SQL DML : Data Manipulation Language

Fabien Coelho
MINES ParisTech

Composé avec LaTeX, révision 4289.

Opérations élémentaires sur les données

sélection de données selon un critère

algèbre relationnelle, opérations ensemblistes...

```
SELECT nom FROM personnes WHERE nom LIKE 'H%';
```

insertion de nouvelles données

```
INSERT INTO personnes(nom) VALUES('calvin');
```

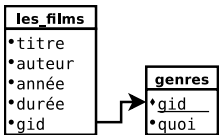
modification de données existantes

```
UPDATE personnes SET nom='Calvin' WHERE nom='calvin';
```

effacement de données

```
DELETE FROM personnes WHERE nom='Hobbes';
```


Faisons notre cinéma avec 2 tables



genres

gid	*quoi*
1	Drame
2	Comédie
3	Comédie dramatique
4	Action

les_films

titre	*auteur*	*année*	*durée*	*gid*
Citizen Kane	Wells	1936	01:59:00	1
The Dictator	Chaplin	1940	02:07:00	2
Modern Times	Chaplin	1936	01:27:00	2
City Lights	Chaplin	1931	01:27:00	3
Ohayô	Ozu	1959	01:37:00	2
Le Procès	Wells	1963	01:58:00	1

La commande SELECT

— manipulation de **tables existantes**
— pour obtenir **une nouvelle table**

projection sélection de colonnes (attributs)

restriction sélection de lignes (tuples)

produit cartésien, jointure de plusieurs tables

tri des lignes ascendant ou descendant

aggrégation des lignes (somme, comptage)

sous requêtes dans expressions ou tables...

union intersect except opérateurs ensemblistes si tables compatibles

Syntaxe de SELECT

```
SELECT [ALL|DISTINCT] expression [AS colname], ...
FROM tablename [AS alias], ...
WHERE tuple condition
GROUP BY expr/colname/colnum, ...
HAVING aggregate condition
UNION/EXCEPT/INTERSECT ... SELECT ... composition
ORDER BY expr/colname/colnum [ASC|DESC], ...
LIMIT number
OFFSET start;
```

Lien avec l'algèbre relationnelle

SELECT FROM WHERE projection, produit cartésien, restrictions

UNION EXCEPT INTERSECT opérateurs ensemblistes

Expressions

```
SELECT 1789 AS année, 'révolution française' AS évènement;
```

année	*évènement*
1789	révolution française

```
SELECT 2*PI()*6300 AS périmètre, 'terre' AS planète;
```

périmètre	*planète*
39584.067435231394	terre

```
SELECT EXTRACT(DOW FROM NOW()) AS jour, CURRENT_DATE;
```

jour	*current_date*
4	2020-12-10

Expressions : Types, Constantes, Opérateurs, Fonctions

entiers et flottants SMALLINT INTEGER REAL FLOAT...

constantes 123 $1.25E12$, opérations `+ - * / %`

fonctions diverses `SQRT(2.0)` `PI()` `ABS(-1)`

booléens BOOLEAN, valeurs TRUE FALSE NULL

opérations AND OR NOT..., comparaisons `= <> > <=`...

textes TEXT CHAR(10) VARCHAR(32)

constante `'hello'`, opérations `||`, comparaison LIKE

autres temps, binaires, géométrie, argent, extensions...

valeur NULL possible pour tous les types !

Projection de colonnes SELECT ... FROM ...

— clause FROM précise les tables (schéma, alias)
— sélection des colonnes, avec nommage éventuel
— appliqué à chaque ligne des tables

```
-- simple
SELECT quoi FROM Genres;

-- nommage et alias
SELECT g.quoi AS genre
FROM Genres AS g;

-- table
SELECT Genres.quoi
FROM Genres;
```

quoi
Drame
Comédie
Comédie dramatique
Action

Projection de colonnes (suite)

```
-- sélection de colonnes...
SELECT titre, année, auteur AS nom
-- précise le schéma
FROM public.les_films;
```

titre	*année*	*nom*
Citizen Kane	1936	Wells
The Dictator	1940	Chaplin
Modern Times	1936	Chaplin
City Lights	1931	Chaplin
Ohayô	1959	Ozu
Le Procès	1963	Wells

Projection de toutes les colonnes

— `*` ou `nomTable.*` ou `alias.*`...

