

Master Biologie-Informatique

Conception d'une Base de Données

Janvier 2017

Costas Bouyioukos Maître de conférences Paris Diderot Paris 7
(costas.bouyioukos@univ-paris-diderot.fr)

Romain Coppée Doctorant Université Paris Descartes – Enseignant Paris Diderot Paris 7
(romain.coppee@univ-paris-diderot.fr ou romain.coppee@hotmail.fr)

Objectifs du cours

- Comprendre ce qu'est une **base de données relationnelle** (BDDR) et un **système de gestion de base de données relationnelle** (SGBDR)
- Apprendre à **concevoir et modéliser** une BDDR
- Apprendre à **utiliser** un SGBDR
- Apprendre à utiliser une **API Python** pour utiliser un SGBDR
- Apprendre à construire une **interface d'interrogation web** d'un SGBDR en Python

Préambule

Le cours n'est pas exhaustif : tous les aspects ne seront pas traités et certains concepts seront passés sous silence par souci de simplification

Plan de la formation

- **Objectifs**

- **Méthodologie :**

- Cours
- Pratique

- **10h-12h30**

- **14h-16h**

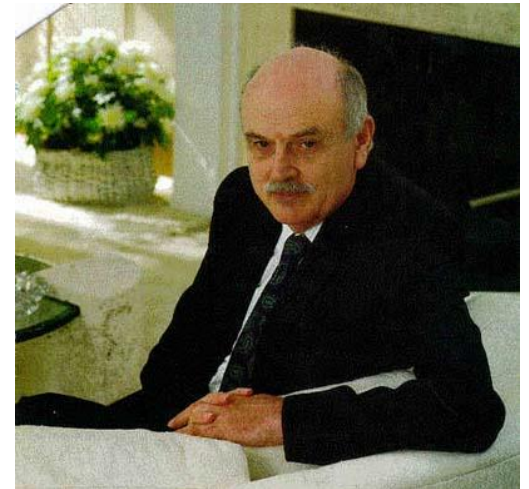
- **Calendrier prévisionnel**

- **Lundi** : Modélisation (MCD, MLD), MySQL et premières commandes
- **Mardi** : Rappels, SQL avancé
- **Mercredi** : Rappels, SQL avancé, HTML, CSS, Python::CGI
- **Jeudi** : Rappels, Python::CGI, cas pratiques (séance libre)

Concepts Généraux

Bref historique

- Développé par **E.F. Codd** (IBM) 1969-70
 - ▣ Décédé en 2003
- Récompensé par le **prix Turing**
- **12 règles** définissent une base de données relationnelle permettant d'aboutir à un langage pour créer, manipuler et rechercher les données dans les bases
- La première règle a permis l'émergence du **Structured Query Language (SQL)** qui est utilisé dans tous les SGBDR sur le marché



Qu'est ce qu'un SGBD ?

- Système de Gestion de Base de données
 - ▣ DBMS (DataBase Managment System)
- Exemples :
 - ▣ Oracle
 - ▣ MySQL
 - ▣ PostgreSQL
 - ▣ MS SQL
 - ▣ Sybase
 - ▣ MS Access
 - ▣ ...



ORACLE®



La modélisation, pourquoi ?

- **Identifier les concepts fondamentaux**
 - Que vont représenter les données de la base ?
 - Découvrir les concepts élémentaires
 - Décrire les concepts agrégés et les sous-concepts
- **Visualisation du système**
 - Diagrammes avec notations simple et précise
 - Compréhension visuelle et non seulement intellectuelle
- **Aider à la maintenance**

Avantages du modèle relationnel

- ❑ **Indépendance des données** par rapport aux traitements
 - ❑ Protège les données des applications
- ❑ **Efficacité**
 - ❑ Stockage (non redondance), recherche (index)
- ❑ **Intégrité/sécurité** des données
 - ❑ Contraintes, contrôle d'accès
- ❑ **Robustesse, fiabilité** (utilisé depuis de nombreuses années)

Comment modéliser ?

- Une **table** est une « **relation** » faite de **colonnes** et de **lignes**
- Les **lignes** sont des **enregistrements**
- Les **colonnes** (champs) sont les **attributs** des enregistrements (entités)
- **L'Intersection entre une colonne et une ligne est une valeur**
 - ▣ Domaine : Entier (integer), caractère de taille variable(VarChar), Texte (text), etc...
- La **puissance** des bases de données est basée sur les relations qui sont construites entre les tables

Exemple : « Recettes de cuisine »

- Choisir **le ou les éléments centraux pour créer les tables associées**
 - ▣ La recette de cuisine proprement dite
- **Eviter la redondance** d'information
 - ▣ Beaucoup de recettes possèdent des ingrédients en commun
 - ▣ Une table à part contient les ingrédients et est reliée à la table principale avec l'information sur la quantité
- **Rendre l'interrogation rapide et simple**
 - ▣ Un critère de recherche fréquent sera le type de plat (entrée, dessert...)
 - ▣ En créant une table « Type » liée à la table « Recette », les liens entre les tables peuvent être exploités plus rapidement

Quelques caractéristiques (1)

- Les **lignes** (enregistrements) ne sont pas dans un ordre particulier
- **Colonnes (champs) sont ordonnées**, numérotées et nommées
- Chaque ligne est identifiée par une **clé primaire unique** – assurant le caractère d'unicité de chaque ligne et qu'une ligne ne peut être vide

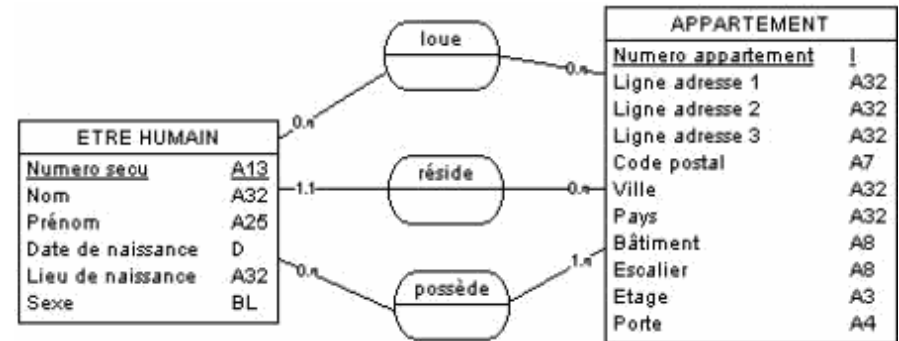
Quelques caractéristiques (2)



- Une « **vue** » est un sous-type de la base qu'un utilisateur ou une application peut utiliser
- Le **diagramme** d'une base de données est la structure de toute la base
- Une **contrainte est une condition** qui s'applique à un attribut de la table

Conception et modélisation

- Utilisation de **méthodes de modélisation** rigoureuses et rationnelles
- **Exemples de méthodes**
 - ▣ Merise
 - ▣ Entité Relation (ER) ou Entité Association (EA)
 - ▣ Rational Unified Process (UML)



Méthodes de modélisation

Modélisation Merise

Étapes de la méthode Merise

- Découpage du processus de développement en **quatre étapes** :
 - ▣ Étude préalable
 - ▣ Étude détaillée
 - ▣ Réalisation
 - ▣ Mise en œuvre

Etude préalable

- **Recueil des informations**
- Diagramme des flux
- **Elaboration** des **MOT** (*modèle organisationnel des traitements*) et **MCD** (*modèle conceptuel des données*) actuels
- **Synthèse et bilan** (services rendus, analyse des insuffisances, synthèse des besoins d'amélioration)
- **Choix MCD et MCT** (*modèle conceptuel des traitements*) nouvelle solution
- **Evaluation** nouvelle solution

Etude détaillée (1)

- **MCD, MCT, MLD** (*modèle logique des données*) et **MOT** (*modèle organisationnel des traitements*)
- Conception générale :
 - Description des MCD, MCT, MLD, MOT. **Définir l'environnement de développement** (matériel et logiciel).
 - Mise en œuvre du **dictionnaire des données**.
 - **Etude préliminaire de la mise en œuvre** (ébauche : plan de formation, documentation, plan de réception, démarrage, initialisation des données).
 - Etude des **solutions dégradées**

Etude détaillée (2)

- Conception détaillée :
 - Spécification détaillée des **phases « temps réel » et « temps différé »**
 - **Validation données-traitements** (optimisation du MLD, ébauche du MPD (*modèle physique des données*))
 - **Evaluation de la charge de réalisation** (durée), évaluation de la **mise en œuvre, plan d'équipement** matériels-logiciels.

Réalisation

- **Etude technique :**

- Description de l'environnement technique
- Description de l'architecture du logiciel
- Description du modèle physique des données

- **Production du logiciel :**

- Ecriture du logiciel (en appliquant les méthodes du génie logiciel)
- Tests unitaires (par unité de traitement) et d'intégration

Mise en œuvre

□ **Mise en place des moyens :**

- Moyens techniques (locaux, matériel informatique, fournitures...)
- Moyens humains (formation des utilisateurs, mise en place des fichiers et de la documentation)

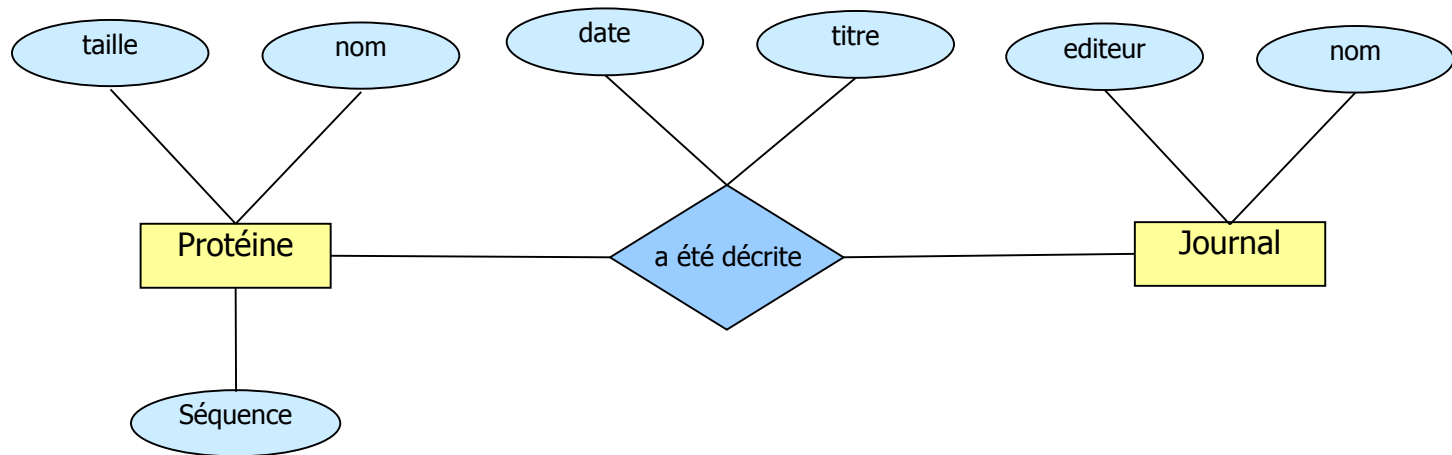
□ **Réception et lancement :**

- Exécution des jeux d'essais utilisateurs
- Conformité avec le dossier d'étude détaillée
- Lancement du nouveau système en vraie grandeur (en parallèle avec le système existant) pendant une période d'observation
- Arrêt de l'ancien système

Modélisation Entité Relation

Présentation du modèle

Le modèle EA (Entité-Association) ou ER (Entité-Relation) fournit un outil pour analyser les situation du monde réel pour créer un **modèle conceptuel des données** (MCD)



Etapes de la modélisation ER

- **Dictionnaire** et catalogue des données
- **Filtration**
- **Détermination des entités**
 - Attributs particulier : les identifiants
- Détermination des **attributs** (propriétés)
- Détermination des **relations** (associations)
- Détermination des **cardinalités**

Construction du dictionnaire de données

Abréviation	Description	Type de donnée	Contrainte d'intégrité
Ac	Numéro accession SwissProt	Alphanumériques	
Family	Famille de la protéine	Texte	
Descriptor	Description	Texte	
Source	Source (organisme ou synthétique)	Texte	
tissue	Provenance tissulaire	Texte	
sw_code	Code SwissProt	Alphanumérique	
Keywords	Mots clés	Texte	
Pubmed_id	Numéro d'identifiant Pubmed	Alphanumérique	
Authors	Auteurs	Texte	
pdb_code	Code PDB	Caractères	Taille fixe
Rms	RMSD par rapport à structure de référence	Réel	
experimental	Méthode expérimentale utilisée	Caractères	2 choix RX ou RMN
chain	Nom de la chaîne	Caractères	Taille fixe de 1 caractère
Lenght	Taille de la séquence	Entier	
resolution	Résolution	Réel	Existe seulement si méthode = RX
Date	Date de la découverte	Date	Format jj/mm/aaaa
Sequence	Séquence de la protéine code 1 lettre	Caractères	

Filtration des données

- **Suppression des données incohérentes**
- **Suppression des doublons**
- **Regroupement de certaines données**

Détermination des entités

□ **Entité :**

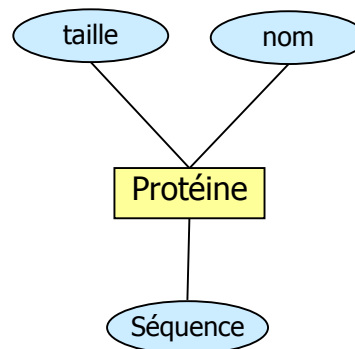
- Une entité est un **objet concret ou abstrait** qui peut être reconnu distinctement
- Exemple d'entité : Protéine, espèce

Protéine

Détermination des attributs

□ **Attributs:**

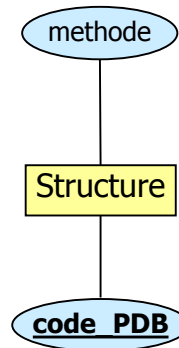
- Un **attribut** (ou une propriété) est une **caractéristique** associée à une entité
- Chaque attribut possède un domaine qui définit **le type de valeurs possibles** qui peuvent être choisies pour le décrire (entier, texte, réel...)
- Un attribut ne peut en aucun cas être partagé par plusieurs entités ou relations (= redondance)
- Exemples d'attributs : Le nombre d'acides aminés d'une protéine, la référence de publication d'une structure de protéine, le code PDB d'une structure



Attributs particuliers : les identifiants

□ *Identifiant :*

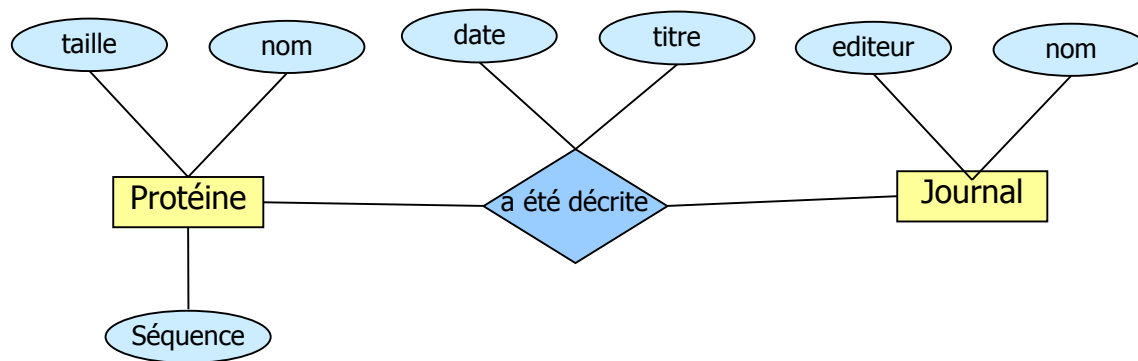
- Un identifiant (ou clé) d'une entité ou d'une relation est constitué par un (identifiant mono-composant) ou plusieurs (identifiant multi-composant) de ses attributs qui doivent avoir **une valeur unique pour chaque entité ou relation**



Détermination des relations

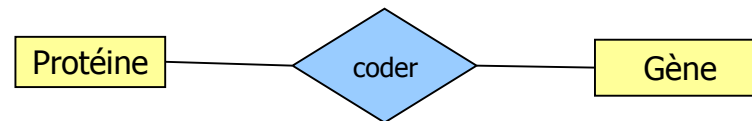
□ **Relation :**

- **Une relation** (ou association) est un **lien entre plusieurs entités (E)**
- Comme les entités, les relations sont définies à l'aide d'attributs qui prennent leur valeur dans les relations
- Exemple de relations : le journal où **a été décrite** une protéine, le type d'organisme duquel **est issue** une protéine => La protéine maurocalcine **a été décrite** dans "FEBS Letters", la maurocalcine **est issue** d'un scorpion.



Relations et cardinalités

□ 1:1



□ 1:N

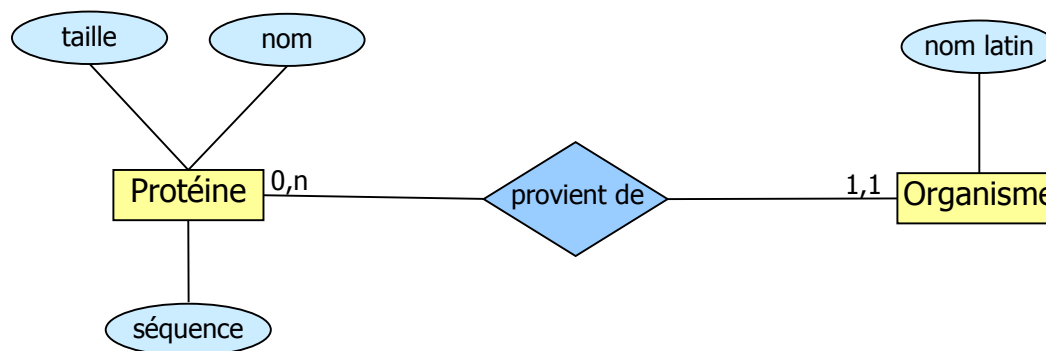


□ N : N

Détermination des cardinalités

□ **Cardinalité :**

- La cardinalité (i,j) d'une relation est le nombre de fois minimal (i) et maximal (j) qu'une entité peut intervenir dans une association (A).
- La cardinalité minimale doit être inférieure ou égale à la cardinalité maximale.
- Exemple de cardinalité : Une protéine provient de 1 organisme et un organisme possède 1 à n protéines.

