Formation Développeur Expert Java

SQL avec PostreSQL

IOCEAN
Les [illegible] dont les clients parlent avec plaisir

Module SQL

1. **Structure d'une base**
2. Interrogation de données
3. Requêtes multi-tables
4. Requêtes complexes
5. Manipulation de données
6. Langage de définition de données
7. Développement de fonctions
8. Triggers
9. Transactions
10. Administration

- Qu'est-ce qu'une base de données
- Les objets d'une base de données
- Les tables
- Notion d'intégrité de données

1. Structure d’une base

Module SQL

Objectif :

➔ Décrire les composants d'un SGBDR
- Qu'est-ce qu'une base de données relationnelle
- Les objets d'une base de données relationnelle
- Les tables et leurs caractéristiques
- La notion d'intégrité de données

Module SQL

Qu’est-ce qu’une base de données ?

- ➔ Permet de stocker la description des objets (par exemple les tables)
- ➔ Permet de stocker des données dans une ou plusieurs tables
- ➔ Permet de gérer l’accès aux données
- ➔ Permet d’assurer l’intégrité des données

Module SQL

Les objets d’une base de données ?

- Tables
- Index
- Vues
- Séquences
- Fonctions & procédures
- Triggers

Module SQL

Les tables et leurs caractéristiques

- ➔ Permettent de stocker les données
- ➔ Les tables possèdent un ensemble de colonnes
- ➔ Chaque colonne comporte un nom et un type de données

Module SQL

Type de données

chaîne de caractères	varchar(n), character varying(n) text
Entier	integer bigint
Numérique	numeric(precision) precision : nombre total de chiffres avant et après la virgule
Entier séquentiel	serial
Date & time	date time timestamp

Module SQL

Notion d’intégrité de données

- ➔ Une colonne peut être NULL / NOT NULL
- ➔ Une colonne est définie par un type de données et une taille

Module SQL

1. Structure d'une base
2. **Interrogation de données**
3. Requêtes multi-tables
4. Requêtes complexes
5. Manipulation de données
6. Langage de définition de données
7. Développement de fonctions
8. Triggers
9. Transactions
10. Administration

- Select
- Opérateurs arithmétiques
- Concaténation
- Clause where
- Agrégats
  - Group By
  - Fonctions d'agrégat
- Tri
- Fonctions de chaîne
- Conversion de type
- Fonctions de date
- Fonctions mathématiques
- Expression case

2. Interrogation de données

Module SQL

Objectif :

- ➔ Connaître la syntaxe basique de l'ordre SELECT
- ➔ Réaliser des extractions mono-table
- ➔ Se servir des fonctions de base

Module SQL

L’ordre SELECT

- Syntaxe de l’ordre SELECT
  - SELECT col1, col2, … FROM table
- Toutes les colonnes : *
- Suppression des lignes identiques
  - SELECT DISTINCT col1, col2, … FROM table
- Alias
  - SELECT col1 AS alias1 FROM table

Module SQL

Tris

- ➔ Les tris sont définis par ORDER BY
- ➔ Syntaxe
  - SELECT col1, col2 FROM table ORDER BY col1 [ASC|DESC], col2 [ASC|DESC]
  - SELECT col1, col2 FROM table ORDER BY n1 [ASC|DESC], n2 [ASC|DESC]
- ➔ Exemple
  - SELECT nom, prenom FROM abonne ORDER BY nom ASC
  - SELECT nom, prenom FROM abonne ORDER BY 1 ASC

Module SQL

Clause WHERE

- ➔ Permet de conditionner la sélection des lignes
- ➔ Syntaxe
  - ◆ SELECT col1,col2 FROM table WHERE condition

Clauses LIMIT et OFFSET

- ➔ Permet de limiter le nombre de lignes renvoyées
- ➔ Syntaxe
  - ◆ SELECT col1,col2 FROM table LIMIT n OFFSET n

