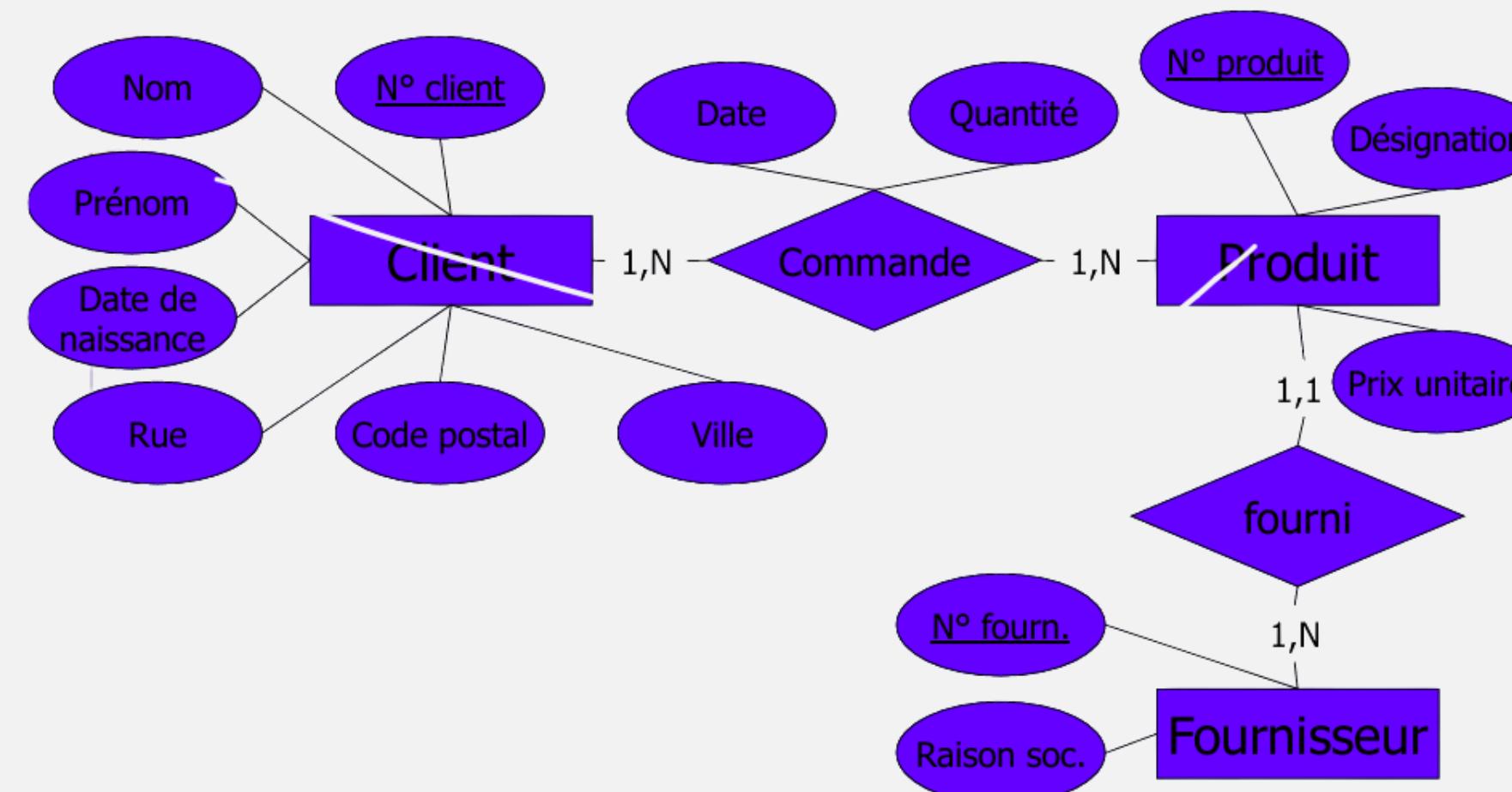
Associations & Cardinalités

- L'Association (Relation) : Lien sémantique (verbe) entre deux entités ou plus (Ex : Un client passe une commande).
- La Cardinalité (Min, Max) : Définit les règles de gestion précises.
 1. (0, 1) ou (1, 1) : Relation unique ("Au plus un" ou "Exactement un").
 2. (0, n) ou (1, n) : Relation multiple ("Aucun ou plusieurs" ou "Au moins un").



Méthodologie de Conception (MCD)

- La construction d'un modèle fiable suit 4 étapes chronologiques :
 1. Dictionnaire des Données : Recenser toutes les informations (ex: Nom, Prix, Date).
 2. Identification des Entités : Regrouper les données par "objets" (Clients, Produits).
 3. Définition des Associations : Établir les verbes qui relient les entités.
 4. Attribution des Cardinalités : Définir les règles de gestion (Min, Max).



Contraintes du Modèle Relationnel

- Clé Primaire (PK) : Ensemble minimal d'attributs identifiant de façon unique un tuple (ligne). Ne peut jamais être nulle (NOT NULL).
- Clé Étrangère (FK) : Attribut qui fait référence à une Clé Primaire d'une autre table. Elle garantit l'Intégrité Référentielle (ex: Impossible de créer une commande pour un client qui n'existe pas).

- Notation
 - Clef primaire
 - *Clef secondaire*
 - Ex. PRODUIT(N°produit, désignation,PU,*N°fournisseur*)
- Contraintes de domaine
 - PU > 0 et PU ≤ 10000

Garantir l'Intégrité : La Normalisation (1NF, 2NF, 3NF)

Objectif : Éliminer la redondance et les anomalies de mise à jour.

- Les 3 Règles d'Or :
 1. 1ère Forme Normale (1NF) : Atomicité des valeurs (une seule donnée par case).
 2. 2ème Forme Normale (2NF) : Pas de dépendance partielle (tout attribut dépend de la totalité de la clé primaire).
 3. 3ème Forme Normale (3NF) : Pas de dépendance transitive (un attribut ne doit pas dépendre d'un autre attribut non-clé).
- Bilan : Un schéma en 3NF est le standard pour une base de données transactionnelle saine.

Passage du Conceptuel (MCD) au Logique (MLD)

- Règle 1 (Entité -> Table) : Chaque entité devient une table SQL. L'identifiant devient la Clé Primaire (PK).
- Règle 2 (Relation 1,n) : La clé du "Père" migre vers le "Fils" et devient une Clé Étrangère (FK).
- Règle 3 (Relation n,m) : L'association devient une Table de Jointure contenant les deux clés primaires (Clé Composite).

- Schéma relationnel complet
 - CLIENT(N°client,nom,prénom,Date de naissance,Rue,CP,Ville)
 - PRODUIT(N°produit,désignation,PU,N°fournisseur)
 - FOURNISSEUR(N°fournisseur,raison sociale)
 - COMMANDE(N°client,N°produit,Date,Quantité)

Chapitre 2 : SQL Avancé

Agrégations, Regroupements et Sous- Requêtes

Langage de Définition de Données (LDD)

- Rôle : Permet de créer, modifier et supprimer les objets de la BDD.
- Commandes Clés :
- CREATE TABLE : Définit la structure (Colonnes, Types).
- ALTER TABLE : Modifie la structure existante.
- DROP TABLE : Supprime définitivement la table.
- Contraintes : Définition des clés (PRIMARY KEY) et des règles (NOT NULL, CHECK).

```
1 • CREATE TABLE Client (
2   IDClient    INT PRIMARY KEY,
3   Nom         VARCHAR(50) NOT NULL,
4   Prenom       VARCHAR(50) NOT NULL,
5   Ville        VARCHAR(50)
6 ) ;
```

Évolution de la Structure (ALTER & DROP)

- Une base de données n'est pas figée. Le LDD permet de la faire évoluer sans perdre les données existantes.
- ALTER TABLE : Permet d'ajouter (ADD), modifier (MODIFY) ou supprimer (DROP COLUMN) une colonne.
- DROP TABLE : Supprime la structure complète et toutes ses données (Irréversible).

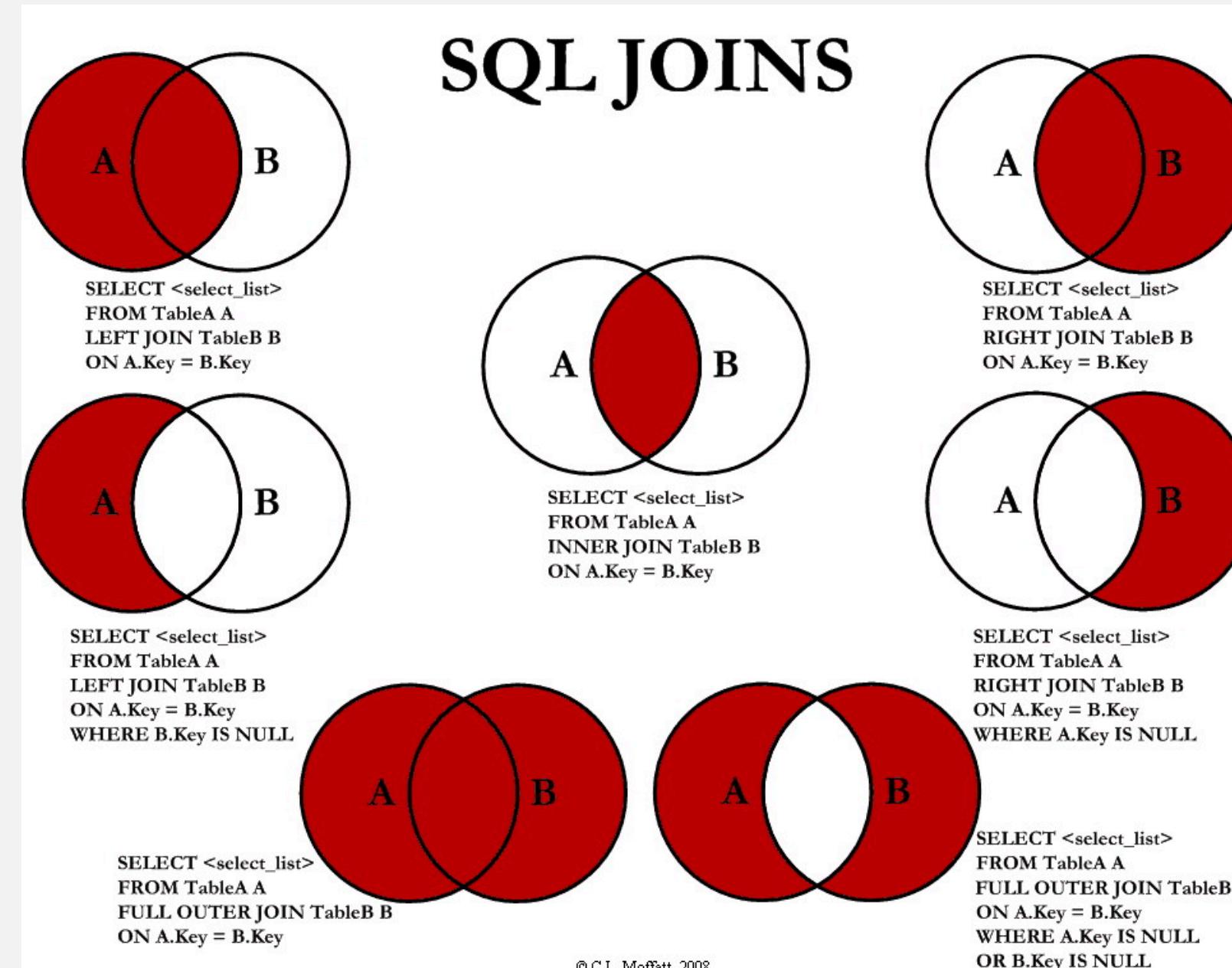
```
1      -- Exemple 1 : Ajouter une colonne email
2 •  ALTER TABLE Client ADD Email VARCHAR(100);
3
4      -- Exemple 2 : Modifier la taille d'une colonne
5 •  ALTER TABLE Client MODIFY Ville VARCHAR(100);
6
7      -- Exemple 3 : Supprimer la table complète
8 •  DROP TABLE Client;
9
10     -- Exemple 4 : Supprimer une colonne inutile
11 • ALTER TABLE Client DROP COLUMN Ville;
```

Contraintes Relationnelles (Foreign Keys)

- Principe : Pour lier deux tables (ex: un Client passe une Commande), on utilise une Clé Étrangère (FOREIGN KEY).
- Intégrité Référentielle (Concept Clé) : C'est une sécurité absolue. La base de données interdit d'insérer une commande pour un client qui n'existe pas. Cela garantit la cohérence des données.

```
1 • CREATE TABLE Commande (
2     IDCmd      INT PRIMARY KEY,
3     DateCmd    DATE,
4     IDClient   INT, -- La colonne de lien
5     Montant    DECIMAL(10,2),
6
7     -- Définition du lien vers la table Client
8     FOREIGN KEY (IDClient) REFERENCES Client(IDClient)
9 ) ;
```

Les Jointures (Interrogation Multi- Tables)



- Principe : Dans une base relationnelle, les données sont séparées pour éviter les doublons. La Jointure permet de les rassembler pour répondre à une question complexe.
- INNER JOIN (Interne) : Retourne uniquement les lignes qui ont une correspondance dans les deux tables (Intersection).
- LEFT JOIN (Externe) : Retourne toutes les lignes de la table de gauche, même si elles n'ont aucune correspondance à droite (les manques sont remplis par NULL).

Syntaxe et Exemples de Requêtes Jointes

- La Clause ON : C'est le moteur de la jointure. Elle définit l'égalité entre la Clé Primaire et la Clé Étrangère.
- Préfixage des colonnes : Utiliser Table.Colonne (ex: Client.Nom) est indispensable pour éviter les erreurs d'ambiguïté si les deux tables ont des noms de champs identiques.

```
1      -- Exemple : Lister les clients et leurs commandes respectives
2 •  SELECT Client.Nom, Commande.IDCmd, Commande.Montant
3   FROM Client
4   INNER JOIN Commande
5     ON Client.IDClient = Commande.IDClient;
```

- Note technique : Si vous remplacez INNER JOIN par LEFT JOIN, la requête affichera aussi les clients "inactifs" (ceux qui n'ont aucune commande), avec la mention NULL pour les colonnes de la table Commande.

Fonctions d'Agrégation & Groupement

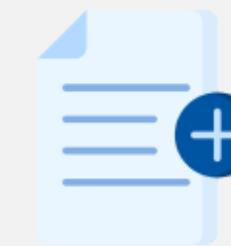
- Objectif : Effectuer des calculs sur un ensemble de lignes pour obtenir un résultat unique (statistiques).
- Fonctions standards :
- COUNT() : Compte le nombre d'enregistrements.
- SUM() / AVG() : Somme ou Moyenne des valeurs.
- MAX() / MIN() : Recherche des valeurs extrêmes.
- GROUP BY : Indispensable pour regrouper les résultats par catégorie (ex: CA par ville).
- HAVING : Filtre les groupes après le calcul (équivalent du WHERE pour les agrégations).

```
1  -- Objectif : Calculer le chiffre d'affaires total par ville
2  SELECT Client.Ville, SUM(Commande.Montant) AS CA_Total
3  FROM Client
4  INNER JOIN Commande ON Client.IDClient = Commande.IDClient
5  GROUP BY Client.Ville
6  HAVING SUM(Commande.Montant) > 1000;
```

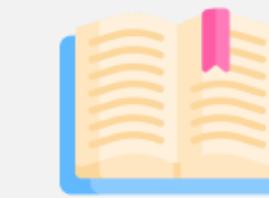
Manipulation des Données (LMD)

- Rôle : Permet d'agir sur le contenu des tables (les enregistrements) sans modifier la structure.
- Commandes Fondamentales :
- INSERT INTO : Ajoute de nouvelles lignes.
- UPDATE : Modifie des valeurs existantes (toujours utiliser avec un WHERE !).
- DELETE : Supprime des lignes (attention : irréversible).

```
1  -- 1. Ajouter un client
2  INSERT INTO Client (IDClient, Nom, Ville)
3  VALUES (101, 'Bennis', 'Casablanca');
4
5  -- 2. Mettre à jour (Ex: Changement de ville)
6 • UPDATE Client
7  SET Ville = 'Tanger'
8  WHERE IDClient = 101;
9
10 -- 3. Supprimer (Ex: Un client spécifique)
11 • DELETE FROM Client
12 WHERE IDClient = 101;
```



Create



Read



Update

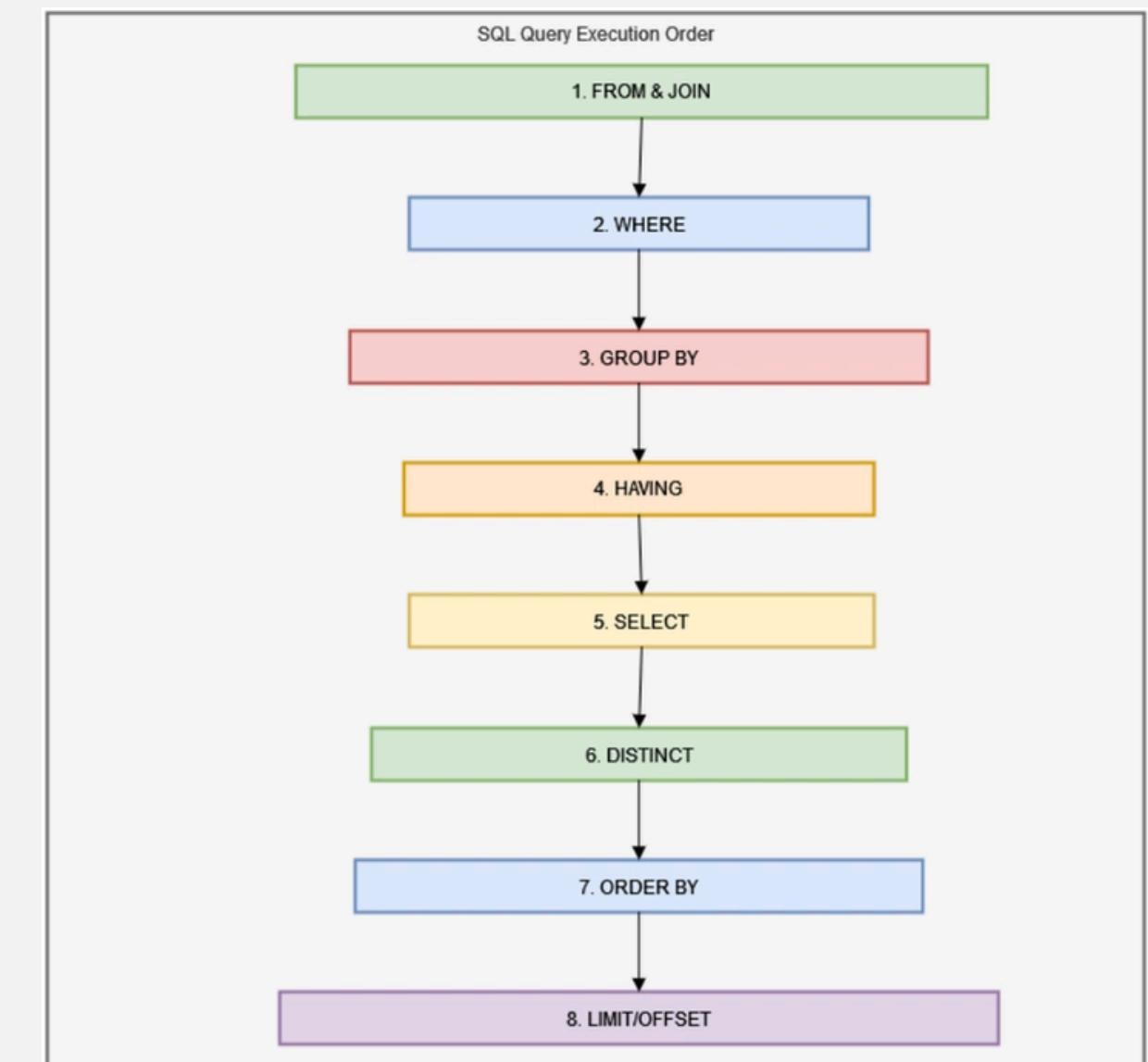


Delete

Ordre Logique d'Exécution & Tri

- L'ordre d'écriture \neq L'ordre d'exécution. * SQL traite les clauses dans cet ordre précis :
- FROM / JOIN (D'abord, on choisit les tables).
- WHERE (On filtre les lignes).
- GROUP BY (On rassemble).
- HAVING (On filtre les groupes).
- SELECT (On affiche enfin les colonnes).
- ORDER BY (On trie le résultat final).

```
1 -- Objectif : CA par ville, trié du plus grand au plus petit
2 SELECT Ville, SUM(Montant) AS CA
3 FROM Client
4 INNER JOIN Commande ON Client.IDClient = Commande.IDClient
5 GROUP BY Ville
6 ORDER BY CA DESC; -- DESC pour Décroissant, ASC pour Croissant
```