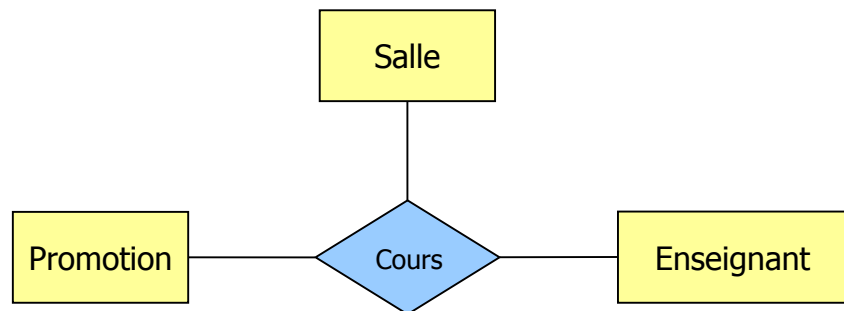


Propriétés des cardinalités

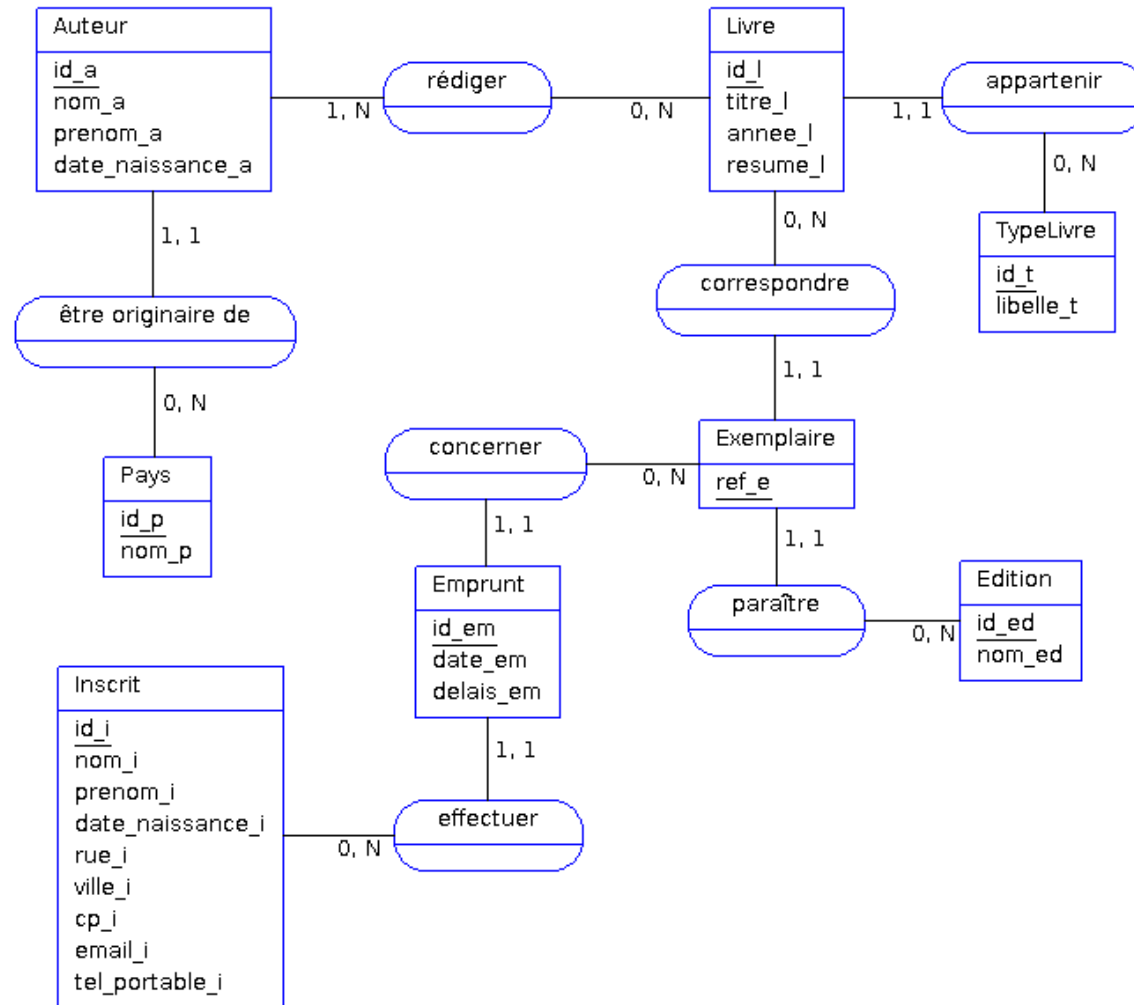
- L'expression de la cardinalité est **obligatoire** pour chaque relation
- La cardinalité minimale peut être :
 - ▣ 0 : Une entité peut exister tout en étant impliquée dans aucune relation
 - ▣ 1 : Une entité ne peut exister que si elle est impliquée dans au moins une relation
 - ▣ N : Une entité ne peut exister que si elle est impliquée dans plusieurs relations
- La cardinalité maximale peut être:
 - ▣ 1 : Une entité ne peut être impliquée au maximum que dans une relation
 - ▣ N : Une entité peut être impliquée dans plusieurs relations

Notions de dimensions

- La **dimension** (ou degré) d'une relation est le nombre d'entités qui y participent
- Si une entité participe plusieurs fois à la relation, il est compté autant de fois que de participations
- Le cas le plus fréquent d'une relation est le **degré 2** (ou binaire).
- Exemple degré 3 :

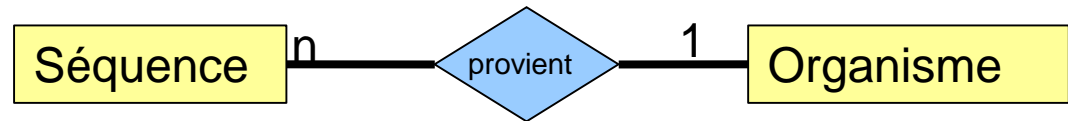


Exemple de diagramme ER



Représentation graphique

□ Chen



□ Martin / IE / Crow's foot



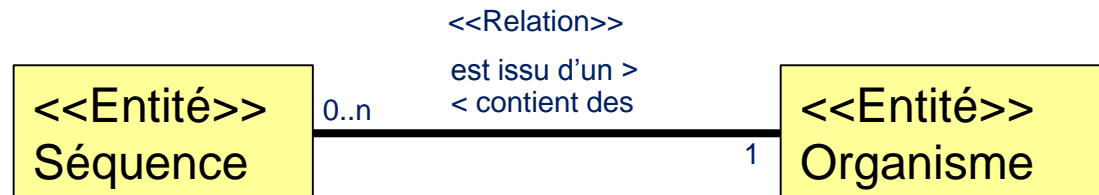
□ Bachman



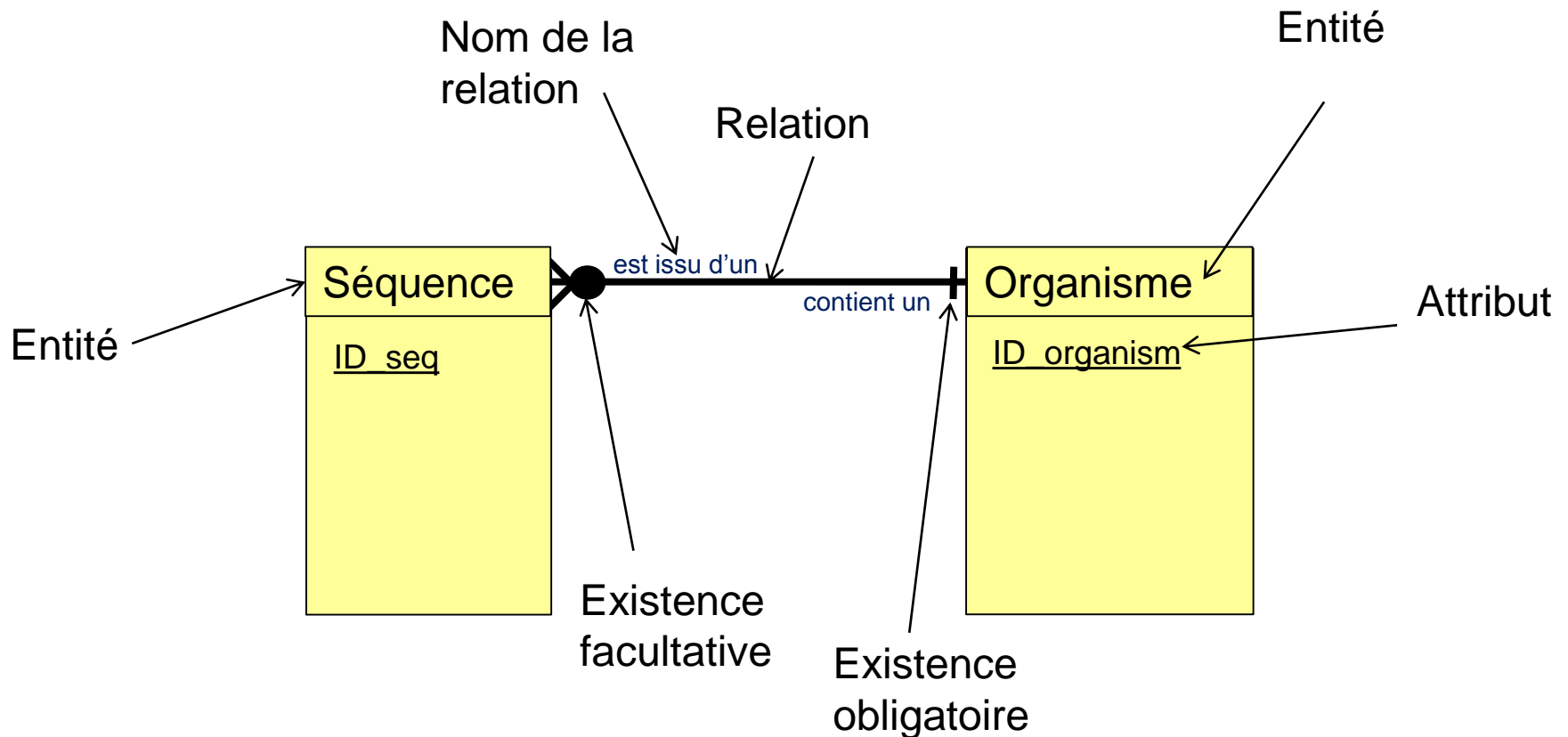
□ Min-max / ISO



□ UML



Représentation détaillée



Concepts avancés

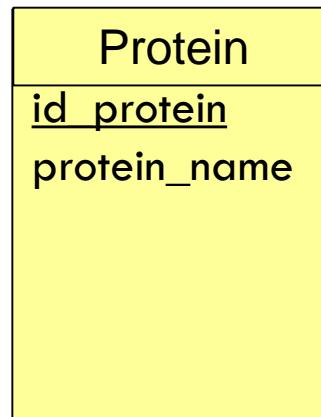
- **Dépendances fonctionnelles**
- **Formes Normales**
- **Entités Fortes/Faibles**

Notion de dépendance fonctionnelle

- Une **dépendance fonctionnelle** (DF) entre les attributs X et Y de la relation R, notée $X \rightarrow Y$, exprime que **la connaissance d'une valeur de X, quelle qu'elle soit, implique la connaissance d'une seule valeur de Y** (la réciproque n'est pas forcément vraie)
- Exemple : $\text{id_protein} \rightarrow \text{protein_name}$
 $\text{id_protein} \rightarrow \text{id_family}$
- X est appelé la « source » de la DF
- Y est appelé la « cible » de la DF

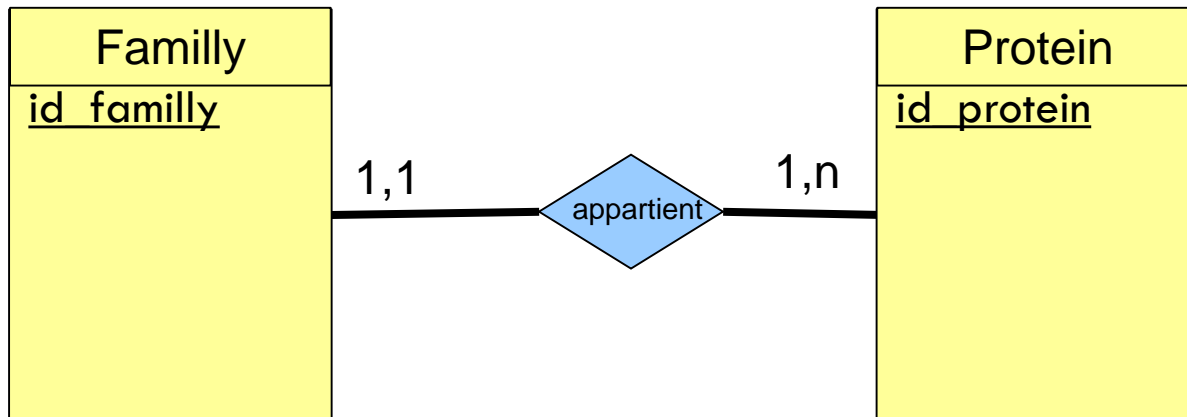
Exemple de DF (1)

- $\text{id_protein} \rightarrow \text{protein_name}$ (**au sein d'une même entité**)
 - ▣ Quelle que soit la protéine (Protein), aujourd'hui et demain, la connaissance d'un identifiant (id_protein) existant implique la connaissance d'un seul nom (protein_name).
 - ▣ Dans ce cas, la notion de DF est à rapprocher du lien qu'on peut établir entre l'identifiant d'une entité et ses propriétés



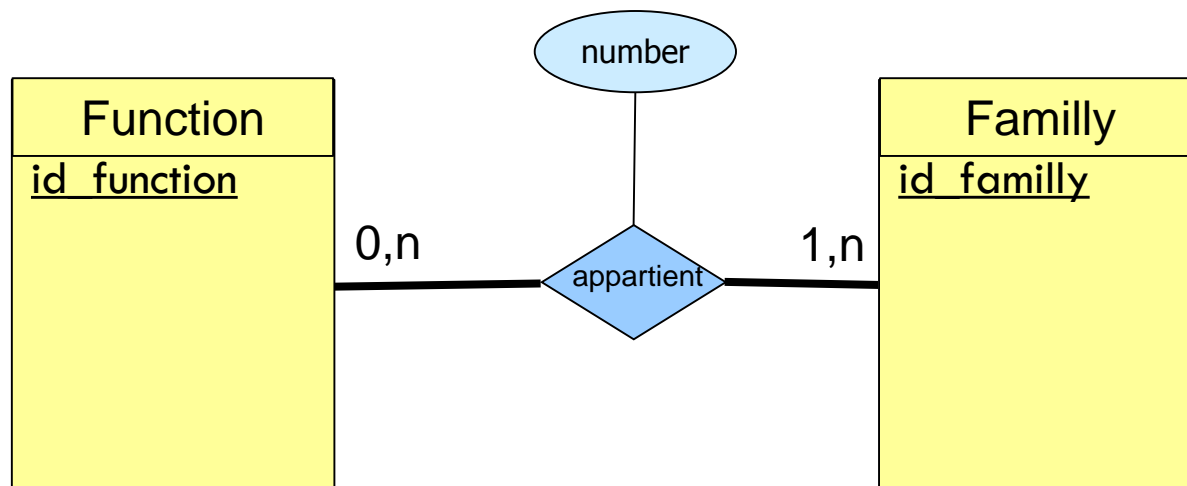
Exemple de DF (2)

- $\text{id_protein} \rightarrow \text{id_family}$ (**deux entités différentes**)
 - ▣ Quelle que soit la protéine (Protein), aujourd'hui et demain, la connaissance d'un identifiant (id_protein) implique la connaissance d'un seul identifiant (id_family) de famille (Family).
 - ▣ Dans ce cas, la notion de DF est à rapprocher du lien qu'on établit entre des entités.



Exemple de DF (3)

- $\text{id_function}, \text{id_family} \rightarrow \text{number}$ (**deux entités par rapport à une relation**)
 - ▣ La connaissance du nombre de fonctions (function) dans la famille (Family) dépend de la connaissance de l'identifiant de fonction (Function) et de l'identifiant de famille (family). ... 0,n avec 1,n ...
 - ▣ Dans ce cas, la notion de DF est à rapprocher du lien qu'on peut établir entre des entités liées et les propriétés de l'association correspondante.



Forme normale

- La **forme normale** désigne une **relation particulière** entre les entités
- En pratique, la première et la deuxième forme normale sont nécessaires pour avoir un modèle relationnel juste. Les formes normales supplémentaires ont leurs avantages et leurs inconvénients.
- Pour des petites bases de données, la troisième forme normale est généralement une des meilleures solutions d'un point de vue architecture de base de données, mais pour des bases de données plus importantes, cela n'est pas toujours le cas.
- Il s'agit de choisir **l'équilibre** entre deux options :
 - ▣ la génération dynamique des données via les jointures entre tables
 - ▣ l'utilisation statique de données correctement mises à jour

FN : Avantages / Inconvénients

- Les **avantages** sont :
 - ▣ de **limiter les redondances** de données
 - ▣ de **limiter les incohérences** de données qui pourrait les rendre inutilisables
 - ▣ d'éviter les processus de mise à jour

- Les **inconvénients** sont :
 - ▣ des **temps d'accès potentiellement plus longs** si les requêtes sont trop complexes
 - ▣ une plus **grande fragilité des données** étant donné la non redondance

1FN

□ 1FN - Première Forme Normale :

Respecte la première forme normale, la relation dont tous les attributs :

- contiennent une valeur atomique (les valeurs ne peuvent pas être divisées en plusieurs sous-valeurs dépendant également individuellement de la clé primaire)
- contiennent des valeurs non répétitives (le cas contraire consiste à mettre une liste dans un seul attribut).
- sont constants dans le temps (utiliser par exemple la date de naissance plutôt que l'âge).
- *Le non respect des deux premières conditions de la 1FN rend la recherche parmi les données plus lente parce qu'il faut analyser le contenu des champs. La troisième condition quant à elle évite qu'on doive régulièrement mettre à jour les données.*

□ exemple:

Cell	Protein
muscle	Alpha actin I, Myosin II, Purinoceptor I

solution:

Cell	Protein
muscle	Alpha actin I
muscle	Myosin II
muscle	Purinoceptor I

2FN

□ 2FN - Deuxième Forme Normale

Respecte la seconde forme normale, la relation respectant la première forme normale et dont :

- tous les attributs non-clés ne dépendent pas d'une partie de la clé primaire mais bien de la totalité de la clé primaire.
- *Le non respect de la 2FN entraine une redondance des données qui encombrent alors inutilement la mémoire et l'espace disque.*
- *exemple:*

Cell	Protein	Gene
myocyte	Purinoceptor I	P2RX1
lymphocyte	Purinoceptor I	P2RX1
myocyte	Alpha actin I	ACTA1

solution:

Cell	Protein
myocyte	Purinoceptor I
lymphocyte	Purinoceptor I
myocyte	Alpha actin I

Ici la clé peut être une clé composite (Protein-Gene). Si changement de Gene de la Protein (suite à une erreur d'annotation par exemple) il y a un problème car le champs Cell ne dépend que d'une partie de la clé ici Protein. Il y a redondance

Solution: la table est scindé en 2

Protein	Gene
Purinoceptor I	P2RX1
alpha actin I	ACTA1

3FN

□ **3FN - Troisième Forme Normale**

Respecte la troisième forme normale, la relation respectant la seconde forme normale et dont :

- Il n'y a aucune dépendance/relation entre attributs non-clés.
- En d'autre terme, la dépendance fonctionnelle est directe.
- *Le non respect de la 3FN peut également entraîner une redondance des données.*

Exemple de 3FN

exemple:

Protein	Gene	Chromosome	Species
Purinoceptor I	P2RX1	17/23	Human
Alpha actin I	ACTA1	8/20	Mouse

Les attributs doivent être indépendants les uns des autres (pas de DF).