— si plusieurs tables, problème colonnes des homonymes

```
-- toutes les colonnes
SELECT *
FROM Genres;
-- idem avec nom table
SELECT Genres.*
FROM Genres;
-- idem avec alias
SELECT g.*
FROM Genres AS g;
-- avec un raccourci
TABLE Genres;
```

gid	*quoi*
1	Drame
2	Comédie
3	Comédie dramatique
4	Action

Projection + expressions + nommage

```
-- opérations arithmétiques
SELECT titre, (année+99)/100 AS siècle
FROM les_films;
```

titre	*siècle*
Citizen Kane	20
The Dictator	20
Modern Times	20
City Lights	20
Ohayô	20
Le Procès	20

Restriction de lignes `WHERE`

— clause `WHERE` *condition*

— expression booléenne impliquant les colonnes

```
-- films d'avant guerre
SELECT titre, auteur, année
FROM les_films
WHERE année < 1940;
```

titre	*auteur*	*année*
Citizen Kane	Wells	1936
Modern Times	Chaplin	1936
City Lights	Chaplin	1931

```
-- films de Chaplin avant 1940
SELECT auteur, titre, année
FROM les_films
WHERE auteur = 'Chaplin'
  AND année < 1940;
```

auteur	*titre*	*année*
Chaplin	Modern Times	1936
Chaplin	City Lights	1931

Restriction exprimées sur l'entrée

— condition vérifiée indépendemment de la projection

```
-- films de l'année 36
SELECT titre, auteur
FROM les_films
WHERE année = 1936;
```

titre	*auteur*
Citizen Kane	Wells
Modern Times	Chaplin

```
-- auteur d'un film
SELECT auteur
FROM les_films
WHERE titre = 'Citizen Kane';
```

auteur
Wells

Restriction du nombre de lignes `LIMIT` / `OFFSET`

— limitation avec `LIMIT` *nombre*

— décalage initial éventuel avec `OFFSET` *décalage*

— résultat plus pertinent si trié *hit-parade*

```
-- trois premiers films
SELECT titre, auteur
FROM les_films
LIMIT 3;
```

titre	*auteur*
Citizen Kane	Wells
The Dictator	Chaplin
Modern Times	Chaplin

```
-- deux suivants
SELECT titre, auteur
FROM les_films
LIMIT 2
OFFSET 3;
```

titre	*auteur*
City Lights	Chaplin
Ohayô	Ozu

Suppression des doublons ? `DISTINCT` vs `ALL`

suppression option `DISTINCT` de `SELECT`, ensembliste

conservation par défaut, option `ALL` non ensembliste

```
-- auteurs avec doublons
SELECT ALL auteur
FROM les_films;
```

auteur
Wells
Chaplin
Chaplin
Chaplin
Ozu
Wells

```
-- auteurs sans doublons
SELECT DISTINCT auteur
FROM les_films;
```

auteur
Chaplin
Ozu
Wells

Tri des lignes `ORDER BY` *expr/colname/colnum* [`ASC` / `DESC`], ...

— par défaut résultats non déterministes (selon stockage, calculs...)

— tri selon type, lexicographique, ordre croissant ou décroissant

```
-- ordre alphabétique auteurs puis années
SELECT auteur, année, titre
FROM les_films
ORDER BY auteur ASC, année ASC;
```

auteur	*année*	*titre*
Chaplin	1931	City Lights
Chaplin	1936	Modern Times
Chaplin	1940	The Dictator
Ozu	1959	Ohayô
Wells	1936	Citizen Kane
Wells	1963	Le Procès

Tri des lignes (suite)

```
-- ordre alphabétique inverse des titres
SELECT auteur, année, titre
FROM les_films
ORDER BY 3 DESC;
```

auteur	*année*	*titre*
Chaplin	1940	The Dictator
Ozu	1959	Ohayô
Chaplin	1936	Modern Times
Wells	1963	Le Procès
Chaplin	1931	City Lights
Wells	1936	Citizen Kane

Tri + Restriction + Projection

```
-- films de Charlie Chaplin
SELECT titre, année
FROM les_films
WHERE auteur='Chaplin'
ORDER BY année ASC;
```

titre	*année*
City Lights	1931
Modern Times	1936
The Dictator	1940

```
-- films ordonnés par année
SELECT titre, auteur
FROM les_films
ORDER BY année ASC;
```

titre	*auteur*
City Lights	Chaplin
Citizen Kane	Wells
Modern Times	Chaplin
The Dictator	Chaplin
Ohayô	Ozu
Le Procès	Wells

