

Modèle Relationnel (MR)

Modèle Relationnel

Les tables/relations :

- Client (N°SS, Nom, Prénom, Bonus, Malus)
- Contrat (N°Contrat, Date, #N°SS, #Numéro, #N°SS-Agent)
- Agent (N°SS-Agent, Nom, Prénom, Adresse,...)
- Véhicule (Numéro, Puissance, Marque, Couleur, Type, # N°Contrat)
- Risque (Code, libellé, Prix)
- Tiers (N°SS-Tiers, #Id)
- Compagnie (Id, Nom, Adresse)
- Sinistre (# N°SS, # N°SS-Tiers, date, lieu)
- Couvre (# N°Contrat, # Code)

Jointures :

Algèbre Relationnelle


- Opérateurs spécifiques - *Jointure* (\bowtie):
- Exemple inéqui-jointure :

Patient

nss	Nom_P	N° médecin	adresse
1807514511	Dupond	4125	Avon
2909112000	Smith	5223	Paris
2873914536	Pierre	4125	Lyon

Médecin

N° médecin	Nom_M	service
4125	House	Urgence
5223	Bruno	génécologie


 $R = \text{JOIN}(\text{Patient}, \text{Médecin})$
 Condition : $\text{nss} > 2000000000$

R

nss	Nom_P	adresse	N° médecin	Nom_M	service
2909112000	Smith	Paris	5223	Bruno	génécologie
2873914536	Pierre	Lyon	4125	House	Urgence

Algèbre Relationnelle

- Opérateurs spécifiques - *Jointure externe* (\bowtie_{\circ}):
- Exemple jointure externe (*outer joint*) :

Patient

nss	Nom_P	N° médecin	adresse
1807514511	Dupond	4125	Avon
2909112000	Smith	5223	Paris
2873914536	Pierre	4100	Lyon

Médecin

N° médecin	Nom_M	service
4125	House	Urgence
2321	Wilson	Urgence
5223	Bruno	génécologie


 $R = \text{Ext-JOIN}(\text{Patient}, \text{Médecin})$

R

nss	Nom_P	adresse	N° médecin	Nom_M	service
1807514511	Dupond	Avon	4125	House	Urgence
2909112000	Smith	Paris	5223	Bruno	génécologie
2873914536	Pierre	Lyon	4100	-	-
-	-	-	2321	Wilson	Urgence

Création de table : Syntaxe

- **Syntaxe**

CREATE TABLE <relation name>

(<attribute definition>+)

[{PRIMARY KEY | UNIQUE} (<attribute name>+)]

- **Exemple**

CREATE TABLE RDV (

NumRdv Integer PRIMARY KEY,

DateRDV Date,

NumDoc Integer,

NumPat Integer,

Motif Varchar(200),

FOREIGN KEY (NumDoc) REFERENCES DOC (NumDoc),

FOREIGN KEY (NumPat) REFERENCES PAT (NumPat)

)

Autre exemple :

DROP TABLE IF EXISTS clients;

CREATE TABLE IF NOT EXISTS clients(

dC INTEGER PRIMARY KEY,

nomC VARCHAR(40),

prenomP VARCHAR(40));

DESC clients;

Insertion de données

- **Syntaxe**

INSERT INTO <relation name>

[(attribute [,attribute] ...)]

{VALUES <value spec.> [, <value spec.>] ...| <query spec.>}

- **Exemples :**

- INSERT INTO DOC VALUES (1, 'Dupont', 'Paris');
- INSERT INTO DOC (NumDoc, NomDoc, VilleDoc) VALUES (1, 'Dupont', 'Paris');
- INSERT INTO DOC (NumDoc, NomDoc) VALUES (2, 'Toto');

Suppression de données

- **Syntaxe :**

DELETE FROM <relation_name>

[WHERE <search_condition>]

- **Exemples :**

Supprimer tous les docteurs

DELETE FROM DOC

Supprimer le docteur numéro 20

DELETE FROM DOC

WHERE NumDoc = 20

Modification de données

- SYNTAXE :

```
UPDATE <relation_name>
SET <attribute> = value_expression
[, <attribute> = value_expression ] ...
[WHERE <search condition> ];
```

- EXEMPLES :

Modifier la ville du docteur dont le numéro est 20

```
UPDATE DOC
SET VilleDoc = 'Valenton'
WHERE NumDoc = 20
```

Augmenter de 10% le prix de tous les médicaments dont le nom contient 'Antibio'

```
UPDATE MED
SET PRIX = PRIX*1.1
WHERE NomMed LIKE '%Antibio%'
```

Modification et suppression de table

Tables :

VITICULTEURS (NVT, NOM, PRENOM, VILLE, REGION)

