Conception de bases de données relationnelles

Langage SQL

Marie Szafranski

ensiie

SQL (Structured Query Language)

Langage déclaratif pour la *manipulation* des SGBD(R)

→ interface standard pour accéder aux BD

- *Langage de Définition de Données* **[LDD / DDL]**
  Création, *modification*, *suppression* d'objets → tables, vues, etc.
- *Langage de Contrôle de Données* **[LCD / DCL]**
  Gestion des droits sur les objets
  → création des utilisateurs, droits et rôles, etc.
- *Langage de Manipulation de Données* **[LMD / DML]**
  Recherche, *insertion*, *mise à jour* et *suppression* de données

SQL (Structured Query Language)

Origine du SQL

Fondé sur le *modèle relationnel* de Codd et l'*algèbre relationnelle* associée
[Codd, 70, IBM]

Les prédécesseurs de SQL

- IBM Sequel [Boyce & Chamberlin, 77, IBM]
  Structured English QUEry Language
- → IBM DB2

La naissance du standard SQL

- SQL [Relationnal Software (Oracle), 79]
- SQL-86 [ANSI (États Unis), 86 ; ISO, 87]

Origine du SQL

Évolution du SQL

- SQL-86 → version originale
- SQL-89 (ou SQL-1) → améliorations mineures
- SQL-92 (ou SQL-2) → extensions majeures
  (types de données, opérations relationnelles, LDD, transactions, etc.)
- SQL-99 (ou SQL-3) → introduction du PSM (procédures stockées)
  → introduction du RO
- SQL-2003 → extensions XML
- SQL-2006 → améliorations mineures (XML)
- SQL-2008 → améliorations mineures (RO)
- SQL-2011 → améliorations mineures (XML, RO, définition des schémas)

Évolution du SQL

Super exemple : fil d'Ariane

Super_heros (SH)

#nom_super	nom_civil	prenom	age
M. Indestructible	Parr	Robert	36
Elastic girl	Parr	Helene	36
Violette	Parr	Violette	16
Fleche	Parr	Roger	12
Jack-Jack	Parr	Jack	2
Frozone	Best	Lucius	36

Super_pouvoirs (SP)

#nom_pouvoir
Elasticite
Champ de glace
Champ de force
Invisibilite
Rapidite
Resistance

Super_heros_pouvoirs (SHP)

#heros=>SH	#pouvoir=>SP
Indestructible	Resistance
Elastic girl	Elasticite
Frozone	Champ de glace
Violette	Invisibilite
Violette	Champ de force
Fleche	Rapidite

Super exemple : fil d'ariane

Sélection (Syntaxe)

(*Produit cartésien* + *restriction*) ou *jointure* + *projection*

Expression basique d'une requête avec SQL

```
SELECT <liste d'attributs projetés>
FROM <liste de relations>
WHERE <condition>;
```

- SELECT : sous-ensemble des attributs *projetés*
  → *schéma* de la requête
- FROM : relations des attributs utilisés
  → en *projection*, en *restriction* ou en *jointure*
- WHERE : conditions sur les attributs
  → *restriction* ou *jointure* si exprimée ici

Sélection (Syntaxe)

Sélection (Exemple)

SELECT

Super_heros

nom_super	nom_civil	prenom	age
M. Indestructible	Parr	Robert	36
Elastic girl	Parr	Helene	36
Violette	Parr	Violette	16
Fleche	Parr	Roger	12
Jack-Jack	Parr	Jack	2
Frozone	Best	Lucius	36

```
SELECT nom_super, age
FROM Super_heros
WHERE nom_civil = 'Parr';
```


Sélection (Exemple)

SELECT

Super_heros

nom_super	nom_civil	prenom	age
M. Indestructible	Parr	Robert	36
Elastic girl	Parr	Helene	36
Violette	Parr	Violette	16
Fleche	Parr	Roger	12
Jack-Jack	Parr	Jack	2
Frozone	Best	Lucius	36

```
SELECT nom_super, age                                   projection
FROM Super_heros
WHERE nom_civil = 'Parr';
```


Sélection (Exemple)

