Algèbre relationnelle / requêtes SQL

algèbre relationnelle	définition	équivalent SQL
attribut		nom de colonne
domaine de l'attribut		type des éléments de la colonne
schéma relationnel	n-uplet d'attributs deux à deux distincts	les entêtes des colonnes
relation, table	ensemble de n-uplets associés au schéma relationnel	table
enregistrement, valeur	un n-uplet associé au schéma relationnel	une ligne dans une table

Nom	algèbre relationnelle (le résultat est une relation)	définition	requête SQL (le résultat est une table sur laquelle on peut faire des requêtes)
Projection	$\pi_{(A_1,A_2)}(R_1)$	Choix de certaines colonnes	SELECT a1, a2 FROM table1;
Sélection	$\sigma_{A_1=\alpha}(R)$	Choix de certaines lignes	SELECT * FROM table WHERE a1 = α;

Attention ! La sélection se fait avec WHERE tandis que la projection se fait avec SELECT

Union	$R_1 \cup R_2$		UNION
Intersection	$R_1 \cap R_2$		INTERSECT
Différence	$R_1 \setminus R_2$		EXCEPT ou MINUS
Produit cartésien	$R_1 \times R_2$		SELECT * FROM table1, table2;
Jointure	$R_1 \underset{A_1=A_2}{\bowtie} R_2$	*idem* $\sigma_{A_1=A_2}(R_1 \times R_2)$	SELECT * FROM table1 JOIN table2 ON a1=a2;

Même si certains SGBD savent optimiser, il est préférable de faire une jointure plutôt qu'une sélection sur un produit cartésien

Renommage	$\rho_{A_1 \leftarrow B_1}(R)$	L'attribut A_1 s'appelle B_1	SELECT a1 AS b1 FROM table;
		Renommage d'une table	SELECT t.a1 FROM table AS t;
Suppression des doublons			SELECT DISTINCT ...;
Fonctions d'agrégation		nombre	SELECT COUNT(*) FROM table
			SELECT a1, COUNT(*) FROM table GROUP BY a1
		somme	SELECT a1, SUM(a2) FROM table GROUP BY a3
		maximum/minimum	MAX/MIN
		moyenne	AVG
Sélection suivant une fonction d'agrégation			SELECT * FROM table GROUP BY a2 HAVING SUM(a3) >= val;
Tri des résultats			SELECT a1, a2 FROM table GROUP BY a3 ORDER BY a2 ASC (ou DESC);

Interagir avec une base de données avec Python

On importe le module `sqlite3`

```
import sqlite3
```

On se connecte à la base

```
db_loc = sqlite3.connect('./base.sqlite')
```

On crée un curseur, outil de manipulation de la base de données, qui « pointe » dessus

```
cursor = db_loc.cursor()
```

On fait une requête SQL

```
resultat = cursor.execute("SELECT * FROM table WHERE cond;")
```

On en récupère le résultat que l'on peut alors afficher (ou en faire autre chose)

```
liste = resultat.fetchall()
print(liste)
```

On alimente la base avec une nouvelle donnée. Ici, pour une table à 3 attributs, on crée un nouvel enregistrement avec les valeurs `'foo'`, `'bar'` et 42

```
cursor.execute("INSERT INTO table VALUES (?, ?, ?);", ('foo', 'bar', 42))
```

On « commit » pour que ça soit effectif dans la base

```
db_loc.commit()
```

Pour voir l'effet sur la base, aller dans SQLite Manager, rafraîchir (ou reconnecter) la base.
On alimente la base avec plusieurs nouvelles données provenant par exemple d'un fichier `csv`

```
data = [
("foo", "bar", 42),
("bla", "bla", 17),
("bla", "blabla", 19),
]
cursor.executemany("INSERT INTO table VALUES (?, ?, ?);", data)
db_loc.commit()
```

Pour s'entraîner

On considère le modèle relationnel :

`etudiant(id, nom, prenom, adresse, CP, ville, telephone, classe)`

1. Afficher toutes les informations des étudiants
2. Afficher les noms et prénoms des étudiants
3. Afficher les villes d'où viennent les étudiants
4. Afficher les noms et prénoms des étudiants dont le code postal est 69380
5. Afficher les noms et numéros de téléphone des étudiants qui habitent Lyon ou Villeurbanne
6. Afficher les noms et prénoms des étudiants dont le numéro de téléphone commence par 07

On considère maintenant le modèle relationnel :

`prof(id, nom, prenom, discipline, bureau)`
`etudiant(id, nom, prenom, adresse, CP, ville, telephone, classe, id_prof_principal)`

où `id_prof_principal` réfère la relation `prof`

7. Afficher par un produit cartésien et une projection le nom des étudiants dont le prof principal est dans le bureau B014
8. Afficher par une jointure le nom des étudiants dont le prof principal est dans le bureau B014
9. Afficher le nombre d'étudiants dont le prof principal est prof de physique