VINS (NV, CRU, MILLESIME, DEGRE, #NVT, PRIX)

BUVEURS (NB, NOM, PRENOM, VILLE)

ABUS (#NV, #NB, DATE, QTE)

1) suppression de la table VINS

```
DROP TABLE VINS /.(DROP DATABASE BDDVIN)
```

4) ajout d'une colonne « prénom » dans la table BUVEURS

```
ALTER TABLE BUVEURS
ADD COLUMN ( prenom VARCHAR(15) );
```

5) modification du type quantité dans la table ABUS :

```
ALTER TABLE ABUS
MODIFY COLUMN ( QTE NUMBER(8,2) ); ( dans les anciennes versions )
```

6) supprimer la colonne QTE de la table ABUS :

```
ALTER TABLE ABUS
DROP COLUMN( QTE );
```

7) Ajouter une contrainte sur la quantité doit être > 0

```
ALTER TABLE ABUS
ADD CONSTRAINT CHECK ( QTE >0 )
```

7) renommer la table BUVEURS en PERSONNES

```
RENAME BUVEURS TO PERSONNES : ( Dans ALTER )
```

Structure générale d'une requête SQL

SELECT [DISTINCT/ALL] expression sur les attributs |*
FROM table [variable],table [variable], ...
[WHERE condition de sélection /sous-question]
[GROUP BY liste d'attributs]
[HAVING condition de sélection /sous-question]
[ORDER BY liste d'attributs];

- Restriction :
 - arithmétique (=, <, >, ≠, ≥, ≤, ...)
 - textuelle (LIKE)
 - sur intervalle (BETWEEN)
 - sur liste (IN)
- Possibilité de blocs imbriqués par :
IN, EXISTS, NOT EXISTS, ALL, SOME, ANY

LES VUES

Création d'une vue

*CREATE [OR REPLACE] [TEMP] VIEW nom
[(nom_colonne [...])]
AS requête
[WITH [CASCADED | LOCAL] CHECK OPTION]*

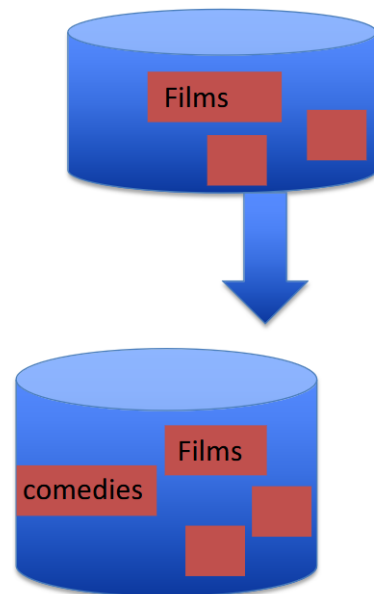
- **temp** : supprimée en fin de session
- **check option** : les conditions de la création doivent être respectées lors des INSERT et des UPDATE
- **local** : uniquement sur cette vue
- **cascaded** : sur toutes les vues-filles

LES VUES

Exemple simple

```
CREATE VIEW comedies AS
SELECT *
FROM films
WHERE genre = 'Comédie';

SELECT *
FROM comedies
WHERE sortie = 2010;
```

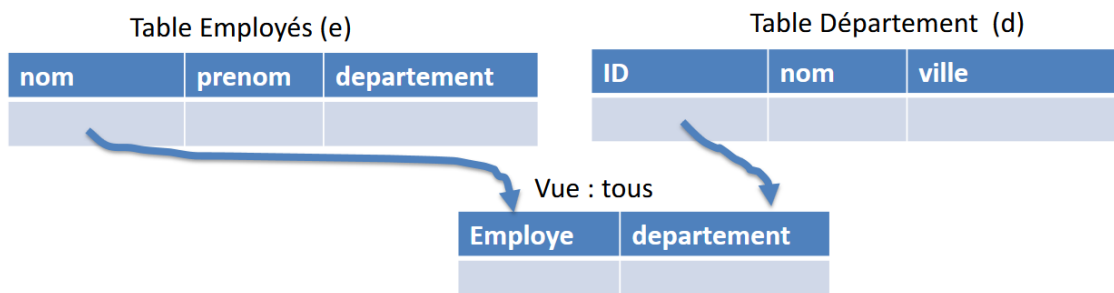


LES VUES

Exemple avec jointure

```
CREATE VIEW tous AS  
SELECT e.nom as Employe, d.nom as departement  
FROM Employes e, Departement d  
WHERE e.departement = d.id
```

```
SELECT * FROM tous;
```



LES VUES

Suppression d'une vue

```
DROP VIEW nom [...]  
[CASCADE | RESTRICT]
```

- **Cascade** : supprime aussi les objets qui dépendent de la vue
- **restrict** : refuse de supprimer la vue si un objet en dépend (valeur par défaut)

