Les bases de données : Démo 1

Formation

Développeur Web et Web Mobile

Module : 04

Développer la partie back-end d'une application web

Séquence : 01

Utiliser une base de données

Séance : 03

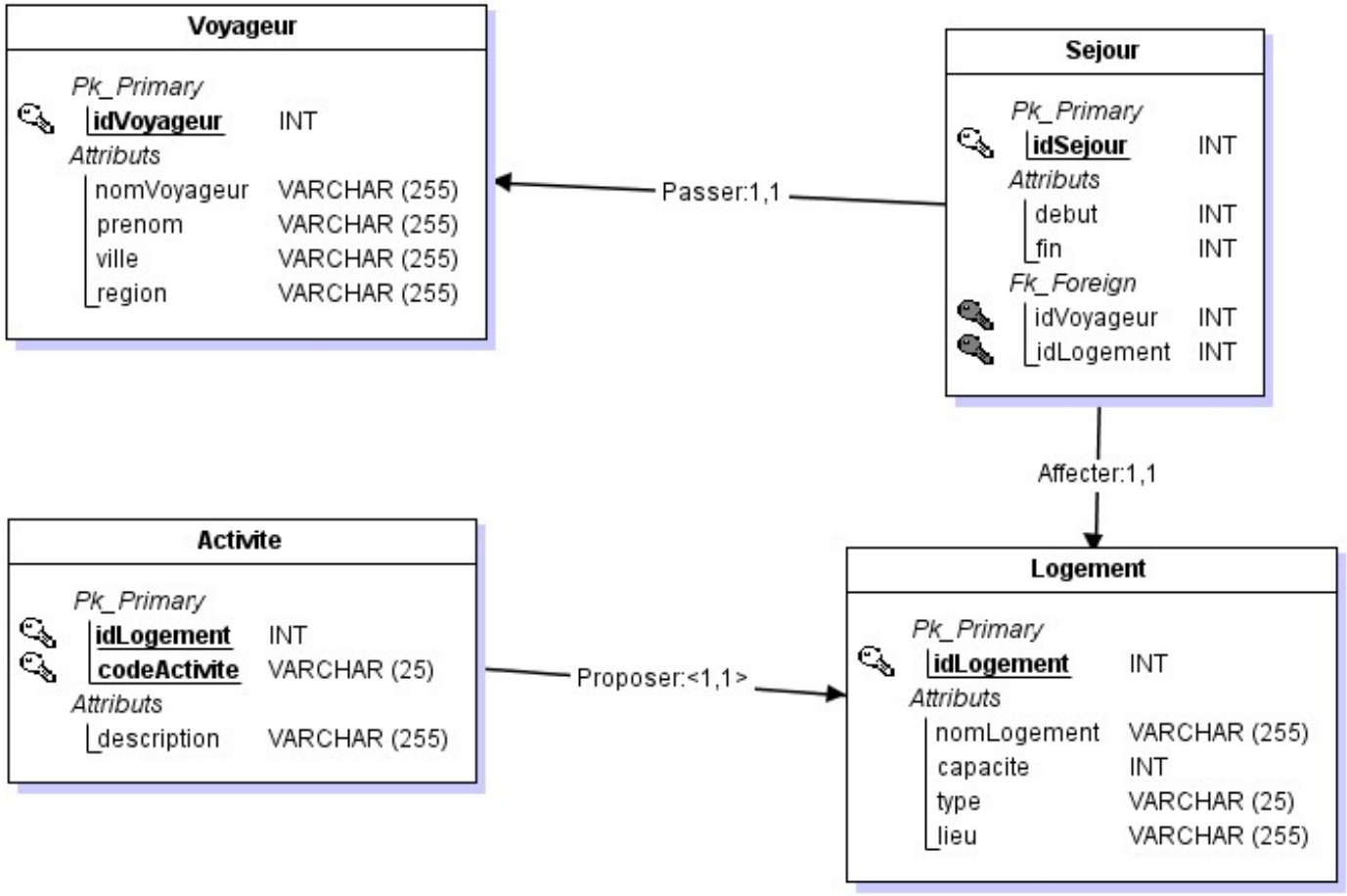
Ecrire des scripts en langage SQL

- Cette démonstration est basée sur l'utilisation d'une base de données simple provenant d'un cours du Cnam de Philippe Rigaux.
- Le schéma est le suivant :

