Polycopié de CS443

— Version 2022/2023 —

Laure Gonnord - (et l'équipe BD de Lyon1)

Premature optimization is the root of all evil (or at least most of it) in programming.

Donald Knuth, *Décembre 1974, Conférence du Prix Turing 1974, Communications of the ACM.*

Afin d'améliorer ce poly n'hésitez pas à me soumettre toute critique, suggestion, remarque ou correction, dans mon casier ou, électroniquement, à l'adresse

`Laure.Gonnord@esisar.grenoble-inp.fr`

Table des matières

Bases de données (CS443)

#1 Introduction, problématiques

Laure Gonnord

Grenoble INP/Esisar

2022-2023

Sources

- équipe BD de Lyon1
- équipe BD de Lille1

Avec leur autorisation, of course.

Contexte

- Nous sommes dans l'ère du BIG DATA
  - Les données sont générées, récoltées, exportées, échangées, vendues
  - Puis mises en entrée d'algorithmes d'analyse ou de modélisation
    - Requêtes, Statistiques, Data Mining, Apprentissage...
  - Enjeux commerciaux, scientifiques, sécuritaires considérables - et moyens qui vont avec.
  - Les données sont un bien précieux de chaque organisation
    - Pour le fonctionnement de quasiment toutes les "applications"
    - Pour améliorer l'efficacité et la qualité de l'entreprise
    - Pour guider les évolutions
    - Pour satisfaire à des exigences légales

Fortes exigeances sur les données

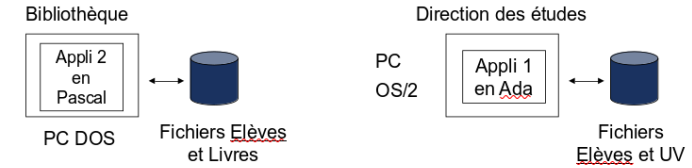
- Stocker les données de façon fiable
- Gérer des accès partagés
- Pouvoir effectuer des recherches complexes et rapides
- être évolutif, réactif, sécurisé
- et bien d'autres choses !

Il faut des outils professionnels : les SGBD

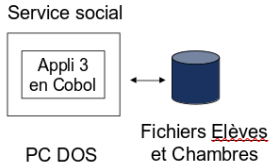
- Exemples : PostgreSQL, Oracle, SQL Server, ...

Plan

L'exemple d'une application



- Approche Fichier :
  - Information conservée dans les différents services.
  - Est-elle sous la forme souhaitée ?
  - Les informations importantes sont dupliquées. Sont-elles identiques ?



Problèmes de l'approche (multi)-fichier

- Fichiers de natures différentes, manipulés par des programmes différents, dans des langages différents avec des formalismes différents.
- Difficulté à saisir les liens entre les données : pas de partage de données entre les utilisateurs ; centralisation physique, mais pas logique.
- Dépendance entre les données et les traitements
- Absence de gestion de la sécurité des données
- Pour une nouvelle application, où sont les informations utiles ?

Comparaison BD/Tableur - à raffiner en TP

Différences sur...	Tableur	Base de données
Utilisation principale	Calculs	Gestion et traitement des données
Structuration des données	Aucune	Structuration et cohérence forte
Contrôles d'intégrité des données	Aucuns	Vérification stricte des valeurs possibles de chaque donnée
Accès aux données	Mono-utilisateur	Multi-utilisateurs
Confidentialité des données	Aucun contrôle	Vérification des droits d'accès de chaque utilisateur
Taille des données	- Une table - Quelques dizaines de lignes	- Plusieurs tables - Plusieurs milliers de lignes par table
Traitement sur les données	Quantitatifs	Qualitatifs et quantitatifs
Interrogations des données	Réalisée par des procédures spécifiques	Langage "universel" : SQL

▶Analyse en TP.

Plan

Abréviations courantes

- SGBD : Système de Gestion de Bases de Données
- DBA : DataBase Administrator

Définitions

Définition

Une Base de Données (BD) est une collection de données électroniques stockées de façon pérenne et reliées entre elles par un contexte applicatif.

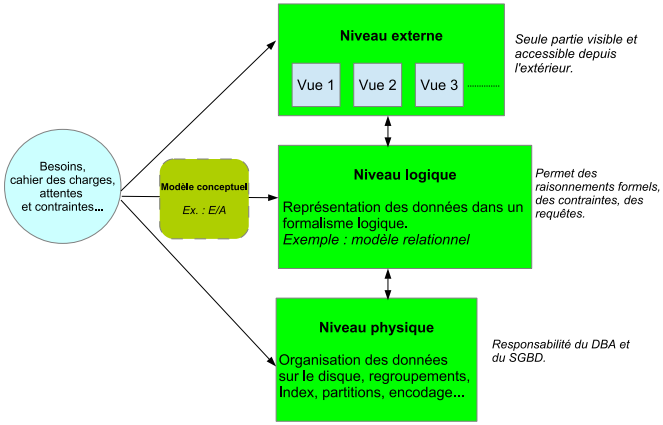
- Quelques exemples :
  - BD de la scolarité de l'université
  - BD des restaurants lyonnais pour la gestion d'un guide
  - BD des ventes d'une entreprise

Contenu d'une BD

Une BD est toujours justifiée par des besoins !

- Ce sont les besoins applicatifs qui guident le contenu, l'organisation, la portée d'une BD.
- La conception des BD est donc un sous-problème de la conception logicielle...
- ... mais plusieurs logiciels pourront accéder à une même base de données !

Niveaux (souhaitables) d'abstraction et d'interactions



Indépendance entre les niveaux

Indépendance logique

- On peut modifier le modèle logique en conservant les mêmes vues externes.
  - Séparation entre BD et applications
  - Permet des évolutions "indépendantes" des deux niveaux

Indépendance physique

- On peut modifier des choix physiques en conservant le modèle logique
  - Sémantique conservée, requêtes inchangées
  - Tout en opérant des réglages physiques

Indépendance (très) peu aboutie dans les modèles "NoSQL".

Résumé : avantage des BD

- Description unique et globale des données
- Langage commun de manipulation.
- Centralisation (pas de duplication, donc pas d'incohérence possible).
- Indépendance entre données et traitements
- Séparation des descriptions logique et physique des données
- Contrôle sémantique des données
- Sécurité (Qui a le droit de lire ou d'écrire, contrôle d'intégrité).
- Partage des données

Plan

Généralités 1/2

Une affaire de spécialiste ?

C'est une étape cruciale à considérer avec soin !

- Conception = modéliser au mieux les données réelles (choix d'un modèle)
- Adaptée aux besoins (réponses aux requêtes)
- Interactions nécessaires avec les experts des données et les concepteurs d'applications
  - Non spécialistes des bases de données
  - Le modèle conceptuel facilitera ces échanges

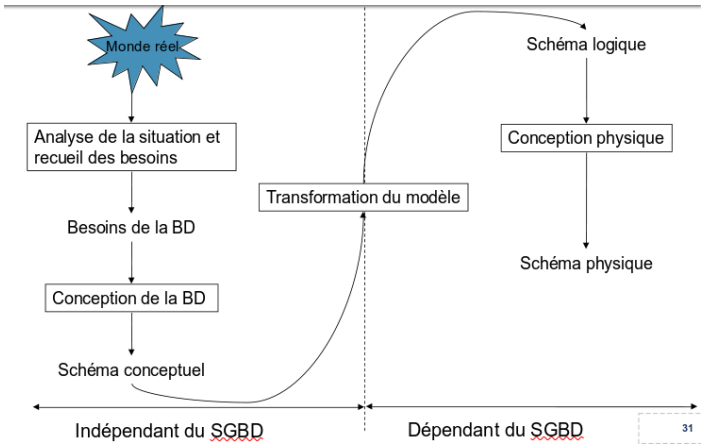
Généralités 2/2

Important

Une bonne conception est certes coûteuse, mais nécessaire !

- Pour les applications : correction, performance.
- Pour la gestion des données : faciliter la maintenance, les évolutions.
- Pour la science des données : sémantique des données, les valeurs manquantes : Une BD bien conçue est bien plus facile à analyser

Étapes de conception



Contenu du cours

- Conception
- Aspects langages et requêtes
- Nouveauté 2022-23 : sous le capot : compilation, distribution
- Si le temps le permet : aspects **sécurité**

Disclaimer

Cours fait par une non-spécialiste. N'hésitez pas à critiquer et reporter d'éventuels bugs/imprécisions/erreurs.

Organisation du cours

- CM : L. Gonnord
- TD, TDM : L. Gonnord et N. Marcos
- Les annonces de cours : mailing via Chamilo
- Les mails : `[CS443]` dans l'objet.

Supports

- Les transparents : sur Chamilo. **ne pas imprimer**
- Les supports de TD : distribués et sur Chamilo
- Les supports de TP : git dédié.

Bases de données (CS443)

#2, Schémas Entités/Associations

Laure Gonnord

Grenoble INP/Esisar

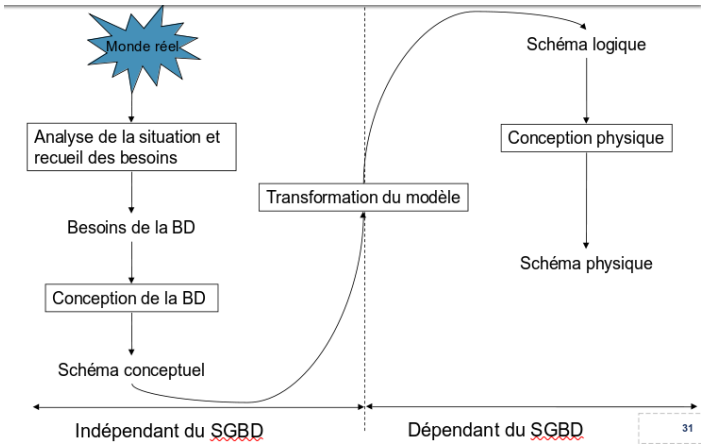
2022-2023

Sources

- équipe BD de Lyon1
- équipe BD de Lille1

Avec leur autorisation, of course.

Étapes de conception



Plan

Un formalisme graphique de travail

- E/R (Entity-Relationship) en anglais.
- Langage graphique pour élaborer un Modèle Conceptuel de Données (MCD)
- Modélise les entités, leurs attributs et leurs associations (interactions)
- Très intuitif, simplifie les échanges autour du cahier des charges
- Bien intégré à la démarche de conception logicielle, proximité avec un diagramme ce classe UML.
- Fourni une documentation précieuse pour l'évolutivité
- Se traduit automatiquement ver le modèle logique (relationnel)

Un formalisme graphique de travail

Attention à ne pas tomber dans la facilité

- Simple langage sans mécanisme de raisonnement
- Des choix dépendent du concepteur, de son expérience
- Pouvoir d'expression limité -> tendance à trop simplifier

Rester proche des besoins

- Démarche itérative
- Bien critiquer ses choix et lister les manques pour la suite
- Importance de connaître les bonnes pratiques

Les élements du langage

- Entités
- Associations
  - Binaires (entre deux entités) ou bien n-aires.
- Attributs
  - Décrivent les entités, où les associations
  - Certains attributs d'entité sont des Identifiants.
- Deux types particuliers d'entités [1], presque indispensables :
  - Entité faibles
  - Entités spécialisées

1. appartiennent au modèle E/A dit "étendu"

Entités et Classes d'entités

- Entité :
  - Une "chose" du monde réel qu'on cherche à modéliser
  - Exemple
    - Un étudiant, un diplôme...
- Classe d'entités
  - Modélisation commune d'entités
  - Exemple : "Un étudiant est représenté par son num, son nom et son prenom."
  - Une entité appartient à une classe d'entité

Par abus de langage, Entité = Classe d'entité.

Associations et Classes d'Association

- Association :
  - Une relation entre deux ou plusieurs entités.
  - Exemple :
    - Tom *est inscrit* en Master Informatique
- Classe d'Association :
  - Décrit de façon commune un ensemble d'associations
    - Exemple : Un étudiant s'inscrit dans un diplôme

Par abus de langage, Assocation = Classe d'association.

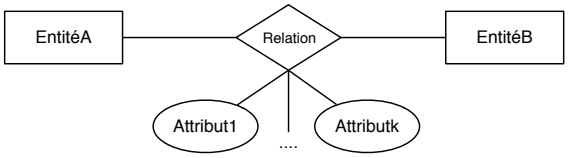
Attributs

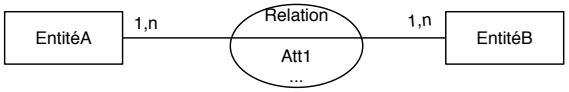
- Attribut :
  - Propriété d'une entité ou d'une association prend ses valeurs dans un domaine de valeurs de type simple (caractère, chaîne de caractères, entier, date).
  - Exemple :
    - L'entité Etudiant a pour attributs Num, Nom et Prenom
    - L'association "inscrit" peut avoir pour attribut la date d'inscription
  - On peut accepter les attributs multivalués, marqués du symbole '*'. Ils peuvent alors prendre une liste de valeurs.
    - Exemple : un document peut avoir plusieurs mots clés, une personne plusieurs prénoms
    - Attention : chaque valeur reste bien atomique.

Au moins un ensemble d'attributs permet d'identifier de façon unique une entité. On souligne cet identifiant [2] .

2. Un attribut multivalué ne peut pas être identifiant

Formalisme de représentation des associations





Connectivité et participation aux associations

Dans une association entre E_1 et E_2, on doit définir :

- A combien d'entité E_2 peut se connecter une entité E_1 et inversement : c'est la connectivité de chaque classe d'entité. Elle vaut soit 1, soit N
- Si chaque entité E_1 (ou E_2) est obligée de participer à l'association : c'est la participation de chaque classe d'entité. Elle vaut soit 0, soit 1.
- Exemples :
  - Un étudiant DOIT être inscrit dans au moins une formation (participation obligatoire)
  - Un étudiant doit être inscrit AU PLUS dans une formation (connectivité simple)
  - Une formation peut avoir PLUSIEURS étudiants (connectivité multiple)
  - Une formation peut n'avoir aucun étudiant (participation optionnelle)

Schéma E/A : cardinalités

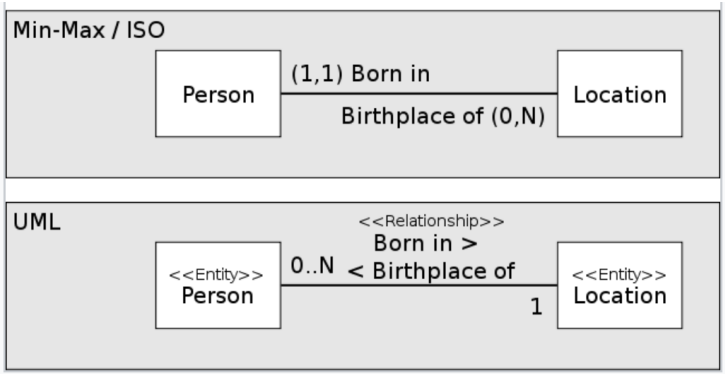
Couples (participation (O ou 1), connectivité (1 ou 'N')), attention deux couples à écrire pour une relation.

- Les valeurs possibles sont : (0,1), (1,1); (0,N), (1,N)

- Min : Correspond à la réponse à la question :
  - combien de fois au moins une entité de A est relié à une entité de B
- Max : correspond à la réponse à la question :
  - combien de fois au plus une entité de A est relié à une entité de B

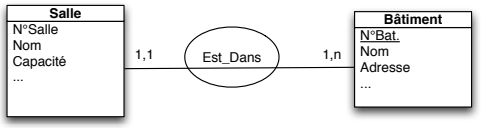
Source du dessin : univ Angers

Schéma E/A : cardinalités - différentes conventions



Various methods of representing the same one to many relationship. In each case, the diagram shows the relationship between a person and a

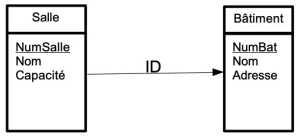
Entités Faibles



Que se passe-t-il s'il y a deux salles 1 dans deux bâtiments différents ?

- L'attribut "N Salle" ne permet pas d'identifier une salle.
- Il faut savoir de quel bâtiment il s'agit.
- Lien existentiel : la salle n'existe que si le bâtiment existe.
- Caractérisé par une flèche ID.

Entités Faibles

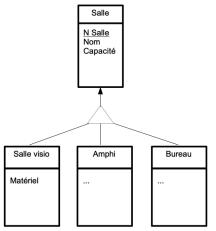


Pour savoir de quelle salle on parle, il faut connaître dans quel bâtiment elle est. Une salle est donc "identifiée" par : un numéro de salle $NumSalle$ (identifiant local) et un numéro de bâtiment $NumBat$.