Les Sous-Requêtes (Subqueries)

- Définition : Une requête placée à l'intérieur d'une autre requête (souvent dans la clause WHERE ou FROM).
- Utilité : Permet de filtrer des données en fonction d'un calcul dynamique ou d'une liste provenant d'une autre table.
- Mots-clés associés : IN, NOT IN, EXISTS, ANY/ALL.

```
1  -- Objectif : Trouver les clients qui ont commandé plus que la moyenne globale
2  SELECT Nom
3  FROM Client
4  WHERE IDClient IN (
5      SELECT IDClient
6      FROM Commande
7      WHERE Montant > (SELECT AVG(Montant) FROM Commande)
8  );
```

Gestion des données absentes (NULL)

- Problème : Lors d'un LEFT JOIN, si un client n'a pas de commande, le résultat affiche NULL. Cela peut fausser les calculs ou l'affichage des rapports.
- Solution : Utiliser la fonction COALESCE() (ou IFNULL en MySQL) pour remplacer le vide par une valeur par défaut.

```
1      -- Sans COALESCE : Affiche "NULL" pour les nouveaux clients
2      -- Avec COALESCE : Affiche "0" ou "Pas de commande"
3 • SELECT
4      Client.Nom,
5      COALESCE(Commande.Montant, 0) AS Montant_Propre
6  FROM Client
7  LEFT JOIN Commande ON Client.IDClient = Commande.IDClient;
```

Alias et Auto-Jointures (Self-Join)

- Les Alias (AS) : Simplifient l'écriture des requêtes longues et améliorent la lisibilité (ex: C pour Client).
- Auto-Jointure : Technique avancée consistant à joindre une table avec elle-même.
- Cas d'usage : Utile pour les structures hiérarchiques (ex: trouver les employés qui travaillent dans le même service qu'un autre).

```
1  -- Utilisation d'Alias pour la clarté
2 • SELECT C.Nom, Cmd.DateCmd
3   FROM Client AS C
4   JOIN Commande AS Cmd ON C.IDClient = Cmd.IDClient
5   WHERE C.Ville = 'Casablanca';
```

Ce qu'il faut retenir (chap 2)

- 1. Les trois piliers du langage :

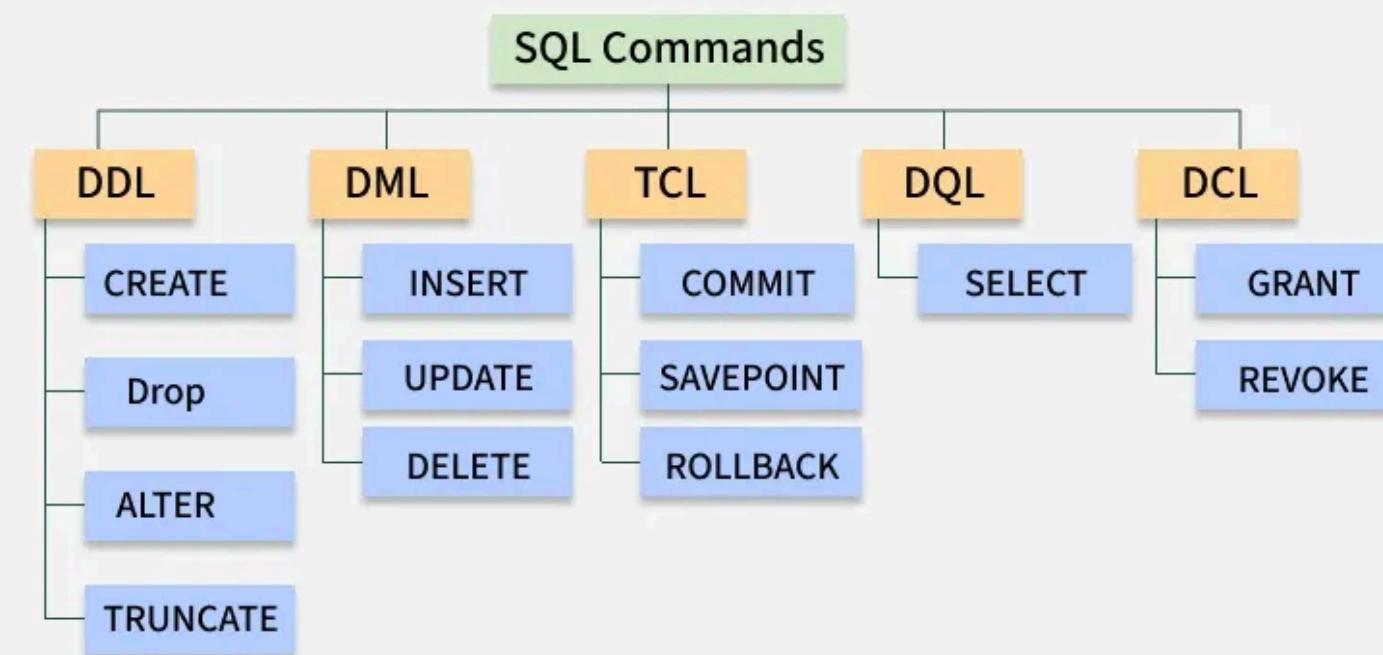
- LDD (Définition) : CREATE, ALTER, DROP pour structurer la base et garantir l'intégrité (Clés Primaires/Étrangères).
- LMD (Manipulation) : INSERT, UPDATE, DELETE pour gérer le cycle de vie des données.
- LID (Interrogation) : SELECT associé aux JOIN pour croiser les informations.

- 2. Les concepts avancés pour l'analyse :

- Agrégations : GROUP BY et HAVING pour transformer des lignes en statistiques.
- Sous-requêtes : Pour des filtres dynamiques et complexes.
- Qualité des données : Gestion des NULL avec COALESCE et lisibilité avec les Alias.

- 3. La règle d'or du développeur :

- L'ordre logique : Toujours se rappeler que le moteur SQL traite le FROM et le WHERE bien avant d'afficher le SELECT.



Chapitre 3 : Optimisation des Bases de Données

*Accélérer les performances, gérer
l'indexation et optimiser les plans
d'exécution pour le traitement de
gros volumes de données.*

Pourquoi l'Optimisation est-elle vitale ?

1. Le problème : La croissance non linéaire des temps de réponse

- Full Table Scan (Parcours Séquentiel) : Sans optimisation, le SGBD lit chaque ligne une par une. Sur 1 million de lignes, c'est une catastrophe en termes de performance.
- Complexité Algorithmique : On passe d'une recherche en $O(n)$ (très lent) à une recherche en $O(\log n)$ avec indexation (quasi instantané).

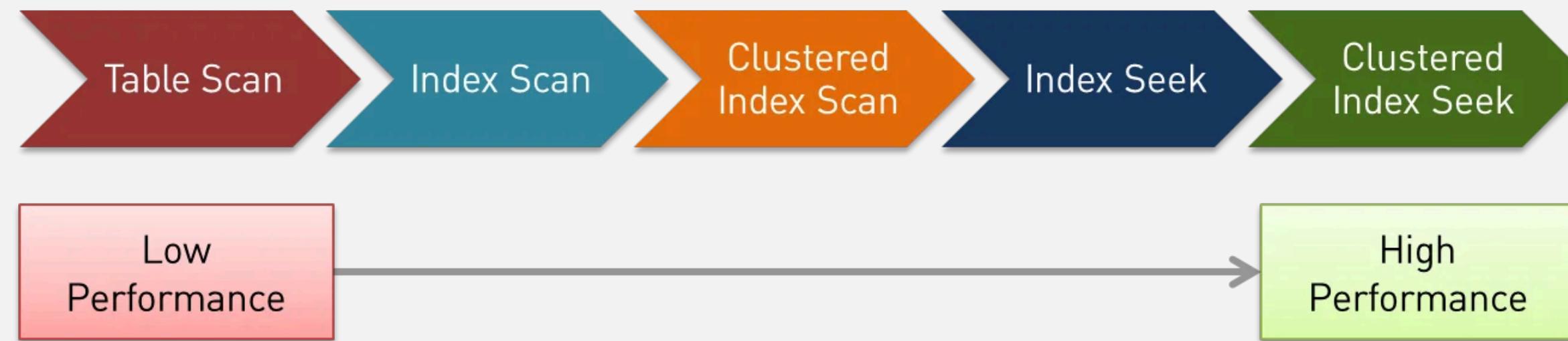
2. Les indicateurs de performance (KPIs)

- La Latence : Réduire le temps d'attente pour une requête unique (ex: afficher le profil d'un client).
- Le Débit (Throughput) : Maximiser le nombre de transactions traitées par seconde par le serveur.

3. Le Goulot d'Étranglement : Les I/O Disque

- L'accès au disque dur est des milliers de fois plus lent que l'accès à la mémoire vive (RAM). L'optimisation vise à minimiser ces "lectures physiques" coûteuses.

La Solution : L'Indexation et les Méthodes d'Accès



- Scan (Lent) : Le moteur lit une grande partie ou la totalité de la table (ex: Table Scan, Index Scan).
- Seek (Rapide) : Le moteur va directement à l'endroit précis de la donnée grâce à l'index (recherche ciblée).
- L'avantage du "Clustered Index" : C'est le niveau ultime de performance car les données sont physiquement ordonnées sur le disque selon la clé de l'index.

Structure Technique : Le fonctionnement du B-Tree

1. Le Concept de l'Arbre Équilibré

- Structure Hiérarchique : L'index n'est pas une liste, mais un arbre composé d'une Racine, de Nœuds internes et de Feuilles.
- Équilibrage Automatique : Le SGBD maintient l'arbre équilibré pour que la distance entre la racine et n'importe quelle donnée soit toujours la même (minimisation des accès disque).

2. Le parcours d'une recherche (Le "Seek")

- Au lieu de lire 1 000 000 de lignes, le moteur descend 3 ou 4 niveaux de l'arbre pour atteindre la feuille contenant la donnée.
- Chaque niveau correspond à une seule lecture I/O, ce qui explique la performance fulgurante.

3. Syntaxe SQL de création :

```
1  -- Création d'un index classique (Non-Clustered)
2  CREATE INDEX idx_ville ON Client(Ville);
3
4  -- L'index sur la Clé Primaire est créé automatiquement
5  -- sous forme de Clustered Index par le SGBD.
```

Analyse de la Structure B-Tree

1. La Hiérarchie des Niveaux :

- Root Node (Racine) : Le point d'entrée unique de la recherche.
- Intermediate Nodes (Nœuds Internes) : Servent d'aiguillage pour diriger le moteur vers la bonne branche.
- Leaf Nodes (Feuilles) : Le niveau final qui contient les entrées de l'index et les pointeurs vers les données réelles de la table.

2. Le parcours d'un "Seek" (Exemple : Recherche de "Garcia") :

- Le moteur commence à la racine "S".
- Comme "G" est avant "S", il descend vers le nœud intermédiaire de gauche (F | G).
- Il trouve la borne "G" et descend directement vers la feuille contenant "Garcia".
- Bilan : En seulement 3 étapes (3 lectures I/O), la donnée est trouvée parmi des milliers.

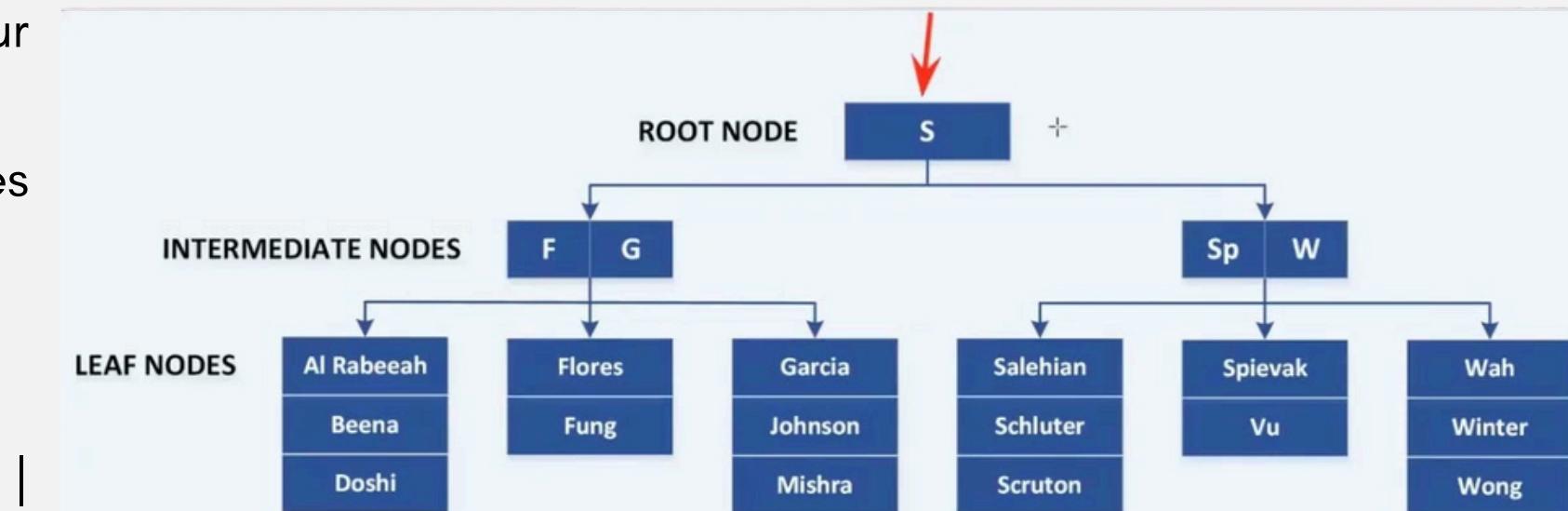


Figure 7.8.
A sample B-Tree index.

Pourquoi l'arbre est-il "Équilibré" ? Le SGBD s'assure que la profondeur (le nombre de niveaux) est identique partout. Cela garantit que le temps de réponse est constant, peu importe que vous cherchiez "Al Rabeeah" ou "Wong". Coût de maintenance : Attention, chaque INSERT ou DELETE oblige le SGBD à réorganiser l'arbre. Trop d'index peuvent donc ralentir les écritures.

Vérification : L'outil EXPLAIN

1. Rôle de la commande :

- Demande au SGBD d'afficher son plan d'exécution avant de lancer la requête réelle.
- Permet de vérifier si le moteur utilise bien un Index Seek ou s'il retombe dans un Table Scan.

2. Lecture du résultat (Ce qu'il faut surveiller) :

- type : Chercher const, eq_ref ou range (bons signes). Éviter ALL (signifie Full Table Scan).
- key : Affiche le nom de l'index réellement utilisé par le moteur.
- rows : Estimation du nombre de lignes que le moteur devra examiner. Plus ce chiffre est bas, plus la requête est optimisée.

3. Exemple de code :

```
1 -- Analyser la performance de la recherche
2 EXPLAIN SELECT * FROM Client WHERE Ville = 'Casablanca';
```

Optimisation de l'Écriture : Le concept de SARGability

1. Définition du SARG (Search ARGumentable)

- Une requête est dite "Sargable" lorsqu'elle permet au moteur SQL d'utiliser efficacement les index existants (Index Seek).
- Le Piège : Appliquer une fonction sur une colonne indexée dans la clause WHERE "casse" l'index et force un Table Scan.

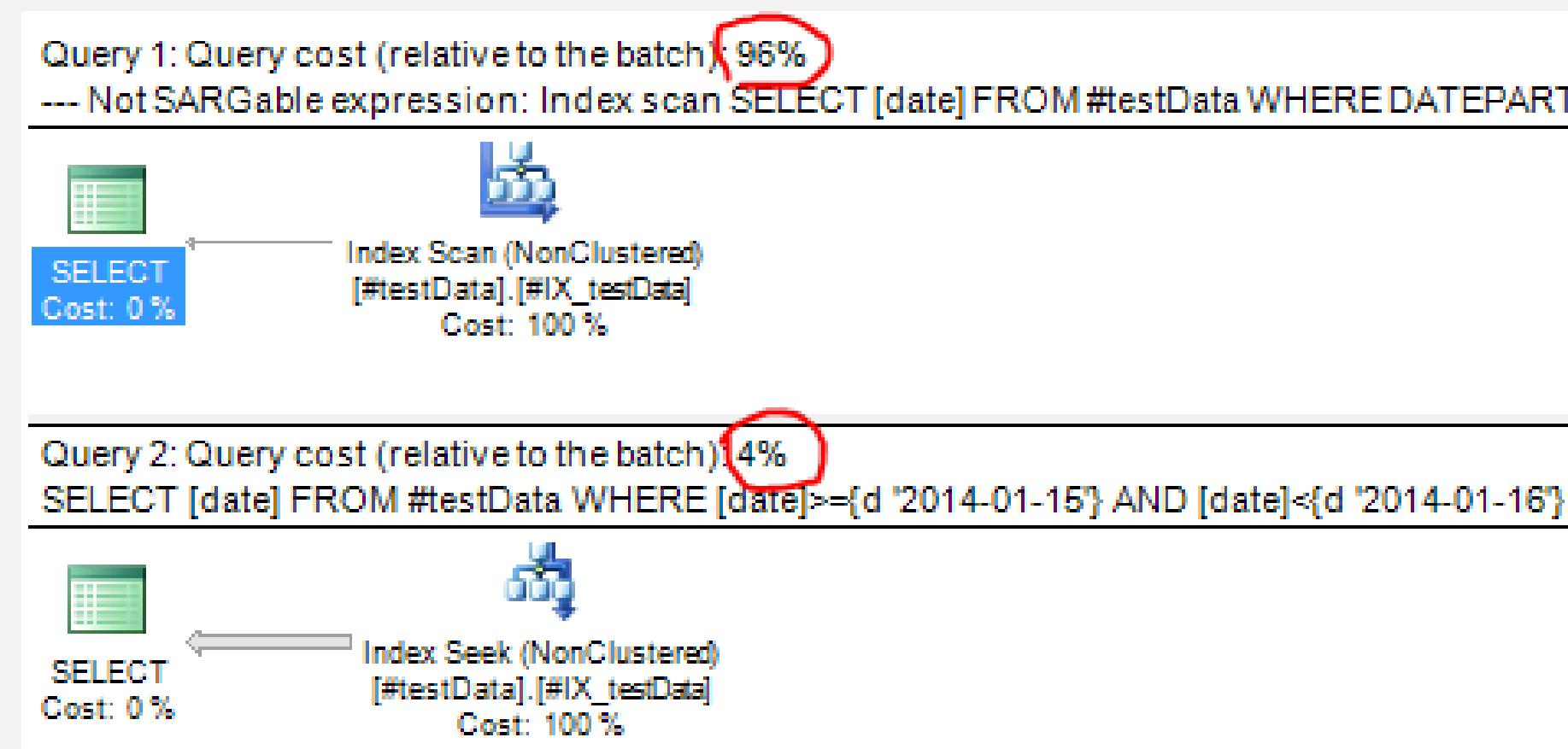
2. Comparaison (Performance) :

Anti-Pattern (Lent)	Optimisé (Rapide)
WHERE YEAR(Date) = 2025	WHERE Date >= '2025-01-01' AND Date < '2026-01-01'
WHERE UPPER(Nom) = 'ALAMI'	WHERE Nom = 'Alami' (Utiliser le Collation de la base)
WHERE Ville LIKE 'CASA%' (L'index fonctionne uniquement par préfixe)	WHERE Ville LIKE 'CASA%' (L'index fonctionne uniquement par préfixe)

3. Le fléau du SELECT *

- Impact : Récupère des colonnes inutiles, sature la bande passante et empêche l'utilisation d'index "couvrants".
- Bonne pratique : Lister uniquement les colonnes nécessaires (ex: SELECT Nom, Email).

