SQL – une longue histoire

Année	Nom	Appellation	Commentaires
1986	ISO/CEI 9075:1986	SQL-86 ou SQL-87	Édité par l'ANSI puis adopté par l'ISO en 1987.
1989	ISO/CEI 9075:1989	SQL-89 ou SQL-1	Révision mineure.
1992	ISO/CEI 9075:1992	SQL-92 (en) alias SQL2	Révision majeure.
1999	ISO/CEI 9075:1999	SQL-99 (en) alias SQL3	Expressions rationnelles, requêtes récursives, déclencheurs, types non-scalaires et quelques fonctions orientées objet (les deux derniers points sont quelque peu controversés et pas encore largement implémentés).
2003	ISO/CEI 9075:2003	SQL:2003 (en)	Introduction de fonctions pour la manipulation XML, « window functions », ordres standardisés et colonnes avec valeurs auto-produites (y compris colonnes d'identité).
2008	ISO/CEI 9075:2008	SQL:2008 (en)	Ajout de quelques fonctions de fenêtrage (ntile, lead, lag, first value, last value, nth value), limitation du nombre de ligne (OFFSET / FETCH), amélioration mineure sur les types distincts, curseurs et mécanismes d'auto incréments.
2011	ISO/CEI 9075:2011	SQL:2011 (en)	

SQL : pour quoi faire ?

- Une interface plus simple pour les utilisateurs ?
  - la majorité des manipulations utilisateurs se font via des formulaires !
  - Mais rend possible les requêtes libres (ex: analyse)
- Simplifier le développement des applications
  - Code plus compact et facile à valider/maintenir
  - Indépendance physique et logique des programmes par rapport aux données
  - Optimisation automatique et dynamique des requêtes par le SGBD
- Une puissance d'expression exploitable de nb façons
  - Sélection, mises à jour, intégrité, droits d'accès, audit …