SELECT

Super_heros

nom_super	nom_civil	prenom	age
M. Indestructible	Parr	Robert	36
Elastic girl	Parr	Helene	36
Violette	Parr	Violette	16
Fleche	Parr	Roger	12
Jack-Jack	Parr	Jack	2

```
SELECT Super_heros.nom_super                            projection
FROM Super_heros
WHERE nom_civil = 'Parr';                               restriction
```


Sélection (Exemple)

Sélection (Exemple)

SELECT DISTINCT

Super_heros

nom_super	nom_civil	prenom	age
M. Indestructible	Parr	Robert	36
Elastic girl	Parr	Helene	36
Violette	Parr	Violette	16
Fleche	Parr	Roger	12
Jack-Jack	Parr	Jack	2
Frozone	Best	Lucius	36

```
SELECT nom_civil, age                    projection
FROM Super_heros
WHERE age = 36;                          restriction
```


Sélection (Exemple)

SELECT DISTINCT

Super_heros

nom_super	nom_civil	prenom	age
M. Indestructible	Parr	Robert	36
Elastic girl	Parr	Helene	36
Frozone	Best	Lucius	36

```
SELECT nom_civil, age                    projection
FROM Super_heros
WHERE age = 36;                          restriction
```


Sélection (Exemple)

SELECT DISTINCT

Super_heros

nom_super	nom_civil	prenom	age
	Parr		36
	Best		36

```
SELECT DISTINCT nom_civil, age           projection
FROM Super_heros
WHERE age = 36;                          restriction
```


Sélection (Exemple)

Sélection (Exemple)

SELECT AS

Super_heros

nom_super	nom_civil	prenom	age
M. Indestructible	Parr	Robert	36
Elastic girl	Parr	Helene	36
Violette	Parr	Violette	16
Fleche	Parr	Roger	12
Jack-Jack	Parr	Jack	2
Frozone	Best	Lucius	36

```
SELECT <nom_attribut> AS <nouveau_nom_attribut>
FROM <liste de relations>;
```


Sélection (Exemple)

SELECT AS

Super_heros

nom_super	nom_civil	prenom	age
M. Indestructible	Parr	Robert	36
Elastic girl	Parr	Helene	36
Frozone	Best	Lucius	36

```
SELECT <nom_attribut> AS <nouveau_nom_attribut>
FROM <liste de relations>;

SELECT nom_super, age
FROM Super_heros WHERE age = 36;
```


Sélection (Exemple)

SELECT AS

Super_heros

Liste de Supers	nom_civil	prenom	age
M. Indestructible	Parr	Robert	36
Elastic girl	Parr	Helene	36
Frozone	Best	Lucius	36

```
SELECT <nom_attribut> AS <nouveau_nom_attribut>
FROM <liste de relations>;

SELECT nom_super AS 'Liste de Supers', age
FROM Super_heros WHERE age = 36;
```


Sélection (Exemple)

Sélection (Synthèse)

- Projeter l'ensemble des attributs
  → SELECT *
- Éliminer les doublons
  → SELECT DISTINCT
- Renommer les attributs
  → SELECT a AS a'
- Préfixer un attribut par une relation *en cas d'ambiguïté*
  → SELECT Relation.attribut

Sélection (synthèse)

Opérateurs de comparaison et opérateurs logiques

Restriction : condition exprimée dans la clause WHERE (résultat booléen)

- Opérateurs de comparaison
  A = C, A <> C, A < C, A > C, A <= C, A >= C
  A BETWEEN C1 AND C2
  A LIKE 'chaine' → joker : %, _
  A IN (C1, C2, ...)
  A IS NULL
- Opérateurs logiques
  OR, AND, NOT

```
condition élémentaire := attribut <opérateur de comparaison> constante
condition := condition <opérateur logique> condition | condition élém.
```


Opérateurs de comparaison et opérateurs logiques

Expression du produit cartésien (Syntaxe)

Produit cartésien : opération binaire R1 × R2

Schéma : juxtaposition des schémas de R1 et R2

Résultat : combinaison des tuples de R1 et R2

Algèbre relationnelle

```
R3 = Produit(R1, R2)                                -- = Produit(R2, R1)
                        -- (modulo l'ordre des attributs dans le schéma)
```

SQL

```
SELECT *
FROM R1, R2;
```


Expression du produit cartésien (Syntaxe)