Impact Réel sur la Performance : Comparaison du Coût d'Exécution



- L'Indice n'est pas tout : Avoir un index est inutile si la requête n'est pas SARGable (cas du 96% de coût).
- Sobriété : Éviter le SELECT * n'est pas qu'une question de style, c'est une nécessité pour économiser la RAM et le CPU.
- Bilan : Une requête bien écrite peut diviser la charge serveur par 24.

Fondements Physiques : La notion de Page (Block)

1. L'Unité de Stockage (La Page) :

- Le SGBD ne lit jamais une "ligne" seule sur le disque. Il lit des Pages (généralement 8 Ko).
- Une page contient plusieurs lignes. Si une requête est mal optimisée, le SGBD charge des milliers de pages inutiles en RAM.

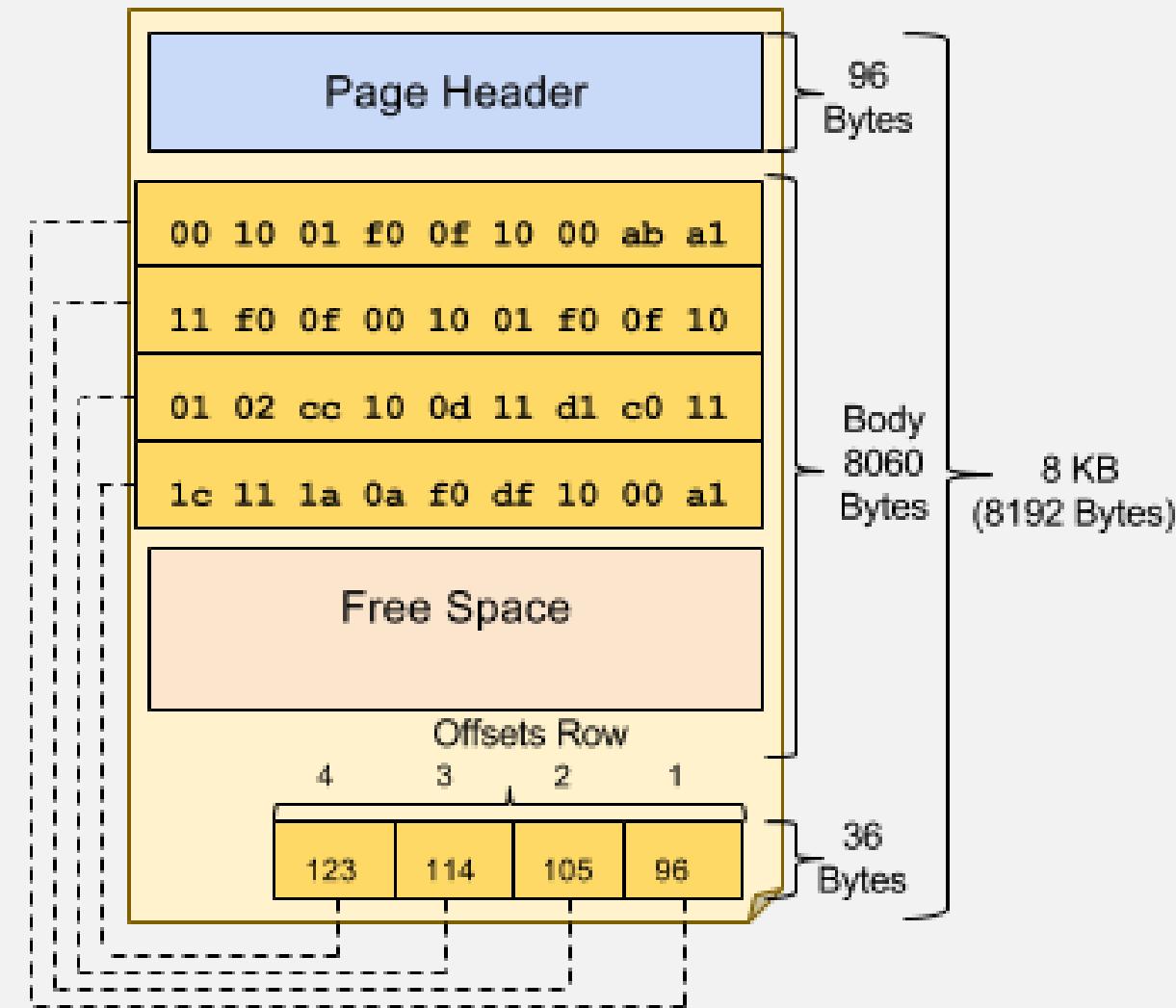
2. Le Goulot d'Étranglement (I/O Disque) :

- Lecture Logique : La page est déjà en RAM (très rapide).
- Lecture Physique : Aller chercher la page sur le disque (très lent).
- Objectif de l'index : Réduire drastiquement le nombre de pages à charger pour répondre à une requête.

3. La Fragmentation :

- À force d'insertions et de suppressions, les pages se vident partiellement ou s'éparpillent.
- Solution : Maintenance des index (REBUILD ou REORGANIZE) pour regrouper les données.

Illustration Technique : Structure interne d'une Page de Données (8 Ko)



- L'Unité d'I/O : Ce bloc de 8 Ko représente la plus petite quantité de données que le SGBD peut lire ou écrire sur le disque en une seule opération (I/O)."
- Densité des données : Les lignes (Data Row 1, 2, etc.) sont empilées. Plus les lignes sont courtes, plus on en met dans une page, moins on fait d'I/O.
- La menace de la Fragmentation : "L'espace libre en bas de page est normal. Mais si cet espace devient trop grand à cause de nombreuses suppressions (DELETE), le SGBD finit par lire des pages 'vides'. C'est la fragmentation, qui ralentit tout le système.

Architecture : Clustered vs Non-Clustered Index

1. Index Clusterisé (Clustered) :

- Définition : L'index est la table. Les données sont physiquement triées et stockées sur le disque selon cet index.
- Contrainte : Un seul par table (généralement sur la Clé Primaire).
- Analogie : Un Dictionnaire (les mots sont rangés directement par ordre alphabétique).

2. Index Non-Clusterisé (Non-Clustered) :

- Définition : Une structure séparée contenant une copie triée de la colonne indexée et un pointeur vers la ligne réelle de la table.
- Contrainte : On peut en créer plusieurs.
- Analogie : L'index à la fin d'un livre (on cherche un mot, il donne le numéro de page pour trouver le contenu).

Caractéristique	Index Clusterisé	Index Non-Clusterisé
Stockage	Physique (Données triées)	Logique (Structure à part)
Vitesse Lecture	Maximale	Très Rapide
Vitesse Écriture	Plus lente (réorganisation)	Coût de maintenance modéré

La règle d'or : Sélectivité et Cardinalité

1. Le concept de Sélectivité :

- Un index est efficace uniquement si la colonne contient beaucoup de valeurs uniques.
- Haute Sélectivité (Cible idéale) : Colonne Email, Identifiant. L'index est ultra-performant.
- Basse Sélectivité (À éviter) : Colonne Sexe (M/F) ou Actif (0/1). Le moteur SQL préférera souvent un Table Scan car l'index n'aide pas à filtrer assez de lignes.

2. Le compromis "Performance / Écriture" :

- Chaque index ralentit les opérations de modification : INSERT, UPDATE, DELETE.
- Pourquoi ? Parce que le SGBD doit mettre à jour l'arbre B-Tree à chaque changement.

3. Conseil :

- Il ne faut indexer que les colonnes fréquemment utilisées dans les clauses WHERE, JOIN ou ORDER BY, tout en évitant de sur-indexer les tables qui subissent beaucoup d'écritures.

L'Optimisation de Haut Niveau : Index Composés & Couvrants

- 1. L'Index Composé (Multi-colonnes)
 - Définition : Un index créé sur plusieurs colonnes simultanément, par exemple : (Nom, Prenom).
 - La Règle du Préfixe : L'ordre des colonnes est crucial. Un index sur (A, B) est utilisable pour une recherche sur A ou sur A et B, mais il est inutile pour une recherche sur B seul.
 - Usage : Idéal pour les filtres combinés fréquents.
- 2. L'Index Couvrant (Covering Index)
 - Le Concept : C'est lorsqu'un index contient toutes les données demandées par la clause SELECT.
 - Le Miracle de la Performance : Le SGBD trouve tout ce dont il a besoin directement dans l'arbre B-Tree. Il n'a aucun besoin d'aller lire la table physique (accès disque évité).
 - Résultat : C'est ce qui permet d'atteindre les coûts d'exécution les plus bas (comme les 4% vus précédemment).
 - 3. La Règle d'Or de la Performance :
 - "L'optimisation n'est pas seulement l'ajout d'index, c'est l'art de minimiser les lectures physiques pour maximiser la vitesse de la RAM. Moins on touche au disque, plus la base est performante."

Conclusion du Chapitre 3

1. Les Fondations Physiques & Théoriques :

- Performance : Passage d'une complexité linéaire $O(n)$ à une complexité logarithmique $O(\log n)$.
- Stockage : Maîtrise de la notion de Pages (8 Ko) et réduction des I/O disque.
- 2. L'Architecture des Index :
- Choix du type : Distinguer le Clustered (physique) du Non-Clustered (logique).
- Sélectivité : Ne pas indexer au hasard ; privilégier les colonnes à forte cardinalité.

3. Techniques Avancées & Écriture :

- SARGability : Écrire des requêtes qui permettent l'usage des index (pas de fonctions sur les colonnes).
- Optimisation "Elite" : Utiliser des Index Composés et viser l'Index Couvrant pour éviter tout accès à la table.

4. Le Diagnostic Permanent :

- Outil : Ne jamais deviner, toujours valider avec la commande EXPLAIN.
- Objectif : Transformer chaque Scan (lent) en un Seek (chirurgical).

Chapitre 4 : Les Bases de Données NoSQL

*L'avènement du 'Not Only SQL' :
S'affranchir des contraintes
relationnelles pour répondre aux
enjeux de flexibilité, de haute
disponibilité et de scalabilité
horizontale.*

Pourquoi sortir du modèle Relationnel ?

1. Le défi de la Scalabilité (Montée en charge)

- SQL (Scalabilité Verticale / Scale-up) : Pour augmenter la puissance, on achète un serveur plus gros (CPU, RAM). C'est extrêmement coûteux et limité par la technologie matérielle.
- NoSQL (Scalabilité Horizontale / Scale-out) : On distribue les données sur des dizaines de serveurs standards en cluster. C'est virtuellement illimité et beaucoup moins onéreux.

2. La Rigidité du Schéma (Schema-fixed)

- En SQL, la structure est définie a priori. Modifier une colonne sur une table de plusieurs téraoctets est une opération risquée qui peut paralyser la production pendant des heures.
- Le NoSQL permet un Schéma Flexible : chaque enregistrement peut avoir ses propres attributs sans impacter les autres.

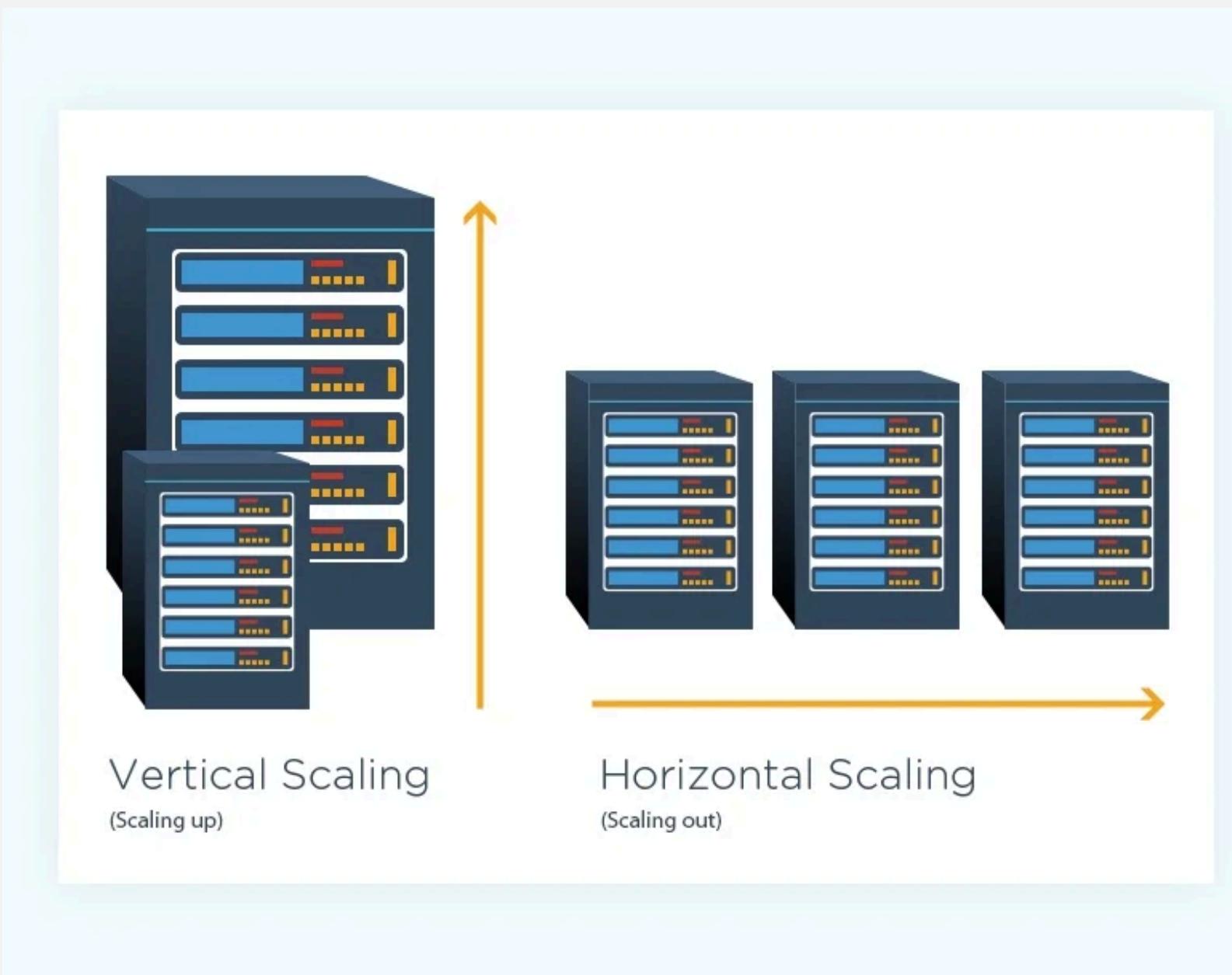
3. Le Contexte du Big Data (Les 3V)

- Volume : Gigantesque quantité de données.
- Vélocité : Vitesse de création et de traitement (ex: flux réseaux sociaux).
- Variété : Données non structurées ou semi-structurées (JSON, logs, images, vidéos).

4. L'Impédance de Structure (Object-Relational Mismatch)

- Les développeurs manipulent des Objets complexes, alors que le SQL force à les décomposer en Tables (Jointures coûteuses). Le NoSQL (notamment Document) stocke la donnée comme elle est utilisée par l'application.

Visualisation : Scalabilité Verticale vs Horizontale



Scalabilité Verticale (Scale-up) :

Action : Augmenter la puissance d'un serveur unique (Plus de RAM, plus de CPU).

- Limite : Plafond technologique du matériel et coût exponentiel.

Scalabilité Horizontale (Scale-out) :

- Action : Distribuer les données sur une multitude de serveurs standards travaillant en cluster.
- Avantage : Capacité virtuellement illimitée et coût linéaire.

Le lien avec le NoSQL :

- C'est cette capacité à "scaler out" (horizontalement) qui permet au NoSQL de gérer des volumes massifs là où le SQL classique atteint ses limites physiques.

Le Théorème de CAP (Théorème de Brewer)

1. Les trois piliers fondamentaux :

- C (Consistency - Cohérence) : Tous les nœuds du système voient la même donnée au même moment après une écriture. C'est l'assurance d'avoir la version la plus récente partout.
- A (Availability - Disponibilité) : Chaque requête reçoit une réponse (succès ou échec), garantissant que le système reste opérationnel même si certains nœuds tombent.
- P (Partition Tolerance - Tolérance au fractionnement) : Le système continue de fonctionner malgré une coupure de communication ou une panne réseau entre les serveurs.

