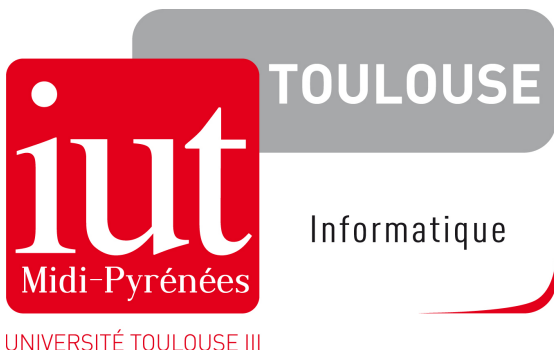
M2106 : Programmation et administration des bases de données

Cours 3/6 – LID

Guillaume Cabanac

guillaume.cabanac@univ-tlse3.fr



version du 9 octobre 2013

LID : Langage d'interrogation des données

1. Projection π
2. Sélection σ
3. Opérateurs ensemblistes $\cup$, $\in$, $\cap$, $\setminus$ et $\exists$
4. Groupement
5. Produit $\times$
6. Jointures $\bowtie$, $\bowtie_\theta$, $\rtimes$, $\ltimes$ et $\ltimes$

Syntaxe simplifiée des requêtes SQL

```
-- Un résultat est une table, un select, une vue
-- ou l'application d'un opérateur ensembliste à deux résultats
Res := {Tab | Sel | Vue | Res1 {union [all] | intersect | minus} Res2}


-- Définition d'un résultat de type select (notion d'algèbre fermée)
Sel := select col1 [[as] aliasCol1], ..., fonc1 [[as] aliasColN], ...
       from Res1 [[as] aliasRes1], ... , ResN [[as] aliasResN]
       [where Cond1 {and | or} ... {and | or} CondN]
       [
         group by col1, ... , colN
         [having cond1 {and | or} ... {and | or} condN]
       ]
       [order by col1 [{[asc] | desc} [nulls {first | last}]], ...


-- Définition d'une condition booléenne
Cond := {    Oper1 [(+)] {= | <> | {^ | ! | < | >} =} Oper2 [(+)]
           | Oper {= | < | >} {all | any | some} Res
           | Oper is [not] null
           | Oper between borneMin and borneMax
           | Oper [not] like 'motif'
           | Oper [not] in Res          -- Opérateur '∈' en théorie des ensembles
           | [not] exists Res           -- Opérateur '∃' en logique
           | [not] Cond                 -- Opérateur '¬' en logique
       }


Légende :
           [X] l'élément X est optionnel
   {A | B | C} un des éléments A, B ou C est obligatoire
```

Base de données exemple

```
select * from adherent order by idA ;
```

IDA	NOM	PRENOM	VILLE	MAIL
1	Eastwood	Clint	Troyes	clint@eastwood.com
45	Vladuboudin	Tintin	Bruxelles	
69	Morgane	Clara	Marseille	clara@hotmail.fr
999	Satan	Michel	Las Vegas	

```
select * from chien order by tatouage ;
```

TATOUAGE	NOM	DDN	SEXE	IDA	IDR
3	Lassie	04-MAY-10	F	999	CO
42	Rantanplan	01-MAY-09	M	45	
51	Milou	15-FEB-99	M	45	CA
666	Cerbère	08-MAR-11	M	999	CA
1664	Iench	12-DEC-12	F	69	

LID : Langage d'interrogation des données

Projection d'une relation : $\pi_{att_1,...,att_N} R$

π Projection

La projection d'une relation R permet de n'afficher qu'un sous-ensemble des attributs de la relation.

```
-- Les colonnes projetées sont 'prenom' et 'nom'
select prenom, nom
from adherent ;

PRENOM               NOM
-------------------- --------------------
Clint                Eastwood
Clara                Morgane
Michel               Satan
Tintin               Vladuboudin
```

LID : Langage d'interrogation des données

Sélection dans une relation : $\sigma_{cond_1,...,cond_N} R$

σ Sélection

La sélection d'une relation R permet de n'afficher que les tuples de la relation correspondant aux conditions booléennes spécifiées.

```
-- Les lignes sélectionnées contiennent toutes les colonnes
-- de la table adherent.
select *
from adherent
where idA = 69
   or idA = 999 ;

 IDA NOM         PRENOM      VILLE          MAIL
---- ---------- ----------- -------------- ----------------
 69 Morgane     Clara       Marseille      clara@hotmail.fr
999 Satan       Michel      Las Vegas
```

Sélection dans une relation : $\sigma_{cond_1,...,cond_N} R$

Chaînes, nombres et dates

Les opérateurs de sélection (σ) :

`a = b`	Égalité entre `a` et `b`.
`a <> b`	Différence entre `a` et `b`.
`a != b`	〃
`a ^= b`	〃
`a > b`	Supériorité de `a` par rapport à `b`.
`a >= b`	Supériorité ou égalité de `a` par rapport à `b`.
`a < b`	Infériorité de `a` par rapport à `b`.
`a <= b`	Infériorité ou égalité de `a` par rapport à `b`.

Sélection dans une relation : $\sigma_{cond_1,...,cond_N} R$

Ensemble de valeurs

Les opérateurs de sélection (σ) sur opérande ensembliste :

`a in (X)`	Présence de la valeur `a` dans l'ensemble `X`.
`a = any (X)`	〃
`a = some (X)`	〃
`a not in (X)`	Absence de la valeur `a` dans l'ensemble `X`.
`a <> all (X)`	〃
`a > all (X)`	La valeur `a` est supérieure aux valeurs de `X`.
`a >= all (X)`	La valeur `a` est supérieure ou égale aux valeurs de `X`.
`.....`	De nombreuses autres combinaisons sont possibles...

