

# Introduction à la SGBD

Pape Abdoulaye BARRO

*Enseignant – chercheur*

*UIDT – EPT – Réseaux des E-LAB*

# PLAN

---

- Généralités
- Le modèle conceptuel
- Le modèle relationnel
- Le langage SQL
- Etude et réalisation d'une base de données
- La sécurité des données

# PLAN

---

- Généralités
- Le modèle conceptuel
- Le modèle relationnel
- Le langage SQL
- Etude et réalisation d'une base de données
- La sécurité des données

# Généralités

objectif

---

L'objectif de ce chapitre est :

- de définir la notion de base de données ainsi que les principaux concepts qui s'y rattachent.
- de présenter:
  - la méthodologie qui permet de les concevoir,
  - les applications informatiques associées à leur mise en œuvre (SGBD)
  - ainsi que les différents métiers possibles.

# Généralités

## définitions – base de données

### ● C'est quoi une base de données?

- Il est très difficile de donner une définition exacte d'une base de données. Mais dès que des données sont stockées de manière organisée dans un ou plusieurs fichiers, on peut donc parler de base de données. Dès que l'on dispose d'un ensemble structuré, son exploitation (**ajout, suppression, mise à jour, etc.**) devient alors possible.

**Une base de données est un ensemble de fichiers – informatiques ou non – structurés et organisés afin de stocker et de gérer de l'information.**

# Généralités

définitions – base de données

---

- Une **information** est un critère de choix parmi les éléments d'un ensemble de données qui permet de restreindre la taille de l'ensemble pour donner une réponse à une question.
  - Un *exemple d'ensemble de données* pourrait être la liste des élèves d'une classe. On pouvait se poser la question suivante: *qui est le meilleur élève de la classe ?*
  - On doit recherche parmi les meilleurs élèves (ceux qui ont obtenu une moyenne  $\geq 12$ ), Recherche : le meilleur d'entre eux (celui qui a la plus grande moyenne)
- L'**informatique** est le traitement automatique de l'information.

# Généralités

définitions – base de données

---

## ● Quelques remarques:

- Vous avez peut-être une idée intuitive des bases de données. Attention toutefois, le mot est souvent utilisé pour désigner n'importe quel ensemble de données. Il s'agit là d'un abus de langage qu'il convient d'éviter.
- Pour mériter le terme de base de données, un ensemble de données non indépendantes doit être interrogable par le contenu.
  - Les données doivent pouvoir être retrouvées selon n'importe quel critère :
    - Il devrait être possible de trouver une liste d'articles qui coûtent moins de 500 francs.
  - Il doit également être possible de retrouver leur structure :
    - un produit a un nom, un prix, une quantité, ...

# Généralités

## définitions – base de données

- Les bases de données se situent au cœur de l'activité des entreprises, de l'administrations, de la recherche et de bon nombre d'activités humaines désormais liées à l'informatique.
- Une base de données est une représentation partielle et très simplifiée du monde réel, que l'on a obtenu par un processus de **modélisation**.
- Pour les manipuler, on utilise généralement un logiciel spécialisé appelé **SGBD** (**S**ystème de **G**estion de **B**ases de **D**onnées).
- Dans le domaine purement informatique, elles interviennent dorénavant à tous les niveaux.
  - Les développeurs d'applications s'appuient sur des bases de données externes pour gérer leurs données alors qu'auparavant elles étaient intégrées dans le programme.
- Les bases de données reposent sur des théories solides et sont à l'origine d'une des plus importantes disciplines de l'informatique : **l'ingénierie des systèmes d'information**.

# Généralités

## définitions - SGBD

---

- Un SGBD peut être perçu comme un ensemble de *logiciels systèmes* permettant aux utilisateurs d'insérer, de modifier et de rechercher efficacement des *données spécifiques* dans une grande masse d'informations (pouvant atteindre quelques milliards d'octets) partagée par de multiples utilisateurs.
  - Le SGBD rend transparent le partage, à savoir donne l'illusion à chaque utilisateur qu'il est seul à travailler avec les données.

# Généralités

## définitions - SGBD

- Les **SGBD** se distinguent clairement des **systèmes de fichiers**.
  - Les **SGBD** permettent la description des données (*définition des types par des noms, des formats, des caractéristiques et parfois des opérations*) de manière séparée de leur utilisation (*mise à jour et recherche*).
  - Ils permettent aussi de retrouver les caractéristiques d'un type de données à partir de son nom (*par exemple, comment est décrit un produit*).
  - Le **système de fichiers** est un composant de plus bas niveau ne prenant pas en compte la structure des données.
- La tendance aujourd'hui est d'intégrer le système de fichiers dans le SGBD.

# Généralités

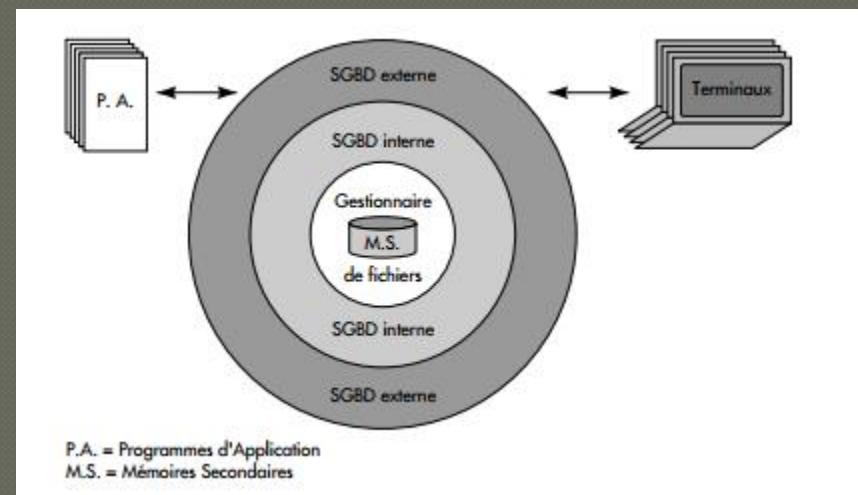
## définitions - SGBD

- Un *SGBD* se compose en première approximation de trois couches emboîtées de fonctions (de la mémoire à utilisateur).
  - Le système de gestion de fichiers (ou gestionnaire de fichiers) fournit aux *couches supérieures* des mémoires secondaires idéales adressables par objets et capables de recherches par le contenu des objets (mécanismes d'indexation notamment).
  - le système d'accès aux données (ou SGBD interne) (*deuxième couche*) qui assure la gestion des données stockées dans les fichiers, l'assemblage de ces données en objets, le placement de ces objets dans les fichiers, la gestion des liens entre objets et des structures permettant d'accélérer les accès aux objets.
  - SGBD externe (*la troisième couche*) dont le rôle consiste à mettre en forme et à présenter des données aux programmes d'applications et aux utilisateurs interactifs. Il a pour rôle, d'assurer:
    - d'une part, l'analyse et l'interprétation des requêtes utilisateurs en primitives internes,
    - d'autre part, la transformation des *données extraites de la base* en *données échangeables avec le monde extérieur*.

# Généralités

## définitions - SGBD

- Au delà des SGBD, les systèmes d'informations intègrent plus souvent **des ateliers de génie logiciel** permettant de modéliser les données d'une base de données et de représenter les traitements associés à l'aide de graphiques et de langages de spécifications.
- **Ces outils d'aide à la conception**, bien que non intégrés dans le SGBD, permettent de spécifier les descriptions des données (supportés par les SGBD)



**composition d'un SGBD**

# Généralités

## Historique des SGBD

Les toutes premières bases de données ont vues le jour vers les années 60 lorsqu'on utilisait les fichiers reliés par des pointeurs.

- C'est vers 1965, avec l'apparition des systèmes IDS.I de Honeywell et IMS.I de IBM (pour la mission APOLLO), que les SGBD modernes ont commencé à voir le jour. Cela a permis de constituer des chaînes d'articles entre fichiers et de parcourir ces chaînes.
  - La première génération de SGBD est marquée par la séparation de la description des données et de la manipulation par les programmes d'application. Elle coïncide avec l'avènement des langages d'accès navigationnels.
  - Elle est basée sur les modèles réseau ou hiérarchique et a été dominée par les SGBD TOTAL, IDMS, IDS 2 et IMS 2. Elle traite encore aujourd'hui une partie importante du volume de données gérées par des SGBD.

# Généralités

## Historique des SGBD

- La **deuxième génération de SGBD** a fait son apparition vers les années 1970 dans les laboratoires (début du modèle relationnel) et vise à enrichir et à simplifier le SGBD externe afin de faciliter l'accès aux données pour les utilisateurs. Les données sont représentées sous forme de tables et les recherches et mises à jour sont effectuées à l'aide d'un langage non procédural standardisé appelé *SQL(Structured Query language)*.
  - Les systèmes de la deuxième génération sont commercialisés depuis 1980 et aujourd'hui, ils représentent l'essentiel du marché des bases de données.
  - Les principaux systèmes sont ORACLE, INGRES, SYBASE, INFORMIX, DB2 et SQL SERVER.

# Généralités

## Historique des SGBD

- La troisième génération de SGBD est apparue vers les années 1980 dans les laboratoires et est fortement adoptés dans le monde industriel. Elle supporte des modèles de données extensibles associant relationnel et objet, ainsi que des architectures mieux réparties, permettant une meilleure collaboration entre des utilisateurs concurrents.
- Parmi les systèmes objet-relationnels nous pouvons citer Oracle 8, IBM DB2 Universal Database, Informix Universal Server et ObjectStore (plus novatrice).
- Tous ces systèmes tentent de répondre aux besoins des nouvelles applications (multimédia, Web, CAO, bureautique, environnement, télécommunications, etc.).

# Généralités

## Historique des SGBD

- La **quatrième génération** quant à elle, est mise en place pour mieux supporter Internet(le Web), **les informations non structurées**, les objets multimédias, mais aussi faciliter l'extraction de connaissances(data mining) ainsi que l'aide à la prise de décisions. Il devient alors de plus en plus difficile de développer un nouvel SGBD. Cette nouvelle génération est juste une évolution des SGBD de 3e génération plutôt qu'à une nouvelle révolution.

**Finalement, l'évolution des SGBD peut être perçue comme celle d'un arbre, des branches nouvelles naissant mais se faisant généralement absorber par le tronc, qui grossit toujours d'avantage.**

# Généralités

## les étapes de la conception

Le processus de conception d'une base de données peut être décomposé en plusieurs étapes allant de l'analyse du monde réel jusqu'à la création et l'utilisation.

- l'**analyse du monde réel** est l'étape de collecte des informations et des besoins des futurs utilisateurs. C'est dans cette étape que l'on détermine les objectifs du système d'information à mettre en place et identifie tous les éléments à prendre en compte.
  - Il permet d'identifier les objets et de modéliser les liens existant entre eux . Par exemple:
    - une personne achète un terrain à 5 000 000 FCFA.
      - Dans cette phrase, nous avons deux objets liés à savoir *personne* et *terrain* et le *prix* est un composant du lien.
  - Il permet également d'exprimer des règles de validité de données. Par exemple:
    - Les terrains dont nous disposons coutent au *minimum* 1 000 000 FCFA et au *maximum* 10 000 000 FCFA.
  - L'exploitation des données ou des liens existant entre les entités est réalisée à l'aide du modèle conceptuel «*entité-association*» ou, plus couramment aujourd'hui, exprimé avec le langage **UML** (Unified Modeling Language).

# Généralités

## les étapes de la conception

- Le **passage au SGBD** est l'étape de transformation de la représentation précédente afin de la rendre acceptable par le SGBD choisi. Lors de cette étape, il est question de vérifier la qualité de la base de données en éliminant les redondances et autres. Le modèle relationnel, avec ses outils, permet d'arriver à cette fin.
  - On obtient alors un schéma des données qui fournira aux utilisateurs les informations nécessaires pour effectuer leurs **requêtes**, par exemple, la description des *noms de tables*, de *champs* et leurs *types*.
  - C'est dans cette phase que l'on définit les « *vues* » du système d'information qui sont adaptées à chaque *catégorie d'utilisateurs*.

# Généralités

## les étapes de la conception

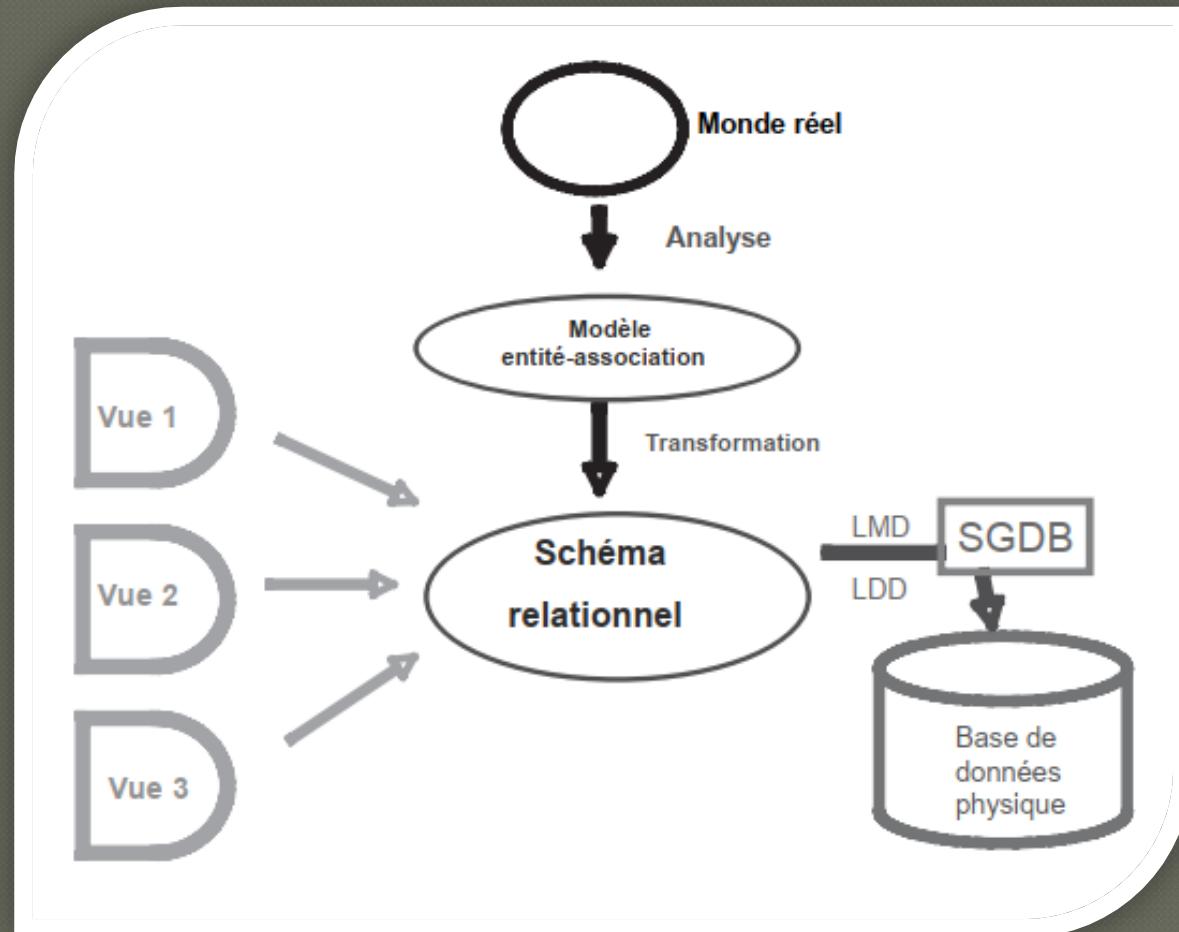
---

- Enfin, nous passons à l'étape de création et d'utilisation de la base de données en utilisant le schéma précédemment défini. Il s'agit de la création des tables qui constituent la base de données, et puis, d'insérer les valeurs dans les tables.
  - La création des tables ainsi que l'utilisation de la base de données nécessitent le *langage SQL*.

# Généralités

## les étapes de la conception

**Étapes de la conception d'une base de données.**



# Généralités

## les différents métiers possibles

---

Dans le processus d'étude et de mise en place d'une base de données, diverses acteurs peuvent y intervenir donnant ainsi naissance à d'intéressants métiers d'**ingénierie des systèmes d'information**. Parmi lesquels, nous avons:

- le métier de *Consultants/Analystes* qui prennent en charge la première étape de conception c'est-à-dire l'étape consistant à analyser les activités et les flux d'information du système (activités d'une entreprise par exemple) à modéliser. Cette phase nécessite beaucoup de dialogue et de psychologie pour bien parvenir à exprimer les besoins des futurs utilisateurs.
  - Il n'est pas toujours nécessaire que l'acteur soit purement technique, il suffit juste qu'il soit capable de correctement exprimer les besoins afin de proposer un modèle conceptuel de donnée (le plus juste possible).

# Généralités

## les différents métiers possibles

- le métier de *Concepteurs de base de données* qui s'occupent de la transformation du modèle conceptuel en un modèle logique pouvant être exploité par le SGBD. C'est le spécialiste qui doit mettre en place les tables, les vues, les schémas d'accès à la base en collaboration avec l'administrateur.
  - Parfois, il peut être analyste et concepteur en même temps, dans ce cas, le cahier des charges aura tendance à être trop technique. Il peut aussi être administrateur, ce qui ne pose aucun souci.
- Le métier d'*Administrateurs de base de données* qui a la responsabilité du fonctionnement général du SGBD.
  - Il crée les ressources(base, comptes) a la demande et donne les droits d'accès aux usagers.
  - Il vérifie la disponibilité des ressources, veille aux défaillance sécuritaire et est en étroite collaboration avec l'administrateur système et réseau de la structure.
- Le métier de *Programmeurs d'application* qui est un utilisateur standard du système d'information. Ils y ont accès grâce aux vues définies par le concepteur de la base. Pour mener à bien leur taches, les développeurs doivent avoir une très bonne connaissance de la SGBD mise en place.

# Généralités

## résumé

- Il est très difficile de donner une définition exacte d'une base de données. Une base de données peut être considérée comme "*un ensemble de fichiers – informatiques ou non – structurés et organisés afin de stocker et de gérer de l'information*".
- Un SGBD peut être perçu comme un ensemble de logiciels systèmes permettant aux utilisateurs de pouvoir efficacement manipuler les données de la base.
- Un SGBD se compose, en première approximation, de trois couches à savoir la couches gestionnaires de fichiers, la couche SGBD interne et la couche SGBD externe.
- Les toutes premières bases de données ont vues le jour vers les années 60 et leurs évolutions ont vu naître diverses générations qui ont suivies le même trajet, en prenant en compte, à chaque étape, les besoins ou exigences des utilisateurs.
- Pour mettre en place une base de données de manière efficace, il est nécessaire de passer par quelques étapes à savoir, l'étude du monde réel, la proposition d'un modèle conceptuel, la traduction du modèle conceptuel en modèle relationnel avant donc de pouvoir mettre en place la base de données.
- Ces différents étapes implique des fois l'intervention de plusieurs acteurs faisant naître des métiers connexes comme le métier de consultant/analiste, le métier de concepteur de base, le métier d'administrateur de base de données et puis le métier de programmeur d'application.

# Généralités

## Discussions

---

- ◉ Q1: Quelles sont les différences majeures entre un fichier informatique et une base de données gérée par un SGBD ?

# Généralités

## Discussions

### Q1: Quelles sont les différences majeures entre un fichier informatique et une base de données gérée par un SGBD ?

- Il n'est pas nécessaire de connaître la méthode de stockage des informations sur le disque pour manipuler les données avec une base de données.
- Un fichier informatique simple n'est pas conçu pour effectuer une recherche d'information par le contenu : pour retrouver les enregistrements, on est obligé de parcourir tout le fichier.
- Les modifications de structure (ajout/suppression d'un champ ou modification de sa taille...) nécessitent de recréer un autre fichier et d'y recopier les données.
- Une base de données contient en général plusieurs fichiers dont les enregistrements sont reliés entre eux.
- Etc.

# Généralités

## Discussions

---

- ◉ Q2: Est-il possible de faire réaliser toutes les étapes de la conception d'une base de données par une même personne ? Si oui, quelles sont alors ses compétences minimales ?

# Généralités

## Discussions

---

● Q2: Est-il possible de faire réaliser toutes les étapes de la conception d'une base de données par une même personne ? Si oui, quelles sont alors ses compétences minimales ?

- Dans une petite structure, c'est souvent la même personne qui réalise l'ensemble du processus de construction d'une base de données. Ce n'est évidemment pas la bonne méthode, car la vision d'un système d'information élaboré par un administrateur de base de données est très orientée par le SGBD qu'il emploiera. On peut facilement faire le parallèle avec le développement de logiciels où un programmeur va avoir une approche déformée par les préoccupations liées au langage plutôt que d'adopter un point de vue sur la structure générale de l'application. Au minimum, la personne devra disposer des compétences en conception de base de données et en administration du SGBD qui sera utilisé

# Généralités

## Discussions

---

- Q3: On décide de recopier régulièrement une base de données complète sur chacun des six sites de l'entreprise. Quel est l'intérêt de cette solution ?

# Généralités

## Discussions

- Q3: On décide de recopier régulièrement une base de données complète sur chacun des six sites de l'entreprise. Quel est l'intérêt de cette solution ?

C'est une solution coûteuse en ressources, en particulier pour la synchronisation de toutes les mises à jour, mais qui peut remplir deux fonctions :

- De toute évidence, l'idée est que les différents sites de l'entreprise accèdent à leur copie locale des données. Cela permet d'accélérer les accès et de répartir la charge sur les serveurs locaux de chaque site. Accessoirement, la circulation des requêtes d'interrogation et de mise à jour peut être limitée au réseau local, ce qui renforce la sécurité.
- Il existe six copies des données sur des sites géographiquement séparés. En cas de sinistre, on peut repartir sans problèmes avec une des copies de la base.

Avant de mettre en place un tel dispositif, on doit se poser la question de la mise à jour des données. Est-ce que les modifications se font uniquement sur la base "maître", ce qui semble plus raisonnable, ou peut-on les effectuer sur toutes les bases et "consolider" ensuite ?

# PLAN

---

- Généralités
- Le modèle conceptuel
- Le modèle relationnel
- Le langage SQL
- Etude et réalisation d'une base de données
- La sécurité des données

# Le modèle conceptuel

objectif

---

L'objectif de ce chapitre est de présenter :

- la première étape du processus de modélisation:
  - Recueil d'information;
  - Transcription pour le passage au modèle relationnel;
- les concepts et la mise en œuvre du modèle entité-association

# Le modèle conceptuel

Démarche d'analyse > approche du monde réel

---

- La question à se poser à cette étape est: *comment appréhender et simplifier le monde réel, afin de pouvoir réaliser la modélisation?* Une question que le consultant doit chercher à répondre à travers ses diverses interventions. Il a la responsabilité d'identifier les besoins des utilisateurs ainsi que les objectifs et les processus d'alimentation en données des systèmes d'information à concevoir.
- Ces différentes étapes se déroulent souvent en même temps que le processus d'**analyse du problème**.
- C'est un processus itératif (des allers-retours entre ces différentes étapes de la conception) permettant de raffiner le modèle conceptuel.
  - L'**entretiens avec les utilisateurs** est la première phase de l'analyse du problème. Il permet effectuer une analyse du discours afin d'en extraire l'**information utile** .

# Le modèle conceptuel

Démarche d'analyse > approche du monde réel

---

- L'**expression des besoins** repose sur la formulation du problème à l'aide de phrases simples qui décrivent la réalité à modéliser. Ces phrases se présentent sous la forme « sujet-verbe-complément », avec une tournure active quand cela est possible. Le but est d'obtenir deux types de phrases :
  - Celles qui décrivent les liens entre les objets du monde réel, généralement une action ou une propriété.
    - **Exemple** : Un lecteur emprunte un livre. Un livre a un auteur
  - Celles qui caractérisent la manière dont sont reliés ces objets (regroupements logiques de données).
    - **Exemple** : Un lecteur est considéré comme lecteur s'il a au moins déjà emprunté un livre. Un livre peut être emprunté par plusieurs lecteurs. Il n'y a pas de livres anonymes, un livre est écrit par au moins un auteur.
- Par la suite, il faut préciser les données qui constituent les objets ainsi que celles qui caractérisent les liens entre les objets.