Le standard SQL

LANGAGE DE DEFINITION DE DONNEES

- **CREATE TABLE**
- **CREATE VIEW**
- **ALTER ….**
- **...**

LANGAGE DE CONTROLE

- **GRANT**
- **REVOKE**
- **COMMIT**
- **ROLLBACK**

LANGAGE DE MANIPULATION DE DONNEES

- **SELECT**
- **INSERT**
- **UPDATE**
- **DELETE**

INTEGRATION AUX LANGAGES DE PROGRAMMATION

- **EXEC SQL**
- **MODULE**
- **PROCEDURE ...**

Le standard SQL

LANGAGE DE DEFINITION DE DONNEES

- **CREATE TABLE**
- **CREATE VIEW**
- **ALTER ….**
- **...**

LANGAGE DE CONTROLE

- **GRANT**
- **REVOKE**
- **COMMIT**
- **ROLLBACK**

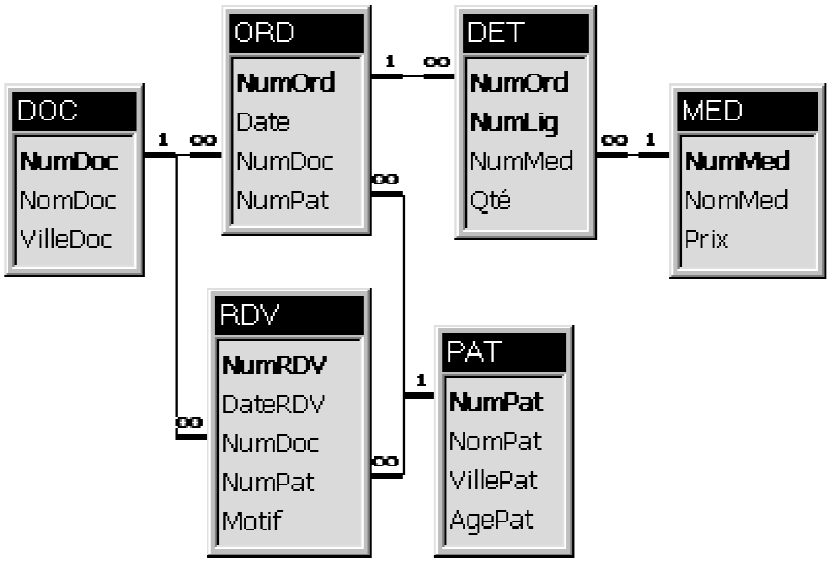
LANGAGE DE MANIPULATION DE DONNEES

- **SELECT**
- **INSERT**
- **UPDATE**
- **DELETE**

INTEGRATION AUX LANGAGES DE PROGRAMMATION

- **EXEC SQL**
- **MODULE**
- **PROCEDURE ...**

EXEMPLE DE BASE DE DONNEES



Création de table

CREATE TABLE <nom_table>
(<def_colonne> * [<def_contrainte_table>*]) ;

< def_colonne > ::= <nom_colonne> < type > [CONSTRAINT nom_contrainte <
NOT NULL |
UNIQUE |
PRIMARY KEY |
CHECK (condition) |
REFERENCES nom_table (colonne) >]

< def_contrainte_table > ::= CONSTRAINT nom_contrainte <
UNIQUE (liste_colonnes) |
PRIMARY KEY (liste_colonnes) |
CHECK (condition) |
FOREIGN KEY (liste_colonnes) REFERENCES nom_table (liste_colonnes)>

Exemple de création de table

CREATE TABLE RDV(
NumRdv Integer,
DateRDV Date,
NumDoc Integer,
NumPat Integer,
Motif Varchar(200),
CONSTRAINT Clé_Primaire_RDV PRIMARY KEY (NumRdv),
CONSTRAINT Réf_DOC FOREIGN KEY (NumDoc) REFERENCES DOC (NumDoc),
CONSTRAINT Réf_PAT FOREIGN KEY (NumPat) REFERENCES PAT (NumPat))

L'association d'un nom à une contrainte est optionnelle. Ce nom peut être utilisé pour référencer la contrainte (ex: messages d'erreurs).

Exercices

Donnez l'expression SQL de la création des tables DOC et DET

CREATE TABLE DOC(
NumDoc integer,
NomDoc char(30),
VilleDoc varchar(50),
CONSTRAINT Clé_Primaire_Doc PRIMARY KEY (NumDoc)) ;

CREATE TABLE DET(
NumOrd integer,
NumLig integer,
NumMed integer,
Qté integer,
CONSTRAINT Clé_Primaire_DET PRIMARY KEY (NumOrd, NumLig),
CONSTRAINT Réf_ORD FOREIGN KEY (NumOrd) REFERENCES ORD (NumOrd)
CONSTRAINT Réf_MED FOREIGN KEY (NumMed) REFERENCES MED(NumMed));

Index, modification du schéma

- Création d'index
  - CREATE [UNIQUE] INDEX **[nom_index]** ON **nom_table (<nom_colonne> *);**
- Suppression
  - DROP TABLE <nom_table>
  - DROP INDEX <nom_index>
- Modification
  - ALTER TABLE <nom_table> ADD COLUMN <def_colonne>
  - ALTER TABLE <nom_table> ADD CONSTRAINT <def_contrainte_table >
  - ALTER TABLE <nom_table> ALTER <def_colonne>
  - ALTER TABLE <nom_table> DROP COLUMN <nom_colonne>
  - ALTER TABLE <nom_table> DROP CONSTRAINT <nom_contrainte >
- Exemples
  - CREATE INDEX Index_date_RDV ON RDV (DateRDV) ;
  - ALTER TABLE RDV ADD COLUMN Commentaires varchar(300);
  - ALTER TABLE RDV ADD CONSTRAINT MotifNN NOTNULL(Motif);

Exercices

Supprimez l'attribut Motif de la table RDV

ALTER TABLE RDV
DROP COLUMN Motif;

Ajoutez une contrainte de clé primaire à la table MED (sur NumMed)

ALTER TABLE MED
ADD CONSTRAINT cle_prim_MED PRIMARY KEY (NumMed) ;

Le standard SQL

LANGAGE DE DEFINITION DE DONNEES

CREATE TABLE
CREATE VIEW
ALTER ….
...