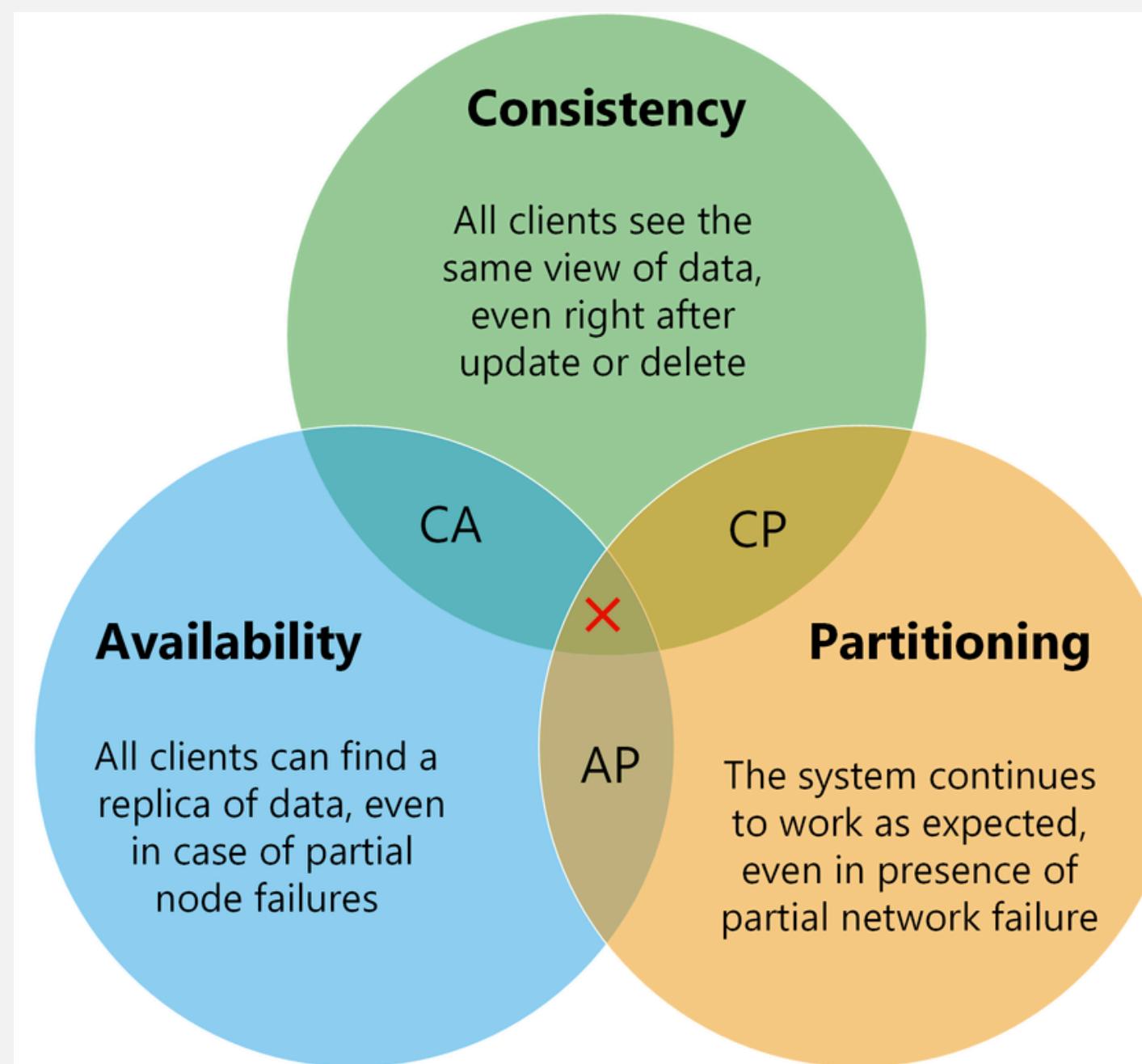
2. La Règle de l'Impossible :

- "Dans un système distribué, il est mathématiquement impossible de garantir simultanément la Cohérence, la Disponibilité et la Tolérance au fractionnement."

3. Le Choix Stratégique : Comme la Tolérance au fractionnement (P) est obligatoire dans les architectures modernes (le réseau n'est jamais fiable à 100%), le choix se résume souvent à :

- CP (Cohérence + Tolérance) : On sacrifie la disponibilité pour garantir l'exactitude des données (ex: MongoDB, HBase).
- AP (Disponibilité + Tolérance) : On sacrifie la cohérence immédiate pour garantir que le service répond toujours (ex: Cassandra, DynamoDB).

Représentation Graphique : L'Arbitrage du Théorème de CAP



- Le Point 'X' (L'Impossible) : Indique qu'aucun système distribué ne peut se situer au centre. Un choix architectural est obligatoire.

L'Intersection CA (Relationnel Classique) :

- Garantit la cohérence et la disponibilité, mais supporte mal les coupures réseau (souvent sur un serveur unique).

L'Intersection CP (Cohérence + Partition) :

- Le système bloque l'accès aux données si elles ne sont pas synchronisées partout pour garantir l'exactitude.

L'Intersection AP (Disponibilité + Partition) :

- Le système répond toujours, même si les données ne sont pas encore à jour sur tous les serveurs (Cohérence à terme).

Classification des Bases NoSQL

Le monde du NoSQL se divise en quatre catégories principales, chacune répondant à des besoins spécifiques de stockage et de performance issus des arbitrages du Théorème de CAP.

1. Orientée Clé-Valeur (Key-Value) [Modèle AP / CP]

- Concept : Le modèle le plus simple. Une clé unique pointe vers un "blob" de données.
- Points forts : Latence ultra-faible, lecture/écriture massive.
- Exemples : Redis, Amazon DynamoDB.
- Usage : Gestion de sessions, caches, paniers d'achat.

2. Orientée Documents (Document Store) [Modèle CP]

- Concept : Stockage de données semi-structurées sous forme de documents (JSON, BSON, XML).
- Points forts : Flexibilité totale (schéma dynamique), supporte les données imbriquées.
- Exemples : MongoDB, CouchDB.
- Usage : Catalogues produits, gestion de contenu (CMS), applications mobiles.

3. Orientée Colonnes (Wide-Column Store) [Modèle AP]

- Concept : Les données sont stockées par familles de colonnes plutôt que par lignes.
- Points forts : Très performant pour les agrégations sur des volumes massifs (pétaoctets).
- Exemples : Cassandra, HBase.
- Usage : Big Data, logs d'événements, séries temporelles.

4. Orientée Graphes (Graph Database) [Modèle CP]

- Concept : Utilise des nœuds (entités) et des arcs (relations) comme citoyens de premier rang.
- Points forts : Performance inégalée pour parcourir des relations complexes.
- Exemples : Neo4j, OrientDB.
- Usage : Réseaux sociaux, systèmes de recommandation, détection de fraude.

MongoDB : Flexibilité et Puissance du format BSON

1. Structure des données : Du JSON au BSON

- Format Document : Les données sont stockées sous forme de documents JSON (JavaScript Object Notation), ce qui est naturel pour les développeurs.
- Le BSON (Binary JSON) : En interne, MongoDB utilise le BSON, une extension binaire du JSON.
- Avantages : Plus rapide à parcourir, supporte plus de types de données (Date, Binary, etc.) et optimise l'espace de stockage.

2. Organisation des données : Terminologie | Concept SQL | Concept MongoDB | | :--- | :--- | | Table | Collection (Groupe de documents) | | Ligne (Row) | Document (Objet unique) | | Colonne | Champ (Field) | | Jointure (Join) | Documents imbriqués (Embedding) |

3. Le Schéma Dynamique (Schema-less)

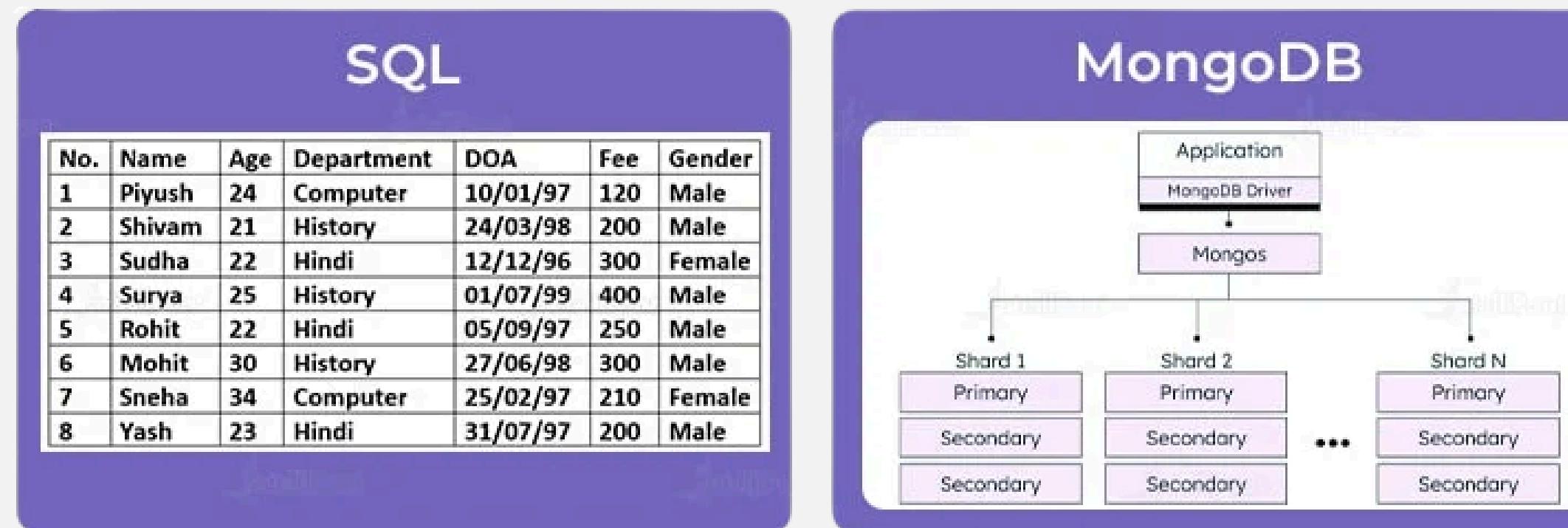
- Contrairement au SQL, une Collection n'impose pas de structure rigide.
- Deux documents dans la même collection peuvent avoir des champs différents.
- Avantage : Permet une itération rapide du développement sans migrations de base de données lourdes (ALTER TABLE).

4. Modélisation : L'imbrication (Embedding)

- Au lieu de séparer les données en plusieurs tables liées par des clés étrangères, MongoDB privilégie le regroupement des données liées dans un seul document.

Comparaison : Centralisation SQL vs Distribution MongoDB

Figure : Illustration de la transition d'un modèle relationnel centralisé (SQL) vers une architecture de cluster distribuée (MongoDB). L'utilisation du Sharding (partitionnement) et de la Réplication permet de garantir à la fois la scalabilité horizontale et la haute disponibilité des données.



Syntaxe et Manipulation : Du SQL au NoSQL

- Le passage au NoSQL transforme le langage déclaratif (SQL) en une manipulation par méthodes d'objets, parfaitement adaptée aux langages de programmation modernes.
- 1. Tableau Comparatif des Opérations de Base

Opération	SQL (Relationnel)	MongoDB (Document)
Create	INSERT INTO users (nom, age) VALUES ('Ali', 22)	db.users.insertOne({ nom: "Ali", age: 22 })
Read	SELECT * FROM users WHERE age > 20	db.users.find({ age: { \$gt: 20 } })
Update	UPDATE users SET age = 23 WHERE nom = 'Ali'	db.users.updateOne({ nom: "Ali" }, { \$set: { age: 23 } })
Delete	DELETE FROM users WHERE age < 18	db.users.deleteMany({ age: { \$lt: 18 } })

Manipulation Avancée & Pipeline d'Agrégation

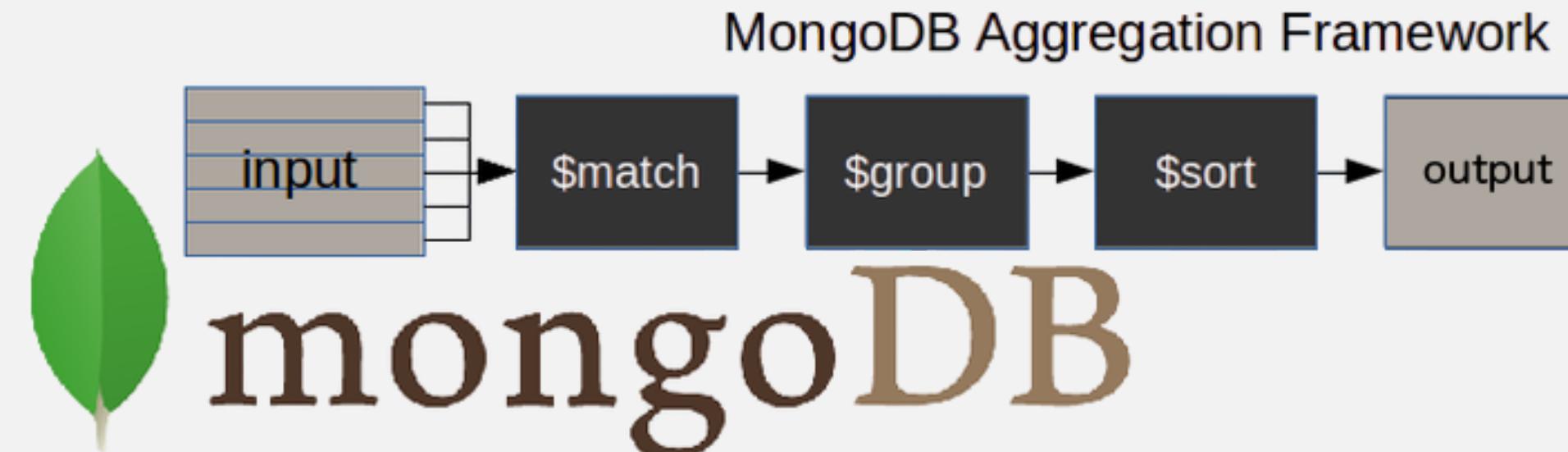
2. Les Particularités de la Syntaxe MongoDB

- L'utilisation des sélecteurs : Contrairement au SQL, on utilise des opérateurs préfixés par \$ comme \$gt (Greater Than), \$lt (Less Than), ou \$in.
- L'atomicité : Les opérations de mise à jour comme \$set permettent de modifier un champ précis sans avoir à réécrire tout le document.
- Le format JSON : Les requêtes sont elles-mêmes des documents JSON, ce qui assure une cohérence totale entre la donnée et la manière de l'interroger.

3. Puissance de l'Agrégation (Aggregation Framework)

- Là où le SQL utilise GROUP BY, MongoDB propose un Pipeline d'agrégation (db.collection.aggregate([...])) permettant de transformer et filtrer les données à travers plusieurs étapes successives (Stage).

Figure : Schéma de fonctionnement du Pipeline d'Agrégation MongoDB



NoSQL vs SQL : Choisir entre Flexibilité et Rigueur

1. Le Modèle BASE (NoSQL)

- Priorise la Disponibilité et la Scalabilité (vu avec le Théorème de CAP).
- Basically Available (Disponibilité de base).
- Soft-state (État fluctuant des données).
- Eventual Consistency (Cohérence à terme) : La donnée finit par être synchronisée, mais peut être temporairement différente d'un serveur à l'autre.
- Idéal pour : Big Data, réseaux sociaux, catalogues produits.

2. Le Modèle ACID (Relationnel)

- Priorise l'Intégrité absolue et la Cohérence forte.
- Chaque modification est une "Transaction" qui doit réussir totalement ou échouer totalement.
- Aucun compromis n'est fait sur l'exactitude des données.
- Idéal pour : Transactions bancaires, systèmes de réservation, stocks critiques.

3. Synthèse :

Besoin	Modèle Privilégié
Volume massif & Vitesse	NoSQL (Modèle BASE)
Précision & Sécurité transactionnelle	SQL (Modèle ACID)

Conclusion : La Persistance Polyglotte (Polyglot Persistence)

1. Le Concept de Polyglot Persistence

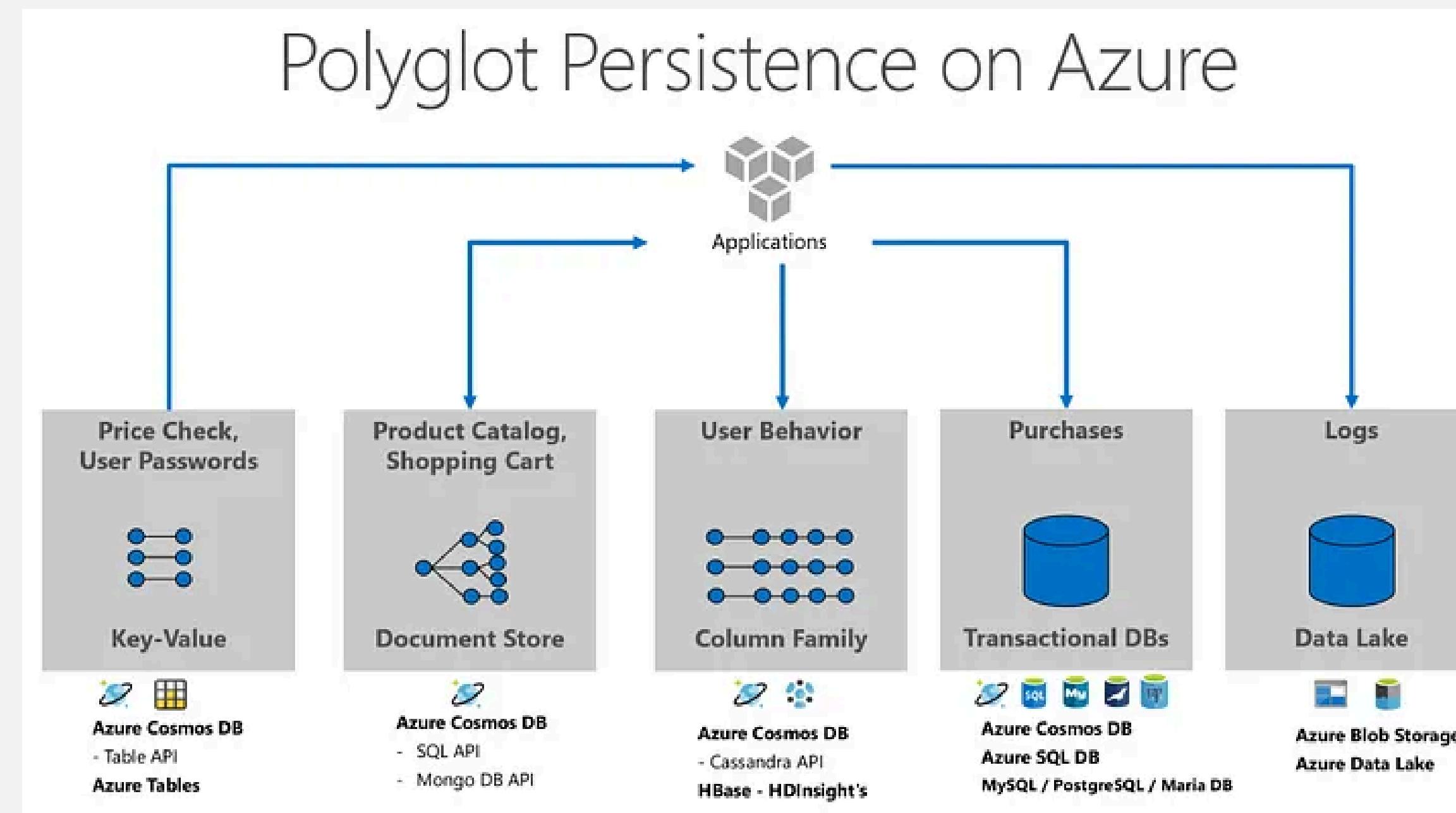
- Une application moderne est souvent décomposée en micro-services, chacun utilisant la base de données la mieux adaptée à ses besoins spécifiques.
- On ne cherche plus une base de données "unique" pour tout faire, mais une combinaison de solutions.

2. Exemple d'Architecture Hybride (E-commerce)

- SQL (PostgreSQL/MySQL) : Pour la gestion des comptes utilisateurs, la facturation et les transactions financières (Besoin de rigueur ACID).
- NoSQL - Document (MongoDB) : Pour le catalogue produit (Besoin de flexibilité pour des attributs variés).
- NoSQL - Clé-Valeur (Redis) : Pour la gestion des sessions et le panier d'achat (Besoin de vitesse extrême).
- NoSQL - Graphes (Neo4j) : Pour le moteur de recommandation "les clients ont aussi acheté..." (Besoin de gérer des relations complexes).

L'architecte de données moderne doit maîtriser la diversité technologique pour garantir à la fois la performance massive et l'intégrité absolue.

