	**Bases de données**	**NSI T^{ale}**
	Concepts	Cours/TD

*

1. Introduction

→ Définitions

Base de Données : Une Base de Données (BD) est un ensemble structuré d'informations agrégées ou élémentaires accessible par une communauté d'utilisateurs.
Système de Gestion de Base de Données : On désigne par Système de Gestion de Base de Données (**SGBD**) le logiciel qui permet d'interagir avec une base de données et implante les fonctions qui permettent d'**archiver**, **retrouver**, **traiter** et enfin **mettre à jour** les données, qui sont mémorisées au sein d'une Base de Données (BD).

2. Objectifs fondamentaux

→ Centraliser l'information

La centralisation de l'information a pour but de:

- supprimer la redondance,
- permettre l'unicité de la saisie,
- centraliser des contrôles.

→ Assurer l'indépendance données-traitements.

Il faut donc permettre:

- à différents programmes d'applications d'avoir différentes vues d'une même donnée,
- qu'une modification de l'image d'une donnée ou d'une stratégie d'accès soit sans impact sur les traitements existants qui y ont éventuellement recours.

→ Permettre des liaisons entre ensembles de données

La mise en œuvre des liaisons, qui aurait nécessité la mise en œuvre de structures en graphe sans base de données, doit pouvoir s'appréhender de façon simple et homogène, quel que soit le type de complexité structurelle.

→ Intégrité.

Il faut s'assurer de la fiabilité et de la cohérence du système d'information.
Pour cela il faut définir des contraintes d'intégrité et des contraintes sémantiques du type:

- • appartenance à une liste de valeurs ou à un intervalle,
- • format (nature, longueur),
- • règles de cohérence avec d'autres informations.

→ Sécurité

Il faut assurer un mécanisme de reprise après panne (ou recovery).
Pour cela, un certain nombre de dispositifs doit permettre:

- la création de points de reprise (un point de reprise est une copie de la BD dans un état cohérent),
- la gestion d'un journal des transactions, c'est-à-dire d'une liste des opérations réalisées sur la BD à partir du dernier point de reprise.

Si une anomalie se produit, une procédure de reprise permet, à partir du point de reprise et du journal des transactions, de restaurer la base dans un état cohérent et de détecter les transactions à l'origine de l'anomalie. .

→ Confidentialité

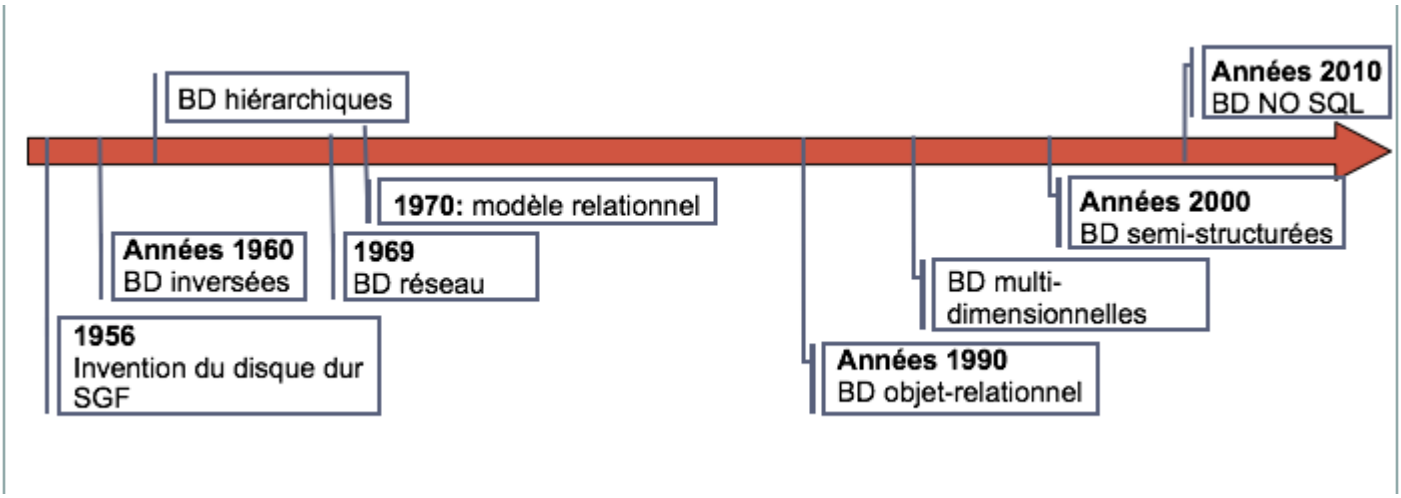
Un SGBD doit offrir des sources de discrétion évitant l'accès par n'importe qui à n'importe quelle information. Généralement, la confidentialité est assurée par des procédures:

- d'identification (par le nom, un numéro de code),
- d'authentification (mot de passe),
- d'autorisation d'accès selon l'utilisateur (droit de création, consultation, modification, suppression) de telle ou telle donnée.

→ Partage des données

Pour préserver l'intégrité d'une BD, le concept de transaction (traitement atomique) a été introduit. Cela permet d'enchaîner des mises à jour. Ainsi si deux mises à jour doivent être réalisées conjointement on les regroupe dans une transaction, pour éviter des phénomènes parasites comme la perte de mise à jour.

3. Un peu d'histoire



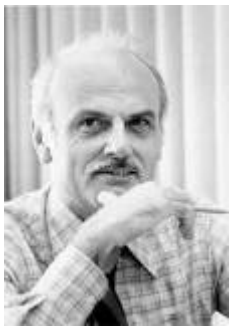
Evolution des SGBD dans le temps

Différentes générations de systèmes de gestion de la persistance des données ont vu le jour dès l'apparition du concept de fichier, dans les années 1960. Parmi eux on peut citer :

- **les bases de données inversées**, qui ont mis en œuvre des mécanismes de multi-indexation,c'est-à-dire la possibilité de gérer plusieurs index sur un même ensemble de données. Les données étaient stockées dans des fichiers sur des supports à accès direct, et le principe consistait à inverser les fichiers par rapport à des rubriques données, construisant ainsi des tables d'index qui associaient à chaque valeur des rubriques la liste des enregistrements des fichiers contenant cette valeur. Ces

tables d'index permettaient de répondre rapidement à une requête. Parmi les représentants de ces bases de données inversées, on peut citer ADABAS et MIISFIIT(systèmes à indexation totale).

- **les bases de données hiérarchiques**, qui ont introduit la possibilité de manipuler des structures arborescentes (nombre quelconque d'éléments en profondeur et en largeur). Les représentants de ces systèmes s'appelaient IMS (Information Management System proposé par IBM) ou System 2000.
- **les bases de données réseau**, qui ont étendu l'utilisation des structures arborescentes et des référencements en permettant de manipuler des structures en graphe ou en réseau. Apparues dans les années 1970, elles ont été définies par le DBTG (Data Base Task Group) du CODASYL (Conference On Data System Language). Les représentants de ces systèmes s'appelaient IDS (Integrated Data Store de CCI-HB), IDMS, TOTAL ou SOCRATE.



Edgar Frank "Ted" Codd (1923-2003)

Le **modèle relationnel** est basé sur la théorie mathématique des relations.
Alors que de nombreux efforts portaient sur la possibilité de manipuler des structures de plus en plus générales et complexes, **E.F. Codd** a publié en juin 1970 son article intitulé *"A relational model for large shared data banks"* dans lequel il propose une remise à plat des structures avec le concept de forme normale.
Le modèle relationnel propose une représentation tabulaire des relations et permet de formaliser des opérateurs de manipulations qui se regroupent dans une algèbre, l'algèbre relationnelle.
On y distingue actuellement neuf opérateurs: **union, intersection, différence, produit cartésien, projection, sélection, jointure, division, agrégation**.
Ce sont ces opérateurs qui constituent le cœur des moteurs relationnels gérant les accès aux données stockées dans des tables. Les structures complexes se mettent œuvre via les **jointures**.

