Bases de Données

Amélie Gheerbrant

Université de Paris
UFR Informatique
Institut de Recherche en Informatique Fondamentale

amelie@irif.fr

28 janvier 2021

Organisation

- 12 semaines
  Aujourd'hui : Introduction, le modèle relationnel
- Des transparents seront mis en ligne au fur et à mesure :
  `https://www.irif.fr/~amelie/BD.html`
- Les TPs sont très importants : mise en pratique des notions vues en cours.
- Attention : tout ce qui sera vu en cours ne sera pas forcément vu en TP.
- Modalités de contrôle des connaissances :
  un projet, un contrôle continu (deux interrogations en TP si présentiel) et un examen final

Ouvrages de référence

- Plusieurs ouvrages et documents vidéos en anglais et en français (c.f. page du cours)
- Par exemple :

  Bases de données et modèles de calcul, Jean-Luc Hainaut

  en particulier les chapitres 2 à 5

Objectifs de ce cours

Apprendre

- les principes généraux qui s'appliquent à la plupart des produits que vous pourriez être amenés à rencontrer
- mais pas les spécificités des systèmes (e.g., MySQL vs. PostgreSQL)

Objectifs de ce cours

Apprendre :

1. **La conception de bases de données**
   - Point de départ : description informelle d'une application
   - Abstraction et optimisation du cahier des charges (modélisation)
   - Création d'entités comprises par le système (extraction des relations de la base de données)
   - Optimisation des relations (normalisation)

2. **L'utilisation d'un système de gestion de bases de données**
   - Ecrire des requêtes dans un langage (SQL) compris par le système (Oracle, PostgreSQL, MySQL, DB2, etc)

Pourquoi étudier les bases de données (BD) ?

- Avant, portée plus limitée :
  salariés d'une entreprise, données bancaires, etc...
- Aujourd'hui le domaine englobe tout ce qui touche aux données :
  - recherches Web
  - fouille de données
  - BDs médicales et scientifiques
  - Intégration d'information
- les BD sont derrière presque tout ce que vous faites sur le Web :
  - recherches Google
  - requêtes Amazon, eBay, etc.
  - organisation de voyage Expedia, TripAdvisor, AirBnB, etc.

La gestion de bases de données, c'est quoi ?

- Trouver (rechercher et interroger) des données
- Mettre à jour et modifier des données
- S'assurer de la cohérence des données
- Protéger les données
  - des accès interdits (contrôle d'accès)
  - des pannes
  - des autres programmes et utilisateurs (contrôle de la concurrence)

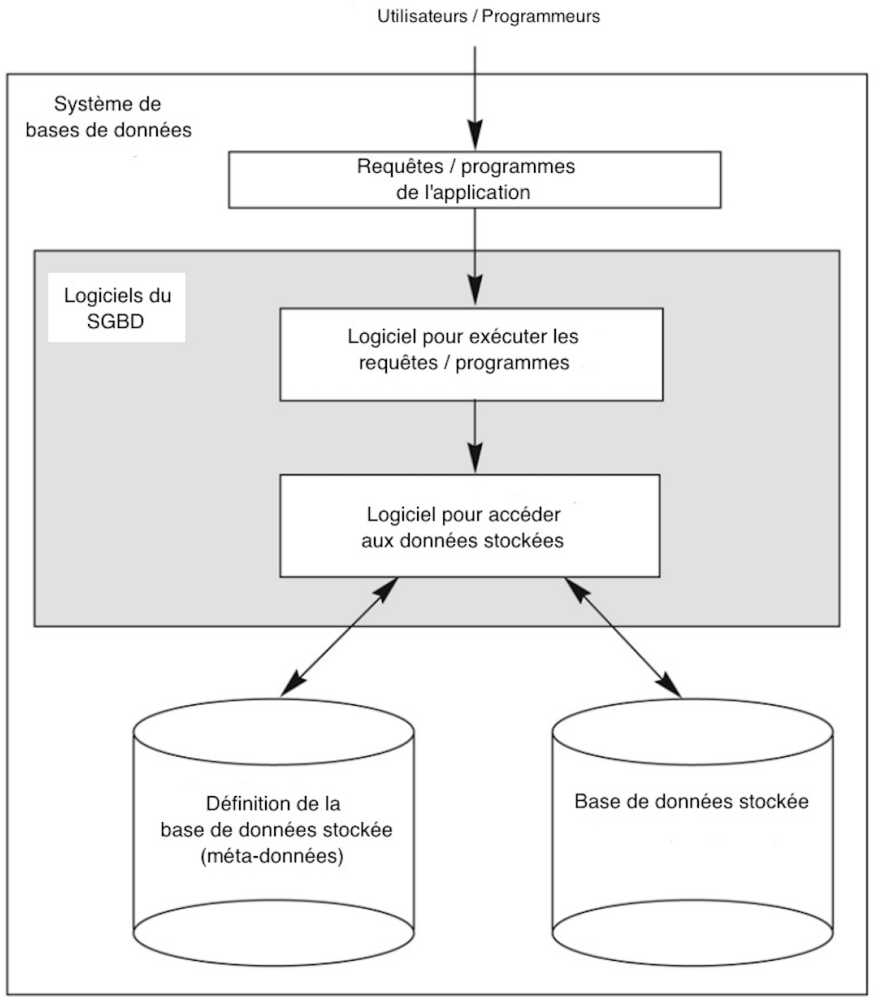
Trouver des données ?

