

## SQL DML : Data Manipulation Language

Fabien Coelho  
MINES ParisTech

Composé avec l'fX, révision 2531.

1

## 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

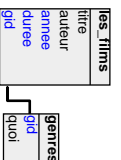
```
UPDATE personnes SET nom='Calvin' WHERE nom='Calvin';
```

**effacement** de données

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

2

## Faisons notre cinéma avec 2 tables



gid	quoi
1	Drame
2	Comédie
3	Comédie dramatique
4	Action

titre	auteur	annee	duree	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

3

## 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

**agrégation** des lignes (somme, comptage)

**sous requêtes** dans expressions ou tables...

**union intersect except** opérateurs ensemblistes si tables compatibles

4

## Syntaxe de SELECT

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

### Lien avec l'algèbre relationnelle

**SELECT ...** projection de colonnes

**FROM ...** produit cartésien des tables

**WHERE ...** restriction des lignes

5

## Expressions

```
SELECT 1789 AS annee, 'révolution française' AS evenement;
annee | evenement
-----|-----
1789  | révolution française
```

```
SELECT 2*PI()*6300 AS perimetre, 'terre' AS planete;
```

perimetre	planete
39564.0674552314	terre

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

jour	date
2	2012-03-27

6

## Expressions : Types, Constantes, Opérateurs, Fonctions

**entiers et flottants** **SMALLINT** **INTEGER** **REAL** **FLOAT**...

constantes 123 1.25 E12, opérations + - \* / %

fonctions diverses **SQL**(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!

7

## 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';
```

2

## 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

**agrégation** des lignes (somme, comptage)

**sous requêtes** dans expressions ou tables...

**union intersect except** opérateurs ensemblistes si tables compatibles

4

## Expressions

```
SELECT 1789 AS annee, 'révolution française' AS evenement;
annee | evenement
-----|-----
1789  | révolution française
```

```
SELECT 2*PI()*6300 AS perimetre, 'terre' AS planete;
```

perimetre	planete
39564.0674552314	terre

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

jour	date
2	2012-03-27

6

## 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;
```

8

**Projection de colonnes (suite)**

```
-- sélection de colonnes...
SELECT titre, annee, auteur AS nom
-- précise le schéma
FROM public.les-films;
```

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

9

**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;
```

gid	quoi
1	Drame
2	Comédie
3	Comédie dramatique
4	Action

10

**Projection + expressions + nommage**

```
-- opérations arithmétiques
SELECT titre, (annee+99)/100 AS siecle
FROM les-films;
```

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

11

**Restriction de lignes WHERE**

```
- clause WHERE condition
```

```
- expression booléenne impliquant les colonnes
```

```
-- films d'avant guerre
```

```
SELECT titre, auteur, annee
```

```
FROM les-films
```

```
WHERE annee<1940;
```

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

```
-- films de Chaplin avant 1940
```

```
SELECT auteur, titre, annee
```

```
FROM les-films
```

```
WHERE auteur='Chaplin'
```

```
AND annee<1940;
```

auteur	titre	annee
Chaplin	Modern Times	1936
Chaplin	City Lights	1931

12

**Restriction exprimées sur l'entrée**

```
- condition vérifiée indépendamment de la projection
```

```
-- films de l'année 36
```

```
SELECT titre, auteur
```

```
FROM les-films
```

```
WHERE annee=1936;
```

titre	auteur
Citizen Kane	Wells
Modern Times	Chaplin

```
-- auteur d'un film
```

```
SELECT auteur
```

```
FROM les-films
```

```
WHERE titre='Citizen Kane';
```

auteur
Wells

13

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

```
- limitation avec LIMIT nombre
```

```
- décalage initial éventuel avec OFFSET décalage
```

```
- résultat plus pertinent si titré hi-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

14

**Suppression des doublons ? DISTINCT vs ALL**

```
suppression option DISTINCT de SELECT, ensemble
```

```
conservation par défaut, option ALL non ensemble
```

```
-- 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

15

**Ti des lignes ORDER BY colonne/expr [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, annee, titre
```

```
FROM les-films
```

```
ORDER BY auteur ASC, annee ASC;
```

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

16

## Tri des lignes (suite)

```
-- ordre alphabétique inverse des titres
SELECT auteur, annee, titre
FROM les-films
ORDER BY titre DESC;
```