# Le modèle conceptuel

Démarche d'analyse > mise en œuvre

---

- On est intéressé sur "Comment procéder de manière intuitive pour l'obtention de ces phrases ?".

- Pour mieux répondre à cette question, il faut d'abord faire l'inventaire des objets tangibles du monde réel. Une fois ces objets identifiés, on cherche à exprimer les liens permettant de les associer.
- **Exemple:** si l'on doit modéliser une activité de location de DVD, les objets que l'on peut appréhender immédiatement sont les DVD et les clients. En ce qui concerne les liens entre ces objets, on note qu'un client réserve un DVD ou qu'un client loue un DVD, etc.
- Ensuite, il faut identifier les objets moins faciles à percevoir directement : les fournisseurs, les acteurs, les réalisateurs...
- Enfin, une fois les objets identifiés, on cherche à qualifier les liens trouvés. Il faut tenir compte du fait que le lien est toujours à double sens.
- **Par exemple**, un client emprunte plusieurs DVD. Un DVD est emprunté plusieurs fois ou n'est jamais emprunté par un client.

# Le modèle conceptuel

## Démarche d'analyse > mise en œuvre

- Pour parvenir à ce résultat, il faut savoir "quelles questions faut-il se poser et quelles questions doit-on poser aux acteurs de l'organisation ?".
  - Comment allez-vous décrire l'activité globalement, en termes simples, sans entrer dans les détails, pour identifier les objets et leurs liens éventuels?
  - Demandez quelles sont les « procédures » utilisées dans l'activité afin de pouvoir caractériser les liens existants entre les objets. Les procédures permettent d'énoncer les contraintes qui seront intégrées ensuite dans la base de données.
- L'objectif est de parvenir à modélisé l'action (le lien entre les objets) qui représente une activité (rarement des éléments statiques).
  - **Exemple d'action:** une personne *emprunte* un DVD, une voiture *est achetée* par un client, ... .
- Le **temps** est une **notion** importante puis qu'une base de données modélise des actions qui ont lieu durant une période de temps. Pour éviter des erreurs de conception, il va falloir toujours avoir à l'esprit cet aspect.
  - Une erreur classique est de confondre l'aspect simultané d'une action avec la possibilité de la réitérer durant la période concernée.
  - Lorsque l'on spécifie qu'« un livre peut être emprunté plusieurs fois », il est évident qu'un livre ne peut être emprunté par deux personnes simultanément, mais plutôt qu'il pourra être emprunté à plusieurs reprises durant la période modélisée du fonctionnement de la bibliothèque.

# Le modèle conceptuel

## Démarche d'analyse > Cas pratique

- Pour des besoins d'illustration, on peut essayer de modéliser schématiquement le fonctionnement d'un hôtel.
- 1. Quelles phrases simples pour décrire l'activité de l'hôtel ?
  - En première approche, on peut dire: *Un hôtel loue des chambres à des clients qui effectuent des réservations.*
  - Après analyse, on peut la réécrire de la manière suivante: *Un client loue une chambre ; un client réserve une chambre.*
  - ❖ On a pu identifier deux objets du monde réel à savoir "*la chambre*" et "*le client*" qui sont liés doublement (cas assez fréquent).
- 2. Comment procéder à la caractérisation des liens ?
  - Une chambre peut n'avoir jamais été louée ni réservée.
  - Un client intègre le système à partir du moment où il a effectué soit une réservation, soit une location.
  - Un client peut réserver ou louer plusieurs chambres.
  - Une chambre peut être réservée ou louée plusieurs fois, mais pas pendant la même période de temps.
  - ❖ On considère toujours une modélisation associée à une période de temps donnée. Au début du processus, une chambre peut ne pas encore avoir été louée.
- 3. On procède à la description des données des objets et des liens.
  - Un client est caractérisé par son nom, son adresse et son numéro de téléphone.
  - Une chambre est caractérisée par son numéro, un nombre de places, son tarif journalier et la présence ou non d'un cabinet de toilettes.
  - Une location est caractérisée par une date de début, un nombre de jours et les consommations annexes (petits déjeuners et autres...).
  - Une réservation est caractérisée par une date de début, un nombre de jours et le versement d'une avance éventuelle.

# Le modèle conceptuel

Modélisation > Modèle entité-association

---

- Le **modèle conceptuel** est l'équivalent du schéma technique d'un appareil ou du plan d'un bâtiment. Il permet de donner une vue d'ensemble des données et des liens qui les caractérisent. Nous allons utiliser le **modèle entité-association** (*ou entity-relationship en anglais*) pour schématiser le besoins .
- Le formalisme "entité-association" utilise une représentation graphique sous forme de diagrammes.
  - Les entités sont les objets concrets ou abstraits du monde réel.
  - Les associations représentent le lien entre ces entités.
- Comme on l'a vu précédemment, on peut identifier les entités et les associations en effectuant une analyse du discours, c'est-à-dire des phrases de type "sujet-verbe-complément". Les sujets et les compléments sont les entités, et le verbe modélise l'association.

# Le modèle conceptuel

Modélisation > Entités

- Une **entité** a un nom et est composée de champs de données appelés attributs. Un des attributs (ou un ensemble d'attributs), doit être choisi comme identifiant (notion de clé) de l'entité, afin d'identifier de manière unique une occurrence (ou représentant) de cette entité.
- Une entité est représentée par un rectangle qui contient son nom et ses attributs. L'identifiant est soit souligné ou soit précédé d'un caractère '#'. Voir exemple ci-dessous:

Entité	Occurrence ou représentant	Occurrence ou représentant
Chambre	Chambre	Chambre
# IDChambre	221	75
NombreDePlaces	2	1
Tarif	25 000	15 000
...	...	...

# Le modèle conceptuel

Modélisation > Entités

---

- Le choix de l'identifiant n'est pas toujours évident. Ce qui pousse des fois à introduire un attribut supplémentaire arbitrairement afin de pouvoir disposer d'un identifiant.
- Si on prenait le cas pratique précédent, pour le cas d'un client, comme aucun des attributs issus de l'analyse ne permet de l'identifier de manière unique, il va falloir lui intégrer un attribut servant d'identifiant. Il peut être nommé IDClient.

Client
# IDClient
Nom
Adresse
NumTéléphone
...

# Le modèle conceptuel

Modélisation > Entités

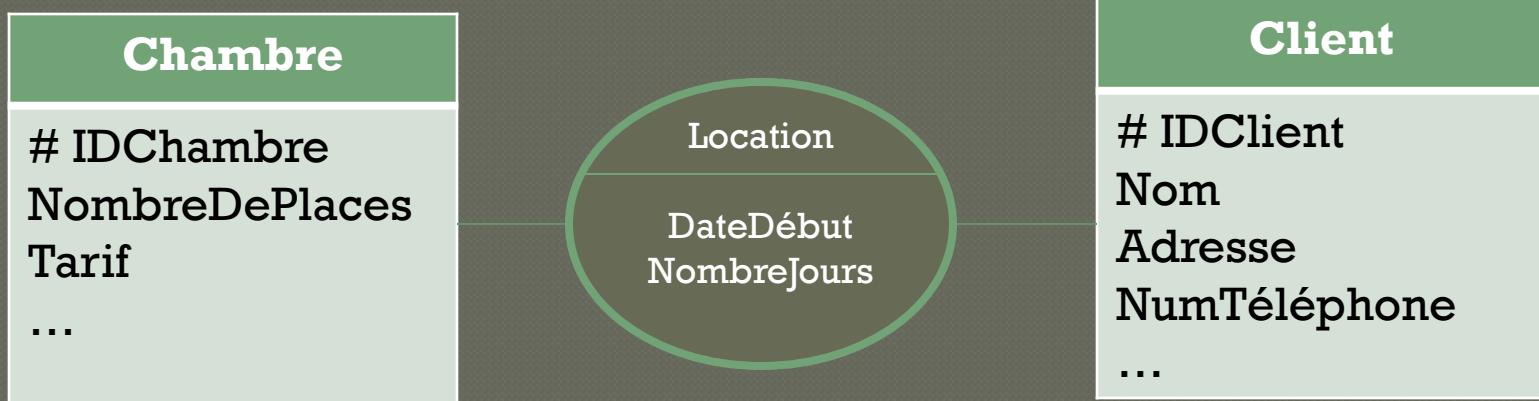
---

- En réalité, un identifiant peut être une juxtaposition de différents attributs. Par exemple, pour un client, on peut avoir "nom+prénom+date\_naissance+ville\_naissance" comme identifiant puisqu'il n'est pas évident de trouver deux clients de même nom et prenom, qui sont nées le même jour et dans la même ville.
- Dans la pratique, il est recommandé de choisir un seul attribut comme identifiant plutôt que de constituer un identifiant composite pour éviter des problèmes d'invalidité lorsque les données évoluent. Par exemple, pour une personne, on préfère l'identifier par un numéro de sécurité sociale plutôt que par un numéro de passeport qui a une durée de validité limitée.

# Le modèle conceptuel

## Modélisation > Associations

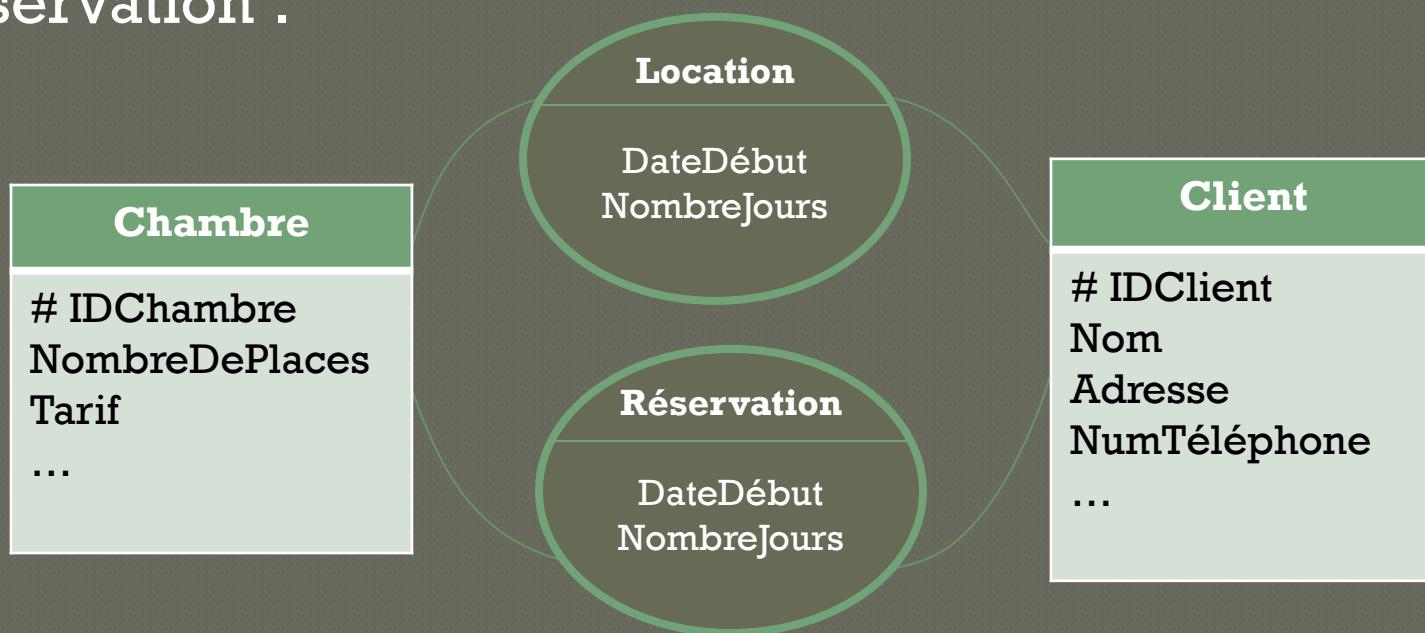
- Une **association** permet de représenter les liens existant entre les entités. Elle peut éventuellement contenir des attributs. Ce qui veut dire que l'identifiant n'est pas nécessaire.
- Les entités sont dites **binaires** lorsqu'elles sont associées par deux. Si c'est plus de deux, on parle de **n-aires**.
- Une association est représentée par un ovale contenant son nom et éventuellement ses attributs.



# Le modèle conceptuel

## Modélisation > Associations

- Deux entités peuvent être liées par plusieurs associations. C'est le cas entre 'client' et 'chambre' qui sont reliées par deux associations, à savoir 'location' et 'réservation'.



# Le modèle conceptuel

## Modélisation > Associations

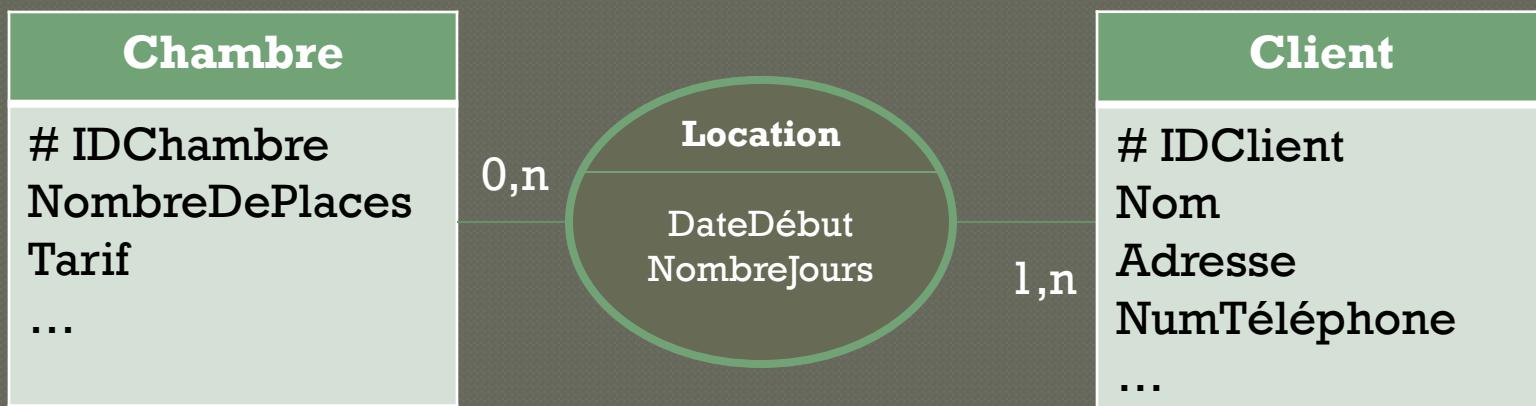
- Une association est dite **réflexive**, si elle relie une même entité. Par exemple, si je prends le cas d'un mariage, "Massamba est marié à Mafatou" , l'entité 'personne' est associée à elle-même par l'association 'est\_marié\_à'.



# Le modèle conceptuel

## Modélisation > Cardinalités

- Les **cardinalités** décrivent les caractéristiques de l'association entre les entités. C'est un couple de valeurs minimales et maximales permettant de caractériser l'association. Ces nombres modélisent le nombre d'occurrences minimales et maximales des entités impliquées dans l'association. Chaque entité participe de manière différente à l'association.



- Une chambre peut être louée plusieurs fois (n) et elle peut ne pas être occupée (0).
- Un client loue au minimum une (1) chambre et il peut en louer plusieurs (n).

# Le modèle conceptuel

Modélisation > Cardinalités

---

Les cardinalités peuvent prendre les valeurs suivantes:

- De un à un, notée 1,1. Exemple: une brosse à dents possède en théorie un (1) et un (1) seul propriétaire.
- De un à plusieurs, notée 1,n. Exemple: un livre a au moins un (1) auteur ; il peut en posséder plusieurs (n).
- Optionnel, notée 0,1. Exemple: une personne (chrétienne) est célibataire (0) ou mariée (légalement...) à une (1) autre personne au plus.
- De zéro à plusieurs, notée 0,n. Exemple: un appartement peut être libre (0) ou habité éventuellement par plusieurs habitants (n).

# Le modèle conceptuel

## Modélisation > Cas pratique

---

On envisage de modéliser la vente d'une voiture.

- On peut facilement identifier deux entités du monde réel que sont client et voiture.
- Une voiture est caractérisée par sa marque, son type, sa couleur.
- Un client (ou une personne) est caractérisé par son nom, son âge, sa ville, son sexe.
- Un client peut acheter plusieurs voitures ou aucune.
- Un client peut acheter aucune ou plusieurs voitures.
- Une voiture peut être vendue ou non.

# Le modèle conceptuel

Modélisation > Cas pratique

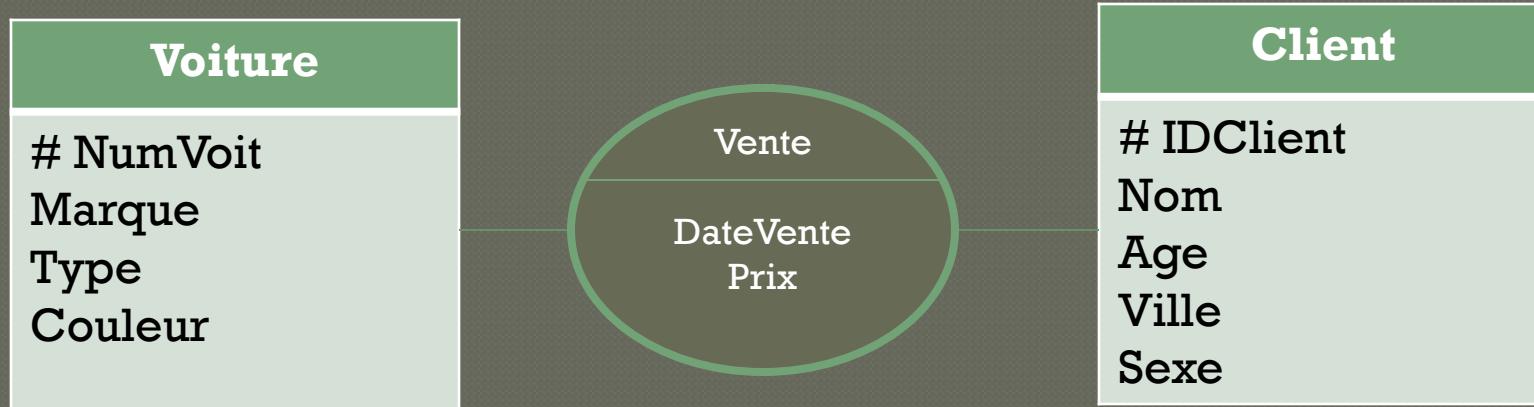
---

- Aucun attribut ni ensemble d'attributs ne permet de définir sans ambiguïté un identifiant. Les identifiants doivent alors être introduits arbitrairement.
- Les cardinalités:
  - Un client peut acheter plusieurs (n) voitures ou aucune (0): 0,n
  - Une voiture peut être vendue une seule fois (1) ou jamais (0): 0,1

# Le modèle conceptuel

Modélisation > Cas pratique

- Ci-dessous, le modèle entité-association:



# Le modèle conceptuel

Raffinage > Qualité des attributs

---

Quelques règles simples pour permettre de guider le choix des attributs.

- Il faut stocker que les attributs strictement nécessaires. Ce choix s'effectue en tenant compte des fonctionnalités attendues du système d'information par élimination systématique des données inutiles. Les personnes ont tendance de prévoir le plus d'attributs possible juste au cas où. Ce qui n'est pas nécessaire ici.
- Il ne faut jamais retenir les attributs qui peuvent être déduits soit par calcul, soit par un lien sémantique d'autres attributs.

## Exemple:

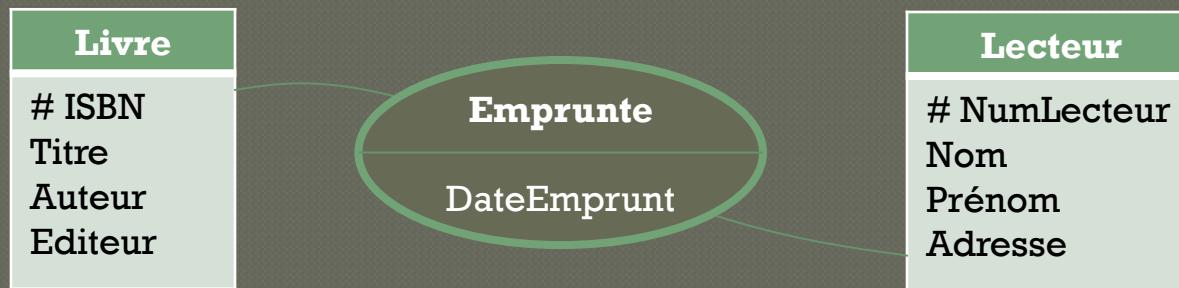
- Un chiffre d'affaire peut être calculé à partir du prix de vente et de la quantité.
  - Une référence peut être construite à partir du nom du produit et du fournisseur.
- Le nom des attributs doit être parlant. Dans ce cas, il ne faut pas hésiter d'utiliser des noms longs si cela facilite la compréhension.  
**Par exemple**, on peut préférer écrire 'Numero\_Serie' plutôt d'écrire 'ns'.

# Le modèle conceptuel

## Raffinage > Réorganisation des entités

Plutôt de découper en entités par une première approche, mieux vaut procéder par une approche itérative. Prenons l'exemple du "*lecteur qui empruntait un livre à la bibliothèque*". Les deux entités déduites sont '*lecteur*' et '*livre*' qui sont liées par l'association '*emprunte*'.

- Un lecteur est caractérisé par son nom, son prénom et son adresse. Comme il n'y a pas d'attribut possédant les caractéristiques d'un identifiant pour l'entité 'lecteur', on ajoute un attribut identifiant (le numéro du lecteur).
- Un livre est caractérisé par son titre, son auteur, son numéro ISBN, son éditeur. Le numéro ISBN est ici un identifiant pour l'entité puisqu'il est unique.
- L'association 'emprunte' a pour attribut la date d'emprunt.



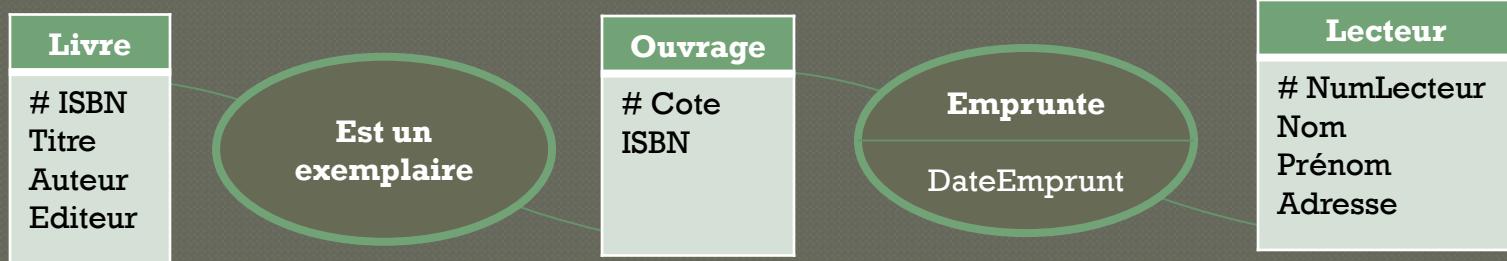
Cette illustration simple permet de se poser quelques questions qui vont conduire à une réorganisation du modèle.

# Le modèle conceptuel

## Raffinage > Réorganisation des entités

Que se passe-t-il si l'on possède plusieurs exemplaires du même ouvrage ?

- Le numéro ISBN n'est plus identifiant puisqu'il est le même pour chacun des exemplaires. Une solution consiste à ajouter un numéro supplémentaire unique pour chaque livre, qui correspond à la notion de cote dans une bibliothèque. Ainsi, même si l'on a dix exemplaires d'un ouvrage, il est possible de les différencier.
  - L'inconvénient de cette solution est que l'on répète les informations communes aux différents ouvrages (titre, auteur...) à chaque exemplaire.
  - L'un des risques est de répéter incorrectement ces informations et d'aboutir ainsi à des incohérences.
- La solution correcte dans ce cas est de séparer l'entité 'livre' en deux entités 'livre' et 'ouvrage'. L'activité de la bibliothèque est alors décrite par deux phrases :
  - Un lecteur emprunte un exemplaire.
  - Un livre représente un exemplaire d'un ouvrage.