On ajoute au modèle relationnel la relation :

`notes(id, info, physique, math, si, francais, langues, id_etudiant)`

où `id_etudiant` réfère la relation `etudiant`

10. Afficher le nombre de notes de langues qui sont strictement supérieures à 10
11. Afficher, pour chaque classe, la moyenne de math
12. Afficher, pour chaque classe, le nombre d'étudiants ayant une note de S.I. inférieure à 10
13. Afficher la meilleure note en info
14. Afficher le nom de l'étudiant ayant la meilleure note en info
15. Afficher les noms des étudiants ayant, dans leur classe, la meilleure note en francais
16. Afficher la liste des étudiants, par ordre alphabétique

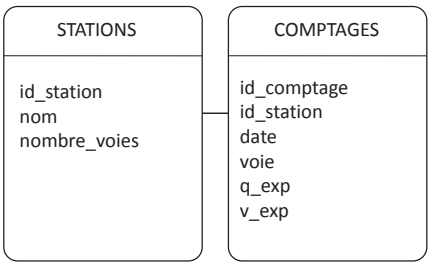
Extrait de cc-INP 2017

L'ensemble des données produites par le réseau est archivé dans une base de données. Les variables d'intérêt moyennées par tranche de temps de 6 min pour chaque point de mesure sont le débit *q_exp*, représentatif du nombre de véhicules par unité de temps et la vitesse moyenne *v_exp* des véhicules en ce point. On peut également accéder à la concentration, *c_exp*, par calcul étant donné que $c\_exp = q\_exp/v\_exp$.

Une version simplifiée de cette base de données est réduite à deux tables.

La table **STATIONS** répertorie les stations de mesures ; elle contient les attributs :

- *id_station* (clé primaire), entier identifiant chaque station ;
- *nom*, chaîne de caractères désignant le nom de la station ;
- *nombre_voies*, entier donnant le nombre de voies de la section d'autoroute.



La table **COMPTAGES** répertorie les différents enregistrements de données réalisés au cours du temps par les stations de comptage. Elle contient les attributs :

- *id_comptage*, entier identifiant chaque comptage ;
- *id_station*, entier identifiant la station concernée ;
- *date*, entier datant la mesure ;
- *voie*, entier numérotant la voie sur laquelle a été effectuée la mesure ;
- *q_exp*, flottant donnant le débit mesuré pendant 6 minutes ;
- *v_exp*, flottant donnant la vitesse moyenne mesurée pendant 6 minutes.

Q1. L'étude se focalise uniquement sur les mesures de l'une des stations, la *M8B*. Écrire une requête SQL qui renvoie les données de comptage (*id_comptage*, *date*, *voie*, *q_exp*, *v_exp*) mesurées à la station de comptage de nom *M8B*.

Le résultat de la requête précédente est stocké dans une nouvelle table **COMPTAGES_M8B** à cinq colonnes (*id_comptage*, *date*, *voie*, *q_exp*, *v_exp*).

On fait l'hypothèse que les mesures sur les différentes voies d'une même station sont enregistrées de façon synchronisée. Lors d'un enregistrement pour une station à trois voies, on écrit donc trois lignes dans la table **COMPTAGES** avec trois dates identiques. Pour chacun des enregistrements de la station *M8B*, trois lignes avec trois dates identiques sont donc présentes dans la nouvelle table **COMPTAGES_M8B**. Pour la suite de l'étude, les résultats expérimentaux de chacune des trois voies doivent être agrégés pour se ramener à une voie unique.

Q2. Écrire une requête SQL qui renvoie, pour chaque date des données de **COMPTAGES_M8B**, le débit correspondant à la somme des débits de chaque voie.

De la même façon, une requête SQL permet d'obtenir la moyenne des vitesses sur l'ensemble des trois voies pour chaque date des données de **COMPTAGES_M8B**. Il n'est pas demandé d'écrire cette requête. Ainsi, dans la suite de l'étude, la portion d'autoroute sera simplifiée en ne considérant qu'une seule voie.

Extrait de Mines-Ponts 2017

On modélise ici un réseau routier par un ensemble de croisements et de voies reliant ces croisements. Les voies partent d'un croisement et arrivent à un autre croisement. Ainsi, pour modéliser une route à double sens, on utilise deux voies circulant en sens opposés. La base de données du réseau routier est constituée des relations suivantes :

- Croisement(id, longitude, latitude)
- Voie(id, longueur, id_croisement_debut, id_croisement_fin)

Dans la suite on considère c l'identifiant (id) d'un croisement donné.