Spécialisation / Généralisation

- Une entité représente un cas particulier d'une autre entité
- Les entités "filles" héritent des attributs des entités "parents", y compris de l'identifiant.
- Donc il ne faut répéter aucun de ces attributs ni identifiants.
- Exemple :
  - Une salle de visio, en plus des autres, possède un matériel spécifique, des identifiants de connexion, un modes de réservation, etc...
  - Il faut donc les modéliser de façon distincte des autres salles
  - Sans perdre de vue que c'est bien une salle !

Exemple



Attention : contrairement à une entité faible qui possède un identifiant local souligné, une entité spécialisée n'a JAMAIS d'attribut souligné. Son identifiant provient totalement de l'entité "mère" à laquelle elle est reliée.

Associations n-aires

Il est possible de faire interagir trois entités ou plus dans une même association :

- C'est parfois une première intention naturelle
- Mais beaucoup de "fausses" associations n-aires, faites un peu rapidement
- Tous les sous-ensembles d'entités peuvent-ils se répéter ?
- Si la réponse est non, utiliser des associations d'associations (aggrégation)

Plan

Quelques exercices 1/3

On considère la base de données d'une galerie d'art. Elle garde des informations sur les artistes (un nom unique, un ville de naissance, un style). Chaque oeuvre est réalisée par un.e artiste, avec une année de réalisation, un titre unique, et un prix. Elle appartient à un ou plusieurs groupes d'œuvres qu'on crée pour les classer (portraits, oeuvre de Picasso,. . .). Chaque groupe est identifié par son nom. La galerie garde des informations sur ses clients (nom unique, adresse, préférences.) Chaque client peut effectuer des commandes à des dates différentes ; chaque commande porte sur une ou plusieurs œuvres.

Quelques exercices 2/3

A l'université, les enseignant.es (identifiés par PID) donnent des cours (CID). Un enseignant.e peut donner plusieurs cours ; un cours est donné par exactement un enseignant.e. On précise le semestre (S4, S5, S6). Faites un deuxième schéma pour qu'un enseignant.e puisse redonner un même cours à un semestre différent. Faites un troisième schéma pour qu'un cours, le même trimestre, ne soit affecté qu'à un seul enseignant.e. Finalement, on ajoute que lorsqu'un enseignant.e donne un cours, c'est toujours le même !

Quelques exercices 3/3

Une entreprise stocke les informations sur ses employé.e.s. Iels ont un numéro de sécurité sociale qui les identifie, un salaire et un numéro de téléphone. Les départements de l'entreprise sont identifiés par leur numéro, possèdent un nom et un budget. Les employé.e.s peuvent avoir des enfants avec un nom et une date de naissance (on fera ici l'hypothèse qu'un seul parent travaille dans l'entreprise). Chaque employé.e. travaille dans des départements, chaque département est dirigé par un.e employé.e.

TODO

Préparation du TD 1

Les schémas A/E des deux derniers exemples sont à finir avant le prochain TD.

Bilan

1. Conception/Schéma conceptuel E/A

2. Exercices

Bases de données (CS443)

#3, Modèle relationnel : modélisation

Laure Gonnord

Grenoble INP/Esisar

2022-2023

Sources

- équipe BD de Lyon1
- équipe BD de Lille1

Avec leur autorisation, of course.

Plan

Qu'est ce qu'un modèle de données ?

Un formalisme commun pour représenter et interroger les données

- Une structure pour représenter les données
- Des contraintes pour garantir la sémantique du cahier des charges
- Des langages pour interroger et modifier les données

Les niveaux logiques et physiques sont dédiés à un modèle de données commun. Le niveau externe est plus libre, peu correspondre à des exportations vers d'autres modèles ou formats.

Les principaux modèles de données

- *Modèle relationnel*
  - *Structure* : ensembles de relations, qui sont des ensembles de *tuples*.
  - *Contraintes* : principalement les clés primaires (identifiants de tuples, dépendances fonctionnelles) et les clés étrangères (liens entre les relations, dépendances d'inclusion)
  - *Manipulation* : algèbre relationnelle, calcul relationnel, SQL, clauses de Horn sans récursion.
- *Modèle déductif*
  - *Structure* : celle du modèle relationnel à laquelle on ajoute des règles de déduction.
  - *Contraintes* : les mêmes que le modèle relationnel
  - *Manipulation* : langages logiques comme *Datalog*. Contrairement aux langages du modèle relationnel, il admet la récursivité.

Les principaux modèles de données

- *Modèle de graphe (e.g., RDF)*
  - *Structure* : graphe orienté et étiqueté, on enregistre les triplets $(s; p; o)$
  - *Contraintes* : un identifiant pour chaque nœud, un mécanisme de référence entre des nœuds
  - *Manipulation* : parcours de graphes, SPARQL.
- *Modèle hiérarchique (e.g., XML)*
  - *Structure* : arborescente (forêt d'arbre)
  - *Contraintes* : un identifiant pour chaque nœud, un mécanisme de référence entre des nœuds
  - *Manipulation* : navigation hiérarchique, XPATH, XQUERY.

Les principaux modèles de données

- *Modèle objet*
  - *Structure* : logique objet, soit des classes, des objets, des attributs et des méthodes. Peut être vu comme un graphe orienté.
  - *Contraintes* : identifiant pour les objets, référence entre objets.
  - *Manipulation* : extensions de SQL comme OSQL ou OQL.
- *Modèle Entité/Association*
  - *Structure* : Entités (avec des attributs) et associations entre des entités.
  - *Contraintes* : identifiants, cardinalités sur les associations
  - *Manipulation* : aucun (c'est un langage de modélisation, qui n'est pas implémenté tel que).

Les bases "de production" sont très très majoritairement relationnelles. Des représentations objet ou XML sont bien souvent intégrées comme des "surcouches" de SGBD relationnels.

Notations

- $\{A; B; C|D\}$ sera noté ABC
- Les notations ABC et BCA sont équivalentes puisque l'ordre n'a pas d'importance.
- L'ensemble $AABC$ est le même que ABC, puisque l'élément A ne peut existe qu'en un seul exemplaire.
- Soit les ensemble $X = ABC$ et $Y = BD$, alors leur union $X \cup Y$ sera noté $XY = ABCD$.

Le modèle relationnel : Intuition

Étudiants	*NUMETUD*	*NOMETUD*	*PRENOMETUD*	*AGE*	*FORMATION*
	28	Codd	Edgar	20	3
	32	Armstrong	William	20	4
	53	Fagin	Ronald	19	3
	107	Bunneman	Peter	18	3

Enseignants	*NUMENS*	*NOMENS*	*PRENOMENS*	*GRADE*
	5050	Tarjan	Robert	PR
	2123	Mannila	Heikki	MCF
	3434	Papadimitriou	Spiros	PR
	1470	Bagan	Guillaume	CR

Encadre	*NUMENS*	*NUMETUD*	*DATE*
	5050	53	2005
	3434	28	2020
	5050	28	2015
	2123	32	2019

TABLE – Exemple de bases de données

Plan

Que veut-on décrire ?

Des tables 2D !

Plan

Vocabulaire : informellement

Sur les tables :

- Une relation = une table à deux dimensions
- Une colonne = un attribut
- En-tête du tableau = description de la relation (schéma)
- Une ligne = un tuple ou un n-uplet
- Ensemble des lignes (= contenu de) la relation

Le modèle relationnel : Intuition

Étudiants	*NUMETUD*	*NOMETUD*	*PRENOMETUD*	*AGE*	*FORMATION*
	28	Codd	Edgar	20	3
	32	Armstrong	William	20	4
	53	Fagin	Ronald	19	3
	107	Bunneman	Peter	18	3

Enseignants	*NUMENS*	*NOMENS*	*PRENOMENS*	*GRADE*
	5050	Tarjan	Robert	PR
	2123	Mannila	Heikki	MCF
	3434	Papadimitriou	Spiros	PR
	1470	Bagan	Guillaume	CR

Encadre	*NUMENS*	*NUMETUD*	*DATE*
	5050	53	2005
	3434	28	2020
	5050	28	2015
	2123	32	2019

TABLE – Exemple de bases de données

Le modèle relationnel : structure

- Soit $\mathcal{U}$, un ensemble infini dénombrable de *noms d'attributs* ou simplement *attributs*, appelé univers.
- Soit $\mathcal{D}$ un ensemble infini dénombrable de constantes (ou valeurs).
- Soit $A \in \mathcal{U}$ un *attribut*, le domaine de A est un sous-ensemble de $\mathcal{D}$, noté $DOM(A)$.

Schémas de relations et de bases de données

- Un schéma de relation R est un ensemble fini d'attributs (donc $R \subseteq \mathcal{U}$).
- Un schéma de base de données $\mathbf{R}$ est un ensemble fini de schémas de relation.

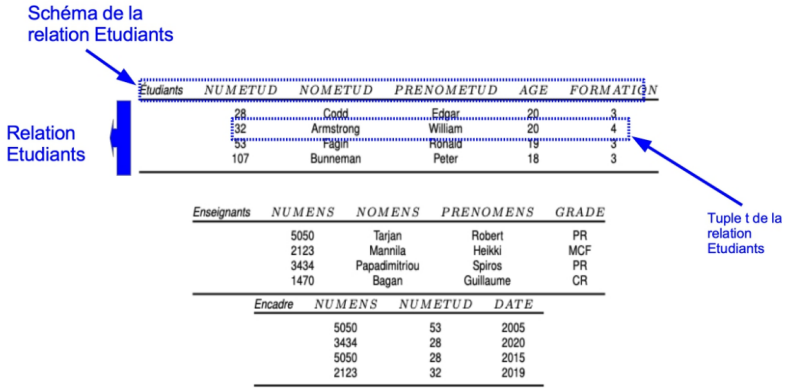
Le modèle relationnel : structure

Tuple, Relation et Base de Données

- Soit $R = A_1...A_n$ un schéma de relation. Un tuple sur R est un élément du produit cartésien $DOM(A_1) \times \ldots \times DOM(A_n)$.
- Une relation r sur R (appelée aussi instance ou vulgairement table) est un ensemble *fini* de tuples.
- Une base de données **d** sur un schéma de base de données $\mathbf{R} = \{R_1, ..., R_n\}$ est un ensemble de relations $\{r_1, ..., r_n\}$ définies sur les schéma de relation de $\mathbf{R}$.

- Si t est un tuple défini sur un schéma de relation R, et X un sous-ensemble de R, on peut restreindre t à X en utilisant la projection, notée $t[X]$ qui est la restriction de r à X.

Synthèse



Etudiants

NUMETUD	NOMETUD	PRENOMETUD	AGE	FORMATION
28	Codd	Edgar	20	3
32	Armstrong	William	20	4
53	Fagin	Ronald	19	3
107	Bunneman	Peter	18	3

Enseignants

NUMENS	NOMENS	PRENOMENS	GRADE
5050	Tarjan	Robert	PR
2123	Mannila	Heikki	MCF
3434	Papadimitriou	Spiros	PR
1470	Bagan	Guillaume	CR

Encadre

NUMENS	NUMETUD	DATE
5050	53	2005
3434	28	2020
5050	28	2015
2123	32	2019

- Les 3 schémas de relations forment le schéma de la BD.
- les 3 relations forment la BD.

Plan

Définition

Clé

La (une) clé d'une relation est un ensemble minimal d'attributs dont chaque valeur détermine un tuple unique dans toute extension de la relation. Il ne doit pas exister plusieurs lignes d'une relation avec la même valeur de clé.

Considérons la relation `EMPLOYE (Nom, Prénom, Adresse, Ville)` :

Durand	Alain	3 rue Rose	Paris
Noël	Anne	19 rue Haute	Paris
Remy	André	46 rue Vilaine	Nantes
Durand	Etienne	10 rue Limite	Nice

Problème : deux employés existent sous le même nom ! la clé 'nom' de la relation 'employé' ne permet pas d'identifier un client de manière unique.

Problème d'identification unique : solution

Solution : gérer un numéro d'employé ce qui permettra une identification totale d'un employé :
`EMPLOYE (NoEmpl, Nom, Prénom, Adresse, Ville)`

101	Durand	Alain	3 rue Rose	Paris
102	Noël	Anne	19 rue Haute	Paris
120	Remy	André	46 rue Vilaine	Nantes
131	Durand	Etienne	10 rue Limite	Nice

Et hop, chaque valeur de `noempl` n'est associé qu'à un seul employé !

Exemple de détermination de clés : livres

- Client (numcli, nom, prenom, adresse)
- Livre (numlivre, titre, auteur, nbexmpl)
- Emprunt (numcli, numlivre, date, retard)

Peut-on identifier un emprunt avec le couple (numcli, numlivre) ?

Exemple de détermination de clés : livres

- Client (numcli, nom, prenom, adresse)
- Livre (numlivre, titre, auteur, nbexmpl)
- Emprunt (numcli, numlivre, date, retard)

Peut-on identifier un emprunt avec le couple (numcli, numlivre) ? Non, si un client a la possibilité d'emprunter deux fois le même livre.

▶**Solution 1** : utiliser en plus la date Clé d'emprunt : (numcli, numlivre, date)

Et si la date ne suffit pas (emprunt de deux fois le même livre le même jour) ?
▶**Solution 2** :

- Identifiant (numemprunt)
- Emprunt (numemprunt, numcli, numlivre, date, retard)

Déterminer une seule clef pour une relation : exemple

Si plusieurs clefs candidates sont possibles, par exemple pour la relation : `Etudiant (n°ss, nom, prenom, adresse, n°filiere, n°inscription)` on peut choisir :

- N°inscription : numero de carte d'étudiant
- N°ss : numéro de sécurité sociale

La clé choisie est appelée **clé primaire**, et les autres clés sont alors des clés **secondaires**.

Primaire, étrangère

Clé étrangère (ou référence)

Une clé étrangère est un champ ou une collection d'attributs dans une relation qui identifie de manière unique une ligne (un élément) d'une autre relation (ou la même, cf le cas précédent).

Notation : on suffixe cette clé avec #.

Contraintes de relation

La valeur nulle (NULL) est une valeur conventionnellement introduite dans une relation pour représenter une information inconnue ou inapplicable.

Ex : pour la relation `Employé (NoEmpl, Nom, Année, Adresse, Téléphone, Nodept)`, on représente un employé à numéro inconnu comme ceci : (10, Durand, 1980, Paris, NULL, 75)

Contrainte de relation : la clé primaire de la relation ne doit pas posséder de valeur nulle.

Usage des clés : liens entre les relations

Les liens entre les clés (étrangères) permettent de faire des liens non ambigus entre les relations (tables) :

101	Durand	Alain	3 rue Rose	Paris
102	Noël	Anne	19 rue Haute	Paris
120	Remy	André	46 rue Vilaine	Nantes
134	Durand	Etienne	10 rue Limite	Nice

L'ensemble des valeurs de la colonne origine de la flèche est inclus dans l'ensemble des valeurs de la clé primaire (cible de la flèche)

54	120	12	30/11/91	N
51	102	12	18/04/90	N
52	101	12	12/10/90	N
53	101	20	26/04/91	0

20	Le prince de Sang mélé	JK Rowling	10
30	Can you keep a secret ?	Sophie Kinsella	4
12	Websphere V3.5 Handbook	Websphere Consulting Team	1

Plan

Modèle relationnel : Langages

- Langages théoriques d'interrogation des données :
  - Langage procédural : algèbre relationnelle
  - Langage déclaratif (logique) : calcul relationnel (tuple ou domaine), Datalog.
- Langage implémenté pour l'interrogation et la manipulation des données
  - Structured Query Langage (SQL)
  - SQL est l'implémentation du calcul relationnel pour la partie interrogation.
  - Mais nombreuses extentions et possibilité d'ajouter des fonctions.

On manipule des relations

Une requête prend une plusieurs relations en entrée, et retourne une relation. On peut donc interroger des sous-requêtes ; toutefois seul Datalog est récursif.

Exemple de requête algébrique

Quel est le prénom et le nom de tous les étudiants ?

$\pi_{PRENOMETUD,NOMETUD}(Etudiants)$

▶On pratiquera en TD.

SQL

Caractéristiques :

- Langage implémenté et universel d'interrogation d'une BD relationnelle.
- "Traduction" du calcul relationnel, donc déclaratif.
- Défini dans les années 80, dernière norme en 92
- Beaucoup d'évolutions : UPSERT, Windows functions, Grouping Set, récursivité (voir la doc !) ...

Se décompose en sous-ensembles :

- DML : Manipulation (màj) et interrogation des données
- DDL : Définition des données (au niveau du schéma)
- DCL : Contrôle des droits des utilisateurs
- TCL : Contrôle des transactions

Exemple de requête SQL

Quel est le prénom et le nom de tous les étudiants ?

```
SELECT PRENOMETUD, NOMETUD FROM Etudiants
```

▶Syntaxe à ne pas connaître. On fournira une doc.