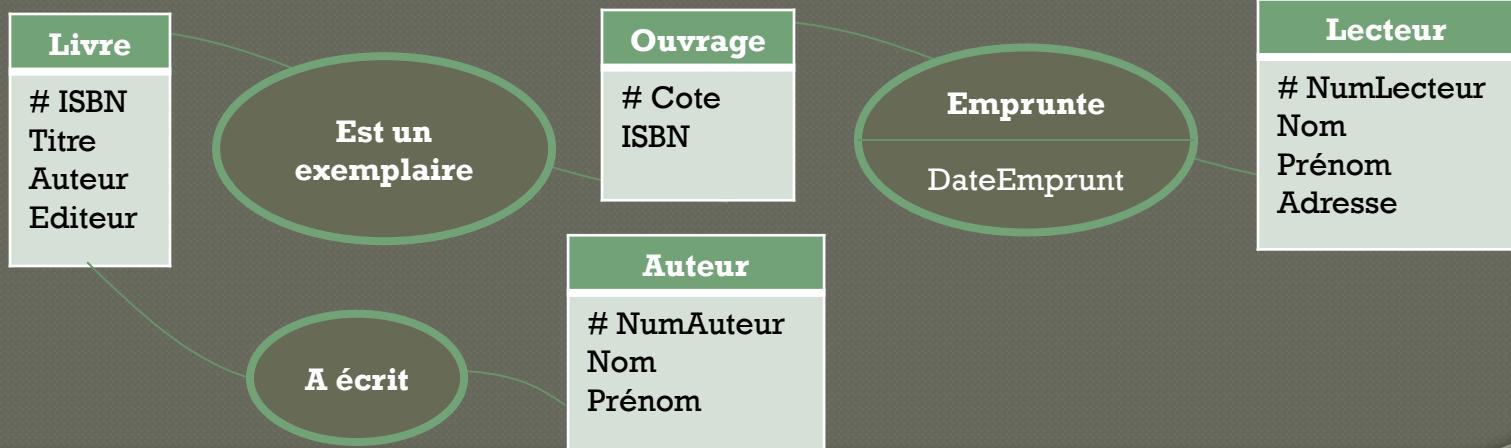
Un ouvrage est caractérisé par un numéro (cote) identifiant et un ISBN. Un livre est caractérisé par un ISBN, qui est bien dans ce cas un identifiant, un titre, un auteur et un éditeur. L'ouvrage est un regroupement d'attributs qui n'a pas d'existence dans le monde réel.

# Le modèle conceptuel

## Raffinage > Réorganisation des entités

Que se passe-t-il si l'ouvrage possède plusieurs auteurs ?

- Une solution simpliste est de prévoir un champ par auteur supplémentaire, c'est-à-dire d'ajouter des champs 'auteur2', 'auteur3', 'auteur4', etc. Cette solution pose de nombreux problèmes :
  - Si seulement dix livres sur un million possèdent plusieurs auteurs, on réserve la place pour les champs auteurs supplémentaires qui sera inutilisée.
  - Si un livre possède un nombre d'auteurs supérieur au nombre de champs prévus, on ne résout pas le problème.
  - Si l'on considère qu'un auteur peut avoir écrit plusieurs ouvrages, on répète dans ce cas les informations le concernant pour chacun de ses ouvrages. Cela constitue un cas typique de redondance qui risque de provoquer des incohérences.
- La solution correcte dans ce cas est de créer une entité supplémentaire pour ces attributs qui sont sémantiquement de même type. On créera ici une entité auteur qui contiendra un numéro d'auteur (identifiant), son nom et son prénom. Cette entité est reliée à l'entité ouvrage par l'association 'a\_écrit'.

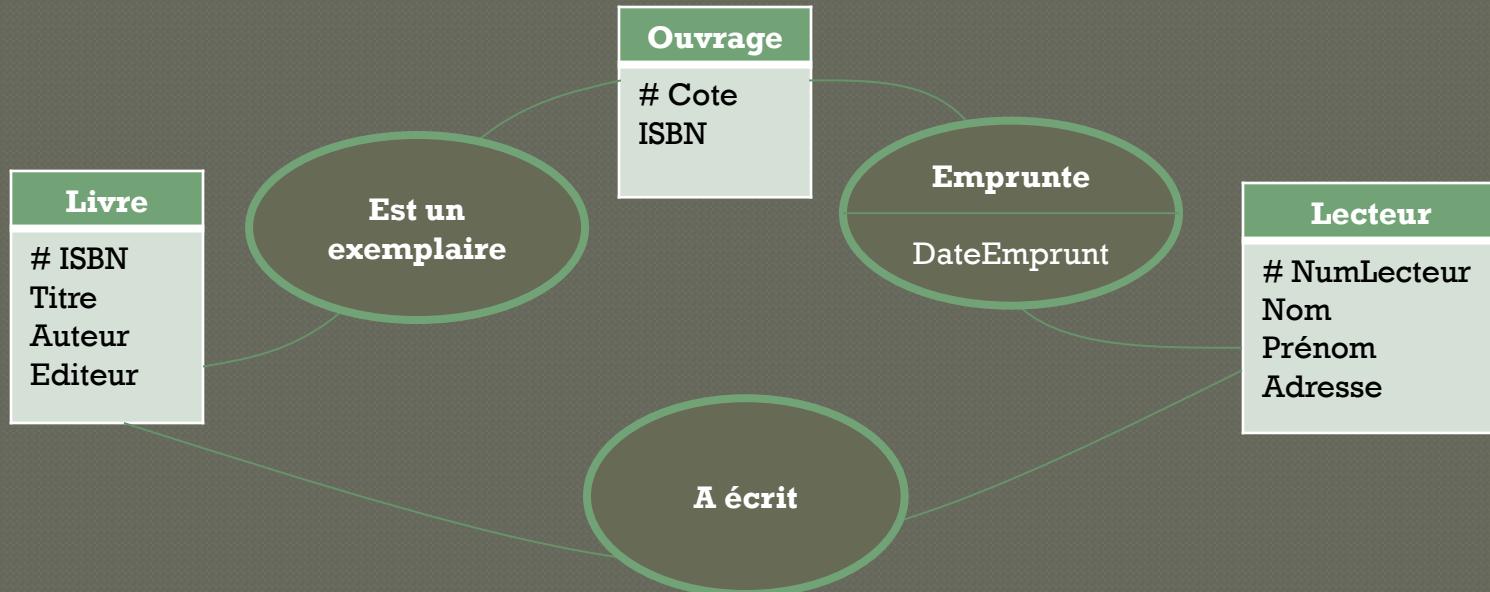


# Le modèle conceptuel

Raffinage > Réorganisation des entités

Que se passe-t-il si un auteur emprunte un livre ?

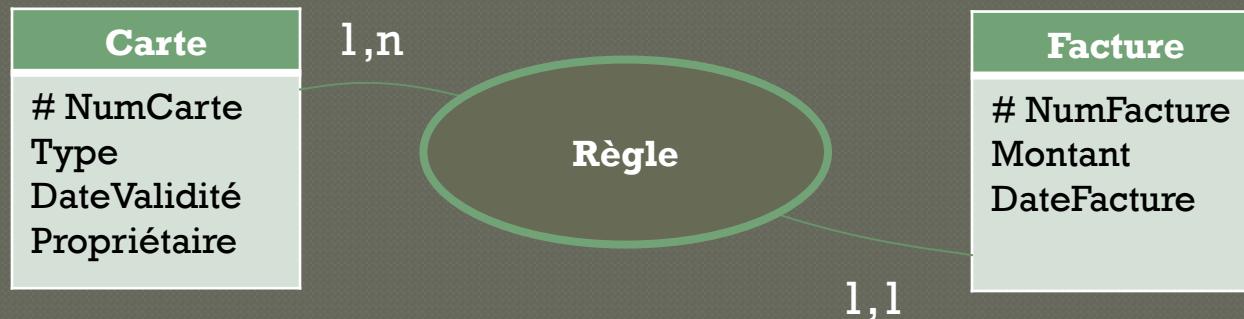
- Un auteur peut également emprunter un livre et revêtir par conséquent le rôle de l'emprunteur. Dans ce cas, on répète les informations le concernant dans l'entité 'lecteur'. Cela provoque de la redondance et peut générer des incohérences.
- Comme les entités 'lecteur' et 'auteur' ont la même structure, la solution consiste à les fusionner en une entité unique 'personne'.



# Le modèle conceptuel

## Raffinage > Elimination d'associations

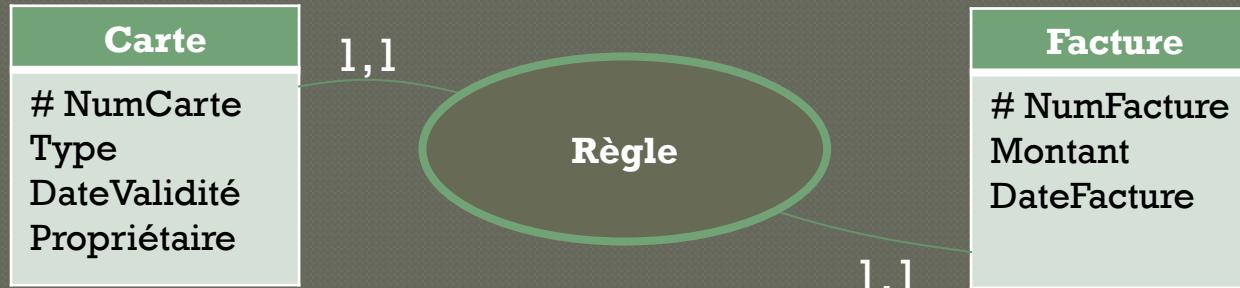
- Si on considère le cas d'un acte d'achat effectué sur Internet avec une carte bancaire. "*Une facture est réglée par une carte*". On peut distinguer les deux entités '*facture*' et '*carte*' .
  - Une facture est identifiée par un numéro de facture, et est constituée d'un montant et d'une date.
  - Une carte bancaire est identifiée par son numéro, son type (Visa, MasterCard), sa date de validité et son propriétaire.
  - Une facture est payée par une et une seule carte (1,1).
  - Une carte peut servir à régler plusieurs factures (1,n).



# Le modèle conceptuel

## Raffinage > Elimination d'associations

- On peut utiliser une E-carte qui permet d'améliorer la sécurité de ces transactions. Une E-carte est une carte "virtuelle" associée à une véritable carte bancaire, valable pour une seule transaction.
  - Une E-carte ne permet de régler qu'une et une seule facture (1,1).



- Dans ce cas, 'règle' n'a plus lieu d'être puisqu'il s'agit d'une pure **bijection**.
  - À une facture correspond une E-carte et une seule, et à une E-carte correspond une facture et une seule.
- On peut donc fusionner les deux entités 'carte' et 'facture' et éliminer l'association

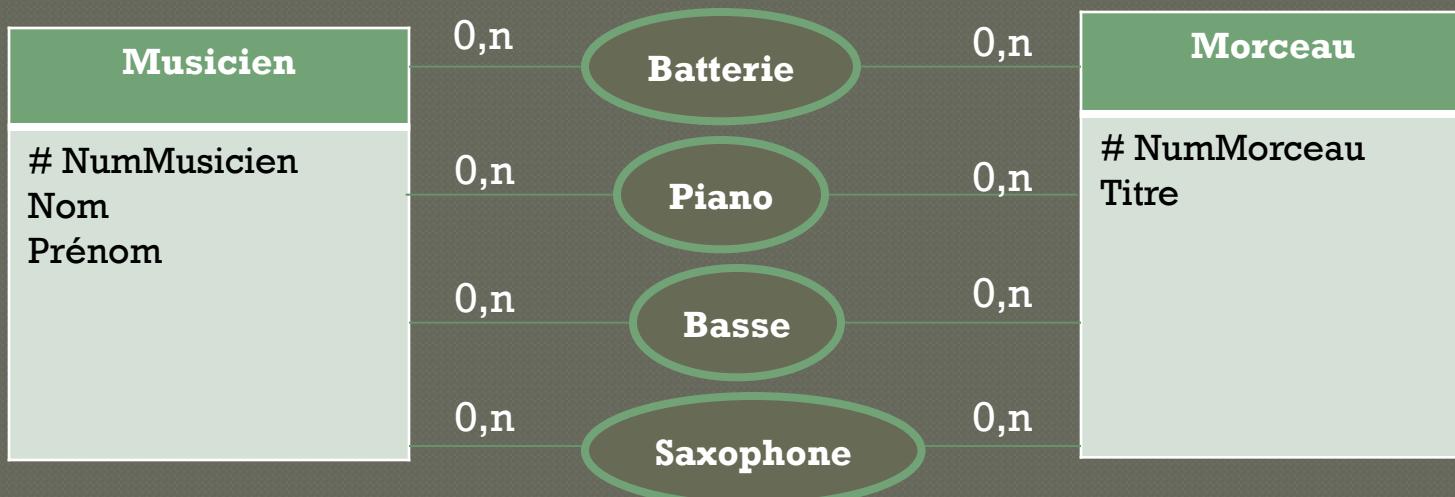
Facture_bis
# NumFacture
Montant
DateFacture
NumCarte
Type
DateValidité
Propriétaire

# Le modèle conceptuel

## Raffinage > Fusion d'associations

Suivant le même principe, il est possible de fusionner plusieurs associations ayant le même rôle sémantique.

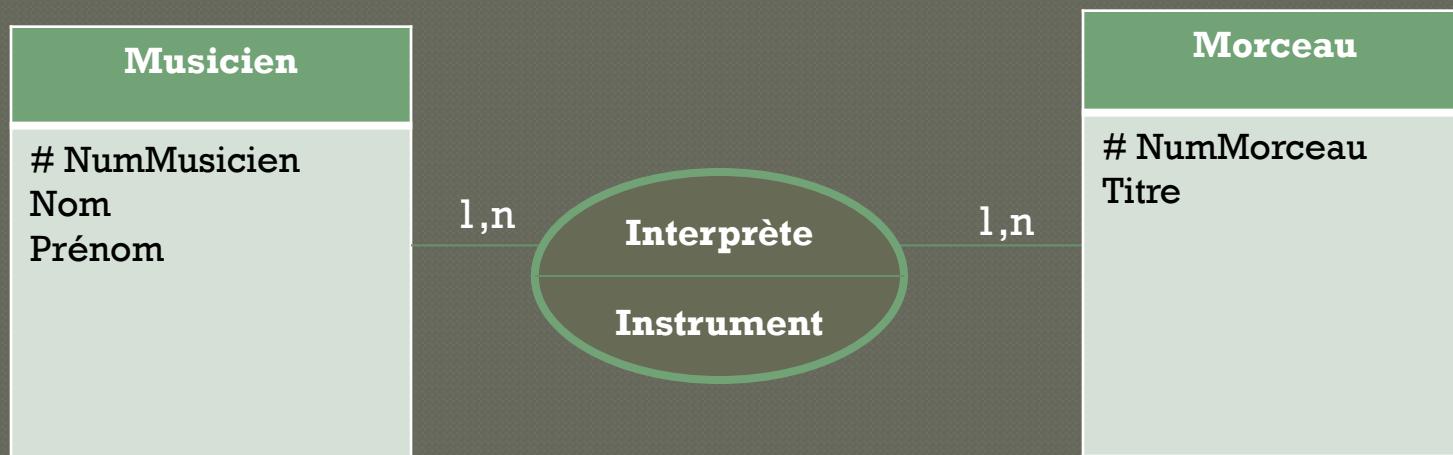
- Si l'on considère la description de l'activité suivante, liée à l'exécution de morceaux de jazz en quartet. Pour un morceau donné, le premier musicien joue la partie de basse, le deuxième celle de batterie, le troisième celle de piano et le quatrième celle de saxophone. Cela donne un schéma du genre:



# Le modèle conceptuel

Raffinage > Fusion d'associations

- Comme il s'agit d'une même activité, "un musicien interprète un morceau", on peut remplacer toutes les associations par une seule 'interprète' contenant l'attribut 'instrument'.



# Le modèle conceptuel

## Résumé

- Le but de ce chapitre consiste d'abord à extraire les informations du monde réel puis à en proposer une représentation formelle appelée modèle conceptuel.
  - Pour parvenir à cela, il faut d'abord identifier les données qui interviennent dans le domaine considéré et de les regrouper dans des objets.
  - Ensuite, caractériser les liens qui les unissent.
  - Enfin, on décrit le système à modéliser par des phrases simples de type "sujet-verbe-complément", que l'on peut classer en deux grandes catégories:
    - celles qui décrivent les objets du monde réel et les liens qui les unissent : le sujet et le complément sont représentés par des entités et le verbe est représenté par une association ;
    - celles qui décrivent la manière dont sont reliés ces objets : on en déduira les cardinalités des associations.
- Les entités sont constituées de champs de données que l'on nomme "attributs".
- Pour identifier de manière unique les représentants d'une entité, appelés également "instance", un attribut (ou un ensemble d'attributs) est choisi: on l'appelle "identifiant".

# Le modèle conceptuel

## Applications

---

### ● Enoncé 1:

On veut modéliser l'activité de vente de billets pour un théâtre. Quelles phrases vont nous permettre d'identifier les entités et la manière dont elles sont associées ?

- Proposez les attributs que vous utiliseriez pour décrire ces entités et leurs associations ainsi que les identifiants de chaque entité.
- Que se passe-t-il si le prix du billet varie pour chaque séance et en fonction de la place ?

# Le modèle conceptuel

## Applications

---

### ● Enoncé 2:

On veut représenter les liens de nourriture entre des humains, des animaux et des végétaux. L'idée, à partir des schémas d'alimentation modélisés, est de pouvoir déduire des chaînes alimentaires de ce type : "un homme mange un lapin qui mange des carottes".

# Le modèle conceptuel

## Applications

---

### ● Enoncé 3:

À partir de la base de données exemple de vente de voitures, on souhaite ajouter les informations concernant le vendeur qui a réalisé la vente. Proposez une (ou plusieurs) modification(s) du modèle entité-association élaboré précédemment. Ajoutez les nouvelles cardinalités introduites par cette modification.

# PLAN

---

- Généralités
- Le modèle conceptuel
- Le modèle relationnel
- Le langage SQL
- Etude et réalisation d'une base de données
- La sécurité des données

# Objectif

---

- Dans ce chapitre, nous présentons le concept de relation fondamental du modèle relationnel, ainsi que les opérations qui lui sont associées.
- Ensuite, nous aborderons les méthodes qui permettent de passer du modèle conceptuel à un ensemble de relations.
- Enfin, nous présenterons quelques méthodes de résolution les incohérences ainsi que la forme normale de Boyce-Codd.

# Concept de base

---

- Le modèle relationnel tire son nom de la notion de relation (mathématique) entre des éléments. Chacun de ces éléments peut prendre des valeurs dans un ensemble défini.
  - Prenons le cas d'une cuisine constituée d'**appareils** (réfrigérateur, cuisinière, hotte, robot, lave-vaisselle) de diverses **couleurs** (rouge, bleu, vert, jaune, blanc, noir, rose, jaune).
  - On pourra donc avoir des couples de valeurs suivantes:
    - (réfrigérateur, rouge)
    - (robot,mauve)
    - (cuisinière,jaune)
    - (lave-vaisselle,rouge)
  - Cet ensemble de couples de valeurs liées entre elles, nommé **tuples**, ou «nuplets» ou «enregistrements» dans le modèle relationnel, représente la relation entre les éléments 'appareil' et 'couleur'.
  - On désigne également les éléments constitutifs de ces couples par les termes «attributs» ou «champs».

# Concept de base

---

- La relation peut être écrite formellement de la manière suivante : *ma\_cuisine(appareil, couleur)*.
  - Cette écriture représente le schéma relationnel de la relation 'ma\_cuisine'.
  - Les valeurs énoncées précédemment pour les champs représentent leurs domaines, c'est-à-dire les ensembles de toutes les valeurs possibles pour un champ.
- Une relation est décrite par:
  - le schéma relationnel ;
  - les domaines des différents champs ;
  - les tuples qui la constituent.
- Le nombre de champs de la relation s'appelle son degré de la relation. La relation 'ma\_cuisine' est de degré 2.
- Le nombre de tuples se nomme la cardinalité de la relation. La relation 'ma\_cuisine' est de cardinalité 4.

# Concept de base

- On représente une **relation** par une **table**, correspondant à la notion de tableau.
  - Les **tuples** correspondent aux **lignes** et les **colonnes** aux **champs** de la relation.

Appareil	Couleur
Réfrigérateur	Rouge
Robot	Mauve
Cuisinière	Jaune
Lave-vaisselle	Rouge

- Dans le **modèle relationnel**, la **relation** est l'élément fondamental.
  - Toutes les opérations sur une ou plusieurs relations retourneront une relation.
  - Un ensemble de relations reliées entre elles par des liens sémantiques constitue une **base de données**.

# Concept de base

## NOTION DE CLÉ D'UNE RELATION ET DÉPENDANCE FONCTIONNELLE

- ◎ Dans une **base de données**, pour accéder à un enregistrement par le contenu d'un ou de plusieurs champs il faut une **clé**.
  - On nomme **clé d'une relation**, un champ, ou un ensemble de champs, d'une relation qui permet d'identifier de manière *unique un enregistrement*.
  - Cette clé doit être minimale (doit contenir le minimum de champs) et est appelée **la clé primaire** de la relation.
  - Une relation peut comprendre plusieurs clés possibles. Dans ce cas, sont **les clés candidates**.

# Concept de base

NOTION DE CLÉ D'UNE RELATION ET DÉPENDANCE FONCTIONNELLE

- Une clé ne peut être déduite simplement à partir du contenu de la relation; on ne peut préjuger du contenu futur des enregistrements.
  - **Par exemple**, si l'on prenait la relation 'ma\_cuisine', le champ 'Appareil' semble être une clé puisqu'il contient une valeur unique pour chacun des enregistrements. **Cependant**, il est tout à fait possible que la cuisine considérée comprenne un autre réfrigérateur de couleur bleue, auquel cas la valeur ne serait plus unique et ne permettrait pas de retrouver l'enregistrement. **Dans ce cas de figure**, la combinaison 'Appareil' 'Couleur' pourrait sembler être une clé, mais on ne peut en être certain, compte tenu de l'**évolution des données**.

# Concept de base

NOTION DE CLÉ D'UNE RELATION ET DÉPENDANCE FONCTIONNELLE

- Pour désigner une **clé primaire**, il faut donc également prendre en compte le «sens» des données dans la vie réelle. *Les relations qui existent entre les différents champs d'une relation vont être importantes:* on exprime ces relations à l'aide de **dépendances fonctionnelles**.
  - Une *dépendance fonctionnelle* existe entre deux ensembles de champs si les valeurs contenues dans l'un des ensembles de champs permettent de déterminer les valeurs contenues dans l'autre ensemble.

# Concept de base

## NOTION DE CLÉ D'UNE RELATION ET DÉPENDANCE FONCTIONNELLE

- Si on considère l'exemple suivant, correspondant à la relation Lecteur(Numero\_carte, Nom, Age, Ville, Etablissement), modélisant les lecteurs d'une bibliothèque.

Numero_carte	Nom	Age	Ville	Etablissement
1	Henri	10	Paris	Université Sorbonne
2	Stanislas	34	Paris	Université Jussieu
3	Henriette	44	Lyon	CHU Bron
4	Dominique	19	Nancy	Université Poincaré
5	Isabelle	56	Nancy	INPL
6	Olivier	51	Marseille	Université Saint Charles
7	Henri	98	Paris	Université Sorbonne
8	Jerome	23	Nancy	INPL
9	Laurence	34	Bordeaux	Université Victor Segalen
10	Christian	41	Paris	Ecole Normale Supérieure
11	Antoine	16	Marseille	Université Saint Charles
12	Laurence	34	Paris	Université Jussieu

- En **examinant les données**, on constate qu'il ne peut y avoir dépendances fonctionnelles puisque qu'il existe un enregistrement pour lequel les valeurs des champs (Nom, Age) correspondent à deux valeurs différentes de (Ville, Etablissement).