  - voyageur (**idVoyageur**, nom, prenom, ville, region)
  - sejour (**idSejour**, *idVoyageur*, codeLogement, debut, fin)
  - logement (**code**, nom, capacite, type, lieu)
  - activite (**codeLogement**, **codeActivite**, description)

- En **gras**, les clés primaires, en *italiques* les clés étrangères

- Le MLD serai :


Voyageur
Pk_Primary
idVoyageur INT
Attributs
nomVoyageur VARCHAR (255)
prenom VARCHAR (255)
ville VARCHAR (255)
region VARCHAR (255)
Passer:1,1
Sejour
Pk_Primary
idSejour INT
Attributs
debut INT
fin INT
Fk_Foreign
idVoyageur INT
idLogement INT
Affecter:1,1
Activite
Pk_Primary
idLogement INT
codeActivite VARCHAR (25)
Attributs
description VARCHAR (255)
Proposer:<1,1>
Logement
Pk_Primary
idLogement INT
Attributs
nomLogement VARCHAR (255)
capacite INT
type VARCHAR (25)
lieu VARCHAR (255)

La table des Voyageurs et des Logements

idVoyageur	nom	prenom	ville	region
10	Fogg	Phileas	Ajaccio	Corse
20	Bouvier	Nicolas	Aurillac	Auvergne
30	David-Néel	Alexandra	Lhassa	Tibet
40	Stevenson	Robert Louis	Vannes	Bretagne

code	nom	capacite	type	lieu
ca	Causses	45	Auberge	Cévennes
ge	Génépi	134	Hôtel	Alpes
pi	U Pinzutu	10	Gîte	Corse
ta	Tabriz	34	Hôtel	Bretagne

La table des Séjours et des Activités

idSejour	idVoyageur	codeLogement	debut	fin
1	10	pi	20	20
2	20	ta	21	22
3	30	ge	2	3
4	20	pi	19	23
5	20	ge	22	24
6	10	pi	10	12
7	30	ca	13	18

codeActivite	codeLogement	description
Piscine	ge	Nage loisir non encadrée
Plongée	pi	Baptèmes et préparation des brevets
Randonnée	ca	Sorties d'une journée en groupe
Ski	ge	Sur piste uniquement
Voile	pi	Pratique du dériveur et du catamaran

- Le « SELECT » est constitué de 2 ou 3 clauses :

  SELECT expression
  FROM nom d'une table
  WHERE condition (cette clause est optionnelle)

- From : où rechercher (c'est toujours une table)
- Where : condition de restriction des enregistrements
- Select : construit un enregistrement résultat à partie de chaque enregistrement du from qui satisfait le where

- On veut récupérer le nom et le prénom d'un voyageur qui a le code d'identification 10 :

```
SELECT nom, prenom        <- projection
FROM voyageur
WHERE idVoyageur=10       <- restriction
```

1. On recherche sur la table « voyageur »
2. La condition porte sur l'attribut « idVoyageur » de cette table
3. On construit l'enregistrement résultat avec les attributs nom et prenom

