Intégration de Données

- SQL -

M2 MIAGE IPM

jiefu.song@ut-capitole.fr

UNIVERSITÉ TOULOUSE CAPITOLE

FACULTÉ D'INFORMATIQUE

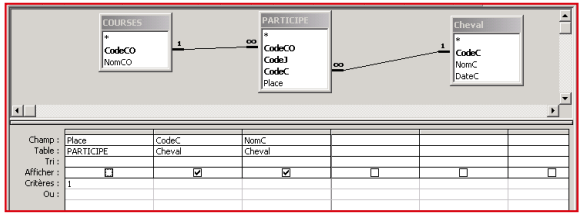
Membre de l'Université Toulouse Capitole

Plan

- **SQL dans le contexte de l'intégration de données**
- **SQL avancé**

CONTEXTE

- **Langages de manipulation de données : 2 catégories**
  - Langages procéduraux : enchaînement d'opérations élémentaires
    - langage algébrique :
      - Avantage : Langage de référence sur lequel s'appuient les autres
      - Inconvénient : connaissance des op. algébriques et des enchaînements
    - Langage graphique type QBE
      - Avantages : facile d'accès et non connaissance du schéma de la BD
      - Inconvénients : Langage spécifique à un éditeur et limité



  - Langages assertionnels (SQL): description du résultat désiré

CONTEXTE

- **SQL : Structured Query Langage**
  - Version commerciale de SEQUEL (IBM 74) : projet de langage de manipulation de BD pour non-informaticiens
  - Langage assertionnel Standard
    - Langage de Définition des Données (LDD)
      - Définition des tables : CREATE TABLE, ALTER TABLE, DROP TABLE
      - Définition de vues externes : CREATE VIEW
      - Définition de contraintes d'intégrité : CONSTRAINT
    - Langage de Manipulation des Données (LMD)
      - Mise à jour des données : INSERT, DELETE, UPDATE
      - Consultation des données : SELECT
    - Langage de Contrôle des Données (LCD)
      - Contrôle de la sécurité et des accès aux données : GRANT, REVOKE

CONTEXTE

- **SQL v.s. intégration de données**
  - Intégration virtuelle de deux BD relationnelles

```
CREATE VIEW view_name AS
    SELECT column1, column2, ...
    FROM nomSchema1.table_name_DB1, nomSchema2.table_name_DB2
    WHERE condition;
```

  - Intégration physique de deux BD relationnelles

```
CREATE TABLE new_table_name AS
     SELECT column1, column2,...
     FROM nomSchema.table_name_DB1, nomSchema.table_name_DB2
     WHERE condition;
```

CONTEXTE

- **SQL v.s. intégration de données**
  - Vue matérialisée

```
CREATE MATERIALIZED VIEW <nomvue>
BUILD { IMMEDIATE|DEFERRED }
REFRESH { COMPLETE|FAST|FORCE|NEVER } { ON DEMAND|ON COMMIT }
AS SELECT … ;
```

  - Options
    - IMMEDIATE : Création de la vue matérialisée et population de la vue
    - DEFERRED : Création de la vue matérialisée sans être alimentée en données. DBMS_MVIEW.REFRESH(<liste_vues>) alimente la vue
    - ON COMMIT : Rafraîchissement à chaque fin de transaction modifiant les tables sources
    - ON DEMAND : Rafraîchissement avec DBMS_MVIEW.REFRESH

CONTEXTE

- **SQL v.s. intégration de données**
  - Vue matérialisée

```
CREATE MATERIALIZED VIEW <nomvue>
BUILD { IMMEDIATE|DEFERRED }
REFRESH { COMPLETE|FAST|FORCE|NEVER } { ON DEMAND|ON COMMIT }
AS SELECT … ;
```

  - Options
    - COMPLETE : Recalcul complet de la vue
    - FAST : Application d'un rafraîchissement incrémental
    - FORCE : FAST si possible, COMPLETE sinon
    - NEVER : pas de rafraîchissement