1. Écrire la requête SQL qui renvoie les identifiants des croisements atteignables en utilisant une seule voie à partir du croisement ayant l'identifiant c.
2. Écrire la requête SQL qui renvoie les longitudes et latitudes des croisements atteignables en utilisant une seule voie, à partir du croisement c.
3. Que renvoie la requête SQL suivante?

```
SELECT V2.id_croisement_fin
FROM Voie as V1
JOIN Voie as V2
ON V1.id_croisement_fin = V2.id_croisement_debut
WHERE V1.id_croisement_debut = c
```

Extrait de X-ÉNS 2017

Soit n un entier naturel. on note D_n l'ensemble des entiers naturels compris entre 0 et $2^n - 1$. On appelle « point de $D_n \times D_n$ » tout couple d'entiers $(x, y) \in D_n \times D_n$. Soient P et Q deux parties de $D_n \times D_n$. On cherche à calculer efficacement l'intersection des ensembles de points P et Q. La résolution de ce problème a des applications en simulation numérique, en robotique ou encore dans l'implémentation d'interfaces utilisateurs.
On suppose que l'on représente les points du problème à l'aide d'une base de données. Cette base comporte deux tables. La tables `POINTS` contient trois colonnes :

- `id` (clé primaire) qui est un entier naturel unique représentant le point ;
- `x` qui est un entier naturel représentant son abscisse ;
- `y` qui est un entier naturel représentant son ordonnée.

On suppose qu'il n'existe pas deux points d'identifiants distincts et de mêmes coordonnées.

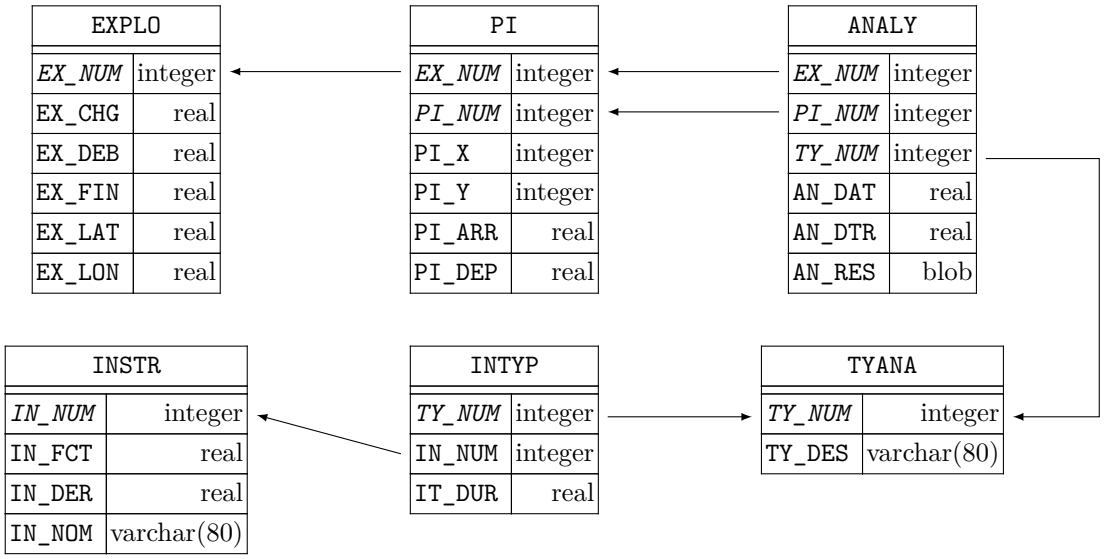
La relation d'appartenance à un ensemble de points estreprésentée par la table `MEMBRE` à deux colonnes :

- `idpoint`, un entier naturel qui identifie un point ;
- `idensemble`, un entier naturel qui identifie un ensemble de points.

1. Écrire une requête SQL qui renvoie les identifiants des ensembles auxquels appartient le point de coordonnées (a, b).
2. Écrire une requête SQL qui renvoie les coordonnées des points qui appartiennent à l'intersection des ensembles d'identifiants i et j.
3. Écrire une requête SQL qui renvoie les identifiants des points appartenant à au moins un des ensembles auxquels appartient le point de coordonnées (a, b).