C'est sur ces propositions que se sont progressivement élaborées les premières spécifications du **langage SQL** (***Structured Query Language***), langage beaucoup plus accessible que ceux proposés par les systèmes des générations précédentes.
Les premiers prototypes implantant les concepts relationnels ont été System-R (IBM) et Ingres (université de Berkeley). La généralisation des systèmes relationnels ne deviendra effective que vers la fin des années 1980 après avoir bénéficié du succès des générateurs d'applications sur micro-ordinateurs utilisant une base de données de type relationnel (DBase, Foxbase, 4ème Dimension, etc). Les systèmes représentant cette génération sont Oracle, Ingres, Sybase, DB2, INFORMIX, MS SQL Server, etc.

Depuis d'autres SGBD sont apparus tels que

- les **bases de données orientées-objet** (**SGBDOO**) telles que Gesmstone, Ontos, O2, Object Store, Objectivity, etc.
- les **bases de données multidimensionnelles** adaptées aux applications décisionnelles tellse que ESSBASE, MS Analysis Services, Oracle OLAP, etc.
- les **bases de données semi-structurées** basées pour la plupart sur le format XML
- les **bases de données NoSQL** pour gérer d'énormes quantités de données telles que Amazon Dynamo, CouchDB (gestion de documents), BigTable (Google), Neo4J (graphes), etc.

4. Concepts fondamentaux du modèle relationnel

→ Définitions et notations

Base de Données : Dans le modèle relationnel, une **base de données** est vue comme un ensemble de ***relations.***

De manière formelle on peut donc écrire : *BD = {Relations}*

Relations : Une **relation** est définie par son **schéma** *SR* et par son **extension** *ER*.

On peut écrire *Relation = (SR, ER)*

Schéma relationnel : Le **schéma** définit l'ensemble des **attributs** d'une relation.

Le schéma *SR* de la relation *R* se note $R[A_1;A_2; ...;A_n]$ où *R* est le nom de la relation et A_i représente un **attribut**. Le nombre d'attributs n est le **degré** de la relation.

A chaque **attribut** est associé l'**ensemble de ses valeurs plausibles**, appelé **domaine** de valeurs et noté $DOM(A_i)$. $DOM(A_i)$ peut être spécifié sous forme d'une liste de valeurs, d'un intervalle, d'un format. Plus les domaines sont définis de façon précise, moins il y aura d'erreurs dans les valeurs d'attributs et donc moins d'incohérence dans la BD.

Extension : L'**extension** *ER* de la relation est un ensemble de ***tuples*** où chaque tuple contient la valeur associée à chaque attribut.

$ER = \{tuples\ ou\ n\text{-}uplets\} = \{t_1; t_2; ...; t_i; ...; t_n\}$

avec $t_i = (v_{i1},v_{i2}, ...,v_{in})$ où $t_i[A_j] = v_{ij}$ et v_{ij} est ***atomique***, c'est à dire qu'il correspond à une donnée élémentaire. $t_i[A_j]$ représente la valeur de l'attribut A_j dans le tuple t_i. Chaque v_{ij} est un élément de $dom(A_i)$ ou une valeur spéciale "nulle". La valeur nulle a deux interprétations:

- il s'agit d'une valeur non-applicable à l'attribut de ce tuple (comme par exemple la date de mariage pour une personne célibataire),
- c'est une valeur possible mais inconnue (comme par exemple l'âge d'un étudiant).

Cardinalité : Le nombre de tuples d'une relation est sa **cardinalité**.

→ Application à la représentation tabulaire des relations

En pratique on a la correspondance :

- **Relation ↔ Table**
- **Attribut ↔ Colonne**
- **Tuple ↔ Ligne**

On verra par la suite que le modèle tabulaire est plus « naturel » pour représenter les rel ations.