NB : avec `X = x1, ... , xN`

Sélection dans une relation : $\sigma_{cond_1,...,cond_N} R$

Cas particulier des valeurs `null`

Les opérateurs de sélection (σ) :

`a is null` Nullité de `a`.

`a is not null` Non-nullité de `a`.

⚠ L'expression « **`null`** = **`null`** » est évaluée à faux !

```
select * from adherent where mail = null ;
-- no rows selected
```

```
... where mail is not null
2 rows selected
```

Projection et sélection dans une relation

Figure 7–1 Projection and Selection

Table DEPARTMENTS

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
60	IT	103	1400
90	Executive	100	1700

Projection
Selection

```
SELECT email,department_name
FROM employees JOIN
departments
ON employees.department_id =
departments.department_id
WHERE employee_id IN (100,103)
ORDER BY email
```

Table EMPLOYEES

EMPLOYEE_ID	LAST_NAME	EMAIL	HIRE_DATE	JOB_ID	MANAGER_ID	DEPARTMENT_ID
100	King	SKING		AD_PRES		90
101	Kochhar	NKOCHHAR	21-SEP-89	AD_VP	100	90
102	De Hann	LDEHANN	13-JAN-93	AD_VP	100	90
103	Hunold	AHUNOLD		IT_PROG	102	60

Illustration de la projection (π) et de la sélection (σ) en SQL

Tri des résultats

Clause `order by`

- Les résultats peuvent être ordonnés selon plusieurs clés de tri : nom ou position des colonnes.
- Deux sens de tri sont offerts : croissant (par défaut) ou décroissant.
- La position des valeurs nulles est également spécifiable.

```
-- Affichage des caractéristiques des chiens, ces derniers étant ordonnés par :
--   1. idR croissant en affichant les valeurs nulles en haut de la liste,
--   2. les lignes ex æquo sont départagées sur le champ nom (croissant),
--   3. les lignes ex æquo sont départagées sur le champ ddn (décroissant).
select idR race, sexe, nom, tatouage, ddn "date de naissance"
from chien
order by idR nulls first, nom, ddn desc ;
```

```
-- Requête équivalent avec une notation positionnée
select idR race, sexe, nom, tatouage, ddn "date de naissance"
from chien
order by 1 nulls first, 3, 5 desc ;
```

Élimination des doublons

Opérateur `distinct`

L'opérateur `distinct` « dédoublonne » le résultat :
il élimine les lignes en double dans le résultat.

```
select idR
from chien
order by 1 nulls first ;

IDR
---


CA
CA
CO
---
```

```
select distinct idR
from chien
order by 1 nulls first ;

IDR
---

CA
CO
---
```

Composition de requêtes : $\pi_{att_1,...,att_N}(\sigma_{cond_1,...,cond_N} R)$

$\pi(\sigma)$ Sélection et projection

Le résultat de toute requête peut être à l'entrée d'une autre requête. On parle d'algèbre fermée : toute opération sur un résultat produit un nouveau résultat, qui est manipulable à son tour.

Parallèle avec les mathématiques : $cos\left(\frac{3\pi}{4}\right) + \left(\sqrt[15]{42} \times 51!\right)$

```
-- Cette requête SQL spécifie 1 projection et 1 sélection
exécution
        4.    select prenom, nom
        1.      from adherent
        2.     where idA = 69
        3.        or idA = 999 ;

              NOM          PRENOM
              ---------- -----------
              Morgane     Clara
              Satan       Michel
```

Composition de requêtes : les sous-requêtes

Généralisation de la composition de requêtes

Le résultat d'une sous-requête peut être à l'entrée d'un opérateur dans la requête englobante.

```
-- Speed dating pour chiens :
-- Adhérents disposant d'un mail, possédant une chienne et habitant
-- dans la même ville que les adhérents possédant un chien.
select nom, prenom
from adherent
where idA in (select idA          --> possesseurs de chiennes
              from chien
              where sexe = 'F')
  and ville in (select ville      --> ville des possesseurs de chiens
                from adherent
                where idA in (select idA  --> possesseurs de chiens
                              from chien
                              where sexe = 'M'))
  and mail is not null  --> possesseurs de chiennes avec email
order by 1, 2 ;
```

Composition de requêtes : les sous-requêtes

Exemples de sous-requêtes non-synchronisées

```
-- Chien le plus âgé ?
select * from chien where ddn >= all(select ddn from chien) ;

--   TATOUAGE NOM        DDN       SEXE        IDA IDR
-- ---------- ---------- --------- ---- ---------- ---
--          51 Milou      15-FEB-99 M            45 CA
```

```
-- Prime la plus faible ?
select prime
from participation
where prime <= all(select prime from participation) ;

-- PRIME
-- --------
--    1.00
```

```
-- Chiens de même sexe que Milou
select *
from chien
where sexe = (select sexe
              from chien
              where tatouage = 51)
and tatouage ^= 51 ;   -- ^= est équivalent à <> et !=

--   TATOUAGE NOM        DDN       SEXE        IDA IDR
-- ---------- ---------- --------- ---- ---------- ---
--         666 Cerbère    08-MAR-11 M           999 CA
--          42 Rantanplan 01-MAY-09 M            45
```

Composition de requêtes : les sous-requêtes

Exemples de sous-requêtes synchronisées

Synchronisation de requêtes

Pour chaque ligne d'une requête, une sous-requête est déclenchée.