Extrait de Centrale-Supélec 2017

Afin d'assurer son autonomie opérationnelle, le robot dispose localement des informations nécessaires à son fonctionnement quotidien. Ainsi, il enregistre la durée d'utilisation de ses différents instruments embarqués. Il connait également les différents types d'analyses qu'il peut effectuer et, pour chacun de ces types, les instruments à utiliser. Il enregistre la prochaine exploration, c'est-à-dire les différents points d'intérêts qu'il doit visiter et pour chacun la ou les analyses qu'il doit effectuer. D'autre part, il conserve les résultats d'analyses effectuées lors de ses explorations passées. Ces résultats ne sont effacés qu'après confirmation de leur bonne transmission sur Terre.
Ces différentes informations sont stockées dans une base de données relationnelle dont le modèle physique est schématisé à la figure suivante :



1 Structure physique de la base de données d'un robot

Cette base comporte les six tables suivantes :

- la table `EXPLO` des explorations, avec les colonnes
  - `EX_NUM` numéro (entier) de l'exploration (clef primaire)
  - `EX_CHG` date de transmission des points d'intérêts de cette exploration
  - `EX_DEB` date de début de l'exploration (`NULL` si l'exploration n'est pas encore commencée)
  - `EX_FIN` date de fin de l'exploration (`NULL` si l'exploration n'est pas encore terminée)
  - `EX_LAT` latitude (en degrés décimaux) du point de coordonnées $(0, 0)$ de la zone d'exploration
  - `EX_LON` longitude (en degrés décimaux) du point de coordonnées $(0, 0)$ de la zone d'exploration
- la table `PI` des points d'intérêts, de clef primaire `(EX_NUM, PI_NUM)`, avec les colonnes
  - `EX_NUM` numéro de l'exploration à laquelle appartient le point d'intérêt
  - `PI_NUM` numéro du point d'intérêt dans l'exploration (au sein d'une exploration les PI sont numérotés en séquence en commençant à 0, ce numéro n'a pas de rapport avec l'ordre dans lequel les PI sont explorés par le robot)
  - `PI_X` l'abscisse du point d'intérêt dans la zone d'exploration (entier positif en millimètres)

  - `PI_Y` l'ordonnée du point d'intérêt dans la zone d'exploration (entier positif en millimètres)
  - `PI_ARR` date d'arrivée du robot au point d'intérêt (`NULL` si ce point n'a pas encore été visité)
  - `PI_DEP` date à laquelle le robot a quitté le point d'intérêt (`NULL` si ce point n'a pas encore été exploré ou si la visite est en cours)

- la table `INSTR` des instruments embarqués, avec les colonnes
  - `IN_NUM` le numéro (entier) de l'instrument (clef primaire)
  - `IN_FCT` la durée pendant lequel l'instrument a déjà été utilisé depuis l'arrivée sur la planète (nombre décimal : fraction de jour martien)
  - `IN_DER` la date de la dernière utilisation de l'instrument
  - `IN_NOM` nom de l'instrument

- la table `TYANA` des types d'analyses à effectuer, avec les colonnes
  - `TY_NUM` le numéro de référence (entier) du type d'analyse (clef primaire)
  - `TY_DES` le nom du type d'analyse

- la table `INTYP` des instruments utilisés pour un type d'analyse, de clef primaire `(TY_NUM, IN_NUM)`, avec les colonnes
  - `TY_NUM` le numéro de référence (entier) du type d'analyse
  - `IN_NUM` le numéro (entier) de l'instrument
  - `IT_DUR` la durée standard d'utilisation de l'instrument dans ce type d'analyse (nombre décimal : fraction de jour martien)

- la table `ANALY` indiquant pour chaque point d'intérêt les types d'analyses à effectuer ou effectuées, de clef primaire `(EX_NUM, PI_NUM, TY_NUM)` et avec les colonnes
  - `EX_NUM` numéro de l'exploration à laquelle appartient le point d'intérêt
  - `PI_NUM` numéro du point d'intérêt dans l'exploration
  - `TY_NUM` type de l'analyse
  - `AN_DAT` date de l'analyse (`NULL` si l'analyse n'a pas été effectuée)
  - `AN_DTR` date de transmission sur Terre des résultats de l'analyse (`NULL` si l'analyse n'a pas été transmise)
  - `AN_RES` résultat de l'analyse (donnée opaque dont la signification dépend du type d'analyse)

Toutes les dates sont stockées sous forme d'un nombre décimal correspondant au nombre de jours martiens depuis l'arrivée du robot sur la planète.

1. Écrire une requête SQL qui donne le numéro de l'exploration en cours, s'il y en a une.
2. Écrire une requête SQL qui donne, pour une exploration dont on connait le numéro, la liste des points d'intérêts de cette exploration avec leurs coordonnées.
3. Écrire une requête SQL qui donne la surface, en mètres carrés, de chaque zone déjà explorée par le robot. La zone d'exploration est définie comme le plus petit rectangle qui englobe l'ensemble des points d'intérêts de l'exploration et dont les bords sont parallèles aux axes de référence (axes des abscisse et des ordonnées).
4. Quelle est la surface maximale d'une zone d'exploration que peut stocker cette base de données ?
5. Écrire une requête SQL qui donne, pour l'exploration en cours, le nombre de fois où chaque instrument doit être utilisé et sa durée d'utilisation théorique (en jours martiens) pour la totalité de l'exploration.