



Systèmes d'Information et Bases de Données

Chapitre 1: La notion de base de données (BD)

Pr. Sara El-Ateif, 12 Février 2025

À propos de moi

Sara El-Ateif

- Maître de Conférences en Big data et Intelligence Artificielle
- Docteur d'Etat en Informatique
- AI & ML Google Developer Expert
- Google PhD Fellow
- Google Cloud Champion Innovator
- NVIDIA Deep Learning Institute Instructor

Email: sara_elateif@um5.ac.ma



Introduction et Concepts Fondamentaux

Définition

Qu'est-ce que ...?

- Système?
- Information?
- Base?
- Donnée ?

Définition

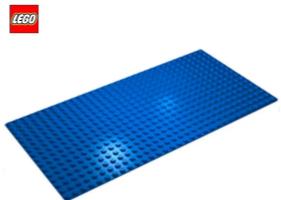
Qu'est-ce que ...?



Donnée



Information



Base



Système

Définition

Qu'est-ce que Donnée?

- **Donnée**

- Les données sont des **faits bruts**, non traités et non organisés.
- Elles existent sous forme de nombres, de symboles, de caractères, d'images, etc.
- Elles n'ont aucune signification en soi.

Exemple : La température enregistrée à un endroit précis à un moment précis (25°C, 14h00, 22/09/2023).

Définition

Qu'est-ce qu'information ?

- **Information**

- L'information est constituée de **données structurées, organisées et contextualisées**, ce qui leur donne une signification.
- Elle répond aux questions "Qui", "Quoi", "Où", "Quand".

Exemple : La température moyenne à Paris en septembre est de 25°C.

Définition

Qu'est-ce qu'un Système?

- **Système:** un système est un ensemble d'éléments interconnectés et interagissant entre eux pour accomplir un objectif précis. Il est caractérisé par :
 - **Des éléments :** composants individuels du système.
 - **Des interactions :** relations et échanges entre les éléments.
 - **Un objectif :** but ou fonction du système.
 - **Des frontières :** ce qui délimite le système de son environnement.

Exemple : système digestif, système solaire, système informatique.

Références :

- Le Moigne, J. L. (1990). La théorie du système général. Théorie de la modélisation. Paris: PUF.
- Bertalanffy, L. von (1968). General System Theory: Foundations, Development, Applications. New York: George Braziller.

Définition

Qu'est-ce qu'une Base?

- **Base**

- ▶ Dans le contexte des bases de données, "base" fait référence au fondement, au socle sur lequel repose l'information. Il s'agit du support physique ou logique du stockage des données.

Exemple : un disque dur, un serveur, une structure de données spécifique (table, arbre, etc.).

Définition

Qu'est-ce qu'un Système d'information?

- **Les Systèmes d'Information (SI)**

- Un Système d'Information (SI) est un ensemble organisé de ressources (humaines, matérielles, logicielles) qui permettent de collecter, stocker, traiter et diffuser l'information au sein d'une organisation. Le but d'un SI est d'aider à la prise de décision et à la gestion efficace des activités.

Exemple : le SI d'une université gère les inscriptions, les notes, les emplois du temps, etc.

Définition

Qu'est-ce qu'une Base de données?

- **Base de Données (BD)**

- Une base de données est une collection organisée de données inter-reliées, stockées de manière persistante et accessible à plusieurs utilisateurs de manière concurrente. L'objectif principal est de fournir un accès efficace à l'information.

Analogie : Un répertoire téléphonique bien structuré.

La Base de Données au Cœur du SI

Rôle des Bases de Données

- La base de données est la composante centrale d'un SI. Elle stocke de manière structurée et persistante les données nécessaires au fonctionnement du SI et de ses applications.

Exemple : Dans le SI d'une université, la BD stocke les informations sur les étudiants, les cours, les professeurs, etc.

