Prise de notes : Base de données

Généralités sur le cours

- SQL pas au programme de l'examen (peu en fait)
- Méthode de normalisation à l'éxam

Table des matières

Définitions

Mot	***Définition***
Association	Relation (au sens mathématique) entre 2 entités
Arité	Nombre d'attributs d'un schéma relationnel
SPI	Sans perte d'information
FNBC	Forme Normale de Boyes-Codd
FN	Forme Normal

1. Introduction

Notation Pour l'exam Il lit notre schéma, en fait un résumé, et c'est le résumé qui est noté. Peut-être un peu de SQL, mais pas beaucoup, au max quelques requêtes.

Disclaimer On se penchera pas sur le fond de la crypto.

1.1. Attaques Cyber

Risques en cas de clic sur lien:

- Vol de données (via cookies par ex) ;
  - détruites (si extraites/existent plus sur appareil) ou chiffrées/cryptées (rançongiciel/ ransomware) ;
    - On peut le contrer/parer avec des sauvegardes (qui sont « isolées » des données pour éviter que le virus les atteigne).
  - Lecture de données : Si le pirate se contente de lire les données on ne sait pas si il les a lu et on risque une usurpation d'identité sans le savoir.
    - On le contre en chiffrant les données

Crypter les données est le principe des BDDs et pour les sauvegarder c'est mieux d'avoir toutes ses données au même endroit → BDDs centralisées.

- Et évite les risques liés au chiffrage/destruction.

1.1.1. Format d'un fichier POV Crypto:

2 mauvaises idées :

1. employer des listes chaînées ;
   - Si on compromet 1 truc dans la chaîne, tout ce qui arrive après est compromis ;
   - le sys d'exploitation (donc le hackeur) connais l'ordre d'entrée des données.
2. employer des tableaux.
   - l'ordre du tableau donne un indice à un éventuel pirate.

⇒ solution techinque est la **table de hachage (ou hashtable)**

La BDD est une réservation d'espace, la donnée est à une @ h (clef) avec h fonction de hachage compliquée à reverse sans la clé.

Modèle principal : modèle relationnel

Plan du cours :

- Modèle Entité-Association (pratique pour la conception d'un schéma BDD) ;
  - Employé dans **MERISE**
- Modèle relationnel ;
  - Définition ;
  - Normalisation relationnelle.

2. Modèle Entité-Association

Etonnant, on y trouve… des entités, ET des relations (non, jure) (dinguerie 😱)

2 concepts :

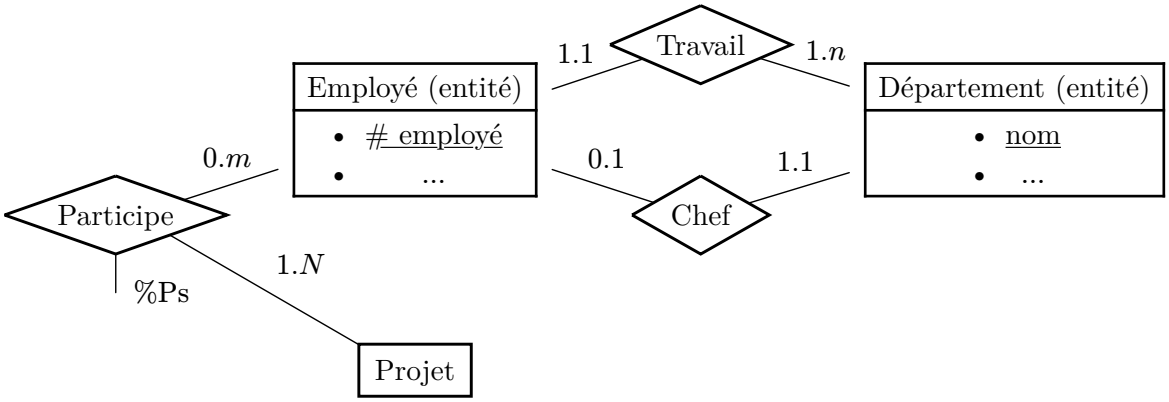
1. Entité ;
   - quelque chose qui existe dans le monde réel ;
   - Ce n'est pas imprimable ;
   - Ce n'est connu que par ses attributs ;

- les attributs sont imprimables ;
- Il y a forcément un identifiant parmi ces attributs, l'identifiant est souligné pour l'identifier ;
- Il est atomique (on mets pas plusieurs entrées dans un même attribut ex: pls prénoms dans le même champ prénom).

exemple

Personne (entité)
• nom ; • prénom ; • @ ; • date de naissance. • ...