Expression du produit cartésien (Exemple)

Super_heros (SH)

nom_super	nom_civil	prenom	age
M. Indestructible	Parr	Robert	36
Elastic girl	Parr	Helene	36
Violette	Parr	Violette	16
Fleche	Parr	Roger	12
Jack-Jack	Parr	Jack	2
Frozone	Best	Lucius	36

Super_pouvoirs (SP)

#nom_pouvoir
Elasticite
Champ de glace
Champ de force
Invisibilite
Rapidite
Resistance

- Schéma SH × SP ?
  (nom_super, nom_civil, prenom, age, nom_pouvoir)
- Nombre de tuples ?
  6 × 6
- Expression ?

```
SELECT *
FROM SH, SP;
```


Expression du produit cartésien (Exemple)

Expression d'une jointure (Syntaxe)

Jointure : opération binaire sur R1 et R2
| condition C sur attributs de R1 et R2 de mêmes domaines

Schéma : juxtaposition de R1 et R2

Résultat : concaténation des tuples de R1 et de R2 vérifiant C

Algèbre relationnelle

```
Jointure(R1, R2, R1.X <operateur de comparaison> R2.Y)
```

SQL

```
SELECT *                                                    -- SQL-92
FROM R1 [INNER] JOIN R2 ON R1.X <op. comparaison> R2.Y;

SELECT *                                                    -- SQL-89
FROM R1, R2
WHERE R1.X <op. comparaison> R2.Y;  restriction sur un produit cartésien
```


Expression d'une jointure (Syntaxe)

Expression d'une jointure (Exemple)

Super_heros (SH)

age	#nom_super
36	Indestructible
36	Elastic girl
36	Frozone
16	Violette
12	Fleche
2	Jack-Jack

Super_heros_pouvoirs (SHP)

#heros=>SH	#pouvoir=>SP
Indestructible	Resistance
Elastic girl	Elasticite
Frozone	Champ de glace
Violette	Invisibilite
Violette	Champ de force
Fleche	Rapidite

```
SELECT nom_super, pouvoir, age
FROM Super_heros INNER JOIN Super_heros_pouvoirs
ON nom_super = heros;
```


Expression d'une jointure (Exemple)

Super_heros (SH)

age	#nom_super
36	Indestructible
36	Elastic girl
36	Frozone
16	Violette
12	Fleche
2	Jack-Jack

Super_heros_pouvoirs (SHP)

#heros=>SH	#pouvoir=>SP
Indestructible	Resistance
Elastic girl	Elasticite
Frozone	Champ de glace
Violette	Invisibilite
Violette	Champ de force
Fleche	Rapidite

nom_super	pouvoir	age
Indestructible	Resistance	36
Elastic girl	Elasticite	36
Frozone	Champ de glace	36
Violette	Invisibilite	16
Violette	Champ de force	16
Fleche	Rapidite	12

Expression d'une jointure (Exemple)

Super_heros (SH)

age	#nom_super
36	Indestructible
36	Elastic girl
36	Frozone
16	Violette
12	Fleche
2	Jack-Jack

Super_heros_pouvoirs (SHP)

#heros=>SH	#pouvoir=>SP
Indestructible	Resistance
Elastic girl	Elasticite
Frozone	Champ de glace
Violette	Invisibilite
Violette	Champ de force
Fleche	Rapidite

```
SELECT heros, pouvoir, age
FROM Super_heros INNER JOIN Super_heros_pouvoirs
ON nom_super = heros;
```


Expression d'une jointure (Exemple)

Super_heros (SH)

age	#nom_super
36	Indestructible
36	Elastic girl
36	Frozone
16	Violette
12	Fleche
2	Jack-Jack

Super_heros_pouvoirs (SHP)

#heros=>SH	#pouvoir=>SP
Indestructible	Resistance
Elastic girl	Elasticite
Frozone	Champ de glace
Violette	Invisibilite
Violette	Champ de force
Fleche	Rapidite

heros	pouvoir	age
Indestructible	Resistance	36
Elastic girl	Elasticite	36
Frozone	Champ de glace	36
Violette	Invisibilite	16
Violette	Champ de force	16
Fleche	Rapidite	12

Expression d'une jointure (Exemple)