# Concept de base

NOTION DE CLÉ D'UNE RELATION ET DÉPENDANCE FONCTIONNELLE

---

- Si on considère que dans une ville, il ne peut y avoir qu'un établissement et un seul, cela signifiera alors qu'il existe une relation de dépendance entre les champs '**Etablissement**' et '**Ville**'. A une valeur donnée de '**Etablissement**' correspond bien une valeur unique de '**Ville**'.
- On constate bien que la valeur du champ '**Numero\_carte**' est unique pour chacune des personnes et ses valeurs sont identifiantes pour tous les autres champs de la relation c'est-à-dire, chaque champ dépend fonctionnellement du champ '**Numero\_carte**'.
  - Ses valeurs sont uniques et jamais vides : c'est une clé candidate.
  - C'est d'ailleurs la seule clé possible car les autres champs n'ont jamais de valeur unique.
  - Il est, dans ce cas, choisi comme **clé primaire** de la relation.

# Concept de base

COHÉRENCE DES DONNÉES ET CONTRAINTES D'INTÉGRITÉ

---

Pour garantir la cohérence des données sur l'ensemble des relations constitutives de la base de données, on applique des restrictions sur le contenu des données que l'on nomme **contrainte d'intégrité**. La vérification de la cohérence se situe à plusieurs niveaux :

- Adapter le contenu des données:

On parle de la cohérence par rapport au sens des données qui est liée à la notion de domaine de valeurs du champs (on parle de **contrainte de domaine** ou de **cohérence sémantique**). *Par exemple, l'âge d'une personne ne peut être négatif ou excéder 120. Ces contraintes sont définies lors de l'étape d'analyse du monde réel et leur mise en œuvre effective interviendra au moment de la création de la relation et sera réalisée par le SGBD.*

# Concept de base

## COHÉRENCE DES DONNÉES ET CONTRAINTES D'INTÉGRITÉ

### □ Préserver la cohérence des données:

Comme une base de données est constituée par un ensemble de relations reliées entre elles, les contenus des champs capables de lier ces relations doivent être cohérents entre eux pour pouvoir effectuer l'opération de jointure.

*Si l'on considère l'exemple de la base de données 'casse', on ne doit pas permettre la saisie d'une valeur identifiant une voiture dans la relation 'vente' qui n'existe pas dans la relation 'voiture'.*

- On fait dans ce cas référence au contenu d'une colonne d'une autre relation (le champ 'NumVoit' de la relation 'voiture' afin de contrôler le contenu du champ 'NumVoit' de la relation 'vente').
- Le champ 'NumVoit' est alors une **clé étrangère** qui permet de réaliser la notion **d'intégrité référentielle**.

ces références sont aussi déterminées lors de la phase d'analyse du monde réel et le SGBD permet de les réaliser.

ces contraintes sont généralement appliquées non pas lors de la création des relations, mais plutôt lorsque les données ont déjà été insérées, en particulier dans les relations de références afin d'éviter des problèmes de références impossibles à résoudre entre les relations, ce qui provoque parfois l'incapacité à insérer des données.

# Concept de base

COHÉRENCE DES DONNÉES ET CONTRAINTES D'INTÉGRITÉ

## □ Éliminer la redondance des données:

Les incohérences provoquées par la redondance d'information représentent le principal souci du concepteur d'une base de données. En effet, lorsque les données sont dupliquées, aucun mécanisme ne peut garantir que le changement de la valeur d'une donnée est répercuté correctement sur les autres données. Dans *l'exemple de la relation des lecteurs de bibliothèque*, on remarque **la redondance d'information** qui existe entre les champs 'Ville' et 'Etablissement'. Ici, si le nom de l'établissement 'INPL' change, il faut mettre à jour toutes les lignes qui contiennent son nom.

- La redondance est mise en évidence par la dépendance fonctionnelle qui existe entre ces deux champs.
- Si dans cet exemple, la détection de la redondance est évident, il ne sera pas toujours le cas lorsque le nombre de relations est élevé ou encore si le sujet modélisé par la base de données n'est pas familier.
- Les incohérences de ce type seront résolues par la réorganisation des relations lors de la phase de **normalisation**.

# Les opérations

ENSEMBLISTES

---

Pour pouvoir manipuler les données dans le modèle relationnel, on fait appel aux opérations formelles qui sont pour la plupart issues de la théorie des ensembles. Parmi lesquelles, nous avons:

- Les opérations ensemblistes

Ces opérations sont utilisés que pour des relations ayant la même structure ou pour des structures compatibles, à l'exception du produit cartésien. Elles sont de type binaire(appliquant à deux relations seulement).

- L'union

L'opération d'union consiste en la mise en commun des enregistrements de chaque relation en intégrant en une seule fois ceux qui sont identiques. Elle est représentée par le caractère «  $\cup$  » et sert typiquement à la « consolidation » de données de même type provenant de différentes sources.

# Les opérations

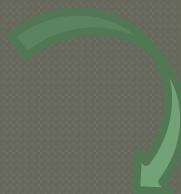
ENSEMBLISTES

- L'exemple ci-dessous, illustre cette opération:

Lecteur_1(Numero_carte, Nom, Age, Ville, Etablissement)				
Numero_carte	Nom	Age	Ville	Etablissement
1	Henri	10	Paris	Université Sorbonne
2	Stanislas	34	Paris	Université Jussieu
3	Henriette	44	Lyon	CHU Bron

Lecteur\_2(Numero\_carte, Nom, Age, Ville, Etablissement)

Numero_carte	Nom	Age	Ville	Etablissement
3	Henriette	44	Lyon	CHU Bron
4	Dominique	19	Nancy	Université Poincaré
5	Isabelle	56	Nancy	INPL



Lecteur\_1 ∪ Lecteur\_2

Numero_carte	Nom	Age	Ville	Etablissement
1	Henri	10	Paris	Université Sorbonne
2	Stanislas	34	Paris	Université Jussieu
3	Henriette	44	Lyon	CHU Bron
4	Dominique	19	Nancy	Université Poincaré
5	Isabelle	56	Nancy	INPL

Numero_carte	Nom	Age	Ville	Etablissement
2	Henri	10	Paris	Université Sorbonne
3	Stanislas	34	Paris	Université Jussieu

# Les opérations

ENSEMBLISTES

## ➤ La différence

L'opération différence consiste à désigner les enregistrements qui appartiennent à une relation sans appartenir à l'autre. Elle est représentée par le caractère « - », sert d'éliminer des enregistrements d'une relation par rapport à une liste et n'est pas symétrique.

Lecteur\_1 – Lecteur\_2

Numero_carte	Nom	Age	Ville	Etablissement
1	Henri	10	Paris	Université Sorbonne
2	Stanislas	34	Paris	Université Jussieu

# Les opérations

ENSEMBLISTES

## ➤ L'intersection

L'opération intersection désigne les enregistrements qui sont communs aux deux relations. Elle est représentée par le caractère «  $\cap$  » et sert à trouver les éléments communs à deux relations.

**Lecteur\_1  $\cap$  Lecteur\_2**

Numero_carte	Nom	Age	Ville	Etablissement
3	Henriette	44	Lyon	CHU Bron

# Les opérations

ENSEMBLISTES

## ➤ Le produit cartésien

Le produit cartésien permet la combinaison des enregistrements de deux relations sans tenir aucun compte du contenu des données. Les relations n'ont donc pas besoin d'avoir la même structure. Le caractère représentant le produit cartésien est « X ».

ma_cuisine(Appareil, Couleur)	
Appareil	Couleur
Réfrigérateur	rouge
Robot	mauve
Cuisinière	jaune

musicien(Nom, Instrument)	
Nom	Instrument
Jaco Pastorius	Basse électrique
Bill Evans	Piano

Bill Evans	Jaco Pastorius
Bill Evans	Jaco Pastorius



Le\_resultat(Appareil, Couleur, Nom, Instrument) = ma\_cuisine(Appareil, Couleur) X  
musicien(Nom, Instrument)

Appareil	Couleur	Nom	Instrument
réfrigérateur	rouge	Jaco Pastorius	Basse électrique
réfrigérateur	rouge	Bill Evans	Piano
robot	mauve	Jaco Pastorius	Basse électrique
robot	mauve	Bill Evans	Piano
cuisinière	jaune	Jaco Pastorius	Basse électrique
cuisinière	jaune	Bill Evans	Piano

# Les opérations

RELATIONNELLES

- Les opérations relationnelles
  - La projection

La projection consiste à extraire certains champs de la relation, ce qui donne à cette dernière un degré inférieur à la relation de départ. Par exemple, sur la relation ‘Lecteur’, on peut projeter les champs ‘Nom’ et ‘Ville’ et on obtient la relation suivante:

Lecteur\_proj(Nom, Ville)

Nom	Ville
Henri	Paris
Stanislas	Paris
...	...

# Les opérations

RELATIONNELLES

- Les opérations relationnelles
  - La sélection ou restriction

La sélection consiste à extraire les enregistrements de la relation en utilisant des critères pour les caractériser (les enregistrements sélectionnés). La structure de la relation résultat est la même que celle de la relation de départ. Par exemple, sur la relation ‘Lecteur’, on peut effectuer une sélection des enregistrements dont le contenu du champs ‘Ville’ vaut ‘Marseille’.

Lecteur\_sel(Numero\_carte, Nom, Age, Ville, Etablissement)

Numero_carte	Nom	Age	Ville	Etablissement
6	Olivier	51	Marseille	Université Saint Charles
11	Antoine	16	Marseille	Université Saint Charles

# Les opérations

RELATIONNELLES

## ➤ La Jointure

La jointure permet d'exprimer le sens du lien existant entre les relations dans le monde réel. La liaison s'effectue par le contenu commun d'un champ. Par exemple, si on considère les deux relations Lecteur\_bis(Numéro\_carte, Nom, Num\_Etablissement) et Etablissement(Num\_Etablissement, Ville, Nom\_Etablissement):

Lecteur\_bis(Numero\_carte, Nom, Num\_Etablissement)

Numero_carte	Nom	Num_Etablissement
1	Henri	1
2	Stanislas	2
3	Henriette	1

Etablissement(Num\_Etablissement, Ville, Nom\_Etablissement)

Num_Etablissement	Ville	Nom_Etablissement
1	Paris	Université Jussieu
2	Lyon	CHU Bron
3	Nancy	Université Poincaré
4	Paris	Université Sorbonne

Lecteur\_joint\_Etablissement(Numero\_carte, Nom, Num\_Etablissement\_1, Num\_Etablissement\_2, Ville, Nom\_Etablissement)

Numero_carte	Nom	Num_Etablissement_1	Num_Etablissement_2	Ville	Nom_Etablissement
1	Henri	1	1	Paris	Université Jussieu
2	Stanislas	2	2	Lyon	CHU Bron
3	Henriette	1	1	Paris	Université Jussieu

jointure naturelle

Lecteur\_jointnat\_Etablissement(Numero\_carte, Nom, Num\_Etablissement, Ville, Nom\_Etablissement)

Numero_carte	Nom	Num_Etablissement	Ville	Nom_Etablissement
1	Henri	1	Paris	Université Jussieu
2	Stanislas	2	Lyon	CHU Bron
3	Henriette	1	Paris	Université Jussieu

# Les opérations

RELATIONNELLES

## ➤ La Jointure

L'opération de jointure peut être vue comme une **sélection des enregistrements obtenus par le produit cartésien des relations**, dont les contenus du champ sur lequel on effectue la jointure sont égaux. On l'appelle dans ce cas une **équijointure**. Les champs dont on compare les contenus sont nommés **champs de jointure**.

Produit_cartesien_Etablissement(Numero_carte, Nom, Num_Etablissement_1, Num_Etablissement_2, Ville, Nom_Etablissement)					
Numero_carte	Nom	Num_Etablissement_1	Num_Etablissement	Ville	Num_Etablissement_2
1	Henri	1	1	Paris	Université Jussieu
1	Henri	1	2	Lyon	CHU Bron
1	Henri	1	3	Nancy	Université Poincaré
1	Henri	1	4	Paris	Université Sorbonne
2	Stanislas	2	1	Paris	Université Jussieu
2	Stanislas	2	2	Lyon	CHU Bron
2	Stanislas	2	3	Nancy	Université Poincaré
2	Stanislas	2	4	Paris	Université Sorbonne
3	Henriette	1	1	Paris	Université Jussieu
3	Henriette	1	2	Lyon	CHU Bron
3	Henriette	1	3	Nancy	Université Poincaré
3	Henriette	1	4	Paris	Université Sorbonne

# Les opérations

RELATIONNELLES

## ➤ La Jointure externe

Il s'agit d'une opération de jointure étendue qui inclut dans le résultat les lignes n'ayant pas de correspondance sur le contenu du champ de jointure. Elle permet de répondre plus facilement à des questions du type « Quels sont les établissements qui n'ont pas de lecteurs ? ». Ici, on met en correspondance les valeurs du champ 'Num\_Etablissement' de toutes les lignes de la relation 'Etablissement' avec celles de la relation 'Lecteur'.

Etablissement\_jointext\_Lecteur(Num\_Etablissement\_1, Ville, Nom\_Etablissement,  
Numero\_carte, Nom, Num\_Etablissement\_2)

Num_Etablissement_1	Ville	Nom_Etablissement	Numero_carte	Nom	Num_Etablissement_2
1	Paris	Université Jussieu	1	Henri	1
2	Lyon	CHU Bron	2	Stanislas	2
3	Nancy	Université Poincaré	NULL	NULL	NULL
4	Paris	Université Sorbonne	NULL	NULL	NULL

# Les opérations

Calculs et agrégats

## ● Les Calculs et agrégats

En base de données, tout ce qui peut se calculer ne doit pas être stocké afin d'éviter des informations redondantes pouvant engendrer des incohérences et une perte de place. Des **fonctions de calculs** ont été définies afin de répondre à ce besoin. Par exemple, en considérant la relation La\_boutique(Num\_facture, Article, Prix, Quantite) suivante:

La\_boutique(Num\_Facture, Article, Prix, Quantite)

Num_Facture	Article	Prix	Quantite
101	Bananes	12	45
1034	Choux	5	129
2345	Riz	4	60
0987	Gazelle	15	48

Il est alors possible d'ajouter un champ 'Total', par exemple, dont le contenu sera calculé par l'expression 'Prix' \* 'Quantité'.

La\_boutique\_total(Num\_Facture, Article, Prix, Quantite, Total)

Num_Facture	Article	Prix	Quantite	Total
101	Bananes	12	45	540
1034	Choux	5	129	645
2345	Riz	4	60	240
0987	Gazelle	15	48	720

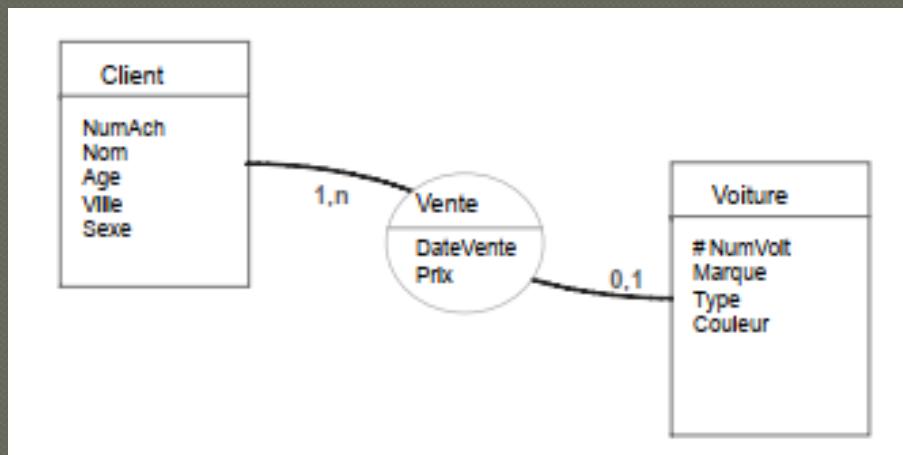
# Passage du modèle conceptuel au relationnel

## ○ Règles générales

En général, nous avons que deux règles avec quelques exceptions que nous utilisons pour simplement d'obtenir un schéma plus compact.

- Une **entité** devient une relation composée des champs de l'entité. La clé de cette relation est la même que celle de l'entité.
- Une **association** devient une relation composée des deux clés des entités associées. S'il possède des champs, ils deviendront les champs de cette nouvelle relation.

Si on prenait l'exemple de la ‘casse’ (vente de voiture):



L'entité client devient:

`client(NumAch, Nom, Age, Ville, Sexe);`

Celle de voiture:

`voiture(#NumVoit, Marque, Type, Couleur);`

Et l'association se transforme en relation aussi:

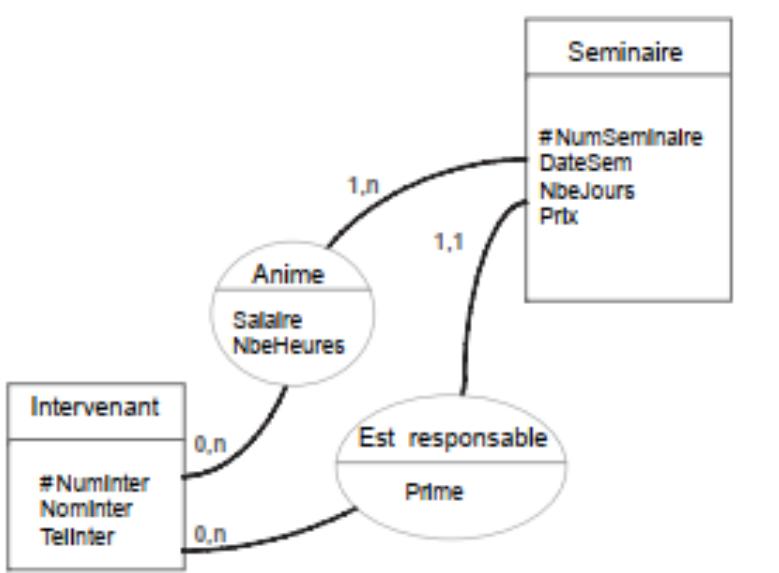
`vente(DateVent, Prix, NumAch, NumVoit).`

# Passage du modèle conceptuel au relationnel

## ○ Cas particulier des associations de type ‘1-1’

Considérons le cas du modèle entité-association qui représente l’activité d’organisation de séminaires décrite très succinctement de la manière suivante :

- Un séminaire est animé par un ou plusieurs intervenants.
- Un séminaire ne possède qu’un seul responsable.



On constate l’existence de deux associations entre les entités ‘Séminaires’ et ‘Intervenant’ : ‘**Anime**’ et ‘**Est\_responsable**’.

Si on applique les règles générales, on obtient quatre relations:

- Intervenant(NumInter, NomInter, TellInter)
- Séminaire(NumSem, DateSem, Prix, NbeJour)
- Anime(NumInter, NumSem, NbeHeure, SalaireHor)
- Est\_responsable(NumInter, NumSem, Prime)

## Passage du modèle conceptuel au relationnel

---

L'association 'Est\_responsable' est de type '1-1' et donc la clé de la relation n'est pas minimal car l'identifiant du séminaire détermine celui du responsable c'est-à-dire qu'il existe une dépendance fonctionnelle entre les champs 'NumSem' et 'NumInter'.

On choisit le champ 'NumSem' comme clé de la relation 'Est\_responsable', nous aurons alors: Est\_responsable(NumSem, NumInter, Prime).

Dans ce cas, les relations 'Est\_responsable' et 'Seminaire' ont alors la même clé, et donc, il va falloir les regrouper. On obtient la relation Seminaire\_Res(NumSem, DateSem, Prix, NbeJour, NumInter, Prime).

De cette démarche, on peut en déduire une règle complémentaire:

- Lorsque l'association entre deux entités comprend une cardinalité de type '1-1', on ne crée pas de relation pour l'association. Les champs de l'association seront intégrés à la relation créée pour l'entité qui est associée avec la cardinalité '1-1'. La clé de cette relation est toujours celle de l'entité.

# Normalisation

La normalisation permet de vérifier une certaines nombres de propriétés afin de tester la qualité et la cohérence des relations et d'apporter des modifications au cas échéant.

## ○ Première forme normale

La première forme normale s'intéresse au contenu des champs. Elle n'accepte pas la présence de plusieurs valeurs (multivaluation) dans un même champ. Celle-ci est, en effet, lié à la notion de dépendance fonctionnelle entre les champs qui ne peut plus être vérifiée s'ils possèdent plusieurs valeurs. Elle s'exprime de la manière suivante :

➤ *Tout champ contient une valeur atomique.*

Considérons l'exemple suivant:

Publication(NumPubli, Titre, Auteurs)

NumPubli	Titre	Auteurs
13490	Le vin et l'avenir	Jean Lasso, Hubert De la Tuque, Stanislas Wilski
21322	Bière et progrès social	Aristide Salem, Jean Lasso, Salome Dupont
45333	Le champagne et la France	Penelope Light, Vanessa Martinez, Salome Dupont

Le champ 'Auteurs' contient plusieurs valeurs:

- La solution est de décomposer cette relation en trois relations :
- Publication(NumPubli, Titre),
  - Auteur(NumAuteur, Nom, Prenom)
  - EstEcrite(NumPubli, NumAuteur).

Ces trois relations sont la représentation de la réalité «une publication est écrite par des auteurs». Elle se modélise par deux entités 'Publication' et 'Auteur' reliées par l'association 'EstEcrite',

# Normalisation

## Deuxième forme normale

### Deuxième forme normale

Pour passer en deuxième forme il faut déjà satisfaire la première forme. Cette dernière sera chargée de rechercher la redondance d'information dans une relation. Elle interdit les dépendances fonctionnelles possibles entre les champs qui composent la clé et les autres champs. On peut l'exprimer de la manière suivante :

- *La relation est en première forme normale.*
- *Tout champ qui n'appartient pas à la clé ne dépend pas d'une partie de la clé.*

Considérons l'exemple suivant:

Produit(Article, Fournisseur, Adresse, Prix)

Article	Fournisseur	Adresse	Prix
Marteau	SOGENO	Paris	5
Tournevis	ARTIFACT	Lille	10
Tournevis	SOGENO	Paris	23
Pince	LEMEL	Paris	34
Mètre	ARTIFACT	Lille	24

La clé est constituée des champs ‘Article’ et ‘Fournisseur’. Or, il y a une relation de dépendance entre le champ ‘Fournisseur’, qui est une partie de la clé, et le champ ‘Adresse’.

- La solution consiste à décomposer la relation en deux. La nouvelle relation créée a pour clé la partie de la clé dont dépendent les autres champs qui constituent ses champs: *il s'agit du champ ‘Fournisseur’*. Les autres champs dépendants constituent le reste de la relation. *Il s'agit ici du champ ‘Adresse’.*

Produit(Article, Fournisseur, Prix)

Article	Fournisseur	Prix
Marteau	SOGENO	5
Tournevis	ARTIFACT	10
Tournevis	SOGENO	23
Pince	LEMEL	34
Mètre	ARTIFACT	24

Fournisseur(Fournisseur, Adresse)

Fournisseur	Adresse
SOGENO	Paris
ARTIFACT	Lille
LEMEL	Paris

# Normalisation

## ○ Troisième forme normale

La troisième forme normale s'intéresse également la redondance d'information dans une relation. On recherche l'existence d'une dépendance entre deux champs qui ne font pas partie d'une clé. Elle n'accepte donc pas les dépendances fonctionnelles dites « transitives ». Elle s'exprime de la manière suivante :

- *La relation est en deuxième forme normale.*
- *Tout champ n'appartenant pas à une clé ne dépend pas d'un autre champ non clé.*

Considérons l'exemple suivant:

Baladeur(NumBal, Marque, Type, Couleur)

NumBal	Marque	Type	Couleur
12	Apple	Ipod	Blanc
43	Creative	Zen	Noir
23	Apple	Ipod	Noir
29	Creative	Zen	Gris
34	Sony	MZ-RH910	Rouge

Baladeur(NumBal, Type, Couleur)

NumBal	Type	Couleur
12	Ipod	Blanc
43	Zen	Noir
23	Ipod	Noir
29	Zen	Gris
34	MZ-RH910	Rouge

Il existe une dépendance fonctionnelle entre le champ 'Type' et le champ 'Marque'.