- Requête : trouver le vol Air France pour Tahiti le moins cher entre le 25 décembre et le 16 Janvier.
- Comment trouver ça en utilisant un système de gestion de fichiers ?
  - difficile...
- Beaucoup plus simple en utilisant un Système de Gestion de Bases de Données (SGBD) !

Une base de données, c'est quoi ?

- **Base de données : collection importante de données**
  - Exemple : bases de données de clients, produits, vols, etc
- **Une BD modélise d'habitude (une partie d')une entreprise ou d'un organisme**
  - Entités (exemple : vols, avions, pilotes)
  - Relations (exemple : Le pilote Annie Cavarero assure le vol numéro 105)
- **Un système de gestion de bases de données (SGBD) est un pack de logiciels qui facilite la création et l'utilisation de la base de données.**
- Multiplicité d'éditeurs de logiciels : IBM, Sybase, Oracle, Microsoft, etc

Système de bases de données : environnement simplifié



Pourquoi utiliser un SGBD

- **Nature auto-descriptive du système de bases de données** :
  - un "catalogue" de SGBD stocke la description d'une BD particulière (e.g., structures de données, types et contraintes)
  - cette description s'appelle "méta-données"
- **Indépendance des données**
  - pas besoin de savoir comment la BD est implémentée pour accéder aux données

Pourquoi utiliser un SGBD - L'indépendance des Données

- **Le fonctionnement des applications est indépendant de la manière dont les données sont structurées et stockées** :
  - changement de l'ordre des enregistrements
  - ajout ou modification de colonnes
  - ajout ou modification d'indexes
- **Les requêtes ne changent pas lorsque les structures physiques changent**

Un des plus gros avantages des SGBD

Pourquoi utiliser un SGBD ?

- **Accès efficace**
  - requêtes optimisées
- **Réduction du temps passé à développer des applications**
  - les requêtes peuvent être exprimées de manière déclarative, pas besoin d'indiquer comment les exécuter
- **Intégrité et sécurité des données**
  - certaines contraintes sur les données sont imposées de manière automatique

Pourquoi utiliser un SGBD - Cohérence des Données

- **Contraintes sur les données**
  - tous les étudiants doivent avoir un numéro étudiant unique (INE)
  - deux étudiants ne peuvent pas avoir le même INE (unicité)
  - un étudiant ne peut avoir qu'une seule note par cours
  - etc.

Pourquoi utiliser un SGBD ?

- **Accès concurrent, récupération des pannes**
  - une multitude d'utilisateurs peuvent accéder à la BD en même temps sans interférence
- **Rapidité - même quand le volume des données est massif**

Systèmes de gestion de bases de données (SGBD)

- Masses de données persistantes
  - téraoctets de données survivant à l'exécution des programmes
- Stockage et accès multi-utilisateurs
  - contrôle de la concurrence
- Sécurité
  - résistance aux pannes
    (hardware, software, courant, utilisateurs malveillants)
- Facilité d'utilisation
  - opérations sur les données indépendantes de l'implémentation physique, langages de requête de haut niveau (déclaratifs)
- Efficacité
  - milliers de requêtes et mises à jour par seconde
- Fiabilité
  - 99,9999% de fiabilité (e.g., systèmes bancaires)

Approche Bases de Données : Séparation en couches indépendantes

- Séparation du problème de la gestion de données en trois « couches » :
  - Externe – Traitements (calcul, affichage, ...) ⇒ Programmes
  - Logique – Représentation logique des données ⇒ SGBD
  - Interne – Représentation physique des données ⇒ SGBD
- Couche = ensemble de sous-problèmes bien définis :
  - Indépendance traitements/représentation logique des données : vues externes cachent les détails de l'organisation logique
  - Indépendance représentation logique/représentation physique : schéma logique cache les détails du stockage physique (organisation sur disque, index, ...)