SQL AVANCÉ

Un peu de SQL, cela nous fera du bien…

Membre de l'Université
Toulouse Capitole

COURS	CodeCO	NomCO	Domaine
	C1	Bases de Données Relationnelles	BD
	C2	Bases de données avancées	BD
	C3	Bases de données multidimensionnelles	
	C4	Système d'information décisionnel	Décisionnel
	C5	Programmation objet JAVA	Prog

ENSEIGNANT	CodeEN	NomEN	PrenomEN	RueEN	CPEN	VilleEN	TelEN
	E1	Ravat	Franck	Rue Ban	31400	Toulouse	05.61.63.36.33
	E2	Thevenin	Jean-Marc	Rue Elle	31520	Ramonville	
	E3	Pierre	Paul-Jacques	Rue Gir	81100	Castres	05.63.63.35.21
	E4	Peuplus	Jean	Rue Bis	31100	Toulouse	
	E5	Bosse	Elle	Rue Bis	31820	Pibrac	

AFFECTER	CodeEN#	CodeCO#	Duree
	E1	C1	10
	E4	C1	12
	E2	C2	24
	E4	C3	15
	E5	C3	9
	E2	C3	10

EFFECTUER	CodeEN#	CodeCO#	DateDeb	HeureDeb	Duree
	E1	C1	02/10/08	08:00	4
	E1	C1	03/10/08	08:00	4
	E1	C1	04/10/08	08:00	2
	E4	C1	05/10/08	14:00	4
	E4	C1	06/10/08	14:00	4
	E4	C1	07/10/08	14:00	4
	E2	C2	05/01/09	08:00	8
	E2	C2	07/01/09	08:00	8
	E2	C2	09/01/09	08:00	8
	E2	C2	13/01/09	08:00	8
	E5	C3	20/03/09	09:30	3
	E5	C3	24/03/09	14:00	3
	E5	C3	28/03/09	09:30	3

FORMAT	CodeFO	NomFO	CodeEN#
	F1	M2 IGSI	E1
	F2	M2 SIAD	E2
	F3	M2 Finance FTI	E3
	F4	M2 Finance FE	E3
	F5	M1SIIO	E5

CASE

```
SELECT
  en.nomEn,
  CASE WHEN LEFT(en.cpen, 2) = '31' THEN 'Haute Garonne'
     WHEN LEFT(en.cpen, 2) = '81' THEN 'Tarn'
  ELSE 'autres' END
FROM ENSEIGNANTS en
```

Affichage alternatif : CASE

- Syntaxe

```
CASE when Sexe= 'M' then 'Masculin' else 'Feminin' end
CASE when AttX < y then 'a' when AttX = y then 'b' else 'c' end
CASE Note when '0' then 'nul' else 'mieux' end
```

- Exemple

```
SELECT status,
        CASE WHEN STATUS IN('a1','a2','a3') THEN 'Active'
                    WHEN STATUS = 'i' THEN 'Inactive'
                    WHEN STATUS = 't' THEN 'Terminated'
         END AS STATUSTEXT
FROM STATUS
```

- Application : Nom des enseignants avec le nom du département (Haute Garonne -31-, Tarn -81-, autres)

IN/NOT IN

- **Vérifie si la valeur d'un attribut appartient ou n'appartient pas à un ensemble de valeurs**

```
select codeco
from affecter
where duree in (9, 24);
```

```
select codeco
from affecter
where duree = 9
or duree = 24;
```

```
CODEC
-----
C2
C3
```

ANY/ALL

- **ALL : comparer la valeur d'un attribut à TOUTES LES VALEURS d'un ensemble**
- **ANY :comparer la valeur d'un attribut à UNE VALEUR d'un ensemble**

```
AttX op ALL (val1, Val2…) OU AttX op ANY(val1, Val2…)
```

Opérateurs classiques : <, >, <=, >=, <> ou !=

```
select codeco
from affecter
where duree > (9, 24);
```

```
message d'erreur
```

ANY/ALL

```
select codeco, duree
from affecter
where duree >= ALL (9, 24);
```

CODEC	DUREE
C2	24

```
select codeco, duree
from affecter
where duree >= 9
and duree >= 24;
```

