Bases de données avancées

Introduction

Équipe pédagogique BD

`https://perso.liris.cnrs.fr/marc.plantevit/doku/doku.php?id=lifbdw2_2020a`

Version du 7 septembre 2020

Objectif de l'enseignement

Approfondir les connaissances du modèle relationnel et les fondements de la conception de bases de données :

- Connaitre les principes des dépendances : systèmes d'inférences, fermeture, bases de règles, énumération
- Savoir concevoir et normaliser une BD,
- Savoir concevoir des BDs en SQL et programmer des déclencheurs en PL/pgSQL.

Pré-requis

Modèle relationnel

- Structure : attributs, relations, bases de données.
- Langages du relationnels : algèbre et calcul relationnel, SQL
- Modélisation Entité/Association.

Maîtriser les contenus de l'UE LIFBDW1 *bases de données et programmation web*
`http://liris.cnrs.fr/~fduchate/BDW1`

Bases théoriques

- éléments de théorie des ensembles (graphes, arbres, treillis)
- éléments de théorie des langages (c.f. LifLF : mots, langages formels, notions de complexité)
- éléments logique classique (c.f. LifLC : syntaxe et sémantique de la logique propositionnelle, système de déduction, logique du premier ordre) ;

Références

- M. Levene, G. Loizou. *A Guided Tour of Relational Databases and Beyond*. Springer, 1999.
- S. Abiteboul, R. Hull et V. Vianu. *Fondements des bases de données.* Addison-Wesley, 1995.
- J. Widom. *Introduction to Databases.* sur `www.db-class.org/db/Winter2013`.
- R. Ramakrishnam et J. Gehrke. *Database Management Systems*. 2003 (troisième édition).
- Matériel disponible en ligne allant avec la référence précédente : `pages.cs.wisc.edu/~dbbook/`
- J.-L. Hainaut. *Bases de données : Concepts, utilisation et développement*. Dunod, 2012 (deuxième édition).