Ici le chromosome n'est pas dépendant de la clé de la table (Protein) mais est fonction de l'espèce.

La solution est de scinder la table en deux

solution normalisée :

Protein	Gene	Chromosome	Chromosome	Species
Purinoceptor I	P2RX1	17/23	17/23	Human
Alpha actin I	ACTA1	8/20	8/20	Mouse

Aide mémoire

Première forme normale (1FN)

Deuxième forme normale (2FN)

Troisième forme normale (3FN)

Forme normale Boyce-Codd (FNBC)

Quatrième forme normale (4FN)

Cinquième forme normale (5FN)

Dépendance de jointure
triviale seulement

Pas de dépendances multivaluées

Seules les dépendances de clé sont admises

Il n'y a pas de dépendances transitives

Les attributs non clés dépendent de la clé entière dans leur table

Toutes les valeurs des attributs sont atomiques (pas de groupes répétitifs)

Tables non normalisées

Entités fortes / faibles

□ Entités fortes :

- **entité qui peut exister de manière isolée**, sans autre relation qu'avec son identifiant.
- exemple : Protéine (code PDB)

□ Entités faibles :

- entité possédant **une relation obligatoire vis à vis d'une autre entité**
- exemple : Publication (Auteur, Journal)

Applications et Exercices

Exercice 1

□ Dictionnaire de données Protéines

Objectif : Recenser tous les attributs permettant la recherche d'une protéine (via un numéro d'accension, code PDB, séquence, etc.)

Dictionnaire de données

Abréviation	Description	Type de donnée	Contrainte d'intégrité
Ac	Numéro accession SwissProt	Alphanumériques	
Family	Famille de la protéine	Texte	
Descriptor	Description	Texte	
Source	Source (organisme ou synthétique)	Texte	
tissue	Provenance tissulaire	Texte	
sw_code	Code SwissProt	Alphanumérique	
Keywords	Mots clés	Texte	
Pubmed_id	Numéro d'identifiant Pubmed	Alphanumérique	
Authors	Auteurs	Texte	
pdb_code	Code PDB	Caractères	Taille fixe
Rms	RMSD par rapport à structure de référence	Réel	
experimental	Méthode expérimentale utilisée	Caractères	2 choix RX ou RMN
chain	Nom de la chaîne	Caractères	Taille fixe de 1 caractère
Lenght	Taille de la séquence	Entier	
resolution	Résolution	Réel	Existe seulement si méthode = RX
Date	Date de la découverte	Date	Format jj/mm/aaaa
Sequence	Séquence de la protéine code 1 lettre	Caractères	

Exercice 2

- Dictionnaire de données et diagramme ER d'une base de données « Cinéma »

Résumé des informations

- ☐ Des acteurs tournent dans des films
- ☐ Un film a un titre
- ☐ Un film est projeté à une date et un horaire précis sur un écran d'une salle de cinéma
- ☐ Un acteur est caractérisé par son nom, sa photo, son âge et sa biographie
- ☐ Un film est tourné avec des acteurs et a une date de sortie
- ☐ Un écran de salle de cinéma est caractérisé par son numéro, sa taille, sa technologie (numérique, pellicule, numérique 3D actif, numérique 3D passif), le nombre de sièges et son accessibilité pour les handicapés
- ☐ Un cinéma a un nom, une adresse et un numéro de téléphone

Dictionnaire de données

Nom Acteur

Biographie d'un acteur

Photo de l'acteur

Titre de Film

Date de sortie

Nombre de sièges dans la salle de cinéma

Numéro de la salle de cinéma

Accessibilité de la salle

Taille Ecran de la salle

Technologie de la projection

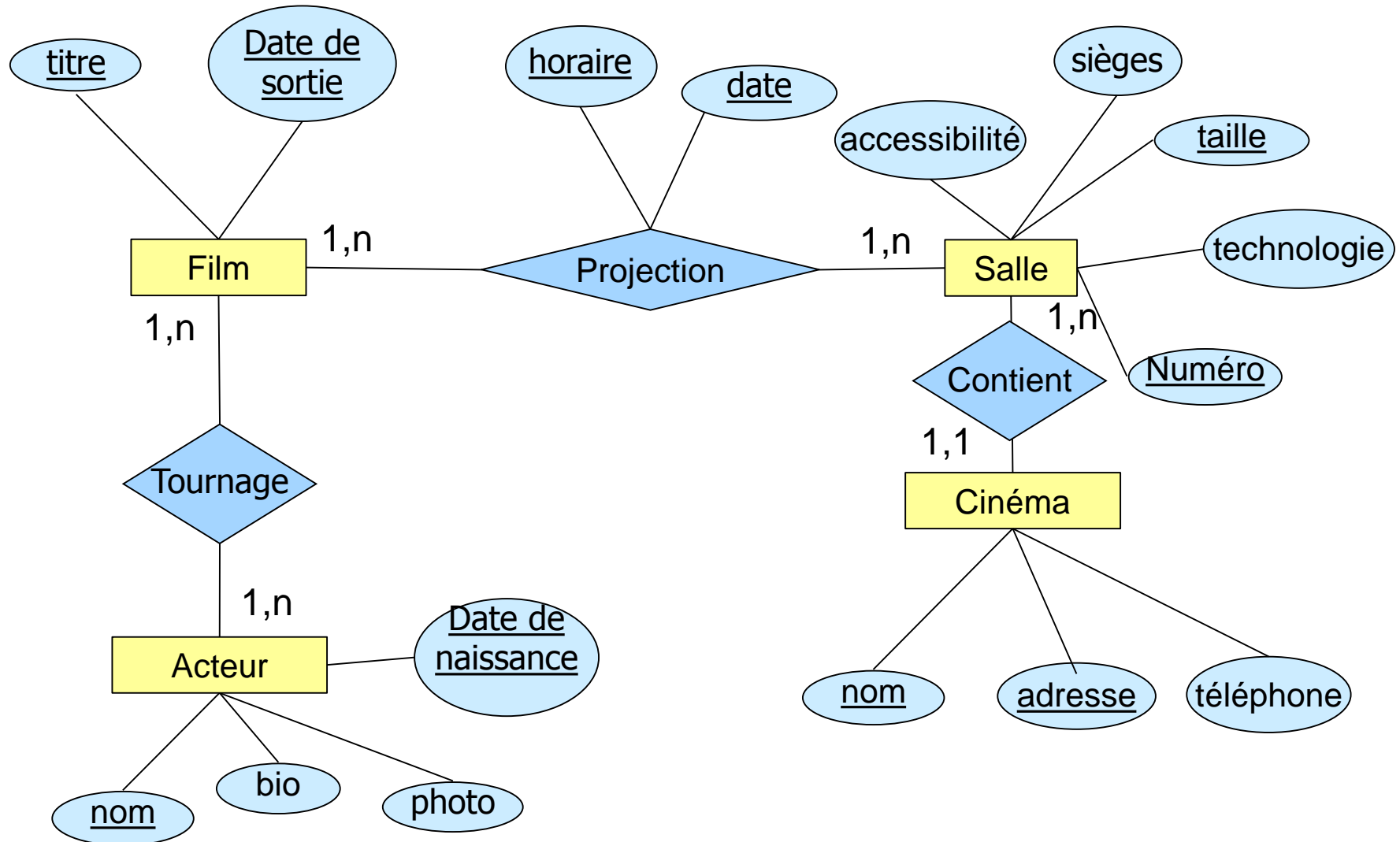
Nom du cinéma

Adresse du cinéma

Numéro de téléphone du cinéma

Tout n'a évidemment
pas été cité...

Diagramme ER



Exercice 3

- Diagramme ER d'un canal potassium d'après une publication

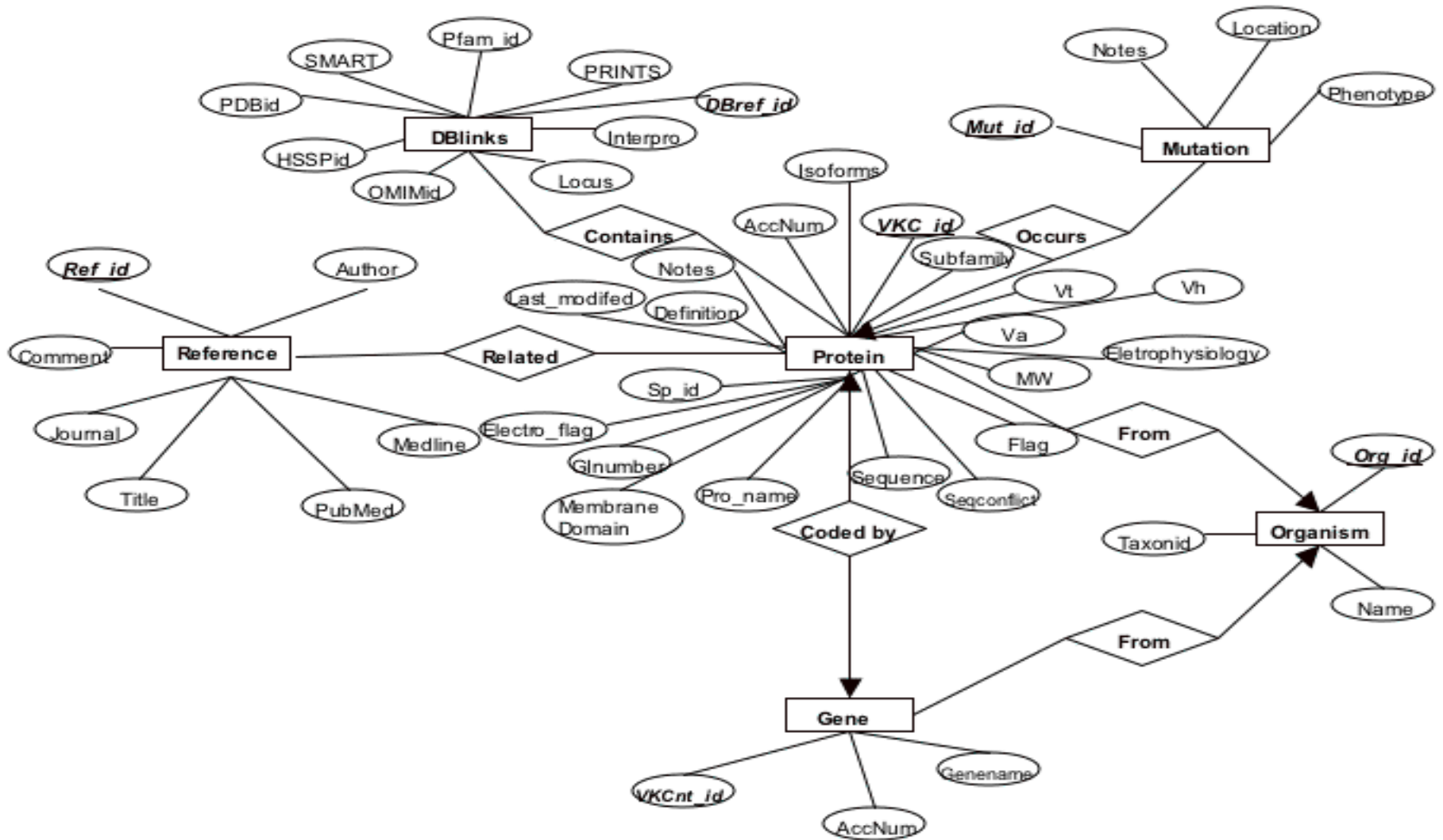
- Source :

<http://www.biomedcentral.com/1471-2105/5/3/abstract>

Dictionnaire des données

- ☐ protein name
- ☐ acc protein
- ☐ mw
- ☐ sequence
- ☐ membrane domain
- ☐ isoforms number
- ☐ subfamilly
- ☐ mutation name
- ☐ location
- ☐ phenotype
- ☐ organism
- ☐ gene
- ☐ acc gene
- ☐ Reference
- ☐ Blast alignment
- ☐ pfam link
- ☐ SMART link
- ☐ pdb link
- ☐ interpro link
- ☐ PRINTS link
- ☐ Blast hit
- ☐ Blast score
- ☐ HSSP link
- ☐ OMIM link

Solution proposée par l'auteur



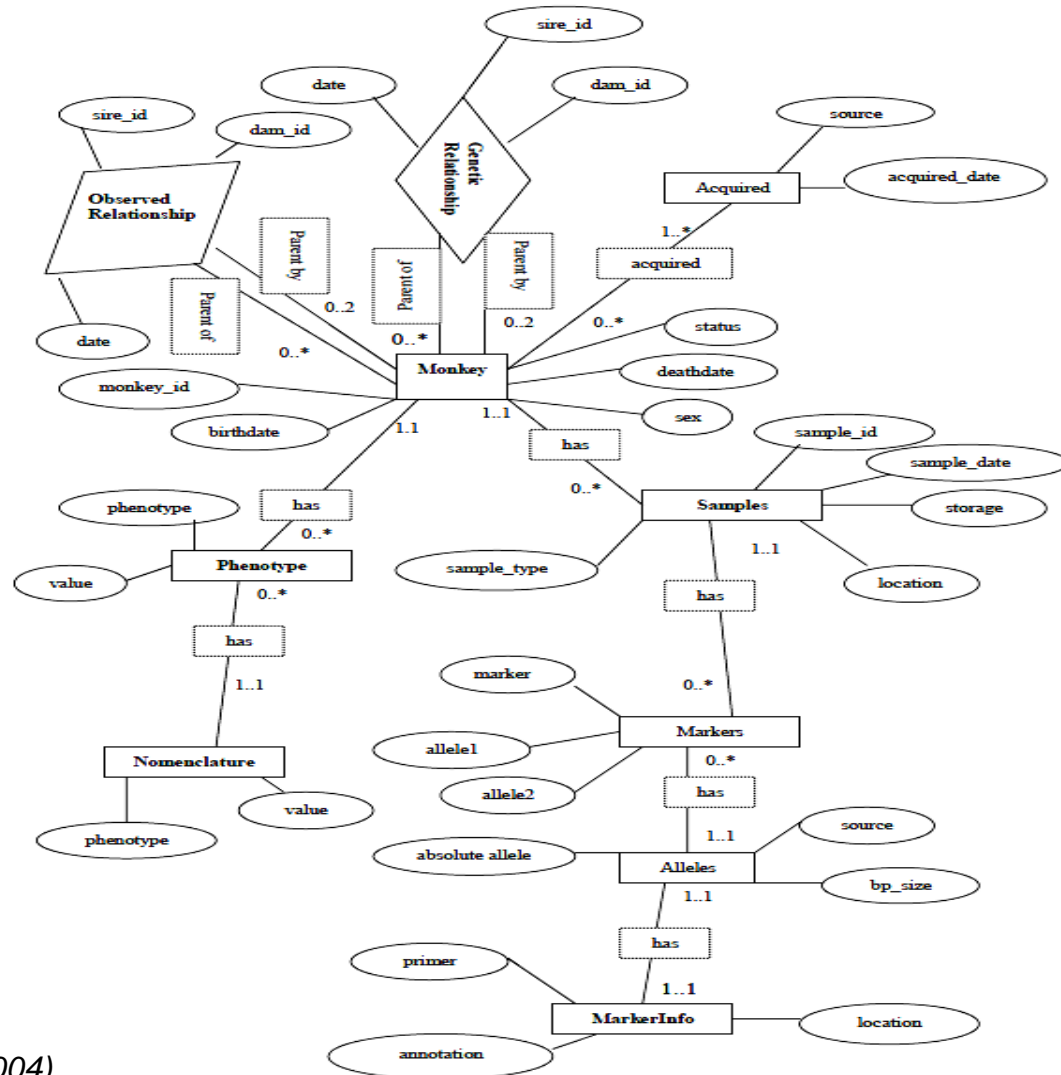
Exercice 4

- Diagramme ER de gestion d'une animalerie et de suivi des descendances
 - Informations sur les singes
 - Informations sur des caractéristiques physiques particulières (phénotype) et une nomenclature associée
 - Informations sur des prélèvements réalisés sur ces singes
 - Informations sur des marqueurs génétiques étudiés

Dictionnaire des données

- ☐ monkey name (nom)
- ☐ monkey sex (sexe)
- ☐ monkey age (âge)
- ☐ monkey source (établissement d'achat)
- ☐ monkey source date (date achat)
- ☐ monkey birthdate (naissance)
- ☐ monkey deathdate (mort)
- ☐ monkey samples (numéro échantillon)
- ☐ monkey size (taille du singe)
- ☐ monkey weight (poids du singe)
- ☐ phenotype date (date de la mesure)
- ☐ monkey eyes color (couleur des yeux du singe)
- ☐ monkey fur color (couleur de la fourrure)
- ☐ monkey phenotype nomenclature
- ☐ monkey parent (parents du singe)
- ☐ sample date (date du prélèvement)
- ☐ sample type (type de prélèvement)
- ☐ sample storage (stockage de l'échantillon)
- ☐ marker name (nom du marqueur)
- ☐ allele 1
- ☐ allele 2
- ☐ allele size (pb)
- ☐ allele name
- ☐ marker primer
- ☐ marker annotation
- ☐ marker location

Solution de l'auteur



(from Khouangsathiene, 2004)

Passage au modèle physique

Modèle logique des données (MLD)

- Une **entité** est représentée par une **table**
- Une table = un objet informatique regroupant tous les occurrences de l'entité
- Une **table** a une **forme de tableau** à l'écran
- Les colonnes ou champs : dans la table de l'entité, chaque colonne correspond à un attribut de l'entité

Propriétés du MLD

- Un attribut particulier est l' « **identifiant** » de chaque "individu" (occurrence), c'est-à-dire un numéro unique pour chaque individu. C'est la « **clé primaire** » de la table
- Les lignes : **chaque ligne correspond à une occurrence de l'entité**, ou « enregistrement ». Une occurrence est notée sur une seule ligne.

