**Les bases de données**

Table des matières

[I. Introduction 2](#_Toc92985241)

[II. Historique 2](#_Toc92985242)

[III. Définition 2](#_Toc92985243)

[IV. Les différentes formes de SGBD 3](#_Toc92985244)

[V. Conception des bases de données. 3](#_Toc92985245)

[Exemple 1 : 5](#_Toc92985246)

[Exemple 2 : 7](#_Toc92985247)

[Exemple 3 : 8](#_Toc92985248)

## I. Introduction

* Une base de données c’est un point de rassemblement centralisé de données qui sert à enregistrer pour les utiliser plus tard à un endroit donné.
* L’utilité d’une base de données est l’enregistrement et le suivi (ou recherche) des données.
* L’avantage d’une base de données est la persistance des données (enregistrement permanant et les données sont accessibles à tout moment)

## II. Historique

* Optimisation de la persistance de données : gravure sur solide < feuilles manuscrites < feuilles dactylographiées < fiches perforées < bandes magnétiques < CD-ROM, clé USB, base de données …

**Définition SI**

* Le système d’information (SI) d’une société est l’ensemble des moyens humains et matériels permettant de véhiculer de l’information dans l’entreprise.

## III. Définition

#### 1. Base de Données

* Espace mémoire (zone nécessaire pour écrire une info)
* Support persistant
* Accessible et sécurisé (soucis de confidentialité et autres…)
* Centralisation (pour garder l’intégrité des données ou éviter deux versions d’une même donnée à la suite d’une erreur humaine)

#### 2. Système de Gestion de Base de Données (SGBD)

* Base de données : terme générique et SGBD : **application**.
* Un système de gestion, de partage, de protection, qui garde l’intégrité, la cohérence et l’indépendance des données.
* Parmi les systèmes de gestion de bases de données relationnelles largement utilisés par les entreprises, citons la base de données Oracle, MySQL, PostgreSQL (une base de données relationnelle open source) et Microsoft SQL Server.
* Les structures RDBMS sont couramment utilisées pour effectuer quatre opérations de base : CRUD (créer, lire, mettre à jour et supprimer), qui sont essentielles pour prendre en charge une gestion cohérente des données.

**Quelques principes de base.**

* Indexation (pour retrouver l'info)
* Regroupement (c’est l’organisationnel et pour maitriser la volumétrie. Plus les données sont de gros volume, plus le temps pour les chercher est grand. Donc c’est pour optimiser les recherches)
* Structure de données et model (c’est l’ossature des informations : l'adresse, la taille et le type de ce qu’on veut enregistrer)

#### 3. Model hiérarchique

* Il y a un cheminement hiérarchisé pour stocker et retrouver une donnée.

#### 4. Model réseau

* Un modèle qui n’est plus utilisé (culture générale)

#### 5. Model relationnel

* Un modèle qui met en évidence des entités (ou structures) et on cherche la relation ou interactions entre les différentes entités. Cette relation est définie par la réalité de l’activité du client.
* On dit que c’est un mode normalisé (optimisation de l’espace de stockage puisqu’avant c’était cher).

## IV. Les différentes formes de SGBD

#### 1. Les infocentres

* Utilisés dans le secteur des banques et assurances et le langage de programmation utilisée est COBOL (COmmon Business Oriented Language).
* Constitué de : base de données – mainframe (SGBD et autres appli) – le réseau et les terminaux.
* Un terminal est un moyen pour se connecter à l'infocentre qui va transmettre les données stockées dans la base de données.

#### 2. BD réparties

Nœud A

* Répartition des données en sous-groupes pour que tous les utilisateurs ne se connectent pas au même endroit à la fois.

BD entière

* Pour éviter la diminution de la vitesse de traitement des requêtes ainsi pour optimiser l’expérience utilisateur.

Nœud B

Nœud C

## V. Conception des bases de données relationnelle.

1. MERISE

Méthode d’études de réalisation des informations dans des systèmes informatiques. Cela se fait à plusieurs niveaux :

* Niveau conceptuel (MCD) : basé sur le besoin fonctionnel
* Niveau logique (MLD) : comment on va implémenter la BD.
* Niveau physique (MPD) : quand la BD est créée en réel

1. MCD

Le modèle conceptuel des données (**MCD**) a pour but d'écrire de façon formelle les données qui seront utilisées par le système d'information. Il se fait en deux temps :

* Analyse des règles de gestion (SFG)
* Elaboration des dictionnaires des données par rapport au besoin fonctionnel du client pour épurer (enlever les éléments qui ne servent pas à la création de la BD)

1. Recherche de dépendance fonctionnelles entre les éléments constitutifs

Les **dépendances fonctionnelles** sont les liens existants entre deux propriétés présentes dans la base de données.

Les éléments constitutifs du MCD :

* + Entités (élément individuel représentable ayant une représentation réelle)
  + Attributs (valeur distinctive qui caractérise l’entité)
  + Clé ou identifiant (valeur unique identifiant une entité)
  + Associations (interaction entre 2 entités)
  + Cardinalité (nombre de fois qu’une relation entre 2 entités peut être réalisé)

1. Règle de passage du MCD au MLD

**Règle numéro 1 :**

* Une entité du MCD devient une table.
* Son identifiant devient la clé primaire de la table.
* Ses autres propriétés deviennent les attributs de la table.