La synchronisation entre les deux requêtes est réalisée par le passage de valeurs de la requête englobante à la sous-requête.

```
-- Pour chaque ligne de 'concours c', la valeur 'c.ville' est passée en entrée
-- de la sous-requête qui est exécutée.
-- Le résultat obtenu 'max(dateC)' est ensuite comparé au 'dateC' courant pour
-- conserver la ligne courante (cas de l'égalité entre les valeurs) ou pas.
select ville, dateC dernierConcours
from concours c
where dateC = (select max(dateC)
               from concours
               where ville = c.ville)
order by dateC desc, ville ;


-- VILLE           DERNIERCONCOURS
-- --------------- ---------------
-- Rodez           28-SEP-12
-- Toulouse        05-MAY-13
```

Composition de requêtes : les vues

Vue

L'expression (⚠ pas le résultat !) d'une requête peut être nommée afin de :

1. décomposer des requêtes complexes,
2. réduire la visibilité de certaines données.

```
-- Définition de la vue
create view chienToulousain as
  select * from chien
  where idA in (select idA from adherent where ville = 'Toulouse') ;
```

Composition de requêtes : les vues

Vue

L'expression (⚠ pas le résultat !) d'une requête peut être nommée afin de :

1. décomposer des requêtes complexes,
2. réduire la visibilité de certaines données.

```sql
-- Définition de la vue
create view chienToulousain as
  select * from chien
  where idA in (select idA from adherent where ville = 'Toulouse') ;
```

```sql
-- Utilisation de la vue : prime et classement des chiens Toulousains
select tatouage, prime, classement
from participation
where tatouage in (select tatouage
                   from chienToulousain)
order by prime desc ;
```

Composition de requêtes : les vues

Vue

L'expression (⚠ pas le résultat !) d'une requête peut être nommée afin de :

1. décomposer des requêtes complexes,
2. réduire la visibilité de certaines données.

```sql
-- Définition de la vue
create view chienToulousain as
  select * from chien
  where idA in (select idA from adherent where ville = 'Toulouse') ;
```

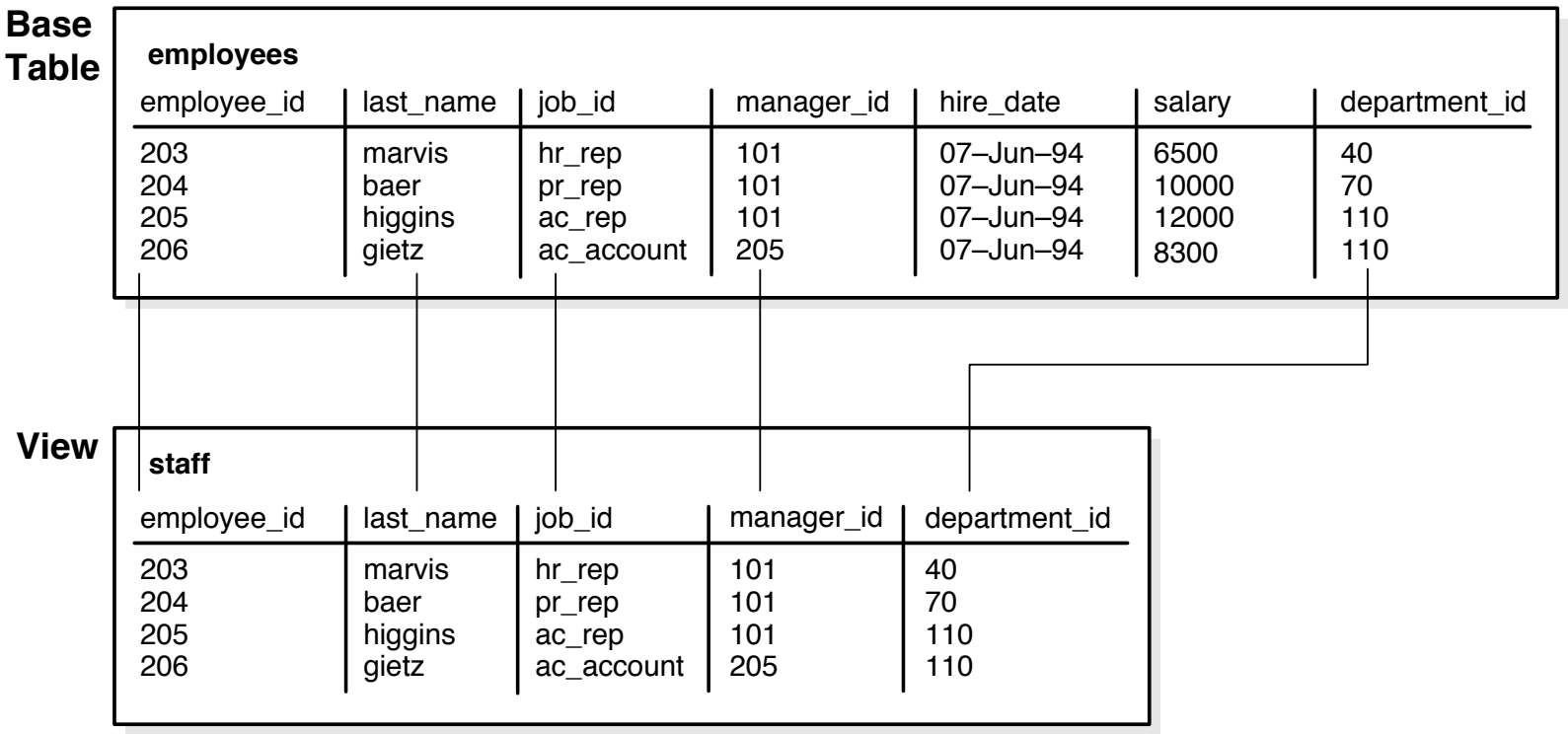
```sql
-- Utilisation de la vue : prime et classement des chiens Toulousains
select tatouage, prime, classement
from participation
where tatouage in (select tatouage
                   from chienToulousain)
order by prime desc ;
```

```sql
-- Cette requête est **réécrite** avant d'être exécutée
select tatouage, prime, classement
from participation
where tatouage in (select tatouage -- vue chienToulousain réécrite ci-dessous
                   from (select * from chien
                         where idA in (select idA from adherent where ville = ' ✂
                              Toulouse')))
order by prime desc ;
```

Composition de requêtes : les vues

Illustration issue de la documentation *Oracle*

Figure 5–5 An Example of a View



Réduction de la visibilité des données par masquage de `hire_date` et `salary`.