Description des langages, et requêtes

Langages pour les BD relationnelles

dans les prochains cours.

Plan

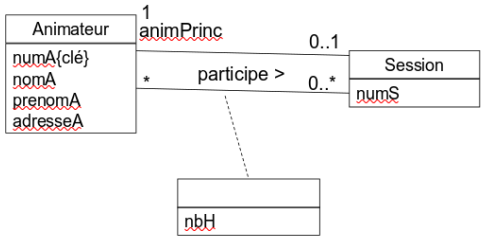
Un algorithme général

- Les entités deviennent des relations
- Les associations sont traduites. . . selon leur type

Cardinalité : rappel

A (min,max) $\rightarrow$ B $\equiv$ "une entité de *A* est reliée min à max fois à une entité de B".

Exemple 1



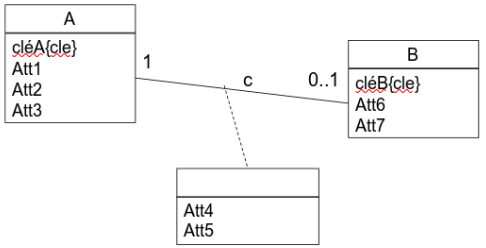
Entités et associations deviennent des relations : observons d'abord les deux entités :

- Animateur(numA, nomA, prenomA, adresseA) : OK !
- Session (numS, **numA#**) pourquoi ?

ne pas regarder l'association participe ici.

Traduction associations cardinalités 1,1 / 0,1

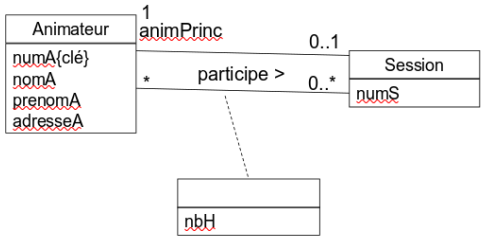
"Une entité de A est relié une fois à une entité de B", d'où le schéma général de traduction :



- A (CléA, Att1, Att2, Att3)
- B (CléB, Att6, Att7, Att4, Att5, CléA#)

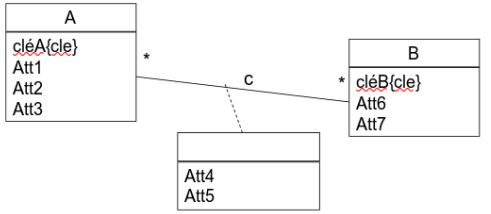
▶Remarquez où vont les attributs de l'association.

Exemple 1 (cont')



- Animateur(num, nomA, prenomA, adresseA) : OK !
- Session (numS, numA#) : OK !
- Participe (numA#, numS#, nbH) pourquoi ? remarquez la clé aussi

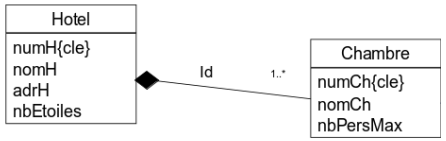
Traduction cardinalités max "étoile" des 2 côtés - cas général



- A (CléA, Att1, Att2, Att3)
- B (CléB, Att6, Att7)
- C (CléA#, CléB#, Att4, Att5)

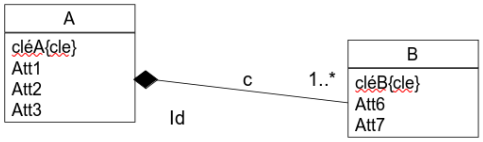
Traduction entité faible - exemple

Une version "hôtel" des salles dans les bâtiments.



- Hôtel(numH, ...)
- Chambre(numH#, numCh, ...)

Traduction des entités faibles - cas général

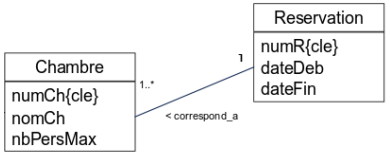


- A (CléA, Att1, Att2, Att3)
- B (CléA#, CléB, Att6, Att7)

Pourquoi ?

Traduction cardinalités 1..* - 1 - exemple

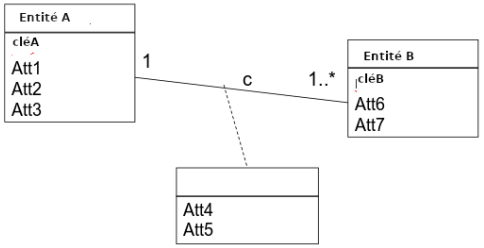
attention au sens de la relation



(avec la clé obtenue pour chambre, nommée `key` ici)

- Chambre(key, nomChambre, nbPersMax)
- Reservation(numR, dateDeb, dateFin)
- Correspond_A(numR#, key#)

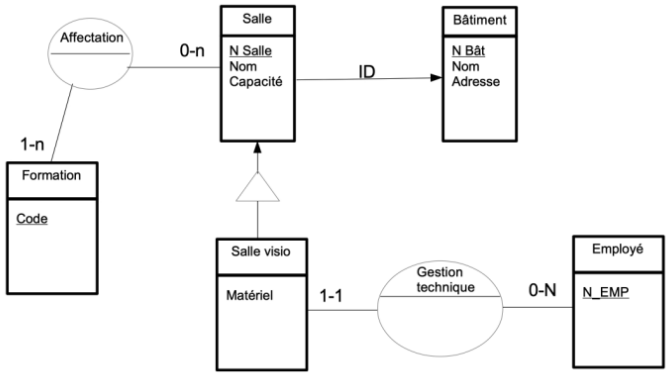
Traduction cardinalités 1..* - 1 - cas général



- A(cleA, ...)
- B(cleB, ..., cleA#, Att4, Att5)

Exercice à préparer pour le TD 1

Traduire vers un schéma relationnel.



Pour aller plus loin

- On vous laisse réfléchir aux relations "spécialisation" et relations ternaires avec cardinalités étoiles (pas si simple).
- Il y a des manières de contraindre le modèle E/A (non vues dans le cadre de ce cours, et donc de traduire ces contraintes (idem).

Bilan

1. Modèle de données
2. Modèle relationnel - définitions
   - Vocabulaire de relations
   - Notion de clé
3. Langages pour le modèle relationnel
4. Traduction entité/association vers relationnel

Bases de données (CS443)

#4, Modèle relationnel: algèbre relationnelle

Laure Gonnord

Grenoble INP/Esisar

2022-2023

Sources

- équipe BD de Lyon1
- équipe BD de Lille1

Avec leur autorisation, of course.

Le modèle relationnel

Rappel, on a une structuration en tables.

Étudiants

NUMETUD	NOMETUD	PRENOMETUD	AGE	FORMATION
28	Codd	Edgar	20	3
32	Armstrong	William	20	4
53	Fagin	Ronald	19	3
107	Bunneman	Peter	18	3

Enseignants

NUMENS	NOMENS	PRENOMENS	GRADE
5050	Tarjan	Robert	PR
2123	Mannila	Heikki	MCF
3434	Papadimitriou	Spiros	PR
1470	Bagan	Guillaume	CR

Encadre

NUMENS	NUMETUD	DATE
5050	53	2005
3434	28	2020
5050	28	2015
2123	32	2019

Maintenant, on veut **manipuler ces tables**, ie faire des **requêtes**.

Le modèle relationnel - requêtes

Définition

Une requête est :

- une expression ensembliste qui calcule une relation (en algèbre relationnelle)
- une instruction (**commande**) qui retourne une table (en SQL).

Plan

Formalisme !

Attention formalisme

Mais ce n'est rien qu'une formalisation des opérations sur les tables !

Dans le modèle relationnel, les tables sont supposées construites.

Algèbre relationnelle : opérations sur les relations

Soient des relations r_i définies sur des schémas R_i :

- La projection $\pi_X(r)$ ne conserve que les attributs de X ($\subseteq R$) ;
- La sélection $\sigma_C(r)$ filtre les tuples de r (lignes) suivant la condition C.
- Le renommage $\rho[X/X']$ renomme l'attribut X' en X.
- La jointure $r_1 \bowtie r_2$ "combine" les tuples de r_1 et r_2.

On fera attention aux domaines de définition.

Projection $\pi_X(r)$

employe

NoEmp	Nom	Année	NoDep
1045	Dupond	1978	03
2067	Dupont	1965	06
0456	Martin	1981	03
0278	Martin	1987	05
0789	Blanc	1981	06

$annee\_naissance = \pi_{Nom,Annee}(employe)$ retourne (vaut) la relation :

Projection $\pi_X(r)$

employe

NoEmp	Nom	Année	NoDep
1045	Dupond	1978	03
2067	Dupont	1965	06
0456	Martin	1981	03
0278	Martin	1987	05
0789	Blanc	1981	06

$annee\_naissance = \pi_{Nom,Annee}(employe)$ retourne (vaut) la relation :

annee_naissance

Nom	Année
Dupond	1978
Dupont	1965
Martin	1981
Martin	1987
Blanc	1981

Sélection $\sigma_C(r)$

employe

NoEmp	Nom	Année	NoDep
1045	Dupond	1978	03
2067	Dupont	1965	06
0456	Martin	1981	03
0278	Martin	1987	05
0789	Blanc	1981	06

Liste des employé(e)s du département 03 né(e)s avant 1980 :

Sélection $\sigma_C(r)$

employe

NoEmp	Nom	Année	NoDep
1045	Dupond	1978	03
2067	Dupont	1965	06
0456	Martin	1981	03
0278	Martin	1987	05
0789	Blanc	1981	06

Liste des employé(e)s du département 03 né(e)s avant 1980 :
$ancien = \sigma_{annee<1980 \wedge NoDep=03}(employe)$

ANCIEN

NoEmp	Nom	Annee	NoDep
1045	Dupond	1978	03

Renommage $\rho[X/X'](r)$

Ce n'est rien d'autre qu'une substitution !

employe

NoEmp	Nom	Année	NoDep
1045	Dupond	1978	03
2067	Dupont	1965	06
0456	Martin	1981	03
0278	Martin	1987	05
0789	Blanc	1981	06

Renommons deux colonnes :

Renommage $\rho[X/X'](r)$

Ce n'est rien d'autre qu'une substitution !

employe

NoEmp	Nom	Année	NoDep
1045	Dupond	1978	03
2067	Dupont	1965	06
0456	Martin	1981	03
0278	Martin	1987	05
0789	Blanc	1981	06

Renommons deux colonnes : $new = \rho[IdEmp/NoEmp, Dpt/NoDep](employe)$

employe

IdEmp	Nom	Année	Dpt
1045	Dupond	1978	03
2067	Dupont	1965	06
0456	Martin	1981	03
0278	Martin	1987	05
0789	Blanc	1981	06

Jointure

Pour la jointure $\bowtie_C$ il nous faut d'abord la définition du produit cartésien, donc passons d'abord aux opérations ensemblistes.

Algèbre Relationnelle (suite)

Puisque les relations sont des ensembles de tuples, on bénéficie en plus de tous les opérateurs ensemblistes.

- A condition d'avoir $R = S$:
  - Différence ($r_1 \setminus r_2$).
  - Intersection ($r_1 \cap r_2$).
  - Union ($r_1 \cup r_2$).
- A condition d'avoir $R \cap S = \emptyset$
  - Produit cartésien ($r_1 \times r_2$). La relation obtenue est sur le schéma $R_1 \cup R_2$.

Pour les conditions sur le schéma, on peut les "forcer" par le renommage préalable

Union, Intersection, Différence

EMPLOYE 1

NoEmp	Nom	Annee	NoDep
1045	Dupond	1978	03
2067	Dupont	1965	06
0456	Martin	1981	03

EMPLOYE 2

NoEmp	Nom	Annee	NoDep
1045	Dupond	1978	03
0278	Martin	1987	05
0789	Blanc	1981	06

Que valent $EMPLOYE1 \oplus EMPLOYE2$ avec $\oplus \in \{\cup, \cap, \setminus\}$?

Domaines

On fera attention aux domaines de définition des opérations.

Produit cartésien

Définition

Le produit cartésien de deux relations $r_1 \times r_2$ (de cardinal n_i) est une relation sur le schéma $R_1 \cup R_2$ (les schémas sont supposés disjoints). Les tuples de la relation sont la concaténation d'un élément/tuple de r_1 et d'un élément/tuple de r_2.

On obtient alors une table de taille de $n_1 * n_2$ tuples

Source : équipe BD LIP6

Produit cartésien, exemple

Employe3

NoEmp	Nom	Annee	NoDep
2067	Dupont	1965	06
0456	Martin	1981	03

Departement

NoDep2	Intitule	Taille
03	Comptabilité	6
06	Informatique	10

$RES = EMPLOYE3 \times DEPARTEMENT$ fournit la table :

NoEmp	Nom	Annee	NoDep	NoDep2	Intitule	Taille
2067	Dupont	1965	06	03	Comptabilité	6
2067	Dupont	1965	06	06	Informatique	10
0456	Martin	1981	03	03	Comptabilité	6
0456	Martin	1981	03	06	Informatique	10

Produit cartésien, exemple

Employe3

NoEmp	Nom	Annee	NoDep
2067	Dupont	1965	06
0456	Martin	1981	03

Departement

NoDep2	Intitule	Taille
03	Comptabilité	6
06	Informatique	10

$RES = EMPLOYE3 \times DEPARTEMENT$ fournit la table :

Jointure (def)

Soient :

- deux tables R et S avec $R \cap S = \emptyset$,
- F une formule logique comportant au moins un atome $A_i \oplus B_j$ ($\oplus$ opérateur de comparaison) avec A_i (resp B_j) attribut de R (resp. S)

Alors la jointure est le sous-ensemble des tuples du produit cartésien $R \times S$ qui satisfont F, cad :

$$R \bowtie_F S =_{def} \sigma(R \times S)$$

Types de jointures

- équijointure : la formule F n'utilise que l'égalité
- θ-jointure : la formule F utilise (aussi) des comparaisons
- la jointure dite naturelle.

La jointure naturelle se fait sur des schémas comportant des attributs en commun :

$$R(\mathbf{X}, Y) \bowtie_{JN} S(\mathbf{X}, Y') =_{def} \pi_{S.X,Y,Y'}\left(\sigma_{R.X=S.X}\left(\underline{\rho[X/R.X](R) \times \rho[X/S.X](S)}\right)\right)$$

- Le renommage souligné sert à effectuer un produit cartésien correct.
- La partie en bleue filtre celui-ci pour ne garder que les lignes dans lesquelles $S.X = R.X$
- La projection sert à éliminer une des deux colonnes $S.X$ (ici) ou $R.X$
- (il resterait à renommer l'autre colonne)

Exemple d'équi-jointure

Employe

n°empl	nom_e	ville_e	age_e	n°chef_e
141	dupond	paris	40	500
36	durand	tours	40	500
251	parent	agen	25	60

Chef

n°chef	nom_c	age_c
500	albert	50
60	jacquet	40

Quelle jointure pour :

n°empl	nom_e	ville_e	age_e	n°chef_e	n°chef	nom_c	age_c
141	dupond	paris	40	500	500	albert	50
36	durand	tours	40	500	500	albert	50
251	parent	agen	25	60	60	jacquet	40

Exemple d'équi-jointure

Employe

n°empl	nom_e	ville_e	age_e	n°chef_e
141	dupond	paris	40	500
36	durand	tours	40	500
251	parent	agen	25	60

Chef

n°chef	nom_c	age_c
500	albert	50
60	jacquet	40

Quelle jointure pour : $Employe \bowtie_{nchef_e=nchef} Chef$

n°empl	nom_e	ville_e	age_e	n°chef_e	n°chef	nom_c	age_c
141	dupond	paris	40	500	500	albert	50
36	durand	tours	40	500	500	albert	50
251	parent	agen	25	60	60	jacquet	40

Exemple de θ-jointure

n°empl	nom_e	ville_e	age_e	n°chef_e)
141	dupond	paris	40	500
36	durand	tours	40	500
251	parent	agen	25	600
27	barbier	paris	53	500
125	lefevre	paris	30	523
208	legrand	evry	39	523

n°chef	nom_c	age_c
500	albert	50
60	jacquet	40
523	durieux	35

Quelle jointure pour ?

n°empl	nom_e	ville_e	age_e	n°chef_e	n°chef	nom_c	age_c
27	barbier	paris	53	500	500	albert	40
208	legrand	evry	39	523	523	durieux	35

Exemple de θ-jointure

n°empl	nom_e	ville_e	age_e	n°chef_e)
141	dupond	paris	40	500
36	durand	tours	40	500
251	parent	agen	25	600
27	barbier	paris	53	500
125	lefevre	paris	30	523
208	legrand	evry	39	523

n°chef	nom_c	age_c
500	albert	50
60	jacquet	40
523	durieux	35

Quelle jointure pour ? $Employe \bowtie_{nchef_e = nchef \wedge age_e > age_c} Chef$

n°empl	nom_e	ville_e	age_e	n°chef_e	n°chef	nom_c	age_c
27	barbier	paris	53	500	500	albert	40
208	legrand	evry	39	523	523	durieux	35

Exemple de jointure naturelle

EMPLOYE

NoEmp	Nom	Année	NoDep
1045	Dupond	1978	03
2067	Dupont	1965	06
0456	Martin	1981	03
0278	Martin	1987	05
0789	Blanc	1981	06

DEPARTEMENT

NoDep	Intitulé	Taille	NoResp
03	Compta	6	0456
06	Info	10	1249
05	Achats	3	0278

INFO-EMP = EMPLOYE $\overset{JN}{><}$ DEPARTEMENT

INFO-EMP

NoEmp	Nom	Année	NoDep	Intitulé	Taille	NoResp
1045	Dupond	1978	03	Compta	6	0456
2067	Dupont	1965	06	Info	10	1249
0456	Martin	1981	03	Compta	6	0456
0278	Martin	1987	05	Achats	3	0278
0789	Blanc	1981	06	Info	10	1249

Schéma de T :
on ne garde
qu'une seule fois
les attributs communs

Pause : Practise time !