Figure : Architecture de Persistance Polyglotte appliquée à un écosystème Cloud



Le Cloud : Database-as-a-Service (DBaaS)

- Concept : Externalisation de l'administration, du patching et de la haute disponibilité.
- Avantages SITD : * Scalabilité Automatique : La base s'adapte en temps réel à la charge.
- Économie (Pay-as-you-go) : On ne paie que pour les ressources consommées.
- Leaders du Marché : AWS RDS (SQL), DynamoDB (NoSQL), Azure Cosmos DB (Multi-modèle).

Chapitre 5 : Les Transactions ACID

*Garantir la fiabilité et l'intégrité
absolue des données
transactionnelles*

Qu'est-ce qu'une Transaction ?

1. Définition Formelle

- Une Transaction est une unité logique de traitement (LUW - Logical Unit of Work) qui regroupe une suite d'opérations (lecture, écriture, modification) exécutées comme un bloc unique.
- Elle fait passer la base de données d'un état cohérent A à un état cohérent B.

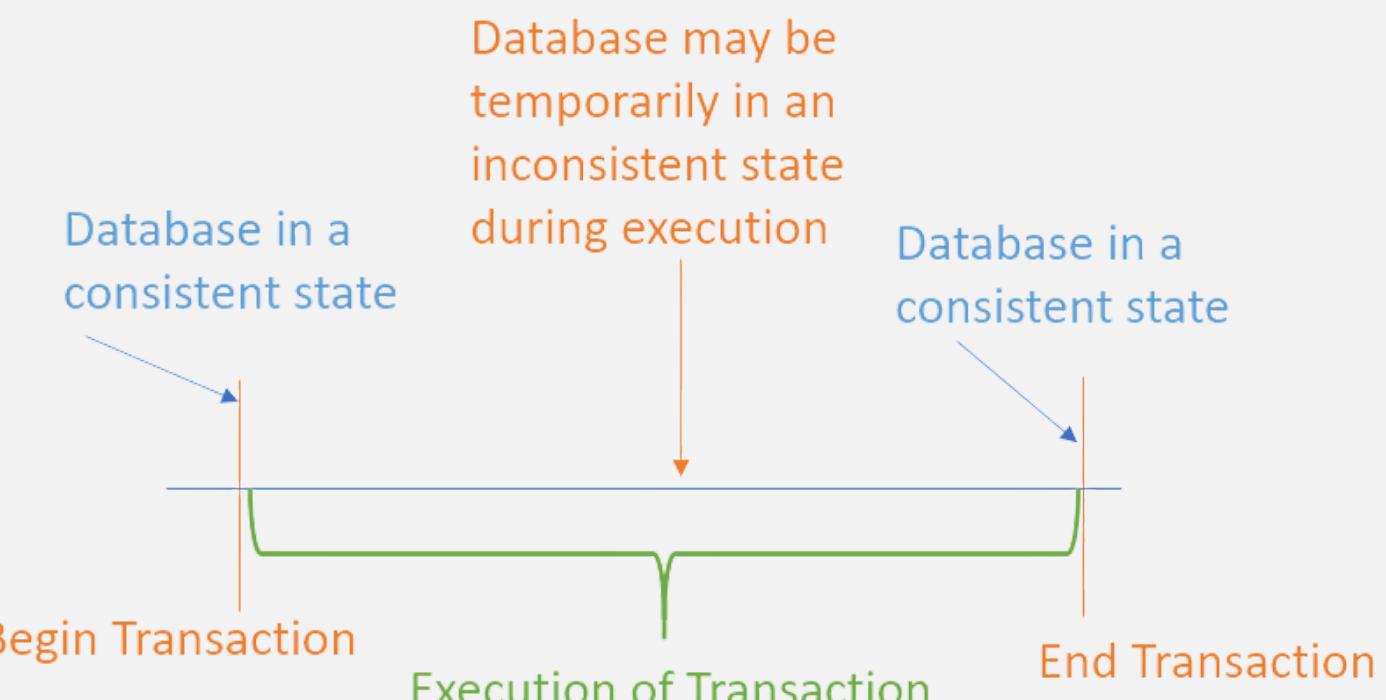
2. L'exemple "Étalon" : Le Virement Bancaire Pour transférer 100 DH du compte X vers le compte Y, le SGBD effectue deux opérations :

- UPDATE Compte_X SET solde = solde - 100
- UPDATE Compte_Y SET solde = solde + 100
- Le danger : Si une panne survient entre l'étape 1 et 2, l'argent "disparaît". La transaction garantit que soit les deux réussissent, soit rien ne change.

3. Cycle de vie simplifié

- BEGIN : Début de la transaction.
- COMMIT : Validation définitive des changements en base.
- ROLLBACK : Annulation totale et retour à l'état initial en cas d'erreur.

Figure : Transition d'états et maintien de la cohérence lors d'une transaction.



Le Modèle ACID – A pour Atomicité

Figure : Visualisation du concept d'Atomicité au sein du modèle ACID.

1. Principe Fondamental

- "All or Nothing" : Soit toutes les opérations de la transaction sont validées en mémoire permanente (disque), soit aucune ne l'est.
- Il est impossible d'avoir une transaction "partiellement exécutée".

2. Mécanisme technique : Le "Rollback"

- En cas d'échec (panne système, erreur logique, perte de connexion), le SGBD utilise les journaux de logs (Transaction Logs) pour défaire les modifications déjà entamées.
- La base de données est remise exactement dans l'état où elle était avant le BEGIN.

3. Pourquoi est-ce vital ?

- Pour éviter les données "fantômes" ou orphelines.
- Pour garantir que chaque action métier est complète avant d'être visible par les autres utilisateurs.

ACID Transactions



Cohérence (Consistency) et Isolation (Isolation)

- Tandis que l'Atomicité garantit que tout est fait, la Cohérence assure que tout est juste et l'Isolation garantit que personne ne se gêne.

1. Le Pilier C : La Cohérence (Respect des règles)

- La transaction doit faire passer la base d'un état valide à un autre état valide en respectant toutes les règles d'intégrité.
- Intégrité Structurelle : Respect des clés primaires (unicité), clés étrangères (références) et types de données.
- Contraintes Métier (Constraints/Check) : Par exemple, un solde bancaire ne peut pas devenir négatif si une contrainte CHECK ($\text{solde} \geq 0$) existe.
- Responsabilité : Si une règle est violée, le système doit annuler la transaction (Rollback) pour protéger la base.

2. Le Pilier I : L'Isolation (Gestion du multi-utilisateur)

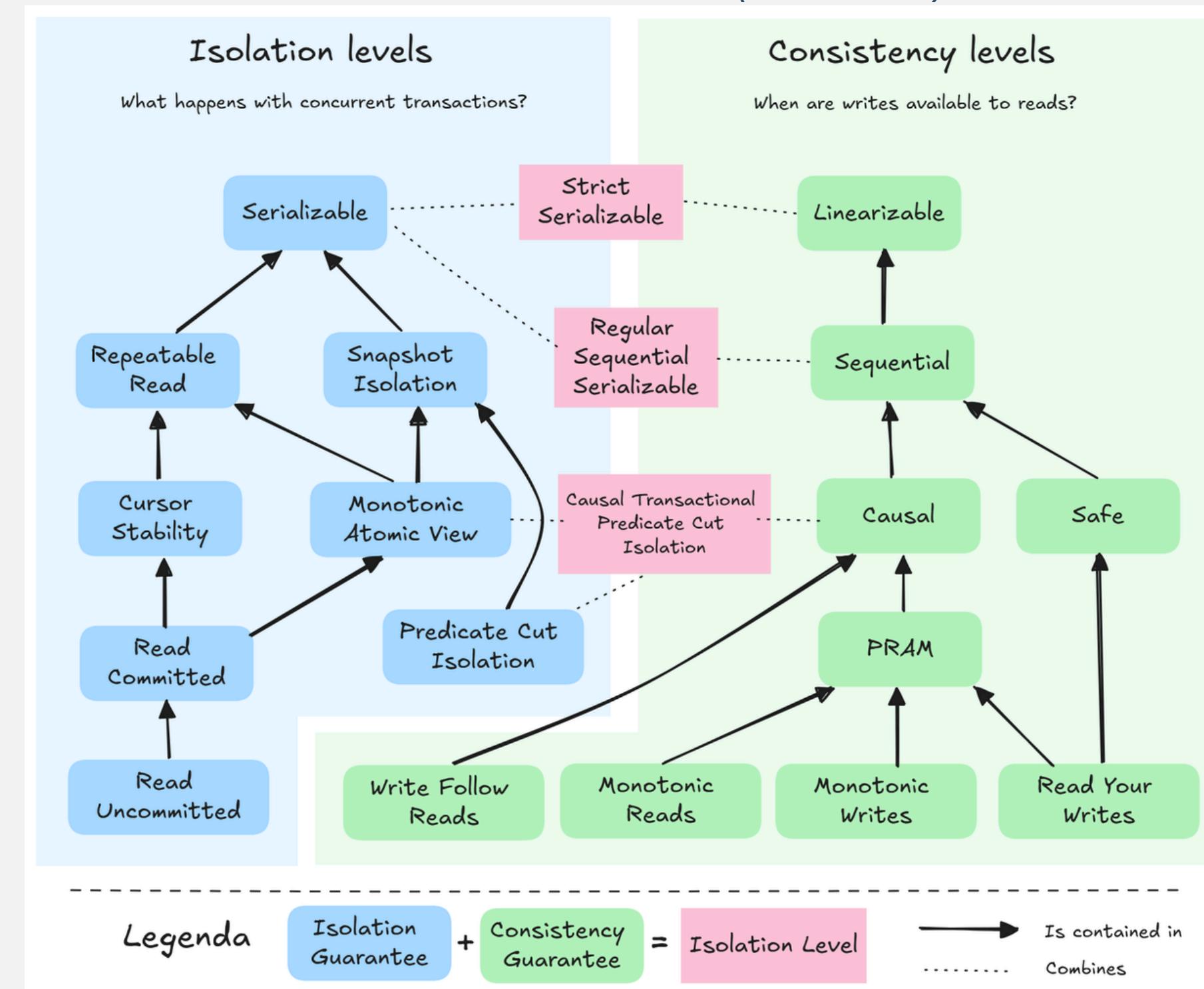
- L'Isolation garantit que l'exécution simultanée de plusieurs transactions produit le même résultat qu'une exécution séquentielle (les unes après les autres).
- Le Principe : Chaque transaction doit avoir l'illusion qu'elle est seule sur la base de données.
- Le Défi : Empêcher les transactions de lire des données intermédiaires (non validées) d'autres transactions concurrentes.
- Mécanismes techniques : Le SGBD utilise des Verrous (Locks) pour bloquer l'accès à une donnée en cours de modification.

3. Pourquoi est-ce complexe ?

- Cohérence : Elle demande une vérification rigoureuse à chaque écriture.
- Isolation : Plus l'isolation est forte, moins le système est performant (car les utilisateurs doivent s'attendre). C'est un arbitrage permanent entre Rigueur et Vitesse.

Illustration des mécanismes d'Isolation et de Cohérence

Figure : Hiérarchie des modèles de cohérence et d'isolation. Ce schéma illustre la complexité des garanties offertes par les SGBD modernes, allant des isolations faibles (Read Uncommitted) aux isolations fortes et strictes (Serializable).



Le Pilier D : La Durabilité (Persistance)

1. Le défi de la mémoire volatile

- Les données en cours de traitement résident en RAM (mémoire vive), qui est ultra-rapide mais s'efface en cas de coupure de courant.
- La Durabilité garantit que le résultat final est écrit sur un support non volatile (Disque dur HDD / SSD).

2. Le mécanisme technique : Le Write-Ahead Logging (WAL)

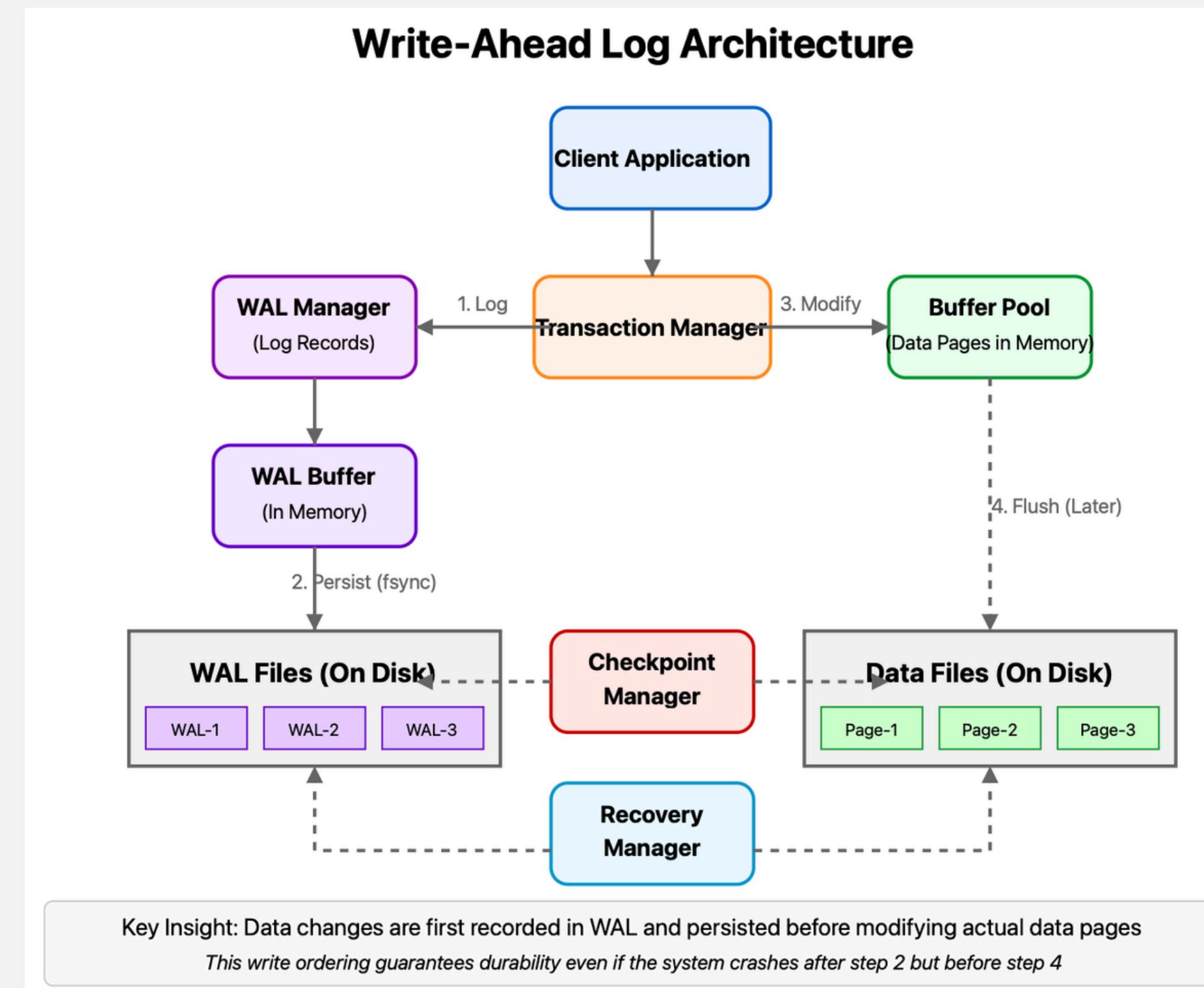
- Pour ne pas ralentir le système, le SGBD n'écrit pas immédiatement dans les fichiers de données (lourd).
- Il écrit d'abord les changements dans un Journal des Transactions (Redo Log) de manière séquentielle et ultra-rapide.
- Règle d'or : Le COMMIT n'est confirmé à l'utilisateur que lorsque l'écriture dans ce journal est sécurisée sur le disque.

3. La Reprise après sinistre (Recovery)

- Si le serveur redémarre après un crash, le SGBD parcourt le journal pour "rejouer" (Redo) les transactions validées qui n'avaient pas encore atteint le fichier de données final.
- Cela assure une intégrité totale même après un incident majeur.

Schéma de Persistance : Le mécanisme du Write-Ahead Logging (WAL)

Figure : Architecture du mécanisme de persistance Write-Ahead Logging (WAL)



Synthèse – Les 4 Piliers du Modèle ACID

Ce tableau récapitule les garanties fondamentales offertes par le modèle ACID pour assurer la fiabilité du système.

Propriété	Définition "En un mot"	Mécanisme / Concept Clé
Atomicité	Indivisibilité	Transaction Logs & Rollback
Cohérence	Validité	Contraintes d'intégrité (PK, FK, CHECK)
Isolation	Indépendance	Verrous (Locks) & Niveaux d'isolation
Durabilité	Permanence	Write-Ahead Logging (WAL)

- Le respect strict de ces quatre propriétés garantit qu'une base de données relationnelle reste une source de vérité fiable, même dans les environnements multi-utilisateurs les plus complexes.

Les Enjeux de la Concurrence

1. Les Anomalies de Lecture

- Lecture Sale (Dirty Read) : Une transaction T1 lit une donnée modifiée par T2, mais T2 n'a pas encore validé (COMMIT). Si T2 échoue (ROLLBACK), T1 a travaillé sur une donnée qui n'a "jamais existé".
- Lecture Non Reproductible : \$T1\$ lit une donnée deux fois, mais entre les deux lectures, T2 a modifié et validé cette donnée. T1 obtient deux résultats différents pour la même ligne.
- Lecture Fantôme (Phantom Read) : T1 effectue une requête sur un ensemble de lignes (ex: prix > 100). Entre-temps, T2 insère une nouvelle ligne qui répond à ce critère. Si T1 relance sa requête, une nouvelle ligne "fantôme" apparaît.

2. L'Anomalie d'Écriture

- Mise à jour Perdue (Lost Update) : Deux transactions lisent la même donnée, la modifient et tentent de l'enregistrer. La seconde écrase la modification de la première sans en avoir conscience.
- Le défi de l'Architecte : Comment autoriser un maximum de parallélisme (performance) tout en garantissant un niveau d'isolation empêchant ces anomalies ?

La Solution – Le Mécanisme de Verrouillage

- Le principe des Verrous (Locks)
- Pour garantir qu'une transaction ne travaille pas sur une donnée en cours de modification, le SGBD utilise des verrous.
C'est un marqueur posé sur une donnée (ligne, table ou page) pour en contrôler l'accès.

1. Les deux types de verrous fondamentaux

- Verrou Partagé (S – Shared Lock) :
 - Utilisé pour les opérations de Lecture.
 - Plusieurs transactions peuvent détenir un verrou S sur la même donnée en même temps.
 - Règle : On peut lire à plusieurs, mais personne ne peut modifier tant qu'un verrou S existe.
- Verrou Exclusif (X – Exclusive Lock) :
 - Utilisé pour les opérations d'Écriture (Insert, Update, Delete).
 - Une seule transaction peut détenir ce verrou.
 - Règle : Tant qu'une transaction possède un verrou X, personne d'autre ne peut ni lire, ni modifier la donnée.

Verrou demandé \ Verrou existant	Aucun	Partagé (S)	Exclusif (X)
Partagé (S)	✓ OK	✓ OK	✗ Attente
Exclusif (X)	✓ OK	✗ Attente	✗ Attente

Le Phénomène du Deadlock (Interblocage)

- Quand le verrouillage mène à l'impasse
- L'utilisation intensive des verrous (Locks) pour garantir l'isolation peut mener à une situation critique : le Deadlock. C'est une étreinte mutuelle où plus aucune transaction ne peut progresser.