- La solution consiste à décomposer la relation en créant une nouvelle qui a pour clé le champ dont dépendent les autres champs constituant la dépendance transitive: *il s'agit dans ce cas du champ 'Type'*. Les autres champs de la nouvelle relation sont composés des champs qui en dépendent fonctionnellement : *ici, le champ 'Marque'*.

Baladeur\_type(Type, Marque)

Type	Marque
Ipod	Apple
MZ-RH910	Sony
Zen	Creative

# Normalisation

## ○ Forme normale de BOYCE-CODD

La forme normale de Boyce-Codd traite un cas un peu différent de ceux de la deuxième et troisième forme normale. Il s'agit du cas où une partie d'une clé ( primaire ou secondaire) dépend d'un champ. Une relation en troisième forme normale n'est pas toujours en forme « Boyce-Codd », mais l'inverse est toujours vrai.

➢ *Tout champ appartenant à une clé ne dépend pas d'un autre champ non clé.*

Considérons l'exemple suivant:

La clé de cette relation est constituée par les champs 'Marque' et 'Produit'. Nous avons une relation de dépendance entre 'Marque' et 'Couleur'. La relation est en troisième forme normale, mais elle n'est pas en forme de Boyce-Codd.

□ Plusieurs décompositions sont possibles :

- Dictaphone(Marque, Produit, Prix) et Marque\_coul(Couleur, Marque). **Mais cette décomposition génère des tuples non désirés au moment de la jointure.**
- Dictaphone(Produit, Prix, Couleur) et Marque\_coul(Couleur, Marque). **Elle permet de reconstituer l'information de départ par une jointure sur le champ 'Couleur'**

Dictaphone(Marque, Produit, Prix, Couleur)

Marque	Produit	Prix	Couleur
Philips	LD 1024	49	Blanc
Olympus	VN 1664	49	Noir
Philips	LD 5047 H	59	Blanc
ImaginR	VN 1664	69	Gris
Olympus	VN 234 PC	79	Rouge

# Le modèle relationnel

## Applications

---

### ● Enoncé 4:

On considère deux relations. La première *Garage* est de degré 7 et de cardinalité 3. La seconde *Film* est de degré 2 et de cardinalité 15.

- Quels sont le degré et la cardinalité du produit cartésien de *Garage* par *Film* ?
- Quels sont le degré et la cardinalité du produit cartésien de *Film* par *Garage* ?

# Le modèle relationnel

## Applications

### ● Enoncé 5:

Quelle est la clé de cette relation Film(Prix, Format, Type, Nombre), avec:

Prix	Format	Type	Nombre
12	4 :3	Couleur	3
4	16 :9	Noir/Blanc	1
12	16 :9	Couleur	1664
35	4 :3	Noir/Blanc	890
12	16 :9	Noir/Blanc	1

# Le modèle relationnel

## Applications

---

### ● Enoncé 6:

Trouver le prix et le type de tous les films de la relation ‘Film’ vue précédemment. Peut-on en déduire que ‘Prix’ & ‘Type’ est une clé candidate, c'est-à-dire que toute combinaison des valeurs du prix et du type d'un film permet d'identifier un film ?

# Le modèle relationnel

## Applications

---

### ● Enoncé 7:

Donnez le prix des films de la base films en « Noir/Blanc ». Quels sont le degré et la cardinalité de la relation obtenue ? Est-il possible de calculer ces valeurs à l'avance, comme on le fait pour un produit cartésien ?

# Le modèle relationnel

## Applications

### ● Enoncé 8:

On considère ces deux relations:

- Film(Prix, Format, Type, Nombre, Numero\_Film)

Prix	Format	Type	Nombre	Numero_Film
12	4/3	Couleur	3	2
4	16/9	Noir/Blanc	1	4
12	16/9	Couleur	1 664	50
35	4/3	Noir/Blanc	890	12
12	16/9	Noir/Blanc	1	12

- Catalogue(Numero\_Film, Titre)

Numero_Film	Titre
2	<i>Le train qui passe</i>
4	<i>A toi !</i>
50	<i>Les chats du Sénégal</i>
111	<i>Le temps expliqué</i>
12	<i>Les impôts faciles</i>

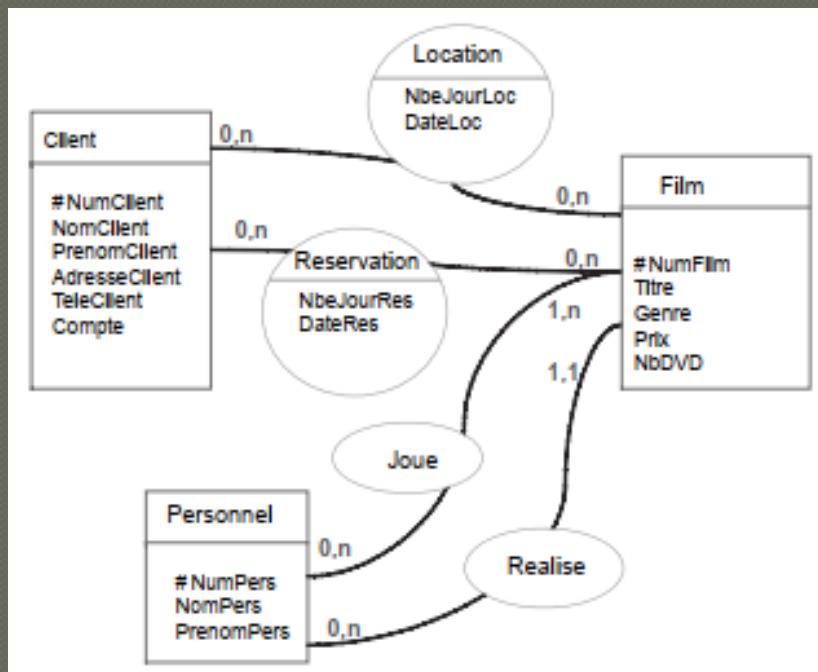
- Trouvez la liste des titres de films et leur format.
- Voyez-vous une incohérence dans le résultat ?
- Est-il possible de faire une jointure entre ces deux relations sur le champ ‘Prix’ de la relation ‘Film’ avec le champ ‘Numero\_film’ de la relation ‘Catalogue’ ? Si oui, que signifie le résultat ?

# Le modèle relationnel

## Applications

### Enoncé 9:

À partir du modèle entité-association modélisant une location de DVD, effectuez le passage au modèle relationnel.



# Le modèle relationnel

## Applications

### ● Enoncé 10:

Cette relation est-elle en première forme normale ?

- Personne(Nom, Adresse\_mail, Poste)

Nom	Adresse_mail	Poste
André Dupont	adup@philips.com, andre@dupond.fr	Directeur
Stanislas De la Motte	sdlm@versailles.mairie.fr	Employé, assernementé
Elisabeth Macroix	elsa@yago.to, mac@rien.fr	Assistante

# PLAN

---

- Généralités
- Le modèle conceptuel
- Le modèle relationnel
- Le langage SQL
- Etude et réalisation d'une base de données
- La sécurité des données

# Objectif

---

- Dans ce chapitre, nous aborderons les concepts et approches du langages SQL.
- Nous présenterons également les quelques opérations regroupées en 3 catégories de familles :
  - L'interrogation et la recherche dans les tables;
  - La gestion de tables et de vues;
  - La manipulation de données.

# Le langage SQL

## concepts

- Le langage SQL est un langage dite assertionnel permettant de manipuler des bases de données relationnelles. Il est issu des résultats du groupe de travail *System-R* qui travaillait sur la réalisation pratique des concepts de l'approche relationnelle chez IBM.
- C'est, en effet, une évolution du langage SEQUEL, qui est lui-même dérivé du langage de recherche SQUARE.
- Aujourd'hui, le langage SQL est normalisé [ISO89, ISO92] et constitue le standard d'accès aux bases de données relationnelles.
  - Le SQL1(ISO89) qui correspond au norme de base, a été accepté en 1989. Il permet l'expression des requêtes composées d'opérations de l'algèbre relationnelle et d'agrégats. En plus des fonctionnalités de définition, de recherche et de mise à jour, il comporte aussi des fonctions de contrôle qui n'appartiennent pas vraiment au modèle relationnel, mais qui sont nécessaires pour programmer des applications transactionnelles.
  - Le SQL2 [ISO92] a été adoptée en 1992, et est sous-divisé en trois niveaux, respectivement entrée, intermédiaire et complet.
    - Le niveau entrée peut être perçu comme une amélioration de SQL1, alors que les niveaux intermédiaire et complet permettent de supporter totalement le modèle relationnel avec des domaines variés, tels date et temps.
    - Le SQL3 quant à lui a été normalisé en 1999, et est essentiellement constitué d'un ensemble de propositions nouvelles traitant plus particulièrement des fonctionnalités objets et déductives.

# Le langage SQL

## concepts

---

De manière générale, SQL utilise des critères de recherche (encore appelés qualifications) construits à partir de la logique **des prédictats du premier ordre** qui comportent quatre opérations de base à savoir:

- **L'insertion** (mot clé INSERT) permet d'ajouter des tuples dans une relation;
- **La recherche** (mot clé SELECT) permet de retrouver des tuples ou parties de tuples vérifiant la qualification citée en arguments;
- **La suppression** (mot clé DELETE) permet de supprimer d'une relation les tuples vérifiant la qualification citée en argument ;
- **La modification** (mot clé UPDATE) permet de mettre à jour les tuples vérifiant la qualification citée en argument à l'aide de nouvelles valeurs d'attributs ou de résultats d'opérations arithmétiques appliquées aux anciennes valeurs.

# Opérations relationnelles avec SQL

- Nous allons créer et manipuler la base de données ‘casse’ ci-dessous:

Voiture			
# NumVoit	Marque	Type	Couleur
1	Peugeot	404	Rouge
2	Citroen	SM	Noire
3	Opel	GT	Blanche
4	Peugeot	403	Blanche
5	Renault	Alpine A310	Rose
6	Renault	Floride	Bleue



Client				
# IDClient	Nom	Age	Ville	Sexe
1	Nestor	96	Paris	M
2	Irma	20	Lille	F
3	Henri	45	Paris	M
4	Josette	34	Lyon	F
5	Jacques	50	Bordeaux	M

Modèle  
entité-association

NumVoit	Marque	Type	Type
1	Peugeot	404	Rouge
2	Citroen	SM	Noire
3	Opel	GT	Blanche
4	Peugeot	403	Blanche
5	Renault	Alpine A310	Rose
6	Renault	Floride	Bleue

voiture

DateVente	Prix	NumVoit	NumAch
1985-12-03	10 000	1	1
1996-03-30	70 000	2	4
1998-06-14	30 000	4	1
2000-04-02	45 000	5	2

vente

NumAch	Nom	Age	Ville	Sexe
1	Nestor	96	Paris	M
2	Irma	20	Lille	F
3	Henri	45	Paris	M
4	Josette	34	Lyon	F
5	Jacques	50	Bordeaux	M

client

Les relations  
ou tables  
avec les  
enregistrements

- Outils de travail:
  - Wampserver
  - HeidiSQL (gestionnaire de session)

# Créer une base de données MySQL ?

---

Pour créer une base de données et des utilisateur MySQL, il est possible de le faire soit par ligne de commande ou via une interface graphique.

- En ligne de commande,

- il va d'abord falloir se connecter en tant qu'utilisateur root:
  - `mysql -u root -p` # il va falloir entrer le mot de passe et puis appuyer sur entrer.
  - Tapez `\q` pour quitter le programme mysql.
- Il faut aussi créer un utilisateur de base de données en tapant la commande suivante:
  - `CREATE USER 'username'@'localhost' IDENTIFIED BY 'password';`
- Pour lui accorder tous les privilèges, il faut exécuter la commande suivante:
  - `GRANT ALL PRIVILEGES ON *.* TO 'username'@'localhost';`
  - `FLUSH PRIVILEGES ;` # pour prendre en compte les changements de droit
- Pour afficher les privilèges d'un utilisateur on fait:
  - `SHOW GRANTS FOR 'username'@'localhost';`

# Créer une base de données MySQL ?

---

- Il est possible de révoquer tous les privilèges d'un utilisateur non-root en faisant:
  - `REVOKE ALL PRIVILEGES ON *.* FROM 'username'@'localhost';`
- Ou certains privilèges:
  - `REVOKE TYPE_OF_PERMISSION ON database.table FROM 'username'@'localhost';`
- Ou supprimer entièrement un utilisateur:
  - `DROP USER 'username'@'localhost';`

Les principaux **TYPE\_OF\_PERMISSION** sont:

- CREATE – Permet aux utilisateurs de créer des bases de données/tableaux
- SELECT – Permet aux utilisateurs de récupérer des données
- INSERT – Permet aux utilisateurs d'ajouter de nouvelles entrées dans les tableaux
- UPDATE – Permet aux utilisateurs de modifier les entrées existantes dans les tableaux
- DELETE – Permet aux utilisateurs de supprimer les entrées de la tableau
- DROP – Permet aux utilisateurs de supprimer des bases de données/tableaux entiers

# Créer une base de données MySQL ?

---

- On pourra alors se connecter à MySQL en tant qu'utilisateur (que nous venez de créer), en tapant la commande suivante:  
➤ `mysql -u username -p`
- Pour créer une base de données, on doit taper la commande suivante:  
➤ `CREATE DATABASE dbname;`
- Oubien:  
➤ `CREATE DATABASE IF NOT EXISTS dbname ;` # pour éviter des erreurs  
# ici `dbname` sera remplacée par notre base de données (`casse`)
- Pour pouvoir travailler avec la nouvelle base de données, on doit taper:  
➤ `USE dbname;`
- Pour connaître les bases disponibles, on utilise la commande:  
➤ `SHOW DATABASES;`

Nous serons maintenant capable de créer des `tables` de la base et d'y insérer des `données`.

# Gestion de tables et de vues

## Les TYPES

Notre base de données ainsi créer est vide. Il faut donc y ajouter des tables (ou objets).

- Chaque table est composée de plusieurs champs et chacun de ces champs doit avoir un type pour spécifier la nature de la donnée qui sera stockée dans ce champ (chiffres, texte, etc.). Parmi les types, nous avons principalement:

- **INTEGER** (contientra des chiffres numériques sous forme de nombre entier « 32bits »);
- **SMALLINT** (nombre entier «16 bits» );
- **FLOAT** (contientra des chiffres décimaux);
- **BOOLEAN** (Il ne peut stocker que les valeurs true (vrai) ou false (faux)) ;
- **CHAR(n)** (contientra une chaîne de caractères de longueur « n »);
- **VARCHAR(n)** (Chaîne de caractères de longueur maximale « n »);
- **DATE** (pour les dates);
- **TIME[(n)]** (pour les heures, n (optionnel) est le nombre de décimales représentant la fraction de secondes);
- **BLOB** (Binary Large Object) pour stocker tout type binaire (photo, fichier texte...);
- ...

# Gestion de tables et de vues

## CONTRAINTE S D'INTÉGRITÉ

Nous avons eu à voir la notion de « domaine », dans la partie conceptualisation, qui permet de décrire l'ensemble des valeurs que peut prendre un attribut. Le langage SQL permettra de définir ces conditions plus finement lors de la création de la table. Une première approche du domaine étant établie lors du choix du type de la colonne.

- On distingue différents types de contraintes sur les colonnes:

- les propriétés générales comme l'unicité:
  - La valeur de la colonne doit être renseignée absolument (**NOT NULL**). Si une telle propriété n'est pas renseignée sur une colonne, cela veut dire qu'elle peut ou pas contenir de données (**NULL**). Elle peut alors contenir « **0** » pour une colonne de type « **entier** » ou un **espace** pour une colonne de type « **caractère** ».
  - La valeur doit être unique comparée à toutes les valeurs de la colonne de la table (**UNIQUE**);

Lorsque les deux conditions précédentes sont réunies, la colonne peut servir à identifier un enregistrement et constitue donc une « **clé candidate** ». La clé sera désignée en SQL par le mot clé **PRIMARY KEY**.

# Gestion de tables et de vues

## CONTRAINTES D'INTÉGRITÉ

- les restrictions d'appartenance à un ensemble ;

Il s'agit de décrire le domaine dans lequel la colonne pourra prendre ses valeurs. Un ensemble peut être décrit :

- En donnant la liste de tous ses éléments constitutifs (**IN**). L'ensemble des jours de la semaine ne peut être exprimé que de cette manière : « **lundi** », « **mardi** », etc.
  - ✓ On vérifie que la colonne ‘couleur’ ne peut prendre que des valeurs « normalisées » : ‘Rouge’, ‘Vert’ ou ‘Bleu’:
    - **CHECK (Couleur in ('Rouge','Vert','Bleu'));**
- Par une expression (**>**, **<** , **BETWEEN...**). Par exemple, le prix doit être supérieur à 1 000.
  - ✓ On vérifie que l’âge est compris entre 1 et 80:
    - **CHECK (Age BETWEEN 1 AND 80).**
- Par une référence aux valeurs d’une colonne d’une autre table (**REFERENCES**).
  - ✓ On vérifie que les valeurs identifiantes des personnes ‘NumAch’ et des voitures ‘NumVoit’ de la table ‘vente’ existent bien dans les tables de référence ‘personne’ et ‘voiture’:
    - **NumAch INT NOT NULL REFERENCES personne(NumAch),**
    - **NumVoit INT NOT NULL REFERENCES voiture(NumVoit),**
    - **PRIMARY KEY (NumAch, NumVoit)**

# Gestion de tables et de vues

## CONTRAINTES D'INTÉGRITÉ

---

- les dépendances entre plusieurs colonnes (contrainte de table):
  - Lorsque l'on désire exprimer des contraintes plus élaborées impliquant plusieurs colonnes, on peut définir une contrainte de table en utilisant le mot clé CONSTRAINT.
    - ✓ On vérifie que la colonne 'Age' et la colonne 'Ville' doivent être renseignées ou vides en même temps:
      - CONSTRAINT la\_contrainte CHECK ( (Age IS NOT NULL AND Ville IS NOT NULL) OR (Age IS NULL AND Ville IS NULL) )

# Gestion de tables et de vues

## CREATION DE TABLES

Lors de la création de table, il faut donc définir le **type de données**, la **clé**, les **index éventuels** et les **contraintes de validation éventuelles** garantissant la bonne qualité des informations entrées.

- **Syntaxe :**
  - **CREATE TABLE <Nom de la table> ( liste des colonnes avec leur type séparé par , ) ;**

```
CREATE TABLE voiture (
    NumVoit INT PRIMARY KEY,
    Marque CHAR(40) NOT NULL,
    Type CHAR(30),
    Couleur CHAR(20),
    CHECK (Couleur in ('Rouge','Vert','Bleu'))
);
```

```
CREATE TABLE personne (
    NumAch INT PRIMARY KEY,
    Nom CHAR(40) NOT NULL,
    Ville CHAR(40),
    Age INT NOT NULL,
    CHECK (Age BETWEEN 1 AND 80)
);
```

```
CREATE TABLE vente (
    DateVente DATE,
    Prix INT,
    NumAch INT NOT NULL REFERENCES personne(NumAch),
    NumVoit INT NOT NULL REFERENCES voiture(NumVoit),
    PRIMARY KEY (NumAch, NumVoit)
);
```

- **Le nom de la table ou d'une colonne** ne doit pas dépasser 128 caractères. Il commence par une lettre, contient des chiffres, des lettres et le caractère « \_ ».
- **SHOW tables** (permet d'afficher toutes les tables présentes dans notre base).
- **SHOW COLUMNS FROM le\_nom\_de\_ma\_table** (permet d'afficher le schéma de chaque table)

# Gestion de tables et de vues

## CREATION DE VUES

- Une « vue » est le résultat d'une requête que l'on peut manipuler de la même façon qu'une table. C'est une sorte de **table dynamique** dont le contenu peut être exprimé à chaque utilisation.
  - Elle est utilisée juste par commodité car, soit:
    - parce qu'il n'est pas nécessaire que certains utilisateurs voient le modèle complet qui peut parfois être complexe;
    - pour restreindre l'accès à certains données pour des raisons de sécurité/confidentialité).
- `CREATE VIEW personne_bis (NumAch, Nom, Age) AS SELECT  
NumAch, Nom, Age FROM personne ;`

# Gestion des données

## INSERTION (*INSERT INTO*)

---

On dispose classiquement de trois opérations pour gérer les données d'une table: l'insertion, la suppression et la mise à jour.

◎ Pour insérer des données dans une table, on utilise la formule générale suivante :

- `INSERT INTO <nom_de_la_table> [ liste_des_colonnes ] VALUES <liste_des_valeurs>`
- Par exemple, pour l'insertion d'un enregistrement dans la table 'voiture', on peut avoir :
  - `INSERT INTO voiture (NumVoit, Marque, Type, Couleur) VALUES (1,'Peugeot',404,'Rouge');`

# Gestion des données

## SUPPRESSION, MODIFICATION

- L'opération de suppression permet de supprimer un ensemble d'enregistrements (lignes) que l'on identifiera avec la clé WHERE:

- `DELETE FROM voiture WHERE Couleur='Rouge' ;`
  - si l'on ne spécifie aucune condition, tous les enregistrements sont supprimés.
  - `DELETE FROM personne ;`

- Pour effectuer la mise à jour, il faut préciser les colonnes concernées, les nouvelles valeurs et les enregistrement pour lesquels on modifiera ces valeurs. on identifiera les enregistrements concernés avec la clé WHERE:

- `UPDATE personne SET Ville='Paris-Centre' WHERE Ville='Paris' ;`

# Interrogation dans les tables

## Projection

- L'opération de projection consiste à sélectionner la (les) colonne(s) dans une table. On spécifie la liste des colonnes après l'instruction **SELECT** en les séparant par des **virgules**. Si l'on désire afficher toutes les colonnes, on les désigne par le caractère «**\***».

- SELECT Nom, Ville FROM personne ;**

Nom	Ville
Nestor	Paris
Irma	Lille
Henri	Paris
Josette	Lyon
Jacques	Bordeaux

- SELECT \* FROM personne ;**

NumAch	Nom	Age	Ville	Sexe
1	Nestor	96	Paris	M
2	Irma	20	Lille	F
3	Henri	45	Paris	M
4	Josette	34	Lyon	F
5	Jacques	50	Bordeaux	M

- Les colonnes peuvent être renommées par le mot clé **AS**:
- SELECT Ville AS City FROM personne ;**

City
Paris
Lille
Paris
Lyon
Bordeaux

# Interrogation dans les tables

## Projection

- Il est possible d'afficher les valeurs distinctes d'une colonne Afin d'éliminer les doublons éventuels. on fait précéder le nom de la colonne par le mot clé DISTINCT:

- SELECT DISTINCT Marque FROM voiture ;

Marque
Peugeot
Citroen
Opel
Renault

- Il est aussi possible d' utiliser des expressions pour créer une colonne (vue que l' on ne stocke pas dans une table ce qui peut être calculé). Les opérateurs suivants peuvent nous permettre d'arriver à ces résultats.

+	Addition
-	Soustraction
*	Multiplication
/	Division
%	Modulo

- SELECT Prix, DateVente, (Prix / 6.5596) AS Prix\_Euros FROM vente ;

Prix	DateVente	Prix_Euros
10 000	1985-12-03	1 524.483 200
70 000	1996-03-30	10 671.382 401
30 000	1998-06-14	4 573.449 601
45 000	2000-04-02	6 860.174 401

# Interrogation dans les tables

## Projection

- SQL dispose de nombreuses autres fonctions intégrées, parfois dépendantes du SGBD utilisé, pour permettre par exemple de traiter des colonnes de types caractères, date...