Étudiants

NUMETUD	NOMETUD	PRENOMETUD	AGE	FORMATION
28	Codd	Edgar	20	3
32	Armstrong	William	20	4
53	Fagin	Ronald	19	3
107	Bunneman	Peter	18	3

Enseignants

NUMENS	NOMENS	PRENOMENS	GRADE
5050	Tarjan	Robert	PR
2123	Mannila	Heikki	MCF
3434	Papadimitriou	Spiros	PR
1470	Bagan	Guillaume	CR

Encadre

NUMENS	NUMETUD	DATE
5050	53	2005
3434	28	2020
5050	28	2015
2123	32	2019

- Prénom et nom de tou.te.s les étudiant.e.s
- Prénom et nom des enseignant.e.s qui ont le grade de PR
- Nom(s) des enseignant.e(s) qui encadrent l'étudiant 107
- Num des étudiant.e.s qui n'ont pas d'encadrant.e
- Prénom et nom de tou.te.s les étudiant.e.s et enseignant.e.s

Requêtes algébriques

- Prénom et nom de tou.te.s les étudiant.e.s :

$$\pi_{PRENOMETUD, NOMETUD}(Etudiants)$$

- Prénom et nom des enseignant.e.s qui sont PR :

$$\pi_{PRENOMEND, NOMENS}(\sigma_{GRADE='PR'}(Enseignants))$$

- Nom(s) des enseignant.e(s) qui encadrent l'étudiant 107 :

$$\pi_{NOMENS}(Enseignants \bowtie_{\sigma_{NUMETUD=107}} (Encadre))$$

- Num des étudiant.e.s qui n'ont pas d'encadrant.e :

$$\pi_{NUMETUD}(Etudiants) \setminus \pi_{NUMETUD}(Encadre)$$

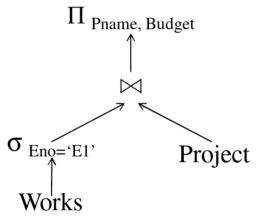
- Nom de tou.te.s les étudiant.e.s et enseignant.e.s :

$$\rho_{NOMETUD/NOM}(\pi_{NOMENS}(Etudiants)) \bigcup \rho_{NOMENS/NOM}(\pi_{NOMENS}(Enseignants))$$

Requête sous forme d'arbre !

Chaque requête peut être décrite sous forme d'arbre :

$$\Pi_{\text{Pname, Budget}}(\text{Project} \bowtie \sigma_{\text{Eno='E1'}}(\text{Works}))$$



Pour quoi faire ?

Calcul Relationnel à Variable Tuples

- Syntaxe :

  $$\{x^{(n)} | F(x)\}$$

  où $x^{(n)}$ est un n-uplet (c'est à dire un tuple à n champs) et F est une formule logique du premier ordre ; $F(x)$ exprime donc de façon déclarative les conditions que chaque tuple x doit vérifier pour appartenir au résultat.
- x est une variable libre de $F(x)$.
- On introduit si besoin des variables liées par des quantificateurs $\exists$ ou $\forall$. Ces variables permettent par exemple de parcourir les relations, pour être comparées à x.

Exemples Calcul Relationnel

- Quel est le prenom et le nom de tous les étudiants
  - $\{x = (x_1, x_2) \mid \exists x' \in Etudiants((x_1, x_2) = x'[PRENOMETUD, NOMETUD])\}$
- Quel est le prenom et le nom des enseignants qui sont PR
  - $\{x = (x_1, x_2) \mid \exists x' \in Enseignants(x'[GRADE] =' PR' \wedge (x_1, x_2) = x'[PRENOMENS, NOMENS])\}$
- Quel est le nom des enseignants qui encadrent l'étudiant 53 ?
  - $\{x = (x_1) \mid \exists x' \in Encadre(x'[NUMETUD] = 107 \wedge \exists y' \in Enseignants(x'[NUMENS] = y'[NUMENS] \wedge y'[NOMENS] = x_1))\}$
- Quels est le num des étudiants qui n'ont pas d'encadrant.
  - $\{x = (x_1) \mid \exists x' \in Etudiants(x'[NUMETUD] = x_1 \wedge \forall y' \in Encadre(x'[NUMETUD] \neq y'[NUMETUD]))\}$
- Lister le nom de tous les étudiants et enseignants.
  - $\{x = (x_1) \mid \exists x' \in Etudiants(x'[NOMETUD] = x_1)\} \cup \{x = (x_1) \mid \exists x' \in Enseignants(x'[NOMENS] = x_1)\}$

Next

Un langage pour ces requêtes ?

Bilan

1. Algèbre Relationnelle

Bases de données (CS443)

#5, Modèle relationnel: SQL

Laure Gonnord

Grenoble INP/Esisar

2022-2023

Rappel du jeu de données

ÉLÈVE (idE, *nomE*, *moyenneLycee*, *effectifLycee*)
CANDIDATURE (*#idE*, *#nomU*, département, *décision*)
UNIVERSITÉ (nomU, *ville*, *effectif*)

idE	nomE	moyenneLycée	effectifLycée
123	Ana	19.5	1000
234	Bob	18	1500
345	Chloe	17.5	500
456	Damien	19.5	1000
543	Chloé	17	2000
567	Éléonore	14.5	2000
654	Ana	19.5	1000
678	Farid	19	200
765	Joana	14.5	1500
789	Gisèle	17	800
876	Irène	19.5	400
898	Hector	18.5	800

Table ÉLÈVE

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Étienne	21000

Table UNIVERSITÉ

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table CANDIDATURE

Dans ces transparents, les tables sont abrégées en *E*, *C* et *U*

Jeu de données francisé inspiré du cours Databases de Stanford

Crédits

Transparents F. Duchateau, pour univ Lyon1, CC by SA.

`https://perso.liris.cnrs.fr/fabien.duchateau/BDW1/`

Dialectes

Suivant le SGBD des différences de syntaxe mineures peuvent apparaître.

On pourra se reporter à la page ci-dessus pour l'écriture de sous-requêtes.

Plan

Éléments de langage

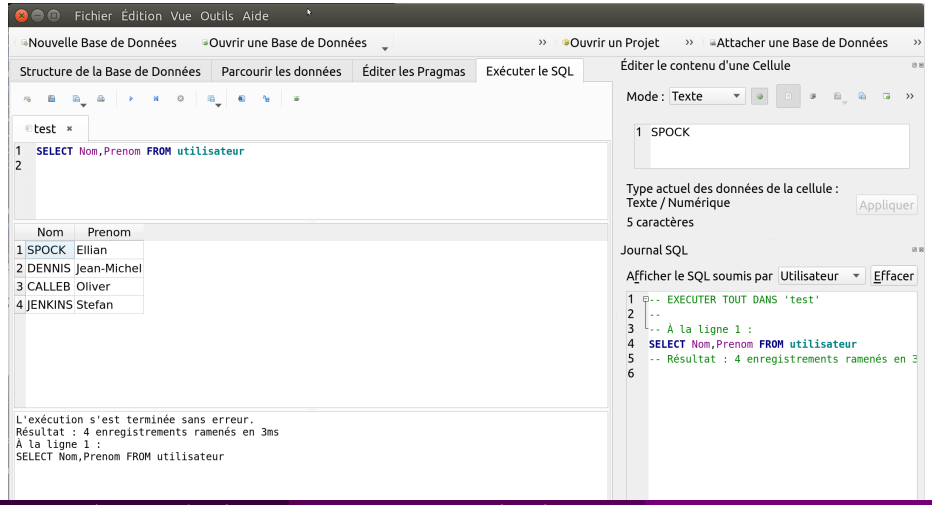
Disclaimer

Ce n'est pas un parcours exhaustif du langage, mais uniquement quelques concepts. On regardera en particulier :

- la correspondance avec les/des opérations de l'algèbre relationnelle
- les fonctions support (comptage, traitement de chaînes) supplémentaires.

Une documentation sera fournie avec les TPs.

Démo - sqlitebrowser



Différences entre la théorie et SQL

- Possibilité de doublons
- Possibilité d'ordonner le résultat des requêtes
- Notion de valeur non définie
- Absence de certains opérateurs ensemblistes

Construire les tables

Les commandes (requêtes de modification) pour construire les tables seront vues en TP. Voici les mots clés :

- CREATE TABLE
- CONSTRAINT ... PRIMARY KEY, NOT NULL, ...
- INSERT INTO, DELETE, ...

Un exemple pour la route :

```
CREATE TABLE auteur (
num_a INTEGER,
nom VARCHAR(30),
CONSTRAINT cle_auteur PRIMARY KEY (num_a)
) ;
```

Plan

Plan

Syntaxe

Syntaxe minimale d'une requête SQL :

SELECT att_1, att_2, ...
FROM *nom_table* ;

- SELECT et FROM sont des clauses SQL
- Projection : récupération des valeurs contenues dans la table `nom_table`, en ne gardant que les attributs `att1, att2, …`
- On peut remplacer att_1, att_2, ... par * pour utiliser tous les attributs des tables listées dans la clause FROM

Équivalences

- En SQL :
  SELECT att_1, att_2, ...
  FROM *nom_table* ;
- En algèbre relationnelle :
  $\pi_{att_1, att_2, \ldots}(nom\_table)$
- En calcul relationnel tuple :
  $\{t.att_1, t.att_2, \cdots \mid nom\_table(t)\}$

Exemple de projection

idE	nomE	moyenneLycée	effectifLycée
123	Ana	19.5	1000
234	Bob	18	1500
345	Chloé	17.5	500
456	Damien	19.5	1000
543	Chloé	17	2000
567	Éléonore	14.5	2000
654	Ana	19.5	1000
678	Farid	19	200
765	Joana	14.5	1500
789	Gisèle	17	800
876	Irène	19.5	400
898	Hector	18.5	800

Table ÉLÈVE

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Étienne	21000

Table UNIVERSITÉ

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table CANDIDATURE

Le nom et l'effectif des universités

Exemple de projection

idE	nomE	moyenneLycée	effectifLycée
123	Ana	19.5	1000
234	Bob	18	1500
345	Chloé	17.5	500
456	Damien	19.5	1000
543	Chloé	17	2000
567	Éléonore	14.5	2000
654	Ana	19.5	1000
678	Farid	19	200
765	Joana	14.5	1500
789	Gisèle	17	800
876	Irène	19.5	400
898	Hector	18.5	800

Table ÉLÈVE

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Étienne	21000

Table UNIVERSITÉ

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table CANDIDATURE

Le nom et l'effectif des universités

```
22 SELECT nomU, effectif
23 FROM Université;
```


Exemple de projection

idE	nomE	moyenneLycée	effectifLycée
123	Ana	19.5	1000
234	Bob	18	1500
345	Chloé	17.5	500
456	Damien	19.5	1000
543	Chloé	17	2000
567	Éléonore	14.5	2000
654	Ana	19.5	1000
678	Farid	19	200
765	Joana	14.5	1500
789	Gisèle	17	800
876	Irène	19.5	400
898	Hector	18.5	800

Table ÉLÈVE

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Étienne	21000

Table UNIVERSITÉ

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table CANDIDATURE

Le nom et l'effectif des universités

```
22 SELECT nomU, effectif
23 FROM Université;
```

nomU	effectif
INSA	36000
UCB	15000
UJF	10000
UJM	21000

Exemple de projection (2)

idE	nomE	moyenneLycée	effectifLycée
123	Ana	19.5	1000
234	Bob	18	1500
345	Chloé	17.5	500
456	Damien	19.5	1000
543	Chloé	17	2000
567	Éléonore	14.5	2000
654	Ana	19.5	1000
678	Farid	19	200
765	Joana	14.5	1500
789	Gisèle	17	800
876	Irène	19.5	400
898	Hector	18.5	800

Table ÉLÈVE

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Étienne	21000

Table UNIVERSITÉ

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table CANDIDATURE

Les informations sur les universités

Exemple de projection (2)

idE	nomE	moyenneLycée	effectifLycée
123	Ana	19.5	1000
234	Bob	18	1500
345	Chloé	17.5	500
456	Damien	19.5	1000
543	Chloé	17	2000
567	Éléonore	14.5	2000
654	Ana	19.5	1000
678	Farid	19	200
765	Joana	14.5	1500
789	Gisèle	17	800
876	Irène	19.5	400
898	Hector	18.5	800

Table Élève

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Étienne	21000

Table Université

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table Candidature

Les informations sur les universités

```
25 SELECT nomU, ville, effectif
26 FROM Université;
```

ou

```
28 SELECT * FROM Université;
```


Exemple de projection (2)

idE	nomE	moyenneLycée	effectifLycée
123	Ana	19.5	1000
234	Bob	18	1500
345	Chloé	17.5	500
456	Damien	19.5	1000
543	Chloé	17	2000
567	Éléonore	14.5	2000
654	Ana	19.5	1000
678	Farid	19	200
765	Joana	14.5	1500
789	Gisèle	17	800
876	Irène	19.5	400
898	Hector	18.5	800

Table Élève

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Étienne	21000

Table Université

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table Candidature

Les informations sur les universités

```
25 SELECT nomU, ville, effectif
26 FROM Université;
```

ou

```
28 SELECT * FROM Université;
```


nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Étienne	21000

Suppression des doublons

Mot clé DISTINCT pour supprimer les n-uplets en doublon :

SELECT DISTINCT att_1, att_2, …
FROM *nom_table* ;

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Etienne	21000
UJF	Grenoble	10000

En rouge, un exemple de n-uplet en doublon dans la table (les trois valeurs des 2 n-uplets étant identiques)

Suppression des doublons (2)

idE	nomE	moyenneLycée	effectifLycée
123	Ana	19.5	1000
234	Bob	18	1500
345	Chloé	17.5	500
456	Damien	19.5	1000
543	Chloé	17	2000
567	Éléonore	14.5	2000
654	Ana	19.5	1000
678	Farid	19	200
765	Joana	14.5	1500
789	Gisèle	17	800
876	Irène	19.5	400
898	Hector	18.5	800

Table Élève

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Étienne	21000

Table Université

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table Candidature

Les villes où se trouvent des universités

Suppression des doublons (2)

idE	nomE	moyenneLycée	effectifLycée
123	Ana	19.5	1000
234	Bob	18	1500
345	Chloé	17.5	500
456	Damien	19.5	1000
543	Chloé	17	2000
567	Éléonore	14.5	2000
654	Ana	19.5	1000
678	Farid	19	200
765	Joana	14.5	1500
789	Gisèle	17	800
876	Irène	19.5	400
898	Hector	18.5	800

Table Élève

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Étienne	21000

Table Université

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table Candidature

Les villes où se trouvent des universités

```
30 SELECT ville FROM Université;
```

ville
Lyon
Lyon
Grenoble
Saint-Étienne

```
32 SELECT DISTINCT ville FROM
       Université;
```

ville
Lyon
Grenoble
Saint-Étienne

Renommage et alias

Mot-clé AS pour renommer un attribut :

SELECT att_1 **AS** att'_1, att_2 **AS** att'_2, …
FROM *nom_table* ;

- L'attribut att_1 sera renommé en att'_1, etc.