2. Association
   - relation entre 2 entités (au sens mathématique) ;



Raisonnement multiplicité:

1. Employé - Département
   - Un employé travaille dans un et un seul département;
   - Un département comporte N employés ;
   - Un département comporte au moins 1 employé.
   - Donc multiplicité: 1...N ou « 1 dots ∞ ou 1... $*$ du coté département.

✨ 🎀 **Citation inspirante** 🎀 ✨ **:**
On est pas à l'armée mexicaine, tout le monde n'est pas chef

🌷 🌺 *Pascal Ostermann* 🦋 *14 mars 2025* 🌺 🌷

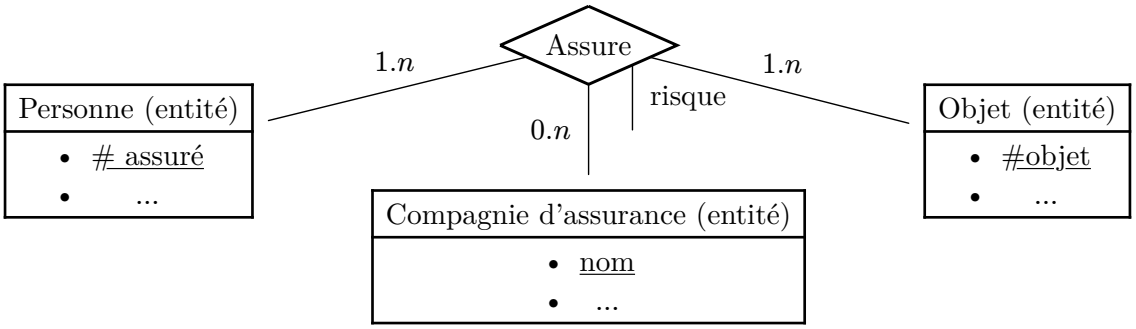
1. Employé - Projet
   - Un employé peut être sur 0 ou plusieurs projets
   - donc multiplicité 0...N do coté
   - Un projet a au moins un employé et peut en avoir plusieurs
   - donc multiplicité 1...N du coté

✨ 🎀 **Citation inspirante** 🎀 ✨ **:**
ça va être de la sodomie

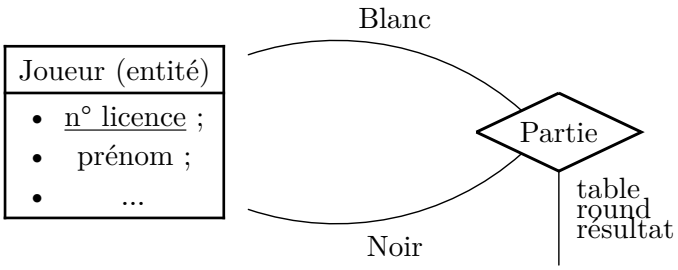
🌷 🌺 *Pascal Ostermann* 🦋 *14 mars 2025* 🌺 🌷

Il existe des associations ternaires, quaternaires, et en général N-aires.

Une relation R est arité SSI elle est N-aire.



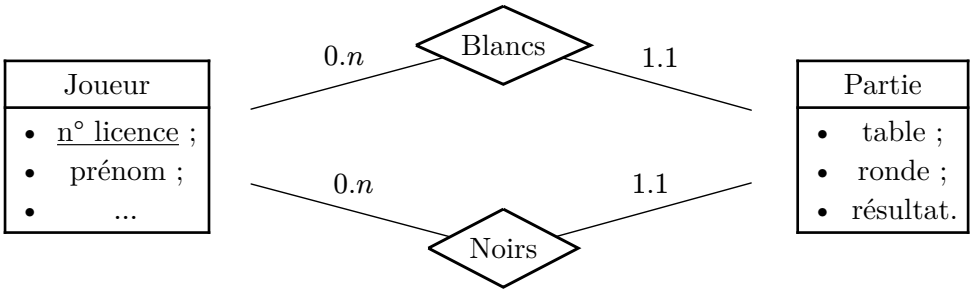
2.1. Exemple Partie d'échecs:



Multiplicité

1. Joueur/Partie
   - On peut avoir, au 1er round joué qu'une seule couleur (donc pas l'autre)
   - Au bout de plusieurs rounds, on peut avoir joué soit l'autre couleur, soit re la notre
   - donc multiplicité de chaque couleur: $0...N$.