Sémantique ?

```
select codeco, duree
from affecter
where duree >= ANY (9, 24);
```

CODEC	DUREE
C1	10
C1	12
C2	24
C3	15
C3	9
C3	10

```
select codeco, duree
from affecter
where duree >= 9
or duree >= 24;
```

ORDER BY

- Gestion valeurs nulles (**NULLS LAST** ou **NULLS FIRST)**

```
select * from cours
order by domaine desc NULLS LAST, nomco asc;
```

CODEC	NOMCO	DOMAINE
C5	Programmation Objet JAVA	Prog
C4	Système d'information décisionnel	Décisionnel
C2	Bases de données avancées	BD
C1	Bases de données relationnelles	BD
C3	Bases de données multidimensionnelles	

```
select * from cours
order by domaine desc NULLS first, nomco asc;
```

CODEC	NOMCO	DOMAINE
C3	Bases de données multidimensionnelles	
C5	Programmation Objet JAVA	Prog
C4	Système d'information décisionnel	Décisionnel
C2	Bases de données avancées	BD
C1	Bases de données relationnelles	BD

JOINTURE

- **Jointure naturelle**

Ecriture Relationnelle	Ecriture SQL 2
Select E.NomEN, E.PrenomEN **From** ENSEIGNANT E, FORMAT F **Where** E.CodeEN=F.CodeEN **And** F.NomFO = 'M2 Miage IPM'	**Select** E.NomEN, E.PrenomEN **From** ENSEIGNANT E **Join** FORMAT F **on** E.CodeEN=F.CodeEN **Where** F.NomFO = 'M2 Miage IPM' **Select** E.NomEN, E.PrenomEN **From** ENSEIGNANT E **Inner Join** FORMAT F **on** E.CodeEN=F.CodeEN **Where** F.NomFO = 'M2 Miage IPM' **Select** E.NomEN, E.PrenomEN **From** ENSEIGNANT E **Inner Join** FORMAT F **Using** CodeEN **Where** F.NomFO = 'M2 Miage IPM'
natural : indique que la jointure se fait sur l'égalité des colonnes qui portent le même nom dans les deux tables	**Select** E.NomEN, E.PrenomEN **From** ENSEIGNANT E **Natural Inner Join** FORMAT F **Where** F.NomFO = 'M2 Miage IPM'

JOINTURE

- **Inéqui-jointure**

Nom de l'enseignant ayant plus d'heures de cours que E4 pour le cours C1

Ecriture Relationnelle

Première écriture : auto-jointure et inéqui-jointure

```
Select E.NomEN, E.PrenomEN
From   ENSEIGNANT E,
       AFFECTER A1,
       AFFECTER A2
Where E.CodeEN=A1.CodeEN
And A1.CodeCO = 'C1'
And A1.Duree > A2.Duree
And A2.CodeEN = 'E4'
And A2.CodeCO = 'C1'
```

Deuxième écriture : sous-requête et inéqui-jointure

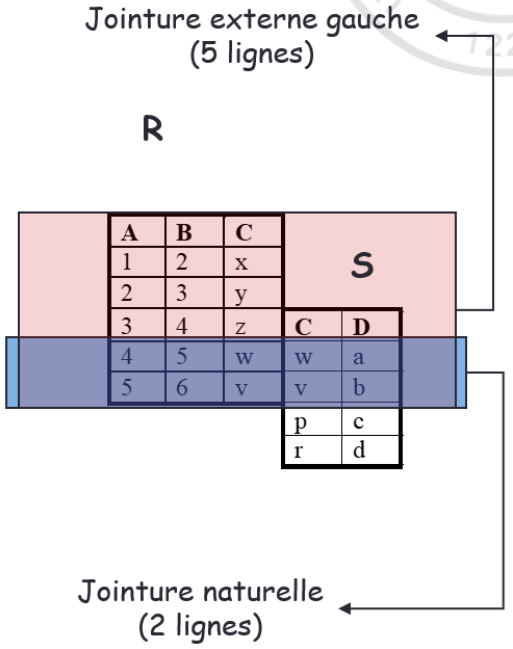
```
Select E.NomEN, E.PrenomEN
From   ENSEIGNANT E,
       AFFECTER A,
Where E.CodeEN=A.CodeEN
And A.CodeCO = 'C1'
And A.Duree >
    (Select A2.Duree
     From AFFECTER A2
     And A2.CodeCO = 'C1'
     And A2.CodeEN = 'E4')
```