Composition de requêtes : les vues « internes »

Vue « interne »

Une vue « interne » créée une vue temporaire le temps d'exécuter la requête. Cette syntaxe permet d'éviter la création de vues dont l'utilité se réduit à une seule requête.

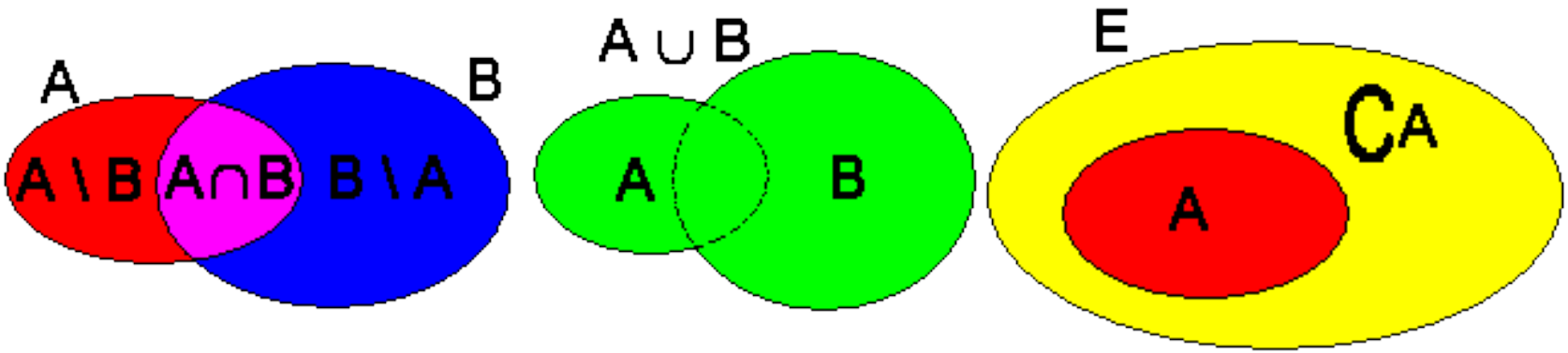
```
-- Prime et classement des chiens Toulousains ?

-- utilisation d'une vue interne qui n'a pas d'existence propre
with chienToulousain as
(
  select *
  from chien
  where idA in (select idA
                from adherent
                where ville = 'Toulouse')
)
select tatouage, prime, classement
from participation
where tatouage in (select tatouage
                   from chienToulousain)
order by prime desc ;
```

LID : Langage d'interrogation des données

Théorie des ensembles de Cantor (XIX^e siècle)

Quelques rappels sur les concepts et les notations...



http://www.maths-informatique-jeux.com/maths/cours/images/operations_ensembles.png

A se lit « l'ensemble A »

$A \cup B$ se lit « A union B »

$A \cap B$ se lit « A inter B »

$A \setminus B$ se lit « A moins B »

$\complement_E A$ se lit « le complément de A dans E »

$x \in A$ se lit « l'élément x appartient à l'ensemble A »

Opérateurs ensemblistes

Union de deux résultats : $R_1 \cup R_2$

$\cup$ Union

Les résultats des deux opérandes sont concaténés verticalement.
Les deux relations doivent être compatibles : même nombre d'attributs de mêmes types.

Commutativité : $A \cup B = B \cup A$

Parallèle avec les maths : $\{1,6,4\} \cup \{6,4,7\} = \{1,4,6,7\}$ $\{5,1\} \cup \emptyset = \{5,1\}$

```sql
-- Avec l'opérateur 'union' : élimination des doublons dans le résultat
select * from adherent where ville = 'Las Vegas'
union
select * from adherent where ville = 'Troyes' ;

-- Requête équivalente avec l'opérateur 'or'
select distinct *
from adherent
where ville = 'Las Vegas'
   or ville = 'Troyes' ;

-- Requête équivalente avec l'opérateur 'in' (∈ en logique)
select distinct *
from adherent
where ville in ('Las Vegas', 'Troyes') ;
```

Opérateurs ensemblistes

Union de deux résultats avec conservation des doublons : $R_1 \uplus R_2$

$\uplus$ Union all

Les résultats des deux opérandes sont concaténés verticalement en conservant lignes identiques (doublons).
Les deux relations doivent être compatibles : même nombre d'attributs de mêmes types.

Commutativité : $A \uplus B = B \uplus A$

Parallèle avec les maths : $\{1,6,4\} \uplus \{6,4,7\} = \{1,6,4,6,4,7\}$ $\quad$ $\{5,1\} \uplus \emptyset = \{5,1\}$

```
-- Avec l'opérateur 'union all' : doublons conservés dans le résultat
select idR from chien where idA = 999
union all
select idR from chien where idA = 45 ;
-- IDR
-- ---
-- CA
-- CO
-- CA
--

-- Avec l'opérateur 'union' : doublons éliminés dans le résultat :
-- {CA, CO, null}
```

Opérateurs ensemblistes

Intersection de deux résultats : $R_1 \cap R_2$

$\cap$ Intersection

Le résultat contient les tuples en commun dans les deux opérandes.
Les deux relations doivent être compatibles : même nombre d'attributs de mêmes types.

Commutativité : $A \cap B = B \cap A$

Parallèle avec les maths : $\{1,6,4\} \cap \{6,4,7\} = \{4,6\} = \{6,4\}$ $\quad \{5,1\} \cap \emptyset = \emptyset$

```sql
-- Concours pour lesquels les chiens 1664 et 666 ont pu se croiser ?

-- Avec l'opérateur 'intersect'
select idC from participation where tatouage = 1664
intersect
select idC from participation where tatouage = 666 ;

-- Requête équivalente avec l'opérateur 'group by'
-- (plus de détail dans quelques diapos...)
select idC
from participation
where tatouage in (1664, 666)
group by tatouage
having count(*) = 2 ;
```

Opérateurs ensemblistes

Différence entre deux résultats : $R_1 \setminus R_2$

$\setminus$ Différence

Le résultat contient les tuples d'une opérande R_1 qui ne sont pas dans une autre opérande R_2. Les deux relations doivent être compatibles : même nombre d'attributs de mêmes types.