**Règle numéro 2 :**

Une **association de type 1:N** (c’est à dire qui a les cardinalités maximales positionnées à « 1 » d’un côté de l’association et à « n » de l’autre côté) se traduit par la création d’une clé étrangère dans la relation correspondante à l’entité côté « 1 ». Cette clé étrangère référence la clé primaire de la relation correspondant à l’autre entité.

**Règle numéro 3 :**

Toute **association de type N:N** devient une table avec comme clé la concaténation des identifiants des entités qu’elle relie et comme attributs les propriétés de l’association.

1. Les formes normales (FN)

**1FN**

Une relation est en première forme normale **1FN** si tous ses attributs sont atomiques (inhérent au modèle relationnel)

* Un attribut atomique n’est pas : multivalué (liste de valeurs) composé (structuré en sous-attributs)

**2FN**

Une relation est en deuxième forme normale **2FN** si et seulement si :

* + Elle est en première forme normale
  + Tout attribut non-clé dépend de la totalité de toutes les clés

**3FN**

Objectif **3FN** : élimination des redondances dues aux dépendances fonctionnelles déduites par transitivité

Définition : Une relation est en troisième forme normale 3FN si et seulement si :

* + Elle est en deuxième forme normale
  + Il n’existe aucune dépendance fonctionnelle entre attributs non-clé

Exemple 1 :

Une image contenant texte, intérieur, capture d’écran

Description générée automatiquement

|  |  |
| --- | --- |
| Les possibilités | |
| n1-n2 | n3-n4 |
| 0-1 | 0-1 |
| 0-n | 0-1 |
| 1-1 | 0-n |
| 1-n | 1-n |

n1 (nombre minimum de produit achaté pour enregistrer le produit)

n2 (nombre maximum de produit pouvant être acheté par un client)

n1 (nombre minimum d’achat d’un client pour qu’il puisse être enregistré)

n1 (nombre maxmum d’achat q’un client puisse faire sur le produit)

Une image contenant texte, capture d’écran, intérieur, ordinateur

Description générée automatiquement

Voici ce que cela donne en table (**primary key** …) :

|  |  |  |  |
| --- | --- | --- | --- |
| **produit** | Pid | libelle | Prix conseillé |
|  | P1  P2  P3  … | B  A  F | 300  10  37 |

|  |  |  |  |
| --- | --- | --- | --- |
| **client** | Cid | nom | … |
|  | C1  C2  C3  … | pierre  paul  jacques  … | … |

|  |  |  |  |
| --- | --- | --- | --- |
| Achat | Pid | Cid | Qte |
|  | P1  P1  … | C1  C1 | 3  2 |

* Si le même client achète le même produit à deux dates différentes on obtient deux même clé (P1.C1) ce qui ne peut pas exister.

Il faut créer une nouvelle entité date contenant une clé unique par achat pour distinguer ces deux relations achats.

Une image contenant texte, intérieur, capture d’écran

Description générée automatiquement

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Achat** | Pid | Cid | Date | Qte |
|  | P1  P1  … | C1  C1 | D1  D2 | 3  2 |

Si le prix des produits évolue au cours du temps, on crée une entité « price » :

|  |  |  |
| --- | --- | --- |
| **price** | Pr id | value |
|  | Pr1  Pr2  Pr3  … | 300  350  400 |

Si on veut suivre le prix à une date ou pendant une période donnée on ajoute les attributs tels que les dates de validité :

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **price** | Pid | Pr id | DDV | DFV | value |
|  | P1  P1  P3  … | Pr1  Pr2  Pr3  … | 15/08/2021  3/01/2022  15/02/2022  … | 2/01/2022  14/02/2022  Null  … | 300  350  400 |

**Type de fonction (schematique) pour entrer le prix dans le tableau :**

Fonction updatePrice (ihm : IHM) : vide ou booléen

Var idPrice

Début

Ihm.price

idPrice = InsertPrice (ihm.price)

insertAvancedPrice (idPrice, ihm.Pid, currentDate)

updatelastAvancedPrice (currentDate) ;

Fin

**Type de requête à faire pour trouver le prix :**

*Select value*

***Where (DDV>= ‘’jj/mm/aaaa ‘’) and (DFV<=’’jj/mm/aaaa’’)*** *// recherche avec date précise*

***Where (DDV <= D1) and (DFV>=D2)*** *// recherche avec un intervalle de date. Et on obtient 2 lignes comme résultat*

## Exemple 2 :

Le cas des pièces automobile : certaines pièces sont constituées d’autres pièces.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | composant | | | | |
|  |  | P1 | P2 | P3 | P4 | P5 |
| composé | P1 | x | v | v | v | v |
| P2 |  | x | v | v |  |
| P3 |  |  | x | v | v |
| P4 |  |  |  | x |  |
| P5 |  |  |  |  | x |