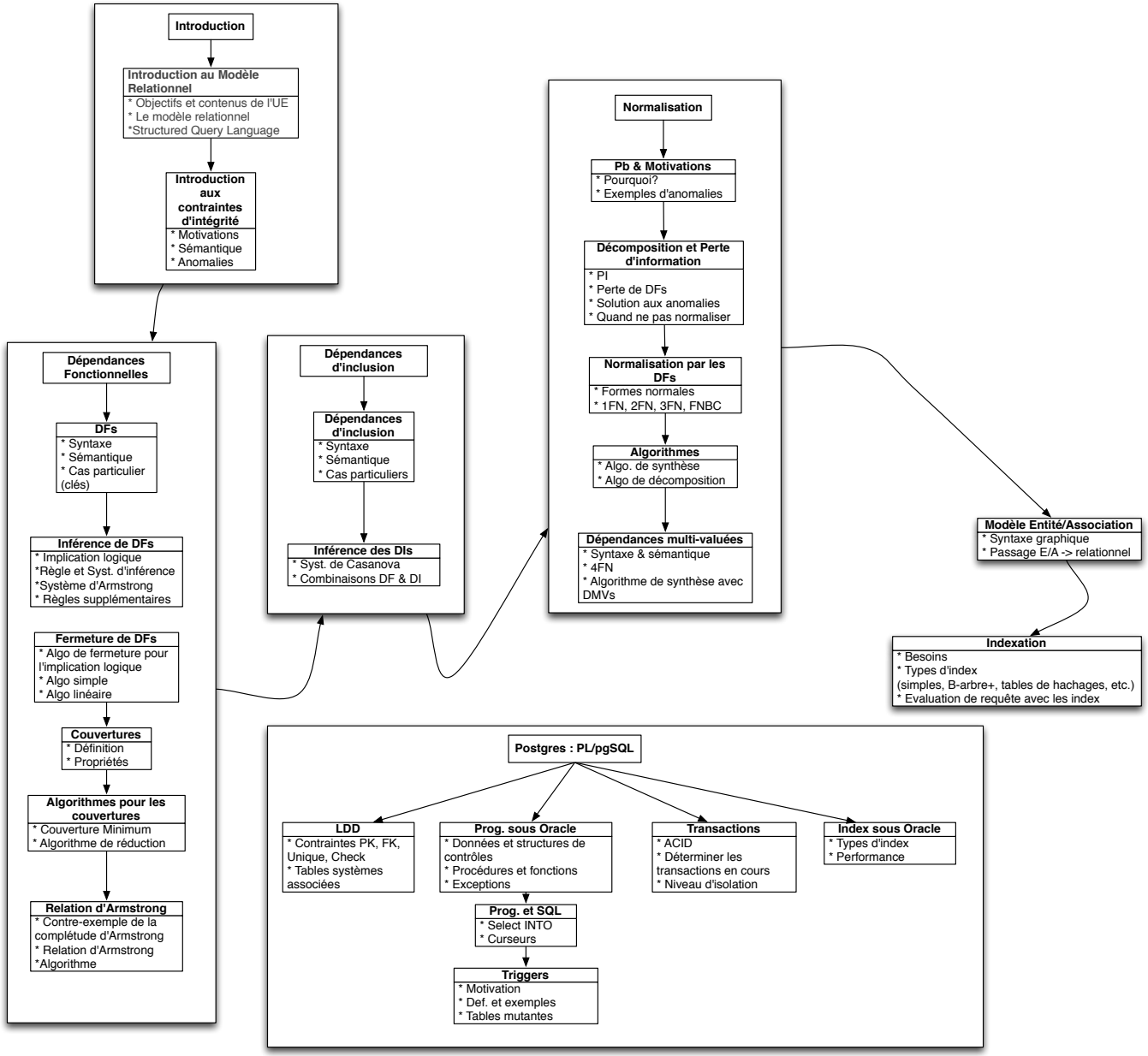
Organisation du cours

- Introduction, rappels (2h)
- Contraintes et dépendances du modèle relationnel (4h)
- PL/pgSQL (4h)
- Conception de BD (4h)
- Structures d'indexation (2h)
- Introduction au modèle XML (2h)

Modalités : CC (1/3) + SESSION1 (2/3) + SESSION2

- CC : 45 minutes en CM
- TP noté :
- 1 contrôle terminal : 1h30.

Détails sur les contenus



Détails sur les contenus

`https://perso.liris.cnrs.fr/marc.plantevit/doku/doku.php?id=lifbdw2_2020a`

Aide-mémoire

- Une synthèse des principaux contenus (lien)
- Fourni avec le CCF (mais pas les CCs)

Organisation des enseignements

Apprentissage actif en TD-TP

- les sujets des TDs et TPs sont transmis à l'avance
- les exercices et questions marqués du symbole (†) sont à préparer

Classe inversée en CM

- les slides sont transmis à l'avance
- les slides sont complétés par des vidéos
- vous prenez connaissance des supports avant le CM

Mail à la promotion pour le mardi avant la semaine suivante

Pourquoi la classe inversée ?

- Vous pouvez suivre le cours quand vous voulez
- Vous avez le temps de digérer et de reprendre
- Un face-à-face en amphi moins formel et plus libre

Principes

Base de données

Une grande quantité de données stockée dans un ordinateur.

Qu'est ce qu'un SGBD ?

Un logiciel qui permet de stocker et manipuler de grandes bases de données.