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

17

## Tri + Restriction + Projection

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

titre	annee
City Lights	1931
Modern Times	1936
The Dictator	1940

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

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

18

## Produit cartésien

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

titre	auteur	annee	durée	gid	gid2	quoi
ChizenKane	Wells	1936	01:59:00	1	1	Drame
ChizenKane	Wells	1936	01:59:00	1	2	Comédie
ChizenKane	Wells	1936	01:59:00	1	3	Comédie dramatique
ChizenKane	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

19

## Jointure simple sur un champ

```
-- jointure sur le genre (join on)
SELECT titre, auteur, annee, 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, annee, quoi
FROM les-films AS f, genres AS g
WHERE f.gid=g.gid;
```

titre	auteur	annee	quoi
Chizen 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

20

## Jointure + Restriction + Projection + Tri

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

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

21

## 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.duree=f2.duree
WHERE f1.titre<>f2.titre;
```

film1	film2
Modern Times	City Lights
City Lights	Modern Times

22

## Agrégation de lignes avec répétitions

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

## Regroupement et Agrégation

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

23

24

**Aggrégation des lignes GROUP BY**

**regroupement** d'attributs de valeurs identiques

clause **GROUP BY** *colName/expression*...

**accumulation** des autres valeurs avec une fonction spécifique

**COUNT** ( \* ) comptage occurrences

**COUNT** (DISTINCT ... ) comptage occurrences distinctes

**MAX** **MIN** minimum, maximum

**SUM** **AVG** **VARIANCE** **STDDEV** somme, moyenne, variance...

**extensions** ajout possible de nouvelles agrégations...

25

**Aggrégation d'un attribut**

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

nfilms
6

```
-- année du premier film
```

```
SELECT MIN(annee) AS annee
FROM les_films;
```

annee
1931

26

**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(annee)-MIN(annee) AS activite
FROM les_films
GROUP BY auteur
ORDER BY activite DESC;
```

auteur	activite
Wells	27
Chaplin	9
Ozu	0

27

**Aggrégation (suite)**

```
-- nombre de film et durée moyenne par auteur
```

```
SELECT auteur, COUNT(*) AS nfilms, AVG(duree) AS mduree
FROM les_films
GROUP BY auteur
ORDER BY mduree DESC;
```

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

28

**Aggrégation (encore)**

```
-- nombre de genres différents par auteur
```

```
SELECT auteur, COUNT(DISTINCT qual) 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	ngenes
Chaplin	2
Ozu	1
Wells	1

29

**Aggrégation avec condition HAVING**

— clause **HAVING** condition

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

```
-- auteurs avec plusieurs films
```

```
SELECT auteur
FROM les_films
GROUP BY auteur
HAVING COUNT(*)>1;
```

auteur
Chaplin
Wells

30

**WHERE vs HAVING**

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

— clause **HAVING** condition sur table agrégée en sortie

```
-- auteur et durée moyenne de plusieurs films avant 1940
```

```
SELECT auteur, AVG(duree) AS mduree
FROM les_films
WHERE annee < 1940
GROUP BY auteur
HAVING COUNT(*)>1;
```

auteur	mduree
Chaplin	01:27:00

31

**Aggrégation d'un attribut**

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

nfilms
6

```
-- année du premier film
```

```
SELECT MIN(annee) AS annee
FROM les_films;
```

annee
1931

26

**Aggrégation (suite)**

```
-- nombre de film et durée moyenne par auteur
```

```
SELECT auteur, COUNT(*) AS nfilms, AVG(duree) AS mduree
FROM les_films
GROUP BY auteur
ORDER BY mduree DESC;
```

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

28

**Aggrégation avec condition HAVING**

— clause **HAVING** condition

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