2.1.1. Autre possibilité de la partie d'echec



✨ 🎀 **Citation inspirante** 🎀 ✨ **:**
Celui qui joue avec les noirs

🌷 🌺 *Pascal Ostermann* 🦋 *14 mars 2025* 🌺 🌷

2.2. Modèle de PP Chain (= ou == ?)

⇒ Se voulait à l'origine un modème d'interface entre le modèle relationnel et le model réseau CODASYL.

Sert maintenant pour parler avec un client et comme modèle de conception.

- A donné naissance à MERISE et MERSISE2/EURO MERISE.

Certains modèles Entité-Association rajoutent des constructeurs (souvent orienté objets).

Le prof proscrit tout ceci:

1. Attributs:
   - Structurés ;
   - Multivalués ;
   - Facultatifs (valeur NULL) .
2. IS_A (relation d'Héritage) ;
   - disjonction.
3. Entités faibles (qui n'ont pas d'identité) ;
4. Associations faibles (qui ne serait pas identifié par les entités qu'elle lie)

C'est surtout un modèle graphique qui peut se faire petit bout par petit bout.

- facilement compréhensible ;
- très facile à transformer en du relationnel.

Exemple :

- Personne (<u>num assuré</u>, nom, date) ;
- Objet_assuré (<u>num objet</u>, description) ;
- Compagine_assurance (<u>nom, @</u>, tél) ;
- Assure (<u>num assuré</u>, <u>num objet</u>, <u>nom, @</u>, risque)

Et on oublie pas la QoS

✨ 🎀 **Citation inspirante** 🎀 ✨ **:**
Mon dieu que j'écris mal

🌷 🌺 *Pascal Ostermann* 🦋 *14 mars 2025* 🌺 🌷

Je confirme *Pierre* | de même *Léa* | tout pareil *Nolann*

2.3. Modèle relationnel

2.3.1. Premier modèle étudié

$R(A_1 \cdot D_1, A_2 \cdot D_2, ..., A_n \cdot D_n)$

- R : nom de relation
- A_2 : nom d'attribut
- relation n-aire, d'arité n

On doit toujours avoir le domaine en tête, ici on les précise pas car évident

Exemple : Etudiant($\text{Numéro}_{\text{etu}}$, nom_{etu}, $\text{prénom}_{\text{etu}}$, $\text{titre}_{\text{cours}}$, $\text{heure}_{\text{cours}}$, $\text{salle}_{\text{cours}}$) **Ici, ça ne va pas car des infos peuvent être redondantes** car le fait qu'un cours ai lieu dans une salle sera dupliqué n_{eleve}-fois (une fois par élève)

ex de n_{uplet} : $(666, \text{Felixe}, \text{Ceras}, \text{Info}, 14h, \text{A002})$

les infos: (Info, 14h, A002) sont répété autant de fois qu'il y a d'étudiant en info, il y a une **redondance** $\Rightarrow$ si on modifie la salle d'un cours il faut modifier $\forall$ étudiant inscrit au cours

- au mieux c'est coûteux ;
- au pire risque d'incohérence.

Conclusion: Multiplier une info c'est risquer qu'une des infos soit modifiée sans modifier les autres itérations.

En base de données, on considère que l'utilisateur ne connaît rien.