Fonctions Principales d'une BD

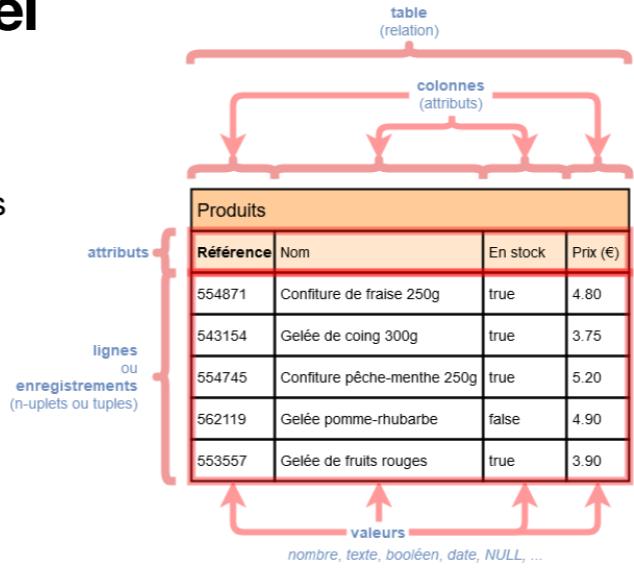
À quoi sert une BD?

- **Stockage persistant** (analogie : mémoire vs. disque dur).
- **Accès efficace** (analogie : recherche dans un dictionnaire vs. dans un texte non structuré).
- **Partage de données** (exemple : réservation de billets d'avion).
- **Intégrité** (analogie : règles de validation d'un formulaire).
- **Sécurité** (analogie : accès sécurisé à un bâtiment).
- **Minimiser la redondance** (exemple : fiche client unique vs. informations dispersées).

Le Modèle Relationnel

Organiser les données

Le modèle relationnel est le modèle le plus utilisé pour organiser les données dans une BD. Il structure les données en tables (relations), composées de lignes (enregistrements ou tuples) et de colonnes (attributs ou champs).



Source: <https://info.blaisepascal.fr/nsi-modele-relationnel/>

Le Modèle Relationnel

Concepts Clés du Modèle Relationnel

Domaine : ensemble de valeurs admissibles pour un attribut (analogie : type de données en programmation).

- **Exemple:** entier, texte, date.

Schéma relationnel : description formelle de la structure de la base, incluant les noms des tables, leurs attributs et leurs domaines (analogie : plan d'architecte).

- **Exemple:** ETUDIANT(NumEtudiant: Entier, Nom: Texte, DateNaissance: Date).

Clé primaire : attribut (ou groupe d'attributs) qui identifie de manière unique chaque tuple d'une table (analogie : numéro de sécurité sociale).

Le Modèle Relationnel

Concepts Clés du Modèle Relationnel

Clé étrangère : attribut dans une table qui fait référence à la clé primaire d'une autre table (analogie : lien hypertexte).

- **Exemple:** Dans la table INSCRIPTION, NumEtudiant est une clé étrangère référençant la table ETUDIANT.

Contraintes d'intégrité : règles pour garantir la cohérence et la validité des données (analogie: règles de validation dans un formulaire).

- **Exemple:** Un numéro d'étudiant ne peut pas être nul.

Le Modèle Relationnel

Exemple: Système de Notation

Exercice interactif :

- Concevoir un schéma relationnel simple pour la gestion des notes des étudiants.
- Tables : Etudiants, Matières, Notes.
- Identifier les clés primaires, les clés étrangères, les domaines et les contraintes d'intégrité.

Voici une solution pour le schéma relationnel de gestion des notes, avec explications, contraintes d'intégrité et alternatives :

Schéma relationnel:

- Etudiants (NumEtudiant, Nom, Prénom, DateNaissance, Adresse)**
 - NumEtudiant : Entier (INT), Clé Primaire
 - Nom : Texte (VARCHAR(255)), Non Null
 - Prénom : Texte (VARCHAR(255)), Non Null
 - DateNaissance : Date (DATE)
 - Adresse : Texte (VARCHAR(255))
- Matières (CodeMatiere, NomMatiere, Coefficient)**
 - CodeMatiere : Texte (VARCHAR(10)), Clé Primaire
 - NomMatiere : Texte (VARCHAR(255)), Non Null
 - Coefficient : Décimal (DECIMAL)
- Notes (NumEtudiant, CodeMatiere, Note, DateEvaluation)**
 - NumEtudiant : Entier (INT), Clé Étrangère référençant Etudiants(NumEtudiant)
 - CodeMatiere : Texte (VARCHAR(10)), Clé Étrangère référençant Matières(CodeMatiere)
 - Note : Décimal (DECIMAL)
 - DateEvaluation: Date (DATE)
 - Clé primaire : (NumEtudiant, CodeMatiere, DateEvaluation)

Domaines:

- NumEtudiant : Entiers positifs.
- CodeMatiere : Chaînes de caractères alphanumériques.
- Nom, Prénom, NomMatiere, Adresse : Chaînes de caractères.
- DateNaissance, DateEvaluation: Dates.
- Coefficient, Note : Nombres décimaux. Le domaine de Note pourrait être contraint ; par exemple, entre 0 et 20, ou 0 et 100.