Expression d'une jointure (Exemple)

Expression d'une jointure (Compléments)

Jointure naturelle SQL

Jointure portant sur des attributs de mêmes noms

```
Super_heros(nom_super, nom_civil, prenom_civil, age)
Super_heros_pouvoirs(nom_super, pouvoirs)


                                                        -- SQL-92
                    (nom_super, nom_civil, prenom_civil, age, pouvoirs)
SELECT *
FROM Super_heros NATURAL JOIN Super_heros_pouvoirs ;
```


Expression d'une jointure (Compléments)

Jointure naturelle SQL

Jointure portant sur des attributs de mêmes noms

```
Super_heros(nom_super, nom_civil, prenom_civil, age)
Super_heros_pouvoirs(nom_super, pouvoirs)


                                                        -- SQL-92
                    (nom_super, nom_civil, prenom_civil, age, pouvoirs)
SELECT *
FROM Super_heros NATURAL JOIN Super_heros_pouvoirs ;


                                                        -- SQL-89
          (nom_civil, prenom_civil, age, nom_super, nom_super, pouvoirs)
SELECT *
FROM Super_heros SH, Super_heros_pouvoirs SHP
WHERE SH.nom_super = SHP.nom_super;
```


Expression d'une jointure (Compléments)

Jointure naturelle SQL

Jointure portant sur des attributs de mêmes noms

```
Super_heros(nom_super, nom_civil, prenom_civil, age)
Super_heros_pouvoirs(nom_super, pouvoirs)


                                                        -- SQL-92

SELECT *
FROM Super_heros [INNER] JOIN Super_heros_pouvoirs USING (...);
                      Restriction quand plusieurs attributs de même nom
```


Expression d'une jointure (Compléments)

Expression d'une jointure (Compléments)

Auto jointure SQL

Jointure portant sur la même relation

```
SELECT *
FROM R1 <alias1>, R1 <alias2>
WHERE <alias1>.attribut = <alias2>.attribut;


                        -- Tous les Supers appartenant à une même famille
SELECT nom_civil, prenom
FROM Super_heros SH1, Super_heros SH2
WHERE SH1.nom_civil = SH2.nom_civil;
```


Expression d'une jointure (Compléments)

Expression d'une jointure externe (Syntaxe)

Jointure externe : opération binaire sur R1 et R2

Schéma : juxtaposition de R1 et R2

Résultat : tuples de la *jointure* de R1 et R2 + tuples de R1 (ou R2) exclus

Algèbre relationnelle

```
JointureExterne(R1, R2, R1.X <op. comparaison> R2.Y)
```

SQL

```
SELECT *
FROM R1 OUTER JOIN R2 ON R1.X <op. comparaison> R2.Y;

SELECT *                    -- JointureExterneGauche(R1, R2, R1.C = R2.X)
FROM R1 LEFT OUTER JOIN R2 ON R1.X <op. comparaison> R2.Y;

SELECT *                    -- JointureExterneDroite(R2, R1, R1.C = R2.X)
FROM R2 RIGHT OUTER JOIN R1 ON R1.X <op. comparaison> R2.Y;
```


Expression d'une jointure externe (Syntaxe)

Expression d'une jointure externe (Exemple)

Super_heros (SH)

age	#nom_super
36	Indestructible
36	Elastic girl
36	Frozone
16	Violette
12	Fleche
2	Jack-Jack

Super_heros_pouvoirs (SHP)

#heros=>SH	#pouvoir=>SP
Indestructible	Resistance
Elastic girl	Elasticite
Frozone	Champ de glace
Violette	Invisibilite
Violette	Champ de force
Fleche	Rapidite

```
SELECT nom_super, pouvoir, age
FROM Super_heros OUTER JOIN Super_heros_pouvoirs
ON nom_super = heros;
```


Expression d'une jointure externe (Exemple)

Super_heros (SH)

age	#nom_super
36	Indestructible
36	Elastic girl
36	Frozone
16	Violette
12	Fleche
2	Jack-Jack

Super_heros_pouvoirs (SHP)