2.3.2. Autre schéma possible de BDD

Et($\text{numéro}_{\text{etudiant}}$, nom, prénom) cours(titre, heure, salle) **perte d'info : il manque Suit($\text{numéro}_{\text{étudiant}}$, titre)**

Normalisation relationnelle : montrer que ce résultat est le bon

Digression1 : algèbre relationnelle $\rightarrow$ dérivée de la théorie des ensembles (t est dans une relation R $\Leftrightarrow$ t $\in$ R)

Opérateurs unaires. Projection et sélection

Projection Π en $\Pi_{\#\text{etu, nom\_etu, pr\_etu}}$ Etudiant $\rightarrow$ Et

Sélection σ axiome de sélection de Ajermelo-Fränhel

Si $\mathbb{E}$ est un ens et φ une condition alors $\{x \in \mathbb{E} \mid \varphi(n)\}$ est un ensemble.

condition de sélection : $A\Theta B$ ou $A\Theta$ cte avec (A,B attributs; $\theta \in \{=, \neq, <, \leq, >, \geq\}$)

- Si C et C' sont des conditions de sélection

$\Rightarrow \neg C, C \vee C', C \wedge C', C' \in C, \ldots$ sont des conditions de sélections

liste des cours après le 1er avril à 14h : $\sigma_{\text{date}>\text{« 1er avril 14«}}$ Cours

Opérateurs ensemblistes : $\cup, \cap, -$ (réunion, intersection, différence)

Deux relations sont $\cup$-compatible ssi :

- elles sont de même arité ;
- les attributs correspondants sont de même domaine

On peut faire la réunion/intersection/différence que sur des relations U-compatibles.

✨ 🎀 **Citation inspirante** 🎀 ✨ **:**
Nolann *: Désolé mais je peux pas lire ce mot*

Pascal Ostermann *: Moi non plus*

🌷 🌺 *Nolann Martin & Pascal Ostermann* 🦋 *21 mars 2025* 🌺 🌷

ex : Cours suivi par aucun étudiant

$$\Pi_{\text{titre}}\text{Cours} - \Pi_{\text{titre}}\text{Suivit}$$

Produit cartésien, jointure :

$R \times S = \{(x_1,, ..., x_n, y_1, ..., y_n) \mid (x_1, ..., x_n) \in R, (y_1, ..., y_n) \in S\}$

R est n-aire, S m-aire

$$R \underset{C}{\bowtie} S = \sigma_C(R \times S)$$

C la condition de jointure

- equi-jointure = une jointure où la condition de jointure est une conjonction d'égalité
- jointure-« naturelle » = une equi-jointure sur les attributs de même nom.

ex: etu $\bowtie$ *Suit* $\bowtie$ *Cours* = *Etudiant* $\rightarrow$ Décomposition SPI

On dit qu'une décomposition $R \rightarrow S_1, ..., S_n$ est **sans perte d'information (SPI)** ssi $R \bowtie S_1 \bowtie ... \bowtie S_n$

$R \rightarrow S_1, ..., S_n$

L'algèbre peut se définir à l'aide de $\cup, -, \sigma, \Pi$ et $\times$

$(R \cap S = R - (R - S))$

L'opérateur coûteux de l'algèbre est la jointure ($\bowtie$).

$R \bowtie S$ de l'ordre de taille$(R) *$ taille(S)

avec un « index », sur les domaines concernées de S taille$(R) \times \log($taille$(S))$

Mais cela reste très coûteux.

$\Rightarrow$ Au final cela optimise les mise à jour et les requêtes.

✨ 🎀 **Citation inspirante** 🎀 ✨ **:**
Pour que mon q (cul) soit une abréviation de requête et pas autre chose

🌷 🌺 *21 Mars 2025* 🦋 *Pascal Ostermann* 🌺 🌷

Digression 2 : Dépendances Fonctionnelles (DF)

X et Y un ensemble d'attributs d'une relation R $X \rightarrow Y$ est vraie dans la relation R ssi « X détermine Y »:

$$\forall s, t \in R \qquad s \cdot X = t \cdot X \Rightarrow s \cdot Y = t \cdot Y$$

$s \cdot X \Leftrightarrow$ « projection de s sur X«

Ex:

Si $Y \subset X$ $X \rightarrow Y$ DF

Si X $\rightarrow$ Y et Y $\rightarrow$ Z alors X $\rightarrow$ Z

X $\rightarrow$ Y et Z $\rightarrow$ Z <==>

$$X$$

->YZ #-etu $\rightarrow$ $\text{nom}_{\text{etu}}, \text{prenom}_{\text{etu}}$

$\text{salle}_{\text{cours}}, \text{heure}_{\text{cours}} \rightarrow \text{titre}_{\text{cours}}$

une DF est réduite à gauche $X \rightarrow Y$ ssi $[Z \subset Y$ et $Z \rightarrow Y \Rightarrow Z = X]$

K est une **superclef** de R ssi $K \to R$(abus de langage pour l'ensemble des attributs de R)

Une clef = une superclef minimale pour l'inclusion.