5. Représentation graphique

Soient:

BD={RELATION1, RELATION2}
RELATION1=(SR1, ER1)
SR1=R1[X,Y,Z]
ER1={(x1,y1,z1), (x1,y2,z), (x2,y1,z1), (x3,y2,z), (x3,y3,z1)}
RELATION2=(SR2,ER2)
SR2=R2[A,B]
ER2={(a1,x1),(a1,x2),(a2,x2),(a3,x3)}

ER1 comprend 5 tuples *t1; t2; t3; t4; t5* avec *t1 =(x1,y1,z1), t1[X]=x1, t1[Y]=y1, t1[Z]=z1*.
La représentation graphique d'un tel modèle est la suivante:

R1	X	Y	Z
t1	x1	y1	z1
t2	x1	y2	z
t3	x2	y1	z1
t4	x3	y2	z
t5	x3	y3	z1

R2	A	B
	a1	x1
	a1	x2
	a2	x2
	a3	x3

6. Clés de la relation

Clé - Clé primaire :
Intuitivement, le concept de **clé** correspond à la façon d'**identifier de manière unique un tuple** dans une table. Une relation peut posséder plusieurs clés possibles pour identifier les tuples de manière uniques. On les appelle des **clés candidates**.

Ces clés candidates peuvent être composée d'un attribut ou d'un groupe d'attributs.

On garde généralement **le plus court** comme identifiant, c'est ce que l'on appelle **la clé primaire** de la relation.

Lors de la définition d'une relation, **la clé primaire est soulignée**.

Clé étrangère

Il existe une contrainte importante au sein des bases de données relationnelles, celle de l'intégrité référentielle (ou contrainte d'inclusion):

- si un tuple t_1 se réfère à un autre tuple t_2, t_2 doit toujours exister,
- la référence se fait par le biais de clés étrangères.

Formellement, une **clé étrangère** FK est définie ainsi:

- FK a le même domaine de valeurs qu'une **clé primaire K**,
- toute valeur de FK dans un tuple t_1 doit correspondre à une valeur de K dans un tuple t_2 ou doit être nulle :

$$t_1[FK]=t_2[K] \text{ ou } t_1[FK]=null.$$

Les clés étrangères sont les seules redondances autorisées dans le schéma relationnel. Une clé étrangère doit être mise en évidence dans la liste des attributs qui caractérisent la relation. On fera souvent précéder les clés étrangères du signe #.

→ Exemple

Considérons les deux relations ELEVE[CODE, NOM, CODEI] et INSTITUTEUR [CODEI, NOMI, NIVEAU].

ELEVE

CODE	NOM	CODEI
e1	Benoit	I1
e2	Dupont	I1
e3	Martin	I2
e4	Pinet	I3
e5	Murol	I2

INSTITUTEUR

CODEI	NOMI	NIVEAU
I1	Bazin	CP
I2	Maubert	CE1
I3	Dervieux	CE2
I4	Bernard	CM1

L'attribut CODEI de la relation ELEVE est une clé étrangère:

- à toute valeur de CODEI dans la table ELEVE doit correspondre une valeur de CODEI dans la table INSTITUTEUR (c'est-à-dire doit correspondre un instituteur): il est par exemple impossible d'insérer le tuple (e6, Bardeau, I5) dans la table ELEVE.
  L'inverse n'est pas forcément vrai, c'est-à-dire que pour une valeur de CODEI dans la table INSTITUTEUR ne correspond pas forcément de valeur de CODEI dans la table ELEVE (comme I4 dans notre exemple).
  Par contre, à une valeur de CODEI dans la table INSTITUTEUR peut correspondre plusieurs valeurs de CODEI dans la table ELEVE (I1 dans notre exemple);

- le domaine de valeur de CODEI dans la table ELEVE est le même que celui de CODEI dans la table INSTITUTEUR.

On notera de la façon suivante les deux relations (la clé étrangère est précédée du signe #):