Module SQL

Opérateurs logiques

Egal	=
Différent	<> ou !=
Inférieur	<
Supérieur	>
Inférieur ou égal	<=
Supérieur ou égal	>=
Faisant parti d'une liste de valeur	IN (val1, val2, ..)
Comprise entre 2 valeurs	BETWEEN val1 AND val2
Nulle	IS NULL (et pas = NULL)
Recherche sur une partie d'un mot (joker % et _)	LIKE

Module SQL

Négation et combinaisons

- Négation : NOT
- Combinaisions
  - Et : AND
  - Ou : OR

Module SQL

Exercice 1

- ➔ Installer pgAdmin
- ➔ Se connecter au serveur
  - host : formation.iocean.fr
  - user : formation[1-12]
  - pass : formationSQL

Module SQL

Exercice 2

➔ Récupérer tous les abonnés habitant à MONTPELLIER et les trier par ordre alphabétique (nom, prénom)

Module SQL

Exercice 3

➔ Lister tous les prénom différents qui commencent par la lettre L

Module SQL

Opérateurs arithmétiques

- ➔ Addition : +
- ➔ Soustraction : -
- ➔ Multiplication : *
- ➔ Division : /

Module SQL

Opérateurs de concaténation de chaîne

- ||
- Concat(val1, val2)

Module SQL

Fonctions de chaîne

Longueur de chaîne	length(chaine)
Conversion en minuscule	lower(chaine)
Conversion en majuscule	upper(chaine)
Partie de chaîne	substring(chaine, debut, nombre)
Début d'une chaîne	left(chaine, nombre)
Fin d'une chaîne	right(chaine, nombre)
Suppression des espaces de début et fin	trim(chaine)
Position d'une sous-chaîne	strpos(chaine, sous-chaine)
Remplacement d'une sous-chaine	replace(chaine, sous-chaine, nouv_sous-chaine)

Module SQL

Exercice 4

➔ Récupérer tous les abonnés habitant à Montpellier sans tenir compte des majuscules et minuscules et les trier par ordre alphabétique. Afficher sous la forme “NOM Prénom”

Module SQL

Fonctions de date

Date & heure système	now() : timestamp current_timestamp current_date current_time
Partie d'une date	date_part(partie, date) date_part(partie, interval) ex : date_part('year', now())
Age d'une date (par rapport à la date courante)	age(date) ex : age(cast('21-07-1969' as date))

Module SQL

Calculs sur les dates

- ➔ Postgresql gère la notion d'intervalle (type interval)
- ➔ Exemples :
  - ◆ select cast('1969-07-21' as timestamp) - cast('2007-02-17 12:15:24' as timestamp)
    - ● *interval : "-13725 days -12:15:24"*
  - ◆ select cast('2015-01-21' as date) + interval '2 days 3 hours 7 minutes'
    - ● *timestamp : "2015-01-23 03:07:00"*

Module SQL

Exercice 5

➔ Récupérer tous les abonnés dont l'abonnement est valide.

Module SQL

Exercice 6

➔ Récupérer tous les abonnés ayant moins de 20 ans

Module SQL

Agrégats

- ➔ Utilisation de GROUP BY pour regrouper les données
- ➔ Syntaxe
  - ◆ SELECT col1, fonction(col2) FROM table GROUP BY col1
- ➔ Explication
  - ◆ Retourne le résultat de fonction sur la colonne 2 pour toutes les valeurs distinctes de col1

Module SQL

Fonctions d'agrégat

- Moyenne : AVG
- Nombre d'enregistrements : COUNT
- Valeur la plus grande : MAX
- Valeur la plus petite : MIN
- Total : SUM
- Exemple
  - select count(*) from abonne
  - select ville, count(id) from abonne group by ville
  - Renvoi la liste des villes avec pour chacune le nombre d'abonné

Module SQL

Condition Having

- ➔ Permet de filtrer des résultats en appliquant un filtre sur un agrégat
- ➔ Exemple
  - select ville, count(id) from abonne group by ville having count(id) > 10
  - Renvoi la liste des villes ayant plus de 10 abonné avec pour chacune le nombre d'abonné

