Exploitation d'une base de données

Département Informatique

IUT2 de Grenoble

BUT1 - Ressource 2.06

Introduction à l'exploration et l'administration des BD

1. Révisions : BD relationnelle et SQL simple
2. SQL pour explorer et analyser
3. SQL augmenté : *triggers*
4. Administration de 1er niveau

2. SQL pour explorer et analyser

2.1. Requêtes pour interroger

2.2. Requêtes pour explorer et analyser

2.3. Tables et vues définies par requêtes

2. SQL pour explorer et analyser

2.1. Requêtes pour interroger

2.2. Requêtes pour explorer et analyser

2.3. Tables et vues définies par requêtes

2.1. Requêtes pour interroger

1. Produit cartésien ou jointure
2. Sélection
3. Projection

```
SELECT prenom,nom
FROM membre
WHERE annee_inscr < 2010;
```

idm	prenom	nom	an._inscr
1	S	H	1990
2	H	P	2004
3	H	P	2020

2.1. Requêtes pour interroger

1. Produit cartésien ou jointure
2. Sélection
3. Projection

```
SELECT prenom,nom
FROM membre
WHERE annee_inscr < 2010;
```

idm	prenom	nom	an._inscr
1	S	H	1990
2	H	P	2004

2.1. Requêtes pour interroger

1. Produit cartésien ou jointure
2. Sélection
3. Projection

```
SELECT prenom,nom
FROM membre
WHERE annee_inscr < 2010;
```

prenom	nom
S	H
H	P

2. SQL pour explorer et analyser

2.1. Requêtes pour interroger

2.2. Requêtes pour explorer et analyser

2.3. Tables et vues définies par requêtes

2.2. Requêtes pour explorer et analyser

Objectif

- non pas seulement rendre l'information enregistrée
- mais l'analyser et la comprendre
- $\Rightarrow$ mettre en forme et ajouter de l'information.

Ajouter de l'information

- Fonctions d'agrégation : `min`, `max`, `avg`...
- Considérer des sous-ensembles : `GROUP BY` et `HAVING`

Mise en forme

- des tuples : `ORDER BY` et `LIMIT`
- de l'information : visualisation des données

2.2. Requêtes pour explorer et analyser

1. Produit cartésien ou jointure
2. Sélection
3. Groupement en sous-ensembles
4. Sélection des groupements
5. Projection et mise en ordre
6. Réduction de l'affichage


```
SELECT localisation, count(*) as nbs
FROM salon
WHERE num_salon > 1
GROUP BY localisation
HAVING count(*) > 1
ORDER BY localisation
LIMIT 1;
```

num_sal.	loc.	idm
1	N	1
2	E	null
3	S	2
4	S	null
5	W	1
6	W	null

2.2. Requêtes pour explorer et analyser

1. Produit cartésien ou jointure
2. Sélection
3. Groupement en sous-ensembles
4. Sélection des groupements
5. Projection et mise en ordre
6. Réduction de l'affichage

```
SELECT localisation, count(*) as nbs
FROM salon
WHERE num_salon > 1
GROUP BY localisation
HAVING count(*) > 1
ORDER BY localisation
LIMIT 1;
```

num_sal.	loc.	idm
2	E	null
3	S	2
4	S	null
5	W	1
6	W	null

2.2. Requêtes pour explorer et analyser

1. Produit cartésien ou jointure
2. Sélection
3. Groupement en sous-ensembles
4. Sélection des groupements
5. Projection et mise en ordre
6. Réduction de l'affichage

```
SELECT localisation, count(*) as nbs
FROM salon
WHERE num_salon > 1
GROUP BY localisation
HAVING count(*) > 1
ORDER BY localisation
LIMIT 1;
```

num_sal.	loc.	idm
5	W	1
6	W	null

num_sal.	loc.	idm
2	E	null

num_sal.	loc.	idm
3	S	2
4	S	null

2.2. Requêtes pour explorer et analyser

1. Produit cartésien ou jointure
2. Sélection
3. Groupement en sous-ensembles
4. Sélection des groupements
5. Projection et mise en ordre
6. Réduction de l'affichage

```
SELECT localisation, count(*) as nbs
FROM salon
WHERE num_salon > 1
GROUP BY localisation
HAVING count(*) > 1
ORDER BY localisation
LIMIT 1;
```

num_sal.	loc.	idm
5	W	1
6	W	null

num_sal.	loc.	idm
3	S	2
4	S	null

2.2. Requêtes pour explorer et analyser

1. Produit cartésien ou jointure
2. Sélection
3. Groupement en sous-ensembles
4. Sélection des groupements
5. Projection et mise en ordre
6. Réduction de l'affichage

```
SELECT localisation, count(*) as nbs
FROM salon
WHERE num_salon > 1
GROUP BY localisation
HAVING count(*) > 1
ORDER BY localisation
LIMIT 1;
```

loc.	nbs
S	2
W	2

2.2. Requêtes pour explorer et analyser

1. Produit cartésien ou jointure
2. Sélection
3. Groupement en sous-ensembles
4. Sélection des groupements
5. Projection et mise en ordre
6. Réduction de l'affichage

```
SELECT localisation, count(*) as nbs
FROM salon
WHERE num_salon > 1
GROUP BY localisation
HAVING count(*) > 1
ORDER BY localisation
LIMIT 1;
```

loc.	nbs
S	2

2.2. Requêtes pour explorer et analyser

Le *Select Club*

8. Ecrire une requête permettant de connaître le nombre de salons réservés pour chaque membre. L'affichage de l'information devra satisfaire les propriétés suivantes :
   - Le premier attribut doit être le nombre de salons réservés pour un membre, affiché sous l'intitulé `nbs` et pouvant être égal à zéro ;
   - Il sera complété par le prénom et le nom du membre : son identifiant ne sera pas affiché ;
   - Les tuples devront être affichés dans l'ordre décroissant du nombre de salons réservés.