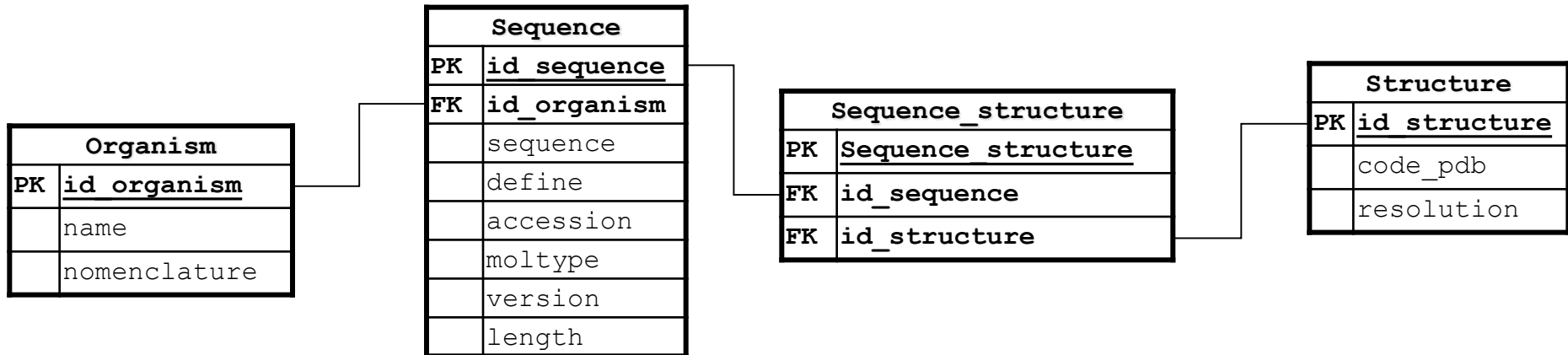
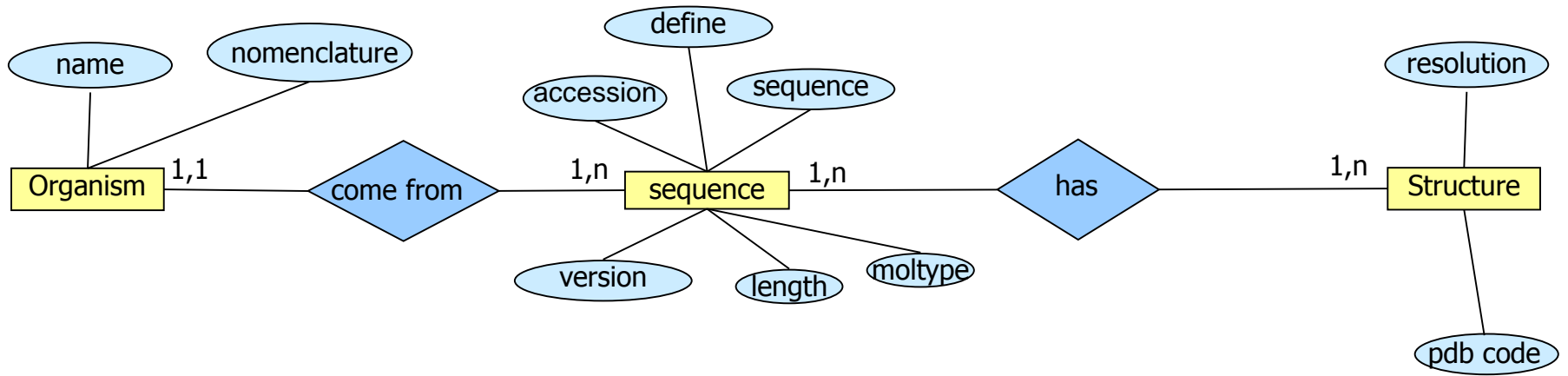
Règles de passage rapide (1)

- **Règles portant sur la transformation des entités :**
 - Pour chaque entité (identifié E), on crée une table. La clé de la table est une des clés de E. Les clés candidates n'étant souvent pas satisfaisantes, une clé est nouvellement créée qui servira de clé primaire.

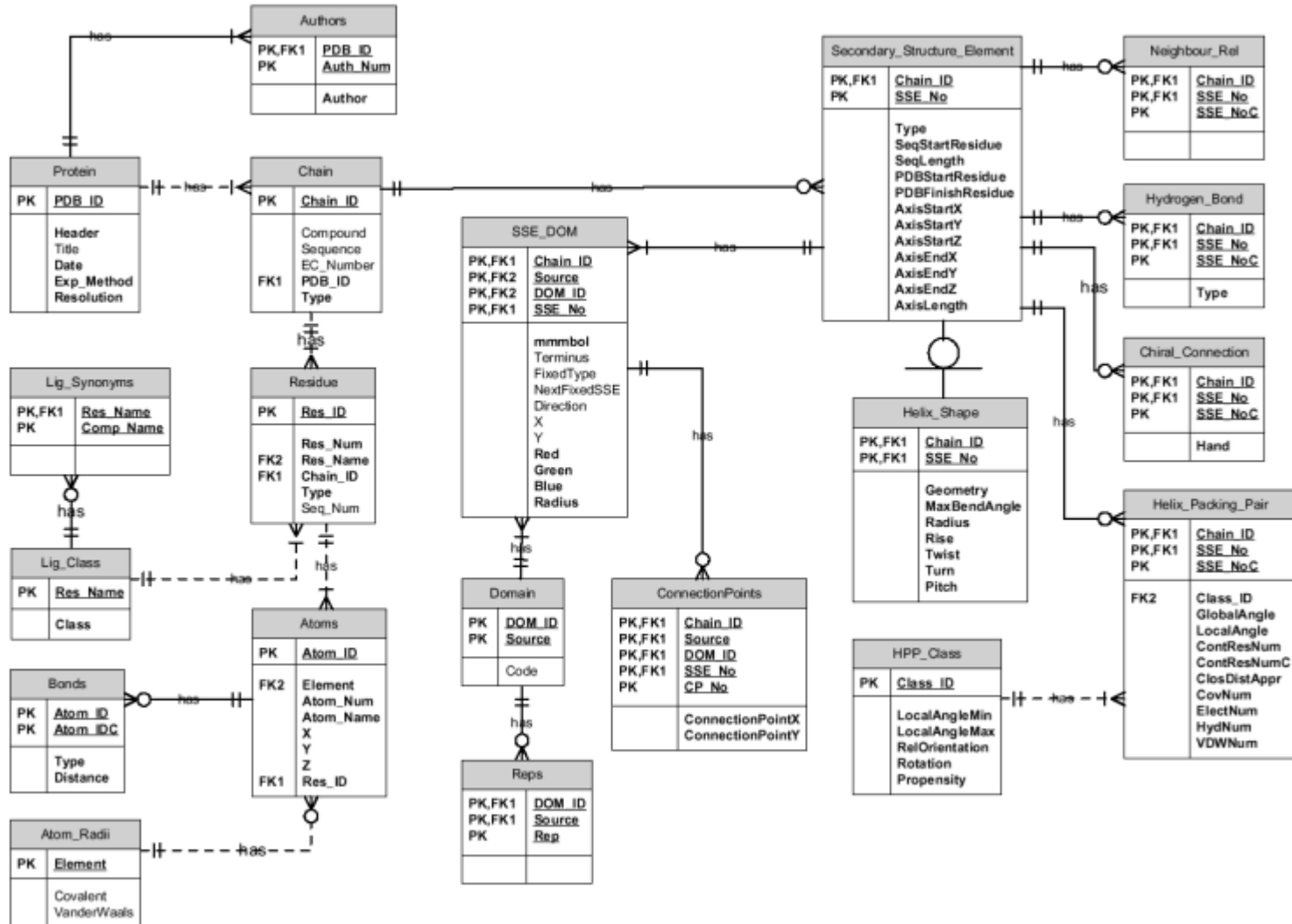
Règles de passage rapide (2)

- **Règles portant sur la transformation des associations :**
 - La clé obtenue se déduit de l'analyse des cardinalités de l'association (c'est au plus la concaténation des clés des entités participantes)
 - Pour chaque entité ayant une association binaire (dont la cardinalité n'est pas 1:N) avec une autre entité dépendant de la première entité, **une clé étrangère est créée pour la deuxième entité à partir de la clé primaire de la première entité**
 - Pour chaque association binaire de type 1:N, **une nouvelle table est créée pour représenter l'association**. Cette nouvelle table a comme clés étrangères, les clés de toutes les entités participant à l'association. La clé primaire de la nouvelle table est la concaténation des clés étrangères.

Example



Exemple mixte



Mise en Application

Base de données Film / Cinéma

Tableau de données

□ Film

- ▣ Titre
- ▣ Date de sortie
- ▣ Date de projection
- ▣ Horaire de projection

□ Salle

- ▣ Accessibilité
- ▣ Numéro
- ▣ Taille
- ▣ Nombre de sièges
- ▣ Technologie

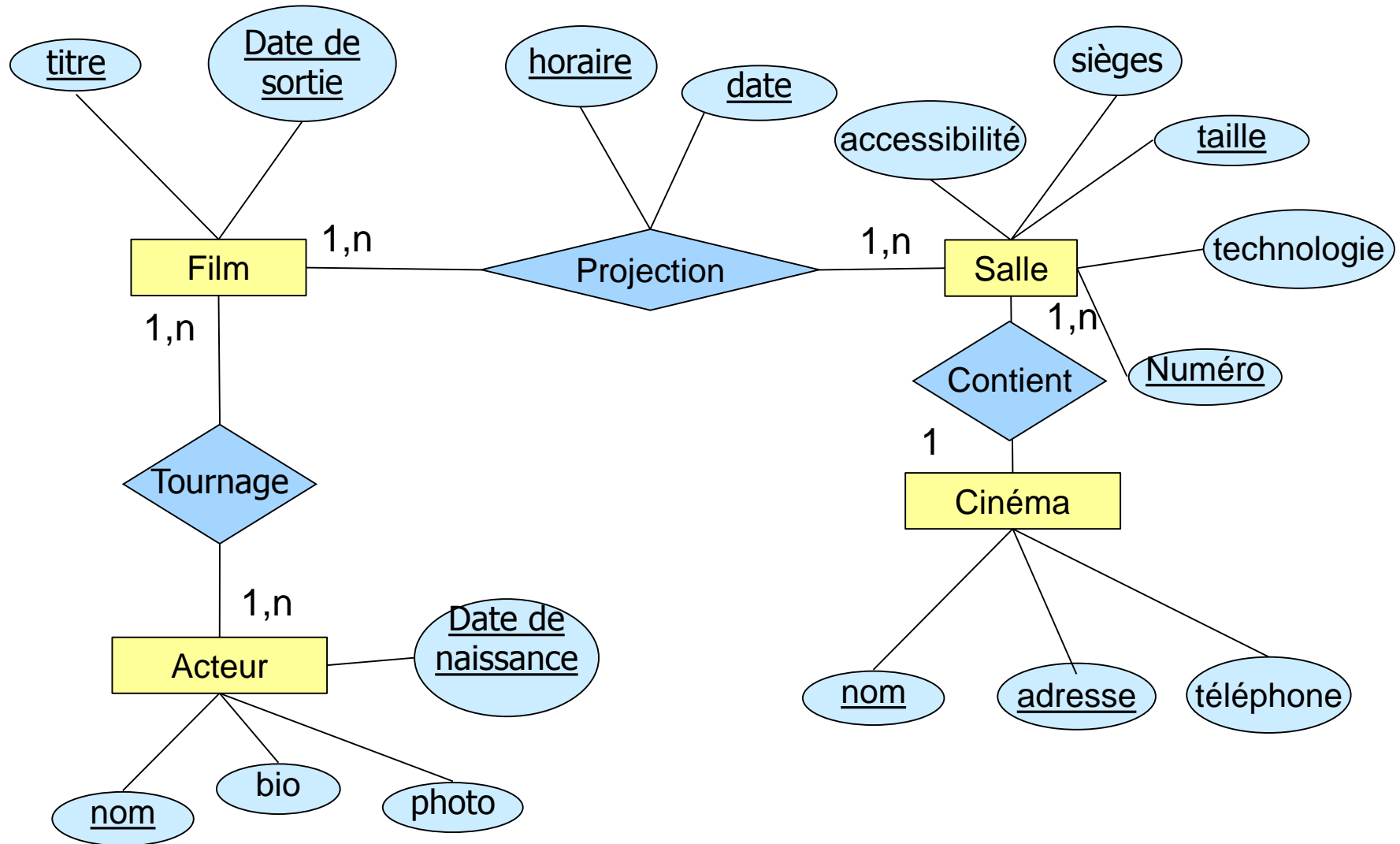
□ Acteur

- ▣ Nom
- ▣ Biographie
- ▣ Date de naissance
- ▣ Photo

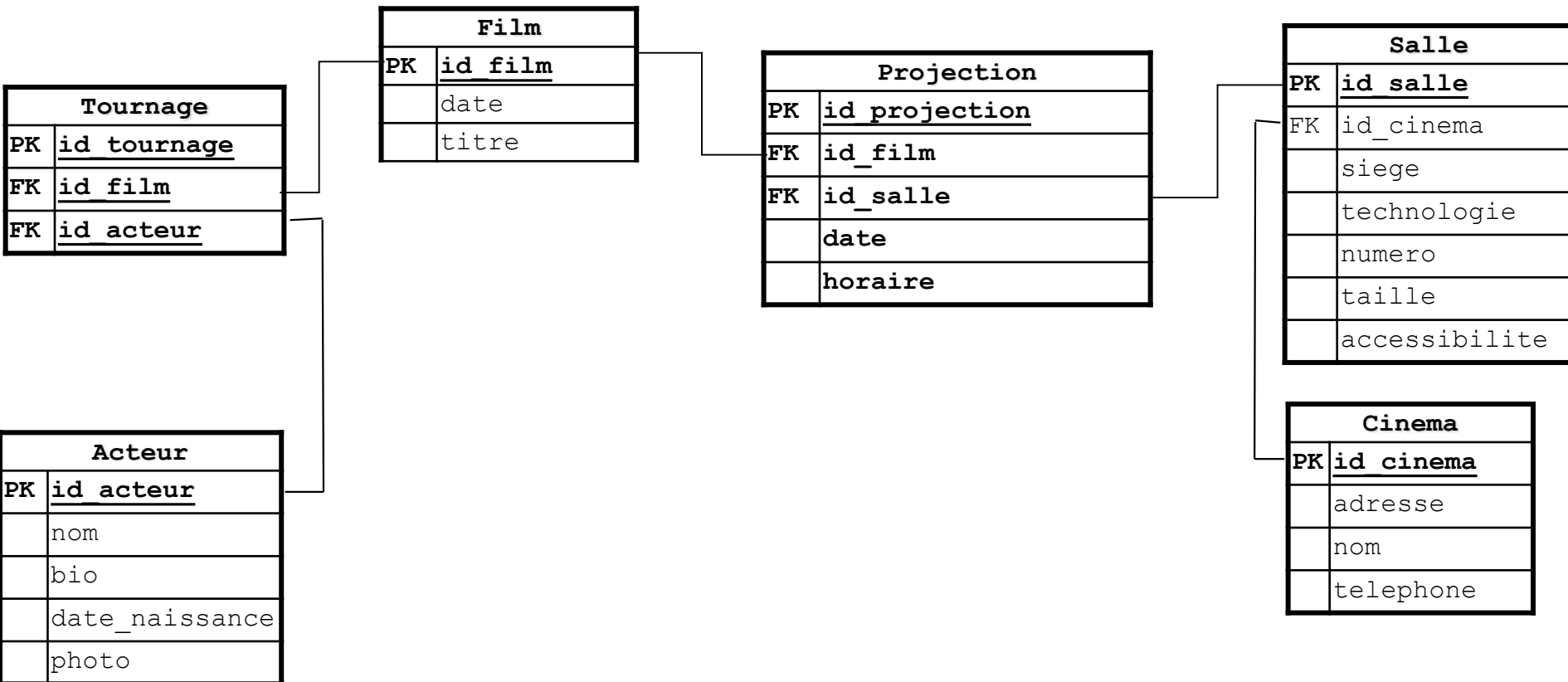
□ Cinéma

- ▣ Nom
- ▣ Adresse
- ▣ Téléphone

Diagramme ER



Modèle Relationnel Logique



Introduction au langage SQL

Présentation de MySQL

MySQL

- ❑ <http://www.mysql.com/>
- ❑ La base de données **la plus populaire** au monde
- ❑ Facile d'utilisation
- ❑ Très rapide pour la lecture (du fait de son architecture)
- ❑ Première version : 1995
- ❑ Open source, Licence GNU GPL ou propriétaire selon le type d'utilisation
- ❑ Appartient depuis 2009 à Oracle



Architecture de MySQL

- **Client / Serveur**

- Serveur : mysqld
- Client : mysql

- **Localisation des fichiers sur Ubuntu :**

- Bases de données : /var/lib/mysql/
- Fichiers de configuration : /etc/mysql/ => my.cnf
- Fichiers log : /var/log/mysql/ => error.log

Commandes Unix utiles pour MySQL

- ❑ **Exporter une base de données :**
`mysqldump -u user -p -r fichier.sql nom_database`
- ❑ **Export possible mais non conseillé :**
`mysqldump -u user -p -r fichier.sql -h ipserveur nom_database`
- ❑ **Exporter toutes les bases de données :**
`mysqldump -u user -p -r file.sql --all-database --skip-lock-tables`
- ❑ **Exporter une table d'une base de données :**
`mysqldump -u user -p -B nom_database --tables tables > fichier.sql`
- ❑ **Importer une base qui a été exportée avec mysqldump**
`mysql -u user -p nom_database < backup-file.sql`
- ❑ **Pour se connecter à mysql :**
`mysql -u user@machine -p`
- ❑ **Pour vérifier l'intégrité d'une base :**
`mysqlcheck -u user -p nom_database`

Commandes MySQL

- ❑ **Pour visualiser les bases de données** accessible par l'utilisateur :
`show databases;`
- ❑ **Pour se « connecter » à une base de données :**
`use nom_database;`
- ❑ **Pour visualiser les tables :**
`show tables;`
- ❑ **Pour visualiser la structure d'une table :**
`show columns from nom_table;`
`desc nom_table;`
- ❑ **Créer et donner tous les droits à un utilisateur sur une base :**
`grant all privileges on nom_database.nom_table to login@'ip'`
`identified by 'password' with grant option;`
`grant all privileges on *.* to login@'%' identified by 'password' with`
`grant option;`
- ❑ **Supprimer les droits à un utilisateur sur une base :**
`revoke all privileges on nom_database from user;`

Langage SQL

« ou comment passer de la théorie à la pratique »

Prérequis

- **Se connecter au serveur qui héberge le serveur MySQL :**

```
ssh LOGIN@glycine3
```

- **Se connecter au serveur MySQL :**

```
mysql -u LOGIN -p
```

Création d'une base

- **Commande MySQL :**

- ▣ `CREATE DATABASE nom_database;`

- **Exemple :**

```
mysql>  
mysql> CREATE DATABASE formation;  
Query OK, 1 row affected (0.00 sec)
```

```
mysql>  
mysql>  
mysql>  
mysql> █
```

Voir les bases & tables accessibles

□ **Commande MySQL :**

SHOW DATABASES;

SHOW TABLES;

□ **Exemples :**

```
mysql> SHOW DATABASES;
+-----+
| Database |
+-----+
| information_schema |
| leonard   |
+-----+
2 rows in set (0.00 sec)

mysql> █
```

```
mysql>
mysql> USE formation;
Database changed
mysql> SHOW TABLES;
Empty set (0.00 sec)

mysql>
```

Créer un table

□ **Commande MySQL :**

```
CREATE TABLE nom_table (  
    id_'nom_table' PROPRIETES,  
    CHAMP_1 PROPRIETES,  
    ...  
    CHAMP_X PROPRIETES  
);
```

CONSEIL :

Utilisez un éditeur de texte pour sauvegarder vos commandes SQL et pour ne pas avoir à les retaper...