Ecriture SQL 2

```
Select E.NomEN, E.PrenomEN
From ENSEIGNANT E
Join AFFECTER A1 on A1.CodeEN=F.CodeEN
Join AFFECTER A2 on A1.Duree > A2.Duree
Where A2.CodeEN = 'E4'
And A1.CodeCO = 'C1'
And A2.CodeCO = 'C1'
```

JOINTURE

- **Jointures externes (outer join)**
  - Principes
    - Extension de la jointure naturelle faisant intervenir une table dominante et table subordonnée
    - Permet d'obtenir en plus des lignes répondant à la condition de jointure les lignes de la table dominante qui n'y répondent pas
  - Syntaxe
    - Ecriture relationnelle : Ajouter (+) dans la condition de jointure derrière l'attribut de la table dont les lignes ne peuvent apparaître dans le résultat (table subordonnée)
    - Ecriture SQL2 : commandes LEFT OUTER JOIN ou RIGHT OUTER JOIN dans le FROM



JOINTURE

- **Jointures externes (outer join)**
  - Externe gauche : retourne tous les tuples de A (table dominante)

Ecriture Relationnelle	Ecriture SQL 2
Select … From A, B Where A.x = B.y (+)	**Select** … From A LEFT OUTER JOIN B On A.x = B.y

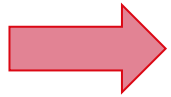
  - Externe droite : retourne tous les tuples de B (table dominante)

Ecriture Relationnelle	Ecriture SQL 2
Select … From A, B Where A.x (+)= B.y	**Select** … From A RIGHT OUTER JOIN B On A.x = B.y

JOINTURE

- **Jointures externes (outer join)**

```
Select EN.NomEN, F.NomFO
From Enseignant EN, Format F
Where EN.CodeEN = F.CodeEN(+)
And F.NomFO like 'M2%';
```



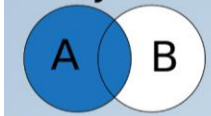
```
NomEN      NomFO
---------  ---------------
Ravat      M2IGSI
Thevenin   M2 SIAD
Pierre     M2 Finance FTI
Pierre     M2 Finance FE
Peuplus
Bosse
```

Sémantique ?

JOINTURE

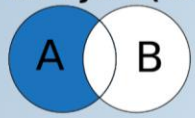
- **les jointures en SQL 2**

LEFT JOIN



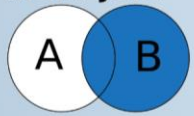

```
SELECT *
FROM A
LEFT JOIN B ON A.key = B.key
```

LEFT JOIN (sans l'intersection de B)



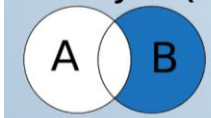

```
SELECT *
FROM A
LEFT JOIN B ON A.key = B.key
WHERE B.key IS NULL
```

RIGHT JOIN



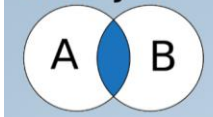

```
SELECT *
FROM A
RIGHT JOIN B ON A.key = B.key
```

RIGHT JOIN (sans l'intersection de A)



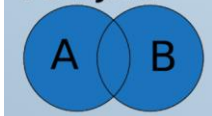

```
SELECT *
FROM A
RIGHT JOIN B ON A.key = B.key
WHERE B.key IS NULL
```

INNER JOIN



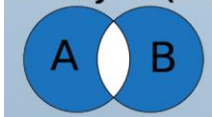

```
SELECT *
FROM A
INNER JOIN B ON A.key = B.key
```