- Résultat :

nom	prenom
Fogg	Phileas

- La condition de la clause « WHERE » peut être :
  - Une comparaison entre un attribut et une constante
    - égalité =
    - différence !=
    - infériorité, supériorité <, >
  - Une comparaison entre deux attributs
  - Elle peut aussi être multiple en utilisant les opérateurs AND, OR NOT et les parenthésages
- Exemple :

```
SELECT * FROM logement
WHERE lieu='Corse'
        OR (capacite > 50 AND type='Hôtel')
```

code	nom	capacite	type	lieu
ge	Génépi	134	Hôtel	Alpes
pi	U Pinzutu	10	Gîte	Corse

- La condition de la clause « WHERE » permet aussi de tester si la valeur d'un attribut appartient à un ensemble ou une liste.
- Pour cela il faut utiliser le mot clé « IN » suivi entre parenthèse de la liste des valeurs possibles.
- Exemple :

```
SELECT * FROM logement
WHERE lieu IN ('Corse', 'Bretagne')
        OR (capacite > 50 AND type='Hôtel')
```

code	nom	capacite	type	lieu
ge	Génépi	134	Hôtel	Alpes
pi	U Pinzutu	10	Gîte	Corse
ta	Tabriz	34	Hôtel	Bretagne

- La liste des valeurs possibles de la clause IN peut être le résultat d'un autre SELECT.
- On parle de Select imbriqués.
- Exemple :

```
SELECT nom FROM Logement
WHERE code IN
        (SELECT codeLogement FROM activite
                WHERE codeActivite='Ski')
```

nom
Génépi

- La clause « LIKE » permet de tester un champ de type chaine en utilisant des caractères de substitution
- Ces caractères sont au nombre de 2 :
  - Le carctère « _ » qui remplace un caractère unique
  - Le caractère « % » qui remplace une suite de caractères
- Exemple :

```
SELECT nom FROM voyageur
WHERE nom LIKE '_o%'
```

nom
Fogg
Bouvier

REQUETES D'INTERROGATION : le SELECT

la clause WHERE … BETWEEN

- La clause « BETWEEN » permet de tester un champ de type numérique est entre 2 valeurs

- Exemple :

```
SELECT * FROM logement
WHERE capacite BETWEEN 15 AND 50
```

code	nom	capacite	type	lieu
ca	Causses	45	Auberge	Cévennes
ta	Tabriz	34	Hôtel	Bretagne

- Il est possible de trier les lignes de la table résultante sur un ou plusieurs champ.
- Le tri se fait par ordre ascendant : ASC (par défaut) ou par ordre descendant : DESC
- Exemple :

```
SELECT nom FROM voyageur
ORDER BY nom
```

idVoyageur	nom ▲ 1	prenom	ville	region
20	Bouvier	Nicolas	Aurillac	Auvergne
30	David_Néel	Alexandra	Lhassa	Tibet
10	Fogg	Phileas	Ajaccio	Corse
40	Stevenson	Robert Louis	Vannes	Bretagne

- La clause « LIMIT » permet de limiter la sortie à un nombre de lignes maximales à partir d'une cretaine ligne.
- Elle se place à la fin suivi de deux valeurs numériques

  numéro de ligne de départ et nombre de lignes
- Exemple : 2 lignes à partir du début

```
SELECT nom FROM voyageur
ORDER BY nom LIMIT 1 2
```

idVoyageur	nom ▲ 1	prenom	ville	region
20	Bouvier	Nicolas	Aurillac	Auvergne
30	David_Néel	Alexandra	Lhassa	Tibet

les Alias AS

- Le renommage des attributs résultats s'exprime avec le mot clé « AS »
  - Si l'alias est un mot unique : pas de quottes
  - S'il est composé : utiliser les simples quottes
- L'alias permet de renommer soit les noms des champs ou le nom de la table
- Exemple :

```
SELECT idVoyageur AS id, prenom AS p, nom AS n
from voyageur AS v
where v.idVoyageur < 30
```