⚠️ Contraintes d'intégrité:

- **Clé primaire unique:** Chaque NumEtudiant dans la table Etudiants, chaque CodeMatiere dans la table Matières, et chaque combinaison (NumEtudiant, CodeMatiere, DateEvaluation) dans la table Notes, doivent être uniques.
- **Clés étrangères valides:** Chaque NumEtudiant dans la table Notes doit exister dans la table Etudiants. Chaque CodeMatiere dans la table Notes doit exister dans la table Matières.
- **Valeurs non nulles:** Nom et Prénom dans la table Etudiants, et NomMatiere dans la table Matières ne peuvent pas être nuls. On peut aussi ajouter Note et DateEvaluation dans Notes comme Non Null, selon les besoins.
- **Contraintes de domaine:** S'assurer que les valeurs des attributs respectent les domaines définis (ex: Note entre 0 et 20).

💡 Explications:

- **Table Etudiants:** Contient les informations sur chaque étudiant. NumEtudiant est la clé primaire pour identifier de manière unique chaque étudiant.
- **Table Matières:** Contient les informations sur chaque matière. CodeMatiere est la clé primaire.
- **Table Notes:** Associe une note à un étudiant pour une matière donnée. La clé primaire est composée de NumEtudiant, CodeMatiere, et DateEvaluation pour permettre à un étudiant d'avoir plusieurs notes dans la même matière à des dates différentes. NumEtudiant et CodeMatiere sont des clés étrangères qui réfèrent respectivement aux tables Etudiants et Matières.

Le Modèle Relationnel

Exemple: Système de Notation



Code + Diagram: <https://dbdiagram.io/d/SIBDE1-67ab27c8263d6cf9a0c04e6a>

Voici une solution pour le schéma relationnel de gestion des notes, avec explications, contraintes d'intégrité et alternatives :

Schéma relationnel:

- **Etudiants (NumEtudiant, Nom, Prénom, DateNaissance, Adresse)**
 - **NumEtudiant** : Entier (INT), Clé Primaire
 - **Nom** : Texte (VARCHAR(255)), Non Null
 - **Prénom** : Texte (VARCHAR(255)), Non Null
 - **DateNaissance** : Date (DATE)
 - **Adresse** : Texte (VARCHAR(255))
- **Matières (CodeMatiere, NomMatiere, Coefficient)**
 - **CodeMatiere** : Texte (VARCHAR(10)), Clé Primaire
 - **NomMatiere** : Texte (VARCHAR(255)), Non Null
 - **Coefficient** : Décimal (DECIMAL)
- **Notes (NumEtudiant, CodeMatiere, Note, DateEvaluation)**
 - **NumEtudiant** : Entier (INT), Clé Étrangère référençant Etudiants(NumEtudiant)
 - **CodeMatiere** : Texte (VARCHAR(10)), Clé Étrangère référençant Matières(CodeMatiere)
 - **Note** : Décimal (DECIMAL)
 - **DateEvaluation** : Date (DATE)
 - **Clé primaire** : (NumEtudiant, CodeMatiere, DateEvaluation)

Domaines:

- NumEtudiant : Entiers positifs.
- CodeMatiere : Chaînes de caractères alphanumériques.
- Nom, Prénom, NomMatiere, Adresse : Chaînes de caractères.
- DateNaissance, DateEvaluation: Dates.
- Coefficient, Note : Nombres décimaux. Le domaine de Note pourrait être contraint ; par exemple, entre 0 et 20, ou 0 et 100.