Produit cartésien `CROSS JOIN` ou `,`

```
-- produit cartésien : films X genres
SELECT f.*, g.gid AS gid2, g.quoi
FROM les_films AS f CROSS JOIN Genres AS g
LIMIT 13;
```

titre	*auteur*	*année*	*durée*	*gid*	*gid2*	*quoi*
Citizen Kane	Wells	1936	01 :59 :00	1	1	Drame
Citizen Kane	Wells	1936	01 :59 :00	1	2	Comédie
Citizen Kane	Wells	1936	01 :59 :00	1	3	Comédie dramatique
Citizen Kane	Wells	1936	01 :59 :00	1	4	Action
The Dictator	Chaplin	1940	02 :07 :00	2	1	Drame
The Dictator	Chaplin	1940	02 :07 :00	2	2	Comédie
The Dictator	Chaplin	1940	02 :07 :00	2	3	Comédie dramatique
The Dictator	Chaplin	1940	02 :07 :00	2	4	Action
Modern Times	Chaplin	1936	01 :27 :00	2	1	Drame
Modern Times	Chaplin	1936	01 :27 :00	2	2	Comédie
Modern Times	Chaplin	1936	01 :27 :00	2	3	Comédie dramatique
Modern Times	Chaplin	1936	01 :27 :00	2	4	Action
City Lights	Chaplin	1931	01 :27 :00	3	1	Drame

Jointure simple sur un champ

```
-- jointure sur le genre (join on)
SELECT titre, auteur, année, quoi
FROM les_films AS f JOIN Genres AS g ON f.gid=g.gid;

-- jointure sur le genre (X et where)
SELECT titre, auteur, année, quoi
FROM les_films AS f, Genres AS g
WHERE f.gid = g.gid;
```

titre	*auteur*	*année*	*quoi*
Citizen Kane	Wells	1936	Drame
The Dictator	Chaplin	1940	Comédie
Modern Times	Chaplin	1936	Comédie
City Lights	Chaplin	1931	Comédie dramatique
Ohayô	Ozu	1959	Comédie
Le Procès	Wells	1963	Drame

Jointure + Restriction + Projection + Tri

```
-- les genres des films de Chaplin
SELECT titre, année, quoi
FROM les_films AS f
JOIN Genres AS g ON f.gid = g.gid
WHERE auteur = 'Chaplin'
ORDER BY année ASC;
```

titre	*année*	*quoi*
City Lights	1931	Comédie dramatique
Modern Times	1936	Comédie
The Dictator	1940	Comédie

Auto-jointure

- lien de deux tuples d'une **même** table !
- conflits nécessitent d'utiliser des **alias**

```
-- films distincts de même durée
SELECT f1.titre AS film1, f2.titre AS film2
FROM les_films AS f1
JOIN les_films AS f2 ON f1.durée = f2.durée
WHERE f1.titre <> f2.titre;
```

film1	*film2*
Modern Times	City Lights
City Lights	Modern Times

Aggrégation de lignes avec répétitions

Nom	Note
Calvin	4
Calvin	2
Calvin	3
Hobbes	1
Hobbes	2
Moe	0
Moe	3
Moe	2

Regroupement `GROUP BY` et Aggrégation `AVG`

	Nom	Note	
Calvin	Calvin	4	
	Calvin	2	AVG=3.00
	Calvin	3	
Hobbes	Hobbes	1	
	Hobbes	2	AVG=1.50
Moe	Moe	0	
	Moe	3	AVG=1.66
	Moe	2	

Aggrégation des lignes `GROUP BY`

regroupement d'attributs de valeurs identiques

clause `GROUP BY` *expr/colname/colnum...*

accumulation des autres valeurs avec une fonction spécifique

`COUNT(*)` comptage occurences

`COUNT(DISTINCT ...)` comptage occurences distinctes

`MAX MIN SUM AVG VARIANCE STDDEV`

minimum, maximum, somme, moyenne, variance, déviation...

`ARRAY_AGG, JSON_AGG...`

extensions ajout possible de nouvelles aggrégations...