Utilisation de l'alias comme nom de table

id	p	n
10	Phileas	Fogg
20	Nicolas	Bouvier

REQUETES D'INTERROGATION : le SELECT

les doublons : DISTINCT

- Lors de la sélection, des doublons peuvent apparaitre dans le résultat.
- Si l'on veut les éliminer, il faut ajouter le mot clé « DISTINCT » après le select :
- Exemple :

```
SELECT type
FROM logement
```

type
Auberge
Hôtel
Gîte
Hôtel

```
SELECT DISTINCT type
FROM logement
```

type
Auberge
Hôtel
Gîte

- Il est possible de ramener une ou plusieurs valeurs qui sont le résultat d'une fonction appliquée sur la valeur d'un attribut de plusieurs enregistrements.
- Exemple :

```
SELECT count(*) AS nbrHotêl,
    sum(capacite) AS totalCapacité
FROM logement WHERE type='Hôtel'
```

nbrHotêl	totalCapacité
2	168

- La clause GROUP BY att1, …, attn regroupe les résultats en fonction des att1, …, attn
- Chaque groupe contient les enregistrements qui ont les mêmes valeurs pour att1, …, attn
- Exemple :

```
SELECT type, count(*) AS nombre,
            sum(capacite) AS totalCapacité
FROM logement GROUP BY type
```

type	nombre	totalCapacité
Auberge	1	45
Gîte	1	10
Hôtel	2	168

la clause HAVING

- Permet un filtrage après le GROUP BY
- Exemple :

```
SELECT type, count(*) AS nombre,
     sum(capacite) AS totalCapacité
FROM logement GROUP BY type
HAVING count(*) = 1
```

type	nombre	totalCapacité
Auberge	1	45
Gîte	1	10

```
SELECT type, count(*) AS nombre,
     sum(capacite) AS totalCapacité
FROM logement GROUP BY type
HAVING totalCapacité > 100
```

type	nombre	totalCapacité
Hôtel	2	168

requêtes multi-table : les jointures

- Une jointure entre 2 tables c'est le produit cartésien des tables avec une restriction (sélection).
- Plusieurs syntaxes existent et sont équivalentes pour faire une jointure :
  - Utiliser le mot clé JOIN entre le nom des 2 tables avec le mot clé ON suivi entre parenthèse d'une condition d'égalité sur un attribut de la 1ère table et un attribut de la seconde.
  - Utiliser le mot clé JOIN entre le nom des 2 tables avec une clause WHERE d'égalité entre un attribut de la 1ère table et un attribut de la seconde.
  - Ne pas utiliser le mot JOIN mais une virgule et la clause WHERE

- Attention : le ON ou le WHERE est obligatoire sinon on obtient le produit cartésien :
- Exemple :

SELECT * FROM sejour JOIN voyageur

idSejour	idVoyageur	codeLogement	debut	fin	idVoyageur	nom	prenom	ville	region
1	10	pi	20	20	10	Fogg	Phileas	Ajaccio	Corse
1	10	pi	20	20	20	Bouvier	Nicolas	Aurillac	Auvergne
1	10	pi	20	20	30	David-Néel	Alexandra	Lhassa	Tibet
1	10	pi	20	20	40	Stevenson	Robert Louis	Vannes	Bretagne
2	20	ta	21	22	10	Fogg	Phileas	Ajaccio	Corse
2	20	ta	21	22	20	Bouvier	Nicolas	Aurillac	Auvergne
2	20	ta	21	22	30	David-Néel	Alexandra	Lhassa	Tibet
2	20	ta	21	22	40	Stevenson	Robert Louis	Vannes	Bretagne
3	30	ge	2	3	10	Fogg	Phileas	Ajaccio	Corse
3	30	ge	2	3	20	Bouvier	Nicolas	Aurillac	Auvergne
3	30	ge	2	3	30	David-Néel	Alexandra	Lhassa	Tibet
3	30	ge	2	3	40	Stevenson	Robert Louis	Vannes	Bretagne
4	20	pi	19	23	10	Fogg	Phileas	Ajaccio	Corse
4	20	pi	19	23	20	Bouvier	Nicolas	Aurillac	Auvergne

- Sauf pour une jointure **naturelle** où la comparaison se fait entre la clé primaire de la table 1 et la clé étrangère de la table 2 (les noms doivent être identiques)
- Il faut ajouter le mot clé « NATURAL » avant le JOIN
- Exemple :