#heros=>SH	#pouvoir=>SP
Indestructible	Resistance
Elastic girl	Elasticite
Frozone	Champ de glace
Violette	Invisibilite
Violette	Champ de force
Fleche	Rapidite

nom_super	age	pouvoir
Indestructible	36	Resistance
Elastic girl	36	Elasticite
Frozone	36	Champ de glace
Violette	16	Invisibilite
Violette	16	Champ de force
Fleche	12	Rapidite
Jack-Jack	2	NULL

Expression d'une jointure externe (Exemple)

Super_heros (SH)

age	#nom_super
36	Indestructible
36	Elastic girl
36	Frozone
16	Violette
12	Fleche
2	Jack-Jack

Super_heros_pouvoirs (SHP)

#heros=>SH	#pouvoir=>SP
Indestructible	Resistance
Elastic girl	Elasticite
Frozone	Champ de glace
Violette	Invisibilite
Violette	Champ de force
Fleche	Rapidite

```
SELECT nom_super, pouvoir, age
FROM Super_heros OUTER JOIN Super_heros_pouvoirs
ON nom_super = heros;
=
SELECT nom_super, pouvoir, age
FROM Super_heros LEFT OUTER JOIN Super_heros_pouvoirs
ON nom_super = heros;
```


Expression d'une jointure externe (Exemple)

Expression d'une jointure externe (Exemple complet)

R1

#A	B	C=>R2
1	Alpha	10
2	Bravo	10
3	Charlie	20
4	Delta	

R2

#X	Y
10	Echo
20	Fox
30	Golf

Expression d'une jointure externe (Exemple complet)

Expression d'une jointure externe (Synthèse)

- Équivalence entre `R1 LEFT OUTER JOIN R2`
  et `R2 RIGHT OUTER JOIN R1`
- `LEFT OUTER JOIN` : relation issue d'une asso. 0..1 : (0..)n
  → sélection de tous les tuples du côté 1 (ayant une référence ou pas)
- `RIGHT OUTER JOIN` : relation issue d'une asso. (0..)1 : 0..n
  → sélection de tous les tuples du côté n (référencés ou pas)

→ Utile pour trouver les tuples non référencés par une clé étrangère

Expression d'une jointure externe (Synthèse)

Opérateurs ensemblistes

Union

SQL

```
SELECT * FROM R1
UNION
SELECT * FROM R2;
```

Intersection

SQL

```
SELECT * FROM R1
INTERSECT
SELECT * FROM R2;
```

Différence

SQL

```
SELECT * FROM R1
EXCEPT
SELECT * FROM R2;
```


Opérateurs ensemblistes

Tris

Résultat : en fonction des valeurs des attributs retournés

SQL

```
SELECT <liste d'attributs>
FROM <liste de relations>
WHERE <conditions>
ORDER BY <liste de (noms | numéro dans la projection) d'attributs à
ordonner>[ASC, DESC];

SELECT nom_super, prenom, age
FROM Super_heros
WHERE age >= 16
ORDER BY nom_super, age1, 3 DESC;
```


Tris

Fonctions de calcul

Fonction de calcul : dans une relation
sur l'ensemble de valeurs d'un attribut

Résultat : une valeur atomique unique (entier, chaine, date, etc.)

- `count(relation, attribut)`
  → nombre de valeurs non nulles d'un attribut pour tous les tuples
- `sum(relation, attribut)` et `avg(relation, attribut)`
  → somme (resp.moyenne) des valeurs d'un attribut numérique
- `min(relation, attribut)` et `max(relation, attribut)`
  → plus petite (resp. grande) valeur d'un attribut parmi les tuples

Fonctions de calcul

Fonctions de calcul (Syntaxe et exemple)

SQL

```
SELECT <liste de fonctions de calcul>
FROM <liste de relations>
WHERE <conditions à appliquer avant le calcul>;

SELECT min(age), max(age), avg(age)
FROM Super_heros
WHERE nom_civil = 'Parr';
```

Remarques

- Nombre de tuples d'une relation → `count` sur une *clé primaire*
- Fonction de calcul d'un *agrégat* → clause `ORDER BY` ou `HAVING`
  jamais dans une clause `WHERE`

Fonctions de calcul (Syntaxe et exemple)

Agrégat

Agrégat : partitionnement horizontal d'une relation en sous-relations
| un ou plusieurs attributs de partitionnement
+ *fonction de calcul* sur chaque attribut des sous-relations