Aggrégation d'un attribut

```
-- nombre de films total
SELECT COUNT(*) AS nfilms
FROM les_films;
```

nfilms
6

```
-- année du premier film
SELECT MIN(année) AS année
FROM les_films;
```

année
1931

Aggrégation

```
-- nombre de films par auteurs
SELECT auteur, COUNT(*) AS nfilms
FROM les_films
GROUP BY auteur
ORDER BY nfilms DESC, auteur;
```

auteur	*nfilms*
Chaplin	3
Wells	2
Ozu	1

```
-- durée d'activité
SELECT auteur,
  MAX(année)-MIN(année) AS activité
FROM les_films
GROUP BY auteur
ORDER BY activité DESC;
```

auteur	*activité*
Wells	27
Chaplin	9
Ozu	0

Aggrégation (suite)

```
-- nombre de film et durée moyenne par auteur
SELECT auteur, COUNT(*) AS nfilms, AVG(durée) AS mdurée
FROM les_films
GROUP BY auteur
ORDER BY mdurée DESC;
```

auteur	*nfilms*	*mdurée*
Wells	2	01:58:30
Chaplin	3	01:40:20
Ozu	1	01:37:00

Aggrégation (encore)

```
-- nombre de genres différents par auteur
SELECT auteur, COUNT(DISTINCT quoi) AS ngenres
FROM les_films AS f
JOIN Genres AS g ON f.gid=g.gid
GROUP BY auteur
ORDER BY ngenres DESC, auteur;
```

auteur	*ngenres*
Chaplin	2
Ozu	1
Wells	1

Aggrégation avec condition `HAVING`

— clause `HAVING` *condition*

— condition porte une **aggrégation** projetée ou non

```
-- auteurs avec plusieurs films
SELECT auteur
FROM les_films
GROUP BY auteur
HAVING COUNT(*)>1;
```

auteur
Chaplin
Wells

`WHERE` vs `HAVING`

— clause `WHERE` condition sur table en **entrée**

— clause `HAVING` condition sur table aggrégée en **sortie**

```
-- auteur et durée moyenne de plusieurs films avant 1940
SELECT auteur, AVG(durée) AS mdurée
FROM les_films
WHERE année < 1940
GROUP BY auteur
HAVING COUNT(*) > 1;
```

auteur	*mdurée*
Chaplin	01:27:00

Exercices

— quel jour sommes-nous ?

— quelle heure est-il ?

— les cinéastes dont les noms commencent par A, B ou C

— les cinéastes ayant fait un film avant guerre

— les titres des films de *Hayao Miyazaki*

— les titres des films dramatiques (*Drame*)

— les titres des films d'après guerre, triés par année

— le nombre de films par an

— le nombre de films par décennie (histogramme)

— le nombre de films par genres

— la moyenne *géométrique* des durées de films

— les cinéastes ayant sorti des films la même année, sans doublons

Expressions : Types, Constantes, Opérateurs, Fonctions

Valeur `NULL` pour tous les types

— valeur absorbante ou ignorée par les opérations...
— comparaisons : `d IS NULL`, `e IS NOT NULL`
— **attention** `x = NULL` vaut `NULL`, donc faux !
mais `NULL OR TRUE` vaut `TRUE`
— `COALESCE(t,u,...)` : premier *non null*

a	*b*	*a=b*	*a!=b*	*a = NULL*	*a != NULL*	*a IS NULL*	*a IS NOT NULL*
0	0	t	f			f	t
1	0	f	t			f	t
	0					t	f
	1					t	f
						t	f

booléens `BOOLEAN` valeurs `TRUE FALSE`... et `NULL` pour **inconnue**
— opérations `AND OR NOT` : *a*`>10 AND NOT` *b*`>100`
— comparaisons directes de valeurs `= != <>`
— double comparaison *a* `BETWEEN` *b* `AND` *c*

entiers `SMALLINT INTEGER NUMERIC`...
— constantes `5432 -10 10000000000000000000000000`
— opérations unaires `+ -`, binaires `+ - * /`, modulo `%`
— comparaisons classiques `< > <= >=`
— priorités classiques, division entre entiers **entière** !
— fonctions `MOD(3,2)=1 ABS(-1)=1`

flottants `FLOAT4 REAL FLOAT8 FLOAT DOUBLE PRECISION`
— constantes `2.71828182845905 -1.23E2`
— précision arbitraire `DECIMAL(10,2)` (précision, échelle)
— arithmétique `+ - * /`, comparaisons...
— nombreuses fonctions `POW(2,3) LN(10) EXP(3.1)`
`LOG(123) COS(1.3) SIN(1.0) ATAN(2.0) SQRT(4)`
`CBRT(27.0) PI() RANDOM()`...

textes constantes *quotées* `'données' 'l''artiste'`
— opérateur *concat.* `||`
— fonctions `CONCAT('hello', 'world')`
`LENGTH('hi') LOWER('Hi') UPPER('hello')`
`TRIM(' hobbes ') SUBSTRING('Calvin',2,4)`