```
-- auteurs avec plusieurs films
```

```
SELECT auteur
FROM les_films
GROUP BY auteur
HAVING COUNT(*)>1;
```

auteur
Chaplin
Wells

30

**Exercices**

- quel jour sommes-nous ?
- quelle heure est-il ?
- les cinéastes dont les noms commencent par A, B ou C
- les titres des films de Hayao Miyazaki
- les titres des films d'action
- les titres des films d'après guerre de Orson Wells, ordonnés 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

32

## Expressions : Types, Constantes, Opérateurs, Fonctions

33

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 1000000000000000000000000000
- opérations unaires + -, binaires + - \* /, modulo %
- comparaisons classiques < > <= >=
- priorités classiques, division entre entiers **entière** !
- fonctions **MOD(3,2)=1** **ABS(-1)=1**

35

Valeur **NULL** pour tous les types

- valeur absorbante ou ignorée par les opérations...
- comparaisons :  $d$  **IS NULL**,  $e$  **IS NOT NULL**
- attention**  $x = \text{NULL}$  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	1	f	f	f	f	f
1	0	f	f	f	f	f	f
0	0	1	f	f	f	f	f
1	1	1	f	f	f	f	f

34

floatants **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**(' hobbies')
- SUBSTRING**('Calvin',2,4)

36

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

37

temps **DATE** **TIME** **TIMETZ** **TIMESTAMP** **INTERVAL**

- constantes '1970-03-20'::**DATE** **TIME** '12:30:45'
- opérateurs arithmétiques + - \* /
- DATE** 'today' + **INTERVAL** '1 month'
- fonctions **NOW**() **CURRENT\_DATE** **CURRENT\_TIME**
- extraction **EXTRACT**(YEAR FROM une.date)
- avec **CENTURY** **DECADE** **DOM** **HOUR** **MINUTE** **SECOND**...
- CFAGE** **JUSTIFYHOURS** **JUSTIFYDAYS** **TOCHAR**...

39

Valeur **NULL** pour tous les types

- valeur absorbante ou ignorée par les opérations...
- comparaisons :  $d$  **IS NULL**,  $e$  **IS NOT NULL**
- attention**  $x = \text{NULL}$  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	1	f	f	f	f	f
1	0	f	f	f	f	f	f
0	0	1	f	f	f	f	f
1	1	1	f	f	f	f	f

34

floatants **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**(' hobbies')
- SUBSTRING**('Calvin',2,4)

36

## -- titres sans deux majuscules

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

auteur	titre
Ozu	Ozu

## -- titres accentués

```
SELECT titre, auteur, annee
FROM les_films
WHERE titre SIMILAR TO '%[ˆa-zA-Z ]%' ;
```

titre	auteur	annee
Ouzô	Ozu	1989
Le Procès	Wells	1983

38

binaires **BIT**(12) **VARBIT**(31) **BYTEA**géométrie **2D** **BOX** **CIRCLE** **LINE** **POINT** **POLYGON** **PATH**...

- constantes '(1,1),(2,3),(0,0)::**POLYGON**
- nombreuses opérations : contenu, 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**argent **MONEY**...ok pour USD

```
SELECT MONEY '10.0' ;
```

money
\$10.00

40

### 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 deces IS NULL THEN 'vivant'
ELSE 'décédé'
END AS "actuel."
FROM personnes;
```

nom	actuel
Chaplin	décédé
Wells	décédé
Ozu	décédé
Allen	vivant
Carré	décédé
Tarentho	vivant

41

### 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 "hé au"
FROM personnes;
```

nom	hé au
Chaplin	XXème
Wells	XXème
Ozu	XXème
Allen	XXème
Carré	XXème
Tarentho	XXème

42

### 3 types de jointures

**cartésienne** (*cross join*), rarement utile!

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

chaque tuple apparait **si correspondance**

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

3 sortes **LEFT / RIGHT / FULL**

tuples apparaissent **même sans correspondance!**

contient la jointure interne

prenom	id	nom
Albert	1	Einstein
Allen	2	Camus
Kurt	3	Gödel

id	nom
1	Einstein
1	Camus
3	Gödel
4	Church

43

### Jointure cartésienne CROSS JOIN 2 syntaxes !

```
SELECT p.prenom, n.nom
FROM prenom AS p CROSS JOIN noms AS n;
```

```
SELECT p.prenom, n.nom
FROM prenom AS p, noms AS n;
```

prenom	nom
Albert	Einstein
Albert	Camus
Albert	Gödel
Albert	Church
Allen	Einstein
Allen	Camus
Allen	Gödel
Allen	Church
Kurt	Einstein
Kurt	Camus
Kurt	Gödel
Kurt	Church

44

### Jointure interne [INNER] JOIN 4 syntaxes !

```
SELECT p.prenom, n.nom -- on condition
FROM prenom AS p INNER JOIN noms AS n ON p.id=n.id;
```

```
SELECT p.prenom, n.nom -- using colname
FROM prenom AS p INNER JOIN noms AS n USING (id);
```

```
SELECT p.prenom, n.nom -- natural = même colname
FROM prenom AS p NATURAL INNER JOIN noms AS n;
```