1. Définition Technique

- Un Deadlock survient lorsque deux transactions (ou plus) s'attendent mutuellement pour libérer un verrou sur une ressource dont elles ont besoin.
- Il s'agit d'une dépendance circulaire infinie qui paralyse les processus concernés.

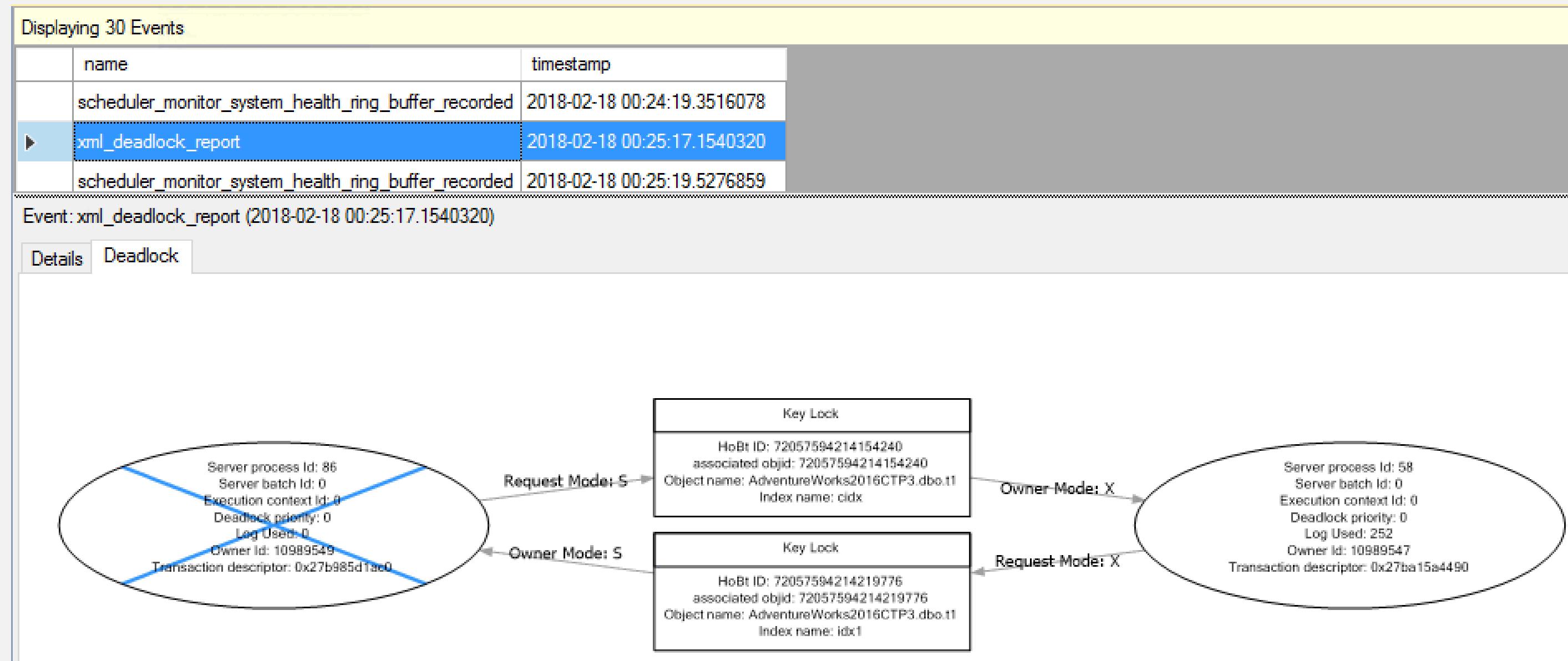
2. Le scénario de "L'étreinte mortelle"

- Transaction T1 possède un verrou exclusif sur la Donnée A et demande un verrou sur la Donnée B.
- Transaction T2 possède un verrou exclusif sur la Donnée B et demande un verrou sur la Donnée A.
- Résultat : T1 attend T2, et T2 attend T1. Sans intervention, l'attente est éternelle.

3. Mécanismes de résolution du SGBD

- Le SGBD ne peut pas empêcher le deadlock, mais il sait le gérer :
- Détection : Le système surveille régulièrement un "Graphe d'attente" (Wait-for Graph) pour repérer les cycles.
- Sélection d'une victime : Le SGBD choisit l'une des transactions (généralement celle ayant effectué le moins de travail) pour être la "victime".
- Rollback : La victime est annulée de force, ses verrous sont libérés, permettant aux autres transactions de finir. La victime devra être relancée plus tard.

Figure : Analyse d'un rapport technique d'interblocage (Deadlock Graph)

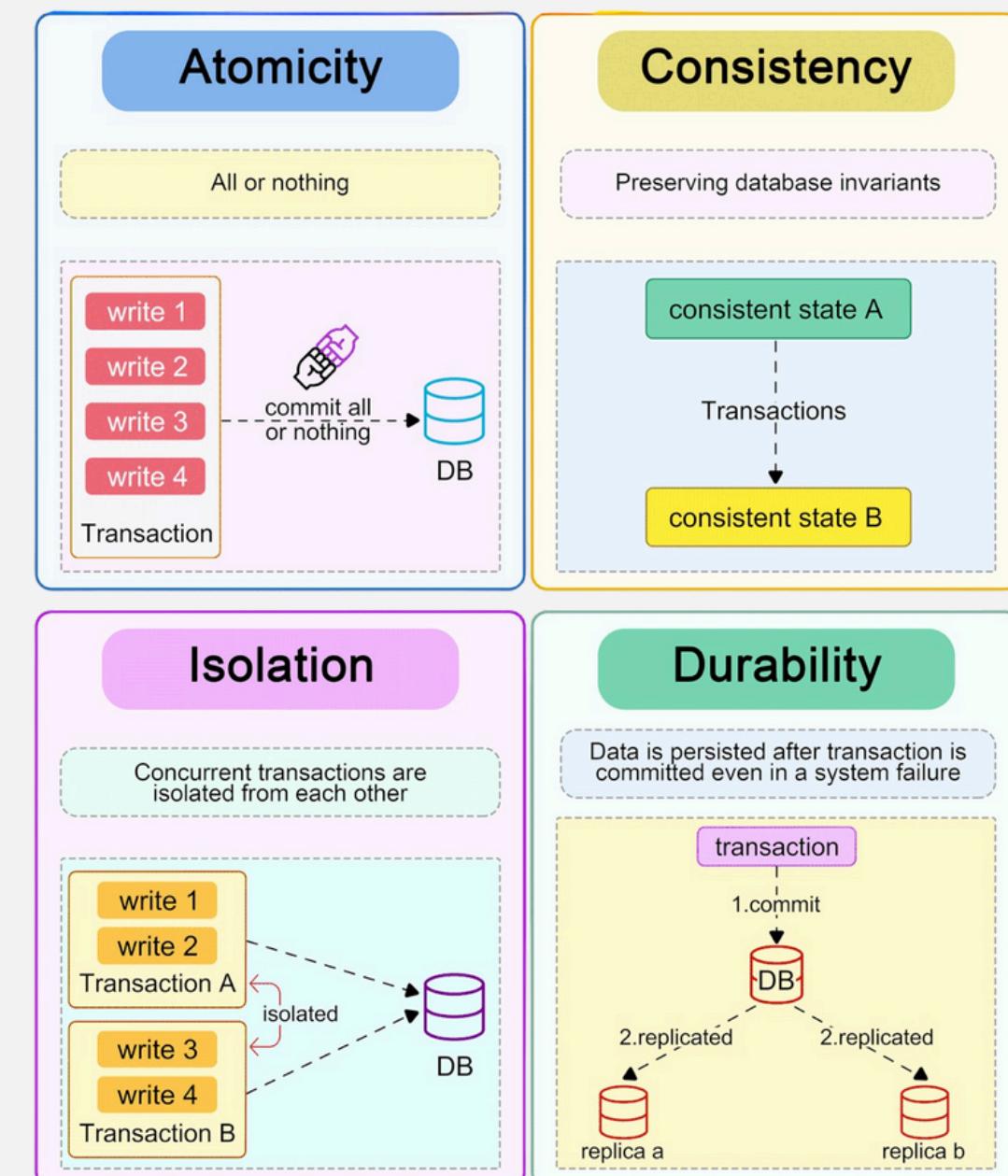


- Cette capture d'écran d'un moniteur système illustre un cycle d'attente réel entre deux transactions. Les flèches indiquent les requêtes de verrous conflictuelles. La croix bleue désigne la transaction 'victime' qui a été automatiquement annulée par le SGBD pour débloquer le système.

Conclusion du Chapitre 5 – La Fiabilité au Service de la Performance

Visualisation des piliers fondamentaux du modèle ACID¹

- Ce qu'il faut retenir :
 - L'ACIDité est le contrat de confiance : Sans Atomicité, Cohérence, Isolation et Durabilité, un système d'information ne peut pas être considéré comme une source de vérité fiable.
 - La gestion de la concurrence est un arbitrage : Le verrouillage (Locking) protège la donnée mais peut impacter la fluidité du système. L'architecte doit choisir le bon niveau d'isolation.
 - La résilience est automatique : Grâce au mécanisme du WAL et à la détection des Deadlocks, les SGBD modernes garantissent la survie des informations même en cas de crise logicielle ou matérielle.
-
- Nous avons maintenant une base de données sûre, robuste et intègre. Cependant, la survie de la donnée n'est que la première étape. Pour qu'une entreprise soit réellement performante, elle doit savoir comment protéger ses accès et optimiser ses traitements.



Chapitre 6 : Administration, Sécurité et Contrôle des Accès

*Garantir la confidentialité et la
gouvernance des données : de la
gestion des privilèges à la protection
contre les menaces extérieures.*

Les Fondements de la Sécurité des Bases de Données

Les Objectifs de la Sécurité (Le Triangle CID)

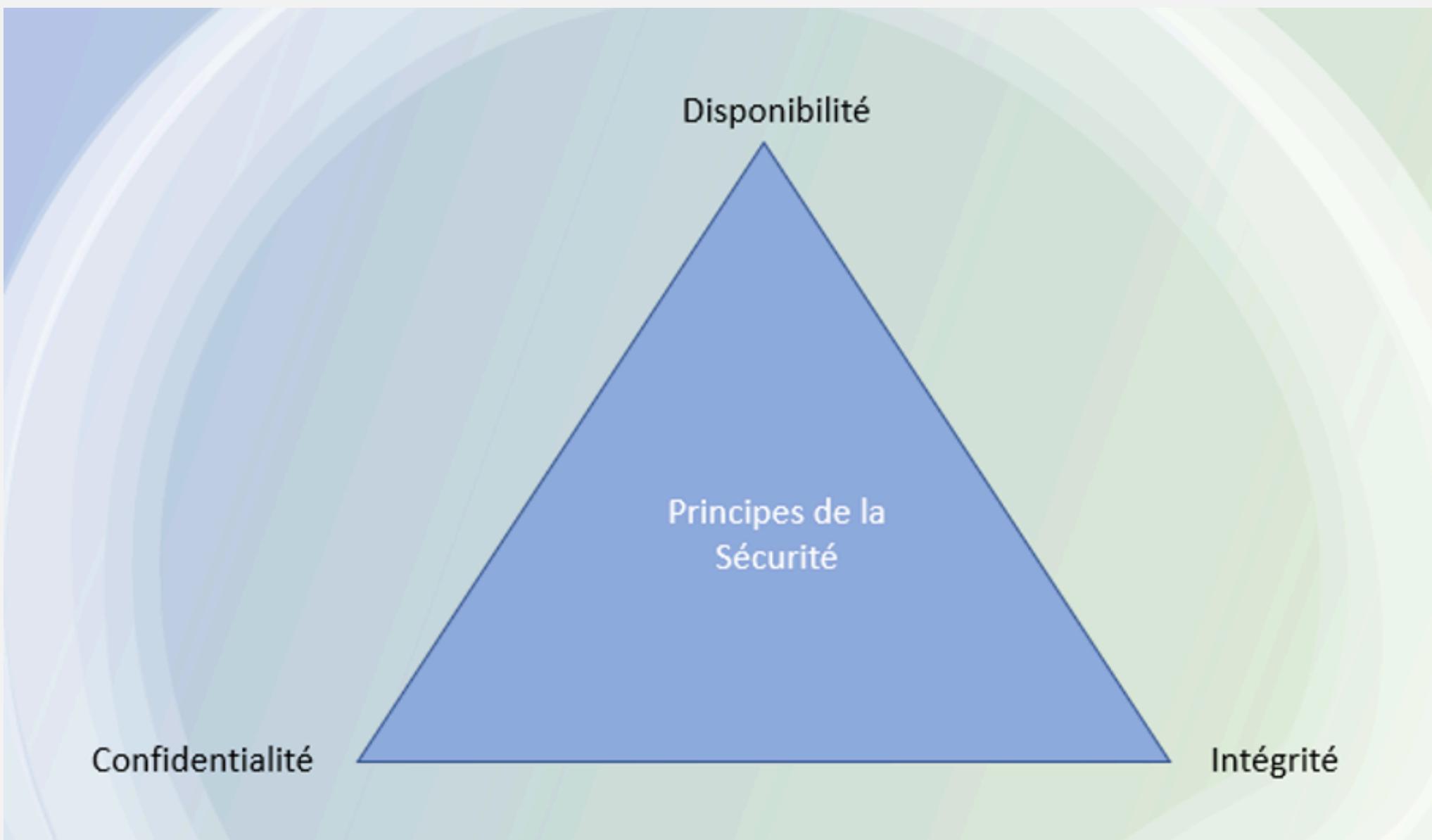
- La sécurité ne se résume pas à un mot de passe ; elle repose sur trois piliers indissociables :
- Confidentialité : Garantir que les données ne sont accessibles qu'aux utilisateurs dûment autorisés.
- Intégrité : Veiller à ce que les données ne soient pas modifiées de manière non autorisée ou accidentelle (lien direct avec la cohérence vue au chapitre 5).
- Disponibilité : Assurer que les utilisateurs autorisés peuvent accéder aux données sans interruption, même en cas d'incident technique.

Le Rôle de l'Administrateur (DBA)

- L'administrateur est le "gardien du temple". Ses responsabilités incluent :
- La gestion des identités : Identification et authentification des utilisateurs.
- Le contrôle d'accès : Définir qui peut faire quoi (Lecture, Modification, Suppression).
- L'audit : Tracer les actions effectuées sur la base pour détecter d'éventuelles anomalies ou tentatives d'intrusion.
- Le Principe du Moindre Privilège

Règle d'or : Un utilisateur ne doit disposer que des droits strictement nécessaires à l'accomplissement de ses tâches, et rien de plus.

Figure : La triade CID, socle de la sécurité des données



Le Modèle AAA : Architecture du Contrôle d'Accès

Maintenant que les objectifs sont fixés (CID), passons au mécanisme concret utilisé par le DBA pour les atteindre. Cette slide est cruciale car elle explique comment le système filtre les accès.

La porte d'entrée du SGBD

Pour accéder à la donnée, un utilisateur doit franchir trois étapes distinctes.

1. Identification & Authentification (Qui êtes-vous ?)

- Identification : Fournir un nom d'utilisateur unique (ex: user_RH).
- Authentification : Prouver cette identité via un facteur :
- Savoir : Mot de passe, code PIN.
- Posséder : Clé USB, token.
- Être : Biométrie (empreinte, visage).

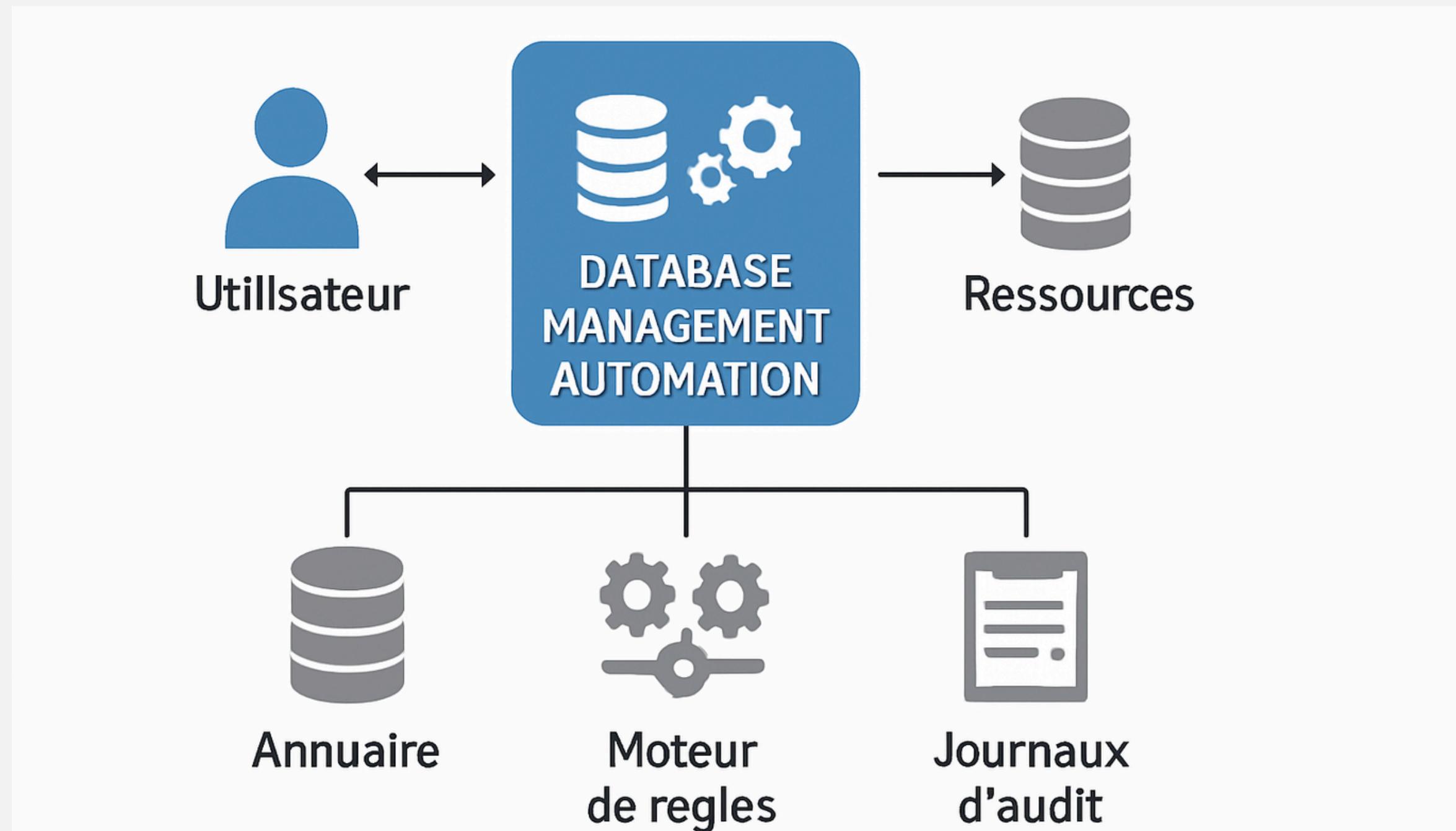
2. Autorisation (Que pouvez-vous faire ?)

- Une fois l'identité vérifiée, le SGBD consulte ses tables de droits :
- Accès limité à certaines tables ou colonnes.
- Restriction des opérations autorisées (ex: SELECT permis, mais DELETE interdit).