Non commutativité : $A \setminus B \neq B \setminus A$

Parallèle avec les maths : $\{1,6,4\} \setminus \{6,4,7\} = \{1\}$ $\{5,1\} \setminus \emptyset = \{5,1\}$

```
-- Participants au concours 12 qui n'ont pas participé au concours 13 ?

-- Avec l'opérateur 'minus' : élimination des doublons dans le résultat
select tatouage from participation where idC = 12
minus
select tatouage from participation where idC = 13 ;

-- Requête équivalente avec l'opérateur 'not in' (∉) et une sous-requête
select tatouage
from participation
where idC = 12
  and idC not in (select idC
                  from participation
                  where idC = 13) ;
```

Opérateurs ensemblistes

Existence de tuples $\exists$

$\exists$ Existence

L'opérateur `exists` renvoie vrai si la sous-requête contient des lignes, faux sinon.

```
-- Propriétaires de chiens : ces deux requêtes donnent le même résultat
select *                                    select *
from adherent                               from adherent
where exists (select *                      where idA in (select idA
              from chien                                  from chien
              where idA = adherent.idA) ;                 where idA is not null) ;

-- IDA NOM          PRENOM     VILLE     MAIL
-- --- ------------ ---------- --------- ----------------
--  45 Vladuboudin  Tintin     Bruxelles
--  69 Morgane      Clara      Marseille clara@hotmail.fr
-- 999 Satan        Michel     Las Vegas

-- NB : la version avec 'exists' est synchronisée
```

LID : Langage d'interrogation des données

Fonctions d'agrégation

Appliquées à un résultat dans son intégralité

Définition

Une fonction d'agrégation synthétise les valeurs d'une colonne (en entrée) en une seule valeur (en sortie).

Parallèle avec les maths : $3+6=9$ $\min(3,4,5)=3$

Fonction	Description
`count(*)`	Nombre total de lignes
`count(col)`	Nombre de lignes dont la valeur de `col` n'est pas vide
`count(distinct col)`	Nombre de valeurs non nulles distinctes dans `col`

Fonctions d'agrégation

Appliquées à un résultat dans son intégralité

Définition

Une fonction d'agrégation synthétise les valeurs d'une colonne (en entrée) en une seule valeur (en sortie).

Parallèle avec les maths : $3+6=9$ $\min(3,4,5)=3$

Fonction	Description
`count(*)`	Nombre total de lignes
`count(col)`	Nombre de lignes dont la valeur de `col` n'est pas vide
`count(distinct col)`	Nombre de valeurs non nulles distinctes dans `col`
`min(col)`	Valeur minimale dans `col`
`max(col)`	Valeur maximale dans `col`

Fonctions d'agrégation

Appliquées à un résultat dans son intégralité

Définition

Une fonction d'agrégation synthétise les valeurs d'une colonne (en entrée) en une seule valeur (en sortie).

Parallèle avec les maths : $3+6=9$ $\min(3,4,5)=3$

Fonction	**Description**
`count(*)`	Nombre total de lignes
`count(col)`	Nombre de lignes dont la valeur de `col` n'est pas vide
`count(distinct col)`	Nombre de valeurs non nulles distinctes dans `col`
`min(col)`	Valeur minimale dans `col`
`max(col)`	Valeur maximale dans `col`
`sum(col)`	Somme des valeurs de `col`
`avg(col)`	Moyenne des valeurs de `col`
`stddev(col)`	Écart-type des valeurs de `col`
`median(col)`	Valeur médiane de `col`

Fonctions d'agrégation

Appliquées à un résultat dans son intégralité. Quelques exemples...

```
select count(*) nbTot, count(idR) as nbRacés, count(*) - count(idP) nbBâtards,
       count(distinct idR) nbRacesDist
from chien ;

--      NBTOT    NBRACÉS  NBBÂTARDS NBRACESDIST
-- ---------- ---------- ---------- -----------
--          5          3          2           2
```

```
-- Nombre de chiens qui ont une race (requêtes équivalentes)
select count(idR)                  select count(*)
from chien ;                       from chien
                                   where idR is not null ;

-- COUNT(IDR)                      COUNT(*)
-- ----------                      --------
--          3                             3
```

```
-- Statistiques sur les gains du chien 1664
select count(prime) nbPrimes, sum(prime) argentGagné, avg(prime) primeMoyenne,
       min(classement) meilleurePlace
from participation
where tatouage = 1664 ;

--   NBPRIMES ARGENTGAGNÉ PRIMEMOYENNE MEILLEUREPLACE
-- ---------- ----------- ------------ --------------
--          4     1563.35       564.45              1
```

Fonctions d'agrégation

Appliquées à des regroupements

Opérateur `group by ... having`

L'opérateur `group by` définit comment grouper des lignes.
L'opérateur `having` définit comment filtrer les groupes.
Les fonctions d'agrégation projetées sont calculées pour chaque groupe.

```
select idR, count(*)
from chien
where idR is not null
group by idR
order by 2 desc, 1 ;


-- IDR   NBCHIENS NBPROPRIOS
-- --- ---------- ----------
-- CA           2          2
-- CO           1          1
```

```
select idR, count(*)
from chien
where idR is not null
group by idR
having count(*) > 1
order by 2 desc, 1 ;

IDR   NBCHIENS NBPROPRIOS
--- ---------- ----------
CA           2          2
```