Théorème 1 :

Tout relation a une superclef

Théorème 2 :

Toute relation a au moins une clef

A partir de maintenant, je veux qu'on souligne une clef ⇒ c'est la clef primaire la clef d'accès à la table de hachage

✨ 🎀 **Citation inspirante** 🎀 ✨ **:**
à savoir que j'écris clef avec un « f » et pas avec un accent sinon on ne peut pas lire

🌷 🌺 *Pas cal Ostermann (aka l'Homme de l'Est)* 🦋 *21 mars 2025* 🌺 🌷

Théorème 3 : Théorème de décomposition

$X \to Y$ (X, Y disjoints) et DF vraie dans R

alors R se décompose SPI en $\Pi_X(R)$ et $\Pi_{C_y}(R)$

Méthode 1 : décomposition

utiliser le théorème de décomposition jusqu'à obtenir des formes normales

Forme Normal de Boyce-Codd (FNBC)

R est au FNBC ssi toute DF $X \to Y$

V dans R est telle que $\begin{cases} Y \subset X \text{(DF triviale)} \\ \text{ou} \\ X \text{ est superclef} \end{cases}$

Théorème 4 :

Toute relation peut se décomposer SPI en des relations en FNBC

✨ 🎀 **Citation inspirante** 🎀 ✨ **:**
J'aurais jamais pensé apprendre les bdd avec un ancien hippie

🌷 🌺 *SORRES Antonin* 🦋 *28 mars 2025* 🌺 🌷

Exemple de normalisation: programmation multiplexe cinéma

Liste des attributs:

- nom_salle
- capacité_salle
- titre_film
- date_séance (format inclus l'heure)

Relation « globale » : R(nom_salle, capacité_salle, titre_film,date_séance)

DFs:
- nom_salle → capacité_salle (1)
- nom_salle & date → titre_film (2)
- titre_film → nom_salle (On dit qu'un film ne passe que dans une salle) (3)

R n'est pas un FNBC à cause de (1)
1. On applique le théorème de décomposition → R se décompose SPI :
   - Salle(salle,capacité)
     - Et un reste

   N'est toujours pas FNBC → (3)+ théorème de décomposition

   Programmation(titre_film,nom_salle)
   - Et un reste (titre_film, date_seance)

Pour avoir l'information complète: « film X passe à heure Y dans salle Z » → il faut faire une jointure (coûteux).

On se demande donc si c'est pas mieux de réster sur l'étape d'avant :
- Salle
- Programmation2(nom_salle, titre_film, date_séance) ⇒ pas en FNBC mais en 3e FN et c'est une décomposition sans perte de dépendance

Troisième Forme Normale (3FN)

R est en 3e FN ⇔ toute DF $X \to A$ (A attribut) … des R est tel que

- X superclef
- A ∈ X
- A fait partie d'une clef

Donc: FNBC ⇒ 3e FN Toute relation peut se décomposer en SPI et sans perte de dépendance en des relations en 3e FN

Définition 1: Formes normales

1. 1ère Forme Normale (obsolète): à l'origine, dans une « case » de tableau, il pouvait y avoir des valeurs non-atomiques. La 1ère FN disait que ces valeurs étaient atomiques.

   Relationnel Non Formal?? Normal Form (NFNF ou NF2)
   - dans une « case » du tableau il peut y avoir une relation
   - variante de l'E…???
2. 2e Forme Normale :

   Si une DF $K \to Y, K$ est une clé viole des codes de la $3^{\text{ème}}$ FN
   - Deuxième cas $Z \leq K$, $Z \to T$, $T < Y$:il y a une DF potentielle ou non élémentaire.

- Ou alors; $K \to Y$ et $Y \to Z \Rightarrow$ Il y a une DF teaudtine??? ou non directe.