Utilisateurs d'un SGBD

- **Utilisateur final** :
  - accède la BD par des formes d'écran, des interfaces applicatives ou, pour les plus experts, des requêtes
- **Développeur d'applications** :
  - construit (avec l'utilisateur) le schéma conceptuel
  - définit et gère le schéma logique et les vues
  - conçoit et implémente des applications qui accèdent la BD
- **Administrateur BD** :
  - gère le schéma physique et règle les performances, charge et organise la BD
  - gère la sécurité et la fiabilité

Langages et interfaces d'un SGBD

- **Langages de conception** : E/A (Entité/Association), UML
  - Utilisation : conception haut-niveau d'applications (données et traitements)
- **Langages base de données** : SQL, XQuery, SPARQL, ...
  - langages déclaratifs : l'utilisateur spécifie quoi (et non comment)
  - puissance d'expression limitée (par rapport à un langage de programmation comme C ou Java)
  - utilisation : définition schémas, interrogation et mises-à-jour, administration
- **Langages de programmation** : PL/SQL, Java, PHP, ...
  - langages impératifs avec une interface SQL
  - langage complet (au sens d'Alan Turing)
  - utilisation : programmation d'applications complètes

Langages BD (SQL)

Langage de Définition de Données (LDD)

- pour définir les schémas externes (vues), logiques et physiques

  ex : CREATE TABLE CLIENT(NOM varchar, TEL integer) ;

Langage de Manipulation de Données (LMD)

- langage déclaratif pour interroger (langage de requêtes) et mettre à jour les données

  ex : SELECT NOM FROM CLIENT ;

  INSERT INTO CLIENT VALUES(Dupont, 0143270771) ;

- peut être autonome (par ex. SQL seul) ou intégré dans un langage de programmation, à travers une API (Application Programming Interface) comme JDBC (Java DataBase Connectivity)

Modèles de données

Modèle de données
=
langage + sémantique pour représenter et manipuler des données

- Modèle conceptuel *: conception*

  - *structuration haut-niveau (conceptuelle)* de l'information (pas d'opérations)
  - modèle ***entité-association (E/A)***, UML, Merise, …
- Modèle logique : *conception* et *développement*
  - *définition et utilisation* des données dans le SGBD
  - modèle hiérarchique, réseau, ***relationnel***, objet

```
R(A,B);
select A from R where B=2;
```

- Modèle physique : *administration*
  - *organisation physique des données et implantation des opérations*
  - modèles de stockage sur disque, indexes, algorithmes …

```
use index RI;
read record r;
```

Le modèle relationnel

- une base de données se compose de tables (relations)
- les colonnes de chaque table sont nommées par des attributs
- chaque attribut est associé à un domaine (ensemble de valeurs admissibles)
- les données dans chaque table sont constituées par l'ensemble des rangées (tuples) fournissant des valeurs pour les attributs
- pas d'ordre sur les tuples (relations = ensembles non ordonnés)
- (en général) ordre sur les valeurs des attributs dans un tuple

Exemple : la base de données "Air France"

PILOTE

PLNUM	PLNOM	PLPRENOM	VILLE	SALAIRE
1	MIRANDA	SERGE	PARIS	21000
2	LETHANH	NAHN	TOULOUSE	21000
3	TALADOIRE	GILLES	NICE	18000
4	BONFILS	ELIANE	PARIS	17000
5	LAKHAL	LOTFI	TOULOUSE	19000
6	BONFILS	GERARD	PARIS	18000
7	MARCENAC	PIERRE	NICE	17000
8	LAHIRE	PHILIPPE	LYON	15000
9	CICCHETTI	ROSINE	NICE	18000
10	CAVARERO	ANNIE	PARIS	20000