Fonctions d'agrégation

Appliquées à des regroupements

Opérateur group by ... having

⚠ Règle absolue : $\{S1,\ldots,Sn\} \subseteq \{G1,\ldots,Gk\} \cup \{count, min, max, \ldots\}$

```
select S1, ..., Sn
from R1, ..., Rm
...
group by G1, ..., Gk ;
```

```
-- Nombre de chiens par race et propriétaire ?
-- KO
-- {idR,idA,count} ⊈ {idR,count...}
select idR, idA, count(*) nbCh
from chien
group by idR ;

-- #fail SQL Error: ORA-00979:
-- not a GROUP BY expression
```

```
OK
{idR,idA,count} ⊆ {idR,idA,count,...}
select idR, idA, count(*) nbCh
from chien
group by idR, idA ;
```

```
-- Mauvaise utilisation : utiliser un 'group by' au lieu d'un 'distinct'
-- KO
select idR
from chien
group by idR ;
```

```
-- OK
select distinct idR
from chien ;
```

Fonctions d'agrégation

Appliquées à des regroupements. Exemples...

```
-- Chiens qui ont participé aux concours sans jamais avoir de prime (loosers)
select tatouage
from participation
group by tatouage
having count(prime) = 0 ;
```

```
-- Chiens qui, lorsqu'ils sont classés, ont toujours eu
-- le même classement (mais pas 2e)
select tatouage
from participation
where classement is not null
group by tatouage
having count(distinct classement) = 1
   and max(classement) != 2 ;
```

Fonctions d'agrégation

Équivalence de requêtes : admirez la concision de la construction `group by ... having` !

```
-- Requête avec 'group by' ... 'having'
select idR, count(*) nbCh
from chien
where idR is not null
group by idR
having count(*) > 1
order by 2 desc, 1 ;
```

Fonctions d'agrégation

Équivalence de requêtes : admirez la concision de la construction `group by ... having` !

```sql
-- Requête avec 'group by' ... 'having'
select idR, count(*) nbCh
from chien
where idR is not null
group by idR
having count(*) > 1
order by 2 desc, 1 ;
```

```sql
-- Requête décomposée et réécrite sans 'group by' ... 'having'
-- Elle donne strictement le même résultat
select idR, nbCh
from
(
  -- calcul du nombre de de chiens d'une race donnée  -- synchronisation sur t
  select idR, (select count(*) from chien where idR = t.idR) nbCh
  from
  (
    -- Calcul des idR distincts (clause du 'group by')
    select distinct idR
    from chien
    where idR is not null
  ) as t
)
-- traduction de la clause 'having'
where nbCh > 1
order by 2 desc, 1 ;
```

LID : Langage d'interrogation des données

Produit cartésien : $R_1 \times R_2$

$\times$ Produit cartésien

Le résultat d'un produit cartésien entre deux relations R_1 et R_2 contient $|R_1| \times |R_2|$ lignes : ce sont tous les couples de lignes possibles, issues de R_1 et R_2 (concaténation horizontale).

```
-- Taille du résultat : |adherent| * |chien| = 4 * 5 = 20 lignes
select * from adherent, chien ;
select * from adherent cross join chien ; -- Syntaxe ANSI pour info (non utilisée en TD/TP)

IDA NOM          PRENOM VILLE     MAIL            TATOUAGE  NOM         DDN       SEXE IDA IDR
--- ---------- ------ --------- -------------     -----  ----------  ---------  ---- --- ---
 45 Vladuboudin Tintin Bruxelles                   1664  Iench       12-DEC-12 F     69
 45 Vladuboudin Tintin Bruxelles                    666  Cerbère     08-MAR-11 M    999 CA
 45 Vladuboudin Tintin Bruxelles                     51  Milou       15-FEB-99 M     45 CA
 45 Vladuboudin Tintin Bruxelles                     42  Rantanplan  01-MAY-09 M     45
 45 Vladuboudin Tintin Bruxelles                      3  Lassie      04-MAY-10 F    999 CO
 69 Morgane     Clara  Marseille clara@hotmail.fr  1664  Iench       12-DEC-12 F     69
 69 Morgane     Clara  Marseille clara@hotmail.fr   666  Cerbère     08-MAR-11 M    999 CA
 69 Morgane     Clara  Marseille clara@hotmail.fr    51  Milou       15-FEB-99 M     45 CA
 69 Morgane     Clara  Marseille clara@hotmail.fr    42  Rantanplan  01-MAY-09 M     45
 69 Morgane     Clara  Marseille clara@hotmail.fr     3  Lassie      04-MAY-10 F    999 CO
999 Satan       Michel Las Vegas                   1664  Iench       12-DEC-12 F     69
999 Satan       Michel Las Vegas                    666  Cerbère     08-MAR-11 M    999 CA
999 Satan       Michel Las Vegas                     51  Milou       15-FEB-99 M     45 CA
999 Satan       Michel Las Vegas                     42  Rantanplan  01-MAY-09 M     45
999 Satan       Michel Las Vegas                      3  Lassie      04-MAY-10 F    999 CO
  1 Eastwood    Clint  Troyes  clint@eastwood.com 1664  Iench       12-DEC-12 F     69
  1 Eastwood    Clint  Troyes  clint@eastwood.com  666  Cerbère     08-MAR-11 M    999 CA
  1 Eastwood    Clint  Troyes  clint@eastwood.com   51  Milou       15-FEB-99 M     45 CA
  1 Eastwood    Clint  Troyes  clint@eastwood.com   42  Rantanplan  01-MAY-09 M     45
  1 Eastwood    Clint  Troyes  clint@eastwood.com    3  Lassie      04-MAY-10 F    999 CO
```

Erreur de débutant. . .