3. Audit (Qu'avez-vous fait ?)

- Toute action sur la base est enregistrée dans des journaux d'audit (**Audit Trails**).
- Objectif : Permettre de retracer l'origine d'un incident ou d'une modification suspecte.
- Règle d'Or : L'authentification ouvre la porte, mais l'autorisation définit dans quelle pièce vous avez le droit d'entrer.

Figure : Workflow du contrôle d'accès selon le modèle AAA



Ce diagramme détaille le parcours de sécurisation d'une requête : de la vérification de l'identité (Authentification) à l'application des permissions spécifiques (Autorisation), jusqu'à l'enregistrement de l'activité dans les journaux système (Audit).

La Mise en Œuvre : Les commandes DCL (Data Control Language)

1. La commande GRANT (Donner un privilège)

- Elle permet d'attribuer des droits spécifiques sur un objet de la base.
- Syntaxe : GRANT <privilège> ON <objet> TO <utilisateur>;
- Exemple : GRANT SELECT, UPDATE ON Produits TO 'comptable_1';

2. La commande REVOKE (Retirer un privilège)

- Elle permet de supprimer des droits précédemment accordés.
- Syntaxe : REVOKE <privilège> ON <objet> FROM <utilisateur>;
- Exemple : REVOKE DELETE ON Clients FROM 'stagiaire_info';

3. Les types de privilèges fondamentaux

- DML (Manipulation) : SELECT, INSERT, UPDATE, DELETE.
- DDL (Structure) : CREATE, DROP, ALTER (réservés au DBA).
- Global : ALL PRIVILEGES (pour un administrateur complet).

Cas pratique : Script d'administration des privilèges (MySQL)

```
1 -- DÉMO SÉCURITÉ FST : GRANT & REVOKE
2 -- 1. Préparation de l'environnement
3 • CREATE DATABASE IF NOT EXISTS FST_Demo_Securite;
4 • USE FST_Demo_Securite;
5 -- Création d'une table sensible
6 • CREATE TABLE IF NOT EXISTS Clients (
7     id INT PRIMARY KEY AUTO_INCREMENT,
8     nom VARCHAR(100),
9     statut_financier VARCHAR(50)
10 );
11
12 -- 2. Création des acteurs (cf. Slide 70)
13 -- On nettoie d'abord au cas où ils existent déjà
14 • DROP USER IF EXISTS 'comptable_1'@'localhost';
15 • DROP USER IF EXISTS 'stagiaire_info'@'localhost';
16
17 • CREATE USER 'comptable_1'@'localhost' IDENTIFIED BY 'PassCompta123!';
18 • CREATE USER 'stagiaire_info'@'localhost' IDENTIFIED BY 'PassStage123!';
19
20 -- 3. Attribution des droits (GRANT)
21 -- Le comptable doit pouvoir lire et modifier les infos clients
22 • GRANT SELECT, UPDATE ON FST_Demo_Securite.Clients TO 'comptable_1'@'localhost';
23
24 -- SIMULATION D'ERREUR : On donne trop de droits au stagiaire (droit de supprimer)
25 • GRANT SELECT, DELETE ON FST_Demo_Securite.Clients TO 'stagiaire_info'@'localhost';
26 • SELECT '--- Droits initiaux du stagiaire (TROP ÉLEVÉS) ---' AS Info;
27 • SHOW GRANTS FOR 'stagiaire_info'@'localhost';
28
29 -- 4. Correction des droits (REVOKE)
30 -- L'administrateur retire le droit dangereux de suppression
31 • REVOKE DELETE ON FST_Demo_Securite.Clients FROM 'stagiaire_info'@'localhost';
32
33 =====
34 -- PREUVE FINALE (C'est le résultat important)
35 =====
36 • SELECT '--- Droits finaux du stagiaire (CORRIGÉS) ---' AS Info;
37 -- On vérifie qu'il ne lui reste que le SELECT
38 • SHOW GRANTS FOR 'stagiaire_info'@'localhost';
```

Cette capture montre l'application du script d'administration : après une erreur volontaire d'attribution (DELETE), la commande REVOKE est utilisée pour rétablir la sécurité. Le résultat final (SHOW GRANTS) confirme que les droits sont limités au strict nécessaire.

Vérification du Retrait des Privilèges (MySQL Output)



Output			
#	Time	Action	Message
6	03:54:05	CREATE USER 'comptable_1'@localhost IDENTIFIED BY 'PassCompta123!'	0 row(s) affected
7	03:54:05	CREATE USER 'stagiaire_info'@localhost IDENTIFIED BY 'PassStage123!'	0 row(s) affected
8	03:54:05	GRANT SELECT, UPDATE ON FST_Demo_Securite.Clients TO 'comptable_1'@localhost'	0 row(s) affected
9	03:54:05	GRANT SELECT, DELETE ON FST_Demo_Securite.Clients TO 'stagiaire_info'@localhost'	0 row(s) affected
10	03:54:05	SELECT --- Droits initiaux du stagiaire (TROP ÉLEVÉS) --- AS Info LIMIT 0, 1000	1 row(s) returned
11	03:54:05	SHOW GRANTS FOR 'stagiaire_info'@localhost	2 row(s) returned
12	03:54:05	REVOKE DELETE ON FST_Demo_Securite.Clients FROM 'stagiaire_info'@localhost	0 row(s) affected
13	03:54:05	SELECT --- Droits finaux du stagiaire (CORRIGÉS) --- AS Info LIMIT 0, 1000	1 row(s) returned
14	03:54:05	SHOW GRANTS FOR 'stagiaire_info'@localhost	2 row(s) returned

- Validation du succès : Les indicateurs d'état (coches vertes) confirment que l'intégralité du script d'administration a été traitée avec succès par le serveur.
- Traçabilité de l'action (Flèche) : La ligne 12 valide l'application réelle de la commande REVOKE, neutralisant ainsi la faille de sécurité simulée sur le compte stagiaire.
- Audit final : La dernière étape (SHOW GRANTS) génère un rapport de contrôle permettant au DBA de s'assurer que seuls les privilèges prévus sont actifs.

La Gestion Avancée par Rôles (RBAC)

1. Le concept de Rôle

- Au lieu de donner des droits à chaque personne (chronophage), on crée des groupes de privilèges appelés Rôles (ex: ROLE_FINANCE, ROLE_DEV).

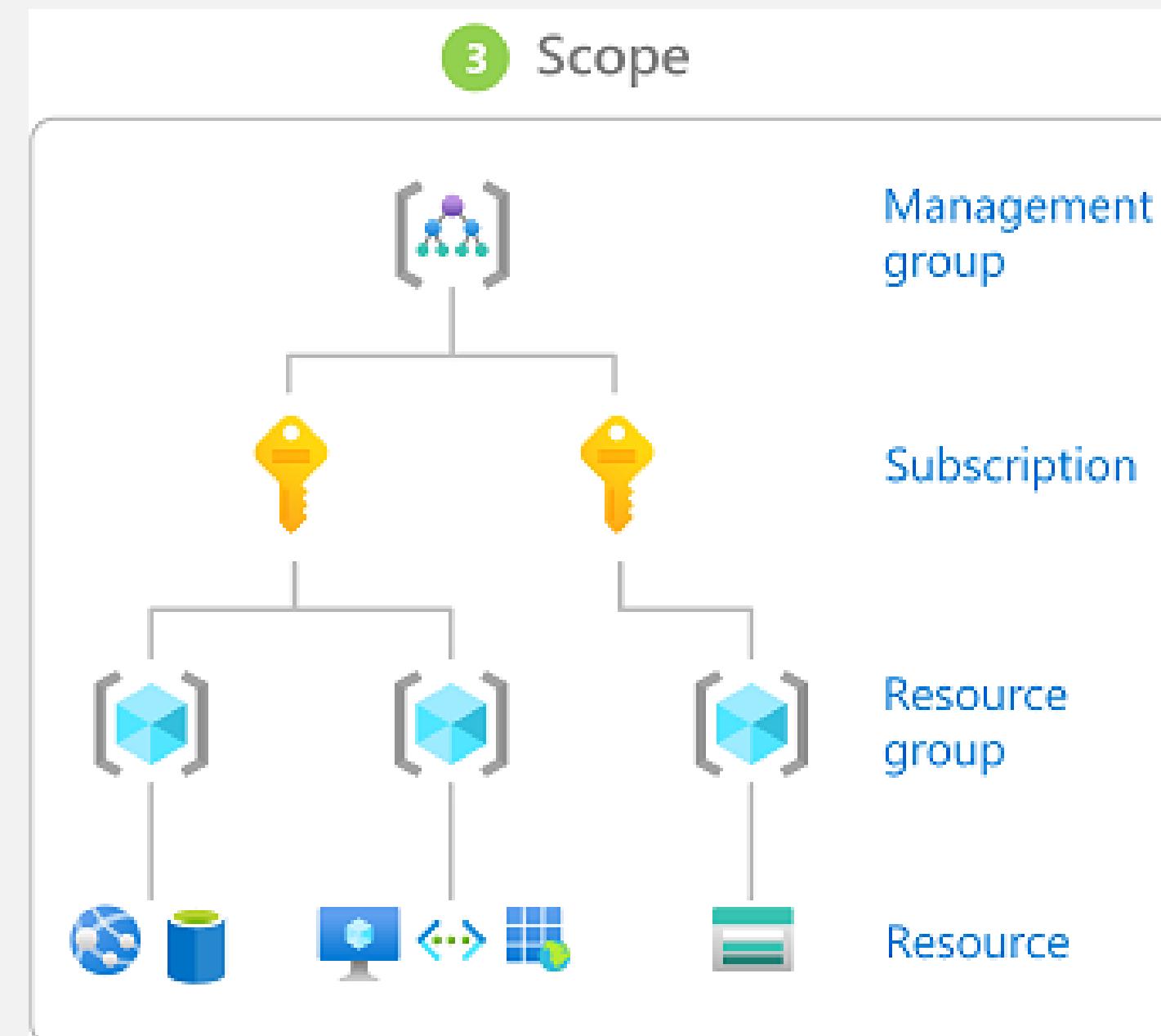
2. Avantages pour le DBA

- Maintenance : Si on ajoute une table, on modifie le rôle une seule fois pour tout le monde.
- Sécurité : Lors d'une nouvelle embauche, il suffit d'affecter l'utilisateur au rôle correspondant pour qu'il ait instantanément ses droits.

3. Exemple SQL :

- CREATE ROLE 'gestionnaire_produits';
- GRANT SELECT, INSERT ON Stock.* TO 'gestionnaire_produits';
- GRANT 'gestionnaire_produits' TO 'comptable_1', 'stagiaire_info';

Figure : Architecture du contrôle d'accès basé sur les rôles (RBAC)



Ce schéma illustre le mécanisme d'héritage des privilèges : les permissions sont regroupées au sein de rôles spécifiques, facilitant ainsi une gestion centralisée et sécurisée des accès pour un grand nombre d'utilisateurs.

Synthèse du Chapitre 6 – De la Sécurité à la Gouvernance

- Ce qu'il faut retenir :
- Le Triangle CID : La sécurité n'est pas qu'une question de technique, c'est un équilibre permanent entre Confidentialité, Intégrité et Disponibilité.
- La barrière AAA : Tout accès repose sur le triptyque Identification, Autorisation et Audit. Sans traçabilité (Audit Trails), le système reste vulnérable.
- La maturité administrative : Passer de la commande unitaire (GRANT) à une gestion par Rôles (RBAC) permet de sécuriser le système à grande échelle et de limiter les erreurs humaines.

Une fois l'infrastructure sécurisée et les accès contrôlés, nous pouvons passer à l'exploitation analytique de ces données : le Chapitre 7 : Data Warehouse.

Chapitre 7 : Data Warehouse (Entrepôts de Données)

Passer de l'exploitation opérationnelle au pilotage stratégique : l'art de transformer la donnée brute en intelligence décisionnelle.

Définition et Caractéristiques Fondamentales

- Pour bien débuter ce chapitre, il faut comprendre qu'un Data Warehouse (DWH) n'est pas une simple base de données volumineuse, mais un système optimisé pour la lecture et l'analyse massive.

1. La Définition de Référence (Bill Inmon)

- « Un entrepôt de données est une collection de données orientées sujet, intégrées, non volatiles et historisées, organisée pour le support du processus de décision. »

2. Les 4 Piliers du Data Warehouse

- Orienté Sujet : Contrairement à tes chapitres précédents (gestion des ventes ou des stocks par exemple), les données ici sont regroupées par thèmes métiers pour faciliter l'analyse globale.
- Intégré : Les données provenant de sources disparates (MySQL, fichiers CSV, ERP) sont nettoyées et uniformisées (ex: unifier les formats de dates) avant d'être stockées.
- Non Volatile : Une donnée entrée dans le DWH ne change jamais. Elle est stockée de manière permanente, ce qui garantit la fiabilité des rapports d'une année sur l'autre.
- Historisé : Alors qu'une BDD opérationnelle affiche l'état instantané (ex : stock actuel), le DWH conserve l'historique complet sur 5 ou 10 ans pour détecter des tendances.

Le Cœur du Système : Le Processus ETL

- L'ETL est la passerelle technique qui permet de transformer les données opérationnelles (vues dans les chapitres précédents) en informations exploitables pour l'analyse.

1. Extraction (Extraire)

- Récupération des données brutes depuis les sources hétérogènes (bases MySQL, fichiers Excel, logs serveurs).
- Objectif : Isoler les données nécessaires sans perturber la production.

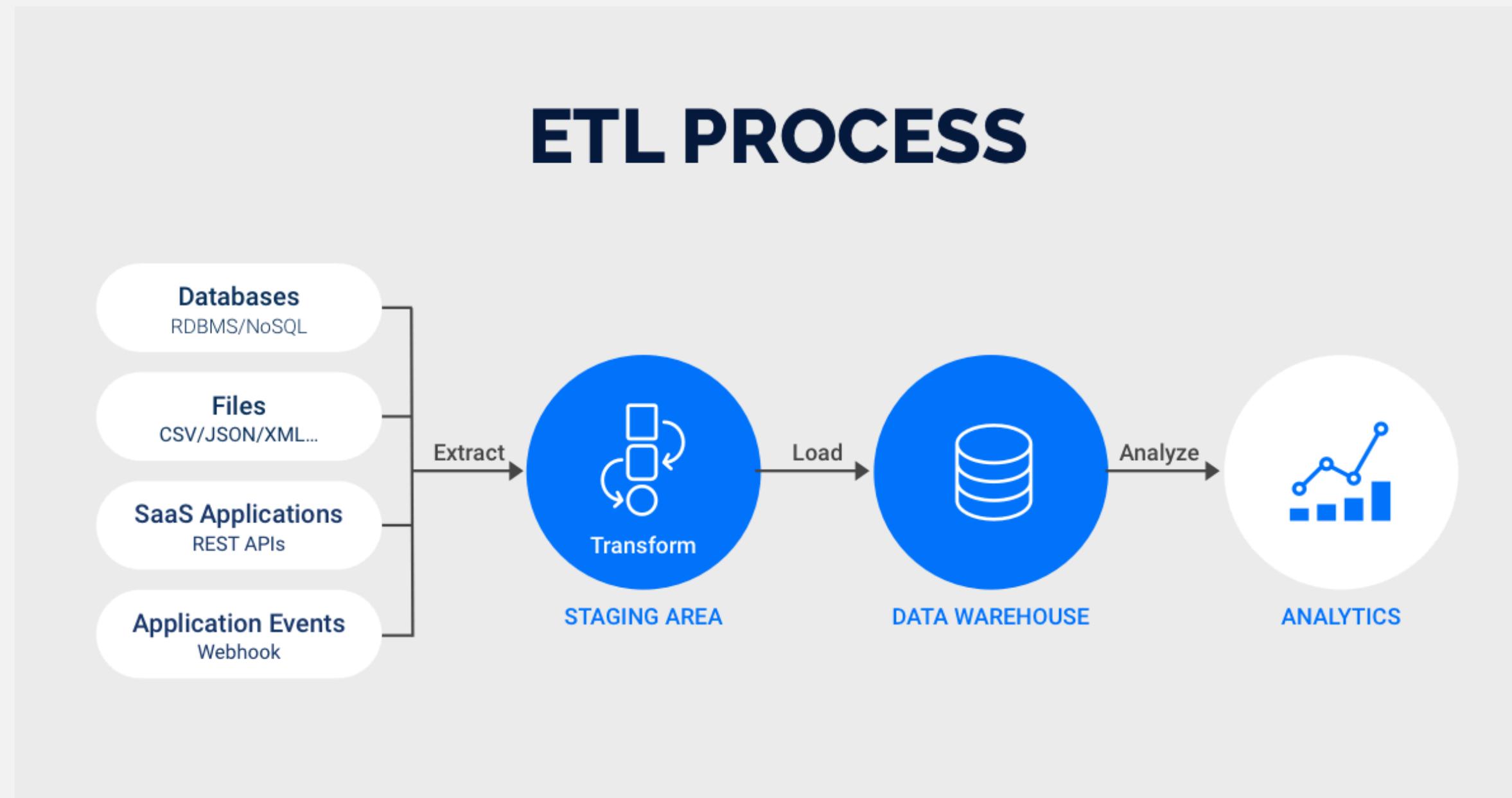
2. Transformation (Transformer)

- C'est l'étape la plus critique :
- Nettoyage : Suppression des doublons et correction des erreurs.
- Standardisation : Harmoniser les formats (ex: transformer "M" et "Masculin" en "1").
- Calculs : Création d'indicateurs (ex: CA = Prix * Quantité).

3. Chargement (Charger)

- Intégration finale des données propres dans les tables de l'entrepôt (DWH).

Figure : Architecture du processus d'alimentation ETL (Extract, Transform, Load)



Ce schéma représente le pipeline de données : l'extraction depuis les sources opérationnelles, la phase de transformation pour le nettoyage et l'harmonisation, et enfin le chargement dans l'entrepôt de données centralisé.

Organisation des Données : Le Schéma en Étoile (Star Schema)

1. Le Concept Central : La Table de Faits

- Située au cœur du schéma, elle contient les données quantitatives et mesurables (ex: montant_vente, quantité_stock, prix_total).
- Elle enregistre les événements métiers clés de l'entreprise.