- `SELECT UPPER(Nom) AS NomMajuscule FROM personne ;`
- `SELECT MONTH(DateVente) AS Mois FROM vente ;`

NomMajuscule
NESTOR
IRMA
HENRI
JOSETTE
JACQUES

Mois
12
3
6
4

- Les colonnes peuvent être constituées de résultats de fonctions statistiques intégrées à SQL. Voici une liste (non exhaustive) des opérateurs statistiques de SQL:

COUNT	Comptage du nombre d'éléments (lignes) de la table
MAX	Maximum des éléments d'une colonne
MIN	Minimum des éléments d'une colonne
AVG	Moyenne des éléments d'une colonne
SUM	Somme des éléments d'une colonne

- `SELECT AVG(Prix) AS Prix_Moyen FROM vente ;`

Prix_Moyen
38 750.000 0

- `SELECT COUNT(*) AS Nombre_Personne FROM personne ;`

Nombre_Personne
5

# Interrogation dans les tables

## SÉLECTION OU RESTRICTION (WHERE)

- L'opération de sélection (ou restriction) consiste à indiquer un ou plusieurs critères (sur le contenu des colonnes) pour choisir les lignes à inclure dans la table « résultat ». Cette critère de sélection est indiqué à la suite du mot clé WHERE et est constitué d'**expressions de conditions composées** (des opérateurs de comparaison et des connecteurs logiques). Ci-dessous:

- la liste des opérateurs de comparaison:

=	Égal
<>	Different
<	Inférieur
>	Supérieur
<=	Inférieur ou égal
>=	Supérieur ou égal

- SELECT \* FROM vente WHERE Prix > 50 000 ;

DateVente	Prix	NumVoit	NumAch
1996-03-30	70 000	2	4

BETWEEN <valeur> AND <valeur>	Appartient à un intervalle
IN <liste de valeurs>	Appartient à un ensemble de valeurs
IS NULL	Teste si la colonne n'est pas renseignée
LIKE	Compare des chaînes de caractères

- SELECT \* FROM voiture WHERE Couleur IN ("Blanc", "Rouge") ;

NumVoit	Marque	Type	Couleur
1	Peugeot	404	Rouge

# Interrogation dans les tables

## SÉLECTION OU RESTRICTION (*WHERE*)

- la liste des connecteurs logiques:

AND	Et : les deux conditions sont vraies simultanément
OR	Ou : l'une des deux conditions est vraie
NOT	Inversion de la condition

- SELECT \* FROM voiture WHERE Couleur="Blanche" OR Marque="Peugeot" ;

NumVoit	Marque	Type	Couleur
1	Peugeot	404	Rouge
3	Opel	GT	Blanche
4	Peugeot	403	Blanche

- SELECT \* FROM personne WHERE NOT (Ville='Paris') ;

NumAch	Nom	Age	Ville	Sexe
2	Irma	20	Lille	F
4	Josette	34	Lyon	F
5	Jacques	50	Bordeaux	M

# Interrogation dans les tables

## AGRÉGATS OU GROUPAGE (*GROUP BY*)

- Les opérations d'« agrégation » ou de « groupage » regroupent les lignes d'une table par valeurs contenues dans une colonne. Pour réaliser cette opération, on utilise le mot clé **GROUP BY** suivi du nom de la colonne sur laquelle s'effectue l'agrégat.

- `SELECT Marque FROM voiture GROUP BY Marque ;`

Marque
Citroen
Opel
Peugeot
Renault

- On applique généralement des opérations de type statistique sur les « sous-tables » ainsi créées.

- `SELECT Marque, COUNT(*) AS Compte FROM voiture GROUP BY Marque ;`

Marque	Compte
Citroen	1
Opel	1
Peugeot	2
Renault	2

# Interrogation dans les tables

## REQUÊTES SUR PLUSIEURS TABLES

- Lorsque l'on utilise plusieurs tables dans une requête SQL, il peut exister une ambiguïté dans les expressions sur les noms de colonnes. Comme deux tables peuvent avoir des noms de colonne identique, il est dans ce cas permis de lever cette ambiguïté en préfixant le nom de la colonne par le nom de la table en question.

- `SELECT voiture.Marque, voiture.Couleur FROM voiture ;`

Marque	Couleur
Peugeot	Rouge
Citroen	Noire
Opel	Blanche
Peugeot	Blanche
Renault	Rose
Renault	Bleue

- Il est commode de désigner la table par un alias pour peut être réduit son nom ou autres. Pour cela, il suffit simplement de mettre à la suite du nom de la table le mot clé **AS**.

- `SELECT Vo.Marque, Vo.Couleur FROM voiture AS Vo ;`

Marque	Couleur
Peugeot	Rouge
Citroen	Noire
Opel	Blanche
Peugeot	Blanche
Renault	Rose
Renault	Bleue

# Interrogation dans les tables

## REQUÊTES SUR PLUSIEURS TABLES

- **Produit cartésien:**

- `SELECT * FROM personne, voiture ;`

- **Jointure interne (INNER JOIN):**

- `SELECT voiture.Marque, voiture.Couleur, vente.Prix FROM voiture, vente WHERE voiture.NumVoit=vente.NumVoit ;`
  - Oubien par un opérateur de jointure spécifique **JOIN**:
  - `SELECT voiture.Marque, voiture.Couleur, vente.Prix FROM vente INNER JOIN voiture ON voiture.NumVoit=vente.NumVoit ;`

Marque	Couleur	Prix
Peugeot	Rouge	10 000
Citroen	Noire	70 000
Peugeot	Blanche	30 000
Renault	Rose	45 000

- `SELECT vo.Marque, vo.Couleur, ve.Prix, pe.Nom, pe.Age FROM voiture AS vo JOIN vente AS ve JOIN personne AS pe ON (vo.NumVoit=ve.NumVoit) AND (pe.NumAch=ve.NumAch);`

Marque	Couleur	Prix	Nom	Age
Peugeot	Rouge	10 000	Nestor	96
Citroen	Noire	70 000	Josette	34
Peugeot	Blanche	30 000	Nestor	96
Renault	Rose	45 000	Irma	20

# Interrogation dans les tables

## REQUÊTES SUR PLUSIEURS TABLES

- Jointure externe (OUTER JOIN): L'opération de jointure interne ne permet pas de répondre à des questions du type : « Quelles sont les voitures qui n'ont pas été vendues ? ». Il s'agit des ‘voiture’ qui n'ont pas de correspondance dans la table « vente ».
  - Cette opération n'est pas symétrique: soit on inclut toutes les lignes d'une table, soit toutes celles de l'autre. On précise cela à l'aide des mots clés LEFT et RIGHT ou en inversant simplement l'ordre des tables dans l'expression de l'instruction de jointure.
  - `SELECT voiture.NumVoit, vente.NumVoit, voiture.Marque, voiture.Couleur, vente.Prix FROM voiture LEFT OUTER JOIN vente ON voiture.NumVoit=vente.NumVoit ;`

NumVoit	NumVoit	Marque	Couleur	Prix
1	1	Peugeot	Rouge	10 000
2	2	Citroen	Noire	70 000
3	NULL	Opel	Blanche	NULL
4	4	Peugeot	Blanche	30 000
5	5	Renault	Rose	45 000
6	NULL	Renault	Bleue	NULL

- `SELECT voiture.NumVoit, vente.NumVoit, voiture.Marque, voiture.Couleur, vente.Prix FROM vente LEFT OUTER JOIN voiture ON voiture.NumVoit=vente.NumVoit ;`

NumVoit	NumVoit	Marque	Couleur	Prix
1	1	Peugeot	Rouge	10 000
2	2	Citroen	Noire	70 000
4	4	Peugeot	Blanche	30 000
5	5	Renault	Rose	45 000

# Interrogation dans les tables

## TRI DU RÉSULTAT D'UNE REQUÊTE

- On utilise le mot clé **ORDER BY** pour spécifier la (les) colonne(s) sur laquelle (lesquelles) on souhaite trier le résultat.

- `SELECT Marque, Type FROM voiture ORDER BY Marque ;`

Marque	Type
Citroen	SM
Opel	GT
Peugeot	404
Peugeot	403
Renault	Alpine A310
Renault	Floride

- Il est possible de préciser l'ordre de tri par les mots clés **ASC** (croissant par défaut) ou **DESC** (décroissant).

- `SELECT Prix, DateVente FROM vente ORDER BY Prix DESC ;`

Prix	DateVente
70 000	1996-03-30
45 000	2000-04-02
30 000	1998-06-14
10 000	1985-12-03

- On peut indiquer plusieurs critères de tri, qui sont lus et traités de gauche à droite (ici, on trie d'abord par villes puis par âges).

- `SELECT Nom, Age, Ville FROM personne ORDER BY Ville, Age ;`

Nom	Age	Ville
Jacques	50	Bordeaux
Irma	20	Lille
Josette	34	Lyon
Henri	45	Paris
Nestor	96	Paris

# PLAN

---

- Généralités
- Le modèle conceptuel
- Le modèle relationnel
- Le langage SQL
- Etude et réalisation d'une base de données
- La sécurité des données

# Etude et réalisation d'une base de données

## Objectif

---

Dans cette session, nous allons proposer une étude complète de la mise en place d'une base de données en partant de l'étude du cahier des charges jusqu'à l'interrogation de la base à l'aide du langage SQL. Nous allons faire:

- l'analyse du monde réel et la création du modèle entité-association ;
- le passage au modèle relationnel ;
- la création des tables en SQL avec intégration des contraintes de contenu ;

# Etude et réalisation d'une base de données

## Présentation de l'activité à modéliser

Nous cherchons à modéliser la gestion d'une entreprise de production et de livraison de repas à domicile: la société "Sen BOUFF" qui est une société qui propose des plats chauds de types sénégalaise.

Le client doit choisir, dans le catalogue proposé, le plat qu'il souhaite commander. Les plats proposés sont:

- pour le repas du midi:
  - Thiébou Dieun, Thiébou Yapp, Thiébou Guanar, Yassa Dieun, Yassa Yapp, Yassa Guanar, Mafé, Mborokhé, Soup Kandia, Domoda et Thiou Boulette.
- Pour le repas du soir:
  - Spaghetti Yapp, Spaghetti Ganar, Frite Guanar, Frite Dieun, Dibi Yapp, Dibi Guanar, Ragout Yapp, Ragout Guanar, Ndambé Yapp, Ndambé Guanar, Vermicelle Yapp, Vermicelle Ganar, Dakhine, Mbakhale Saloum, Thiéré Siim, Thiéré Baassé, Couscous Yapp, Couscous Guanar.

Chaque plats est caractérisé par son nom, les ingrédients additionnels qui la composent et son prix de base. Il existe quatre type de plats: normale, large, normale amélioré (plus chère que le normale simple) ou large amélioré (plus chère que le large simple)

# Etude et réalisation d'une base de données

## Présentation de l'activité à modéliser

---

### ● Le mode de distribution

Les plats sont livrés par des livreurs qui circulent en voiture ou à moto.

- On doit pouvoir effectuer le suivi des activités.

### ● Les modalités de vente

Le mode de vente est du type prépayé: Avant toute commande, les clients doivent approvisionner leur compte. On vérifie le solde du compte avant de préparer et de livrer la commande.

- Il existe deux systèmes de bonification :

- Un plat gratuit est offert au bout de 10 plats achetés.
- Tout plat livré en plus d'un heure est gratuit.

# Etude et réalisation d'une base de données

## Présentation de l'activité à modéliser

---

### ● Les objectifs du système

Le but de la mise en place de cette base de données est principalement de gérer l'activité quotidienne de vente et de livraison des plats de type sénégalaise:

- vérification du solde du compte et facturation aux clients ;
- suivi du chiffre d'affaires ;
- refus d'honorer les commandes pour lesquelles le solde du compte client est insuffisant ;
- non-facturation des plats gratuits (retard ou fidélité).

On veut également effectuer des statistiques diverses sur l'activité:

- identification du meilleur client ;
- identification du plus mauvais livreur (nombre de retards dans la livraison) et du véhicule utilisé ;
- identification du plat le plus ou le moins demandé ;

# Etude et réalisation d'une base de données

## Le modèle entité-association

---

Dans cette étape, nous allons essayer de déterminer les éléments qui permettront de constituer les futures tables de la base de données. Pour cela, nous procèderons en deux temps:

- la construction du graphe des entités reliées par des associations;
- la qualification des associations par leurs cardinalités.

### ○ Identification des entités et des associations

Nous procéderons d'abord au repérage des différentes entités avec des phrases simples qui identifient l'activité par rapport à l'énoncé en cherchant, en même temps, à identifier les objets concrets du monde réel qui semblent impliqués dans le système. Puis, pour chaque entité, il nous faut déterminer une clé parmi les attributs. Enfin, il faut caractériser les liens entre les entités par des associations.

# Etude et réalisation d'une base de données

## Le modèle entité-association

### ➤ Entités et attributs

Nous commencerons à identifier les mots clés dans la description générale: ‘plat’, ‘ingrédient’, ‘client’, ‘livreur’, ‘véhicule’. Ces descripteurs représentent les objets familiers du monde réel. Nous avons d’autres mots clés comme ‘taille’, ‘prix’, ‘compte’, ‘retard’, ‘nom’ ... qui sont clairement plus des qualifiants que des objets.

C'est la notion de **commande** (objet abstrait) qui relie tous ces objets afin de constituer la représentation de l'activité.

Après ce constat, nous sommes en mesure de proposer quelques phrases permettant d'effectuer une première synthèse de l'activité.

- « Un plat est constitué de plusieurs ingrédients. »
- « Un client passe une commande. »
- « Une commande est livrée par un et un seul livreur. »
- « Une commande est livrée par un et un seul véhicule. »

# Etude et réalisation d'une base de données

## Le modèle entité-association

On peut en déduire les entités suivantes: commande, client, plat, livreur, véhicule, ingrédient.

Après un dialogue avec les différents acteurs de l'entreprise, on obtient les renseignements suivants :

- Un client est caractérisé par son nom et son adresse.
- Un livreur est caractérisé par son nom et son numéro de téléphone.
- Un véhicule est caractérisé par sa marque, son type et son numéro d'immatriculation.

Ces informations complémentaires permettent de déterminer quelques attributs des entités ainsi constituées. Les autres attributs se trouvent dans l'énoncé: par exemple « un plat est caractérisé par son nom et par son prix » ou « un ingrédient est caractérisé par son nom ».

L'affectation d'un attribut à une entité n'est pas toujours évidente. On va créer un attribut 'compte' qui contiendra les informations de solvabilité du client puisque l'on fonctionne en mode prépayé. Il faut se poser la question : « *est-ce une propriété caractéristique du client ?* » La réponse dans notre cas est aisée et l'attribut sera affecté à l'entité 'client'.

# Etude et réalisation d'une base de données

## Le modèle entité-association

---

### ➤ Entités et attributs

Comment prendre en compte le **type** de plat? Le type sert uniquement à pondérer le prix de base du plat: il n'est donc pas associé au **plat**, dont le prix est fixe; il ne constitue pas non plus une caractéristique d'un **client**. On l'associe logiquement à l'entité '**commande**'.

Par un raisonnement semblable, on détermine que le **retard** est associé à la commande alors que la **fidélisation** est associée au client.

Si l'on récapitule, on obtient les entités munies de leurs attributs suivants:

- commande** (DateCom, Taille, Retard) ;
- client** (NomClient, Adresse, Compte, PointsRaPlat) ;
- plat** ((NomPlat, Prix) ;
- livreur** (NomLivreur, Téléphone) ;
- véhicule** (NumImmat, Marque, Type) ;
- ingrédient** (NomIngrédient).

# Etude et réalisation d'une base de données

## Le modèle entité-association

### ➤ Choix de la clé

Les attributs étant identifiés, on doit maintenant choisir une clé pour chaque entité.

Si l'on prend l'entité ‘**client**’, il est clair qu'aucun attribut ne peut convenir pour constituer une clé. De même, l'association de plusieurs attributs ne permet pas de créer une clé. On ajoute alors

classiquement un attribut identifiant qui sert de clé.

En utilisant ces mêmes arguments, on est amené à ajouter des attributs numériques comme clés pour les entités ‘**livreur**’ et ‘**commande**’.

Pour l'entité ‘**plat**’, on peut supposer que le nom du plat est significatif et qu'il peut donc servir de clé. En pratique, même si le nom est unique, on pourrait décider de ne pas l'utiliser car le contenu est un peu long.

L'entité ‘**véhicule**’ dispose d'un champ identifiant: son numéro d'immatriculation qui est par définition unique.

Enfin, en ce qui concerne les **ingrédients**, il peut être moins évident que le nom seul de l'ingrédient suffise à l'identifier. Il est préférable de lui ajouter un numéro.

On obtient les entités suivantes munies de leurs attributs :

- commande** (NumCommande, DateCom, Taille, Retard) ;
- client** (NumClient, NomClient, Adresse, Compte, PointsRaPlat) ;
- plat** (NomPizza, Prix) ;
- livreur** (CodeLivreur, NomLivreur, Téléphone) ;
- véhicule** (NumImmat, Marque, Type) ;
- ingrédient** (NumIngre, NomIngrédient).

# Etude et réalisation d'une base de données

## Le modèle entité-association

### ➤ Associations et attributs

On peut, à partir des phrases qui ont permis de repérer les entités, en déduire les associations suivantes :

- ‘Livre’ entre ‘Livreur’ et ‘Commande’ ;
- ‘Transporte’ entre ‘Véhicule’ et ‘Commande’ ;
- ‘Passe’ entre ‘Client’ et ‘Commande’ ;
- ‘Constitue’ entre ‘Plat’ et ‘Commande’ ;
- ‘Compose’ entre ‘Plat’ et ‘Ingrédient’.

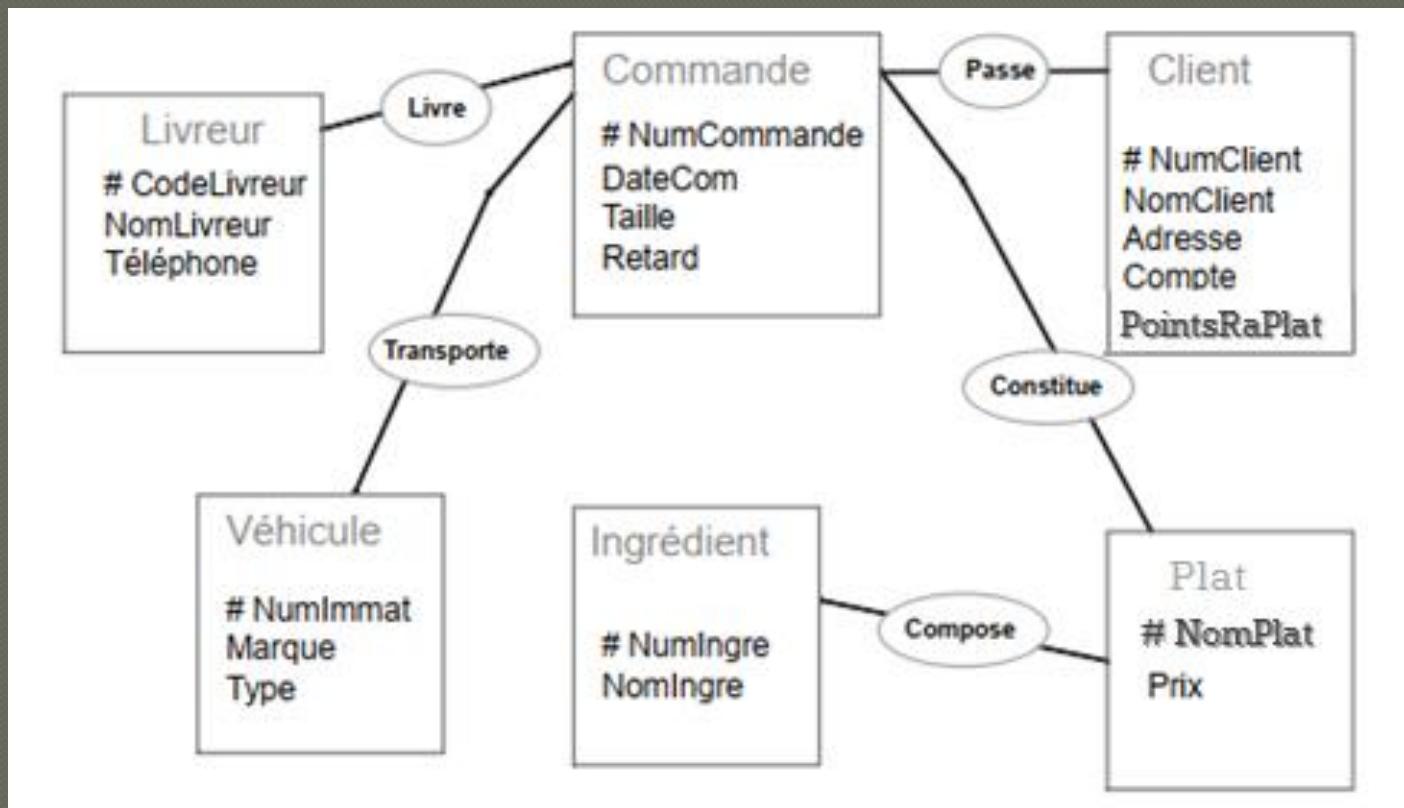
Il reste à déterminer les attributs éventuels des associations. Dans notre cas, le choix d'utiliser une entité ‘commande’ fait que l'on regroupe naturellement dans cette entité les attributs qui auraient pu se retrouver sur les associations.

Éventuellement, une date de commande pourrait être un attribut de l'association ‘Passe’ si elle était différente de la date de la commande.

# Etude et réalisation d'une base de données

## Le modèle entité-association

### ➤ Associations et attributs



# Etude et réalisation d'une base de données

## Le modèle entité-association

### ○ Détermination des cardinalités

On se pose ensuite des questions sur la nature des liens entre les entités capables de déterminer les cardinalités qui seront utilisées pour le passage au modèle relationnel.

On doit trouver deux cardinalités, maximales et minimales, par entité associée, ce qui correspond à deux questions doubles par association.

#### ➤ Association 'Compose'

- **Ingédient** : un ingrédient peut-il ne jamais être utilisé dans la composition d'un plat et peut-il être utilisé plusieurs fois ? On suppose que si un ingrédient est au catalogue, c'est qu'il est utilisé au moins une fois (1) et qu'il peut entrer dans la composition de plusieurs (n) plats. Les cardinalités associées sont de type '1-n'.
- **Plat** : un plat peut-il n'avoir aucun ingrédient et peut-elle en avoir plusieurs ? On suppose qu'un plat est constitué d'au moins un (1) ingrédient et qu'il peut en avoir plusieurs (n). Les cardinalités associées sont de type '1-n'.

#### ➤ Association 'Passe'

- **Client** : un client peut-il n'avoir jamais passé de commandes et peut-il en avoir passé plusieurs ? Il peut y avoir une période pendant laquelle le client a approvisionné son compte mais n'a pas encore passé (0) de commande et il est évidemment encouragé à en passer plusieurs (n). Les cardinalités associées sont de type '0-n'.
- **Commande** : une commande peut-elle avoir été passée par aucun client ou par plusieurs ? Une commande donnée est passée par un (1) et un (1) seul client. Les cardinalités associées sont de type '1-1'.

# Etude et réalisation d'une base de données

## Le modèle entité-association

### ➤ Association ‘Livre’

- **Livreur** : un livreur peut-il n'avoir jamais livré de pizzas et peut-il en avoir livré plusieurs ? Un livreur a au moins effectué une (1) livraison, sinon il n'est pas considéré comme tel. On peut imaginer que l'on ne rentre les informations associées à un livreur qu'à partir du moment où il a réellement effectué une livraison. Il est supposé faire plusieurs (n) livraisons. Les cardinalités associées sont de type ‘1-n’.
- **Commande** : une commande peut-elle avoir été livrée par plusieurs livreurs ou par aucun ? Une commande est livrée par un (1) et un (1) seul livreur. Les cardinalités associées sont de type ‘1-1’.