LANGAGE DE CONTROLE

GRANT
REVOKE
COMMIT
ROLLBACK

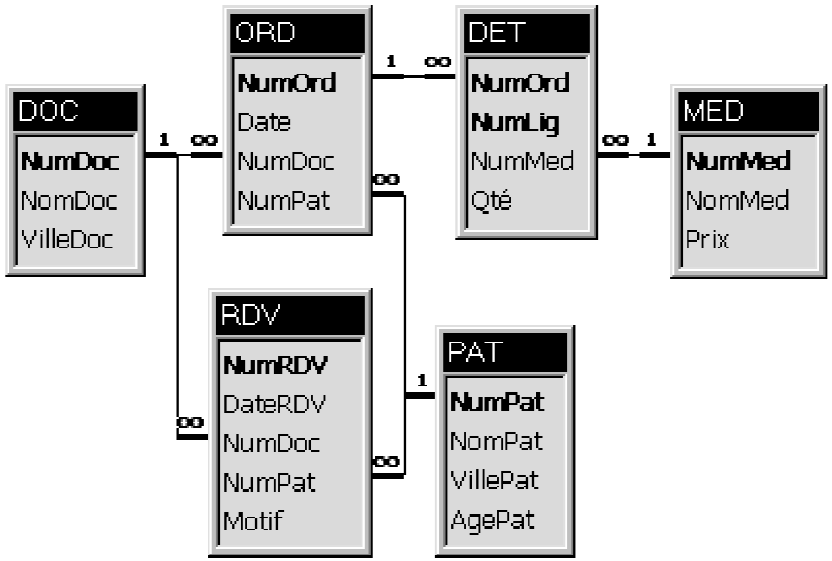
LANGAGE DE MANIPULATION DE DONNEES

SELECT
INSERT
UPDATE
DELETE

INTEGRATION AUX LANGAGES DE PROGRAMMATION

EXEC SQL
MODULE
PROCEDURE ...

EXEMPLE DE BASE DE DONNEES



SELECT : forme générale

SELECT **[**DISTINCT| ALL**]** **{ * | <value exp.> [, <value exp.>]...}**

FROM **relation [variable], relation [variable]…**

[WHERE **<search condition>]**

[GROUP BY **<attribute> [,<attribute>]...]**

[HAVING **<search condition>]**

[ORDER BY **<attribute> [{**ASC | DESC**}] [,<attribute>[{**ASC | DESC**}]]...]**

* EXPRESSION DE VALEURS
  - Calculs arithmétiques
  - Fonctions agrégats

* CONDITION DE RECHERCHE
  - Sélection, projection, jointure
  - Recherche textuelle
  - Recherche par intervalle
  - Recherche sur valeur nulle

Forme générale de la condition de recherche

<**search condition**> ::= **[**NOT**]**

- <**nom_colonne**> **θ constante** | <**nom_colonne**>
- <**nom_colonne**> LIKE **<modèle_de_chaîne>**
- <**nom_colonne**> IN <**liste_de_valeurs**>
- <**nom_colonne**> **θ (**ALL | ANY | SOME**)** <**liste_de_valeurs**>
- EXISTS <**liste_de_valeurs**>
- UNIQUE <**liste_de_valeurs**>
- <**tuple**> MATCH **[**UNIQUE**]** <**liste_de_tuples**>
- <**nom_colonne**> BETWEEN **constante** AND **constante**
- <**search condition**> AND | OR <**search condition**>

avec

θ ::= < | = | > | ≥ |≤ | <>

Remarque: <**liste_de_valeurs**> **peut être déterminée par une requête**

Projections et restrictions simples

Liste des médicaments de plus de 100 € ➔ NomMed

SELECT **NomMed** FROM **MED** WHERE **Prix> 100 ;**

Liste des médicaments de plus de 100 € ➔ NomMed (prix stocké en FF)