  SELECT * FROM sejour NATURAL JOIN voyageur

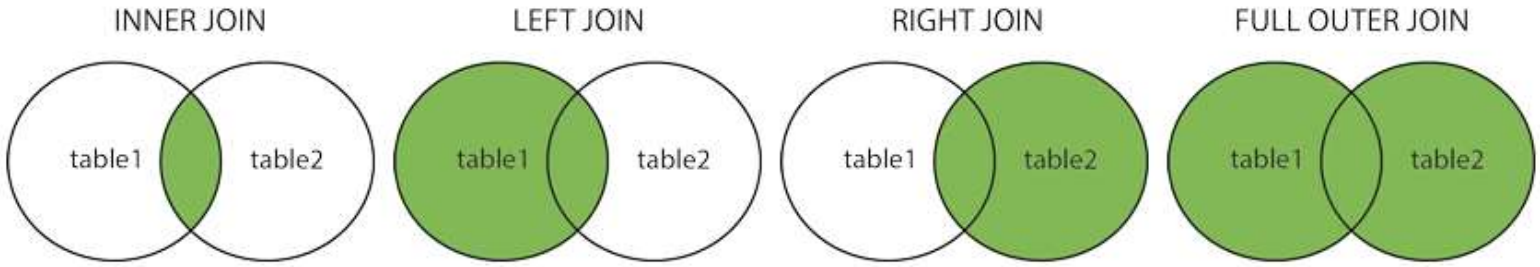
idVoyageur	idSejour	codeLogement	debut	fin	nom	prenom	ville	region
10	1	pi	20	20	Fogg	Phileas	Ajaccio	Corse
20	2	ta	21	22	Bouvier	Nicolas	Aurillac	Auvergne
30	3	ge	2	3	David-Néel	Alexandra	Lhassa	Tibet
20	4	pi	19	23	Bouvier	Nicolas	Aurillac	Auvergne
20	5	ge	22	24	Bouvier	Nicolas	Aurillac	Auvergne
10	6	pi	10	12	Fogg	Phileas	Ajaccio	Corse
30	7	ca	13	18	David-Néel	Alexandra	Lhassa	Tibet

Remarque : codeLogement apparaît qu'une fois

- Il existe plusieurs types de JOIN :
  - **(INNER) JOIN**: retourne les enregistrements des 2 tables qui correspondent à la condition
  - **LEFT (OUTER) JOIN**: retourne tous les enregistrements de la table de gauche et les enregistrements de la table de droite qui correspondent à la condition
  - **RIGHT (OUTER) JOIN**: retourne tous les enregistrements de la table de droite et les enregistrements de la table de gauche qui correspondent à la condition
  - **FULL (OUTER) JOIN**: retourne tous les enregistrements de la table de droite et de gauche qui correspondent à la condition

REQUETES D'INTERROGATION : le SELECT

requêtes multi-table : les différents JOIN



SELECT * FROM sejour AS s JOIN voyageur AS v ON (s.idVoyageur = v.idVoyageur)

idSejour	idVoyageur	codeLogement	debut	fin	idVoyageur	nom	prenom	ville	region
1	10	pi	20	20	10	Fogg	Phileas	Ajaccio	Corse
2	20	ta	21	22	20	Bouvier	Nicolas	Aurillac	Auvergne
3	30	ge	2	3	30	David-Néel	Alexandra	Lhassa	Tibet
4	20	pi	19	23	20	Bouvier	Nicolas	Aurillac	Auvergne
5	20	ge	22	24	20	Bouvier	Nicolas	Aurillac	Auvergne
6	10	pi	10	12	10	Fogg	Phileas	Ajaccio	Corse
7	30	ca	13	18	30	David-Néel	Alexandra	Lhassa	Tibet