Opérateurs `LIKE` / `SIMILAR TO` / `~`

— expressions régulières : comparaison texte et **motif**
caractères réguliers et **spéciaux** pour joker, répétition...
— reconnaissance en temps **linéaire** par **automates**
— 3 versions : SQL92, SQL99 et POSIX

`[NOT] LIKE` _ un caractère, `%` une chaîne

`[NOT] SIMILAR TO` *idem* plus `| * + [a-z] (...)`

opérateur `~` expressions régulières standard POSIX

```
-- films commençant par C
SELECT titre, auteur
FROM les_films
WHERE titre LIKE 'C%';
```

titre	*auteur*
Citizen Kane	Wells
City Lights	Chaplin


```
-- titres sans deux majuscules
SELECT auteur, titre
FROM les_films
WHERE titre NOT SIMILAR TO '%[A-Z]%[A-Z]%';
```

auteur	*titre*
Ozu	Ohayô

```
-- titres accentués
SELECT titre, auteur, année
FROM les_films
WHERE titre SIMILAR TO '%[^a-zA-Z ]%';
```

titre	*auteur*	*année*
Le Procès	Wells	1963
Ohayô	Ozu	1959

temps `DATE TIME TIMETZ TIMESTAMP INTERVAL`
— constantes `'1970-03-20'::DATE TIME '12:30:45'`
— opérateurs arithmétiques `+ - * /` et `OVERLAPS`
`DATE 'today' + INTERVAL '1 month'`
— fontions `NOW() CURRENT_DATE CURRENT_TIME`
extraction `EXTRACT(YEAR FROM une_date)`
avec `CENTURY DECADE DOW HOUR MINUTE SECOND`...
cf `AGE JUSTIFY_HOURS JUSTIFY_DAYS TO_CHAR`...

binaires `BIT(12) VARBIT(31) BYTEA`

géométrie 2D `BOX CIRCLE LINE POINT POLYGON PATH`...
— constantes `'(1,1),(2,3),(0,0)'::POLYGON`
— nombreuses opérations : contient, rotation, distance...
`POINT '(2,0)' @ BOX '(-1,-1),(1,1)'`

réseau `INET MACADDR`

`INET '192.168.1.5' << INET '192.168.1/24'`

extensions ajoutables dynamiquement !

bibliothèque `ISSN ISBN`

Expression conditionnelle `CASE` (1)

CASE

WHEN cond1 THEN res1 condition et valeur

WHEN cond2 THEN res2 condition et valeur...

ELSE resdef END valeur finale par défaut

```
SELECT nom,
  CASE
    WHEN décès IS NULL THEN 'vivant'
    ELSE 'décédé'
  END AS "actuel."
FROM Personnes;
```

nom	*actuel.*
Chaplin	décédé
Wells	décédé
Ozu	décédé
Allen	vivant
Carné	décédé
Tarentino	vivant

Expression conditionnelle `CASE` (2)

CASE expr calcule une expression

WHEN val1 THEN res1 énumération de valeurs

ELSE def END valeur par défaut

```
SELECT nom,
  CASE EXTRACT(CENTURY FROM naissance)
    WHEN 21 THEN 'XXIème'
    WHEN 20 THEN 'XXème'
    WHEN 19 THEN 'XIXème'
    ELSE 'ne sait pas'
  END AS "né au"
FROM Personnes;
```

nom	*né au*
Chaplin	XIXème
Wells	XXème
Ozu	XXème
Allen	XXème
Carné	XXème
Tarentino	XXème

3 types de jointures

cartésienne *(cross join)*, rarement utile !

interne *(inner join)* condition `ON` / `USING` / `NATURAL` / `WHERE`

chaque tuple apparaît **si correspondance**

éviter `NATURAL` et `WHERE`...

externe *(outer join)* condition `ON` / `USING` / `NATURAL`

3 sortes `LEFT` / `RIGHT` / `FULL`

tuples apparaissent **même sans correspondance** !

contient la jointure interne

prénoms

prénom	*id*
Albert	1
Allan	2
Kurt	3

noms

id	*nom*
1	Einstein
1	Camus
3	Gödel
4	Church

Jointure cartésienne `CROSS JOIN` 2 syntaxes !

```
SELECT p.prénom, n.nom
FROM Prénoms AS p CROSS JOIN Noms AS n;

SELECT p.prénom, n.nom
FROM Prénoms AS p, Noms AS n;
```

prénom	*nom*
Albert	Einstein
Albert	Camus
Albert	Gödel
Albert	Church
Allan	Einstein
Allan	Camus
Allan	Gödel
Allan	Church
Kurt	Einstein
Kurt	Camus
Kurt	Gödel
Kurt	Church