```
SELECT p.prenom, n.nom -- X et where condition
FROM prenom AS p, noms AS n
WHERE p.id=n.id;
```

prenom	nom
Albert	Einstein
Albert	Camus
Kurt	Gödel

45

### 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.prenom, n.nom
FROM prenom AS p NATURAL LEFT OUTER JOIN noms AS n;
```

prenom	nom
Albert	Einstein
Albert	Camus
Allen	
Kurt	Gödel

46

### Jointure externe droite RIGHT [OUTER] JOIN

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

– ajout d'un tuple **gauche NULL** si nécessaire

```
SELECT p.prenom, n.nom
FROM prenom AS p RIGHT JOIN noms AS n USING (id);
```

prenom	nom
Albert	Einstein
Albert	Camus
Kurt	Gödel
Church	

47

### Jointure externe totale FULL [OUTER] JOIN

– tous les tuples apparaissent, ajout tuples **NULL**

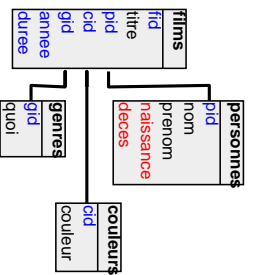
```
SELECT p.prenom, n.nom
FROM prenom AS p FULL JOIN noms AS n ON p.id=n.id;
```

prenom	nom
Albert	Einstein
Albert	Camus
Allen	
Kurt	Gödel
Church	

48

### Faisons notre cinema avec 4 tables

- modèle normalisé, pas de redondance



49

<i>fid</i>	<i>titre</i>	<i>pid</i>	<i>cid</i>	<i>genre</i>	<i>annee</i>	<i>duree</i>
1	Citizen Kane	2	1	1	1936	01:53:00
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	1989	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	Limeight	1	1	3	1952	02:21:00
10	King in New York	1	1	3	1957	01:50:00
11	Alice	4	2	2	1980	01:42:00

51

### 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;

SELECT id FROM prenom
INTERSECT
SELECT id FROM noms;

```

53

### 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...;

-- expressions booléennes, opérateurs EXISTS IN ANY ALL
sous-requête SELECT simplifié (pas de ORDER...)
SELECT ... FROM ...
WHERE attr NOT IN
(SELECT ... FROM ... WHERE ...);

```

55

| <i>pid</i> | <i>nom</i> | <i>prenom</i> | <i>naissance</i> | <i>deces</i> |
|------------|------------|---------------|------------------|--------------|
| 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       |              |

| <i>genre</i> | <i>quoi</i>        | <i>cid</i> | <i>couleur</i> |
|--------------|--------------------|------------|----------------|
| 1            | Drame              | 1          | N&B            |
| 2            | Comédie            | 2          | couleur        |
| 3            | Comédie dramatique |            |                |
| 4            | Action             |            |                |

50

### Jointure externe pour passage au complémentaire

```

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

```

| <i>prenom</i> | <i>nom</i> |
|---------------|------------|
| Marcel        | Carné      |

52

### Exemple avec EXCEPT

```

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

```

| <i>nom</i> | <i>prenom</i> |
|------------|---------------|
| Carné      | Marcel        |

54

### Sous-requête dans un FROM

```

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

```

| <i>duree</i> |
|--------------|
| 01:37:00     |

56

**Sous-requête dans une condition WHERE**

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

| titre            |
|------------------|
| Modern Times     |
| City Lights      |
| Okay!            |
| Kill Bill v1     |
| King in New York |
| Alice            |

57

**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;
```

58

**Plus simple avec UNION et jointure externe**

```
-- auteurs avec films, jointure interne
SELECT nom, COUNT(*) AS nfilms
FROM personnes NATURAL JOIN films
GROUP BY nom

-- auteurs sans films, jointure externe
SELECT nom, 0
FROM personnes NATURAL LEFT JOIN films
WHERE fid IS NULL

-- ordre commun
ORDER BY nfilms DESC, nom ASC;
```

| nom      | nfilms |
|----------|--------|
| Chaplin  | 6      |
| Wells    | 2      |
| Allen    | 1      |
| Ozu      | 1      |
| Tarenino | 1      |
| Carré    | 0      |

59

**Astuce spécifique PostgresSQL**

```
-- 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 |
|----------|-------|
| Carré    | 0     |
| Chaplin  | 6     |
| Ozu      | 1     |
| Allen    | 1     |
| Wells    | 2     |
| Tarenino | 1     |

60

**Conseils pour le développement des requêtes**

- déterminer les tables utiles
  - résultats ou conditions
  - tables de liaisons
  - tables multiples ?
- créer le **SELECT FROM** et les jointures **ON USING WHERE**
- ajouter les conditions **WHERE**
- ajouter les agrégations **GROUP BY** et condition sur agrégation **HAVING**
- ajouter les tris **ORDER BY**
- décomposer si nécessaire (sous requêtes, ensembles) ?
- simplifier les requêtes (multiples...) avec des vues ?