□ **Exemple :**

```
mysql>  
mysql> CREATE TABLE Tournage (  
    -> id_tournage int not null auto_increment primary key,  
    -> id_acteur varchar(255) not null,  
    -> id_film varchar(255) not null  
    -> );  
Query OK, 0 rows affected (0.00 sec)  
  
mysql>
```

Importance des clés

□ Clés primaires :

- Identifiants UNIQUES de chaque enregistrement
- Définition via la propriété **PRIMARY KEY**
 - Nécessite l'ajout de la propriété NOT NULL pour empêcher l'ajout d'enregistrement sans clé
 - En cas de variable auto incrémentale, ajouter la propriété AUTO_INCREMENT

□ Clés étrangères :

- Clé de l'entité forte présente dans l'entité faible
- Définition via la propriété **FOREIGN KEY** nom_clé REFERENCES nom_table (clé_primaire_table_externe)

Création de clés primaires/étrangères

□ Clés primaires :

▣ Sur une colonne :

```
CREATE TABLE <Table> (  
    ID_nom Type PRIMARY KEY, ...);
```

▣ Sur plusieurs colonnes

```
CREATE TABLE <Table> (...,  
    CONSTRAINT nom_cle PRIMARY KEY  
    (Colonne1, Colonne2));
```

□ Clés étrangères :

```
CREATE TABLE <Table> (...,  
    CONSTRAINT nom_cle FOREIGN KEY  
    (Colonne1,...) REFERENCES <Table_etrange>  
    (clé_primaire_table_etrangere));
```


Structure d'une table

□ **Commande MySQL :**

DESC nom_table;

□ **Exemple :**

```
mysql>
```

```
mysql> DESC Tournage;
```

Field	Type	Null	Key	Default	Extra
id_tournage	int(11)	NO	PRI	NULL	auto_increment
id_acteur	varchar(255)	NO		NULL	
id_film	varchar(255)	NO		NULL	

```
3 rows in set (0.00 sec)
```

```
mysql>
```

Visualiser les enregistrements

- **Commande MySQL :**

```
SELECT * from nom_table;
```

- **Exemple :**

```
mysql>  
mysql> SELECT * FROM Tournage;  
Empty set (0.00 sec)
```

```
mysql>  
mysql>
```

Suppression d'une table

- Commande MySQL :

```
DROP TABLE nom_table;
```

- **ATTENTION : cette commande est irréversible, une fois supprimée, la table n'existera plus !**

- Exemple :

```
mysql> SHOW TABLES;
+-----+
| Tables_in_information |
+-----+
| Tournage               |
+-----+
1 row in set (0.00 sec)

mysql> DROP TABLE Tournage;
Query OK, 0 rows affected (0.00 sec)

mysql> SHOW TABLES;
Empty set (0.00 sec)

mysql>
```

Autres commandes pour les tables

□ **Supprimer ou ajouter une colonne :**

```
ALTER TABLE nom_table [ADD nom_colonne Type_colonne]
                        [DROP COLUMN nom_colonne]
                        [ADD CONSTRAINT nom_contrainte];
```

□ **Renommer une table :**

```
RENAME nom_table_ancien TO nouveau_nom_table;
```

Suppression d'une base

- Commande MySQL :

```
DROP DATABASE nom_database;
```

- **ATTENTION : cette commande est irréversible, une fois supprimée, la base n'existera plus !**

- Exemple :

```
mysql>  
mysql> DROP DATABASE formation;  
Query OK, 0 rows affected (0.00 sec)  
  
mysql>
```

Mise en Application

Mise en application

Base de données Film / Cinéma

- ❑ Implémenter sur MySQL les tables définies par le diagramme ER
- ❑ Dans un premier temps, ignorer les clés étrangères (FK)

Exercice EXPERT

- ❑ Prendre en compte les clés étrangères

Solution (1)

```
CREATE TABLE Cinema(  
    id_cinema INT NOT NULL AUTO_INCREMENT PRIMARY KEY,  
    nom VARCHAR(255) NOT NULL,  
    adresse VARCHAR(25) NOT NULL,  
    tel VARCHAR(25) NOT NULL );  
  
CREATE TABLE Film (  
    id_film INT NOT NULL AUTO_INCREMENT PRIMARY KEY,  
    titre TEXT NOT NULL,  
    date INT NOT NULL);  
  
CREATE TABLE Acteur (  
    id_acteur INT NOT NULL AUTO_INCREMENT PRIMARY KEY,  
    nom VARCHAR(120) NOT NULL,  
    bio TEXT NOT NULL,  
    date_naissance YEAR NOT NULL,  
    photo TEXT);
```


Solution (2)

```
CREATE TABLE Tournage (  
  id_tournage int not null auto_increment  
    primary key,  
  id_film varchar(255) not null,  
  id_acteur varchar(255) not null  
);
```

```
CREATE TABLE Projection (  
  id_projection INT NOT NULL  
    AUTO_INCREMENT,  
  id_film INT NOT NULL,  
  id_salle INT NOT NULL,  
  date INT NOT NULL,  
  horaire INT NOT NULL,  
  FOREIGN KEY (id_film) REFERENCES  
    Film(id_film),  
  FOREIGN KEY (id_salle) REFERENCES  
    Salle(id_salle),  
  PRIMARY KEY (id_projection)  
);
```

```
CREATE TABLE Salle (  
  id_salle INT NOT NULL AUTO_INCREMENT,  
  id_cinema INT NOT NULL,  
  sieges INT NOT NULL,  
  technologie TEXT,  
  numero INT NOT NULL,  
  taille INT NOT NULL,  
  accessibilité TEXT NOT NULL,  
  FOREIGN KEY (id_cinema) REFERENCES  
    Cinema(id_cinema),  
  PRIMARY KEY (id_salle)  
);
```

Mise en Application 2

Mise en application2

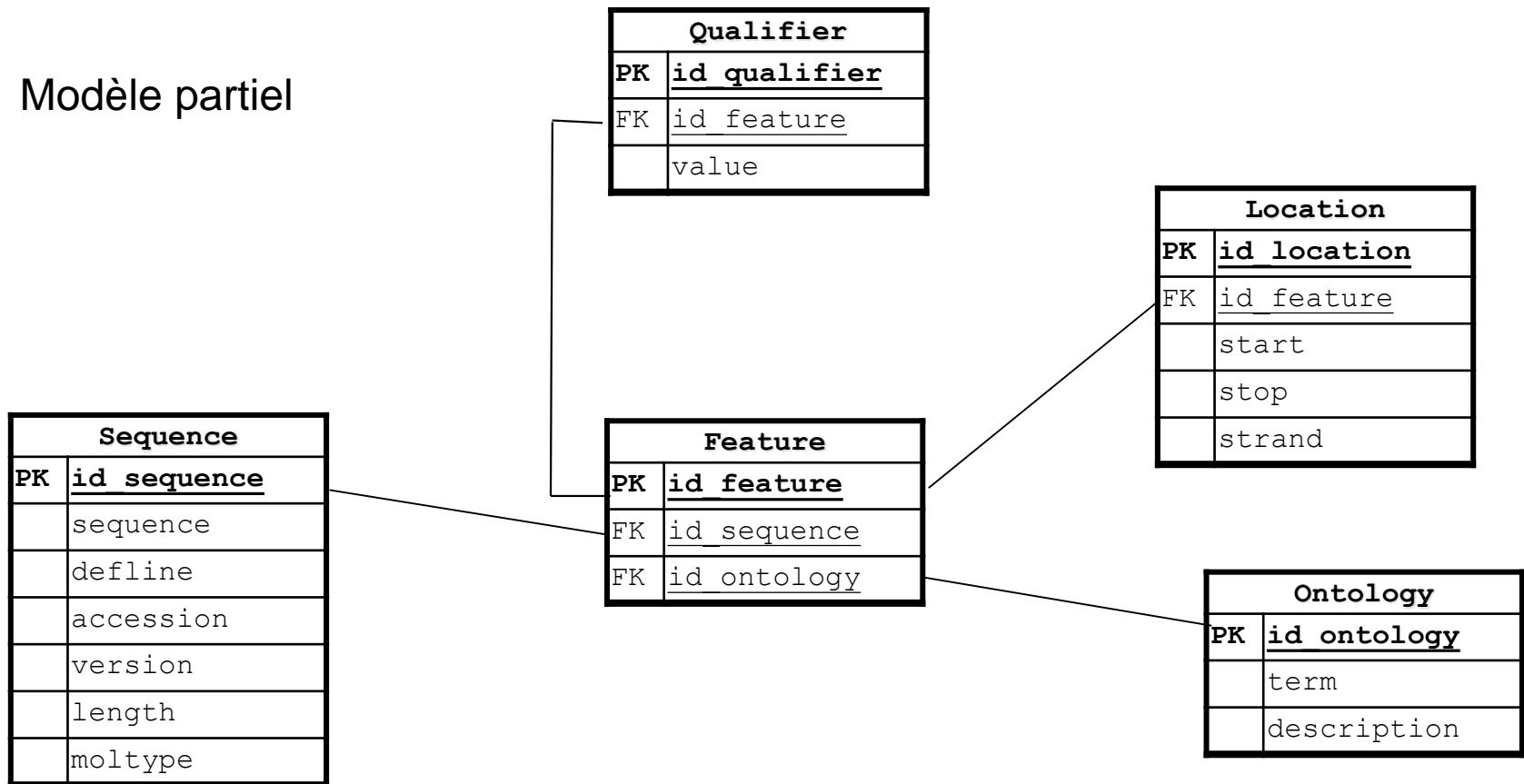
Base de données “GenBank”

Exemple fichier genbank :

```
LOCUS       NG_011471                53881 bp    DNA    linear    PRI 01-APR-2012
DEFINITION  Homo sapiens purinergic receptor P2X, ligand-gated ion channel, 7
            (P2RX7), RefSeqGene on chromosome 12.
ACCESSION   NG_011471 REGION: 4800..58680
VERSION     NG_011471.2  GI:300388175
FEATURES             Location/Qualifiers
     source          1..53881
     gene            146..53878
     exon            146..422
     start_codon     146..148
     exon            155..422
ORIGIN
      1 aaaatgccca tcctctgaac accatctttg ttagggcagc tgggggaggc cagctggggt
     61 gaggtcatct gccagccagg ccgtaggac ttggcgcttc ttgtttatca cagccacatg
    121 tggggccact gccagggcc gccccaactc tgcagtcatt ggaggagctt gaagttaaag
    181 actcctgcta aaaaccagta cgtttcattt tgcagttact gggagggggc ttgctgtggc
    241 cctgtcagga agagtagagc tctggtccag ctccgcgcag ggagggaggc tgtcaccatg
    301 ccggcctgct gcagctgcag tgatgttttc cagtatgaga cgaacaaagt cactcggatc
    361 cagagcatga attatggcac cattaagtgg ttcttcacag tgatcatctt ttcctacgtt
    421 tggtaaagtgg gatctgggga ggaccagat ctctgcagtg gccgacagca cagaaagccc
```

Mise en application 2

Modèle partiel



(Feature : élément d'annotation pour une séquence)

(Qualifier : type de feature (sequence, reference, texte libre...))

CREATE Sequence

```
CREATE TABLE Sequence (  
    id_sequence INT NOT NULL  
    AUTO_INCREMENT,  
    sequence LONGTEXT NOT NULL,  
    defline TEXT,  
    accession VARCHAR(255) NOT NULL,  
    version INT DEFAULT 0,  
    length INT DEFAULT 0,  
    moltype INT NOT NULL,  
    PRIMARY KEY(id_sequence)  
);
```

Sequence	
PK	<u>id sequence</u>
	sequence
	defline
	accession
	version
	length
	moltype

CREATE Ontology

```
CREATE TABLE Ontology (  
    id_ontology INT NOT NULL AUTO_INCREMENT,  
    term VARCHAR(255) NOT NULL,  
    description TEXT NOT NULL,  
    PRIMARY KEY (id_ontology)  
);
```

Ontology	
PK	<u>id ontology</u>
	term
	description

CREATE Feature

```
CREATE TABLE Feature (  
    id_feature INT NOT NULL AUTO_INCREMENT,  
    id_sequence INT NOT NULL,  
    id_ontology INT NOT NULL,  
    FOREIGN KEY (id_sequence) REFERENCES Sequence,  
    FOREIGN KEY (id_ontology) REFERENCES Ontology,  
    PRIMARY KEY(id_feature)  
);
```

Feature	
PK	<u>id feature</u>
FK	<u>id sequence</u>
FK	<u>id ontology</u>

CREATE Location

```
CREATE TABLE Location (  
    id_location INT NOT NULL AUTO_INCREMENT,  
    id_feature INT NOT NULL,  
    start INT NOT NULL,  
    stop INT NOT NULL,  
    strand INT NOT NULL,  
    FOREIGN KEY (id_feature) REFERENCES  
    Feature,  
    PRIMARY KEY(id_location)  
);
```

Location	
PK	<u>id location</u>
FK	<u>id feature</u>
	start
	stop
	strand

CREATE Qualifier

```
CREATE TABLE Qualifier (  
    id_qualifier INT NOT NULL AUTO_INCREMENT,  
    id_feature INT NOT NULL,  
    value TEXT NOT NULL,  
    FOREIGN KEY (id_feature) REFERENCES Feature,  
    PRIMARY KEY (id_qualifier)  
);
```

Qualifier	
PK	<u>id_qualifier</u>
FK	<u>id_feature</u>
	value

L'insertion de données

Insérer des données

- **Commande MySQL :**

- `INSERT INTO nom_table (champ1,..., champX) VALUES ('values1', ... , 'valuesX');`

- **Exemple :**

```
mysql> INSERT INTO Ontology (term, description) VALUES  
-> ('start codon', 'denotes an Methionine codon of a transcript');  
Query OK, 1 row affected (0.00 sec)
```

Visualisation des données

□ Commande MySQL :

▣ `SELECT ... FROM nom_table;`

□ Exemple :

```
mysql> SELECT * FROM ontology;
```

id_ontology	term	description
1	start codon	denotes an Methionine codon of a transcript

```
1 row in set (0.01 sec)
```

Mise en Application (suite)

Mise en application

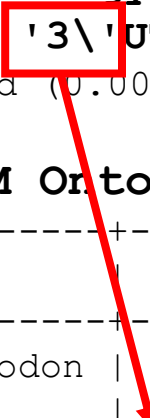
- Insérer dans la table Ontology les données suivantes :
 - ▣ un exon défini comme : an exon in genomic sequence
 - ▣ un exon type défini comme : 3'UTR, initial, internal, terminal, 5'UTR
- Insérer dans la table Location, les informations suivantes :
 - ▣ Feature 1 sur le brin 1 de 1 à 3
- Insérer dans la table Sequence l'enregistrement suivant :
 - ▣ >seq1
ATGACGATCAGCATCAGCTACAGCTG
version 1 de longueur 26 et de type 1
- Créer une feature liant la séquence 1 et l'ontologie 1

INSERT into Ontology

```
mysql> INSERT INTO Ontology (term, description) VALUES
-> ('exon', 'an exon in genomic sequence');
Query OK, 1 row affected (0.00 sec)
```

```
mysql> INSERT INTO Ontology (term, description) VALUES
-> ('exon type', '3\'UTR, initial, internal, terminal, 5\'UTR');
Query OK, 1 row affected (0.00 sec)
```

```
mysql> SELECT * FROM Ontology;
```



id_ontology	term	description
1	start codon	denotes an Methionine codon of a transcript
2	exon	an exon in genomic sequence
3	exon type	3'UTR, initial, internal, terminal, 5'UTR

```
3 rows in set (0.00 sec)
```

INSERT into Sequence

```
mysql> DESC Sequence;
```

Field	Type	Null	Key	Default	Extra
id_sequence	int(11)		PRI	NULL	auto_increment
sequence	longtext				
define	text	YES		NULL	
accession	varchar(255)				
version	int(11)	YES		0	
length	int(11)	YES		0	
moltype	int(11)			0	

```
7 rows in set (0.00 sec)
```

```
mysql> INSERT INTO Sequence (sequence, define, accession, version, length, moltype)
-> VALUES ('ATGACGATCAGCATCAGCTACAGCTG', '>seq1', 'seq1', 1, 26, 1);
```

```
Query OK, 1 row affected (0.00 sec)
```


INSERT into Feature

```
mysql> SELECT * FROM Sequence;
```

```
+-----+-----+-----+-----+-----+
| id_sequence | sequence                | accession | version | length | moltype |
+-----+-----+-----+-----+-----+
|          1 | ATGACGATCAGCATCAGCTACAGCTG | seq1     |      1 |     26 |        1 |
+-----+-----+-----+-----+-----+
1 row in set (0.03 sec)
```

```
mysql> SELECT * FROM Ontology;
```

```
+-----+-----+-----+
| id_ontology | term           | description
+-----+-----+-----+
|          1 | start codon    | denotes an Methionine codon of a transcript |
|          2 | exon           | an exon in genomic sequence
|          3 | exon type      | 3'UTR, initial, internal, terminal, 5'UTR
+-----+-----+-----+
3 rows in set (0.00 sec)
```

```
mysql> INSERT INTO Feature (id_sequence, id_ontology)  
-> VALUES (1, 1);
```

```
Query OK, 1 row affected (0.00 sec)
```