⚠️ Contraintes d'intégrité:

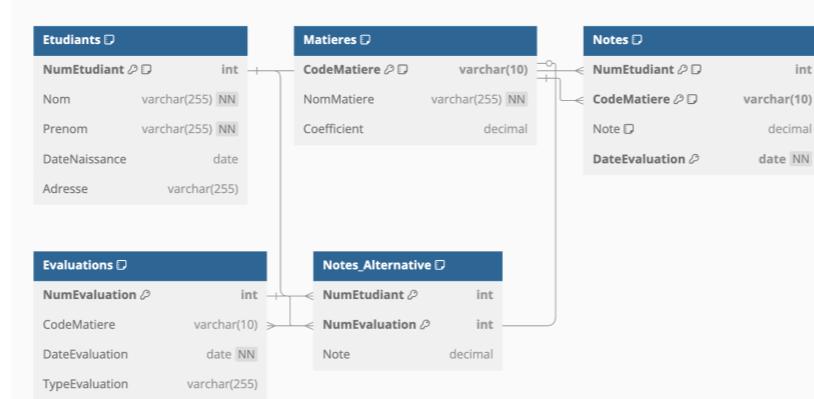
- **Clé primaire unique:** Chaque NumEtudiant dans la table Etudiants, chaque CodeMatiere dans la table Matières, et chaque combinaison (NumEtudiant, CodeMatiere, DateEvaluation) dans la table Notes, doivent être uniques.
- **Clés étrangères valides:** Chaque NumEtudiant dans la table Notes doit exister dans la table Etudiants. Chaque CodeMatiere dans la table Notes doit exister dans la table Matières.
- **Valeurs non nulles:** Nom et Prénom dans la table Etudiants, et NomMatiere dans la table Matières ne peuvent pas être nuls. On peut aussi ajouter Note et DateEvaluation dans Notes comme Non Null, selon les besoins.
- **Contraintes de domaine:** S'assurer que les valeurs des attributs respectent les domaines définis (ex: Note entre 0 et 20).

💡 Explications:

- Table Etudiants: Contient les informations sur chaque étudiant. NumEtudiant est la clé primaire pour identifier de manière unique chaque étudiant.
- Table Matières: Contient les informations sur chaque matière. CodeMatiere est la clé primaire.
- Table Notes: Associe une note à un étudiant pour une matière donnée. La clé primaire est composée de NumEtudiant, CodeMatiere, et DateEvaluation pour permettre à un étudiant d'avoir plusieurs notes dans la même matière à des dates différentes. NumEtudiant et CodeMatiere sont des clés étrangères qui réfèrent respectivement aux tables Etudiants et Matières.

Le Modèle Relationnel

Exemple: Système de Notation



Code + Diagram: <https://dbdiagram.io/d/SIBDE1-67ab27c8263d6cf9a0c04e6a>

💡 Autres solutions possibles :

- Table Evaluations:** Au lieu d'avoir la DateEvaluation dans la table Notes, on pourrait créer une table Evaluations(NumEvaluation, CodeMatiere, DateEvaluation, TypeEvaluation) pour gérer les différents types d'évaluations (examen, devoir, etc.). La table Notes contiendrait alors NumEtudiant, NumEvaluation, et Note. Ceci est utile pour gérer des informations plus détaillées sur chaque évaluation.
- Table Types d'évaluations:** On pourrait raffiner en ajoutant une table TypeEvaluation(IdType, Libelle) si on souhaite gérer une nomenclature des types d'évaluations (ex: Examen, Devoir Surveillé, etc.). La table Evaluations aurait alors un champ IdType (clé étrangère).
- Décomposition de l'adresse:** L'attribut Adresse peut être décomposé en plusieurs attributs (Numéro, Rue, Ville, CodePostal, Pays) pour des requêtes plus précises.
- Spécialisations d'étudiants :** Si nécessaire, on peut créer des sous-types d'étudiants (ex: Etudiant_Local, Etudiant_Erasmus) avec des attributs spécifiques à chaque type. Ceci nécessite l'utilisation de l'héritage, souvent implémenté par une relation 1:1 avec la table Etudiants.

Ce schéma est flexible et peut être adapté en fonction des besoins spécifiques de votre application. L'important est de bien identifier les entités, leurs attributs, les relations entre elles, et surtout les contraintes d'intégrité pour garantir la cohérence et l'intégrité des données.