SQL

```
SELECT <liste d'attributs de partitionnement et de fonctions de calcul>
FROM <liste de relations>
WHERE <conditions à appliquer avant le calcul de l'agrégat>
GROUP BY <liste ordonnée d'attributs de partitionnement>
HAVING <condition sur les fonctions de calcul>;

SELECT pouvoir, avg(age)
FROM Super_heros INNER JOIN Super_heros_pouvoir ON nom_super = heros
GROUP BY pouvoir
HAVING avg(age) > 10;
```


Agrégat

Agrégat : partitionnement horizontal d'une relation en sous-relations
| un ou plusieurs attributs de partitionnement
+ *fonction de calcul* sur chaque attribut des sous-relations

SQL

```
SELECT <liste d'attributs de partitionnement et de fonctions de calcul>
FROM <liste de relations>
WHERE <conditions à appliquer avant le calcul de l'agrégat>
GROUP BY <liste ordonnée d'attributs de partitionnement>
HAVING <condition sur les fonctions de calcul>;

SELECT Societe.nom, avg(Personne.age)
FROM Societe INNER JOIN Personne ON Personne.societe = Societe.nom
WHERE Societe.nom LIKE 'A%'
GROUP BY Societe.nom
HAVING count(num_ss) >10;
```


Agrégat (Remarques)

- *Restriction* : avant le calcul de l'*agrégat* (clause `WHERE`)
  après le calcul de l'*agrégat* (clause `HAVING`)
- *Projection* : attributs dans la clause `SELECT` sans fonction de calcul
  → également dans la clause `GROUP BY`
- *Projection* : inverse vrai dans certains SGBDR (Oracle par exemple)
- *Fonction de calcul* si `GROUP BY` : dans les clauses `SELECT` ou `HAVING`

Agrégat (Remarques)

Requêtes imbriquées

Requêtes imbriquées : recherches plus complexes
impossibles à écrire en *algèbre relationnelle*

→ *sous-requête* est une clause FROM ou WHERE

```
SELECT <liste d'attributs>
FROM <liste de relations>
WHERE <sous-requête>;

SELECT nom_civil, prenom
FROM Super_heros
WHERE nom_super IN
    (SELECT heros FROM Super_heros_pouvoirs);                    MAL

SELECT nom
FROM Chercheur
WHERE nom IN
    (SELECT nom FROM Enseignant);                                MAL
```


Requêtes imbriquées

Sous-requête d'existence IN

Présence de la *projection* d'un tuple de la *requête* dans la *sous-requête*

```
SELECT <liste d'attributs>
FROM <liste de relations>
WHERE <projection d'un tuple> IN (<sous-requête>);



                                                        -- une colonne
SELECT C.nom
FROM Chercheur C
WHERE C.universite IN (
    SELECT U.nom
    FROM Universite U
    WHERE U.ville = 'Paris');
```


Sous-requête d'existence IN

Présence de la *projection* d'un tuple de la *requête* dans la *sous-requête*

```
SELECT <liste d'attributs>
FROM <liste de relations>
WHERE <projection d'un tuple> IN (<sous-requête>);



                                                        -- plusieurs colonnes
SELECT num_ss
FROM Chercheur
WHERE (nom, prenom, age) IN (
     SELECT nom, prenom, age
     FROM Enseignant);
```

Sous-requête d'existence IN

Présence de la *projection* d'un tuple de la *requête* dans la *sous-requête*

```
SELECT <liste d'attributs>
FROM <liste de relations>
WHERE <projection d'un tuple> IN (<sous-requête>);


                                          -- imbrications multiples
SELECT nom
FROM Chercheur
WHERE universite = 'Paris 6' IN (
     SELECT nom
     FROM Enseignant
     WHERE universite IN (
         SELECT nom
         FROM Universite
         WHERE Ville = 'Paris'));
```


BD – SQL LMD : langage de manipulation des données

Sous-requête d'existence IN

Présence de la *projection* d'un tuple de la *requête* dans la *sous-requête*

```
SELECT <liste d'attributs>
FROM <liste de relations>
WHERE <projection d'un tuple> IN (<sous-requête>);
```

- Alternative à la *jointure* → à éviter
- Non existence d'un tuple dans une *sous-requête* → NOT IN