SELECT * FROM sejour AS s RIGHT JOIN voyageur AS v ON (s.idVoyageur = v.idVoyageur)

idSejour	idVoyageur	codeLogement	debut	fin	idVoyageur	nom	prenom	ville	region
1	10	pi	20	20	10	Fogg	Phileas	Ajaccio	Corse
6	10	pi	10	12	10	Fogg	Phileas	Ajaccio	Corse
2	20	ta	21	22	20	Bouvier	Nicolas	Aurillac	Auvergne
4	20	pi	19	23	20	Bouvier	Nicolas	Aurillac	Auvergne
5	20	ge	22	24	20	Bouvier	Nicolas	Aurillac	Auvergne
3	30	ge	2	3	30	David-Néel	Alexandra	Lhassa	Tibet
7	30	ca	13	18	30	David-Néel	Alexandra	Lhassa	Tibet
NULL	*NULL*	*NULL*	*NULL*	*NULL*	40	Stevenson	Robert Louis	Vannes	Bretagne

requêtes multi-table : remarques

- Dans les requêtes précédentes, on a utilisé le renommage pour les tables par le mot clé « AS » et utiliser ces alias devant les noms d'attributs.
- Sans cela, comme dans les deux tables les attributs ont les mêmes noms, il y a ambiguïté pour SQL.
- Ces alias peuvent aussi être utilisés après le SELECT pour la projection (liste de champs).
- Exemple :

```
SELECT v.nom AS nomVoyageur, l.nom AS nomLogement
FROM voyageur AS v, logement AS l
…
```

REQUETES D'INTERROGATION : le SELECT

requêtes multi-table : plus de 2 tables

- On les énumère dans la clause FROM
- Exemple : le nom des voyageurs et des logements qu'ils ont occupés.

```
SELECT v.nom AS nomVoyageur, l.nom AS nomLogement
FROM logement AS l, sejour AS s, voyageur AS v
WHERE code = s.codeLogement
AND s.idVoyageur = v.idVoyageur
```

nomVoyageur	nomLogement
Fogg	U Pinzutu
Bouvier	Tabriz
David-Néel	Génépi
Bouvier	U Pinzutu
Bouvier	Génépi
Fogg	U Pinzutu
David-Néel	Causses

Remarque 1 : dans la clause from, on utilise le plus souvent la virgule à la place de join.

Remarque 2 : le renommage as sert, notamment, à gérer les ambiguités soulevées par les noms d'attributs.

requêtes multi-table : plus de 2 tables

- Exemple bis : le nom des voyageurs et des logements qu'ils ont occupés.

```
SELECT v.nom AS nomVoyageur,
            l.nom AS nomLogement
FROM voyageur as v NATURAL JOIN sejour AS s
JOIN logement As l ON l.code=s.codeLogement
```

les vues

- Toutes les requêtes que l'on vient de faire produisent en sortie une table virtuelle (une relation).
- Il est possible de demander au sgbdr de conserver cette relation en lui donnant un nom.
- On pourra ensuite interroger cette vue comme toute autre table.
- La création d'une vue se fait par la syntaxe :

  CREATE VIEW nomVue AS clause select
- L'interrogation se fera sur nomVue :

  SELECT * FROM nomVue

- Exemple :

```
CREATE VIEW LogementCorse AS
SELECT *
FROM logement
WHERE lieu='Corse'
```

- L'interrogation :

```
SELECT * FROM LogementCorse
```

code	nom	capacite	type	lieu
pi	U Pinzutu	10	Gîte	Corse

les vues

- Les vues sont des requêtes nommées que l'on peut traiter comme des tables.
- Elles permettent :
  - de faciliter l'accès (jointures pré-définies)
  - de restreindre la visibilité des données (on ne donne accès qu'à la vue)
- Attention :
  - Les insertions dans les vues sont soumises à fortes restrictions.
    - la vue doit être basée sur une seule table
    - on ne peut pas mettre à jour un attribut qui résulte d'un calcul ou d'une opération.