AVION

AVNUM	AVNOM	CAPACITE	LOCALISATION
1	A300	300	NICE
2	A310	300	NICE
3	B707	250	PARIS
4	A300	280	LYON
5	CONCORDE	160	NICE
6	B747	460	PARIS
7	B707	250	PARIS
8	A310	300	TOULOUSE
9	MERCURE	180	LYON
10	CONCORDE	160	PARIS

VOL

VOLNUM	*PLNUM*	*AVNUM*	VILLEDEP	VILLEARR	HEUREDEP	HEUREARR
100	1	1	NICE	TOULOUSE	11:00:00	12:30:00
101	1	8	PARIS	TOULOUSE	17:00:00	18:30:00
102	2	1	TOULOUSE	LYON	14:00:00	16:00:00
103	5	3	TOULOUSE	LYON	18:00:00	20:00:00
104	9	1	PARIS	NICE	06:45:00	08:15:00
105	10	2	LYON	NICE	11:00:00	12:00:00
106	1	4	PARIS	LYON	08:00:00	09:00:00
107	8	4	NICE	PARIS	07:15:00	08:45:00
108	1	8	NANTES	LYON	09:00:00	15:30:00
109	8	2	NICE	PARIS	12:15:00	13:45:00
110	9	2	PARIS	LYON	15:00:00	16:00:00
111	1	2	LYON	NANTES	16:30:00	20:00:00
112	4	5	NICE	LENS	11:00:00	14:00:00
113	3	5	LENS	PARIS	15:00:00	16:00:00
114	8	9	PARIS	TOULOUSE	17:00:00	18:00:00
115	7	5	PARIS	TOULOUSE	18:00:00	19:00:00

Exemple : la table "Avion"

AVNUM	AVNOM	CAPACITE	LOCALISATION
1	A300	300	NICE
2	A310	300	NICE
3	B707	250	PARIS
4	A300	280	LYON
5	CONCORDE	160	NICE
6	B747	460	PARIS
7	B707	250	PARIS
8	A310	300	TOULOUSE
9	MERCURE	180	LYON
10	CONCORDE	60	PARIS

un **tuple** de la relation AVION = une **ligne** de la table AVION

Une **donnée numérique** de la relation AVION = une **valeur de type « entier »** de la table AVION

un **attribut** de la relation AVION = une **colonne** de la table AVION

L'**extension** dela **relation** AVION = La **table** AVION

Schéma d'une relation : "Déclaration de type"

- Nom de la relation
- Ensemble des attributs
- domaine de chaque attribut
- contraintes d'intégrité

Exemple : AVION(AVNUM, AVNOM, CAPACITE, LOCALISATION)

- AVNUM : entier
- AVNOM, LOCALISATION : chaîne de caractères limitée à 30
- CAPACITÉ : entier < 1000

Types d'attribut

- Au moins un attribut par relation
- Chaque attribut d'une relation a un nom
- L'ensemble des valeurs admises pour chaque attribut est appelé le domaine de l'attribut
- Les valeurs d'attributs doivent normalement être atomiques (i.e., indivisibles)
- Jamais deux attributs identiques (nom, domaine)
- Parfois la valeur spéciale NULL est incluse dans le domaine
  NULL= absence de valeur $\neq$ 0 ou chaîne de caractères vide