SELECT **NomMed** FROM **MED** WHERE **Prix/6,55957 > 100 ;**

Nom des docteurs de LAON ➔ NomDoc

SELECT **NomDoc** FROM **DOC** WHERE **VilleDoc = "Laon"**

Restrictions complexes et jointures

Liste des patients ayant un RDV avec le docteur "Dupont" ➔ NomPat

SELECT **DISTINCT PAT.NomPat** FROM **PAT, RDV, DOC**
WHERE **PAT.NumPat = RDV.NumPat and RDV.NumDoc = DOC.NumDoc and DOC.NomDoc like 'Dupont';**

Médicaments commençant par « ASPI » prescrits le 25/12/2006 ➔ NomMed

SELECT **DISTINCT M.NomMed** FROM **MED M, DET D, ORD O**
WHERE **M.NumMed = D.NumMed and D.NumOrd = O.NumOrd and O.Date = '25/12/2006' and NomMed like 'ASPI%';**

Exercices

- **Age des patients en mois (il est stocké en année) ➔** NomPat, AgeMois

```
SELECT NomPat, AgePat*12 AgeMois
FROM PAT
```

- **Docteurs ayant le même nom qu'un de leur patient ➔** NomDoc

```
SELECT DISTINCT D.NomDoc
FROM PAT P, DOC D
WHERE P.NomPat = D.NomDoc;
```

Requêtes imbriquées : IN et EXISTS

Liste des patients ayant un RDV avec le docteur "Dupont" ➔ NomPat

```
SELECT DISTINCT P.NomPat FROM PAT P, RDV R, DOC D
WHERE P.NumPat = R.NumPat and R.NumDoc = D.NumDoc and D.NomDoc = "Dupont";
```

```
SELECT P.NomPat FROM PAT P WHERE P.NumPat in
  (SELECT R.NumPat FROM RDV R WHERE R.NumDoc in
    (SELECT D.NumDoc FROM DOC WHERE D.NomDoc = "Dupont"));
```

```
SELECT P.NomPat FROM PAT P WHERE EXISTS
  (SELECT * FROM RDV R WHERE P.NumPat = R.NumPat and EXISTS
    (SELECT * FROM DOC D WHERE R.NumDoc=D.NumDoc and D.NomDoc = "Dupont"));
```


Exercices

- **Nom des docteurs ayant au moins un RDV pour une grippe (en requête imbriquée) ➔** NomDoc

```
SELECT D.NomDoc FROM DOC D WHERE D.NumDoc IN
    (SELECT R.NumDoc FROM RDV R WHERE R.Motif LIKE 'grippe'):
```

- **Nom des patients qui n'ont jamais eu de rendez-vous ➔** NomPAT

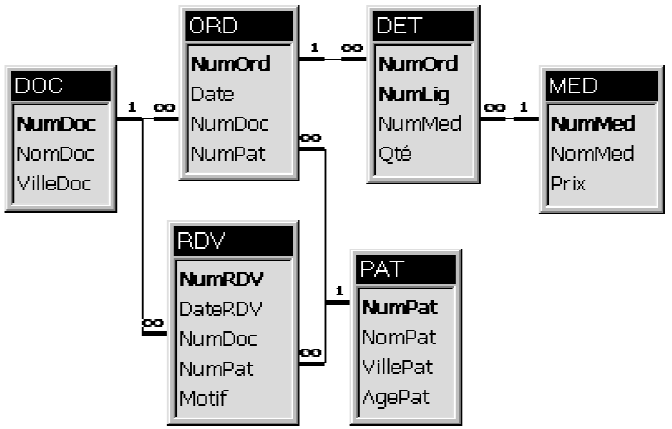
```
SELECT P.NomPat FROM PAT P WHERE P.NumPat NOT IN
    (SELECT R.NumPat FROM RDV R )

SELECT P.NomPat FROM PAT P WHERE NOT EXISTS
    (SELECT * FROM RDV R WHERE P.NumPat = R.NumPat)
```