Module SQL

Exercice 7

➔ Compter le nombre d’abonnés

Module SQL

Exercice 8

➔ Compter le nombre d’abonnés entre 30 et 40 ans

Module SQL

Exercice 9

➔ Afficher le nombre d’abonné pour chaque ville et trier par ordre descendant

Module SQL

Exercice 10

➔ Limiter le résultat précédant aux villes ayant au moins 20 abonnés

Module SQL

Expression Case

- ➔ Permet de conditionner la valeur renvoyée
- ➔ Syntaxe :
  - ◆ CASE expression
    WHEN valeur1 THEN resultat1
    WHEN valeur2 THEN resultat2
    ELSE resultat
    END
  - ◆ CASE
    WHEN condition1 THEN resultat1
    WHEN condition2 THEN resultat2
    ELSE resultat
    END

Module SQL

Expression Case

➔ Exemples :

- SELECT CASE sexe
  WHEN ‘F’ THEN ‘Madame ’ || nom
  WHEN ‘M’ THEN ‘Monsieur ’ || nom
  ELSE nom END
  FROM personnes ;
- SELECT CASE
  WHEN sexe = ‘F’ THEN ‘Madame ’ || nom
  WHEN sexe = ‘M’ THEN ‘Monsieur ’ || nom
  ELSE nom END
  FROM personnes ;

Module SQL

Conversion de type

- ➔ Postgresql effectue de conversion implicite quand il le peut
  - ◆ exemple select 1 || ‘ mot’ retournera un text
- ➔ Fonction CAST
- ➔ Syntaxe
  - ◆ cast(valeur as type)
- ➔ Exemples
  - ◆ SELECT cast(10 as varchar)
  - ◆ SELECT cast(‘01/12/2015’ as date)

Module SQL

Fonctions de formatage

Vers du texte	to_char(timestamp, format) to_char(interval, format) to_char(integer, format) to_char(numeric, format)
Vers un nombre	to_number(text, format)
Vers une date	to_date(text, format)
Vers un timestamp	to_timestamp(text, format) to_timestamp(number)

- ➔ format est une chaîne contenant un format d'affichage
- ➔ Exemple :
  - to_date('01/12/2015', 'dd/mm/yyyy')
  - to_char(now(), 'HH24:MI')

Module SQL

Exercice 11

➔ Récupérer les abonnés dont le nom commence par A et afficher leur statut d'abonnement sous la forme "abonné qu'au dd/mm/yyyy" ou "expiré"

Module SQL

Traitement de la valeur nulle

- Fonction COALESCE
- Syntaxe
  - COALESCE(val1, val2, val3, …)
  - Renvoi la 1ere valeur non nulle
- Exemple
  - SELECT COALESCE(libelle_long, libelle, ‘vide’) FROM table1

Module SQL

Fonctions mathématiques

Modulo	% mod()
Valeur absolue	abs()
Arrondi	round(valeur, precision) ex : round(42.9876, 2) : 42.99
Troncature	trunc(valeur) ex : trunc(42.9876, 2) : 42.98
Valeur aléatoire	random()
Signe	sign(valeur) ex : sign(-12345) : -1 sign(123345) : 1

Module SQL

Exercice 12

➔ Compter le nombre de membre de chaque famille (même nom) et retourner pour chacune la date de naissance du plus jeune et du plus vieux

Module SQL

Exercice 13

➔ Calculer le nombre d’abonné par tranche d'âge (10-20 ans, 20-30 ans, …)

Module SQL

1. Structure d'une base
2. Interrogation de données
3. **Requêtes multi-tables**
4. Requêtes complexes
5. Manipulation de données
6. Langage de définition de données
7. Développement de fonctions
8. Triggers
9. Transactions
10. Administration

- Notion de jointure
- Produit cartésien
- Jointure
- Jointure externe

3. Requêtes multi-tables

Module SQL

Objectif :

- ➔ Comprendre la notion de jointure
- ➔ Réaliser des extractions sur plusieurs tables