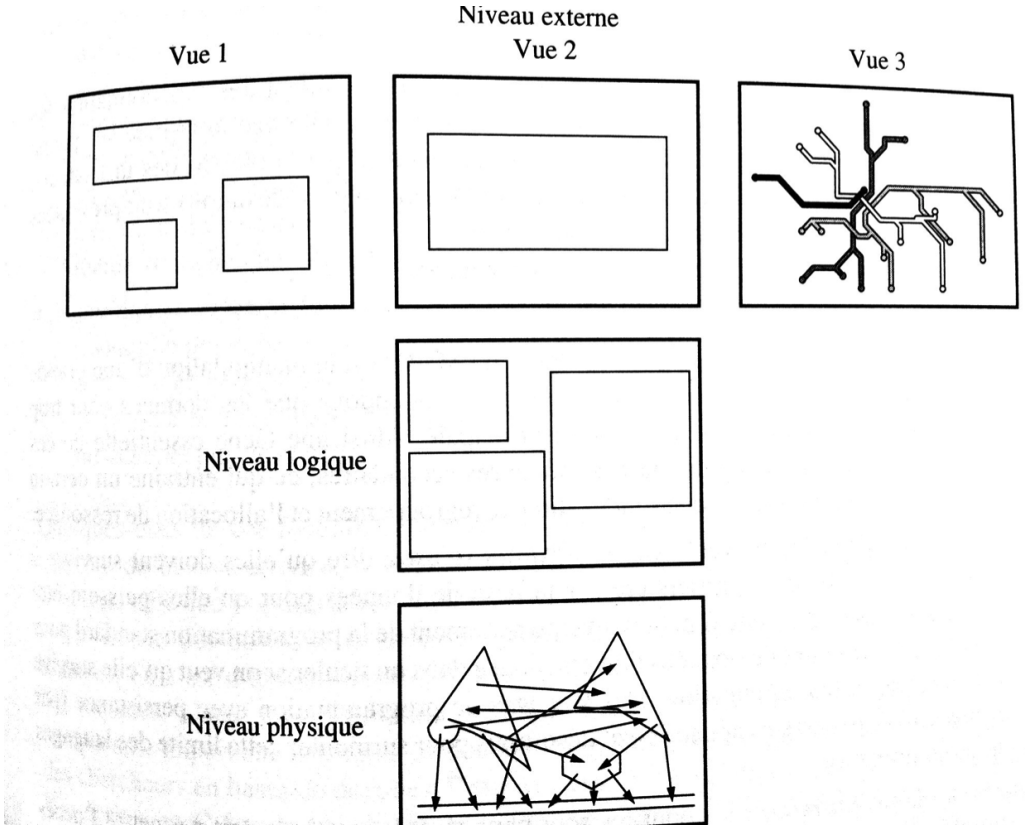
Fonctionnalités d'un SGBD

- Persistance ;
- Contrôle de concurrence ;
- Protection des données ;
- Interface homme-machine ;
- Distribution ;
- Compilation et optimisation.

Architecture 3 niveaux

Détacher l'utilisateur de l'implémentation physique.

- physique,
- logique,
- applicatif.



Séparation logique / physique : principe d'indépendance des données

- Définition de *modèle logique de données* : séparer physique et logique pour permettre à l'utilisateur de se concentrer uniquement sur une représentation logique des données sans se soucier de la représentation physique des données.
  - langage de définition des données (LDD), pour définir les aspects structurels des données.
  - langage de manipulation des données (LMD) pour y accéder et les mettre-à-jour.
- plusieurs aspects de l'implémentation physique peuvent être changés sans avoir à modifier la vision abstraite de la base de données.
- distinction fondamentale qui existe entre les systèmes de fichiers et les bases de données.

Séparation niveaux physique et applicatif

- permet des points de vue différent sur les bases de données qui sont adaptés à des besoins spécifiques.
- les vues cachent ainsi les informations sans intérêt et restructure les données qu'elles retiennent.

Complexité et diversité

Applications : finances, personnel, inventaire, ventes, dossiers, biologie, etc.

Utilisateurs : programmeurs d'applications, sav, secrétaires, administrateurs de bases de données, geek, etc.

Modes d'accès : LMD linéaires et graphiques, interfaces graphiques, entrée des données, génération de rapports, etc.