Le Modèle Relationnel

Exemple: Système de Notation



Code + Diagram:

<https://dbdiagram.io/d/SIBDE1-67ab27c8263d6cf9a0c04e6a>

💡 Autres solutions possibles :

- Table Evaluations:** Au lieu d'avoir la DateEvaluation dans la table Notes, on pourrait créer une table Evaluations(NumEvaluation, CodeMatiere, DateEvaluation, TypeEvaluation) pour gérer les différents types d'évaluations (examen, devoir, etc.). La table Notes contiendrait alors NumEtudiant, NumEvaluation, et Note. Ceci est utile pour gérer des informations plus détaillées sur chaque évaluation.
- Table Types d'évaluations:** On pourrait raffiner en ajoutant une table TypeEvaluation(IdType, Libelle) si on souhaite gérer une nomenclature des types d'évaluations (ex: Examen, Devoir Surveillé, etc.). La table Evaluations aurait alors un champ IdType (clé étrangère).
- Décomposition de l'adresse:** L'attribut Adresse peut être décomposé en plusieurs attributs (Numéro, Rue, Ville, CodePostal, Pays) pour des requêtes plus précises.
- Spécialisations d'étudiants :** Si nécessaire, on peut créer des sous-types d'étudiants (ex: Etudiant_Local, Etudiant_Erasmus) avec des attributs spécifiques à chaque type. Ceci nécessite l'utilisation de l'héritage, souvent implémenté par une relation 1:1 avec la table Etudiants.

Ce schéma est flexible et peut être adapté en fonction des besoins spécifiques de votre application. L'important est de bien identifier les entités, leurs attributs, les relations entre elles, et surtout les contraintes d'intégrité pour garantir la cohérence et l'intégrité des données.

Technologie et Sécurité

Systèmes de Gestion de Bases de Données (SGBD)

Les Gardiens des Données

- Un **SGBD** est un logiciel qui permet de créer, gérer et interroger une base de données. Il **agit comme une interface** entre les utilisateurs, les applications et les données elles-mêmes.
- **Analogie:** Le bibliothécaire qui gère le catalogue et l'accès aux livres.
- **Exemples:** MySQL, PostgreSQL, Oracle, SQL Server, MongoDB.



Systèmes de Gestion de Bases de Données (SGBD)

Fonctions Principales d'un SGBD

- Un SGBD offre trois fonctionnalités principales :

⌚ **Langage de Définition de Données (DDL)** : permet de créer et modifier la structure de la base de données (analogie : créer le plan de la bibliothèque).

○ **Exemple:** CREATE TABLE Etudiants...

⌚ **Langage de Manipulation de Données (DML)** : permet d'interroger et de modifier le contenu de la base de données (analogie : ajouter, déplacer, supprimer des livres).

○ **Exemple:** SELECT * FROM Etudiants, INSERT INTO Notes...

⌚ **Langage de Contrôle de Données (DCL)** : permet de gérer les droits d'accès et les permissions des utilisateurs (analogie : contrôler qui peut entrer dans la bibliothèque et emprunter des livres).

○ **Exemple:** GRANT SELECT ON Notes TO Etudiant1;

Langage SQL

Parler aux Données

SQL (Structured Query Language) est le langage standard pour interagir avec les SGBD relationnels. Il permet de réaliser les opérations DDL, DML et DCL.
Analogie : le langage utilisé pour communiquer avec le bibliothécaire.

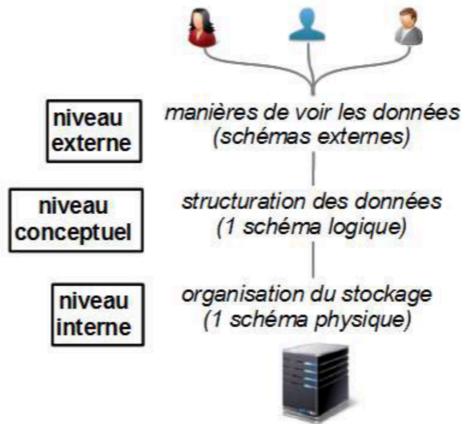
- Exemple simple de requête :

```
SELECT nom  
FROM Etudiants  
WHERE age > 18;
```

Architecture d'un SGBD (Modèle simplifié)

Architecture à trois niveaux (ANSI/SPARC)

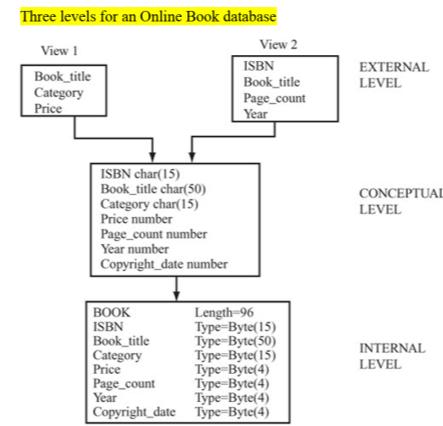
- **Niveau externe:** vue utilisateur des données (analogie : la section "Romans" de la bibliothèque).
- **Niveau conceptuel:** représentation globale des données (analogie : le catalogue complet de la bibliothèque).
- **Niveau interne:** organisation physique des données sur disque (analogie : l'emplacement physique des livres dans les étagères).