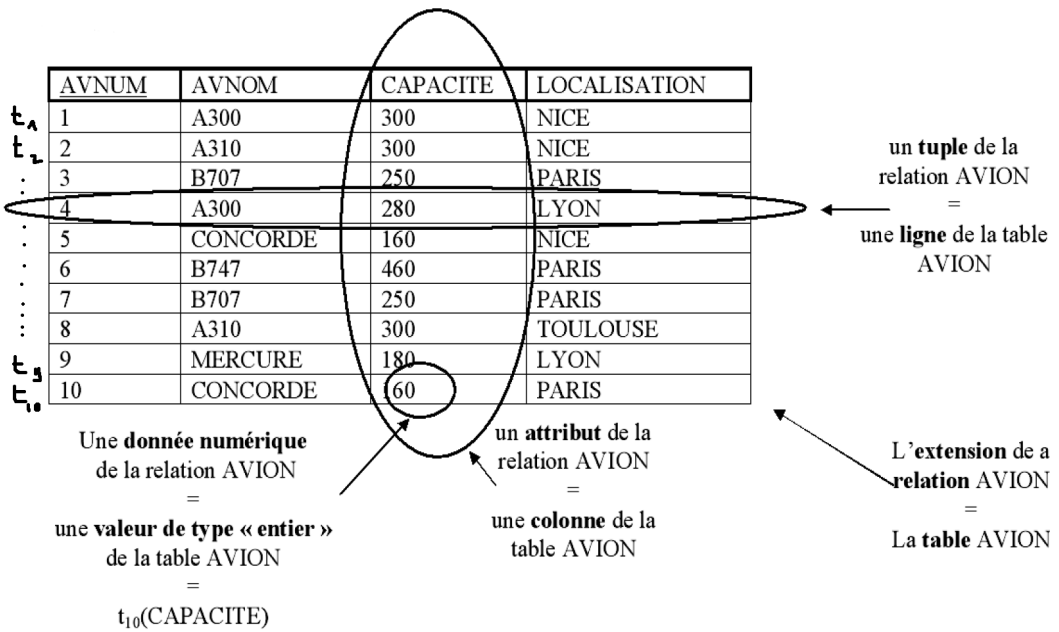
Schéma et Instance

Comparable type / valeur d'une variable dans
les langages de programmation

- Schema : la structure logique de la base de données
  - Exemple : la BD contient des informations au sujet d'avions, de pilotes, de vols et de relations qu'ils entretiennent
  - $\approx$ type de la variable dans un programme
- Instance : le contenu de la base de données à un moment donné
  - $\approx$ valeur de la variable

Les tuples

- On désigne chaque valeur composant un tuple t par $t(A_i) = v_i$: la valeur de l'attribut A_i pour le tuple t
- De même, on désigne par $t(A_u, A_v, \ldots, A_w)$ les sous tuples de t contenant les valeurs des attributs $A_u, A_v, \ldots, A_w$, respectivement



Base de données

- Une base de données se compose de plusieurs relations.
- L'information qui concerne une application est divisée en parties, chaque relation stockant une partie de l'information
  - pilote : stocke l'information sur les pilotes
  - avion : stocke l'information sur les avions
  - vol : stocke l'information sur les vols (dont le pilote et l'avion du vol)
- Stocker toute l'information dans une seule relation comme *airfrance(plnum, plnom, plprenom, ville, salaire, avnum, avnom, capacité, localisation, volnum, villedep, villearr, heuredep, heurearr)*

  est possible mais pas souhaitable :
  entraîne répétition de l'information et valeurs de données nulles

Contraintes d'intégrité

Une contrainte d'intégrité est une condition (logique)
qui doit être satisfaite par les données stockées dans la BD.

Exemple : pour qu'un pilote apparaisse dans la relation Vol il faut qu'il apparaisse dans la relation Pilote.

But : maintenir la cohérence / l'intégrité de la BD :

- Vérifier / valider automatiquement (en dehors de l'application) les données lors des mises à jour : insertions, modifications, effacements
- Déclencher automatiquement des mises à jour entre tables pour maintenir la cohérence globale

Exemples : clefs primaires, clefs étrangères

Clefs primaires

- La clé primaire d'une relation R est l'attribut ou l'ensemble d'attributs (avec le moins d'attributs possible) qui identifie de manière unique chaque tuple de la relation.
- Exemple :
  PLNUM est la clé primaire de PILOTE car (on suppose que) chaque pilote possède un numéro unique.
- Au transparent 23, les clés primaires étaient soulignées.
- Il n'y a qu'une seule clé primaire par relation.

Clefs primaires

La valeur des attributs clefs primaires ne peut jamais être nulle dans aucun tuple de R.

⇒ Clefs primaires utilisées pour identifier les tuples individuels

$t(A) \neq null$ pour tout tuple t d'une instance valide de R, où A est une clef primaire

Note : on peut aussi requérir que des attributs n'appartenant pas à la clef primaire soient non nuls.

Clefs primaires

Exemple de relation avec une clef primaire composée de plusieurs attributs :

RESULTAT

INE	SESSION	UE	NOTE
62B86	2016-2017-1	BD3	6
62B86	2016-2017-2	BD3	19
52934	2016-2017-1	BD3	10
...	...	...	...

- Clef primaire : (INE, SESSION, UE)
- Chaque étudiant ne peut avoir qu'une note par session pour chaque UE.

Dépendance d'inclusion

Exemple : tous les vols sont assurés par des avions déjà recensés dans la base de données.