Lorsque nous établissons une liste des DFs, il faut l'éllaborer de la manière la plus concise.

En particulier pas de DF partielle

suite transitive :
- si on ecrit $X \to Y$ et $Y \to Z$
- sinon si on ecrit $X \to Z$

3. 4e Forme Normale: R est en 4e Forme Normale (4NF) ssi toute DMV $X \to Y|Z(X, Y, Z$ partitions de R)
   - et .... Y = 0 ou Z = 0 (DMV triviale)
   - ou X super clef (X $\to$ YZ)
   - On peut toujours décomposer SPI en des relation de 4FN

✨ 🎀 **Citation inspirante** 🎀 ✨ **:**
Quand on va dans des université américaines, tant qu'il en reste

🌷 🌺 *Pascal Ostermann* 🦋 *28 mars 2025* 🌺 🌷

Exemple: Etudiant(num_etu, titre_cours, sport)

Pas de DF, donc en FNBC:

#et	cours	sport
813	Histoire	Savate
813	Géo	Escrime
813	Géo	Savate
813	Histoire	Escrime

on tire des deux premières les deux autres car il faut toutes les associations possibles (je crois)

Il faudrait décomposer en: X(num_etu, cours), et Y(num_etu, sport)

Dépendances Multi-Valuées (DMVs)

soient X, Y, Z disjoints 2 à 2

$X \to Y|Z$ vraie dans R

$$\Leftrightarrow \forall s,t \in \mathbb{R} \quad s \cdot X = t \cdot X \Rightarrow$$
$$\exists u \;\; X = s \cdot X \;\; \text{si } Y = s \cdot Y$$
$$\text{si } Z = t \cdot Z$$

Théorème 5 : Décomposition pour les DMVs

Si X, Y, Z partition de R

$X \to Y|Z \Leftrightarrow$ R se décompose en SPI en $\Pi_{XY}(R)$ et $\Pi_{XZ}(R)$

$X \to Y \mid Z \Leftrightarrow Z \mid Y$

$X \to Y \mid \emptyset$ est vraie

X, Y disjoints alors $X \to Y \to X \to Y \mid Z$ $\quad \forall Z$ disjoint de X et de R

4 FN = FNBC

Exemple: buveurs de bièèèèère

✨ 🎀 **Citation inspirante** 🎀 ✨ **:**
Buveurs(buveur, bière, bar) une relation globale, « globalement nulle à chier »

🌷 🌺 *hippie* 🦋 *28 mars 2025* 🌺 🌷

Comment faire une meilleure relation buveurs -> bière | bar ?

- la relation est multivaluée ssi Buveurs = Boits(buveurs, bière) ⋈ Fréquente(Buveur, bar)
  - Ici perte de l'info Sert
- bière → buveurs | bar ssi Buveurs = Boit(buveurs, bière) ⋈ Sert(bar, bière)
  - Ici perte de l'info Fréquence

La seule décomposition possible est:
- Buveur = Boit ⋈ Frequente ⋈ Sert

On dit qu'il y a une Dépendance de Jointure (DJ) et qu'il y a 5e Forme Normale ssi les DJ se déduisent des clefs

Autre normalisation pour la relation **Buveur**. On prends comme attributs:
- Buveur2(Buveur, bbière_bue, bar, bière_servie)
  - buveur → bière_bue | bar, bière_servie
  - bar → buveur | bière_servie

  ⇒ Fréquente, sert, boit.

2.3.3. Méthode de normalisation
A réviser sinon hippie pas content

1. Etablir une liste des attributs (indépendants les uns des autres)
   - **Pas d'attributs calculables.** (ex: nombre de cours) => à proscrire
   - Que des attributs atomiques (pas de listes)
   - Éviter les boucles de signification
2. Lister les Dépendances Fonctionnelles et Multi-Valuées (DF, DMVs)
   - Pas de dépendances fonctionnelles transitives
   - pas de dépendances partielles
   - toutes les DFs réduites à gauche
3. Décomposition la plus puissante possible

2.4. Résumé Normalisation
- DFs + theo de décomposition pour les DFs
  - 1FN : totalement optionnel (on s'en fous)
  - 2FN : optionnel
  - 3FN « dénormalisation »
  - FBNC (*la plus importante*) + thm de décomposition pour DMV