BD – SQL LMD : langage de manipulation des données

Sous-requête d'existence IN

BD – SQL LMD : langage de manipulation des données

Sous-requête d'existence IN

Sous-requête d'existence EXISTS

Existence d'au moins un tuple dans la *sous-requête*

```
SELECT <liste d'attributs>
FROM <liste de relations>
WHERE <projection d'un tuple> EXISTS (<sous-requête>);

SELECT nom_super, nom_civil, prenom
FROM Super_heros
WHERE EXISTS (
    SELECT *
    FROM Super_heros_pouvoirs
    WHERE nom_super = heros);
```


Sous-requête d'existence EXISTS

Existence d'au moins un tuple dans la *sous-requête*

```
SELECT <liste d'attributs>
FROM <liste de relations>
WHERE <projection d'un tuple> EXISTS (<sous-requête>);

SELECT Chercheur.nom
FROM Chercheur
WHERE EXISTS (
     SELECT *
     FROM Universite;
     WHERE Universite.nom = Chercheur.universite);
```


Sous-requête d'existence EXISTS

Existence d'au moins un tuple dans la *sous-requête*

```
SELECT <liste d'attributs>
FROM <liste de relations>
WHERE <projection d'un tuple> EXISTS (<sous-requête>);
```

- Pour valider l'existence d'un tuple
  → *projection* inutile dans dans la *sous-requête* (SELECT *)
- Non présence d'un tuple dans une *sous-requête* → NOT IN

Sous-requête d'existence EXISTS

Sous-requête de comparaison ALL

Vérification d'une condition : pour les tuples de la *requête*
sur tous les tuples de la *sous-requête*

```
SELECT <liste d'attributs>
FROM <liste de relations>
WHERE <attribut> <operateur de comparaison> ALL (<sous-requête>);

SELECT Nom
FROM Chercheur
WHERE age > ALL (
    SELECT age
    FROM Eleve;
```


Sous-requête de comparaison ALL

Sous-requête de comparaison ANY

Vérification d'une condition : pour les tuples de la *requête*
sur au moins un tuple de la *sous-requête*

```
SELECT <liste d'attributs>
FROM <liste de relations>
WHERE <attribut> <operateur de comparaison> ANY (<sous-requête>);

SELECT Nom
FROM Chercheur
WHERE age < ANY (
    SELECT age
    FROM Eleve);
```

SOME peut être utilisé comme synonyme de ANY

Sous-requête de comparaison ANY

Raffinement des requêtes dans la clause WHERE

Raffinement : enchaînement récursif de *n requêtes* dans la clause FROM

```
SELECT <liste d'attributs>
FROM <requête>;

SELECT avg(s)
FROM (
    SELECT sum(b) as s
    FROM t
    GROUP BY a);
```


BD – SQL LMD : langage de manipulation des données

Raffinement des requêtes dans la clause WHERE

2016-2017 ensiie 31

Exercice

Potion magique

- Modèle conceptuel
- Liste des potions et de leurs inventeurs
- Liste des ingrédients de la potion magique
- Nombre d'ingrédients de la potion magique
- Nombre d'ingrédients dans chaque potion

BD – SQL LMD : langage de manipulation des données

Exercice

2016-2017 ensiie 32

Super exemple : fil d'Ariane

Super_heros (SH)

#nom_super	nom_civil	prenom	age
M. Indestructible	Parrr	Robert	36
Elastic girl	Parrr	Helene	36
Violette	Parrr	Violette	16
Fleche	Parrr	Roger	12
Jack-Jack	Parrr	Jack	2
Frozone	Best	Lucius	36

Super_pouvoirs (SP)

#nom_pouvoir
Elasticite
Champ de glace
Champ de force
Invisibilite
Rapidite
Resistance

Super_heros_pouvoirs (SHP)

#heros=>SH	#pouvoir=>SP
Indestructible	Resistance
Elastic girl	Elasticite
Frozone	Champ de glace
Violette	Invisibilite
Violette	Champ de force
Fleche	Rapidite