FULL JOIN




```
SELECT *
FROM A
FULL JOIN B ON A.key = B.key
```

FULL JOIN (sans intersection)




```
SELECT *
FROM A
FULL JOIN B ON A.key = B.key
WHERE A.key IS NULL
OR B.key IS NULL
```

SOUS-REQUETE

- **Sous-Requête avec un opérateur =, >,<, …**

```
select a1.codeco
from affecter a1
where a1.duree >
     (select max(a2.duree)
      from affecter a2
      where a2.codeco = 'C1');
```

```
select a1.codeco
from affecter a1
where a1.duree > all
     (select a2.duree
      from affecter a2
      where a2.codeco = 'C1');
```

EXISTS/NOT EXISTS

- **Syntaxe**

```
SELECT   C1, C2, … Cn
FROM      R
WHERE   Condition_sélection1
AND      [NOT]  EXISTS          (SELECT   Cj
                                FROM      R2
                                WHERE   Condition_sélection2
                                AND       Condition_Jointure);
```

EXISTS/NOT EXISTS

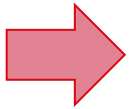
```
select C.nomco
from cours c
where not exists
      (select *
       from affecter a
       );
```

Résultat : aucune ligne

Pourquoi?

```
select C.nomco
from cours c
where c.codeco not in
     (select a.codeco
      from affecter a);

--
select C.nomco
from cours c
where not exists
     (select *
      from affecter a
      where a.codeco = c.codeco);
```



```
Programmation Objet JAVA
Système d'information décisionnel
```

Sémantique de la requête?

EXISTS/NOT EXISTS

- **Division** Utilisé pour les requêtes Tous / Toutes

```
select c.nomco
from cours c
where NOT EXISTS
     (select *
      from enseignant e
      where NOT EXISTS
           (select *
            from affecter a
            where a.codeen = e.codeen
            and a.codeco = c.codeco));
```

Sémantique?

FONCTIONS DE CALCUL

- ABS(n) : Valeur absolue de n
- CEIL(n) : Plus petit entier ≥ n
- FLOOR(n) : Plus grand entier ≤ n
- MOD(m, n) : Reste de m/n
- POWER(m, n) : m^n
- SIGN(n) : Signe de n
- SQRT(n) : Racine carrée de n
- ROUND(n, m) : Arrondi à 10^{-m}
- TRUNC(n, m) : Troncature à 10^{-m}
- CHR(n) : Caractère ASCII n°n
- INITCAP(ch) : 1ère lettre en maj.

- LOWER(ch) : c en minuscules
- UPPER(ch) : c en majuscules
- LTRIM(ch, n) : Troncature à gauche
- RTRIM(ch, n) : Troncature à droite
- REPLACE(ch, car) : Remplacement de caractère
- SUBSTR(ch, pos, lg) : Extraction de chaîne
- SOUNDEX(ch) : Cp. Phonétique
- LPAD(ch, lg, car) : Compléter à gauche
- RPAD(ch, lg, car) : Compléter à droite

FONCTIONS DE CALCUL

- ASCII(ch) : Valeur ASCII de ch
- INSTR(ch, ssch) : Recherche de ssch dans ch
- LENGTH(ch) : Longueur de ch
- ADD_MONTHS(dte, n) : Ajout de n mois à dte
- LAST_DAY(dte) : Dernier jour du mois
- MONTHS_BETWEEN(dt1, dt2) : Nombre de mois entre dt1 et dt2
- NEXT_DAY(dte) : Date du lendemain
- SYSDATE : Date/heure système

- TO_NUMBER(ch) : Conversion de ch en nombre
- TO_CHAR(x) : Conversion de x en chaîne
- TO_DATE(ch) : Conversion de ch en date
- NVL(x, val) : Remplace par val si x a la valeur NULL
- GREATEST(n1, n2…) : + grand
- LEAST (n1, n2…) : + petit
- UID : Identifiant numérique de l'utilisateur
- USER : Nom de l'utilisateur