Alias de table comme moyen de désambiguïsation ou pour utiliser deux fois la même table :

SELECT DISTINCT $t.att_1$, att_2, …
FROM *nom_table* [**AS**] t ;

- L'alias de la table est t, accès à l'attribut att_1 par $t.att_1$

Renommage et alias (2)

idE	nomE	moyenneLycée	effectifLycée
123	Ana	19.5	1000
234	Bob	18	1500
345	Chloé	17.5	500
456	Damien	19.5	1000
543	Chloé	17	2000
567	Éléonore	14.5	2000
654	Ana	19.5	1000
678	Farid	19	200
765	Joana	14.5	1500
789	Gisèle	17	800
876	Irène	19.5	400
898	Hector	18.5	800

Table Élève

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Étienne	21000

Table Université

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table Candidature

Le nom et la ville des universités, avec renommage

```
34 SELECT nomU AS nom-univ, ville AS ville-univ FROM
       Université;
```


Renommage et alias (2)

idE	nomE	moyenneLycée	effectifLycée
123	Ana	19.5	1000
234	Bob	18	1500
345	Chloé	17.5	500
456	Damien	19.5	1000
543	Chloé	17	2000
567	Éléonore	14.5	2000
654	Ana	19.5	1000
678	Farid	19	200
765	Joana	14.5	1500
789	Gisèle	17	800
876	Irène	19.5	400
898	Hector	18.5	800

Table Élève

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Étienne	21000

Table Université

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table Candidature

Le nom et la ville des universités, avec renommage

```
34 SELECT nomU AS nom-univ, ville AS ville-univ FROM
       Université;
```

nom-univ	ville-univ
INSA	Lyon
UCB	Lyon
UJF	Grenoble
UJM	Saint-Étienne

Renommage et alias (3)

idE	nomE	moyenneLycée	effectifLycée
123	Ana	19.5	1000
234	Bob	18	1500
345	Chloé	17.5	500
456	Damien	19.5	1000
543	Chloé	17	2000
567	Éléonore	14.5	2000
654	Ana	19.5	1000
678	Farid	19	200
765	Joana	14.5	1500
789	Gisèle	17	800
876	Irène	19.5	400
898	Hector	18.5	800

Table ÉLÈVE

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Étienne	21000

Table UNIVERSITÉ

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table CANDIDATURE

Le nom des universités avec alias de table

```
36 SELECT u.nomU, u.ville FROM Université AS u;
```


Renommage et alias (3)

idE	nomE	moyenneLycée	effectifLycée
123	Ana	19.5	1000
234	Bob	18	1500
345	Chloé	17.5	500
456	Damien	19.5	1000
543	Chloé	17	2000
567	Éléonore	14.5	2000
654	Ana	19.5	1000
678	Farid	19	200
765	Joana	14.5	1500
789	Gisèle	17	800
876	Irène	19.5	400
898	Hector	18.5	800

Table ÉLÈVE

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Étienne	21000

Table UNIVERSITÉ

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table CANDIDATURE

Le nom des universités avec alias de table

```
36 SELECT u.nomU, u.ville FROM Université AS u;
```

nomU	ville
INSA	Lyon
UCB	Lyon
UJF	Grenoble
UJM	Saint-Étienne

En résumé

- Clause SELECT ≡ projection (d'attributs)
- Clause FROM ≡ liste des tables utilisées
- Renommage d'attribut (AS) et alias de table
- Syntaxe d'une requête SQL (crochets = option) :

SELECT att_1 [, att_2 [**AS** att'_2], ...]
FROM nom_table_1 [, nom_table_2 [alias] ...] ;

Plan

Syntaxe

Clause WHERE pour la sélection :

SELECT att_1, att_2, …
FROM *nom_table*
WHERE *condition* ;

- La clause WHERE permet de sélectionner les lignes en filtrant celles qui ne remplissent pas la condition (i.e., éliminées du résultat)

Conditions

Condition du WHERE : une combinaison d'expressions connectées par **AND** ($\wedge$) ou **OR** ($\vee$)

Une expression effectue une opération entre un attribut et une constante ou entre deux attributs

Expressions simples :

- Opérateurs de comparaison (`=`, `!=`, `<`, `<=`, `>`, `>=`)
- Différents types de données utilisés pour les constantes :
  - nombres : *1, 36000, 1.5*
  - chaînes de caractères : *'UCBL', 'Grenoble'*
  - dates : *'1986-04-26'* (le formatage des dates peut varier selon le SGBD)

Exemple de sélection

idE	nomE	moyenneLycée	effectifLycée
123	Ana	19.5	1000
234	Bob	18	1500
345	Chloé	17.5	500
456	Damien	19.5	1000
543	Chloé	17	2000
567	Éléonore	14.5	2000
654	Ana	19.5	1000
678	Farid	19	200
765	Joana	14.5	1500
789	Gisèle	17	800
876	Irène	19.5	400
898	Hector	18.5	800

Table ÉLÈVE

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Étienne	21000

Table UNIVERSITÉ

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table CANDIDATURE

Nom et effectif des universités avec un effectif inférieur à 20000

Exemple de sélection

idE	nomE	moyenneLycée	effectifLycée
123	Ana	19.5	1000
234	Bob	18	1500
345	Chloé	17.5	500
456	Damien	19.5	1000
543	Chloé	17	2000
567	Éléonore	14.5	2000
654	Ana	19.5	1000
678	Farid	19	200
765	Joana	14.5	1500
789	Gisèle	17	800
876	Irène	19.5	400
898	Hector	18.5	800

Table ÉLÈVE

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Étienne	21000

Table UNIVERSITÉ

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table CANDIDATURE

Nom et effectif des universités avec un effectif inférieur à 20000

```
40 SELECT nomU, effectif
41 FROM Université
42 WHERE effectif < 20000;
```

Exemple de sélection

idE	nomE	moyenneLycée	effectifLycée
123	Ana	19.5	1000
234	Bob	18	1500
345	Chloé	17.5	500
456	Damien	19.5	1000
543	Chloé	17	2000
567	Éléonore	14.5	2000
654	Ana	19.5	1000
678	Farid	19	200
765	Joana	14.5	1500
789	Gisèle	17	800
876	Irène	19.5	400
898	Hector	18.5	800

Table Élève

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Étienne	21000

Table Université

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table Candidature

Nom et effectif des universités avec un effectif inférieur à 20000

```
40 SELECT nomU, effectif
41 FROM Université
42 WHERE effectif < 20000;
```

nomU	effectif
UCB	15000
UJF	10000

Exemple de sélection (2)

idE	nomE	moyenneLycée	effectifLycée
123	Ana	19.5	1000
234	Bob	18	1500
345	Chloé	17.5	500
456	Damien	19.5	1000
543	Chloé	17	2000
567	Éléonore	14.5	2000
654	Ana	19.5	1000
678	Farid	19	200
765	Joana	14.5	1500
789	Gisèle	17	800
876	Irène	19.5	400
898	Hector	18.5	800

Table Élève

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Étienne	21000

Table Université

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table Candidature

Informations sur les universités de Lyon avec un effectif inférieur à 25000

Exemple de sélection (2)

idE	nomE	moyenneLycée	effectifLycée
123	Ana	19.5	1000
234	Bob	18	1500
345	Chloé	17.5	500
456	Damien	19.5	1000
543	Chloé	17	2000
567	Éléonore	14.5	2000
654	Ana	19.5	1000
678	Farid	19	200
765	Joana	14.5	1500
789	Gisèle	17	800
876	Irène	19.5	400
898	Hector	18.5	800

Table Élève

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Étienne	21000

Table Université

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table Candidature

Informations sur les universités de Lyon avec un effectif inférieur à 25000

```
44 SELECT *
45 FROM Université
46 WHERE ville = 'Lyon'
47 AND effectif < 20000;
```


Exemple de sélection (2)

idE	nomE	moyenneLycée	effectifLycée
123	Ana	19.5	1000
234	Bob	18	1500
345	Chloé	17.5	500
456	Damien	19.5	1000
543	Chloé	17	2000
567	Éléonore	14.5	2000
654	Ana	19.5	1000
678	Farid	19	200
765	Joana	14.5	1500
789	Gisèle	17	800
876	Irène	19.5	400
898	Hector	18.5	800

Table Élève

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Étienne	21000

Table Université

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table Candidature

Informations sur les universités de Lyon avec un effectif inférieur à 25000

```
44 SELECT *
45 FROM Université
46 WHERE ville = 'Lyon'
47 AND effectif < 20000;
```

nomU	ville	effectif
UCB	Lyon	15000

Autres opérateurs

- Opérateur **IN** (val_1, val_2, …) :
  - spécifie un ensemble de valeur possibles

- Opérateur **BETWEEN** val_1 **AND** val_2 :
  - spécifie un intervalle de valeurs (bornes val_1 et val_2 incluses)
  - attention à ne pas confondre le AND du **BETWEEN** avec celui qui correspond au $\wedge$

Exemple de sélection (3)

idE	nomE	moyenneLycée	effectifLycée
123	Ana	19.5	1000
234	Bob	18	1500
345	Chloé	17.5	500
456	Damien	19.5	1000
543	Chloé	17	2000
567	Éléonore	14.5	2000
654	Ana	19.5	1000
678	Farid	19	200
765	Joana	14.5	1500
789	Gisèle	17	800
876	Irène	19.5	400
898	Hector	18.5	800

Table Élève

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Étienne	21000

Table Université

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table Candidature

Les informations sur les universités de Lyon ou Grenoble

```
49 SELECT *
50 FROM Université
51 WHERE ville IN ('Lyon', 'Grenoble');

53 SELECT *
54 FROM Université
55 WHERE ville = 'Lyon' OR ville = 'Grenoble';
```


Exemple de sélection (3)

idE	nomE	moyenneLycée	effectifLycée
123	Ana	19.5	1000
234	Bob	18	1500
345	Chloé	17.5	500
456	Damien	19.5	1000
543	Chloé	17	2000
567	Éléonore	14.5	2000
654	Ana	19.5	1000
678	Farid	19	200
765	Joana	14.5	1500
789	Gisèle	17	800
876	Irène	19.5	400
898	Hector	18.5	800

Table Élève

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Étienne	21000

Table Université

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table Candidature

Les informations sur les universités de Lyon ou Grenoble

```
49 SELECT *
50 FROM Université
51 WHERE ville IN ('Lyon', 'Grenoble');

53 SELECT *
54 FROM Université
55 WHERE ville = 'Lyon' OR ville = 'Grenoble';
```

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000

Exemple de sélection (4)

idE	nomE	moyenneLycée	effectifLycée
123	Ana	19.5	1000
234	Bob	18	1500
345	Chloé	17.5	500
456	Damien	19.5	1000
543	Chloé	17	2000
567	Éléonore	14.5	2000
654	Ana	19.5	1000
678	Farid	19	200
765	Joana	14.5	1500
789	Gisèle	17	800
876	Irène	19.5	400
898	Hector	18.5	800

Table Élève

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Étienne	21000

Table Université

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table Candidature

Les universités avec un effectif entre 12000 et 25000

```
57 SELECT * FROM Université
58 WHERE effectif BETWEEN 12000 AND
       25000;

60 SELECT * FROM Université
61 WHERE effectif >= 12000 AND
       effectif <= 25000;
```

Exemple de sélection (4)

idE	nomE	moyenneLycée	effectifLycée
123	Ana	19.5	1000
234	Bob	18	1500
345	Chloé	17.5	500
456	Damien	19.5	1000
543	Chloé	17	2000
567	Éléonore	14.5	2000
654	Ana	19.5	1000
678	Farid	19	200
765	Joana	14.5	1500
789	Gisèle	17	800
876	Irène	19.5	400
898	Hector	18.5	800

Table ÉLÈVE

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Étienne	21000

Table UNIVERSITÉ

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table CANDIDATURE

Les universités avec un effectif entre 12000 et 25000

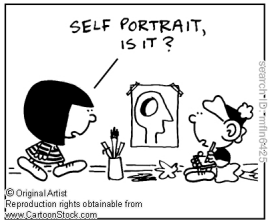
```
57 SELECT * FROM Université
58 WHERE effectif BETWEEN 12000 AND
       25000;

60 SELECT * FROM Université
61 WHERE effectif >= 12000 AND
       effectif <= 25000;
```

nomU	ville	effectif
UCB	Lyon	15000
UJM	Saint-Étienne	21000

Valeurs non définies

En pratique, il est possible d'avoir des valeurs non définies :

- Elles sont représentées par le mot clé NULL
- On peut tester si une valeur n'est pas définie grâce à la condition **IS NULL** (ou son contraire **IS NOT NULL**)



Exemple de sélection (5)

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	**NULL**
UJM	Saint-Etienne	21000

Les informations sur les universités avec un effectif non défini

```
65 SELECT * FROM Université
66 WHERE effectif IS NULL;
```


Exemple de sélection (5)

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	**NULL**
UJM	Saint-Etienne	21000

Les informations sur les universités avec un effectif non défini

```
65 SELECT * FROM Université
66 WHERE effectif IS NULL;
```

nomU	ville	effectif
UJF	Grenoble	NULL

En résumé

- Clause WHERE ≡ sélection (d'instances)
- Condition = combinaison (AND, OR) d'expressions (=, !=, <, <=, >, >=)
- Opérateurs spécifiques (IN, BETWEEN), valeur non définie (NULL)
- Syntaxe d'une requête SQL (crochets = option) :

SELECT att_1 [, att_2 [**AS** att'_2], ...]
FROM nom_table_1 [, nom_table_2 [alias], ...]
[**WHERE** *condition*] ;

Plan

Traitement des chaînes 1/2

SELECT ... **FROM** ... **WHERE** attribut **LIKE** pattern

le pattern est une forme de *regexp*.
Ex : `select distinct titre from livre where titre like 'H%' from utilisateur` :

titre
Harry Potter à l'école des sorciers
Harry Potter et la chambre des secrets

Traitement des chaînes 2/2

- On peut comparer des chaînes, ...
- On peut concaténer des chaînes, utiliser `upper`

```
select upper(nom || ' ' || prenom) as nom_prenom from utilisateur;
```

nom_prenom
CALLEB OLIVER
JENKINS STEFAN
SPOCK ELLIAN
DENNIS JEAN-MICHEL

noter l'utilisation de as pour nommer le résultat et encore : BETWEEN (nombres, chaînes).

Plan

Syntaxe du tri

Clause ORDER BY pour trier le résultat d'une requête :

SELECT att_1, att_2, ...
FROM *nom_table*
WHERE *condition*
ORDER BY att_i, att_j, ... ;

- Le résultat de la requête est trié selon l'ordre naturel croissant de l'attribut att_i
- En cas d'égalité entre deux lignes au niveau de l'attribut att_i, on utilise l'attribut suivant att_j, etc.
- Le nom d'un attribut peut être suivi par ASC ou DESC pour indiquer un ordre croissant (défaut) ou décroissant

Exemple de tri

idE	nomE	moyenneLycée	effectifLycée
123	Ana	19.5	1000
234	Bob	18	1500
345	Chloé	17.5	500
456	Damien	19.5	1000
543	Chloé	17	2000
567	Éléonore	14.5	2000
654	Ana	19.5	1000
678	Farid	19	200
765	Joana	14.5	1500
789	Gisèle	17	800
876	Irène	19.5	400
898	Hector	18.5	800

Table Élève

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Étienne	21000

Table Université

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table Candidature

Les informations sur les universités, triées par effectif croissant

```
71 SELECT * FROM Université
72 ORDER BY effectif ASC;
```


Exemple de tri

idE	nomE	moyenneLycée	effectifLycée
123	Ana	19.5	1000
234	Bob	18	1500
345	Chloé	17.5	500
456	Damien	19.5	1000
543	Chloé	17	2000
567	Éléonore	14.5	2000
654	Ana	19.5	1000
678	Farid	19	200
765	Joana	14.5	1500
789	Gisèle	17	800
876	Irène	19.5	400
898	Hector	18.5	800

Table Élève

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Étienne	21000

Table Université

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table Candidature

Les informations sur les universités, triées par effectif croissant

```
71 SELECT * FROM Université
72 ORDER BY effectif ASC;
```

nomU	ville	effectif
UJF	Grenoble	10000
UCB	Lyon	15000
UJM	Saint-Étienne	21000
INSA	Lyon	36000

Exemple de tri (2)

idE	nomE	moyenneLycée	effectifLycée
123	Ana	19.5	1000
234	Bob	18	1500
345	Chloé	17.5	500
456	Damien	19.5	1000
543	Chloé	17	2000
567	Éléonore	14.5	2000
654	Ana	19.5	1000
678	Farid	19	200
765	Joana	14.5	1500
789	Gisèle	17	800
876	Irène	19.5	400
898	Hector	18.5	800