61

**Exercices**

- 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 long

62

**Réfléchir au problème suivant**

- soient deux tables de même schéma (*id INT PK, val TEXT NV*)
- 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    |

63

**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, ...

64

**Insertions d'une ligne INSERT**

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

```
-- sauf 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());
```

65

**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 ... ;
```

66

**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 matTable**
- WHERE id=123;**
- **effacement complet**
- DELETE FROM matTable;**

67

**Mise à jour de lignes UPDATE**

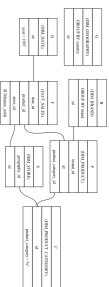
- commande **UPDATE ... SET ... =... WHERE ...**
- modifications d'un ou plusieurs attributs
- condition similaire à un **SELECT**
- peut toucher plusieurs lignes
- **corrigeons...**
- UPDATE matTable**
- SET prenom='Calvin', copain='Hobbes'**
- WHERE age=6;**

68

**Visualisation de requêtes complexes**

<http://revj.sourcelforge.net/>

```
SELECT B.Brand, G.Country, SUM (P.UnitsSold)
FROM Fact.Sales F
JOIN Dim.Date D ON F.DateId = D.Id
JOIN Dim.Store S ON F.StoreId = S.Id
JOIN Dim.Geography G ON S.GeographyId = G.Id
JOIN Dim.Product P ON F.ProductId = P.Id
JOIN Dim.Brand B ON P.BrandId = B.Id
JOIN Dim.Brand G ON C.ProductCategory = 'tv'
GROUP BY B.Brand, G.Country;
```



69

**List of Slides**

- 1 SQL DML : Data Manipulation Language
- 2 Opérations élémentaires sur les données
- 3 Faisons notre cinéma avec 2 tables
- 4 La commande **SELECT**
- 5 Syntaxe de **SELECT**
- 6 Lien avec l'algorithme relationnelle
- 7 Expressions : Types, Constantes, Opérateurs, Fonctions
- 8 Projection de colonnes **SELECT ... FROM ...**
- 9 Projection de colonnes (suite)
- 10 Projection de toutes les colonnes

11 Projection + expressions + renommage

12 Restriction de lignes **WHERE**

13 Restriction exprimées sur l'entée

14 Restriction du nombre de lignes **LIMIT/OFFSET**15 Suppression des doublons ? **DISTINCT vs ALL**16 Tri des lignes **ORDER BY colnameexpr [ASC/DESC] ...**

17 Tri des lignes (suite)

18 Tri + Restriction + Projection

19 Produit cartésien

20 Jointure simple sur un champ

21 Jointure + Restriction + Projection + Tri

22 Auto-jointure

23 Aggrégation de lignes avec répétitions

24 Regroupement et Aggrégation

25 Aggrégation des lignes **GROUP BY**

26 Aggrégation d'un attribut

27 Aggrégation

28 Aggrégation (suite)

29 Aggrégation (encore)

30 Aggrégation avec condition **HAVING**31 **WHERE vs HAVING**

32 Exercices

33 Expressions : Types, Constantes, Opérateurs, Fonctions

34 Valeur **NULL** pour tous les types37 Opérateurs **LIKE / SIMILAR TO / ~**41 Expression conditionnelle **CASE (1)**

- 42 Expression conditionnelle **CASE** (2)
- 43 3 types de jointures
- 44 Jointure cartésienne **CROSS JOIN** 2 syntaxes!
- 45 Jointure interne **INNER JOIN** 4 syntaxes!
- 46 Jointure externe gauche **LEFT [OUTER] JOIN**
- 47 Jointure externe droite **RIGHT [OUTER] JOIN**
- 48 Jointure externe totale **FULL [OUTER] JOIN**
- 49 Faisons notre cinéma avec 4 tables
- 52 Jointure externe pour passage au complémentaire
- 53 Opérations ensemblistes
- 54 Exemple avec **EXCEPT**
- 55 Sous requêtes (subqueries)
- 56 Sous-requête dans un **FROM**

- 57 Sous-requête dans une condition **WHERE**
- 58 Sous-requête + **UNION**
- 59 Plus simple avec **UNION** et jointure externe
- 60 Astuce spécifique PostgreSQL
- 61 Conseils pour le développement des requêtes
- 62 Exercices
- 63 Réfléchir au problème suivant
- 64 Requetes graphiques
- 65 Insertions d'une ligne **INSERT**
- 66 Insertion de lignes **INSERT . . . SELECT**
- 67 Effacement de lignes **DELETE**
- 68 Mise à jour de lignes **UPDATE**
- 69 Visualisation de requêtes complexes