Modèles logiques : les plus diffusées sont les modèles en *réseau/graphes*, *hiérarchique*, *relationnel*, *déductif*, *orienté objet*. Émergence de nouveaux paradigmes de type *clefs-valeurs* ou *column-wise*...

Modèles de données

Tout SGBD est conçu autour d'un modèle de données bien défini. Il s'agit de la combinaisons de 3 éléments :

- Une structure, qui correspond à l'organisation logique des données, la forme sous laquelle les utilisateurs vont les percevoir ou les représenter.
- Des contraintes d'intégrité, que l'on peut définir sur les données, afin d'en assurer l'intégrité et la cohérence avec le monde réel et les besoins des applications.
- Des langages d'interrogation de données, pour les requêtes en lecture, et des langages de manipulation des données, pour les mises à jour.

- *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.

- *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.

- *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).

Notations

- Un ensemble $\{A_1; A_2; ...; A_n\}$ sera noté par la concaténation de ses éléments : $A_1A_2...A_n$.
- Si X et Y sont deux ensemble, leur union $X \cup Y$ sera notée XY.
- Une séquence (ou liste) d'éléments, est notée $< A_1, ..., A_n >$ ou $(A_1, ..., A_n)$.
- Séquences et ensembles seront notés de la même façon par concaténation quand le contexte le permet.
- Attention : dans le cas des séquences, $BAC \neq ABC \neq AABC$!

- $\{A; B; C\}$ 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 $XY = ABCD$.

Intuition : *relation = table*

Étudiants	*NUM*	*NOM*	*PRENOM*	*AGE*	*FORMATION*
	28	Codd	Edgar	20	3
	32	Armstrong	William	20	4
	53	Fagin	Ronald	19	3
	107	Bunneman	Peter	18	3

Enseignants	*NUM*	*NOM*	*PRENOM*	*GRADE*
	5050	Tarjan	Robert	Pr.
	2123	De Marchi	Fabien	MCF
	3434	Papadimitriou	Spiros	Pr.
	1470	Bagan	Guillaume	CR

TABLE – Exemple de bases de données

- 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.

Exemple

Une application gérant les cours dispensés au sein de de la licence, regroupant des informations sur les étudiants, les formations, les enseignants, les modules, les UE, les salles de cours et les ressources matérielles (projecteurs, etc...).
La modélisation des étudiants pourrait se faire à l'aide du *schéma de relation*

$$ETUDIANTS = \{NUM; NOM; PRENOM; AGE; FORMATION\}.$$

Si on représente les autres informations dans d'autres schémas de relations, alors l'union de tous les schémas de relation obtenus constituera le *schéma de la base de données*.

Tuple, Relation et Base de Données

- Soit $R = A_1 ... A_n$ un schéma de relation. Un tuple sur R est une fonction $t : R \rightarrow \mathcal{D}$ telle que $t(A) \in DOM(A)$.
- 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 **R**.

Alternativement, on peut voir un tuple comme une séquence de valeurs, prises par les attributs du schéma de relation, c'est-à-dire un élément du produit cartésien des domaines $DOM(A_1) \times \ldots \times DOM(A_n)$. C'est plutôt cette notation que l'on utilisera en pratique.

- 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.

Structure du modèle relationnel

Exemple

Prenons la relation *Étudiants* : Si on nomme t_1 le premier tuple, alors $t_1[NOM] = (Codd)$, $t_1[NUM, AGE] = (28, 20)$.
L'ensemble de toutes les relations "peuplées" par des tuples constitue la base de données.

Plusieurs langages ont été définis pour l'interrogation et la manipulation des données dans le modèle relationnel. *Le résultat d'une requête est toujours une relation* : les langages diffèrent sur la façon de définir la relation en sortie. Ils sont de deux sortes :

- *les langages procéduraux* : algèbre relationnelle,
- *déclaratif* : calcul relationnel (tuple ou domaine), Datalog.