Table Élève

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Étienne	21000

Table Université

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table Candidature

Les informations sur les universités de plus de 12000 étudiant-e-s, avec un tri par ville décroissante puis par effectif croissant

```
77 SELECT * FROM Université
78 WHERE effectif > 12000
79 ORDER BY ville DESC,
       effectif ASC;
```


Exemple de tri (2)

idE	nomE	moyenneLycée	effectifLycée
123	Ana	19.5	1000
234	Bob	18	1500
345	Chloé	17.5	500
456	Damien	19.5	1000
543	Chloé	17	2000
567	Éléonore	14.5	2000
654	Ana	19.5	1000
678	Farid	19	200
765	Joana	14.5	1500
789	Gisèle	17	800
876	Irène	19.5	400
898	Hector	18.5	800

Table Élève

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Étienne	21000

Table Université

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table Candidature

Les informations sur les universités de plus de 12000 étudiant-e-s, avec un tri par ville décroissante puis par effectif croissant

```
77 SELECT * FROM Université
78 WHERE effectif > 12000
79 ORDER BY ville DESC,
       effectif ASC;
```

nomU	ville	effectif
UJM	Saint-Étienne	21000
UCB	Lyon	15000
INSA	Lyon	36000

Syntaxe de la limitation

Clause LIMIT pour conserver un nombre restreint de résultats :

SELECT $att_1, att_2, \ldots$
FROM *nom_table*
WHERE *condition*
ORDER BY $att_i, att_j, \ldots$
LIMIT n ;

- Les n premières instances (après le tri) sont conservés dans le résultat
- Pas un top-K, qui récupère toutes les instances avec les n meilleures valeurs

Dans la norme SQL, la syntaxe est FETCH FIRST n ROWS ONLY

Exemple de limitation

idE	nomE	moyenneLycée	effectifLycée
123	Ana	19.5	1000
234	Bob	18	1500
345	Chloé	17.5	500
456	Damien	19.5	1000
543	Chloé	17	2000
567	Éléonore	14.5	2000
654	Ana	19.5	1000
678	Farid	19	200
765	Joana	14.5	1500
789	Gisèle	17	800
876	Irène	19.5	400
898	Hector	18.5	800

Table Élève

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Étienne	21000

Table Université

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table Candidature

Deux universités qui ont les effectifs les plus élevés

nomU	effectif
INSA	36000
UJM	21000

Exemple de limitation

idE	nomE	moyenneLycée	effectifLycée
123	Ana	19.5	1000
234	Bob	18	1500
345	Chloé	17.5	500
456	Damien	19.5	1000
543	Chloé	17	2000
567	Éléonore	14.5	2000
654	Ana	19.5	1000
678	Farid	19	200
765	Joana	14.5	1500
789	Gisèle	17	800
876	Irène	19.5	400
898	Hector	18.5	800

Table ÉLÈVE

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Étienne	21000

Table UNIVERSITÉ

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table CANDIDATURE

Deux universités qui ont les effectifs les plus élevés

```
81 SELECT nomU, effectif FROM
       Université
82 ORDER BY effectif DESC LIMIT 2;
```

nomU	effectif
INSA	36000
UJM	21000

En résumé

- Clause ORDER BY ≡ tri du résultat
- Clause LIMIT ≡ troncature du résultat aux premières instances
- Syntaxe d'une requête SQL (crochets = option) :

SELECT att_1 [, att_2 [**AS** att'_2], ...]
FROM nom_table_1 [, nom_table_2 [**AS** alias]]
[**WHERE** *condition*]
[**ORDER BY** att_i [, att_j, ...]]
[**LIMIT** n] ;

Plan

Fonction COUNT

COUNT(att) : le nombre de valeurs de l'attribut *att*

COUNT(**DISTINCT** att) : le nombre de valeurs distinctes de l'attribut *att*

- Les valeurs NULL ne sont pas comptées
- * peut remplacer *att*, cela compte alors le nombre de n-uplets

Exemple de COUNT sans regroupement

idE	nomE	moyenneLycée	effectifLycée
123	Ana	19.5	1000
234	Bob	18	1500
345	Chloé	17.5	500
456	Damien	19.5	1000
543	Chloé	17	2000
567	Éléonore	14.5	2000
654	Ana	19.5	1000
678	Farid	19	200
765	Joana	14.5	1500
789	Gisèle	17	800
876	Irène	19.5	400
898	Hector	18.5	800

Table ÉLÈVE

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Étienne	21000

Table UNIVERSITÉ

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table CANDIDATURE

Le nombre d'élèves

```
28 SELECT COUNT(*)
29 FROM Élève;
```

ou

count
12

```
36 SELECT COUNT(nomE)
37 FROM Élève;
```


Exemple de COUNT avec DISTINCT

idE	nomE	moyenneLycée	effectifLycée
123	Ana	19.5	1000
234	Bob	18	1500
345	Chloé	17.5	500
456	Damien	19.5	1000
543	Chloé	17	2000
567	Éléonore	14.5	2000
654	Ana	19.5	1000
678	Farid	19	200
765	Joana	14.5	1500
789	Gisèle	17	800
876	Irène	19.5	400
898	Hector	18.5	800

Table ÉLÈVE

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Étienne	21000

Table UNIVERSITÉ

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table CANDIDATURE

Le nombre de noms d'élèves distincts

```
40 SELECT COUNT(DISTINCT nomE)
41 FROM Élève;
```

count
10

Déroulement pas à pas d'une requête avec regroupement

Le nombre de candidatures par université, avec tri par nom :

SELECT nomU, **COUNT**(*) **FROM** *C*
GROUP BY nomU **ORDER BY** nomU ;

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table CANDIDATURE

Déroulement pas à pas d'une requête avec regroupement

Le nombre de candidatures par université, avec tri par nom :

SELECT nomU, **COUNT**(*) **FROM** *C*
GROUP BY nomU **ORDER BY** nomU ;

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Déroulement pas à pas d'une requête avec regroupement

Le nombre de candidatures par université, avec tri par nom :

SELECT nomU, **COUNT**(*) **FROM** *C*
GROUP BY nomU **ORDER BY** nomU ;

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

⇒

nomU	count
UJM	6
UCB	6
UJF	4
INSA	3

Déroulement pas à pas d'une requête avec regroupement

Le nombre de candidatures par université, avec tri par nom :

SELECT nomU, **COUNT**(*) **FROM** *C*
GROUP BY nomU **ORDER BY** nomU ;

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

⇒

nomU	count
UJM	6
UCB	6
UJF	4
INSA	3

⇒

nomU	count
INSA	3
UCB	6
UJF	4
UJM	6

Having

Idem que where, mais sur les regroupements (voir TP).

Plan

Syntaxe

SELECT att_1, att_2, ...
FROM nom_table_1, nom_table_2, ... ;

- Plusieurs tables séparées par des virgules = produit cartésien entre ces différentes tables
- Si la requête utilise un attribut *att* présent dans plusieurs tables, on doit l'écrire *nom_table.att* ou utiliser un alias de table *alias.att*

Exemple de produit cartésien

idE	nomE	moyenneLycée	effectifLycée
123	Ana	19.5	1000
234	Bob	18	1500
345	Chloé	17.5	500
456	Damien	19.5	1000
543	Chloé	17	2000
567	Éléonore	14.5	2000
654	Ana	19.5	1000
678	Farid	19	200
765	Joana	14.5	1500
789	Gisèle	17	800
876	Irène	19.5	400
898	Hector	18.5	800

Table Élève

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Étienne	21000

Table Université

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table Candidature

Les paires de nom d'université et de nom d'élève

```
19 SELECT nomU,nomE
20 FROM Université, Élève;
```

nomU	nomE
INSA	Ana
UCB	Ana
UJF	Ana
UJM	Ana
INSA	Bob
...	...

Total de 48 tuples (12 × 4)

Plan

Syntaxe

SELECT att_1, att_2, ...
FROM nom_table_1 **NATURAL JOIN** nom_table_2
[**WHERE** *autres_conditions*] ;

- Soient att_{c_1}, ..., att_{c_k} les attributs communs des tables nom_table_1 et nom_table_2
- Les instances de nom_table_1 et nom_table_2 qui possèdent des valeurs égales sur tous leurs attributs communs att_{c_1}, ..., att_{c_k} sont "assemblées" en un tuple qui est ajouté dans le résultat

Syntaxe de la jointure naturelle tronquée

SELECT att_1, att_2, ...
FROM nom_table_1 **NATURAL JOIN** nom_table_2
USING $(att_{c_x}, \ldots, att_{c_y})$
[**WHERE** *autres_conditions*] ;

- Soient att_{c_1}, ..., att_{c_k} les attributs communs des tables nom_table_1 et nom_table_2, et $\{att_{c_x}, \ldots, att_{c_y}\} \in \{att_{c_1}, \ldots, att_{c_k}\}$
- Le mot-clé USING permet de faire une jointure naturelle sur un sous-ensemble des attributs communs $\{att_{c_x}, \ldots, att_{c_y}\}$ de nom_table_1 et nom_table_2

Exemple de jointure naturelle

idE	nomE	moyenneLycée	effectifLycée
123	Ana	19.5	1000
234	Bob	18	1500
345	Chloé	17.5	500
456	Damien	19.5	1000
543	Chloé	17	2000
567	Éléonore	14.5	2000
654	Ana	19.5	1000
678	Farid	19	200
765	Joana	14.5	1500
789	Gisèle	17	800
876	Irène	19.5	400
898	Hector	18.5	800

Table ÉLÈVE

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Étienne	21000

Table UNIVERSITÉ

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table CANDIDATURE

Les élèves avec plus de 19 de moyenne qui ont candidaté

```
30 SELECT DISTINCT e.idE, nomE,
       moyenneLycée, nomU
31 FROM Élève e NATURAL JOIN
       Candidature c
32 WHERE moyenneLycée > 19;
```

idE	nomE	moyenneLycee	nomU
123	Ana	19.5	INSA
123	Ana	19.5	UCB
123	Ana	19.5	UJM
876	Irene	19.5	UCB
876	Irene	19.5	UJF

Plan

Syntaxe

La **jointure interne** est fréquemment utilisée : seuls les tuples qui respectent la condition de jointure sont conservés

SELECT att_1, att_2, ...
FROM nom_table_1 **INNER JOIN** nom_table_2
ON $nom\_table_1.att_x$ Θ $nom\_table_2.att_x$
[**WHERE** *autres_conditions*] ;

- Soit Θ un opérateur parmi $=$, $\neq$, $<$, $>$, $\leq$, $\geq$, LIKE, ...
- La condition de jointure $nom\_table_1.att_x$ Θ $nom\_table_2.att_x$ s'exprime avec le mot-clé ON
- Les autres conditions s'expriment dans le WHERE et sont appliquées après la condition de jointure

Exemple de jointure interne

idE	nomE	moyenneLycée	effectifLycée
123	Ana	19.5	1000
234	Bob	18	1500
345	Chloé	17.5	500
456	Damien	19.5	1000
543	Chloé	17	2000
567	Éléonore	14.5	2000
654	Ana	19.5	1000
678	Farid	19	200
765	Joana	14.5	1500
789	Gisèle	17	800
876	Irène	19.5	400
898	Hector	18.5	800

Table Élève

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Étienne	21000

Table Université

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table Candidature

Les élèves qui ont candidaté dans une université grenobloise

```
34 SELECT e.idE, nomE
35 FROM Élève e INNER JOIN Candidature c ON e.idE
       = c.idE
36      INNER JOIN Université u ON c.nomU = u.nomU
37 WHERE u.ville = 'Grenoble';
```

idE	nomE
345	Chloe
543	Chloe
876	Irene
876	Irene

Syntaxe obsolète de jointure interne

SELECT att_1, att_2, ...
FROM nom_table_1, nom_table_2, ...
WHERE $nom\_table_1.att_x \ \Theta \ nom\_table_2.att_x$
[**AND** $autres\_conditions$] ;

- Jointure interne $\equiv$ sélection sur le produit cartésien
- La condition de jointure $nom\_table_1.att_x \ \Theta \ nom\_table_2.att_x$ s'exprime ici dans le WHERE

Syntaxe obsolète, à éviter !

Exemple de jointure interne obsolète

idE	nomE	moyenneLycée	effectifLycée
123	Ana	19.5	1000
234	Bob	18	1500
345	Chloé	17.5	500
456	Damien	19.5	1000
543	Chloé	17	2000
567	Éléonore	14.5	2000
654	Ana	19.5	1000
678	Farid	19	200
765	Joana	14.5	1500
789	Gisèle	17	800
876	Irène	19.5	400
898	Hector	18.5	800

Table Élève

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Étienne	21000

Table Université

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table Candidature

Les élèves qui ont candidaté dans une université grenobloise

```
39 SELECT e.idE, nomE
40 FROM Élève e, Candidature c, Université u
41 WHERE e.idE = c.idE AND c.nomU = u.nomU AND
       u.ville ='Grenoble';
```

idE	nomE
345	Chloe
543	Chloe
876	Irene
876	Irene

Exemple de jointure interne obsolète

idE	nomE	moyenneLycée	effectifLycée
123	Ana	19.5	1000
234	Bob	18	1500
345	Chloé	17.5	500
456	Damien	19.5	1000
543	Chloé	17	2000
567	Éléonore	14.5	2000
654	Ana	19.5	1000
678	Farid	19	200
765	Joana	14.5	1500
789	Gisèle	17	800
876	Irène	19.5	400
898	Hector	18.5	800

Table Élève

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Étienne	21000

Table Université

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table Candidature

Les élèves qui ont candidaté dans une université grenobloise

```
39 SELECT e.idE, nomE
40 FROM Élève e, Candidature c, Université u
41 WHERE e.idE = c.idE AND c.nomU = u.nomU AND
       u.ville ='Grenoble';
```

OBSOLÈTE

idE	nomE
345	Chloe
543	Chloe
876	Irene
876	Irene

Plan

Syntaxe

SELECT att_1, att_2, ...
FROM nom_table_1 < **LEFT** | **RIGHT** | **FULL** >
[**OUTER**] **JOIN** nom_table_2
ON $nom\_table_1.att_x = nom\_table_2.att_x$
[**WHERE** *autres_conditions*] ;

- Non exprimable en Algèbre Relationnelle
- Une requête avec jointure OUTER JOIN retourne les tuples qui remplissent la condition de la jointure, mais aussi certains tuples qui ne la satisfont pas
- Ces tuples qui ne satisfont pas la condition de jointure dépendent du mot-clé LEFT, RIGHT ou FULL

Syntaxe (2)

Sélection des tuples de la jointure externe :

- LEFT (ou RIGHT) : les tuples de la table de gauche (ou de droite) sans correspondance dans l'autre table sont inclus dans le résultat avec une valeur NULL pour les attributs de l'autre table
- FULL : toutes les lignes de chacune des tables sont retournées. Les lignes sans correspondance ont leurs attributs complétés par des valeurs NULL

Exemple de jointure externe à gauche

idE	nomE	moyenneLycée	effectifLycée
123	Ana	19.5	1000
234	Bob	18	1500
345	Chloé	17.5	500
456	Damien	19.5	1000
543	Chloé	17	2000
567	Éléonore	14.5	2000
654	Ana	19.5	1000
678	Farid	19	200
765	Joana	14.5	1500
789	Gisèle	17	800
876	Irène	19.5	400
898	Hector	18.5	800

Table ÉLÈVE

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Étienne	21000

Table UNIVERSITÉ

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table CANDIDATURE

Les élèves avec une moyenne supérieure à 19 et les éventuelles universités où ils/elles ont candidaté

```
45 SELECT DISTINCT e.idE, nomE, nomU
46 FROM Élève e LEFT OUTER JOIN
       Candidature c
47   ON c.idE = e.idE
48 WHERE moyenneLycee > 19;
```

idE	nomE	nomU
123	Ana	INSA
123	Ana	UCB
123	Ana	UJM
456	Damien	NULL
654	Ana	NULL
876	Irène	UCB
876	Irène	UJF

Exemple de jointure externe complète

idE	nomE	moyenneLycée	effectifLycée
123	Ana	19.5	1000
234	Bob	18	1500
345	Chloé	17.5	500
456	Damien	19.5	1000
543	Chloé	17	2000
567	Éléonore	14.5	2000
654	Ana	19.5	1000
678	Farid	19	200
765	Joana	14.5	1500
789	Gisèle	17	800
876	Irène	19.5	400
898	Hector	18.5	800

Table Élève

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Étienne	21000

Table Université

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table Candidature

Requête identique, mais avec un FULL OUTER JOIN

```
50 SELECT e.idE, nomE, nomU
51 FROM Élève e FULL OUTER JOIN
       Candidature c
52   ON c.idE = e.idE
53 WHERE moyenneLycee > 19;
```

idE	nomE	nomU
123	Ana	UJM
123	Ana	INSA
123	Ana	UCB
456	Damien	NULL
654	Ana	NULL
876	Irène	UJF
876	Irène	UCB

Exemple de jointure externe complète (2)

idE	nomE	moyenneLycée	effectifLycée
123	Ana	19.5	1000
234	Bob	18	1500
345	Chloé	17.5	500
456	Damien	19.5	1000
543	Chloé	17	2000
567	Éléonore	14.5	2000
654	Ana	19.5	1000
678	Farid	19	200
765	Joana	14.5	1500
789	Gisèle	17	800
876	Irène	19.5	400
898	Hector	18.5	800

Table Élève

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Étienne	21000

Table Université

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table Candidature

Pourquoi la requête FULL OUTER JOIN donne le même résultat que celle avec LEFT OUTER JOIN ?