Droits / Privilèges

Niveau objets

objet = table, vue, fonction, procédures
des droits sur les objets (privilèges)

privilèges	Tables	Vues	Fonctions
SELECT	X	X	
UPDATE	X	X	
DELETE	X	X	
INSERT	X	X	
ALTER	X		
REFERENCES	X		
EXECUTE			X

Droits / Privilèges

Exemples avancés

On crée un utilisateur 'john'@'localhost', en lui donnant les privilèges SELECT, INSERT et DELETE sur la table `elevage.Animal`, et UPDATE sur les colonnes `nom`, `sexe` et `commentaires` de la table `elevage.Animal`.

```
GRANT SELECT,  
UPDATE (nom, sexe, commentaires),  
DELETE,  
INSERT  
ON elevage.Animal  
TO 'john'@'localhost' IDENTIFIED BY 'exemple2012';
```

Droits / Privilèges

Supprimer des droits

REVOKE privilege [, privilege, ...]

ON niveau_privilege

FROM utilisateur;

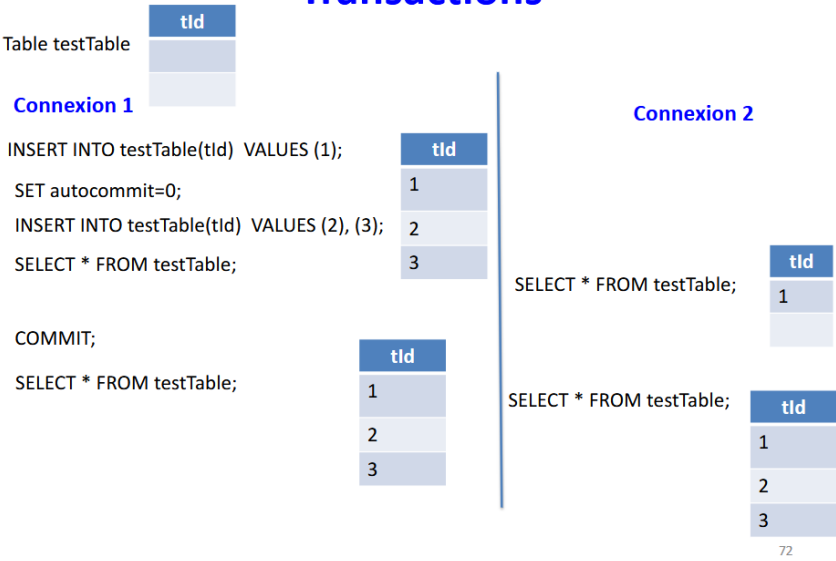
Exemple

REVOKE DELETE

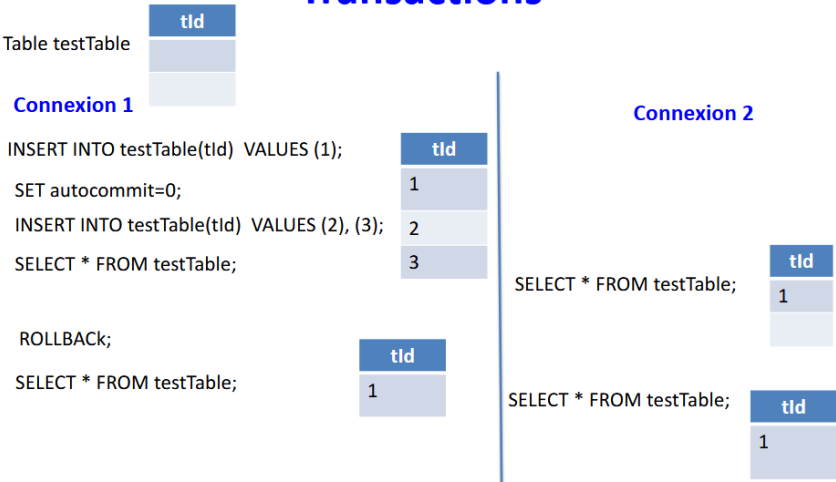
ON elevage.Animal

FROM 'john'@'localhost';

Transactions



Transactions



Fichier XML

- DTD : Exemple complet

```
<?xml version="1.1" encoding="UTF -8" standalone="yes" ?>
<! DOCTYPE film SYSTEM "modele.dtd" >
<film>
  <titre>
    <saga>Star Wars </saga>
    <opus>Le Retour du Jedi </opus>
  </titre>
  <auteur>George Lucas </auteur>
  <realisateur>Richard Marquand </realisateur>
  <acteur genre ="M">Mark Hamill </acteur>
  <acteur genre ="F">Carrie Fisher </acteur>
  <date_sortie>
    <jour>12 </jour>
    <mois>novembre </mois>
    <annee>1982 </annee>
  </date_sortie>
</film>
```