- **Nom des patients qui ont eu rendez-vous avec tous les médecins ➔** NomPAT

<=> Quels sont les patients tels qu' il n'existe pas de docteur tel qu' il n'existe pas de rendez-vous entre ce patient et ce docteur ?

```
SELECT P.NomPat FROM PAT P WHERE NOT EXISTS
    (SELECT * FROM DOC D WHERE NOT EXISTS
        (SELECT * FROM RDV R WHERE
            P.NumPat = R.NumPat and R.NumDoc = D.NumDoc)
```

Agrégation, Union, Intersection, Différence



Calculs d'agrégats

Les fonctions d'agrégation (Count, Sum, Avg, Min, Max) permettent de réaliser des calculs sur des ensembles de données

- **Calcul de statistiques globaux**
  - **Nombre de patients : SELECT** count(*) **FROM** PAT
  - **Prix moyen des médicaments : SELECT** avg(Prix) **FROM** MED
- **Calcul de statistiques par groupe**
  - Nombre de patients par ville

  SELECT VillePat, count(*) NbPatient **FROM** PAT **GROUP BY** VillePat
  - Nombre de patients par ville ayant consulté pour un mal de tête

  SELECT VillePat, count(**DISTINCT**(NumPat)) NbPatient **FROM** PAT, RDV
  WHERE PAT.NumPat = RDV.NumPat and Motif = 'mal de tête' **GROUP BY** VillePat ?
  - Ville où plus de 10 patients ont consulté pour un mal de tête

  SELECT VillePat **FROM** PAT, RDV
  WHERE PAT.NumPat = RDV.NumPat and Motif = 'mal de tête'
  GROUP BY VillePat
  HAVING count(**DISTINCT**(NumPat)) > 10 ?

Exercices

- **Total des prix des médicaments prescrits par patient**

  SELECT P.NomPat, sum(M.prix * O.Qté) PrixTotal
  FROM PAT P, ORD O, DET D, MED M
  WHERE M.NumMed = D.NumMed and D.NumOrd = O.NumOrd and O.NumPat = P.NumPat
  GROUP BY P.NomPat
- Nom des docteurs ayant fait plus de 1000 ordonnances

  SELECT DOC. NomDoc
  FROM DOC, ORD
  WHERE DOC.numDoc = ORD.NumDoc
  GROUP BY DOC.NumDoc, DOC.NOMDOC
  HAVING Count(DISTINCT NumORD) > 1000

Union/Intersection/Différence

<requêteSQL_A>

| **UNION** [**ALL**]
| **INTERSECT**
| **EXCEPT**

<requêteSQL_B>

[**ALL**] permet de conserver les doublons dans le résultat d'une union

Attention, les tables opérandes doivent avoir le même schéma

Union/Inter°/Diff. : exemples et exercices

- **Ensemble des personnes de la base médicale**

SELECT NomMed NomPers **FROM** MED **UNION SELECT** NomPat NomPers **FROM** PAT

- **Patients qui sont aussi médecin**

SELECT NomPat PatMed **FROM** PAT **INTERSECT SELECT** NomMed PatMed **FROM** MED

- **Patients qui ne sont pas médecin**

SELECT NomPat Patient **FROM** PAT **EXCEPT SELECT** NomMed Patient **FROM** MED

Jointure Interne / Externe

<Join_expression> ::=

<table-ref> [**NATURAL**] [<join_type>] **JOIN** <table-ref>

[**ON** <condition> | **USING** <nom_colonne> ***]**

<join_type> ::=

INNER

LEFT [OUTER]

RIGHT [OUTER]

FULL [OUTER]

Nom des docteurs et dates de leurs RDV s'ils en ont

SELECT **DOC. NomDoc, RDV.DateRDV**
FROM **DOC NATURAL LEFT OUTER JOIN RDV**

Insertion de données

INSERT INTO < nom_table >

[(attribute [,attribute] …)]

{VALUES (<value spec.> [, <value spec.>] …) |

<query specification>} ;