Le langage SQL

Le SQL correspond à l'implantation du calcul relationnel de tuple : c'est donc un langage déclaratif. SQL possède toutefois de nombreuses extensions pour faciliter l'usage et augmenter le pouvoir d'expression.

Algèbre Relationnelle

Opérations

- Sélection ($\sigma_C(R)$) retourne le sous-ensemble de tuples de la relation R vérifiant la condition C.
- Projection ($\pi_X(R)$) supprime les colonnes non désirées de R.
- Jointure ($R_1 \bowtie_\theta R_2$) combine deux relations selon θ.
- Renommage ($\rho_{[x/x']}(R)$) opération syntaxique sur de renommage des attributs.
- Produit cartésien ($R_1 \times R_2$) combine librement deux relations.
- Division ($R_1 \div R_2$) retourne les tuples de R_1 qui apparaissent tous dans R_2.
- Différence ($R_1 \setminus R_2$) sémantique ensembliste.
- Intersection ($R_1 \cap R_2$) sémantique ensembliste.
- Union ($R_1 \cup R_2$) sémantique ensembliste.

Algèbre Relationnelle

Ces opérateurs sont inter-définissables, en théorie, on ne garde généralement qu'un jeu d'opérations restreint à :

- sélection, projection, jointure et renommage (SPJR),
- augmentés éventuellement de l'union (SPJRU) i
- et de la différence (SPJRUD).

Exprimer $R_1 \bowtie R_2$ et $R_1 \cap R_2$ à partir des autres opérateurs.

Calcul Relationnel à Variable Domaine (CRVD)

- Syntaxe :

$$\{x_1, \ldots, x_n \mid F\}$$

où F est une formule dont les variables libres sont $x_1, \ldots, x_n$ et composée à partir de :

  - De prédicats des relations de la base appliquées à des variables : $R(A_1 : y_1, \ldots, A_k : y_k)$
  - de comparaisons entre variables ou variables et constantes : $x \geq y$, $x = 123$, ...
  - De connecteurs logiques et quantificateurs : $\vee, \wedge, \Rightarrow, \forall, \exists, \ldots$

Calcul Relationnel à Variable Tuples (CRVT)

- Syntaxe :

$$\{x_1.A_1, \ldots, x_n.A_n \mid F\}$$

où F est une formule dont les variables libres sont $x_1, \ldots, x_n$ avec la possibilité d'avoir plusieurs fois la même variable x_i, composée de la manière suivante :

  - $R(x)$ où x est une variable représentant un tuple ;
  - de comparaisons : $x.A \bullet x'.A'$ ou $x.A \bullet c$ avec $\bullet \in \{<, >, \leq, \geq, =, \neq\}$ et c constante.
  - De connecteurs logiques et quantificateurs entre formules : $\vee, \wedge, \Rightarrow, \forall, \exists, \ldots$

Requêtes SQL Basiques

```
SELECT [DISTINCT] target-list
FROM relation-list
WHERE qualification
```

- relation-list : Une liste de noms de relation (possibilité de mettre des variables).
- target-list : Une liste d'attributs de relations appartenant à relation-list
- qualification : comparaisons (*Attr op const* ou *Attr1 op Attr2* où $op \in \{<, >, =, \leq, \geq, \neq\}$) combinées en utilisant AND, OR et NOT.
- DISTINCT est un mot-clé optionnel indiquant que le résultat ne doit pas contenir de duplications. Par défaut, les duplications ne sont pas éliminées !

Expressions et chaînes de caractères

- `Reserves[sid,bid,day]`
- `Sailors[sid,sname,rating,age]`
- `Boat[bid,color]`

```
SELECT S.age, age1=S.age-5, 2*S.age AS age2
FROM Sailors S
WHERE S.sname LIKE 'B_%B'
```

Utilisation d'expressions arithmétiques et de string matching

- Trouver des triplets (âge des marins, et 2 champs décrits par des expressions) pour les marins dont les noms commencent et finissent par B et contiennent au moins 3 caractères.
- AS et = sont deux façons de nommer les champs retournés.
- LIKE est utilisé pour les expressions régulières ('_' pour un caractère, '%' pour autant de caractères que l'on veut).