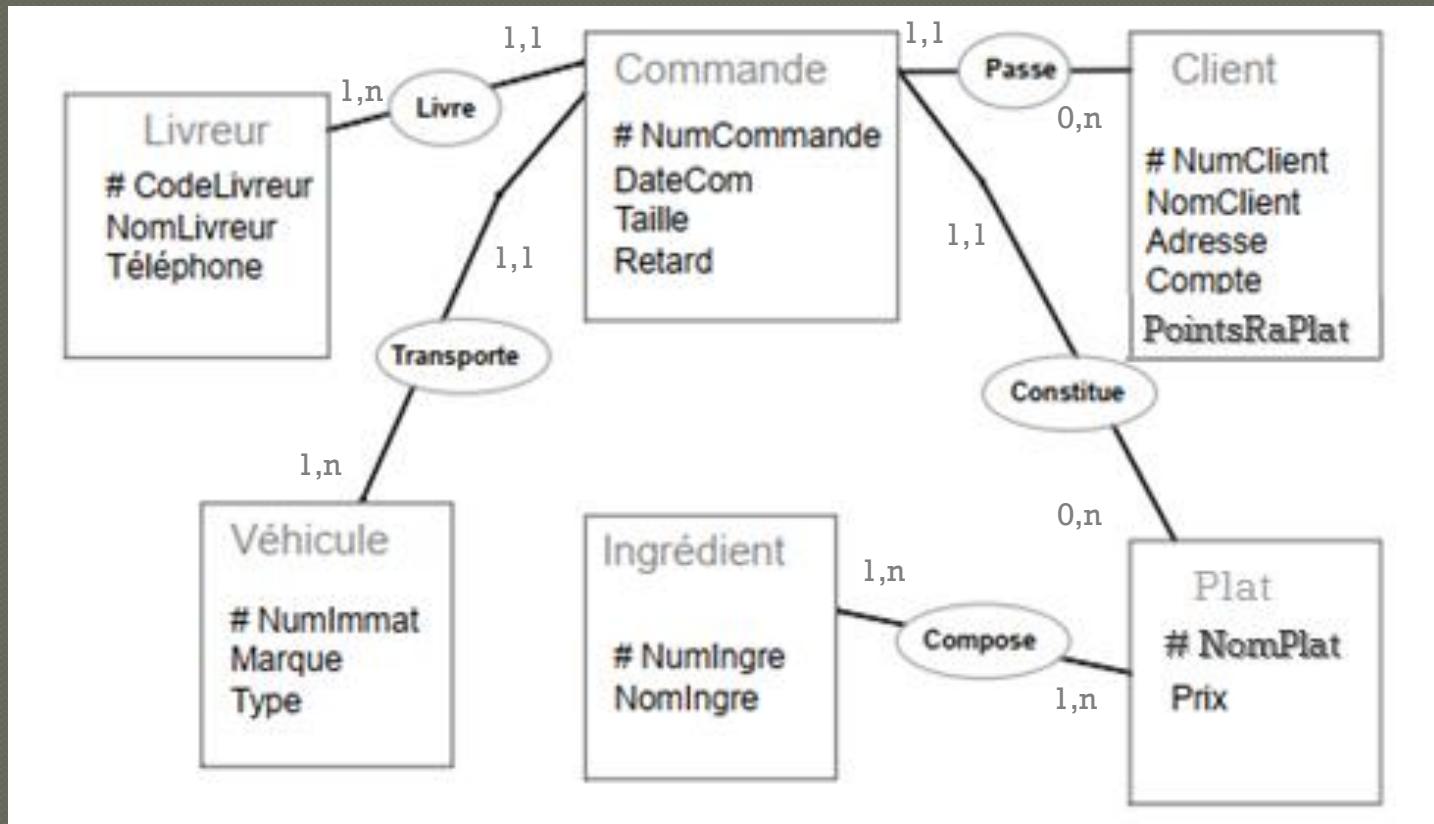
### ➤ Association ‘Transporte’

- **Véhicule** : un véhicule peut-il n'avoir jamais livré de plat et peut-il en avoir livré plusieurs ? Dans ce cas, la situation n'est pas tout à fait la même que pour les livreurs. On peut imaginer que les informations concernant un véhicule pourraient être insérées dans la base de données sans que le véhicule n'ait encore effectué une livraison (0). Si c'était le cas, on aurait des cardinalités de type ‘0-n’. On choisit ici des cardinalités associées de type ‘1-n’. Dans les deux cas, un véhicule est supposé être utilisé pour livrer plusieurs (n) commandes.
- **Commande** : une commande peut-elle avoir été livrée par plusieurs véhicules ou par aucun ? Une commande est livrée par un (1) et un (1) seul véhicule. Les cardinalités associées sont de type ‘1-1’.

# Etude et réalisation d'une base de données

## Le modèle entité-association

On obtient le modèle entité-association suivant :



# Etude et réalisation d'une base de données

## Passage au modèle relationnel

Dans cette étape, nous présenterons le travail préliminaire à effectué en vue du passage à *l'utilisation d'un SGBD*. Elle comprend plusieurs parties :

- la transformation du modèle entité-association en modèle relationnel ;
- la vérification de la conformité des relations créées par rapport à la définition des formes normales du modèle relationnel ;
- la discussion sur le type des données à adopter pour chaque champ ainsi que sur les contraintes d'intégrités à définir ;
- la création des tables en SQL.

### ○ Transformation du modèle entité-association

Les deux règles générales de passage du modèle entité-association vers le modèle relationnel sont les suivantes :

- Une entité donne une relation de même clé que l'entité qui contient les mêmes attributs que l'entité.
- Une association donne une relation dont la clé est composée des deux clés des entités associées et des attributs de l'association.

# Etude et réalisation d'une base de données

## Passage au modèle relationnel

### ➤ Application de la règle générale

On obtient les relations suivantes à partir des entités :

- commande (NumCommande, DateCom, Taille, Retard) ;
- client (NumClient, NomClient, Adresse, Compte, PointsRaPlat) ;
- pizza (NomPlat, Prix) ;
- livreur (CodeLivreur, NomLivreur, Telephone) ;
- véhicule (NumImmat, Marque, Type) ;
- ingrédient (NumIngre, NomIngredient).

En appliquant la règle générale, on obtient les relations suivantes à partir des associations :

- livre (CodeLivreur, NumCommande) ;
- transporte (NumImmat, NumCommande) ;
- passe (NumClient, NumCommande) ;
- constitue (NomPlat, NumCommande) ;
- compose (NumIngre, NomPlat).

# Etude et réalisation d'une base de données

## Passage au modèle relationnel

### ➤ Cas particuliers des associations de cardinalité 1-1

Un cas particulier existe lorsque l'une des cardinalités d'une association est de type '1-1'. La relation représentant l'association disparaît et fusionne avec la relation représentant l'entité associée avec la cardinalité '1-1'. On peut dire qu'il y a « aspiration » de l'association par l'entité.

Nous aurons finalement les relations suivantes:

- client (NumClient, NomClient, Adresse, Compte, PointsRaPlat) ;
- plat (NomPlat, Prix) ;
- livreur (CodeLivreur, NomLivreur, Telephone) ;
- véhicule (NumImmat, Marque, Type) ;
- ingrédient (NumIngre, NomIngredient) ;
- commande(NumCommande, DateCom, Taille, Retard, CodeLivreur, NumImmat, NumClient, NomPizza) ;
- compose (NumIngre, NomPizza).

# Etude et réalisation d'une base de données

## Passage au modèle relationnel

---

### ● Vérification de la conformité aux formes normales

On doit vérifier que l'ensemble de relations créées à l'étape précédente est bien en conformité avec les formes normales. On peut se limiter en général aux trois premières formes normales.

#### ➤ Première forme normale

Chaque champ doit posséder une valeur « atomique ». Ici, aucune des relations ne dispose de champs à plusieurs valeurs. On peut affirmer avec certitude qu'elles sont en première forme normale.

#### ➤ Deuxième forme normale

La recherche de non-conformité à la deuxième forme normale n'a de sens que si la clé est composée de multiples champs. Dans notre cas, la seule relation qui possède une clé « composite » est la relation... ‘compose’. De plus, cette relation ne contient aucun attribut qui ne fasse pas partie de la clé. La relation est en deuxième forme normale. Toutes les relations sont en deuxième forme normale.

# Etude et réalisation d'une base de données

## Passage au modèle relationnel

### ➤ Troisième forme normale

La troisième forme normale concerne les liens qui peuvent exister entre des champs qui ne font pas partie de la clé d'une relation.

Si l'on considère la relation 'client', les champs ne faisant pas partie de la clé sont les suivants : 'NomClient', 'Adresse', 'Compte', 'PointsRaPlat'. Il est clair que le solde du compte ne permet de déterminer ni le nom du client, ni son adresse, ni son nombre de points de fidélité. Pareil pour les autres aussi. La question pourrait éventuellement se poser pour le champ 'Adresse', susceptible dans certains cas, d'établir le nom du client si l'on suppose que deux clients n'habitent pas à la même adresse.

suivant le même raisonnement, sur la relation 'vehicule', on peut être amené à identifier une relation de dépendance entre le type qui est unique pour un véhicule. Mais sa décomposition risque de devenir complexe. On gagnera à accepter une redondance relativement faible.

on ne trouve de relations de dépendance entre aucun des champs de la relation 'commande' qui ne font pas partie de la clé, c'est-à-dire les champs 'DateCom', 'Taille', 'Retard', 'CodeLivreur', 'NumImmat', 'NumClient' et 'Nomlat'.

Toutes les relations ne sont pas strictement en troisième forme normale, mais on considère les redondances résiduelles comme acceptables.

# Etude et réalisation d'une base de données

## Passage au modèle relationnel

- Types de données et contraintes
  - Types de données

Un type des données a pour but de faire une première restriction sur les valeurs que peut prendre un champ et surtout de spécifier les opérations et fonctions qu'il sera possible de lui appliquer.

### ***Table client***

NumClient : entier

NomClient : chaîne de caractères (obligatoire)

Adresse : chaîne de caractères

Compte : réel

PointsRaPlat: entier

### ***Table livreur***

CodeLivreur: entier

NomLivreur : chaîne de caractères (obligatoire)

Telephone: chaîne de caractères (obligatoire)

### ***Table ingredient***

NumIngréd: entier

NomIngréd:chaîne de caractères(obligatoire)

### ***Table compose***

NumIngre : entier

NomPlat : chaîne de caractères (obligatoire)

### ***Table plat***

NomPlat: chaîne de caractères

Prix: réel (obligatoire)

### ***Table véhicule***

NumImmat : chaîne de caractères

Marque : chaîne de caractères (obligatoire)

Type : chaîne de caractères (obligatoire)

### ***Table commande***

NumCommande: entier

DateCom : date (obligatoire)

Taille: chaîne de caractères

retard: chaîne de caractères

CodeLivreur : entier

NumImmat : chaîne de caractères

NumClient : chaîne de caractères

NomPlat : chaîne de caractères

# Etude et réalisation d'une base de données

## Passage au modèle relationnel

### ➤ Contraintes d'intégrité

Les contraintes permettent de décrire de manière plus précise les ensembles auxquels appartiennent les champs.

- Le champ ‘Prix’ de la table ‘plat’ peut être limité à l’intervalle 500 .. 2500.
- On pourrait définir un intervalle de validité pour les dates de commande (champ ‘DateCom’ de la table ‘commande’), par exemple la date de commande doit être supérieure ou égale à la date du jour.
- Le champ ‘Retard’ de la table ‘commande’ doit contenir les valeurs ‘O’ ou ‘N’.
- Le champ ‘Taille’ de la table ‘commande’ doit contenir les valeurs « normale », « large », « normale amélioré » ou « large amélioré ».
- Les champs ‘NomPlat’ dans les autres champs constituent une contrainte de référence faisant référence au champ ‘NomPlat’ de la table ‘plat’.
- ‘CodeLivreur’ de la relation ‘commande’ fait référence au champ ‘CodeLivreur’ de la relation ‘livreur’.
- ‘NumImmat’ de la relation ‘commande’ fait éférence au champ ‘NumImmat’ de la relation ‘vehicule’.
- ‘NumClient’ de la relation ‘commande’ fait référence au champ ‘NumClient’ de la relation ‘client’.

# Etude et réalisation d'une base de données

## Passage au modèle relationnel

### ○ Création des tables

À la création des tables, on spécifie le nom des champs, leur type, les contraintes d'intégrité et la définition des clés.

#### Table ‘client’

```
CREATE TABLE client (
    NumClient INT PRIMARY KEY,
    NomClient VARCHAR(20) NOT NULL,
    Adresse VARCHAR(150) ,
    Compte REAL,
    PointsRaPlat INT
);
```

#### Table ‘livreur’

```
CREATE TABLE livreur (
    CodeLivreur INT PRIMARY KEY,
    NomLivreur CHAR(30) NOT NULL,
    TeleLivreur CHAR(30) NOT NULL
);
```

#### Table ‘plat’

```
CREATE TABLE plat(
    NomPlat CHAR(30) PRIMARY KEY,
    Prix CHAR(30) NOT NULL,
    CHECK (Prix BETWEEN 500 AND 2500)
);
```

#### Table ‘ingredient’

```
CREATE TABLE ingredient(
    NumIngred INT PRIMARY KEY ,
    NomIngred CHAR(30) NOT NULL
);
```

#### Table ‘vehicule’

```
CREATE TABLE vehicule(
    NumImmat CHAR(30) PRIMARY KEY,
    Marque CHAR(30) NOT NULL,
    Type CHAR(30) NOT NULL
);
```

# Etude et réalisation d'une base de données

## Passage au modèle relationnel

Table 'compose'

```
CREATE TABLE compose (
    NomPlat CHAR(30) REFERENCES plat(NomPlat),
    NumIngred INT REFERENCES ingredient(NumIngred),
    PRIMARY KEY (NomPlat, NumIngred)
);
```

Table 'commande'

```
CREATE TABLE commande (
    NumCommande INT PRIMARY KEY,
    DateCom DATE,
    Taille CHAR(30) NOT NULL,
    Retard CHAR(1) NOT NULL,
    NumClient INT REFERENCES client(NumClient),
    NomPlat CHAR(30) REFERENCES plat(NomPlat),
    CodeLivreur INT REFERENCES livreur(CodeLivreur),
    NumImmat CHAR(30) REFERENCES vehicule(NumImmat),
    CHECK Taille in ('normale', 'large', 'normale ameliore', 'large ameliore'),
    CHECK Retard in ('O','N')
);
```

# PLAN

---

- Généralités
- Le modèle conceptuel
- Le modèle relationnel
- Le langage SQL
- Etude et réalisation d'une base de données
- La sécurité des données

# La sécurité des données

## Objectif

- Dans cette section, nous allons traiter la sécurité des données qui est une notion fondamentale des bases de données.
- En effet, des pertes ou des modifications de données peuvent accidentellement ou intentionnellement causer de sérieux problèmes aux structures ou organisme dont les activités reposent essentiellement sur les données.
- Il convient alors de les protéger et aussi d'en assurer la disponibilité permanente.
  - Il s'agira de la protection de l'accès à la machine d'un point de vue physique ou réseau, les aspects de durée de vie du système ainsi que la politique de sauvegarde ;
  - De la politique de restriction d'accès aux données du SGBD par des comptes et l'utilisation des vues ;
  - De la capacité des outils du SGBD à protéger les opérations sur les données, à les restaurer et à revenir en arrière, comme pour les transactions.
  - Il convient également d'aborder la notion de trigger pour assurer une bonne gestion du contenu des données lors des opérations d'insertion, de mise à jour ou de suppression de données).

# La sécurité des données

## Contrôle d'accès et sauvegarde

### ● Contrôle d'accès et sauvegarde

On décrit dans cette partie, plusieurs types de préoccupations:

#### ➤ L'accès physique à la machine

Un accès physique permet généralement de contourner toutes les protections liées aux système d'exploitation ou au SGBD en question.

- Il suffit de démarrer une machine avec un autre système d'exploitation présent sur un CD ou une clé USB par exemple pour obtenir un accès au disque et ainsi aux données associées au SGBD.
  - La première précaution à prévoir consiste à protéger le BIOS (Basic Input Output System) ou l'équivalent de la machine pour l'empêcher de démarrer sur un autre disque que celui qui contient le « bon » système d'exploitation.
- Le fait de pouvoir accéder directement à la machine permet également à une personne animée de mauvaises intentions d'empêcher l'accès aux données en détruisant physiquement le disque sur lequel se trouvaient les fichiers ou même la machine.
  - Par conséquent, la ou les machines qui contiennent le SGBD et les fichiers de données doivent se trouver dans des pièces dont l'accès est, au minimum, contrôlé.
- L'alimentation électrique ne doit pas aussi être accessible.

Ces précautions ne sont pas toujours d'ordre purement technique ; elles relèvent souvent du simple bon sens commun.

# La sécurité des données

## Contrôle d'accès et sauvegarde

### ➤ l'accès distant à la machine à travers le réseau

L'accès à la machine par le réseau est devenu une porte d'entrée privilégiée pour les personnes mal intentionnées, car rares sont les serveurs qui ne sont pas connectés aujourd'hui. On distingue ainsi principalement deux aspects de problèmes de sécurité :

- les attaques « internes » qui proviennent de personnes ayant accès à la machine par un compte standard associé à un mot de passe correct ;
- les attaques « externes » qui se font en profitant d'une faille du système d'exploitation ou d'une des applications installées sur la machine.

Dans ces cas de figure, les protections dépendent spécifiquement de l'administrateur système et réseau et pas forcément de l'administrateur du SGBD.

- Le système et les applications présentes sur la machine doivent être maintenus et mis à jour régulièrement pour éviter les problèmes.
  - L'administrateur système doit également tenir à jour les comptes utilisateurs qui possèdent les droits d'accès à la machine et vérifier la qualité des mots de passe associés.
- ❖ Classiquement, une intrusion se fait en utilisant un compte « oublié », créé par exemple pour un besoin ponctuel, dont le mot de passe est assez simple pour être deviné.

# La sécurité des données

## Contrôle d'accès et sauvegarde

### ➤ la pérennité du système

Une base de données se conçoit généralement pour de nombreuses années d'utilisation. Il est donc essentiel, avant de choisir une machine, son système d'exploitation ainsi que le logiciel de SGBD, de prendre en considération leurs pérennités respectives.

Le choix peut alors se porter sur des ensembles moins performants mais que l'on considère comme plus viables en termes de durée de vie. Parmi lesquels, nous considérons:

- **Le suivi de la machine elle-même:** la durée de la garantie matérielle, la possibilité de retrouver les pièces etc.
- **La durée de vie du système d'exploitation:** Un système qui n'est plus mis à jour par son éditeur et qui présente des trous importants de sécurité se révèle un très mauvais choix.
- **L'évolutivité du SGBD:** Un SGBD propriétaire qui n'offre pas les échanges de données avec d'autres logiciels handicape la migration vers un autre SGBD. De même que pour les points précédents, un SGBD qui présente des failles importantes de sécurité d'accès ou de fonctionnement et qui n'est plus mis à jour par l'éditeur constitue un problème potentiel qu'aucune autre précaution ne saurait combler.

# La sécurité des données

## Contrôle d'accès et sauvegarde

### ➤ la sauvegarde et la redondance

- **Sauvegarde des données**

Des copies de sauvegarde des données contenues dans la base, mais aussi des métadonnées (dictionnaire de données) du SGBD, doivent être réalisées régulièrement.

- Outre les problèmes de malveillance, un incendie ou une inondation peuvent réduire à néant des années de travail.
  - ❖ Il faut disposer d'une armoire de stockage adaptée à l'abri des inondations. Si ce stockage se trouve dans un lieu différent, c'est encore mieux.

- **Redondance des données**

Un procédé plus efficace et plus sûr consiste à disposer de copies de la base en plusieurs endroits géographiquement distincts. La plupart des SGBD sont capables d'effectuer des mises à jour sur des copies de la base de données situées sur d'autres machines via le réseau.

Cette méthode permet de plus d'améliorer la disponibilité des données :

- Si l'un des serveurs est inaccessible, un autre peut prendre le relais.
- Il est possible ainsi de répartir la charge sur plusieurs serveurs.

# La sécurité des données

## Limitations d'accès au SGBD

### ● Limitations d'accès au SGBD

Les limitations concernent les autorisations accordées ou non sur les différents éléments que contient le SGBD et vont permettre de contrôler l'accès général au SGBD.

#### ➤ La gestion des utilisateurs

Les instructions qui permettent d'affecter les droits sont définies dans la norme SQL.

- Classiquement, pour protéger une ressource, on commence par en interdire l'accès à tous. Puis on décrit ceux qui sont autorisés à l'utiliser (utilisateurs). On associe ensuite à chaque utilisateur un identifiant (mot de passe) pour éviter les emprunts d'identité.
- Le SGBD stocke les informations suivantes pour chaque objet qu'il contient :
  - Le type de droit : Sélection, création, destruction...
  - L'objet sur lequel s'appliquent les droits. Une base de données, une table, une vue...
  - Le nom de l'utilisateur.
  - L'identifiant, au sens du mot de passe.
  - Le donneur des droits.
- pour créer un utilisateur avec la norme SQL, c'est sous la forme suivante:
  - CREATE USER <type de droit> IDENTIFIED BY <identifiant> ;
  - DROP USER permet de détruire l'utilisateur créé.
- Un utilisateur peut prendre le(s) rôle(s) dont il a besoin (rôle) pour pouvoir travailler sur les objets du SGBD qui le concerne. Pour créer un rôle, on fait:
  - CREATE ROLE consultation\_seulement ;

# La sécurité des données

## Limitations d'accès au SGBD

### ➤ La gestion des droits

Une fois que l'utilisateur est connecté au système, il convient de gérer son accès aux données. Les droits sont distribués à des utilisateurs, des groupes ou des rôles.

- Avec SQL, il est possible de spécifier des droits essentiellement sur les opérations de manipulation de tables (ou de vues considérées comme des tables) suivantes :
  - interrogation (SELECT) ;
  - insertion (INSERT) ;
  - mise à jour (UPDATE) ;
  - destruction (DELETE) ;
  - référence à une table (REFERENCE).
- L'instruction GRANT permet de réaliser l'association des droits et du bénéficiaire. Sa forme générale est la suivante :
  - \* GRANT <type de droit> ON <objet> TO <nom utilisateur>

# La sécurité des données

## Limitations d'accès au SGBD

Créons un utilisateur nommé my\_user:

- \* CREATE USER my\_user IDENTIFIED BY password;
- \* L'option 'ALL PRIVILEGES' sert à donner tous les droits à l'utilisateur. L'administrateur dispose de tous les droits sur tous les objets. Exemple:

```
$ CREATE DATABASE ma_base;
$ GRANT ALL PRIVILEGES ON ma_base.* TO 'my_user';
$ CREATE USER user_2 IDENTIFIED BY password_2;
$ USE ma_base;
$ CREATE TABLE ma_classe (NumClasse INT PRIMARY KEY, Titre CHAR(50));
$ GRANT SELECT ON ma_classe TO 'user_2'; (ERREUR)
```
- \* L'utilisateur my\_user n'a pas le droit de retransmettre ses droits à un autre utilisateur. L'administrateur aurait dû entrer la commande suivante :

```
$ GRANT ALL PRIVILEGES ON ma_base.* TO 'my_user' WITH GRANT OPTION;
```
- \* Il pourra alors attribuer des droits aux autres utilisateurs:

```
$ GRANT SELECT ON ma_classe TO 'user_2'; (CORRECTE)
```
- \* on peut donner un droit à tous les utilisateurs du SGBD avec le mot clé PUBLIC, comme suit:

```
$ GRANT SELECT ON ma_classe TO PUBLIC;
```
- \* On peut affiner ces permissions en ne les accordant que pour certains champs de la table.

```
$ GRANT UPDATE Titre ON ma_classe TO 'user_2';
```

# La sécurité des données

## Limitations d'accès au SGBD

L'instruction GRANT permet également de donner un rôle à un utilisateur ou à un ensemble d'utilisateurs et s'utilise alors de la manière suivante :

- \* GRANT <rôle> TO <liste des noms d'utilisateur séparés par des virgules>
- \* On crée les rôles et on met à jour les droits nécessaires. On peut considérer les rôles suivant (consultation : interrogation ; utilisation : mise à jour et insertion ; et gestion : destruction) sur ma\_classe.

```
$ CREATE ROLE consultation;
$ CREATE ROLE utilisation;
$ CREATE ROLE gestion;
$ GRANT SELECT ON ma_classe TO consultation;
$ GRANT UPDATE, INSERT ON ma_classe TO utilisation;
$ GRANT DELETE ON ma_classe TO gestion;
```
- \* On affecte les rôles aux utilisateurs.

```
$ GRANT consultation TO 'my_user', 'user_2';
$ GRANT utilisation TO 'my_user', 'user_2';
$ GRANT gestion TO 'my_user';
```
- \* Pour retirer les droits accordés par l'instruction GRANT, on utilise l'instruction REVOKE. Elle est de la forme suivante :

```
$ REVOKE <type de droit> ON <objet> FROM <nom utilisateur>;
```
- \* On peut affiner ces permissions en ne les accordant que pour certains champs de la table.

```
$ GRANT UPDATE Titre ON ma_classe TO 'user_2';
```
- \* Exemple: REVOKE SELECT ON ma\_classe FROM 'user\_2' ;

# La sécurité des données

## Limitations d'accès au SGBD

### ➤ Utilisation des vues

Une approche complémentaire consiste à définir plus finement les objets sur lesquels on affecte les droits en se servant simplement des vues SQL. Celles-ci permettent de « cacher » certaines données à l'aide de critères de sélection précis. De plus, les vues permettent de masquer aux utilisateurs la complexité de la structure des données, leur fournissant ainsi une interface plus commode avec la base de données. Nous allons manipuler la base de données ‘casse’, en guise d'exemple.