Plan

Syntaxe

SELECT *nom_table*$_2$.*
FROM *nom_table*$_1$ **NATURAL JOIN** *nom_table*$_2$;

- Une semi-jointure est une jointure qui ne garde dans le résultat que les attributs d'une seule table (ici les attributs de *nom_table*$_2$)
- Construite avec *nom_table*.* dans la clause SELECT (tous types de jointures acceptés, i.e., interne, externe)

Exemple de semi-jointure

idE	nomE	moyenneLycée	effectifLycée
123	Ana	19.5	1000
234	Bob	18	1500
345	Chloé	17.5	500
456	Damien	19.5	1000
543	Chloé	17	2000
567	Éléonore	14.5	2000
654	Ana	19.5	1000
678	Farid	19	200
765	Joana	14.5	1500
789	Gisèle	17	800
876	Irène	19.5	400
898	Hector	18.5	800

Table Élève

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Étienne	21000

Table Université

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table Candidature

Les informations sur les élèves qui ont candidaté dans une université grenobloise

```
61 SELECT e.*
62 FROM Élève e NATURAL JOIN
       Candidature c
63   NATURAL JOIN Université
64 WHERE ville = 'Grenoble';
```

idE	nomE	moyenneLycee	effectifLycee
345	Chloe	17.5	500
543	Chloe	17	2000
876	Irene	19.5	400
876	Irene	19.5	400

Plan

Syntaxe

SELECT t1.att_1, t2.att_1, ...
FROM nom_table_1 t1 **NATURAL JOIN** nom_table_1 t2 ;

- Auto-jointure = jointure d'une table avec elle même (tous types de jointures acceptés, i.e., interne, externe), en utilisant des alias de table (ici *t1* et *t2*)
- Exemples fréquents d'auto-jointure : *personne/parents*, *employée/supérieure hiérarchique*, *pièce/composant*

Exemple d'auto-jointure

idE	nomE	moyenneLycée	effectifLycée
123	Ana	19.5	1000
234	Bob	18	1500
345	Chloé	17.5	500
456	Damien	19.5	1000
543	Chloé	17	2000
567	Éléonore	14.5	2000
654	Ana	19.5	1000
678	Farid	19	200
765	Joana	14.5	1500
789	Gisèle	17	800
876	Irène	19.5	400
898	Hector	18.5	800

Table Élève

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Étienne	21000

Table Université

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table Candidature

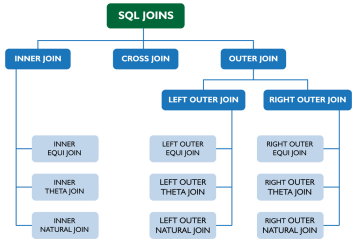
Les paires d'élèves qui ont candidaté dans la même université et le même département

```
68 SELECT DISTINCT c1.idE, c2.idE
69 FROM Candidature c1 INNER JOIN Candidature
       c2 ON c1.nomU = c2.nomU
70   AND c1.département = c2.département
71   AND c1.idE < c2.idE;
```

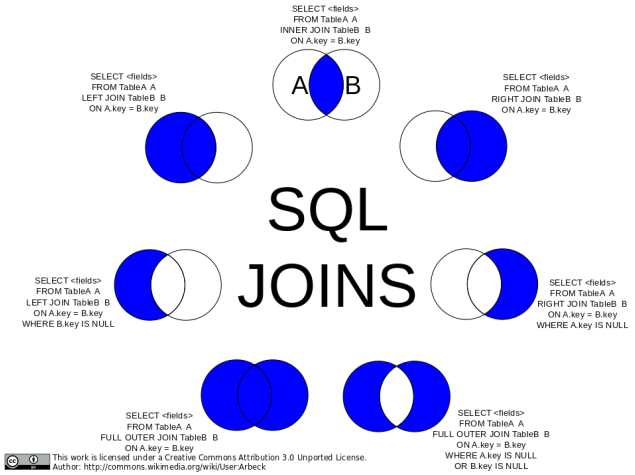
c1.idE	c2.idE
123	898
123	876
123	345
678	765
876	898

Classification des jointures

- La condition = naturelle, équi-jointure, θ-jointure
- Les n-uplets conservés dans le résultat = jointure interne, jointures externes (et produit cartésien)
- Les attributs conservés dans le résultat = semi-jointure



En résumé



À suivre : optimisation

Exemple de deux requêtes à résultat équivalent ?

Bilan

Éléments de langage

1. Basics
   - Projection (select... from)
   - Sélection (where)
   - Chaînes
   - Tri, limites
   - Agrégation (group by), et calculs
2. Jointures

Bases de données (CS443)

#5, Modèle relationnel: optimisation de requêtes

Laure Gonnord

Grenoble INP/Esisar

2022-2023

Crédits

Transparents F. Duchateau, pour univ Lyon1, CC by SA.

`https://perso.liris.cnrs.fr/fabien.duchateau/BDW1/`

Motivation

Motivation (2)

Une requête SQL est exécutée par le SGBD, mais :

- Quel est l'impact des différentes manières d'écrire une requête pour un même résultat (e.g., NOT IN / NOT EXISTS) ?
- Comment passe t-on de la requête (définie dans un langage déclaratif) à un programme (impératif) manipulant les données ?
- Comment optimise t-on une requête afin de l'exécuter efficacement pour trouver un résultat correct ?
- Pourquoi les requêtes préparées permettent de gagner en performance (entre autre) ?

---

http://sqlpro.developpez.com/cours/optimiser/

Optimiseurs

L'optimiseur de requêtes d'un SGBD est un composant crucial :

- Il réécrit la requête sous différentes formes
- Il choisit la réécriture la plus performante pour exécuter la requête
- Des benchmarks évaluent et comparent les SGBD, notamment les performances de leur optimiseur

---

Jarke, Matthias, and Jurgen Koch. *Query optimization in database systems*. ACM Computing surveys (1984)

Graefe Goetz. *Query evaluation techniques for large databases*. ACM Computing Surveys (1993)

Ioannidis, Yannis. *Query Optimization*. The Computer Science and Engineering Handbook (1996)

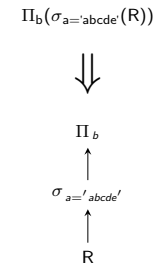
Plan

Traitement d'une requête

Où l'on retrouve l'algèbre relationnelle...

Une requête en algèbre relationnelle peut se représenter sous forme d'arbre algébrique

- Racine de l'arbre = résultat de la requête
- Feuilles de l'arbre = relations
- Noeuds intermédiaires de l'arbre = un opérateur algébrique (e.g., sélection, union, jointure)



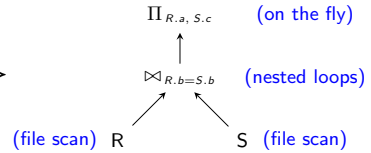
Arbre algébrique ≈ plan d'exécution d'une requête

Où l'on retrouve l'algèbre relationnelle... (2)

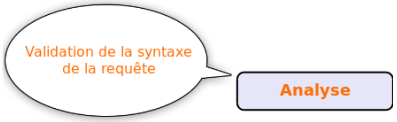
- Le plan d'exécution d'une requête est généralement optimisé selon trois opérateurs (projection, sélection et jointure)
- Les autres opérateurs (e.g., regroupement, tri) sont réalisés ensuite (ajoutés au plan optimisé des trois opérateurs)
- Un plan d'exécution indique quel algorithme est appliqué pour chaque opérateur

SELECT R.a, S.c
FROM R, S
WHERE R.b = S.b ;

$\Rightarrow$



Étape d'analyse

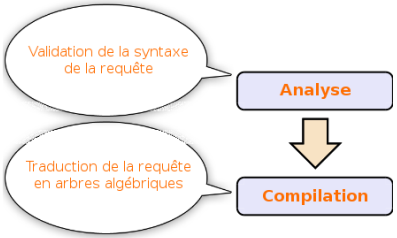


Étape d'analyse - détails

Analyse lexicale et syntaxique :

- Validation par rapport à la syntaxe SQL
- Vérification des types
  - présence des attributs et relations dans le schéma
  - compatibilité des types dans les prédicats
- Décomposition en requêtes/sous-requêtes

Étape de compilation



Étape de compilation - détails

Règles de passage d'une requête SQL en AR :

- SELECT pour définir une ??**projection**
- FROM pour définir les ?? **relations** (feuilles de l'arbre) et ??**jointures / produits cartésiens**
- WHERE pour définir :
  - avec les comparaisons *attribut/valeur* des ??**sélections**
  - avec les comparaisons *attribut/attribut* des ??**jointures**

Étape de compilation - exemple

SELECT R.a
FROM R, S
WHERE R.b = S.b
  AND S.c = 99 ;

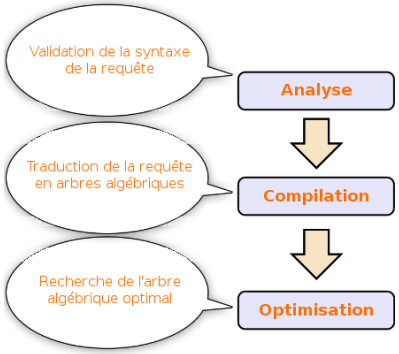
Compilation "naïve" :

- $\pi_a(\sigma_{R.b=S.b \wedge c=99}(R \times S))$

Autres possibilités :

- $\pi_a(\sigma_{R.b=S.b}(R \times \sigma_{c=99}(S)))$
- $\pi_a(R \bowtie_{R.b=S.b} (\sigma_{c=99}(S)))$

Étape d'optimisation



Étape d'optimisation - définition

Optimiser, c'est trouver le plan d'exécution d'une requête dont le coût soit minimal (i.e., le temps de réponse à la requête soit le plus rapide possible)

Qu'est-ce qui peut être optimisé dynamiquement ?

- Les accès disques

Il est indispensable que le temps lié à l'optimisation soit négligeable par rapport au temps imparti à l'exécution

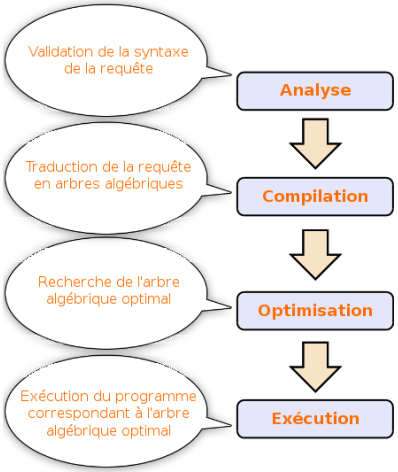
Étape d'optimisation - détails

Intuitions pour optimiser et sélectionner le plan optimal :

- Des compositions de projections et/ou sélections peuvent se réécrire en une opération de filtrage
- Réduire au plus tôt la taille et le nombre de tuples manipulés :
  - tuples moins nombreux : grâce à la ??**sélection**
  - tuples plus petits : grâce à la ??**projection**
- Réordonner les jointures, en fonction de la taille des relations manipulées
- Choisir des algorithmes efficaces pour chaque opérateur (e.g., *merge join*, *hash join* pour la jointure)

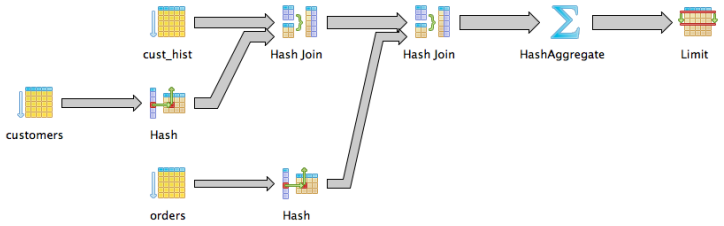
Explications détaillées dans les parties suivantes

Étape d'exécution



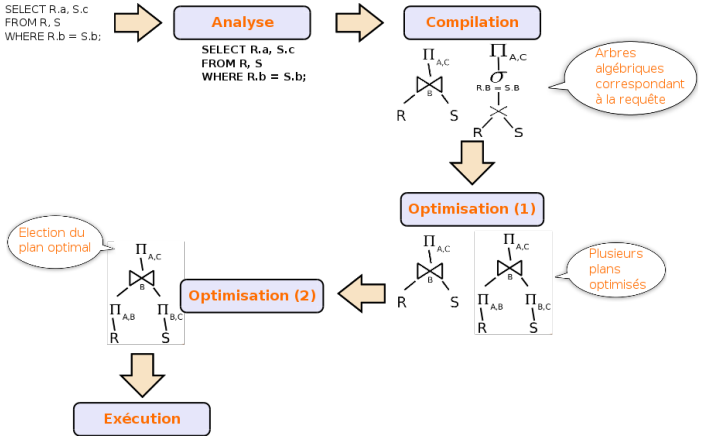
Étape d'exécution - détails

Application des algorithmes pour chaque opération du plan d'exécution sélectionné



http://mariadb.com/kb/en/mariadb/analyze-and-explain-statements/

Exemple complet de traitement d'une requête



En résumé

Traitement d'une requête :

- Validation
- Compilation en arbre algébrique
- Optimisation (génération de plans optimisés et sélection du plan optimal)
- Exécution du plan optimal

http://vidberg.blog.lemonde.fr/

Plan

Optimisation

L'objectif est de trouver une stratégie d'évaluation permettant d'accélérer l'évaluation :

- Par manipulations algébriques, indépendante de la manière dont sont stockées les informations
- En utilisant une stratégie dépendante du stockage (clés, index)

Dans ce cours, manipulations algébriques (à base de règles)

Plan d'exécution

Un plan d'éxécution $\approx$ arbre algébrique

Plus formellement, un **plan d'exécution** est un programme qui vise à évaluer une requête en algèbre relationnelle et qui consiste en une suite d'étapes parmi les suivantes :

- Application d'un opérateur unaire (sélection ou projection)
- Application d'une sélection puis d'une projection
- Application d'un opérateur binaire ($\times, \cup, \cap, \setminus, \bowtie$)
  - éventuellement précédé et/ou suivi d'opérateurs unaires

Un algorithme d'optimisation a pour but de trouver un bon (le meilleur si possible) plan d'exécution

Espace de recherche

Espace de recherche = l'ensemble des plans "équivalents" pour une même requête :

- Ceux qui donnent le même résultat
- Générés en appliquant des règles de transformation

Caractéristiques des plans d'un espace de recherche :

- Le coût de chaque plan est en général différent
- L'ordre des jointures est important

Si l'espace de recherche est très grand, l'optimiseur peut chercher un plan raisonnable, mais pas forcément optimal (par exemple sous PostgreSQL, dès qu'il y a plus de 12 jointures)

Exemple d'espace de recherche

ÉLÈVE (idE, *nomE*, *moyenneLycee*, *effectifLycee*)
CANDIDATURE (#*idE*, #*nomU*, département, *décision*)
UNIVERSITÉ (nomU, *ville*, *effectif*)

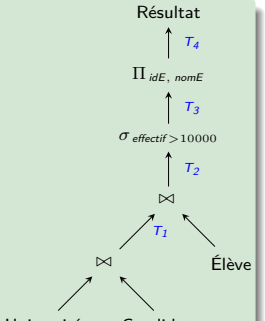
L'identifiant et le nom des élèves qui ont candidaté dans des universités avec un effectif supérieur à 10000

```
SELECT DISTINCT idE, nomE
FROM Élève e INNER JOIN Candidature c
     ON e.idE = c.idE INNER JOIN
     Université u ON c.nomU = u.nomU
WHERE effectif > 10000 ;
```


Exemple d'espace de recherche - plan 1

```
SELECT DISTINCT idE, nomE
FROM Élève e INNER JOIN Candidature c
     ON e.idE = c.idE INNER JOIN
     Université u ON c.nomU = u.nomU
WHERE effectif > 10000 ;
```

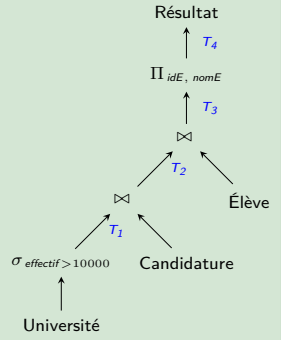
- $T_1 \leftarrow$ Joindre la table UNIVERSITÉ avec la table CANDIDATURE
- $T_2 \leftarrow$ Joindre T_1 avec la table ÉLÈVE
- $T_3 \leftarrow$ Lire T_2 et sélectionner les tuples d'*effectif* > 10000
- $T_4 \leftarrow$ Projeter T_3 sur *idE*, *nomE*



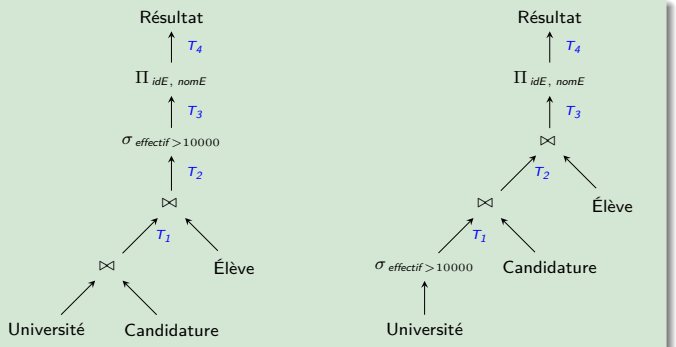
Exemple d'espace de recherche - plan 2

```
SELECT DISTINCT idE, nomE
FROM Élève e INNER JOIN Candidature c
     ON e.idE = c.idE INNER JOIN
     Université u ON c.nomU = u.nomU
WHERE effectif > 10000 ;
```

- $T_1 \leftarrow$ Lire la table ÉLÈVE et sélectionner les tuples d'*effectif* > 10000
- $T_2 \leftarrow$ Joindre T_1 avec la table CANDIDATURE
- $T_3 \leftarrow$ Joindre T_2 avec la table ÉLÈVE
- $T_4 \leftarrow$ Projeter T_3 sur *idE*, *nomE*



Exemple d'espace de recherche - choix de plan



Espace de recherche contenant deux plans d'exécution. Quel est le meilleur plan (sans considérer d'autres informations) ?