INSERT into Location

```
mysql> SELECT * From Feature;
```

```
+-----+-----+-----+
| id_feature | id_sequence | id_ontology |
+-----+-----+-----+
|          1 |          1 |          1 |
+-----+-----+-----+
1 row in set (0.01 sec)
```

```
mysql> DESC Location;
```

```
+-----+-----+-----+-----+-----+-----+
| Field          | Type      | Null | Key | Default | Extra          |
+-----+-----+-----+-----+-----+-----+
| id_location    | int(11)   |      | PRI | NULL    | auto_increment |
| id_feature     | int(11)   |      | MUL | 0        |                |
| start          | int(11)   |      |     | 0        |                |
| stop           | int(11)   |      |     | 0        |                |
| strand         | int(11)   |      |     | 0        |                |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)
```

```
mysql> INSERT INTO Location (id_feature, start, stop, strand)  
-> VALUES (1,1,3,1);
```

```
Query OK, 1 row affected (0.02 sec)
```

```
mysql> SELECT * FROM Sequence WHERE id_sequence = 2;
+-----+-----+-----+-----+-----+-----+-----+
| id_sequence | sequence | defline | accession | version | length | moltype |
+-----+-----+-----+-----+-----+-----+-----+
| 2 | ATGACGATCAGCATCAGCTACAGCTG | > seq1 | seq1 | 1 | 26 | 1 |
+-----+-----+-----+-----+-----+-----+
1 row in set (0.04 sec)
```

```
mysql> SELECT * FROM Feature WHERE id_sequence = 1;
+-----+-----+-----+
| id_feature | id_sequence | id_ontology |
+-----+-----+-----+
| 1 | 1 | 1 |
| 2 | 1 | 2 |
| 3 | 1 | 2 |
+-----+-----+-----+
3 rows in set (0.03 sec)
```

```
mysql> SELECT * FROM Location;
+-----+-----+-----+-----+-----+
| id_location | id_feature | start | stop | strand |
+-----+-----+-----+-----+-----+
| 1 | 1 | 1 | 3 | 1 |
| 2 | 2 | 1 | 6 | 1 |
| 3 | 3 | 15 | 20 | 1 |
+-----+-----+-----+-----+-----+
3 rows in set (0.20 sec)
```

```
mysql> SELECT * FROM Ontology;
+-----+-----+-----+
| id_ontology | term | description |
+-----+-----+-----+
| 1 | start codon | denotes an Methionine codon of a transcript |
| 2 | exon | an exon in genomic sequence |
| 3 | exon type | 3'UTR, initial, internal, terminal, 5'UTR |
+-----+-----+-----+
3 rows in set (0.00 sec)
```

```
mysql> SELECT * FROM Qualifier;
+-----+-----+-----+-----+
| id_qualifier | id_feature | id_ontology | value |
+-----+-----+-----+-----+
| 1 | 2 | 3 | initial |
| 2 | 3 | 3 | internal |
+-----+-----+-----+-----+
```

Ajouter les informations ci-contre dans vos tables

Langage SQL (suite)

Visualisation avancée

```
mysql> SELECT * FROM Sequence;
```

id_sequence	sequence	accession	version	length	moltype
1	ATGACGATCAGCATCAGCTACAGCTG	seq1	1	26	1
2	SLKLSKLPSPLYQVCLE	seq2	1	17	3

```
2 rows in set (0.00 sec)
```

```
mysql> SELECT sequence FROM Sequence WHERE accession = 'seq1';
```

sequence
ATGACGATCAGCATCAGCTACAGCTG

```
1 row in set (0.12 sec)
```

```
mysql> SELECT length FROM Sequence WHERE id_sequence = 1;
```

length
26

```
1 row in set (0.03 sec)
```

Jointure entre tables

```
mysql> SELECT * FROM Feature;
```

id_feature	id_sequence	id_ontology
1	1	1
2	2	2
3	2	2

```
3 rows in set (0.04 sec)
```

Retourner les descriptions contenues dans la table Features

```
mysql> SELECT Feature.id_feature, Ontology.description  
-> FROM Feature, Ontology  
-> WHERE Feature.id_ontology = Ontology.id_ontology;
```

id_feature	description
1	denotes an Methionine codon of a transcript
2	an exon in genomic sequence
3	an exon in genomic sequence

```
3 rows in set (0.04 sec)
```

Mise à jour des enregistrements

□ Commande MySQL :

```
UPDATE nom_table SET value = 'nouvelle valeur'  
WHERE identifiant;
```

□ Exemple :

```
mysql> UPDATE Qualifier SET value = 'terminal'  
      -> WHERE id_qualifier = 2;
```

```
Query OK, 1 row affected (0.00 sec)
```

```
Rows matched: 1   Changed: 1   Warnings: 0
```

Suppression d'un enregistrement

□ Commande MySQL :

```
DELETE FROM nom_table WHERE identifiant;
```

□ Exemple :

```
mysql> DELETE FROM Qualifier  
      -> WHERE id_qualifier = 2;
```

```
Query OK, 1 row affected (0.04 sec)
```

```
mysql> SELECT * FROM Qualifier;
```

qualifier_id	feature_id	ontology_id	value
1	2	5	initial

```
1 row in set (0.03 sec)
```


Mise en Application

Application

- Afficher la sous-séquence (paramètre SUBSTRING) et les coordonnées du start codon de la 'seq1'

Solution

```
mysql> SELECT SUBSTRING(sequence, start, (stop-start)+1), start, stop,
term
-> FROM Sequence, Ontology, Feature, Location
-> WHERE accession = 'seq1' AND term = 'exon' AND
-> Feature.id_ontology = Ontology.id_ontology AND
-> Feature.id_sequence = Sequence.id_sequence AND
-> Feature.id_feature = Location.id_feature;

+-----+-----+-----+-----+
| SUBSTRING(sequence, start, (stop-start)+1) | start | stop | term |
+-----+-----+-----+-----+
| ATGACG                                     |      1 |      5 | exon |
| CAGCTACAGCTG                             |     15 |     20 | exon |
+-----+-----+-----+-----+

2 rows in set (0.04 sec)
```

Langage SQL (suite)

Agréger les requêtes

```
mysql> SELECT * FROM Sequence;
```

id_sequence	sequence	defline	accession	version	length	moltype
1	ATGACGATCAGCATCAGCTACAGCTG	> seq1	seq1	1	26	1
2	SLKLSKLPSPPLYQVCLE	> seq2	L32174	1	17	3
3	TAGACTTACGATACTACT	> seq3	GB123	2	17	1

```
mysql> SELECT count(*), moltype from Sequence GROUP BY moltype;
```

count(*)	moltype
1	3
2	1

```
2 rows in set (0.08 sec)
```

Limiter l'affichage

□ Commande MySQL :

```
SELECT * FROM nom_table LIMIT nb;
```

□ Exemple :

```
mysql> SELECT * FROM Sequence LIMIT 2;
```

id_sequence	sequence	defline	accession	version	length	moltype
2	ATGACGATCAGCATCAGCTACAGCTG	> seq1	seq1	1	26	1
3	SLKLSKLPSPPLYQVCLE	> seq2	L32174	1	17	3

2 rows in set (0.08 sec)

MySQL avancé

Optimisation et Indexation

- En général, l'indexation accélère la recherche de données de plusieurs ordres de magnitude
- Considérons une liste de 1 000 000 de séquences caractérisées par un "accession number"
- Vous souhaitez trouver la séquence ayant l' « accession number » 'AC123456'
 - La réponse nécessite $O(1\ 000\ 000)$ opérations si le champ AC n'est pas indexé (Equivalent à un scan complet de la liste)
 - La réponse nécessite $O(\log(1\ 000\ 000)) = O(6)$ opérations si le champ AC est indexé (recherche par B-Tree)

Problèmes de l'indexation

- Nécessite **plus d'espace disque**
- Peut **ralentir les insertions**
- Implique que vous devez essayer de **connaître les données et les requêtes coûteuses** qui vont être régulièrement lancées
 - ▣ Indexer uniquement les champs qui vont être utilisés lors de vos requêtes
 - ▣ Nécessite de passer du temps pour définir les meilleurs index

Création et suppression d'un index

□ Création :

```
mysql> CREATE INDEX acindex ON Sequence(accession) ;  
Query OK, 1 row affected (0.18 sec)  
Records: 1  Duplicates: 0  Warnings: 0
```

□ Suppression :

```
mysql> DROP INDEX acindex;  
Query OK, 1 row affected (0.18 sec)  
Records: 1  Duplicates: 0  Warnings: 0
```

Les vues (ou table “virtuelle”)

- Table « virtuelle » spécifique pour un domaine ou pour une classe d'utilisateurs
- **Une vue représente à un instant t l'image des tables qu'elle utilise**
- Avantages :
 - ▣ Accélère et simplifie les requêtes
 - ▣ Masquer certaines données sensibles à certains utilisateurs
- Inconvénients :
 - ▣ Mettre à jour lors de l'insertion de données
 - ▣ Connaître à l'avance le type de requêtes réalisées

Création et suppression d'une vue

□ Création :

```
mysql> CREATE VIEW nom_Vue AS SELECT maRequête...;  
Query OK, 1 row affected (0.18 sec)  
Records: 1  Duplicates: 0  Warnings: 0
```

□ Suppression :

```
mysql> DROP VIEW nom_Vue;  
Query OK, 1 row affected (0.18 sec)  
Records: 1  Duplicates: 0  Warnings: 0
```

Commandes EXPERT

□ Commandes utiles pour l'administration

■ Check

- `CHECK TABLE nom_table;`
- vérifie l'intégrité d'une table

■ Repair

- `REPAIR TABLE nom_table;`
- tente de réparer une table en erreur

```
mysql> CHECK TABLE customers;
```

Table	Op	Msg_type	Msg_text
formation.customers	check	status	OK

```
1 row in set (0.00 sec)
```

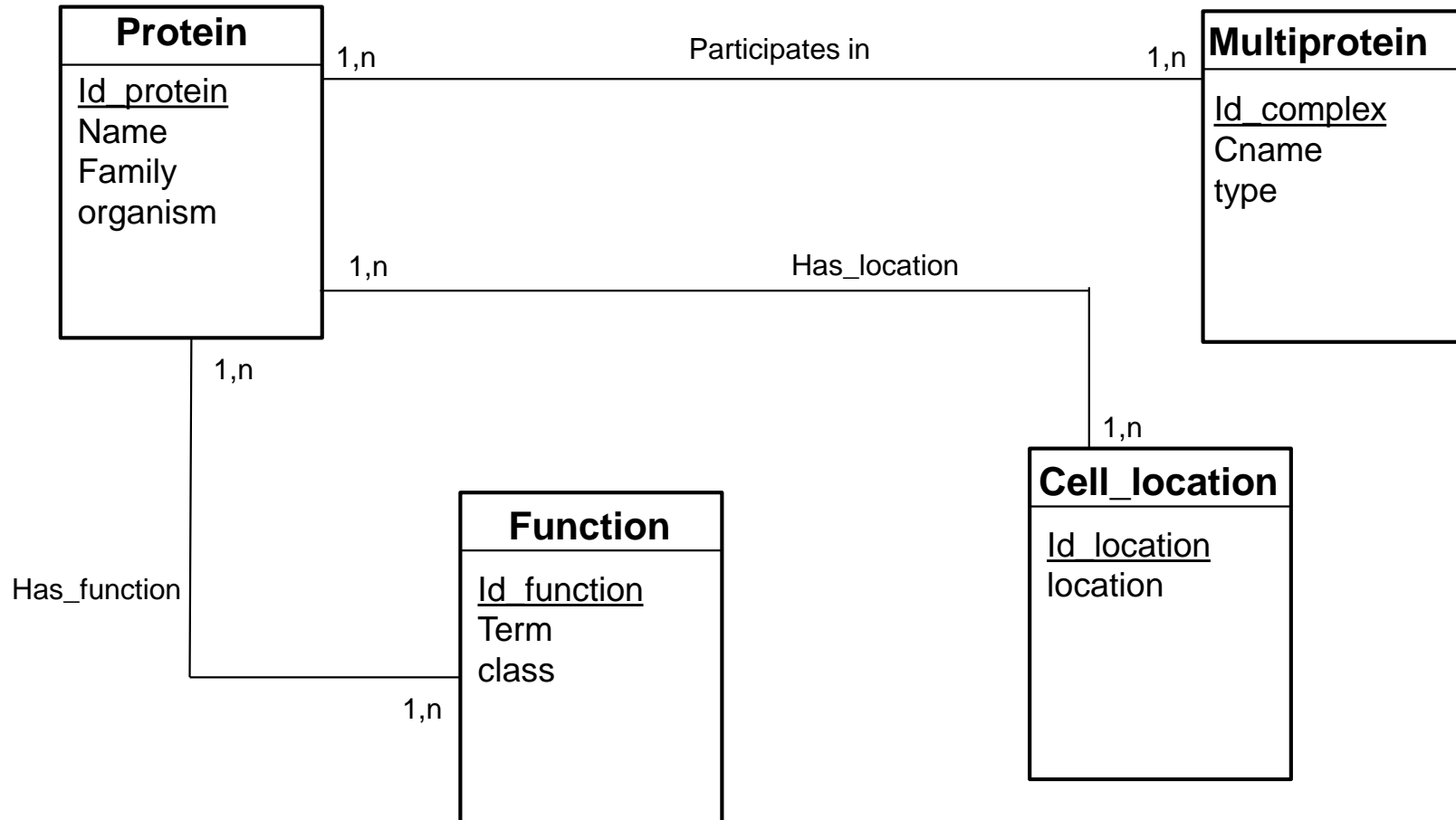
```
mysql>
```

Mise en Application

Exercice 1

- Base de données « **Interaction** »
- Soit le schéma suivant :
 - ▣ Protein (id_protein, name, family, organism)
 - ▣ Function(id_function,term,class)
 - ▣ Multiprotein_complex(id_complex,cname,type)
 - ▣ Cellular_location(id_location,location)
 - ▣ Has_function(#id_protein,#id_function)
 - ▣ Has_location(#id_protein,#id_location)
 - ▣ Participates_in(#id_protein,#id_complex)
- Faire le diagramme ER et le modèle relationnel

Exercise 1



Les requêtes...

Commandes

1. Afficher toutes les protéines
2. Afficher les protéines dont le nom commence par `Th`
3. Afficher le nombre de type complexes différents
4. Afficher les complexes ayant une fonction contenant le terme `dehydrogenase`
5. Afficher tous les complexes multi-protéiques et le nombre de protéines impliqués dans ceux-ci

Commandes avancées

6. Trouver le nombre maximum de protéines différentes qui participent à n'importe quel complexe (utilisation de `MAX` et d'une sous-requête)

Correction (1)

```
1) SELECT * FROM protein;
```

Correction (2)

- 1) `SELECT * FROM protein;`
- 2) `SELECT name FROM protein WHERE name like 'Th%';`

Correction (3)

- 1) `SELECT * FROM protein;`
- 2) `SELECT name FROM protein WHERE name like 'Th%';`
- 3) `SELECT COUNT(DISTINCT type)`
`FROM multiprotein_complex;`

Correction (4)

- 1) `SELECT * FROM protein;`
- 2) `SELECT name FROM protein WHERE name like 'Th%';`
- 3) `SELECT COUNT(DISTINCT type)`
`FROM multiprotein_complex;`
- 4) `SELECT cname`
`FROM multiprotein_complex, participates_in, has_function, function WHERE function`
`like '%dehydrogenase%'`
`AND [jointure]; (protein n'est pas utile car protein_id est partagé)`

Correction (5)

- 1) `SELECT * FROM protein;`
- 2) `SELECT name FROM protein WHERE name like 'Th%';`
- 3) `SELECT COUNT(DISTINCT type)
FROM multiprotein_complex;`
- 4) `SELECT cname
FROM multiprotein_complex, participates_in, protein, has_function, function WHERE
function like '%dehydrogenase'
AND [jointure];`
- 5) `SELECT cname, COUNT(protein.name)
FROM multiprotein_complex, participates_in, protein
WHERE [jointure]
GROUP BY cname;`

Correction (6)

- 1) `SELECT * FROM protein;`
- 2) `SELECT name FROM protein WHERE name like 'Th%';`
- 3) `SELECT COUNT(DISTINCT type)
FROM multiprotein_complex;`
- 4) `SELECT cname
FROM multiprotein_complex, participates_in, protein, has_function, function WHERE
function like '%dehydrogenase'
AND [jointure];`
- 5) `SELECT cname, COUNT(protein.name)
FROM multiprotein_complex, participates_in, protein
WHERE [jointure]
GROUP BY cname;`
- 6) `SELECT MAX(number)
FROM (
SELECT COUNT(*) as number
FROM participates_in
GROUP BY id_complex
);`

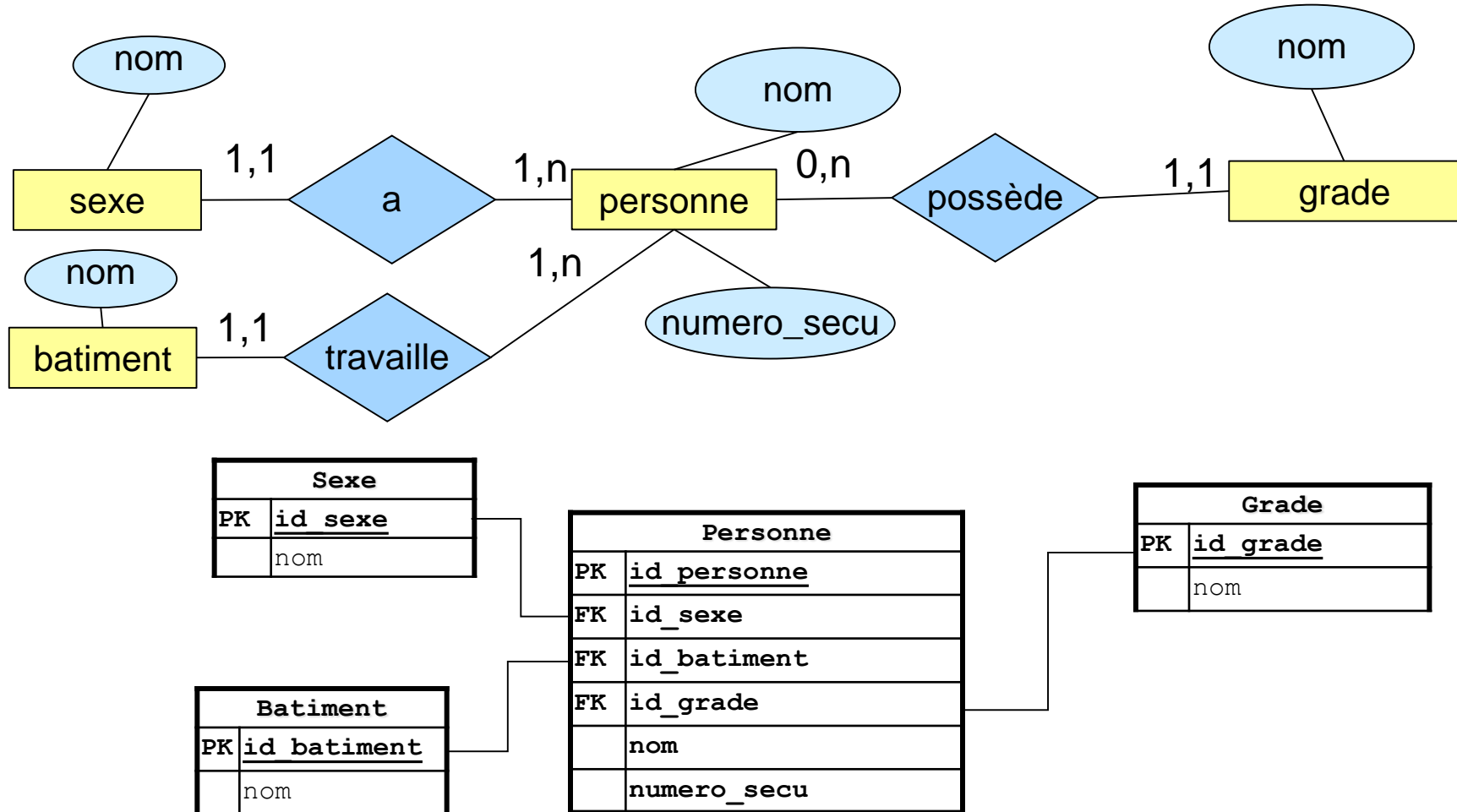
Exercice 2

- Base de données “**faculty**”
- A partir du fichier suivant “faculty.csv”
 - ▣ Créer le dictionnaire de données
 - ▣ Faire le MCD (diagramme ER)
 - ▣ Faire le modèle relationnel logique (diagramme relationnel)
 - ▣ Réaliser l’implémentation physique
 - ▣ Répondre aux questions