- Vue utilisant une jointure simple

Le service marketing veut vérifier s'il n'y a pas de corrélations entre la couleur des véhicules vendus et la ville dans laquelle résident les clients qui les achètent. Il est inutile de fournir à ce service la procédure pour obtenir ces informations qui nécessite de lier les trois tables ‘voiture’, ‘vente’ et ‘client’.

- On crée une vue qui contient uniquement la projection de ces deux informations et qui sélectionne les voitures vendues.  
\$ CREATE VIEW couleur\_ville (Couleur, Ville) AS SELECT voiture.couleur, client.ville FROM voiture JOIN vente JOIN client ON voiture.NumVoit=vente.Numvoit AND client.NumAch = vente.Numach ;
- On donne les droits de visualiser ces informations aux personnes du service marketing.  
\$ GRANT SELECT ON couleur\_ville TO ‘my\_user’, ‘user\_2’ ;

# La sécurité des données

## Limitations d'accès au SGBD

- Vue utilisant une jointure plus élaborée

On souhaite que tous les utilisateurs puissent consulter le catalogue des voitures non encore vendues. Cette liste de voitures est accessible par une requête plus complexe que la précédente, de type jointure externe. Il s'agit d'un cas d'utilisation d'une vue, très pratique pour les utilisateurs qui n'ont pas à connaître les notions avancées de jointure externe.

```
$ CREATE VIEW catalogue (NumVoit, Marque, Type, Couleur) AS  
SELECT voiture.NumVoit, voiture.Marque, voiture.Type,  
voiture.Couleur FROM voiture LEFT OUTER JOIN vente ON  
voiture.NumVoit=vente.Numvoit WHERE vente.Numvoit IS NULL;
```

- On donne la permission à l'utilisateur PUBLIC, c'est-à-dire à tous les utilisateurs.

```
$ GRANT SELECT ON catalogue TO PUBLIC;
```

# La sécurité des données

## Limitations d'accès au SGBD

- Vue avec possibilité de mise à jour

Le service comptabilité doit pouvoir accéder aux informations de vente et de clientèle afin de facturer et calculer le chiffre d'affaires. En outre, ce service a besoin de mettre à jour le prix de vente.

- L'option CHECK OPTION permet les mises à jour des données au travers de la vue.

```
$ CREATE VIEW journal_compta (Date_de_vente, Prix_de_vente,  
Nom_client, Ville_client) AS SELECT vente.DateVent, vente.Prix,  
client.Nom, client.Ville FROM vente JOIN client ON  
vente.NumAch=client.NumAch WITH CHECK OPTION ;
```

- On donne les droits à un rôle que les utilisateurs vont utiliser.

```
$ CREATE ROLE comptabilite ;
```

```
$ GRANT SELECT ON journal_compta TO comptabilite;
```

```
$ GRANT UPDATE Prix_de_vente ON journal_compta TO comptabilite;
```

- On donne le rôle 'comptabilité' aux utilisateurs

```
$ GRANT comptabilite TO 'my_user', 'user_2';
```

# La sécurité des données

## Transactions

### ○ Transactions

Nous présentons dans cette section, plusieurs outils procurés par les SGBD pour protéger les opérations effectuées sur les données. Les incidents liés aux données dans un SGBD proviennent essentiellement de l'accès concurrent à ces dernières par plusieurs utilisateurs. Ces problèmes sont habituellement résolus par les mécanismes associés aux *transactions*. Elles permettent également de résoudre les pertes dues aux erreurs de manipulation ou celles liées aux erreurs dans le traitement des données, par exemple en cas de panne matérielle.

#### ➤ Accès concurrentiel aux données

En définissant des accès et des permissions sur les différents objets gérés par le SGBD, on a donné accès à plusieurs utilisateurs ou applications, en même temps, aux données. De plus, la plupart des machines sont connectées à un réseau, ce qui augmente encore le nombre potentiel d'utilisateurs simultanés.

*Nous allons voir quelques types d'incohérences qui peuvent être provoquées par l'accès concurrent aux données.*

# La sécurité des données

## Transactions

- Lecture étrange

L'accès multiple en lecture ne pose pas habituellement de problèmes, mais que se passe-t-il lorsqu'un ou plusieurs utilisateurs décident de modifier les mêmes données au même moment ?

On considère la séquence d'instructions suivante appliquée à la base de données « casse ». On suppose que tous les utilisateurs disposent de tous les droits sur tous les objets (tables et champs) de cette base de données :

- L'utilisateur 'my\_user' consulte la liste des voitures non vendues.
- L'utilisateur 'user\_2' enregistre la vente d'une voiture.
- L'utilisateur 'my\_user' relance la même requête et trouve un résultat différent.
- L'utilisateur 'user\_2' invalide la vente de cette voiture.
- L'utilisateur 'my\_user' pensant s'être trompé relance la même requête qui aboutit au résultat initial.

Pour trois exécutions de la même requête, l'utilisateur 'my\_user' va obtenir des résultats différents.

On pourrait utiliser ici le terme de **lecture fantôme** pour qualifier les problèmes rencontrés : une donnée apparaît puis disparaît.

# La sécurité des données

## Transactions

- Incohérence de résultats

On peut imaginer une autre série d'instructions qui réalisent des modifications de données effectuées par différents utilisateurs. En raison de l'augmentation des frais de structure, le service comptable a décidé d'une augmentation générale du prix de vente de 5%.

Afin que cette dernière passe inaperçue et dans le cadre d'une campagne de communication, le service marketing offre 100 euros de ristourne sur tout le catalogue pendant un mois.

Supposons que l'utilisateur 'my\_user' appartient au service comptable et l'utilisateur 'user\_2' au service marketing. Les vérifications sont effectuées par un troisième utilisateur 'user\_3' de la direction.

- L'utilisateur 'my\_user' consulte le prix de vente de la voiture 'X'.
- L'utilisateur 'user\_2' effectue la modification de promotion sans vérifier le prix de vente ( $\text{Prix\_vente} = \text{Prix\_vente} - 100$ ).
- L'utilisateur 'my\_user' effectue la modification d'augmentation ( $\text{Prix\_vente} = \text{Prix\_vente} \times 1,05$ ).

L'utilisateur 'user\_3' vérifie le résultat de l'opération de modification du prix de vente et constate une différence avec le résultat attendu: le prix est égal à  $(\text{Prix\_vente} - 100) \times 1,05$  alors qu'il s'attendait à un prix égal à  $(\text{Prix\_vente} \times 1,05) - 100$ .

On obtient alors une « incohérence » due à la séquence de modification des données.

# La sécurité des données

## Transactions

- **Verrous**

On a donc besoin de disposer d'un mécanisme qui garantisse une certaine « exclusivité » sur les données lors des opérations de mise à jour: il s'agit des verrous.

L'idée est simple: on bloque une ressource pour effectuer les opérations et on la libère dès que les opérations sont effectuées.

Même si cette méthode semble résoudre le problème ; elle présente cependant quelques pièges et doit être utilisée avec prudence. En effet, on peut rapidement parvenir à une situation de blocage que l'on nomme **étreinte fatale** (deadlock en anglais) ou parfois **interblocage**. Pour illustrer cette situation, on suppose que deux vendeurs veulent mettre à jour la base de données ‘casse’ :

- L'utilisateur 'my\_user' veut insérer une vente de voiture dans la table 'vente' et constate à cette occasion une erreur sur la couleur dans la table 'voiture' qu'il veut modifier.
- L'utilisateur 'user\_3' veut insérer à la fois une nouvelle voiture dans la table 'voiture' et la mention de sa vente dans la table 'vente'.

La séquence d'instruction peut être la suivante :

- L'utilisateur 'my\_user' pose un verrou sur la table 'vente'.
- L'utilisateur 'user\_3' pose un verrou sur la table 'voiture'.
- L'utilisateur 'my\_user' a terminé la mise à jour de 'vente' et veut réaliser celle de 'voiture'... mais la table 'voiture' est verrouillée par 'user\_3'.
- L'utilisateur 'user\_3' a terminé la mise à jour de 'voiture' et veut procéder à celle de 'vente'... mais la table 'vente' est verrouillée par 'my\_user'.

On parvient à une situation où les deux utilisateurs attendent que l'autre libère la ressource pour continuer.

# La sécurité des données

## Transactions

---

D'autres phénomènes de blocage peuvent survenir dans l'utilisation des verrous. On parle parfois dans ce cas de **famine**. Voici un exemple illustrant cette possibilité (algorithme des « lecteurs-rédacteurs »).

- Tous les lecteurs peuvent lire en même temps.
- Si un rédacteur veut écrire, aucun lecteur ne doit plus utiliser la donnée.
- Si un rédacteur est en train d'écrire, aucun lecteur ne peut lire la donnée et aucun autre rédacteur ne peut y accéder (accès exclusif).

S'il existe un grand nombre de lecteurs qui se succèdent en lecture sur la donnée, l'attente pour un rédacteur qui voudrait la modifier peut alors devenir quasi infinie.

Laisser la gestion des verrous à un utilisateur est donc délicat et peut conduire à des blocages du système. Les SGBD performants disposent d'outils capables de détecter et résoudre ces phénomènes, voire de les prévenir le cas échéant.

# La sécurité des données

## Transactions

### ➤ Mécanisme des transactions

Nous allons essayer de résoudre les problèmes d'accès concurrents vus précédemment, en utilisant les transactions. L'idée est de considérer un ensemble d'instructions comme une seule instruction. L'ensemble d'instructions est dit **unitaire ou atomique**.

- Propriétés des transactions

La notion d'atomicité peut être décrite en ces termes :

- Soit le SGBD est capable d'exécuter toutes les instructions qui composent une transaction et il effectue les mises à jour provoquées par ces instructions.
- Soit il n'y parvient pas et il remet la base de données dans l'état cohérent précédent le début de l'exécution des instructions de la transaction.

Les propriétés que doivent vérifier les transactions sont résumées par le terme ACID :

- **Atomicité.** Une transaction est atomique : elle est exécutée entièrement ou abandonnée.
- **Cohérence.** La transaction doit se faire d'un état cohérent de la base vers un autre état cohérent.
- **Isolement.** Des transactions simultanées ne doivent pas interférer entre elles.
- **Durabilité.** La transaction a des effets permanents même en cas de panne.

# La sécurité des données

## Transactions

---

- Réalisation des transactions dans les SGBD

Pour mettre en œuvre les transactions, le SGBD doit offrir l'exclusivité d'accès aux données ainsi que la possibilité d'annuler des modifications effectuées. Afin de garantir l'accès exclusif aux données, le SGBD utilise les verrous vus précédemment. Ces mécanismes sont associés à des algorithmes de protection sophistiqués afin de prévenir les problèmes des méthodes de verrouillage.

La possibilité de revenir en arrière repose sur l'utilisation d'un journal de transactions. Pour ce faire, le SGBD garde une trace de toutes les requêtes de la transaction et des données qui sont modifiées par ces instructions. On conserve donc un journal de l'ensemble des lectures et écritures afin de pouvoir reconstituer un système cohérent. En utilisant ce journal, à partir d'un état courant du système, on est capable de réexécuter les instructions qui n'ont pu l'être effectivement ou on revient en arrière si ce n'est pas possible.

# La sécurité des données

## Transactions

---

- Différents niveaux de transaction

La norme SQL fixe plusieurs degrés de qualité pour les transactions : on parle de niveaux d'isolation. Les niveaux standard sont les suivants :

- 0, READ UNCOMMITTED. Lecture de données non validées en cours de transaction (niveau le plus bas).
- 1, READ COMMITTED. Modification des données possible en cours de transaction.
- 2, REPEATABLE READ. Insertion de nouveaux enregistrements possible en cours de transaction.
- 3, SERIALIZABLE. Toutes les transactions sont traitées en « série » (niveau le plus sûr).

La sécurité a un coût en matière de performances et, dans certains contextes, il n'est pas nécessaire de demander au SGBD de traiter une requête avec le niveau maximal d'exclusivité. Il est clair que, pour une simple opération de lecture, le niveau 1 qui garantit l'impossibilité de lire des données non validées peut être suffisant

# La sécurité des données

## Transactions

- Syntaxe SQL des transactions

L'instruction pour débuter une transaction est START TRANSACTION. La transaction prendra fin par l'une des instructions suivantes :

- COMMIT. Les instructions effectuées depuis START TRANSACTION sont validées.
- ROLLBACK. Les instructions effectuées depuis START TRANSACTION sont annulées.

Voici un exemple d'insertion et de mise à jour dans la base de données 'casse' que l'on annule ensuite :

```
#Démarrage de la transaction
START TRANSACTION ;
#Insertion d'un tuple dans la table 'client'
INSERT INTO client VALUES (6,'Laetitia',34,'Paris','F');
#Mise à jour des données de prix (augmentation de 5 %)
UPDATE vente SET Prix=Prix*1.05 ;
#Annulation des modifications faites depuis START TRANSACTION
ROLLBACK ;
```

Voici un exemple de destruction dans la table 'client' de la base de données 'casse' que l'on valide ensuite :

```
#Démarrage de la transaction
START TRANSACTION ;
#Destruction des tuples dont le champ 'Ville' vaut 'Paris' dans la table 'client'
DELETE FROM client WHERE Ville='Paris';
#Validation des modifications faites depuis START TRANSACTION
COMMIT ;
```

On peut choisir le niveau d'isolation au moment où l'on démarre la transaction. Dans l'exemple suivant, on fixe le niveau maximal de sécurité.

```
START TRANSACTION
ISOLATION LEVEL SERIALIZABLE ;
...
```

# La sécurité des données

## Transactions

- « Points de retour » dans les transactions

Les instructions contenues dans une transaction sont considérées par le SGBD comme « unitaires », c'est-à-dire qu'elles sont exécutées d'un seul bloc ou ne sont pas exécutées. L'instruction ROLLBACK annule toutes les instructions effectuées depuis l'instruction START TRANSACTION. Il est possible de diviser cet ensemble d'instructions en sous ensembles par la définition de « points de retour » utilisés par l'instruction ROLLBACK. Au lieu d'annuler les instructions exécutées depuis START TRANSACTION, on revient à l'état de la base de données depuis le point défini par l'instruction SAVEPOINT.

```
#Démarrage de la transaction  
START TRANSACTION ;  
  
#Insertion d'un tuple dans la table 'client'  
INSERT INTO client VALUES (8,'Annabelle',29,'Paris','F');  
  
# Définition d'un point de retour que l'on nomme 'UN'  
SAVEPOINT 'UN'  
  
#Mise à jour des données de prix (augmentation de 5 %)  
UPDATE vente SET Prix=Prix*1.05 ;  
  
#Destruction des tuples dont le champ 'Ville' vaut 'Paris' dans la table 'client'  
DELETE FROM client WHERE Ville='Paris';  
  
#Annulation des modifications faites depuis le point 'UN'  
ROLLBACK TO 'UN' ;  
  
...
```

Une différence essentielle par rapport à l'instruction ROLLBACK vue précédemment est que, dans ce cas, la transaction n'est pas terminée. Un autre ROLLBACK reviendrait à l'état de la base de données avant l'exécution de l'instruction START TRANSACTION.

Le mécanisme des points de retour est très utile pour prévenir les cas de mauvaise manipulation de données. On combine ainsi la possibilité d'effectuer les modifications dans une seule transaction avec celle de ne pas annuler toutes les modifications. Certaines d'entre elles prennent en effet un temps considérable.

# La sécurité des données

## Triggers

### ● Triggers

un trigger est un ensemble d'instructions SQL, définies par l'utilisateur, qui seront déclenchées lors d'une action d'ajout, de suppression ou de mise à jour de données. Il s'applique donc à un objet de type table. On peut choisir d'exécuter un trigger avant ou après une instruction de mise à jour.

Les données manipulées par le trigger sont stockées dans des tables temporaires que l'on désigne dans le code du trigger par les termes:

- NEW valeurs, après l'exécution de l'opération de mise à jour ;
- OLD valeurs, avant l'exécution de l'opération de mise à jour.

### ➤ Syntaxe générale pour la création et la destruction d'un trigger

L'instruction, normalisée par la norme SQL, est de la forme générale suivante :

```
CREATE TRIGGER 'Nom du trigger'  
'Moment où le trigger est exécuté'  
'Opération concernée'  
ON 'Nom de la table'  
FOR EACH ROW(ou STATEMENT)  
BEGIN  
  'Instructions'  
END
```

- Le moment peut prendre les valeurs BEFORE ou AFTER.
- Les opérations concernées sont INSERT, UPDATE et DELETE.
- La différence entre ROW et STATEMENT est que l'on exécute les instructions pour chaque ligne ou pour toute la table.

Pour détruire un trigger, on utilise l'instruction DROP et on spécifie le nom du trigger que l'on veut détruire.

```
DROP TRIGGER conversion
```

# La sécurité des données

## Triggers

### ➤ cas d'utilisation

- Contrôle de la forme du contenu des champs

Pour le formatage, on veut imposer que les noms d'une ville dans la table 'client' soient transformés en majuscules avant d'être insérés. On fait référence ici à la nouvelle table 'NEW' que l'on modifie avant l'insertion.

```
CREATE TRIGGER ville_majuscule BEFORE INSERT ON client FOR EACH ROW
BEGIN
SET NEW.Ville=UPPER(NEW.Ville) ;
END
```

- Modification automatique du contenu d'un champ

Les triggers permettent, par exemple, de compléter le mécanisme d'intégrité référentielle par des contrôles plus fins et la définition des actions associées à ces derniers. On considère l'exemple de la base de données 'casse'. Si le prix de vente est supérieur à 40 000 lors d'une insertion de données dans la table 'vente', on suppose que la personne a saisi par erreur le prix en francs et non pas en euros. On effectue automatiquement la conversion des francs en euros. Là encore, on modifie les données de la table 'NEW' qui contient celles que l'on va insérer.

```
CREATE TRIGGER conversion BEFORE INSERT ON vente FOR EACH ROW
BEGIN
IF New.Prix> 40000
THEN SET
NEW.Prix=NEW.Prix/6.55 ;
END IF;
END
```

# La sécurité des données

## Triggers

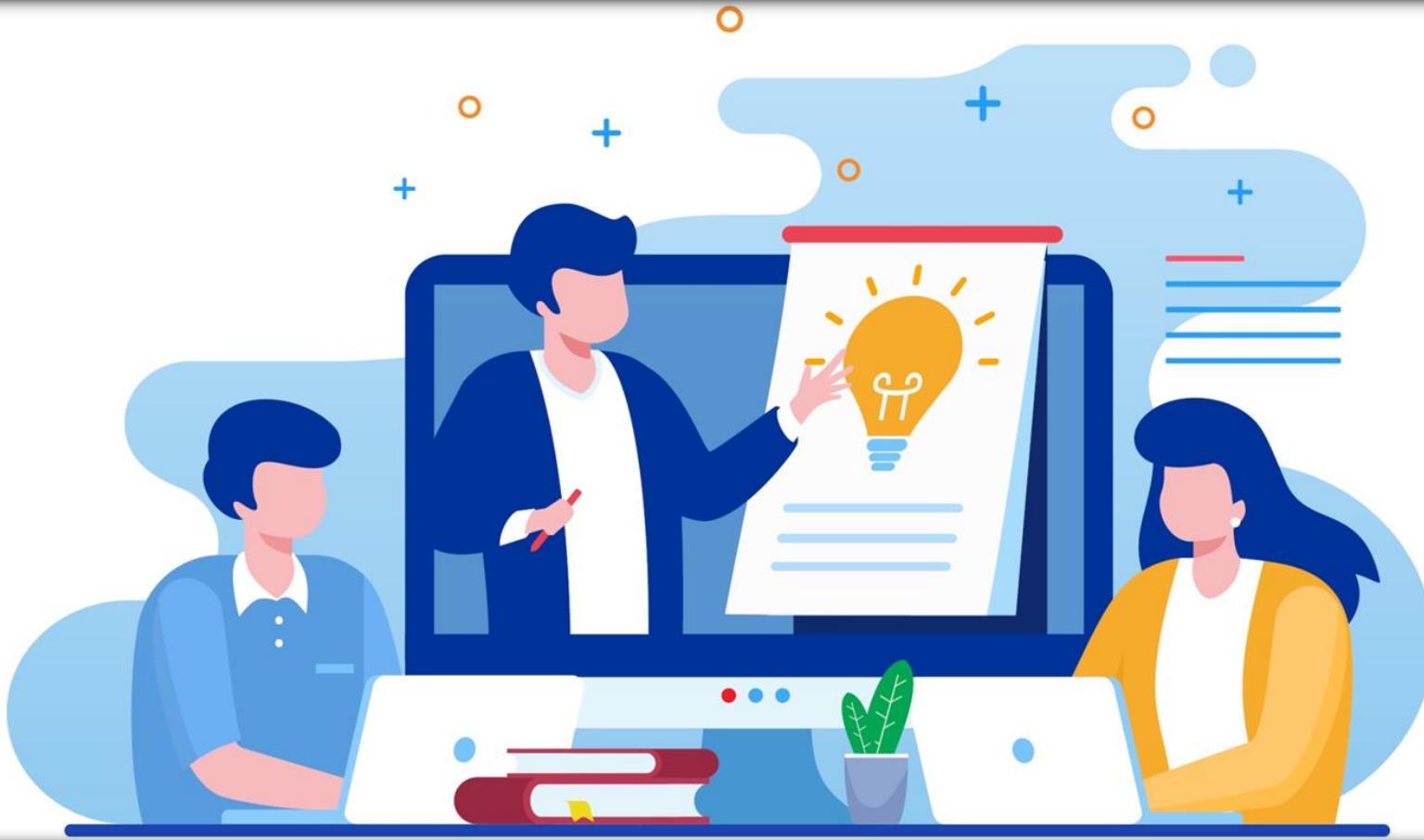
- **Mise à jour d'autres tables**

On peut également provoquer la mise à jour d'autres tables à l'aide d'un trigger. Si l'on ajoute un champ 'compte' à la table 'client' qui représente l'état du solde de son compte vis- à-vis de l'entreprise, on veut pouvoir mettre à jour automatiquement le champ 'compte' de l'acheteur lors de l'insertion d'une opération de vente. À cette occasion, on soustrait le prix de vente de la voiture du montant du compte du client. C'est la seule manière de réaliser cette opération de modification d'une autre table lors d'une simple insertion.

```
# Ajout d'un champ 'compte' à la table 'client'  
ALTER TABLE client ADD COLUMN compte INT ;  
  
# Création du trigger  
CREATE TRIGGER compte BEFORE INSERT ON vente FOR EACH ROW  
BEGIN  
    UPDATE client  
    SET client.Compte=client.Compte-NEW.Prix  
    WHERE client.NumAch=New.NumAch ;  
END  
  
# Ajout d'une vente dans la table 'vente'  
INSERT INTO vente VALUES (6, '2005-03-01',50000,2,5)
```

Le compte du client de numéro 'NumAch' égal à '2' sera automatiquement diminué de la somme de '50000'.

Du point de vue de la sécurité des données, le code associé au trigger s'exécute sur le serveur sans interaction avec le client. On réduit ainsi les risques d'erreur liés aux échanges entre le client et le serveur.



**Feedback**

pape.abdoulaye.barro@gmail.com