- Exemples :
  - INSERT INTO DOC VALUES (1, 'Dupont', 'Paris');
  - INSERT INTO DOC (NumDoc, NomDoc) VALUES (2, 'Toto');
  - INSERT INTO PAT (NumPat, NomPat, VillePat)
    SELECT NumDoc, NomDoc, VilleDoc FROM DOC;

Mise à jour : UPDATE

SYNTAXE :

UPDATE **<relation_name>**
SET **<attribute> = value_expression [, <attribute> = value_expression] …**
[WHERE **<search condition>**];

EXEMPLES :

Mettre "Inconnue" quand VilleDoc n'est pas renseignée

UPDATE **DOC** SET **VilleDoc = "Inconnue"** WHERE **VilleDoc** is NULL

Mettre en majuscule le nom des docteurs qui n'ont jamais rien prescrit

UPDATE **DOC** SET **NomDoc = UPPER(NomDoc)** WHERE **NumDoc** NOT IN **(**SELECT **NumDoc** FROM **ORD)**

ATTENTION AUX CONTRAINTES D'INTEGRITE REFERENTIELLES !!!

Suppression : DELETE

SYNTAXE :

DELETE FROM **<relation_name>**
[WHERE **<search_condition>**]

EXEMPLES :

Supprimer les docteurs quand VilleDoc n'est pas renseignée

DELETE FROM **DOC** WHERE **VilleDoc** is NULL

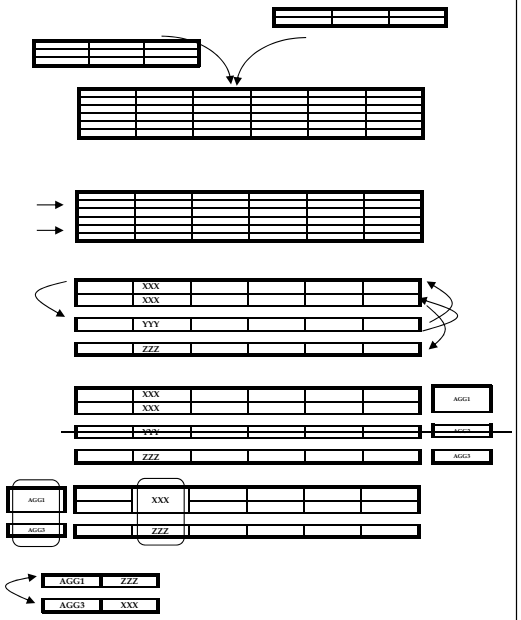
Supprimer les docteurs qui n'ont jamais rien prescrit

DELETE FROM **DOC** WHERE **NumDoc** NOT IN **(**SELECT **NumDoc** FROM **ORD)**

ATTENTION AUX CONTRAINTES D'INTEGRITE REFERENTIELLES !!!

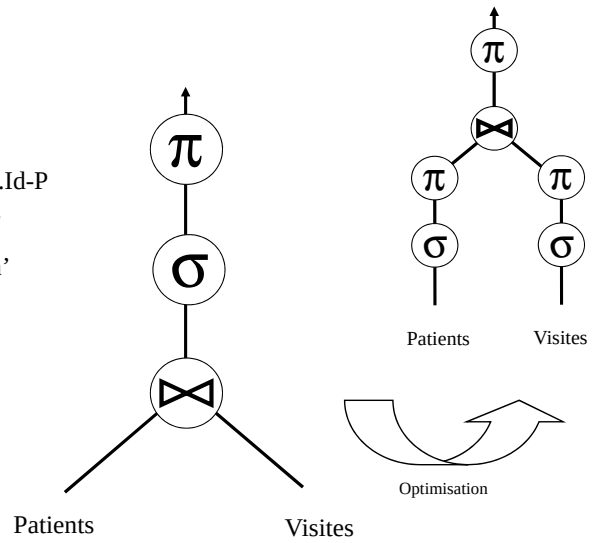
Évaluation « sémantique » d'une requête SQL