Les requêtes

1. Afficher tous les personnels
2. Afficher la liste des départements
3. Afficher tous les personnels dont le nom commence par S
4. Afficher le nombre de personnels dont le nom commence par S
5. Afficher le nom et le code de localisation des individus dont le code de localisation est T
6. Afficher toutes les informations des personnels (nom, grade, sexe, bâtiment...)
7. Afficher le nom et le code de localisation des individus classés par ordre alphabétique des noms et dont le code de localisation est T ou P
8. Afficher les personnes qui ne sont pas entre Catley et Harriman

Diagramme ER /Modèle Relationnel



Création des tables

```
❑ CREATE TABLE Sexe (  
    nom text not null,  
    id_sexe INT NOT NULL  
        AUTO_INCREMENT,  
    PRIMARY KEY (id_sexe));
```

```
❑ CREATE TABLE Batiment (  
    nom text not null,  
    id_batiment INT NOT NULL AUTO_INCREMENT,  
    PRIMARY KEY (id_batiment));
```

Création des tables (2)

```
□ CREATE TABLE Personne (  
    id_grade int not null, id_sexe int not null,  
    id_batiment int not null, nom text not null,  
    numero_secu int not null, id_personne INT NOT NULL  
    AUTO_INCREMENT,  
    PRIMARY KEY (id_personne),  
    FOREIGN KEY (id_grade) REFERENCES grade,  
    FOREIGN KEY (id_sexe) REFERENCES sexe,  
    FOREIGN KEY (id_batiment) REFERENCES batiment);
```

Correction de requêtes

1) Afficher tous les personnels

```
SELECT personne.nom, personne.numero_secu  
FROM personne;
```

2) Afficher la liste des départements

```
SELECT batiment.nom  
FROM batiment;
```

3) Afficher tous les personnels dont le nom commence par K

```
SELECT personne.nom  
FROM personne  
WHERE personne.nom LIKE 'K%';
```

4) Afficher le nombre de personnels dont le nom commence par K

```
SELECT COUNT(*)  
FROM personne  
WHERE personne.nom LIKE 'K%';
```

5) Afficher le nom et le code de la localisation des individus dont le code de localisation est T

```
SELECT personne.nom, batiment.nom  
FROM personne, batiment  
WHERE personne.id_batiment=batiment.id_batiment AND  
batiment.nom = "T";
```

Correction des requêtes (2)

6) Afficher toutes les données

```
SELECT personne.nom, personne.numero_secu, grade.nom, sexe.nom, batiment.nom
FROM personne, grade, sexe, batiment
WHERE personne.id_grade=grade.id_grade AND
personne.id_sexe=sexe.id_sexe AND
personne.id_batiment=batiment.id_batiment;
```

7) Afficher le nom et le code de localisation des individus classer par ordre alphabétique des noms et dont le code de localisation est T ou P (et sens inverse)

```
SELECT personne.nom, batiment.nom
FROM personne, batiment
WHERE personne.id_batiment=batiment.id_batiment AND
(batiment.nom = "T" OR batiment.nom = "P")
ORDER BY personne.nom;
```

```
SELECT personne.nom, batiment.nom
FROM personne, batiment
WHERE personne.id_batiment=batiment.id_batiment AND
(batiment.nom = "T" OR batiment.nom = "P")
ORDER BY personne.nom DESC;
```

8) Afficher les personnes qui ne sont pas entre Catley et Harriman

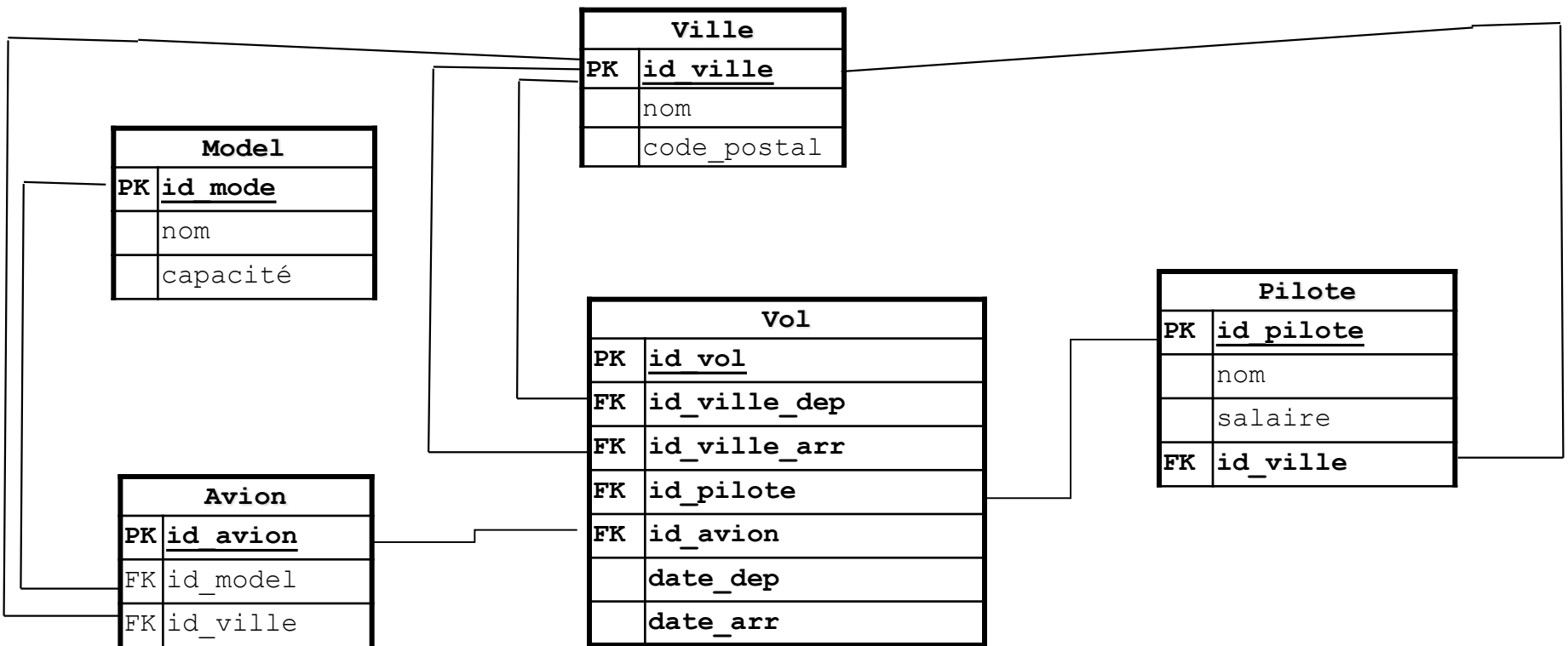
```
SELECT personne.nom FROM personne
WHERE personne.nom
NOT BETWEEN 'Catley' AND 'Harriman';
```

Exercice 3

- Base de données “**Compagnie aérienne**”
- A partir du fichier suivant : compagnie.csv
 - Créer le dictionnaire de données
 - Faire le diagramme ER
 - Faire modèle relationnel logique
 - Réaliser l'implémentation physique
 - Répondre aux requêtes

Diagramme relationnel

□ Compagnie aérienne



Les requêtes



1. Afficher toutes les informations administratives sur les pilotes employés.
2. Afficher la liste des avions ayant une capacité supérieure à 350.
3. Quels sont les numéros des pilotes en service et les villes de départ (avec horaires) de ceux ci
4. Quels sont les numéros et les noms des avions basés à Nice.
5. Quel est le nom des pilotes vivant à Paris et dont le salaire est supérieur à 200000 euros.
6. Quels sont les avions (numéro et nom) basés soit à Nice soit dont la capacité est supérieure à 350.
7. Afficher la liste des vols Paris-Marseille
8. Quels est le nombre de pilotes qui ne sont pas en service.

Les requêtes (suite)

9. Quels sont les vols (numéro et ville de départ) accomplis par les pilotes numéro 1 et 5.
10. Afficher le nombre de vols accomplis de Marseille par des pilotes de Marseille.
11. Quels sont les vols accomplis par des avions qui ne sont pas basé à Nice.
12. Quel sont les pilotes (numéro et nom) qui ont assuré au moins un vol de Marseille avec un avion d'une capacité supérieure à 250.
13. Quel est le nom du pilote vivant à Paris et assurant un vol de Marseille avec un airbus.
14. Quels est le nombre de vols accomplis par un pilote vivant à Marseille, partant de Marseille ou allant vers Marseille avec un avion basé à Nice.
15. Quels sont les pilotes (numéro et nom) vivant dans la même ville que le pilote M. Dupont ?
16. Quels sont les numéro des pilotes autre que Dupont qui sont en service.
17. Quelles sont les villes qui sont à la destination d'un vol qui a débuté par l'arrivée d'un vol provenant de Paris ?
18. Quels sont les avions (numéro) basés dans la même ville que l'avion 101.

Création des tables définitives

```
-- ville (premiere table cree)
CREATE TABLE Ville(
nom VARCHAR(64),
id_ville INT NOT NULL AUTO_INCREMENT PRIMARY KEY
);

--model (modele avion)
CREATE TABLE Model(
nom VARCHAR(64),
capacite INT,
id_model INT NOT NULL AUTO_INCREMENT PRIMARY KEY
);

--pilote
CREATE TABLE Pilote(
id_ville INT NOT NULL,
FOREIGN KEY (id_ville) REFERENCES
    ville(id_ville),
nom VARCHAR(64),
salaire INT,
id_pilote INT NOT NULL AUTO_INCREMENT PRIMARY KEY
);
```

```
--avion
CREATE TABLE Avion(
id_model INT NOT NULL,
FOREIGN KEY (id_model) REFERENCES ville(id_model),
id_ville INT NOT NULL,
FOREIGN KEY (id_ville) REFERENCES ville(id_ville),
id_avion INT NOT NULL AUTO_INCREMENT PRIMARY KEY
);

--vol
CREATE TABLE Vol(
id_pilote INT NOT NULL,
FOREIGN KEY (id_pilote) REFERENCES Pilote(id_pilote),
id_avion INT NOT NULL,
FOREIGN KEY (id_avion) REFERENCES Avion(id_avion),
date_dep TIME,
date_arr TIME,
id_ville_dep INT NOT NULL,
FOREIGN KEY (id_ville_dep) REFERENCES Ville(id_ville),
id_ville_arr INT NOT NULL,
FOREIGN KEY (id_ville_arr) REFERENCES Ville(id_ville),
id_vol INT NOT NULL AUTO_INCREMENT,
PRIMARY KEY(id_vol)
);
```

Correction des requêtes (1)

```
-- 1
SELECT
    Pilote.id_pilote, Pilote.nom, Pilote.salaire, V
    ille.nom AS "Nom de la Ville"
FROM Pilote, Ville
WHERE Ville.id_ville = Pilote.id_ville
ORDER BY Pilote.id_pilote;
```

```
-- 2
SELECT id_avion
FROM Avion, Model
WHERE Avion.id_model = Model.id_model
AND Model.capacite > 350;
```

```
-- 3
SELECT Pilote.id_pilote, Pilote.nom, Ville.nom AS
    "Nom de la Ville", Vol.dep_h
FROM Vol, Pilote, Ville
WHERE Vol.id_pilote = Pilote.id_pilote
AND Ville.id_ville = Vol.id_ville_dep;
```

```
-- 4
SELECT Avion.id_avion, Model.nom
FROM Avion , Model, Ville
WHERE Avion.id_ville=Ville.id_ville
AND Ville.nom LIKE 'Nice'
AND Avion.id_model=Model.id_model;
```

```
-- 5
SELECT Pilote.id_pilote, Pilote.nom
FROM Pilote, Ville
WHERE Ville.nom = "Paris"
AND Ville.id_ville = Pilote.id_ville
AND Pilote.salaire > 200000;
```

```
-- 6
SELECT Avion.id_avion, Model.nom
FROM Avion, Ville, Model
WHERE Avion.id_ville = Ville.id_ville
AND Model.id_model = Avion.id_model
AND (Ville.nom LIKE "Nice" OR Model.capacite >=
350);
```

```
-- 7
SELECT Vol.id_vol, v1.nom, v2.nom
FROM Vol, Ville as v1, Ville as v2
WHERE v1.nom = "Paris"
AND v1.id_ville = Vol.id_ville_dep
AND v2.nom = "Marseille"
AND v2.id_ville = Vol.id_ville_arr;
```

```
-- 8
SELECT COUNT(id_pilote)
FROM Pilote
WHERE Pilote.id_pilote
NOT IN (
    SELECT DISTINCT Pilote.id_pilote
    FROM Pilote, Vol
    WHERE Pilote.id_pilote = Vol.id_pilote
);
```

Correction des requêtes (2)

```
-- 9
SELECT Vol.id_vol
FROM Vol
WHERE Vol.id_pilote = 1 OR Vol.id_pilote = 5;
```

```
-- 10
SELECT COUNT(Vol.id_vol)
FROM Vol,Pilote,Ville
WHERE Pilote.id_pilote = Vol.id_pilote
AND Vol.id_ville_dep = Pilote.id_ville
AND Pilote.id_ville = Ville.id_ville
AND Ville.nom LIKE "Marseille";
```

```
-- 11
SELECT Vol.id_vol
FROM Vol,Avion
WHERE Vol.id_avion = Avion.id_avion
AND Avion.id_avion
NOT IN ( SELECT id_avion
        FROM Avion, Ville
        WHERE Avion.id_ville = Ville.id_ville
        AND Ville.nom LIKE "Nice");
```

```
-- 12
SELECT Pilote.id_pilote,Pilote.nom
FROM Pilote,Vol,Avion,Ville,Model
WHERE Vol.id_ville_dep = Ville.id_ville
AND Pilote.id_pilote = Vol.id_pilote
AND Ville.nom LIKE "Marseille"
AND Vol.id_avion = Avion.id_avion
AND Avion.id_model = Model.id_model
AND Model.capacite > 250;
```

```
-- 13
SELECT Pilote.id_pilote,Pilote.nom,Model.nom
FROM Vol,Pilote,Ville as v1, Ville as v2,Avion,Model
WHERE Pilote.id_pilote = Vol.id_pilote
AND Vol.id_ville_dep = v1.id_ville
AND v1.nom LIKE "Marseille"
AND Pilote.id_ville = v2.id_ville
AND v2.nom LIKE "Paris"
AND Vol.id_avion=Avion.id_avion
AND Avion.id_model=Model.id_model AND Model.nom LIKE "A%";
```

```
-- 14
SELECT COUNT(Vol.id_vol)
FROM Avion,Pilote,Vol,Ville as v1,Ville as v2
WHERE v1.nom LIKE "Marseille"
AND v2.nom LIKE "Nice"
AND Vol.id_pilote = Pilote.id_pilote
AND Pilote.id_ville = v1.id_ville
AND (Vol.id_ville_dep = v1.id_ville OR Vol.id_ville_arr =
v1.id_ville)
AND Avion.id_avion = Vol.id_avion
AND Avion.id_ville = v2.id_ville;
```

```
-- 15
SELECT p2.id_pilote,p2.nom
FROM Pilote as p1, Pilote as p2
WHERE p1.nom LIKE "Dupont"
AND p1.id_ville = p2.id_ville
AND NOT p2.nom LIKE "Dupont";
```

```
-- 16
SELECT Pilote.id_pilote,Pilote.nom
FROM Pilote
WHERE Pilote.nom NOT LIKE "Dupont";
```

Correction des requêtes (3)

```
-- 17
SELECT DISTINCT Ville.nom
FROM Ville,Vol
WHERE Vol.id_ville_arr = Ville.id_ville
AND Vol.id_avion
IN(  SELECT Vol.id_avion
      FROM Vol, Ville
      WHERE Vol.id_ville_dep = Ville.id_ville
      AND Ville.nom = "Paris")
;
```

```
-- 18
SELECT a2.id_avion
FROM Avion as a1,Avion as a2
WHERE a1.id_avion = 2
AND a2.id_ville = a1.id_ville;
```

Interface de programmation

API pour MySQL

CGI



HTML

HTML

- ☐ Voir le cours de Costas

Python::CGI

Interaction navigateur/ programme
(HTML/Python)

CGI

- ♦ CGI (Common Gateway Interface)
- ♦ Normes pour les programmes permettant **l'interférence avec les serveurs d'information** (tels que les serveurs HTTP)
- ♦ Indépendant de tout langage
- ♦ Exemple : http://www.dsimb.inserm.fr/~gelly/DB/exemple_cgi/index.html

« Web browsing » (1)

Quelques rappels :

- ♦ Un **lien hypertexte** permet de naviguer sur une page web (ou URL)
- ♦ Le navigateur **contacte le serveur web** (protocole HTTP) et réclame à partir de l'URL le fichier
- ♦ Le server web analyse l'URL et recherche le fichier correspondant. Si le fichier est trouvé, celui-ci est interprété et est renvoyé au navigateur web (sinon : message d'erreur : **404 Page not Found**)

Not Found

The requested URL /oldpage.html was not found on this server.

Apache/2.2.3 (CentOS) Server at www.example.com Port 80

« Web browsing » (2)

Et le rapport avec le CGI ?

- ♦ Possibilité de configurer le **serveur HTTP** pour qu'un fichier ne soit pas renvoyé.
- ♦ Fichier exécuté **comme un programme**, et tout ce que le programme envoie est renvoyé vers le navigateur
- ♦ Les programmes sont appelés **scripts CGI**
- ♦ Peuvent être rédigés en **Python, PERL, Shell, C, C++,** etc.