- Dépendance d'inclusion "E inclus dans F"
  - entre un sous-ensemble d'attributs E d'une relation R et un autre F d'une relation S
  - notée R.E $\subseteq$ S.F
- Si et Seulement Si
  l'ensemble des valeurs de chaque tuple de R pour les attributs de E est inclus dans l'ensemble des valeurs de chaque tuple de S pour les attributs de F.

Clefs étrangères

On a :
VOL.AVNUM⊆AVION.AVNUM
car toute valeur de
VOL.AVNUM est dans
AVION.AVNUM.

AVION

AVNUM	AVNOM	CAPACITE	LOCALISATION
1	A300	300	NICE
2	A310	300	NICE
3	B707	250	PARIS
4	A300	280	LYON
5	CONCORDE	160	NICE
6	B747	460	PARIS
7	B707	250	PARIS
8	A310	300	TOULOUSE
9	MERCURE	180	LYON
10	CONCORDE	160	PARIS

VOL

VOLNUM	*PLNUM*	*AVNUM*	VILLEDEP	VILLEARR	HEUREDEP	HEUREARR
100	1	1	NICE	TOULOUSE	11:00:00	12:30:00
101	1	8	PARIS	TOULOUSE	17:00:00	18:30:00
102	2	1	TOULOUSE	LYON	14:00:00	16:00:00
103	5	3	TOULOUSE	LYON	18:00:00	20:00:00
104	9	1	PARIS	NICE	06:45:00	08:15:00
105	10	2	LYON	NICE	11:00:00	12:00:00
106	1	4	PARIS	LYON	08:00:00	09:00:00
107	8	4	NICE	PARIS	07:15:00	08:45:00
108	1	8	NANTES	LYON	09:00:00	15:30:00
109	8	2	NICE	PARIS	12:15:00	13:45:00
110	9	2	PARIS	LYON	15:00:00	16:00:00
111	1	2	LYON	NANTES	16:30:00	20:00:00
112	4	5	NICE	LENS	11:00:00	14:00:00
113	3	5	LENS	PARIS	15:00:00	16:00:00
114	8	9	PARIS	TOULOUSE	17:00:00	18:00:00
115	7	5	PARIS	TOULOUSE	18:00:00	19:00:00

Clefs étrangères

Exemple de clef étrangère :
AVNUM est clef étrangère de VOL car

- on a la dépendance d'inclusion :
  VOL.AVNUM $\subseteq$ AVION.AVNUM
  (clef étrangère en partie gauche & clef primaire en partie droite)
- et AVNUM est clé primaire de AVION

Contraintes d'intégrité

- Contraintes dites d'intégrité référentielle (relatives aux dépendances d'inclusion)
  si R.E$\subseteq$S.F, alors :
  - quand on insère dans R une nouvelle valeur pour l'attribut E,
  - on doit s'assurer que cette valeur existe dans l'attribut F de S
- Exemple : VOL.AVNUM $\subseteq$ AVION.AVNUM
  - pour ajouter un vol dans la relation VOL,
  - l'avion correspondant doit figurer dans la relation AVION.

Autres types de contraintes

Il existe d'autres types de contraintes plus fines :

- "tous projets cumulés, un même employé ne peut travailler plus de 56h par semaine"
- "le salaire d'un employé ne peut jamais être baissé"

$\Rightarrow$ langages de spécification de contraintes
$\Rightarrow$ triggers, ASSERTIONS

Objectifs de ce cours

Apprendre :

1. **La conception de bases de données**
   - Point de départ : description informelle d'une application
   - Abstraction et optimisation du cahier des charges (modélisation)
   - Création d'entités comprises par le système (extraction des relations de la base de données)
   - Optimisation des relations (normalisation)

2. **L'utilisation d'un SGBD**
   - Ecrire des requêtes dans un langage (SQL) compris par le SGBD (Oracle, PostgreSQL, MySQL, DB2, etc)

Structure prévisionnelle du cours

- **Séances 1 à 3** : modèle relationnel, rudiments de SQL, algèbre relationnel
- **Séances 4** : langage de définition de données, contraintes
- **Séances 5 à 6** : modélisation conceptuelle et formes normales
- **Séances 7** : SQL avancé (agrégation, sous-requêtes)
- **Séance 8** : information incomplète
- **Séance 9** : SQL avancé (vues et tables temporaires, mise à jour de vues, requêtes récursives)
- **Séance 10** : Théorème de Codd
- **Séance 11** : Graphes de propriété : neo4j et Cypher (ou triggers et procédures stockées ?)
- **Séance 12** : Révisions