Union

Trouver les marins qui ont réservé un bateau rouge **ou** vert.

```
SELECT S.sid
FROM Sailors S, Boats B, Reserves R
WHERE S.sid=R.sid AND R.bid=B.bid
  AND (B.color='red' OR B.color='green')
```

```
SELECT S.sid
FROM Sailors S, Boats B, Reserves R
WHERE S.sid=R.sid AND R.bid=B.bid AND B.color='red'

UNION

SELECT S.sid
FROM Sailors S, Boats B, Reserves R
WHERE S.sid=R.sid AND R.bid=B.bid AND B.color='green'
```

Intersection

Que fait cette requête ?

```
SELECT S.sid
FROM Sailors S, Boats B, Reserves R
WHERE S.sid=R.sid AND R.bid=B.bid AND B.color='red'

INTERSECT

SELECT S.sid
FROM Sailors S, Boats B, Reserves R
WHERE S.sid=R.sid AND R.bid=B.bid AND B.color='green'
```

Requêtes Imbriquées

Trouver le nom des marins qui ont réservé le bateau #103.

```
SELECT S.sname
FROM   Sailors S
WHERE S.sid IN (SELECT R.sid
                FROM   Reserves R
                WHERE R.bid=103)
```

```
SELECT S.sname
FROM    Sailors S
WHERE EXISTS (SELECT *
              FROM   Reserves R
              WHERE    R.bid=103 AND S.sid=R.sid
```

Un peu plus sur la comparaison d'ensembles

- IN, EXISTS, NOT IN, NOT EXISTS.
- Aussi op ANY, op ALL, op IN $<, >, \leq, \geq, =, \neq$

Les marins dont la note est supérieure à celles de marin(s) nommé(s) Horatio.

```
SELECT *
FROM   Sailors S
WHERE  S.rating > ANY  (SELECT  S2.rating
                        FROM   Sailors S2
                        WHERE S2.sname='Horatio')
```

Opérateurs d'agrégation

- Une extension significative de l'AR (en quoi ?)
- `COUNT ([DISTINCT] [A|*]), SUM ([DISTINCT] A), AVG ([DISTINCT] A), MAX (A), MIN (A)`

Exemples

- `SELECT COUNT (*) FROM Sailors S`
- `SELECT AVG (S.age) FROM Sailors S WHERE S.rating=10`
- `SELECT COUNT (DISTINCT S.rating) FROM Sailors S WHERE S.sname='Bob'`
- `SELECT AVG ( DISTINCT S.age) FROM Sailors S WHERE S.rating=10`
- `SELECT S.sname FROM Sailors S WHERE S.rating= (SELECT MAX(S2.rating) FROM Sailors S2)`

GROUP BY et HAVING

```
SELECT [DISTINCT] target-list
FROM relation-list
WHERE qualification
GROUP BY grouping-list
HAVING group-qualification
```

- La target-list contient (i) des noms d'attributs (ii) des opérateurs d'agrégation (e.g., MIN(S.age)).
- La liste d'attributs (i) doit être un sous-ensemble de grouping-list.

L'âge du plus jeune marin majeur pour chaque note ayant au moins 2 marins

```
SELECT S.rating, MIN(S.age)
FROM   Sailors S
WHERE S.age>=18
GROUP BY S.rating
HAVING COUNT(*)>1
```

Pour chaque bateau rouge, trouver le nombre de réservations

```
SELECT B.bid, COUNT (*) AS scount
FROM   Sailors S, Boats B, Reserves R
WHERE S.sid=R.sid AND R.bid=B.bid AND B.color='red'
GROUP BY B.bid
```

Fin du cours.