Optimisation à base de règles

Application de règles pour transformer une expression de l'algèbre relationnelle en plan d'exécution optimisé :

- Entrée : arbre représentant l'expression à évaluer
- Sortie : plan d'exécution pour la requête

L'arbre passé en entrée est construit comme suit :

- Les noeuds internes de l'arbre sont les opérateurs de l'expression
- Chaque noeud a pour fils le ou les sous-arbres construits à partir de son ou ses arguments
- Les feuilles sont des relations de la bases de données

Règles de transformation

Idées :

- Effectuer les sélections le plus tôt possible
- Combiner les sélections et les produits cartésiens pour faire des jointures
- Combiner les séquences d'opérations unaires, comme les sélections et les projections
- Chercher les sous-expressions communes dans une expression

Pour cela, on exploite des identités remarquables de l'algèbre relationnelle

Lois sur les jointures et les produits

Commutativité :

$$E_1 \bowtie_C E_2 \equiv E_2 \bowtie_C E_1 \quad (1)$$
$$E_1 \bowtie E_2 \equiv E2 \bowtie E_1 \quad (2)$$
$$E_1 \times E_2 \equiv E_2 \times E_1 \quad (3)$$

Associativité :

$$E_1 \bowtie_C (E_2 \bowtie_D E_3) \equiv (E_1 \bowtie_C E_2) \bowtie_D E_3 \quad (4)$$
$$E_1 \bowtie (E_2 \bowtie E_3) \equiv (E_1 \bowtie E_2) \bowtie E_3 \quad (5)$$
$$E_1 \times (E_2 \times E_3) \equiv (E_1 \times E_2) \times E_3 \quad (6)$$

Lois sur les projections et les sélections

Cascade de projections :

$$\pi_{A_1,\ldots,A_n}(\pi_{B_1,\ldots,B_k}(E)) \equiv \pi_{A_1,\ldots,A_n}(E) \quad (7)$$

Cascade de sélections :

$$\sigma_C(\sigma_D(E)) \equiv \sigma_{C\wedge D}(E) \quad (8)$$
$$\sigma_C(\sigma_D(E)) \equiv \sigma_D(\sigma_C(E)) \quad (9)$$

Permutations de projections et sélections :

- Si C ne porte que sur des attributs parmi $A_1, \ldots, A_n$:

$$\pi_{A_1,\ldots,A_n}(\sigma_C(E)) \equiv \sigma_C(\pi_{A_1,\ldots,A_n}(E)) \quad (10)$$

Lois sur les sélections et les autres opérations

Sélections et produits cartésiens/jointures :

- Si C ne porte que sur les attributs de E_1 :

$$\sigma_C(E_1 \times E_2) \equiv \sigma_C(E_1) \times E_2 \quad (11)$$
$$\sigma_{C\wedge D}(E_1 \times E_2) \equiv \sigma_D(\sigma_C(E_1) \times E_2) \quad (12)$$

- Si C ne porte que sur les attributs de E_1 et D sur les attributs de E_2 :

$$\sigma_{C\wedge D}(E_1 \times E_2) \equiv \sigma_C(E_1) \times \sigma_D(E_2) \quad (13)$$

Sélections et union / différence :

$$\sigma_C(E_1 \cup E_2) \equiv \sigma_C(E_1) \cup \sigma_C(E_2) \quad (14)$$
$$\sigma_C(E_1 - E_2) \equiv \sigma_C(E_1) - \sigma_C(E_2) \quad (15)$$

Projection et autres opérations

Projection et autres opérations :

- Produit cartésien :

$$\pi_{A_1,\ldots,A_i,\ldots,A_n}(E_1 \times E_2) \equiv \pi_{A_1,\ldots,A_i}(E_1) \times \pi_{A_{i+1},\ldots,A_n}(E_2) \quad (16)$$

- Union :

$$\pi_{A_1,\ldots,A_n}(E_1 \cup E_2) \equiv \pi_{A_1,\ldots,A_n}(E_1) \cup \pi_{A_1,\ldots,A_n}(E_2) \quad (17)$$

Application des règles

1. Utiliser (8) pour transformer les sélections $\sigma_{C_1\wedge\ldots\wedge C_n}(E)$ en cascades $\sigma_{C_1}(\ldots\sigma_{C_n}(E))$
2. Pour chaque sélection, utiliser (9), (10), (11), (12), (13), (14) et (15) pour pousser les sélections en bas de l'arbre
3. Pour chaque projection, utiliser les règles (7), (16), (17) et (10) pour pousser la projection au plus bas dans l'arbre. Éliminer une projection qui conserve tous les attributs
4. Utiliser les règles (7), (8) et (10) pour combiner les cascades de sélections et projections en une sélection unique, une projection unique, ou une sélection suivie d'une projection

Application des règles (2)

5. Partitionner l'arbre résultant en groupes comme suit :
   - Créer un groupe par noeud binaire ($\times, \cup, \cap, \setminus, \bowtie$)
   - Le groupe inclut tous les ancêtres unaires intermédiaires
   - Le groupe inclut toute branche étiquetée par des opérateurs unaires et se terminant à une feuille (table)

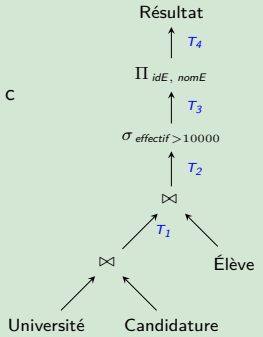
6. Chaque groupe correspond à une étape. Les étapes sont à évaluer dans n'importe quel ordre tant qu'un groupe est évalué après ses descendants

Exemple d'application des règles

L'identifiant et le nom des élèves qui ont candidaté dans des universités avec un effectif supérieur à 10000

SELECT DISTINCT idE, nomE
FROM Élève e **INNER JOIN** Candidature c
ON e.idE = c.idE **INNER JOIN**
Université u **ON** c.nomU = u.nomU
WHERE effectif > 10000 ;

⇒ Plan non optimisé !

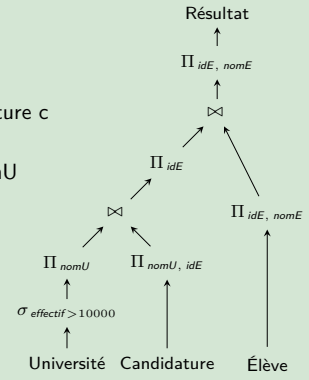


Exemple d'application des règles (2)

L'identifiant et le nom des élèves qui ont candidaté dans des universités avec un effectif supérieur à 10000

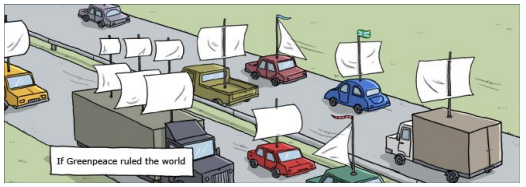
SELECT DISTINCT idE, nomE
FROM Élève e **INNER JOIN** Candidature c
ON e.idE = c.idE **INNER JOIN**
Université u **ON** c.nomU = u.nomU
WHERE effectif > 10000 ;

⇒ Plan optimisé par les règles : (11), (10), (16)



En résumé

- Optimisation à base de règles = application de règles de transformation entre les opérateurs de l'AR
- Une quinzaine de règles et un algorithme à suivre
- Idée générale : placer les sélections et les projections en bas de l'arbre (le plus tôt possible)



http://wumo.com/

Chapitre 1

Bonus

Bases de données (CS443)

#6 Transactions

Laure Gonnord

Grenoble INP/Esisar

2022-2023

Source

Anne-Cécile Caron, Anne Étien, Mikaël Monet, Sylvain Salvati pour université Lille1 et Polytech'Lille Merci pour le source complet LaTeX.

Pourquoi ?

Le concept de transaction va permettre de définir des processus garantissant que l'état de la base est toujours cohérent

- même en cas d'accès concurrents à la base.
- même en cas de panne logicielle ou matérielle.

Plan

Exemple

On considère le célèbre exemple de virement bancaire :

```
procedure virement(A,B,X) {
  A := A-X ;
  B := B+X ;
}
```

- A et B sont des "entités" de la base de donnée, X est la valeur du virement.
- A := A-X est une façon simplifiée d'écrire :
  `update COMPTE set solde = solde-X`
  `where refCompte = refA ;`
- Par la suite, un appel à cette procédure se fera à l'intérieur d'une *transaction*.

Lecture/Ecriture

```
procedure virement(A,B,X) {
  A := A-X ;
  B := B+X ;
}
```

Pour mettre en évidence les problèmes, on s'intéressera aux lectures et écritures faites par une transaction. La procédure de virement devient alors :

```
debut transaction
lire(A)
ecrire(A)
lire(B)
ecrire(B)
fin transaction
```


Premier problème

Un utilisateur exécute un virement bancaire :

```
debut transaction
lire(A)
ecrire(A)
PANNE SYSTEME
```

- Le rôle du système transactionnel est de garantir que la transaction se fait complètement ou pas du tout,
- il doit donc annuler la modification de A (`rollback`).
- Une transaction est *Atomique*.

Gestion de la concurrence

Deux utilisateurs exécutent des virements des mêmes comptes, à l'aide de deux transactions T1 et T2 :

```
T1 : virement(A,B,100)
T2 : virement(A,B,200)
```

Pour des raisons de performance, les actions des différentes transactions sont entrelacées. Il faut différencier les actions de T1 de celles de T2, et considérer l'ordre dans lequel ces actions vont s'exécuter.

Ordonnancement

Un ordonnancement est une séquence d'actions de la forme (nomTransaction, opération, donnée).

Exemple d'ordonnancement O_1 de T1 et T2 :

```
(T1, lire, A)
(T1, ecrire, A)
(T2, lire, A)
(T2, ecrire, A)
(T1, lire, B)
(T1, ecrire, B)
(T2, lire, B)
(T2, ecrire, B)
```


Deuxième problème

```
T1 : virement(A,B,100)
T2 : virement(A,B,200)
```

Considérons l'ordonnancement O_2

```
(T1, lire, A)
(T2, lire, A)
(T1, ecrire, A)
(T1, lire, B)
(T1, ecrire, B)
(T2, ecrire, A)
(T2, lire, B)
(T2, ecrire, B)
```

Exercice : quelles sont les modifications faites par les transactions `T1` et `T2`.

Deuxième problème

```
T1 : virement(A,B,100)
T2 : virement(A,B,200)
```

Considérons l'ordonnancement O_2

```
(T1, lire, A)
(T2, lire, A)
(T1, ecrire, A)
(T1, lire, B)
(T1, ecrire, B)
(T2, ecrire, A)
(T2, lire, B)
(T2, ecrire, B)
```

Exercice : quelles sont les modifications faites par les transactions `T1` et `T2`.
Ici, on a perdu une instruction de A et la base est dans un état *inconsistant*. Les effets des transactions sont modifiés à cause de la *concurrence*.

Propriétés des ordonnancements

- Deux actions d'un ordonnancement sont *conflictuelles* si elles concernent la même entité et qu'au moins l'une des deux est une écriture.
- Deux ordonnancements O_1 et O_2 des mêmes transactions sont *équivalents* si pour toutes actions conflictuelles a et a', a est avant a' dans O_1 ssi a est avant a' dans O_2.
- Un ordonnancement est *sérialisable* s'il est équivalent à une exécution en série des transactions.
- Seuls les ordonnancements sérialisables sont corrects.

Example

- L'ordonnancement O_1 est sérialisable car équivalent à `T1;T2`.
- L'ordonnancement O_2 n'est pas sérialisable.

Plan

Synthèse : Concept de transaction

- Une transaction est un programme qui modifie la base de données et forme une unité de traitement.
- Elle doit respecter les propriétés ACID
  - Atomicité : une transaction s'effectue entièrement ou pas du tout
  - Consistance : Une transaction qui prend la base dans un état cohérent doit la rendre dans un état cohérent.
  - Isolement : pas d'interférence avec les utilisateurs concurrents.
  - Durabilité : Les actions effectuées par une transaction terminée sont prise en compte dans la base de données.

Concept de transaction (2)

- Les transactions sont gérées par un moniteur transactionnel.
- Une transaction peut être dans différents états :
  - Active : pendant le déroulement du programme, tant qu'aucun problème n'apparaît.
  - Partiellement validée : Lorsque la dernière instruction a été atteinte
  - Validée : Après une exécution totalement terminée (ordre `commit`)
  - Echouée : après un problème qui a interrompu la transaction
- L'instruction `commit` permet de signaler que la transaction s'est bien terminée, les modifications qu'elle a effectuées sont rendues visibles aux autres transactions.
- Si une transaction échoue, le `rollback` permet d'annuler toutes les actions effectuées par cette transaction.

Plan

Norme SQL2 - 1992

La norme définit deux caractéristiques pour une transaction :

- Le mode, i.e. les opérations possibles,
  - `READ ONLY` *transaction-level read consistency*
  - `READ WRITE` (par défaut) *statement-level read consistency*
- Le niveau d'isolement : Le TP illustre quelques problèmes que l'on peut rencontrer avec SQL utilisé de manière concurrente. La norme définit des niveaux d'isolement pour empêcher ces problèmes.

Norme SQL3 - 1999

- *points de contrôle* dans les transactions.

```
begin
  insert into joueur
    values ('165789','Bisk','Otto');
  savepoint p1;
  insert into joueur
    values ('376487','Biss','Scott');
  rollback to p1;
  commit ;
end ;
```

  La première instruction `insert` est validée, pas la seconde.
- Le concept de transaction atomique est étendu à celui de **transactions imbriquées**
- Ces deux ajouts permettent de manipuler des transactions plus longues, mais étant composées de sous-transactions (courtes) qui peuvent être annulées indépendamment.
- Les points de contrôle sont juste un support pour la forme la plus simple d'imbrication (1 niveau)

Niveau d'isolement de la norme SQL

- `READ UNCOMMITTED` : aucun isolement des transactions. On peut avoir des lectures inconsistantes (dirty read) des tables, i.e. lire une valeur modifiée par une autre transaction mais non validée.
- `READ COMMITTED` : évite les lectures inconsistantes. On ne lit que des données dont les modifications ont été validées.
- `REPEATABLE READ` : empêche le problème des lectures répétées tq entre 2 lectures, certaines lignes ont disparu ou ont été modifiées. N'empêche pas le problème des lignes "fantômes", ajoutées entre une lecture et la suivante.
- `SERIALIZABLE` : Garantit la sériabilité des ordonnancements. Évidemment ça pose des problèmes de performance.

Demo ?

TODO

Bilan

1. Problèmes de pannes et concurrence
   - Concept de transaction
2. Norme SQL