- 4NF:
- 5FN : à éviter

3. Languages relationnels

- Language de Définition des Données (LDD)
  - défini des relations, des cléfs, primaires et secondaires, des indexes, des attributs
- Language de Manipulation des Données (LMD)
  - mises à jour (MAJ)
  - requêtes (q̂)
    - écriture de Algèbre relationnel
    - fonctions d'agrégation résultat « statistique »

4. SQL (Structured Query Language)

C'est un language déclaratif avec optimisation implicite

4.1. Algèbre relationnelle en SQL

c'est la complétude au sense de Codd

Exemple du TD1:

- Symbole(GPS, valeur)
- Lieu(GPS, nom, tel, GPS_Sym)
- Ligne(GPS, nature, nom, desc)
- Rencontre(GPS_ligne, GPS_sym, num d'ordre)

lire de la manière:

forme en algèbre relationelle $\iff$ *requete sql*

- Pour récupérer des infos:

$\Pi_{\text{nom, tel}}$ Lieu $\iff$ SELECT nom, tel FROM Lieu;

- Pour selectionner certaines infos particulières

$\sigma_{\text{nature = 'cours d'eau}}$ Ligne $\iff$ SELECT * FROM Ligne WHERE nature = 'cours d eau'

- $\cap, \cup, -$: UNION, INTERSECTS, MINUS

Symbole qui ne corresponds à aucun Lieu. En algèbre relationelle : Π_{GPS} Symbole $-$ Π_{GPS} Lieu

En SQL:
SELECT GPS FROM Symbole
MINUS
SELECT GPS_symbole FROM Lieu;

- Produit cartésien de 2 relations R et S:

SELECT <Des attributs> FROM R,S;

Attention sans "WHERE" cela donne le produit carthésien

- Liste Des « Hébergements »:

$\Pi_{\text{Lieu, GPS, tél}}(\sigma_{\text{nature = hébergement}}) \bowtie$ Lieu

SELECT Lieu, GPS, tél FROM Symbole, Lieu WHERE nature = 'hébergement' AND Symbole.GPS = Lieu.GPS_Symbole

SQL est complet au sens de Codd (ie: permets d'exprimer l'algebre relationnel)

- Autre écriture de la jointure($\bowtie$) précédente:

  SELECT GPS, tél FROM Lieu

  WHERE GPS IN

  (SELECT GPS_Symbole WHERE nature = 'hébergement');

Attention parenthèses obligatoires: sinon on sait pas de quel « WHERE » on parle.

- **IN** est un prédicat du 2nd ordre, usage : A IN (SELECT …) est vrai ssi A est un élément de (SELECT …)
- « Grande » jointure utile si en résultat, on veut des attributs vennant des 2 relations.

SELECT utile surtout pour des MAJs.

par exemple:

UPDATE Lieu WHERE GPS_Symbole IN

(SELECT GPS FROM Symbole WHERE nature = 'hébergement')

SET desc = 'hébergement' $\wedge$ desc;

$\wedge$: Concaténation de chaînes de caractères

- Tout les prédicats du 2nd ordre: NOT IN, CONTAINS ...
  - Cas du **NOT IN**: tout les Symboles qui ne correspondent à aucun Lieu

SELECT GPS, nature FROM Symbole

WHERE GPS, NOT IN (SELECT GPS_Symbole FROM Lieu)

(S1) CONTAINS (S2) ssi S2 $\subset$ S1 (inclusion ensembliste)

/!\ S1 et S2 sont totalement indépendants de l'extérieur. Il faut écrire EXPLICITEMENT les conditions de jointure

4.2. Fonctions d'agrégation

Calculer des sommes (**COUNT, AVG, ...**)

- Nombre de symboles sur la carte:

SELECT COUNT(*) FROM Lieu;

COUNT(*) => compte les lignes

COUNT(nom) => compte les lignes où nom est NOT NULL

COUNT(tel) => compte les lignes où tel n'est pas vide

COUNT(DISTINCT tel) => compte les lignes où les tel sont distincts (et non NULL)