Source: cours-intro-base-donnees; PDF
(gilles-hunault.leria-info.univ-angers.fr)

Architecture d'un SGBD (Modèle simplifié)

Architecture à trois niveaux (ANSI/SPARC)



Source: Lesson10; PDF (courses.aiu.edu)

Pr. Sara El-Ateif

Systèmes d'Information et Bases de Données

2024-2025

Sécurité des SGBD

Pourquoi la sécurité des BD est-elle cruciale ?

- Les bases de données contiennent souvent des **informations sensibles**. Il est crucial de les protéger contre :
 - **Accès non autorisé** (exemple : un étudiant accédant aux notes d'un autre étudiant).
 - **Modification malveillante** (exemple : modification frauduleuse des salaires dans une BD d'entreprise).
 - **Fuite de données** (exemple : vol de données de cartes de crédit).
 - **Destruction accidentelle ou malveillante** (exemple : suppression de données critiques par erreur ou par un attaquant).

Sécurité des SGBD

Mécanismes de sécurité

- Les SGBD offrent divers mécanismes de sécurité, notamment :
 - **Contrôle d'accès** : restriction de l'accès aux données en fonction du rôle de l'utilisateur (analogie : badges d'accès, différents niveaux d'autorisation).
 - **Authentification** : vérification de l'identité de l'utilisateur avant de lui accorder l'accès (analogie : présenter sa carte d'identité). Exemple: login/mot de passe.
 - **Chiffrement** : protection des données en les rendant illisibles sans la clé de déchiffrement (analogie : code secret).
 - **Audit** : enregistrement des actions des utilisateurs sur la BD (analogie : caméras de surveillance).

Types de BD : NoSQL

- Les BD NoSQL (une alternative aux BD relationnelles) sont conçues pour gérer des volumes importants de **données non structurées ou semi-structurées**. Catégories de BD NoSQL :



- **Orientées documents** (Exemple : MongoDB) – pour stocker des données sous forme de documents JSON, utiles pour les applications web et mobiles.
- **Orientées clé-valeur** (Exemple : Redis) – pour des accès rapides aux données par clé, utile pour le caching. The Redis logo, showing a red 3D cube with a white keyhole icon.
- **Orientées graphes** (Exemple : Neo4j) – pour modéliser les relations entre les données, utile pour les réseaux sociaux. The Neo4j logo, featuring a blue eye icon with the word "neo4j" below it.
- **Orientées colonnes** (Exemple : Cassandra) – pour des requêtes analytiques sur de larges datasets.



Conclusion

- Les bases de données sont au cœur des SI modernes et sont essentielles pour gérer efficacement le déluge de données actuel.
- Le modèle relationnel et le langage SQL sont des concepts fondamentaux à maîtriser.
- La sécurité est un aspect crucial dans la conception et l'utilisation des BD.
- L'évolution des technologies NoSQL offre des alternatives pour les nouveaux types de données et d'applications.

28

Stockage des données

❑ Plusieurs applications nécessitent le stockage des données

pour de longues périodes

❑ L'information peut être stockée:

❑ dans des fichiers

- Ne nécessite aucun logiciel supplémentaire

- Facile à utiliser

- Ne nécessite pas beaucoup d'espace disque

❑ dans des bases de données

- Plus sécurisées

- Les données sont mieux organisées et donc plus facile à manipuler

❑ Possibilité de gérer les accès concurrents.

Inconvénients des fichiers

❑ Redondance des données et incohérence

- même information dupliquée dans plusieurs fichiers

- mise à jour dans un fichier et pas dans les autres

❑ Difficulté à accéder aux données

❑ Isolation des données: plusieurs fichiers, plusieurs formats

② Problèmes d'intégrité

- contraintes d'intégrité codées dans des programmes

- difficile d'en ajouter ou de les changer

③ Problèmes d'atomicité

- si crash du système durant l'exécution d'un programme, état incohérent

④ Problèmes de concurrence

- anomalies si accès concurrents

⑤ Problèmes de sécurité