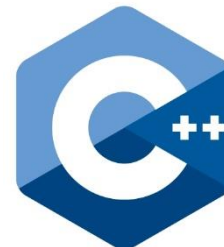
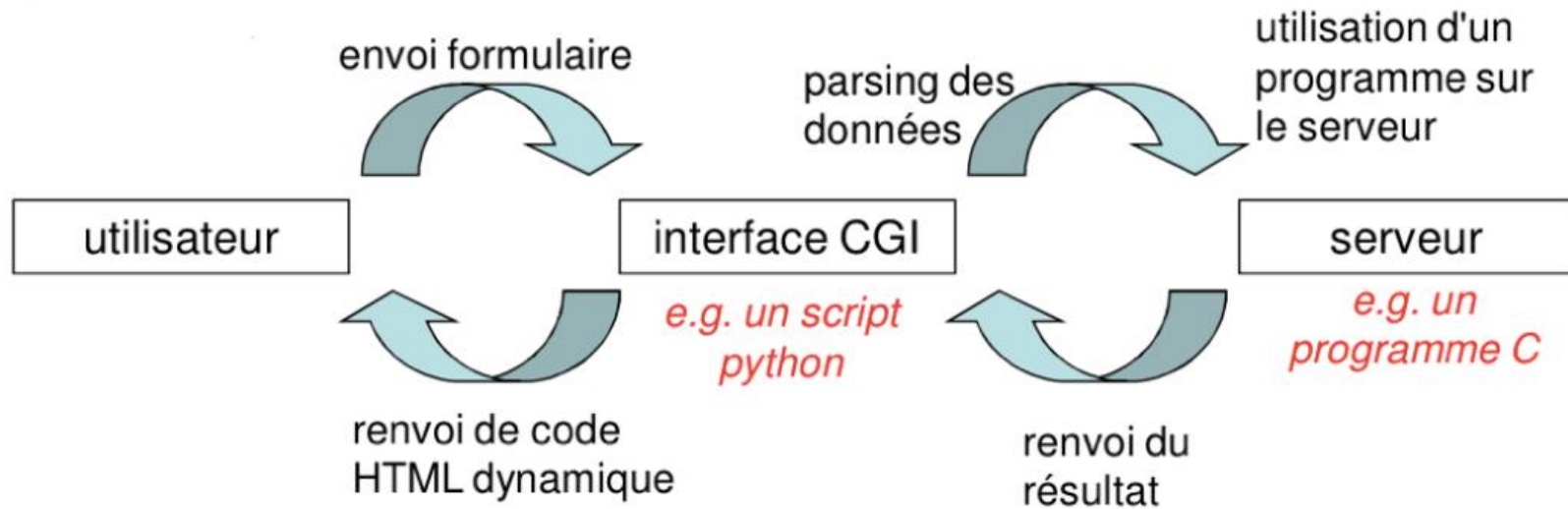


Diagramme de l'architecture CGI



Configuration CGI (1)

- ♦ Avant de réaliser ses premiers scripts CGI, il est nécessaire de configurer le serveur.
- ♦ Tous les programmes à exécuter par le serveur HTTP sont conservés dans un répertoire préconfiguré. Par convention, il est nommé :

`/var/www/cgi-bin`

- ♦ Par convention, les fichiers CGI ont pour extension `.cgi` (bien qu'il soit possible de garder l'extension propre au langage)

Exemple pour python : `test.py`.

Configuration CGI (2)

- ♦ Par défaut, le serveur Linux est configuré pour exécuter uniquement les scripts dans le répertoire

`/var/www/cgi-bin`

- ♦ Pour spécifier un autre répertoire où exécuter les scripts CGI, il faut commenter les lignes suivantes du fichier `httpd.conf` (`/etc/apache2/httpd.conf`)

```
<Directory "/var/www/cgi-bin">  
    AllowOverride None  
    Options ExecCGI  
    Order allow,deny  
    Allow from all  
</Directory>  
  
<Directory "/var/www/cgi-bin">  
    Options All  
</Directory>
```


Mon premier « Hello World »

- ♦ Création d'un premier script CGI nommé hello.py
- **ATTENTION** : avant de lancer un programme CGI, assurez-vous d'avoir changé les modes de lecture/écriture pour le rendre exécutable :

```
chmod 755 hello.py
```

```
#!/usr/bin/python
```

← Localisation de Python

```
print "Content-type:text/html\r\n\r\n"
```

← Spécification du langage
OBLIGATOIRE

```
print '<html>'
print '<head>'
print '<title>Hello Word - First CGI Program</title>'
print '</head>'
print '<body>'
print '<h2>Hello Word! This is my first CGI
program</h2>'
print '</body>'
print '</html>'
```

} Code HTML

```
Hello Word! This is my first CGI program
```

Variables environnements CGI

- ♦ Il existe un certain nombre de variables prédéfinis
 - **PATH_INFO** : retourne le chemin permettant l'accès au script
 - **SCRIPT_NAME** : retourne le nom du script CGI
 - **SERVER_NAME** : retourne l'adresse IP et le nom d'hôte du serveur
 - **SERVER_SOFTWARE** : retourne le nom et la version du serveur
- ♦ D'autres variables et exemples sur <http://www.cgi101.com/book/ch3/text.html>

Interaction avec l'utilisateur : formulaires

- ◆ Les méthodes **POST** et **GET** : Passage d'informations du navigateur au server web

Exemple : la soumission d'une séquence nucléotidique pour la traduire en séquence protéique

- ◆ **POST** : s'impose pour la transmission d'informations confidentielles, de taille importante, où il faut éviter de ré-exécuter le code facilement

- ◆ **GET** : est très utile quand la page demande une ressource simple (un ID) et doit (ou peut) être accessible par une URL

Exemple avec la méthode GET

- ♦ La méthode **GET** est la méthode par défaut pour passer les informations du navigateur au serveur web.

N'utilisez jamais la méthode GET pour des données cryptées comme les mots de passe !

Exemple :

```
http://www.test.com/cgi-bin/hello.py?key1=value1&key2=value2
```

- ♦ Les informations peuvent être transmises par l'intermédiaire d'un formulaire HTML à l'aide de la méthode GET.

Exemple de la méthode GET par simple URL

```
#!/usr/bin/python
```

```
# Import modules for CGI handling
```

```
import cgi, cgitb
```

```
# Create instance of FieldStorage
```

```
form = cgi.FieldStorage()
```

```
# Get data from fields
```

```
first_name = form.getvalue('first_name')
```

```
last_name = form.getvalue('last_name')
```

```
print "Content-type:text/html\r\n\r\n"
```

```
print "<html>"
```

```
print "<head>"
```

```
print "<title>Hello - Second CGI Program</title>"
```

```
print "</head>"
```

```
print "<body>"
```

```
print "<h2>Hello %s %s</h2>" % (first_name, last_name)
```

```
print "</body>"
```

```
print "</html>"
```

Importation des modules cgi

Création d'un dictionnaire qui renfermera les éléments de la méthode GET

Stockage des variables présents dans l'URL

Affichage des variables

Le résultat :

Hello ZARA ALI

Exemple de la méthode GET par formulaire

```
<form action="/cgi-bin/hello_get.py" method="get">  
First Name: <input type="text" name="first_name"> <br />  
Last Name: <input type="text" name="last_name" />  
<input type="submit" value="Submit" />  
</form>
```

First Name:
Last Name:

Hello Jean-Christophe Gelly

Exemple avec la méthode POST

- La méthode généralement plus fiable de transmission d'information dans un programme CGI est la méthode **POST**.
- Similaire à la méthode GET, **il ne renvoie toutefois pas les informations dans l'URL**, mais dans un message distinct qui interférera avec le script CGI sous la forme d'une entrée standard.
- Le script CGI d'affichage est le même pour les deux méthodes; seul le formulaire devra être ajusté

Exemple de la méthode POST par formulaire

```
<form action="/cgi-bin/hello_get.py" method="post">  
First Name: <input type="text" name="first_name"><br />  
Last Name: <input type="text" name="last_name" />  
<input type="submit" value="Submit" />  
</form>
```

First Name:

Last Name:

Hello Catherine Etchebest

Les autres éléments d'un formulaire (1)

Les cases à cocher (Checkbox) :

```
<form action="/cgi-bin/checkbox.cgi" method="POST" target="_blank">
<input type="checkbox" name="maths" value="on" /> Maths
<input type="checkbox" name="physics" value="on" /> Physics
<input type="submit" value="Select Subject" />
</form>
```

```
#!/usr/bin/python
```

```
# Import modules for CGI handling
import cgi, cgitb
```

```
# Create instance of FieldStorage
form = cgi.FieldStorage()
```

```
# Get data from fields
if form.getvalue('maths'):
    math_flag = "ON"
else:
    math_flag = "OFF"
```

```
if form.getvalue('physics'):
    physics_flag = "ON"
else:
    physics_flag = "OFF"
```

☐ Maths ☐ Physics

CheckBox Maths is : OFF

CheckBox Physics is : OFF

Les autres éléments d'un formulaire (2)

Les boutons « Radio » :

```
<form action="/cgi-bin/radiobutton.py" method="post" target="_blank">
<input type="radio" name="subject" value="maths" /> Maths
<input type="radio" name="subject" value="physics" /> Physics
<input type="submit" value="Select Subject" />
</form>
```

```
#!/usr/bin/python
```

```
# Import modules for CGI handling
```

```
import cgi, cgiib
```

```
# Create instance of FieldStorage
```

```
form = cgi.FieldStorage()
```

```
# Get data from fields
```

```
if form.getvalue('subject'):
```

```
    subject = form.getvalue('subject')
```

```
else:
```

```
    subject = "Not set"
```

☐ Maths ☒ Physics

Selected Subject is Physics

Les autres éléments d'un formulaire (3)

Les zones de texte (Textarea) :

```
<form action="/cgi-bin/textarea.py" method="post"
target="_blank">
<textarea name="textcontent" cols="40" rows="4">
Type your text here...
</textarea>
<input type="submit" value="Submit" />
</form>
```

```
#!/usr/bin/python
```

```
# Import modules for CGI handling
```

```
import cgi, cgiib
```

```
# Create instance of FieldStorage
```

```
form = cgi.FieldStorage()
```

```
# Get data from fields
```

```
if form.getvalue('textcontent'):
```

```
    text_content = form.getvalue('textcontent')
```

```
else:
```

```
    text_content = "Not entered"
```

DRING DRING ! C'est l'heure de se réveiller pour ceux
qui dorment !

Submit

Entered Text Content is DRING DRING ! C'est l'heure de se réveiller pour
ceux qui dorment !

Les autres éléments d'un formulaire (4)

Les volets déroulants (Drop down box) :

```
<form action="/cgi-bin/dropdown.py" method="post" target="_blank">
<select name="dropdown">
<option value="Maths" selected>Maths</option>
<option value="Physics">Physics</option>
</select>
<input type="submit" value="Submit"/>
</form>
```

```
#!/usr/bin/python
```

```
# Import modules for CGI handling
```

```
import cgi, cgitb
```

```
# Create instance of FieldStorage
```

```
form = cgi.FieldStorage()
```

```
# Get data from fields
```

```
if form.getvalue('dropdown'):
```

```
    subject = form.getvalue('dropdown')
```

```
else:
```

```
    subject = "Not entered"
```



Maths ▼ Submit

Selected Subject is Maths

Python::DBI

Informations

- Python peut prendre en charge un large panel de serveurs de base de données, comme :
 - MySQL
 - PostgreSQL
 - Microsoft SQL Server
 - Oracle
 - Etc.

MySQLdb est une interface pour se connecter à un serveur de base de données MySQL depuis Python

Installation de MySQLdb

- Si MySQLdb est non-installé, l'écriture de ce code entrainera l'erreur suivante :

```
#!/usr/bin/python
```

```
import MySQLdb
```



Traceback (most recent call last):

File "test.py", line 3, in <module>

import MySQLdb

ImportError: No module named MySQLdb

- Pour installer le module MySQLdb (attention, il faut avoir les privilèges (commande sudo)) :

```
$ gunzip MySQL-python-1.2.2.tar.gz
```

```
$ tar -xvf MySQL-python-1.2.2.tar
```

```
$ cd MySQL-python-1.2.2
```

```
$ python setup.py build
```

```
$ python setup.py install
```

Connexion à la base de données

```
#!/usr/bin/python
```

```
import MySQLdb
```

```
# Open database connection
```

```
db = MySQLdb.connect("localhost","testuser","test123","TESTDB" )
```

Nom de la base de données : TESTDB

```
# prepare a cursor object using cursor() method
```

```
cursor = db.cursor()
```

ID et mot de passe d'accès à la base de données :

testuser

test123

```
# execute SQL query using execute() method.
```

```
cursor.execute("SELECT VERSION()")
```

« localhost » fait ici référence au serveur local (votre PC)

```
# Fetch a single row using fetchone() method.
```

```
data = cursor.fetchone()
```

```
print "Database version : %s " % data
```

```
# disconnect from server
```

```
db.close()
```

Database version : 5.0.45



Exemple complet (1)

```
1  #!/usr/bin/python
2
3  import MySQLdb
4
5  # Open database connection
6  db = MySQLdb.connect("localhost","testuser","test123","TESTDB" )
7
8  # prepare a cursor object using cursor() method
9  cursor = db.cursor()
10
11 # Prepare SQL query to INSERT a record into the database
12 sql = "SELECT * FROM EMPLOYEE \
13 WHERE INCOME > '%d'" % (1000)
14
15 try:
16     # Execute the SQL command
17     cursor.execute(sql)
18     # Fetch all the rows in a list of lists
19     results = cursor.fetchall()
20     for row in results:
21         fname = row[0]
22         lname = row[1]
23         age = row[2]
24         sex = row[3]
25         income = row[4]
26         # Now print fetched result
27         print "fname=%s,lname=%s,age=%d,sex=%s,income=%d" % \
28             (fname, lname, age, sex, income )
29 except:
30     print "Error: unable to fetch data«
31
32 # disconnect from server
33 db.close()
```

Que fait ce code ?

Exemple complet (2)

```
1 #!/usr/bin/python
2
3 import cgi, cgitb
4 import MySQLdb
5
6 form = cgi.FieldStorage()
7
8 my_search = form.getvalue("pdb")
9
10 db = MySQLdb.connect("localhost", "testuser", "test123", "database-PDB")
11
12 cursor_search = db.cursor()
13
14 cursor_search.execute("SELECT P.PDB_id, P.Chain, P.Header
15                        FROM pdb
16                        WHERE P.PDB_id = %s ", my_search)
17
18 result = cursor_search.fetchone()
19
20 if result :
21     print("""<h2>Detailed PDB results</h2>
22           <table class="tableau zebre avec tri">
23               <thead>
24                   <tr>
25                       <th data-pos="1" data-tri="1">PDB_id</th>
26                       <th data-pos="2" data-tri="1">Chain</th>
27                       <th data-pos="3" data-tri="1">Header</th>
28                   </tr>
29               </thead>
30               <tbody>
31               """)
32     for row_result in result:
33         print "<tr><td>", row_result[0], "</td><td>", row_result[1], "</td><td>", row_result[2], "</td></tr>"
34
35     print("""</tbody>
36           </table>""")
37 else :
38     print "Aucun résultat"
39
40 db.close()
```

Que fait ce code ?

Mise en application

Votre première application

Création d'un système complet d'interrogation sur la base
Compagnie Aérienne :

- **Interface 1** : Interface publique : Affichage des vols départ – arrivé (avec heure et durée du vol)
- **Interface 2** : Interface publique : Conception interface de recherche par mot(s) clé(s)
- **Interface 1 BIS** : Interface publique : Affichage des sélections avec un tableur dynamique (utilisation de javascript/jquery) pour trier selon la ville, l'heure de départ, etc.
- **Interface 3** : Interface privée avec saisie de mot de passe: Insertion de nouvelles données

FIN

Merci pour votre attention
N'hésitez pas si vous avez des questions

Exemple 4: Conférences

- PARTICIPANTS(**numparticipant**, nom, prénom, datenaissance, numrue, nomrue, codepostal, ville, pays, langue)
- CONFERENCES(**numconférence**, titre, date, heuredébut, durée, thème, *reforateur*, *refanimateur*, *refsalle*)
- RESERVATIONS(*refconférence*, *refparticipant*)
- SALLES(**numsalle**, nom, batiment, numéro, superficie, capacité, équipement)

Exemple 4 : Les requêtes

1. Afficher la liste des personnes (numéro et nom) qui ont réservé une place pour la conférence numéro 16, dans l'ordre alphabétique des noms.
2. Afficher pour chaque conférence la liste des personnes ayant réservé dans l'ordre des numéros de conférences et dans l'ordre alphabétique pour chaque conférence.
3. Afficher pour chaque conférence son numéro et le nom du conférencier ainsi que la liste des personnes ayant réservé.
4. Afficher le programme de chaque salle pour le 15/11/2004.
5. Afficher pour chaque participant la liste des conférences pour lesquelles il a réservé.
6. Id mais on affichera également ceux n'ayant pas réservé.
7. Un participant signale qu'il arrivera le 15/11/2004 à 11h et qu'il doit repartir le 16/11/2004 à 17h. Donnez la liste des conférences auxquelles il pourrait assister.
8. Afficher la liste des participants parlant la même langue que monsieur DECOENINCK.
9. Affichez la liste des salles données par leur numéro, leur emplacement et la précision petite, moyenne ou grande selon que la capacité est inférieure à 40, entre 40 et 100 ou supérieure à 100.
10. La salle A7125 du bâtiment A ne peut être utilisée. Affichez les salles de capacité supérieure.
11. La salle A7125 du bâtiment A ne peut être utilisée. Affichez les salles de capacité supérieure et de même niveau d'équipement.
12. Affichez le nombre de salles et la capacité totale des salles.
13. Affichez la capacité totale des salles de moins de 100 places.
14. Affichez la capacité totale des salles par niveau d'équipement.
15. Affichez pour chaque conférence le nombre de participants.
16. Affichez la liste des conférences ayant plus de 100 participants.
17. Lorsque la langue du participant est différente de la langue de l'orateur, il faut prévoir un casque pour la traduction. Donnez le nombre de casques à prévoir pour chacune des conférences.
18. Affichez la liste des participants aux conférences du thème finances et des participants aux conférences données en anglais.
19. Affichez la liste des conférences ayant le lieu le même jour et dans la même salle que la conférence 37.
20. Affichez la liste des salles où sont données des conférences avec des participants parlant le russe.
21. Donnez les noms des orateurs qui n'assistent à aucune conférence .
22. Donnez les noms des participants ayant assisté à toutes les conférences.
23. Donnez le noms de l'orateur ayant eu le plus de succès, c'est-à-dire, l'audience la plus importante à une de ses conférences.
24. Donnez le noms de l'orateur ayant eu le plus de succès, c'est-à-dire, l'audience cumulée la plus importante à toutes ses conférences.