4.2.1. Digression sur les valeurs nulles

Valeur nulle NULL → valeur nulle de Codd (non-existentielle : la valeur n'est pas définie)

$$\longrightarrow \begin{cases} \text{NULL} =4\rightarrow \text{NULL} \\ \text{NULL} -7\rightarrow \text{NULL} \\ \text{NULL} = \text{NULL} \rightarrow \text{NULL} \\ \text{NOT NULL} \rightarrow \text{NULL} \end{cases}$$

SELECT * FROM Lieux WHERE tel IS NULL;

→ reponse vide

pour obtenir les lieu dont le tel n'est pas NULL : operateur IS NOT

Dans une proposition (en logique), est interprété comme Faux

Tous les cours d'eau avec leur nb de points, mais seulement pour ceux qui ont + 10 points :

SELECT Ligne, GPS, nom, COUNT(DISTINCT num_ordre)

FROM Ligne, Rencontre

WHERE nature = 'cours d eau'

AND Ligne.GPS = Rencontre GPS.ligne

GROUP BY Ligne.GPS, nom

HAVING COUNT(DISTINCT num_ordre) > 10;

Autres fonctions d'agrégation SUM, AVG, MIN, MAX, VARIANCE, ...

Listes des points du GR32, dans l'ordre :

SELECT Rencontre.GPS.Point FROM Ligne, Rencontre WHERE Ligne.GPS = Rencontre.GPS.ligne ORDER BY num_ordre;

Note: jsp ce qu'il cook mais il a repris la Query d'avant avec des modifs pour en faire une vue:

CREATE VIEW Statistiques AS

SELECT Ligne, GPS, nom, COUNT(DISTINCT num_ordre) nombre de points

FROM Ligne, Rencontre

WHERE nature = 'cours d eau'

AND Ligne.GPS = Rencontre GPS.ligne

GROUP BY Ligne.GPS, nom

HAVING COUNT(DISTINCT num_ordre) > 10;

en brun, operateur de renommage

Attention, il est hors de question d'utiliser un ORDER BY sur une vue

vue: représentation d'un calcul sous forme de relation, la vue statistique est recalculée chaque fois qu'on l'utilise.

Définition 2: MERISE

Méthode d'Etude et de Représentation Informatique des Systèmes d'Entreprise

système d'information -> la mémoire de l'entreprise = données + traitements (tts)

Principe de MERISE : étude séparée des donnes des tts

Données (E-A)	Traitements
Modèle conceptuel des données brut ↓ validation → Modèle validé ↑ Modèles externes	Modèle conceptuel des traitements ↓ ←Phases { manuelle / automatique / converssationnelle (homme/machine)

Rapport (pas sûr) : Phases / MXs

On peut dire quelles données sont utiles à tel traitement.

(∗ 1)

traitements données	R1	R2	...	Rn
traitement 1	X	s	...	s/w
traitement 2	w	x	...	x
...				
traitement m				

⇒ je peux savoir combien de fois sont appliquées { telle jointure / ou / telle màj

→ fichiers : représenté par une table de hachage

problème :

- optimiser les màj ⇒ rend difficile la moindre requête.
- optimiser les requêtes ⇒ dégrader les màj.
- optimiser les requêtes c'est rajouter une donnée là où il y en a déjà une, il y a collision -> deux données identiques pour la fonction de hachage. En cas de collision:
  - hachage linéaire : on met alors le donnée à la case suivante .
  - hachage quadratique

  -on applique une seconde fonction de hachage

Complexité d'un ajout de données = nombre moyen de collisions → nettoyer les données inutiles rend plus efficace les recherches.

Compléxité d'une jointure entre deux relations R et S

o(tailleR $*$ taille S) sans index (outil d'optimisation)

o(taille R $*$ nombreDecollisionsDeS) avec index

problème de l'index :

- quand on ajoute une donnée il faut aussi mettre à jour l'index.

Les choix d'optimisation :

- mettre un index ou pas
- forme normale plus ou moins forte peuvent se déduire du tableau ($*$ 1)

Les modèles externes se déduisent assez facilement.

à l'examen :

- pour SQL il est possible qu'il y ai 2 ou 3 questions dessus très simple.
- @ mail du prof sur le TD2 pour les questions