Jointure interne `[INNER] JOIN` 4 syntaxes !

```
SELECT p.prénom, n.nom -- on condition
FROM Prénoms AS p INNER JOIN Noms AS n ON p.id=n.id;

SELECT p.prénom, n.nom -- using colname
FROM Prénoms AS p INNER JOIN Noms AS n USING (id);

SELECT p.prénom, n.nom -- natural = même colname
FROM Prénoms AS p NATURAL INNER JOIN Noms AS n;
SELECT p.prénom, n.nom -- X et where condition
FROM Prénoms AS p, Noms AS n
WHERE p.id=n.id;
```

prénom	*nom*
Albert	Einstein
Albert	Camus
Kurt	Gödel

Jointure externe gauche `LEFT [OUTER] JOIN`

— **tous** les tuples de la table **gauche** apparaissent

— ajout d'un tuple **droit** `NULL` si nécessaire

— pas de condition de **jointure** dans le `WHERE` (ambiguïté)

```
SELECT p.prénom, n.nom
FROM Prénoms AS p
LEFT JOIN Noms AS n USING (id);
```

prénom	*nom*
Albert	Einstein
Albert	Camus
Allan	
Kurt	Gödel

Jointure externe droite `RIGHT [OUTER] JOIN`

— **tous** les tuples de la table **droite** apparaissent

— ajout d'un tuple **gauche** `NULL` si nécessaire

```
SELECT p.prénom, n.nom
FROM Prénoms AS p RIGHT JOIN Noms AS n USING (id);
```

prénom	*nom*
Albert	Einstein
Albert	Camus
Kurt	Gödel
	Church

Jointure externe totale `FULL [OUTER] JOIN`

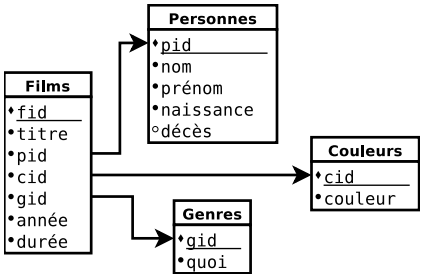
— **tous** les tuples apparaissent, ajout tuples `NULL`

```
SELECT p.prénom, n.nom
FROM Prénoms AS p FULL JOIN Noms AS n ON p.id=n.id;
```

prénom	*nom*
Albert	Einstein
Albert	Camus
Allan	
Kurt	Gödel
	Church

Faisons notre cinema avec 4 tables

— modèle normalisé, pas de redondance



Personnes

pid	*nom*	*prénom*	*naissance*	*décès*
1	Chaplin	Charles	1889-04-16	1977-12-25
2	Wells	Orson	1915-05-06	1985-10-10
3	Ozu	Yasujiro	1903-12-12	1963-12-12
4	Allen	Woody	1935-12-01	
5	Carné	Marcel	1909-08-18	1996-10-31
6	Tarentino	Quentin	1963-03-27	

Genres

gid	*quoi*
1	Drame
2	Comédie
3	Comédie dramatique
4	Action

Couleurs

cid	*couleur*
1	N&B
2	couleur

Films

fid	*titre*	*pid*	*cid*	*gid*	*année*	*durée*
1	Citizen Kane	2	1	1	1936	01:59:00
2	The Dictator	1	1	2	1940	02:07:00
3	Modern Times	1	1	2	1936	01:27:00
4	City Lights	1	1	3	1931	01:27:00
5	Ohayô	3	2	2	1959	01:37:00
6	Le Procès	2	2	1	1963	01:58:00
7	Kill Bill v1	6	2	4	2003	01:51:00
8	Monsieur Verdoux	1	1	3	1947	02:04:00
9	Limelight	1	1	3	1952	02:21:00
10	King in New York	1	1	3	1957	01:50:00
11	Alice	4	2	2	1990	01:42:00
12	The Trial	2	1	1	1962	01:58:00