1:n

Est composé de

composé

1:n

Moteur est composé

|  |  |  |
| --- | --- | --- |
| Pièce | ref | label |
|  | P1  P2  P3  P4  P5  … | A  B  C  D  E  … |

|  |  |  |  |
| --- | --- | --- | --- |
| **composition** | composé | composant | Qte |
|  | P1  P1  P1  P1  P2  … | P2  P3  P4  P5  P3 | 2  10  1  2  2  … |

* Pour trouver les pièces composées :

***Select Distinct Composé from composant***

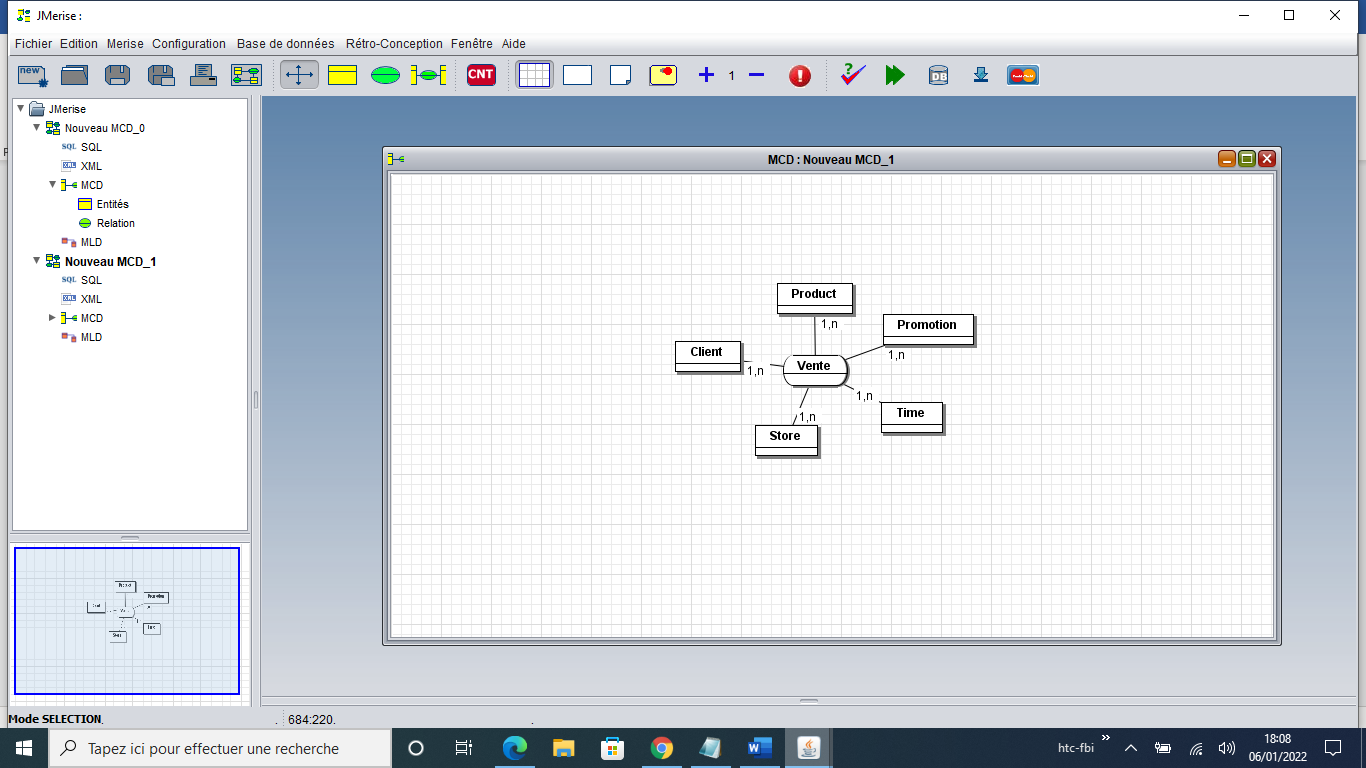
* Pour trouver les composants d’une pièce (P1 par exemple) :

***Select composant where composé = ‘’P1’’***

* Pour trouver toutes les pièces composées par une pièce défaillante (P4 par exemple) :

***Select \* from composition where composant = ‘’P4’’***

## Exemple 3 :

V1 (c1,p1,s1,P1,t1)

T1= 05/01/2022

Par vente, on obtient la table ci-dessous :

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Sale** | Cid | pid | Pid | sid | tid | storePrice | Qte |
|  | 100  … | 130 | 1345 | 12 | 730 | 15 | 2 |

A partir de ce model si on nous demande la liste des ventes d’un produit au jour t1 ou pour un mois ou un an donné, il est nécessaire de décomposer la donnée avec la structure **Time** comme suit.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Time** | tid | date | numDay | Num month | year | quart | … |
|  | 730  … | 06/01/2022 | 06 | 01 | 2022 | 1 | … |

Cela a pour but de faciliter le traitement lorsqu’il faudra regrouper, en fonction du temps, une grande quantité de données.