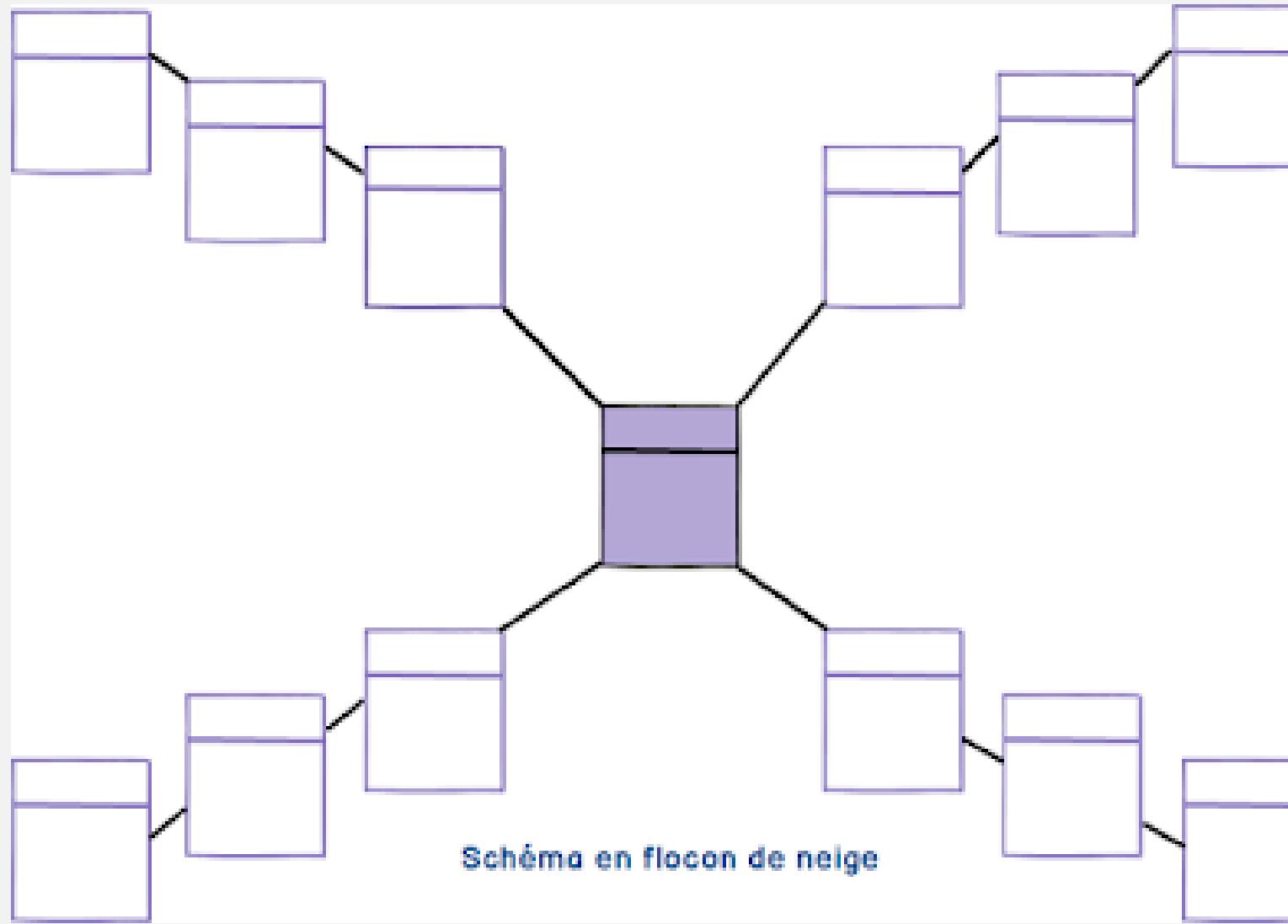
2. Les Axes d'Analyse : Les Tables de Dimensions

- Elles entourent la table de faits.
- Elles contiennent les attributs descriptifs permettant de filtrer ou de regrouper les données (ex: Temps, Produit, Magasin, Client).

3. Pourquoi ce choix ?

- Simplicité : Les requêtes SQL sont plus simples à écrire.
- Performance :

Figure : Architecture d'un Schéma en Flocon (Snowflake Schema)

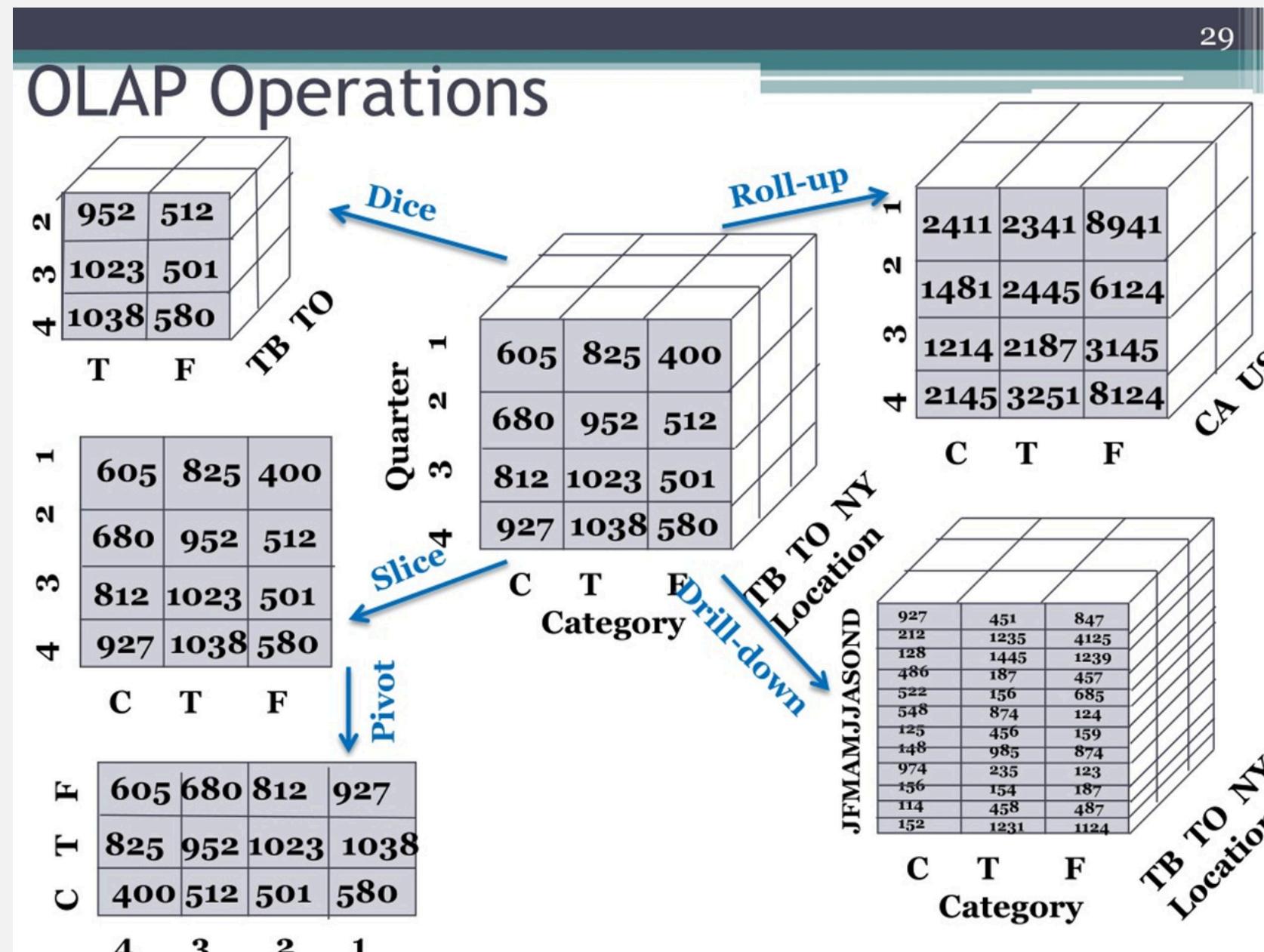


Ce schéma illustre la normalisation des tables de dimensions : chaque axe d'analyse est décomposé en sous-niveaux hiérarchiques. Cette structure réduit la redondance des données et optimise l'espace disque, bien qu'elle complexifie légèrement les jointures SQL par rapport au modèle en étoile.

Analyse Multidimensionnelle : Le Cube OLAP

- Concept du Cube : C'est une structure de données qui permet d'analyser les mesures (faits) selon plusieurs axes (dimensions) simultanément.
- Les bénéfices pour l'entreprise :
- Vitesse : Les agrégations sont précalculées, permettant des réponses quasi instantanées sur des volumes massifs.
- Flexibilité : L'utilisateur peut changer d'angle de vue sans avoir à réécrire de requêtes complexes.
- Les Opérations de Navigation :
- Drill-down : Passer d'une vue globale à une vue détaillée (ex : Année -> Mois).
- Roll-up : Synthétiser les données pour une vue d'ensemble (ex : Ville ->Pays).
- Slice & Dice : Isoler une portion spécifique du cube pour une analyse ciblée.

Figure : Opérations de navigation multidimensionnelle sur un Cube OLAP



Ce schéma illustre les manipulations dynamiques des données : le Roll-up pour agréger l'information (montée en hiérarchie), le Drill-down pour détailler les mesures (descente en hiérarchie) et le Slicing pour isoler une tranche spécifique d'analyse.

Au-delà du SQL : Le Big Data & Data Lake

- Le Problème : Le DW classique est limité aux données structurées.
- Le Data Lake : Un réservoir central pour stocker des données brutes (Logs, IoT, Vidéos) sans schéma prédéfini (Schema-on-Read).
- Outils Clés : * Hadoop (HDFS) : Pour le stockage distribué.
- Apache Spark : Pour le traitement ultra-rapide en mémoire.
- Synergie : Le DW sert au reporting précis, le Data Lake sert au Machine Learning.

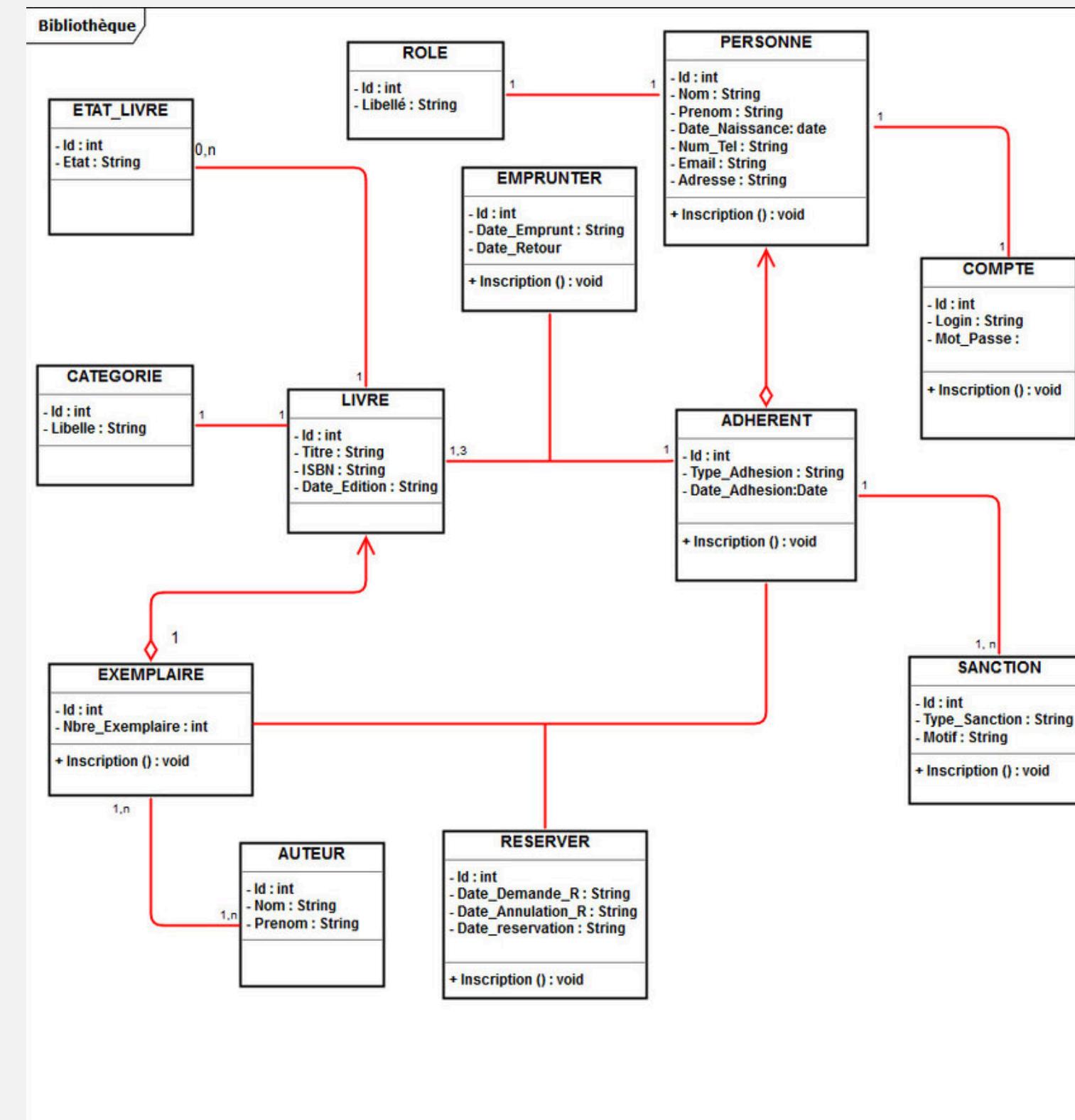
Chapitre 8 – Projet Intégrateur

*De la théorie à l'ingénierie :
Synthèse et mise en œuvre pratique
d'un système de gestion de données
complet.*

Présentation du Système de Gestion de Librairie

- Le projet consiste à concevoir l'architecture backend complète d'une plateforme de gestion de livres et d'emprunts, articulée autour de quatre modules majeurs :
1. Gestion du Catalogue (Ressources)
 - Inventaire détaillé des ouvrages (Titres, Auteurs, ISBN, Catégories).
 - Suivi de l'état physique et de la disponibilité des exemplaires.
 2. Gestion des Adhérents (Utilisateurs)
 - Administration des profils (Coordonnées, historique d'activité).
 - Gestion des niveaux d'accès (Utilisateurs simples vs Administrateurs).
 3. Gestion des Flux (Cœur Métier)
 - Enregistrement des emprunts et des retours en temps réel.
 - Calcul automatique des délais et gestion des disponibilités.
 4. Analyse et Évaluation (Qualité)
 - Système de notation et d'avis sur les ouvrages.
 - Traçabilité des mouvements pour l'aide à la décision.

Figure : Architecture Fonctionnelle du Système de Gestion de Librairie



Ce schéma illustre l'interconnexion entre les quatre piliers du projet. On y voit comment le système centralise la gestion du catalogue et des adhérents pour alimenter le moteur de transactions (emprunts/retours), tout en générant les données nécessaires au module d'analyse décisionnelle.

Une mise en pratique intégrale des Chapitres 1 à 7

1. Modélisation & SQL (Ch. 1 à 4)

- Objectif : Mise en œuvre du schéma relationnel normalisé présenté à la slide précédente.
- Action : Création d'une structure de tables robuste garantissant l'intégrité référentielle et l'absence de redondances.

2. Fiabilité & Propriétés ACID (Ch. 5)

- Objectif : Sécuriser les processus critiques du métier (Emprunts / Retours).
- Action : Utilisation des Transactions pour garantir que le changement d'état d'un livre et la création de l'emprunt se font de manière atomique et cohérente.

3. Sécurité & Gouvernance (Ch. 6)

- Objectif : Appliquer le principe du moindre privilège.
- Action : Configuration d'une gestion d'accès par rôles (RBAC) permettant de différencier les droits des adhérents de ceux des administrateurs système.

4. Vision Décisionnelle (Ch. 7)

- Objectif : Transformer les données d'emprunt en informations stratégiques.
- Action : Création de Vues SQL complexes simulant un mini-entre�ot de données pour l'analyse des tendances (ex: catégories les plus populaires).

Les Règles du Système : Le "Cerveau" de la Base de Données

- Gestion des Emprunts :
- Un livre ne peut être emprunté que s'il est en état "Disponible".
- Un adhérent ne peut pas dépasser un nombre maximum d'emprunts simultanés.
- Automatisation (Triggers) :
- Mise à jour automatique du stock lors d'un retour.
- Génération d'une alerte ou d'une sanction en cas de retard.
- Intégrité des Données :
- Interdiction de supprimer un auteur s'il possède encore des livres référencés dans le catalogue.

Validation du Système : Scénarios d'Utilisation

- Pour prouver la fiabilité de la base, le projet s'appuie sur des scénarios concrets qui seront détaillés dans le compte-rendu :
- Le Scénario nominal : Un adhérent emprunte un livre, le stock diminue, puis le rend, le stock remonte.
- Le Test de Sécurité : Tentative de modification du catalogue par un utilisateur n'ayant pas le rôle "Administrateur" (vérification du RBAC).
- Le Test de Cohérence (ACID) : Simulation d'une coupure durant une transaction pour vérifier que les données restent propres.
- Le Test Analytique : Génération d'un rapport sur les catégories les plus populaires via les vues SQL.

Livrables du Projet Intégrateur

- Cette slide annonce au professeur les documents qu'il trouvera dans le dossier séparé :
- Dossier Scripts (.sql) :
- Schema_Creation.sql (Structure DDL).
- Data_Insertion.sql (Données de test).
- Business_Logic.sql (Triggers, Procédures, Vues).
- Security_Setup.sql (Rôles et Privilèges).
- Le Compte-rendu Technique :
- Explication détaillée de chaque script.
- Captures d'écran des tests réussis.
- Justifications des choix de modélisation.

Conclusion Générale & Perspectives

- Maîtrise de la chaîne de valeur de la donnée : Ce dossier a permis de couvrir l'intégralité du cycle de vie d'un système d'information, de la modélisation conceptuelle à l'analyse décisionnelle.
- De la théorie à l'ingénierie : L'application des concepts avancés (ACID, RBAC, ETL) sur le projet "BookFlow" a démontré la viabilité technique des solutions proposées.
- Professionnalisme et Rigueur : La séparation claire entre exploitation opérationnelle et pilotage stratégique prépare le terrain pour des infrastructures de données modernes et évolutives.

Intégration SI : API REST & JSON

- Principe : La base de données ne vit pas seule. Elle communique avec le Front-end via une couche de services.
- Format JSON : Standard léger d'échange de données.
- Sécurisation (JWT) : Authentification par jetons pour protéger les accès aux données via le Web.
- Flux : Requête React/Next.js -> API Node.js -> SQL Query -> Réponse JSON.

Mise en Œuvre : Logique Métier (SQL & Triggers)

- Exemple : Trigger de contrôle des emprunts

```
4  CREATE PROCEDURE sp_effectuer_emprunt(
5      IN p_id_adherent INT,
6      IN p_id_exemplaire INT
7  )
8  BEGIN
9      DECLARE v_nb_emprunts INT;
10
11      -- 1. Vérification du quota (Règle métier : Max 5 livres)
12      SELECT COUNT(*) INTO v_nb_emprunts
13      FROM EMPRUNTER
14      WHERE Id_Adherent = p_id_adherent AND Date_Retour_Effectif IS NULL;
15
16      -- 2. Logique de décision
17      IF v_nb_emprunts >= 5 THEN
18          -- Déclenchement d'une erreur personnalisée
19          SIGNAL SQLSTATE '45000'
20          SET MESSAGE_TEXT = 'Erreur : Cet adhérent a atteint son quota d''emprunts (5).';
21      ELSE
22          -- 3. Validation de la transaction
23          INSERT INTO EMPRUNTER (Date_Emprunt, Date_Retour_Prevue, Id_Adherent, Id_Exemplaire)
24          VALUES (NOW(), DATE_ADD(NOW(), INTERVAL 14 DAY), p_id_adherent, p_id_exemplaire);
25
26          -- Mise à jour automatique de l'état du livre
27          UPDATE EXEMPLAIRE SET Code_Etat = 'Indisponible' WHERE Id_Exemplaire = p_id_exemplaire;
28      END IF;
29  END //
```

Perspectives d'Évolution & Scalabilité



Bien que ce projet se soit concentré sur la robustesse et l'intelligence du moteur SQL, il constitue le socle indispensable d'une future application Full-Stack. L'objectif à court terme est d'intégrer cette architecture de données à une interface utilisateur moderne en Next.js via une API sécurisée, tout en explorant la scalabilité Cloud et le reporting décisionnel avancé. Cette approche modulaire garantit que le système de gestion de librairie reste performant, évolutif et prêt pour une mise en production réelle, transformant ainsi la théorie des bases de données en une solution logicielle complète.

Merci