1. FROM
   Réalise le produit cartésien des relations
2. WHERE
   Réalise restriction et jointures
3. GROUP BY
   Constitue les partitions
   (e.g., tri sur l'intitulé du groupe)
4. HAVING
   Restreint aux partitions désirées
5. SELECT
   Réaliser les projections/calculs finaux
6. ORDER BY
   Trier les tuples résultat



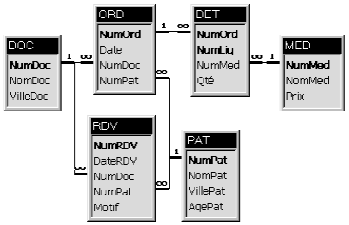
Evaluation opérationnelle d'une requête SQL

Select Nom, Prénom
From Patients, Visites
Where Patients.Id-P = Visites.Id-P
and Patients.Ville = 'Paris'
and Visites.Date = '15 juin'



Eléments de méthodologie

- Avant de se lancer dans l'écriture d'une requête, il faut bien comprendre le schéma des tables sur lequel on va s'appuyer.
- Si le schéma est complexe, il faut le dessiner, c.a.d. dessiner les tables et les relations entre ces tables.
- En lisant la question, on repère sur le schéma dans quelle(s) relation(s) se trouve chaque donnée.
- Si la question est complexe, il faut la reformuler et/ou la décomposer.
-



Reformulation : Négation

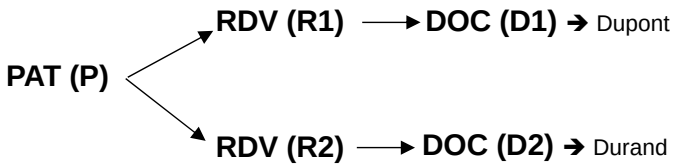
Souvent l'inverse de la requête est plus facile à exprimer.
Cela est particulièrement vrai lorsque la requête contient :

- **Que**
  - **Dans quelles villes n'y a-t-il que des patients de plus de 40 ans?**
  - **L'ensemble des villes moins celles où il y a au moins un patient de 40 ans ou moins**
- **Aucun**
  - **Dans quelles villes n'y a-t-il aucun patient de plus de 40 ans?**
  - **L'ensemble des villes moins celles où il y a au moins un patient de plus de 40 ans**
- **Tous**
  - **Quels sont les patients dont tous les motifs de rendez-vous sont « mal de tête »**
  - **L'ensemble des patients qui ont un RDV pour un mal de tête moins les patients qui ont un RDV pour un motif différent**
- **Tous**
  - **Quels sont les patients qui ont RDV avec tous les médecins**
  - **Les patients pour lesquels il n'existe pas de docteur avec qui ils n'ont pas eu de RDV**
  - **Les patients qui ont vu un nombre de médecins égal au nombre total de médecins de la base**

Instances de tables

- Il faut parfois utiliser plusieurs instances de la même table. Comment savoir?
- Lorsqu'une même table est utilisée pour obtenir deux informations différentes**,** il faut prendre plusieurs instances de cette table.
- Exemple : Nom des patients ayant eu des RDV avec les docteurs "Dupont" et "Durand"

```
SELECT DISTINCT P.NomPat
FROM PAT P, RDV R, DOC D , RDV R1, DOC D1
WHERE P.NumPat = R1.NumPat and R1.NumDoc = D1.NumDoc and
      D1.NomDoc = "Dupont" and P.NumPat = R2.NumPat and
      R2.NumDoc = D2.NumDoc and D2.NomDoc = "Durand");
```



Doublons

- **Deux types de doublons peuvent apparaître.**
  - **plusieurs réponses sont identiques**
  - **plusieurs réponses sont 'sémantiquement équivalentes'**
- **Les premiers s'éliminent en utilisant la clause distinct.**
- **Les second se produisent lorsque l'on demande des combinaisons (couples de personnes, ensemble de 3 pièces etc...). Or le résultat (A,B) est 'sémantiquement équivalent' à (B,A).**
- **Dans ce cas, il suffit d'utiliser un prédicat > entre chaque composante afin de n'obtenir qu'une seule des combinaisons.**