2.2. Requêtes pour explorer et analyser

R2.08 statistiques descriptives

Exploration : **dplyr**

- `filter()` ↔ `WHERE`
- `arrange()` ↔ `ORDER BY`
- `mutate()` ↔ opération dans `SELECT`
- `summarise()` ↔ fonction d'aggrégation dans `SELECT`
- `group_by()` ↔ `GROUP BY`

2.2. Requêtes pour explorer et analyser

R2.08 statistiques descriptives

Visualisation de données : objectif possible de l'exploration

- nature de la donnée : catégorie vs. ordonnée vs. quantité
- sémantique de la donnée (temporelle, géographique, etc.)
- prendre en compte la perception :
  - théorie de la *Gestalt*
  - encoder les données dans des variables graphiques choisies en conséquence
- visualiser les données elles-mêmes ou une synthèse/analyse (via fonction d'agrégation, calcul statistique etc.)

2.2. Requêtes pour explorer et analyser

Exemples de graphiques selon l'analyse que vous voulez illustrer

👉 `https://img.labnol.org/di/choosing_a_good_chart2.pdf`

(A. Abela, *Advanced Presentations by Design : Creating Communication that Drives Action*, 2006)

Camembert

- › comparer des parties d'un tout
- ⚠ pas plus de 5 catégories
- ⚠ par pour comparer des graphiques
- ⚠ ordonner les triangles correctement
- ⚠ s'assurer que l'ensemble fait 100%

Bâtons

- › changement au fil d'instants discrets (vert.)
- › comparer des catégories (hor. si lab. longs)
- ⚠ choisir des couleurs cohérentes
- ⚠ demi-largeur d'un, entre deux bâtons
- ⚠ labels horizontaux

Bâtons cumulés

- › comparer des parties d'un tout
- › à même haut. si msg est % de chq. sous-cat

Lignes

- › séries temporelles avec données continues
- ⚠ pas plus de 4 lignes
- ⚠ poser un label sur chq courbe
- ⚠ 0-baseline si possible

Aires entre lignes

- › parties d'un tout au cours du temps
- ⚠ pas plus de 4 catégories

Nuages de points

- › relation entre éléments fondée sur 2 var.
- › corrélation sur un grand nb. de données
- ⚠ utiliser taille et non coul. pour autre var.

2. SQL pour explorer et analyser

2.1. Requêtes pour interroger

2.2. Requêtes pour explorer et analyser

2.3. Tables et vues définies par requêtes

2.3. Table définie par une requête (1/2)

Création d'une table comme résultat d'une requête

`CREATE TABLE nom AS requête;`

- standard : **`CREATE TABLE nom AS (requête) WITH [NO] DATA;`**
- schéma :
  - hérite nom et type des attributs de la requête.
  - pour redéfinir les noms :
    `CREATE TABLE nom(new_col1,new_col2) AS req;`

❓ répartition des données entre plusieurs tables après importation de l'ensemble dans une table unique.

Destruction d'une table

`DROP TABLE [IF EXISTS] nom;`

2.3. Table définie par une requête (2/2)

Création d'une table **temporaire** comme résultat d'une requête

`CREATE TEMP TABLE` *`nom`* `AS` *`requête`*`;`

- mêmes remarques que pour une table non temporaire ;
- ⊕ la table est automatiquement détruite à la fin de la session.

- ▶ utile si :
  - pas de cohérence à maintenir entre tables d'origine et table temp.
  - la conservation des données au-delà de la session n'est pas utile.

- ? remplacer une sous-req. utilisée plusieurs fois dans une session.
- ? table de travail.

2.3 Vue définie par une requête (1/2)

Création d'une vue comme résultat d'une requête

`CREATE VIEW nom AS requête;`

- s'utilise comme une table :
  - hérite nom et type des attributs de la requête.
  - pour redéfinir les noms :
    `CREATE VIEW nom(new_col1,new_col2) AS req;`
- mais les données ne sont pas copiées, seule la définition est stockée : la requête est exécutée à chaque utilisation de la vue.

- utile si :
  - cohérence à maintenir entre tables d'origine et vue.
  - conservation des données au-delà de la session souhaitée.

2.3 Vue définie par une requête (2/2)

- maintenir une organisation alternative des données.
- restreindre les droits d'accès à une relation.
- remplacer une sous-req. utilisée plusieurs fois.

En phase d'étude pour sa définition (essais-erreurs)

```
CREATE OR REPLACE VIEW nom AS requête;
```

Destruction

```
DROP VIEW [IF EXISTS] nom;
```