Fichier XML

- DTD : modele.dtd

```
<! ELEMENT film (titre , auteur , realisateur , acteur +,
date_sortie )>
<! ELEMENT titre (saga , opus )>
<! ELEMENT saga (# PCDATA )>
<! ELEMENT opus (# PCDATA )>
<! ELEMENT auteur (# PCDATA )>
<! ELEMENT realisateur (# PCDATA )>
<! ELEMENT acteur (# PCDATA )>
<! ATTLIST acteur genre (M|F) # REQUIRED >
<! ELEMENT date_sortie ( jour ?,mois , annee )>
<! ELEMENT jour (# PCDATA )>
<! ELEMENT mois (# PCDATA )>
<! ELEMENT annee (# PCDATA )>
```

Fichier XML

Exemples	Résultats
/film/acteur	<acteur genre="M">Mark Hamill</acteur>
	<acteur genre="M">Harrison Ford</acteur>
//acteur	<acteur genre="F">Carrie Fisher</acteur>
/film/titre/*	<saga>Star Wars</saga> <opus>Le Retour du Jedi</opus>
/film/titre	<titre> <saga>Star Wars</saga> <opus>Le Retour du Jedi</opus> </titre>
/film/titre/opus	<opus>Le Retour du Jedi</opus>
/film/titre/opus /text()	Le Retour du Jedi
//auteur //realisateur	<auteur> <nom>Lucas</nom> <prenom>George</prenom> </auteur> <realisateur>Richard Marquand</realisateur>

Fichier XML

Exemples	Résultats
/film/acteur[@genre="M"]	<acteur genre="M">Mark Hamill</acteur> <acteur genre="M">Harrison Ford</acteur>
/film/auteur[Nom="Lucas" and prenom="George"]	<auteur> <nom>Lucas</nom> <prenom>George</prenom> </auteur>
/film[auteur[Nom="Lucas"]]/titre	<titre> <saga>Star Wars</saga> <opus>Le Retour du Jedi</opus> </titre>
/film/acteur[2] ou bien /film/acteur[position()=2]	<acteur genre="M">Harrison Ford</acteur>
/film/acteur[last()]	<acteur genre="F">Carrie Fisher</acteur>

XML /

JSON

```
<agenda>
  <contact>
    <nom> Dupont </nom>
    <prenom> Noé </prenom>
    <tel-fixe> 01 23 45 67 89 </tel-fixe>
    <adresse>
      <voie> 42 rue de la paix </voie>
      <code-postal> 75001 </code-postal>
      <ville> Paris </ville>
    </adresse>
  </contact>
  <contact>
    <nom> Dupont </nom>
    <prenom> Léa </prenom>
    <tel-fixe> 01 23 45 67 89 </tel-fixe>
    <tel-mobile> 06 11 22 33 44 </tel-mobile>
    <adresse>
      <voie> 42 rue de la paix </voie>
      <code-postal> 75001 </code-postal>
      <ville> Paris </ville>
    </adresse>
  </contact>
</agenda>
```

```
{
  "agenda": {
    "contact": [
      {
        "nom": "Dupont",
        "prenom": "Noé",
        "tel-fixe": "01 23 45 67 89",
        "adresse": {
          "voie": "42 rue de la paix",
          "code-postal": "75001",
          "ville": "Paris"
        }
      },
      {
        "nom": "Dupont",
        "prenom": "Léa",
        "tel-fixe": "01 23 45 67 89",
        "tel-mobile": "06 11 22 33 44",
        "adresse": {
          "voie": "42 rue de la paix",
          "code-postal": "75001",
          "ville": "Paris"
        }
      }
    ]
  }
}
```

XPATH/JSON Path

Xpath	JSONPath	Description
/	\$	root object
.	@	current object
/	. or []	child operator
..	n/a	parent operator
//	..	recursive descent
*	*	joker
@	n/a	attribute access
[]	[]	subscript operator
n/a	[start :end :step]	array slice operator
[]	?()	définition d'un filtre

JSON

JSON Path

```
{
  "tool": {
    "json": {
      "creator": {
        "name": "D. Crockford",
        "location": [
          "Paris",
          "Rome"
        ]
      }
    }
  }
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- New document created with EditIX at Sun May 02 16:16:45 CEST 2021 -->
<tool>
  <json>
    <creator>
      <name>D. Crockford</name>
      <location>Paris</location>
      <location>Rome</location>
    </creator>
  </json>
</tool>
```

\$..tool.json.creator.location[1]
\$[tool][json][creator][location][1]

/tool/json/creator/location[2]/text()