Super exemple : fil d'Ariane

Insertion de données

Insertion *directe* des valeurs

```
INSERT INTO <table> (<liste ordonnée d'attributs>)
VALUES (<liste ordonnée des valeurs à affecter aux attributs>);

            -- si les parvenus étaient des super-héros comme les autres
INSERT INTO Super_heros (nom_super, nom_civil, prenom)
VALUES (Syndrome, Pine, Buddy);
```

Insertion de valeurs par *sélection*

```
INSERT INTO <table> (<liste ordonnée d'attributs>)
SELECT ...;                  -- nombre et type identiques que ci-dessus

            -- si les parvenus étaient des super-héros comme les autres
INSERT INTO Super_heros (nom_super, nom_civil, prenom)
SELECT nom_parvenu, nom_civil, prenom
FROM Parvenus;
```


Insertion de données

Insertion de données (Remarques)

Possibilité de ne pas spécifier un (ou plusieurs) attribut(s) NULL

```
                                    -- l'attribut age n'est pas spécifié
INSERT INTO Super_heros (nom_super, nom_civil, prenom)
VALUES (Syndrome, Pine, Buddy);

→ Syndrome, Pine, Buddy, NULL
```

Possibilité de ne pas spécifier la liste d'attributs (cas simples)

→ tous les attributs dans l'ordre de définition de la relation

```
INSERT INTO Super_heros            -- (nom_super, nom_civil, prenom, age)
SELECT *
FROM Parvenus;
```


Insertion de données (Remarques)

Mise à jour de données

```
UPDATE INTO <table>
SET <liste d'attributs = valeurs>
WHERE <condition pour filtrer les enregistrements>;
```

Mise à jour *directe* des valeurs

```
                                -- correction de l'orthographe du nom Parr
UPDATE Super_heros
SET nom_civil = 'Parr'
WHERE nom_civil = 'Parrr';
```

Mise à jour de valeurs par *calcul*

```
                                          -- happy birthday Violette...
UPDATE Super_heros
SET age=age+1
WHERE nom_super = 'Violette';
```


Mise à jour de données

Suppression de données

```
DELETE FROM <table>
WHERE <condition pour filtrer les enregistrements>;
```

Suppression de *tous* les enregistrements d'une relation

```
            -- si les parvenus étaient des super-héros comme les autres
DELETE FROM Parvenus;
```

Suppression *sélective* de enregistrements

```
           -- si les parvenus étaient des super-héros comme les autres,
                                                    -- ça se saurait...
DELETE FROM Super_heros
WHERE nom_super = 'Syndrome';
```


BD – SQL LMD : langage de manipulation des données

Suppression de données

LCD : langage de contrôle des données

Création d'utilisateurs
Définition de leurs droits → *SGBD*, *BD*, *relations* ou *attributs*

Remarque

Gestion de la création d'utilisateur (`CREATE USER`) → par le *SGBD* en général

BD – SQL LCD : langage de contrôle des données

LCD : langage de contrôle des données

Attribution des droits

```
GRANT <liste de droits> ON <table> TO <utilisateur>
[WITH GRANT OPTION];                              -- transfert de droits

GRANT SELECT, UPDATE ON Super_heros TO Edna Mode;
GRANT ALL PRIVILEGES ON Super_heros TO Government;
```

Liste des droits

- SELECT
- INSERT
- DELETE
- UPDATE
- ALTER
- ALL PRIVILEGES

Attribution de droits

Attribution des droits (Remarques)

- Droits sur un *SGBD* → Certains SGBD (MySQL)

  Droits CREATE et DROP généralement ajoutés
  (création et suppression de *BD* et des tables)

- Droits sur une *BD* → Certains SGBD (MySQL)

  Droits CREATE et DROP généralement ajoutés
  (création et suppression de tables)

- Droits sur une *vue*

  Syntaxe quasi-identique (VIEW à la place de TABLE)

Attribution de droits (Remarques)

Révocation de droits

```
REVOKE [GRANT OPTION FOR] <liste de droits>
ON <object> TO <utilisateur>

REVOKE UPDATE ON Super_heros TO Edna Mode;
REVOKE ALL PRIVILEGES ON Super_heros TO PUBLIC;
```

Droit supprimé pour un utilisateur U

→ droit supprimé en cascade pour les utilisateurs ayant obtenu le droit par U

BD – SQL LCD : langage de contrôle des données

Révocation de droits