Jointure externe pour passage au complémentaire

```
-- auteurs sans films avec NULL...
SELECT prénom, nom
FROM Personnes AS p LEFT JOIN Films AS f USING(pid)
WHERE f.fid IS NULL;
```

prénom	*nom*
Marcel	Carné

Opérations ensemblistes

— opérateurs entre tables `UNION INTERSECT EXCEPT`

— éventuellement non ensembliste (duplications) `ALL`

— tables compatibles ! attributs de même types

```
SELECT 1 AS nb
UNION ALL
SELECT 2 AS nb
UNION ALL
SELECT 1 AS nb;
```

nb
1
2
1

```
SELECT id FROM Prénoms
INTERSECT
SELECT id FROM Noms;
```

id
3
1

Exemple avec `EXCEPT`

```
-- les auteurs sans films...
-- tous les auteurs
SELECT nom, prénom
FROM Personnes
-- moins
EXCEPT
-- ceux avec des films
SELECT DISTINCT nom, prénom
FROM Personnes AS p
JOIN Films AS f ON p.pid=f.pid;
```

nom	*prénom*
Carné	Marcel

Sous requêtes *(subqueries)*

— **algèbre relationnelle** : requête **dans** une requête

— à la place d'une table dans une clause `FROM`

```
SELECT ...
FROM
  (SELECT ...
   FROM ...
   WHERE ...) AS alias...
WHERE ...;
```

— expressions booléennes, opérateurs `EXISTS IN ANY ALL`

sous-requête `SELECT` simplifié (pas de `ORDER...`)

```
SELECT ... FROM ...
WHERE attr NOT IN
  (SELECT ... FROM ... WHERE ...);
```

Sous-requête dans un `FROM`

```
-- plus petite durée moyenne par auteur
SELECT MIN(moy) AS durée
FROM
 (SELECT nom, AVG(durée) AS moy
  FROM Films AS f
  JOIN Personnes AS p ON f.pid=p.pid
  GROUP BY nom) AS avdur;
```

durée
01:37:00

Sous-requête dans une condition `WHERE`

```
-- films courts
SELECT titre
FROM Films
WHERE durée < ANY(SELECT AVG(durée) FROM Films);
```

titre
Modern Times
City Lights
Ohayô
Kill Bill v1
King in New York
Alice

Sous-requête + `UNION`

```
-- nb de films pour tous les auteurs
SELECT nom, SUM(nfilms) AS nfilms
FROM
 (-- nb films des auteurs actifs
  SELECT nom, COUNT(*)
  FROM Personnes AS p
  JOIN Films AS f USING (pid)
  GROUP BY nom
  UNION
  -- tous les auteurs
  SELECT nom, 0
  FROM Personnes) AS f(nom,nfilms)
GROUP BY nom
ORDER BY nfilms DESC, nom ASC;
```

nom	*nfilms*
Chaplin	6
Wells	3
Allen	1
Ozu	1
Tarentino	1
Carné	0

Plus simple avec `UNION` et jointure externe

```
-- auteurs avec films, jointure interne
SELECT nom, COUNT(*) AS nfilms
FROM Personnes JOIN Films USING (pid)
GROUP BY nom
UNION
-- auteurs sans films, jointure externe
SELECT nom, 0
FROM Personnes LEFT JOIN Films USING (pid)
WHERE fid IS NULL
-- ordre commun
ORDER BY nfilms DESC, nom ASC;
```

nom	*nfilms*
Chaplin	6
Wells	3
Allen	1
Ozu	1
Tarentino	1
Carné	0

Astuce spécifique PostgreSQL

— jointure externe pour **tous**

— `COUNT(NULL)` vaut 0 !

```
-- nb films tous auteurs (PGSQL!)
SELECT nom, COUNT(f.fid)
FROM Personnes AS p
LEFT JOIN Films AS f USING(pid)
GROUP BY nom;
```

nom	*count*
Carné	0
Chaplin	6
Tarentino	1
Ozu	1
Allen	1
Wells	3

Conseils pour le développement des requêtes

1. déterminer les tables utiles
   — résultats ou conditions
   — tables de liaisons
   — tables *multiples* ?
2. créer le `SELECT FROM` et les jointures `ON USING`
3. ajouter les conditions `WHERE`
4. ajouter les aggrégations `GROUP BY`
   et condition sur aggrégation `HAVING`
5. ajouter les tris `ORDER BY`
6. décomposer si nécessaire (sous requêtes, ensembles) ?
7. simplifier les requêtes (multiples...) avec des vues ?