Confusion fatale entre produit cartésien $\times$ et jointure $\bowtie$

L'oubli de clauses de jointures conserve des données invalides !

```
-- Les propriétaires et leurs chiens ?
-- KO : oubli de la clause de jointure (adherent.idA = chien.idA)
select *
from adherent, chien ;

 --------------------- Il faudrait qu'ils soient égaux !!! -------------------------
 |                                                                                 |
 V                                                                                 V
IDA NOM          PRENOM VILLE     MAIL           TATOUAGE  NOM         DDN        SEXE IDA IDR
--- ---------- ------ --------- -------------     ----- ---------- --------- ---- --- ---
[...]
 69 Morgane     Clara  Marseille clara@hotmail.fr    666 Cerbère    08-MAR-11 M    999 CA
[...]
999 Satan       Michel Las Vegas                      51 Milou      15-FEB-99 M     45 CA
[...]
  1 Eastwood    Clint  Troyes    clint@eastwood.com  666 Cerbère    08-MAR-11 M    999 CA


-- Clara NE possède PAS Cerbère !
-- Satan NE possède PAS Milou !
-- Clint NE possède PAS Cerbère !
```

LID : Langage d'interrogation des données

Jointure naturelle : $R_1 \bowtie R_2$

$\bowtie$ Jointure naturelle

Une jointure naturelle est un produit cartésien suivi d'une sélection portant sur les attributs en commun dans les deux relations :

$$R_1 \bowtie R_2 = \sigma_{R_1.att=R2.att}(R_1 \times R_2)$$

Propriété : $|R_1 \bowtie R_2| \leqslant |R_1 \times R_2|$

```
select *
from chien, race
where chien.idR = race.idR ;

-- Avec syntaxe ANSI (à titre informatif : non utilisée en TD/TP)
select *
from chien natural join race ;
```

```
                                                  =
                                                  |   |
                                                  V   V
  TATOUAGE NOM        DDN       SEXE        IDA IDR IDR LIBELLE         DESCRIPTION
---------- ---------- --------- ---- ---------- --- --- --------------- -----------
       666 Cerbère    08-MAR-11 M           999 CA  CA  Caniche         Petit et frisé
        51 Milou      15-FEB-99 M            45 CA  CA  Caniche         Petit et frisé
         3 Lassie     04-MAY-10 F           999 CO  CO  Colley          Poilu et doré


-- NB : Seuls les chiens possédant une race sont listés
--      ==> Inutile de spécifier 'where race.idR is not null' dans cet exemple
```

Semi-jointures gauche $R_1 \ltimes R_2$ et droite $R_1 \rtimes R_2$

$\ltimes$ Semi-jointure $\rtimes$

Une semi-jointure gauche $\ltimes$ (resp. droite $\rtimes$) est une jointure naturelle dont on ne projette que les attributs provenant de la relation de gauche (resp. droite).

$$R_1 \ltimes R_2 = \pi_{R_1.*}(R_1 \bowtie R_2)$$

$$R_1 \rtimes R_2 = \pi_{R_2.*}(R_1 \bowtie R_2)$$

```
-- Semi-jointure gauche (pour la droite : select race.*)
select chien.*
from chien, race
where chien.idR = race.idR ;
```

```
-- Le résultat ne contient que les colonnes de chien
  TATOUAGE NOM        DDN       SEXE        IDA IDR
---------- ---------- --------- ---- ---------- ---
       666 Cerbère    08-MAR-11 M           999 CA
        51 Milou      15-FEB-99 M            45 CA
         3 Lassie     04-MAY-10 F           999 CO
```

θ-jointure : $R_1 \bowtie_\theta R_2$

θ-jointure

Une θ-jointure exprime une condition booléenne de sélection avec les opérateurs binaires $\leqslant, <, =, \neq, >, \geqslant$.

```
-- Couples mâle et femelles avec un mâle plus âgé que la femelle ?
select male.tatouage m, femelle.tatouage f
from chien male, chien femelle
where male.ddn > femelle.ddn
  and male.sexe = 'M'
  and femelle.sexe = 'F' ;

--          M          F
-- ---------- ----------
--        666          3

-- NB : cette requête illustre le cas particulier de l'auto-jointure sur chien.
```

équi-jointure : $R_1 \bowtie_{R_1.att_i = R_2.att_j} R_2$

$\bowtie_{R_1.att_i = R_2.att_j}$ équi-jointure

L'équi-jointure est une forme particulière de θ-jointure,
où θ est l'opérateur d'égalité.

```
-- Propriétaires et leurs chiens ?
select *
from adherent, chien
where adherent.idA = chien.idA ;

-- NB1 : Ce n'est pas une jointure naturelle car les tables adherent et chien
--       ont 2 colonnes en commun. Or ici on ne joint que sur la colonne idA
-- Avec syntaxe ANSI : attention, à manipuler avec précaution...
select * from adherent natural join chien ;      -- naturelle : KO
select * from adherent join chien using (idA) ; -- theta : OK
```

```
 ------------------------------------(=)-------------------------------------------
 |                                                                                 |
 V                                                                                 V
IDA NOM          PRENOM VILLE     MAIL            TATOUAGE NOM         DDN       SEXE IDA IDR
--- ---------- ------ --------- ---------------- ---- ---------- --------- ---- --- ---
 45 Vladuboudin Tintin Bruxelles                      51 Milou      15-FEB-99 M     45 CA
 45 Vladuboudin Tintin Bruxelles                      42 Rantanplan 01-MAY-09 M     45
 69 Morgane     Clara  Marseille clara@hotmail.fr   1664 Iench     12-DEC-12 F     69
999 Satan       Michel Las Vegas                      3 Lassie     04-MAY-10 F    999 CO
999 Satan       Michel Las Vegas                    666 Cerbère    08-MAR-11 M    999 CA

-- NB2 : Client Eastwood n'est pas présent dans le résultat car il ne possède aucun chien.
```