Exercices

— quand tombe $2^{31} - 1$ secondes après le 1 janvier 1970 ?
— les cinéastes vivants
— les titres des films d'action
— les cinéastes nés au XIXème siècle
— les cinéastes par ordre de naissance
— les titres des comédies en couleurs de *Charles Chaplin*
— les cinéastes avec des comédies en noir et blanc
— la durée de vie de tous les cinéastes, en ordre décroissant
— les cinéastes décédés plus jeunes que la moyenne
— les films des cinéastes nés avant 1910
— le(s) cinéaste(s) dont les films sont en moyenne les plus longs

Réfléchir au problème suivant

— soient deux tables de même schéma *(id INT PK, val TEXT NN)*
— identifier (id) et caractériser (insert/update/delete) les différences pour passer le T1 à T2

Relation T1

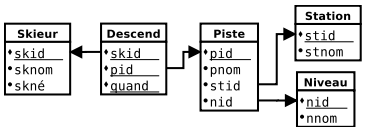
id	*val*
1	un
2	deux
3	trois

Relation T2

id	*val*
1	one
2	deux
4	quatre

Différences

id	*opération*
1	UPDATE
3	DELETE
4	INSERT



- Quelles sont les pistes *noir*es de *Courchevel* par ordre alphabétique ?
- Quelles pistes de quelles stations a descendu *Susie* en *2018*, ordonnées ?
- Combien de pistes *bleue*s a descendu *Calvin* à *Val d'Isère* en *2020* ?
- Quelles pistes, par stations, n'ont jamais été descendues, ordonnées ?
- Quels skieurs ont descendu la piste la plus fréquentée en *2018*, ordonnés ?
- Pour toutes les pistes, donner leur nombre de skieurs différents, ordonnés.
- Pour tous les skieurs, de 2018 à 2020, le nombre de descentes, ordonné.
- La fréquentation (nombre de skieurs) de toutes les stations en 2018.

Requêtes graphiques

- interface graphique de fabrication d'un `SELECT`
- *à la* MS Access, pgaccess...
- limitations éventuelles pour des requêtes avancées
  sous-requêtes, ensembles, fonctions, types de jointures, expressions...

Insertions d'une ligne `INSERT`

- commande `INSERT INTO ... VALUES (...), ...`
- donne la table et éventuellement la liste de colonnes
  il est préférable de préciser cette liste (modif du schéma)
- si colonne non spécifiée, valeur par défaut
- types des valeurs doivent être compatibles !

```
-- salut les copains
INSERT INTO uneTable(id,quoi,quand) VALUES
  (5, 'Calvin', DATE '1985-03-15');

-- attributs anonymes, à éviter !
INSERT INTO uneTable VALUES
  (6, 'Hobbes', NOW()),
  (7, 'Moe', DATE 'yesterday');
```


Insertion de lignes `INSERT ... SELECT`

- remplissage d'une table à partir d'une requête
- permet de transférer des données entre tables
- voir aussi `CREATE TEMPORARY TABLE ... AS ...`

```
INSERT INTO Relation
SELECT ... FROM ... WHERE ... ;
```


Effacement de lignes `DELETE`

- commande `DELETE FROM ... WHERE ...`
- précise la table concernée
- condition similaire à un `SELECT`
  peut toucher plusieurs lignes
- voir aussi `TRUNCATE TABLE ...`

```
-- efface un tuple
DELETE FROM maTable
WHERE id=123;

-- effacement complet
DELETE FROM maRelation;
```


Mise à jour de lignes `UPDATE`

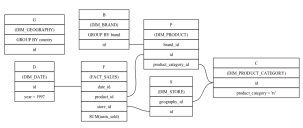
- commande `UPDATE ... SET ...=... WHERE ...`
- modifications d'un ou plusieurs attributs
- condition similaire à un `SELECT`
- peut toucher plusieurs lignes

```
-- corrigeons...
UPDATE maTable
SET prenom='Calvin', copain='Hobbes'
WHERE age=6;
```


Visualisation de requêtes complexes

http://revj.sourceforge.net/

```
SELECT B.Brand, G.Country, SUM (F.Units_Sold)
FROM Fact_Sales F
JOIN Dim_Date D ON F.Date_Id = D.Id
JOIN Dim_Store S ON F.Store_Id = S.Id
JOIN Dim_Geography G ON S.Geography_Id = G.Id
JOIN Dim_Product P ON F.Product_Id = P.Id
JOIN Dim_Product_Category C ON P.Product_Category_Id = C.Id
JOIN Dim_Brand B ON P.Brand_Id = B.Id
WHERE D.Year = 1997 AND C.Product_Category = 'tv'
GROUP BY B.Brand, G.Country;
```



Embellissement de requêtes

https://paste.depesz.com/

http://sqlformat.darold.net/

List of Slides