Expressivité de l'équi-jointure

Reprise de l'exemple de la diapositive 16 : « Speed dating pour chiens »

```
-- Speed dating pour chiens :
-- Adhérents disposant d'un mail, possédant une chienne et habitant
-- dans la même ville que les adhérents possédant un chien.
```

```
-- Version imbriquée
select nom, prenom
from adherent
where idA in (select idA          --> possesseurs de chiennes
              from chien
              where sexe = 'F')
  and ville in (select ville      --> ville des possesseurs de chiens
                from adherent
                where idA in (select idA  --> possesseurs de chiens
                              from chien
                              where sexe = 'M'))
  and mail is not null  --> possesseurs de chiennes avec email
order by 1, 2 ;
```

```
-- Version relationnelle (avec jointures)
select a1.nom, a1.prenom
from adherent a1, chien c1, adherent a2, chien c2
where a1.idA = c1.idA         -- jointure 1/2 : a1 possède c1
  and a2.idA = c2.idA         -- jointure 2/2 : a2 posède c2
  and c1.sexe = 'F'           -- c1 est une chienne
  and c2.sexe = 'M'           -- c2 est un chien
  and a1.ville = a2.ville     -- a1 et a2 habitent dans la même ville
  and a1.mail is not null     -- a1 possède une adresse mail
order by 1, 2 ;
```

Jointure externe gauche $R_1 \overset{\frown}{\bowtie} R_2$

Jointure externe gauche $R_1 \overset{\frown}{\bowtie} R_2$

Résultat de $R_1 \bowtie R_2$ auquel on ajoute les lignes de R_1 qui n'ont pas été appariées avec R_2.

```
-- Les propriétaires, avec leurs chiens éventuels ?
select *
from adherent, chien
where adherent.idA = chien.idA (+) ;

-- Avec syntaxe ANSI
select *
from adherent left join chien on adherent.idA = chien.idA ;
```

```
IDA NOM          PRENOM VILLE     MAIL              TATOUAGE NOM         DDN       SEXE IDA IDR
--- ---------- ------ --------- ----------------- ---- ---------- --------- ---- --- ---
  1 Eastwood     Clint  Troyes    clint@eastwood.com
 45 Vladuboudin Tintin Bruxelles                    51 Milou      15-FEB-99 M     45 CA
 45 Vladuboudin Tintin Bruxelles                    42 Rantanplan 01-MAY-09 M     45
 69 Morgane     Clara  Marseille clara@hotmail.fr 1664 Iench      12-DEC-12 F     69
999 Satan       Michel Las Vegas                    3 Lassie     04-MAY-10 F    999 CO
999 Satan       Michel Las Vegas                  666 Cerbère    08-MAR-11 M    999 CA
```

```
-- Autre utilité : les propriétaires qui n'ont pas de chien ?
select *
from adherent, chien
where adherent.idA = chien.idA (+)
  and chien.idA is null ;
```

Jointure externe droite $R_1 \rtimes R_2$

Jointure externe droite $R_1 \rtimes R_2$

Résultat de $R_1 \bowtie R_2$ auquel on ajoute les lignes de R_2 qui n'ont pas été appariées avec R_1.

```
-- Les propriétaires, avec leurs chiens éventuels ?
select *
from chien, adherent
where chien.idA (+) = adherent.idA ;

-- Avec syntaxe ANSI
select *
from chien right join adherent on chien.idA = adherent.idA ;
```

```
TATOUAGE NOM        DDN       SEXE   IDA IDR IDA NOM          PRENOM VILLE      MAIL
---------- ---------- --------- ---- --- --- --- ----------   ------ --------- ------------------
                                                1 Eastwood     Clint  Troyes     clint@eastwood.com
        51 Milou      15-FEB-99 M        45 CA    45 Vladuboudin Tintin Bruxelles
        42 Rantanplan 01-MAY-09 M        45       45 Vladuboudin Tintin Bruxelles
      1664 Iench      12-DEC-12 F        69       69 Morgane     Clara  Marseille clara@hotmail.fr
         3 Lassie     04-MAY-10 F       999 CO   999 Satan       Michel Las Vegas
       666 Cerbère    08-MAR-11 M       999 CA   999 Satan       Michel Las Vegas
```

```
-- Autre utilité : les propriétaires qui n'ont pas de chien ?
select *
from chien, adherent
where chien.idA (+) = adherent.idA
  and chien.idA is null ;
```

Jointure externe complète $R_1 \overset{\ltimes}{\rtimes} R_2$

Jointure externe complète $R_1 \overset{\ltimes}{\rtimes} R_2$

$R_1 \overset{\ltimes}{\rtimes} R_2 = (R_1 \rtimes R_2) \cup (R_1 \ltimes R_2)$

```sql
select *
from chien, race
where chien.idR (+) = race.idR
union
select *
from chien, race
where chien.idR = race.idR (+)

-- Avec syntaxe ANSI
select *
from chien full outer join race on chien.idR = race.idR ;
```

```
--   TATOUAGE NOM         DDN       SEXE        IDA IDR IDR LIBELLE          DESCRIPTION
-- ---------- ---------- --------- ---- ---------- --- --- ---------------- --------------
--          3 Lassie     04-MAY-10 F            999 CO  CO  Colley           Poilu et doré
--         42 Rantanplan 01-MAY-09 M             45
--         51 Milou      15-FEB-99 M             45 CA  CA  Caniche         Petit et frisé
--        666 Cerbère    08-MAR-11 M            999 CA  CA  Caniche         Petit et frisé
--       1664 Iench      12-DEC-12 F             69
--                                                 BA  Berger Allemand  cf. Rintintin
```

Sources

Les illustrations sont reproduites à partir du document suivant :

- Oracle Database Documentation Library 11.1
  (`http://www.oracle.com/pls/db111/homepage`)
  - Oracle® Database : Concepts 11g Release 1 (11.1) B28318-06

- Oracle Database Documentation Library 11.2
  (`http://www.oracle.com/pls/db112/homepage`)
  - Oracle® Database : Concepts 11g Release 2 (11.2) E40540-01