Module SQL

Relation entre les tables

➔ Relation 1 à plusieurs

table : edition	
noEdition	clé primaire
edition	
adresse	
telephone	

relation
1 à n

- une maison d'édition peut avoir plusieurs livre
- un livre à qu'une seule maison d'édition

table : livre	
no	clé primaire
titre	
sujet	
auteur	
pages	
noEdition	clé étrangère

Module SQL

Relation entre les tables

table : eleve	
noEleve	clé primaire
nom	
prenom	
annee	

relation
m à n

- un élève a plusieurs cours
- un cours a plusieurs élèves

table : cours	
cote	clé primaire
titreCours	
description	

table : eleve	
noEleve	clé primaire
nom	
prenom	
annee	

2 relations 1 à n

1 à n **n à 1**

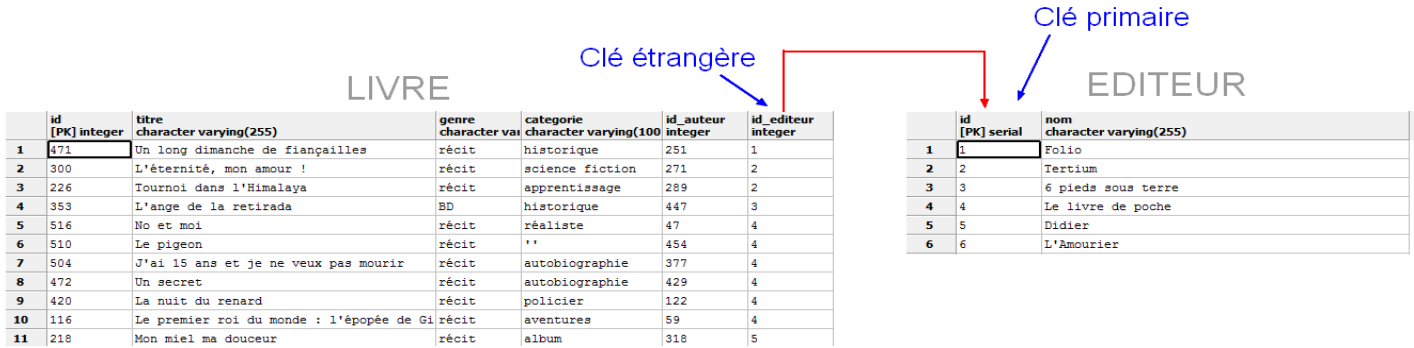
table : suitCours	
noEleve	clé primaire
cote	clé primaire

table : cours	
cote	clé primaire
titreCours	
description	

Module SQL

Notion de jointure

➔ Une jointure permet de “lier” 2 tables



	id [PK] integer	titre character varying(255)	genre character var	categorie character varying(100	id_auteur integer	id_editeur integer
1	471	Un long dimanche de fiançailles	récit	historique	251	1
2	300	L'éternité, mon amour !	récit	science fiction	271	2
3	226	Tournoi dans l'Himalaya	récit	apprentissage	289	2
4	353	L'ange de la retirada	BD	historique	447	3
5	516	No et moi	récit	réaliste	47	4
6	510	Le pigeon	récit	''	454	4
7	504	J'ai 15 ans et je ne veux pas mourir	récit	autobiographie	377	4
8	472	Un secret	récit	autobiographie	429	4
9	420	La nuit du renard	récit	policier	122	4
10	116	Le premier roi du monde : l'épopée de Gi	récit	aventures	59	4
11	218	Mon miel ma douceur	récit	album	318	5

	id [PK] serial	nom character varying(255)
1	1	Folio
2	2	Tertium
3	3	6 pieds sous terre
4	4	Le livre de poche
5	5	Didier
6	6	L'Amourier

➔Tous les éditeurs sont stockées dans la table “editeur”

➔Chaque livre est lié à un éditeur

Module SQL

Produit cartésien

- ➔ Il s'agit d'une jointure sans condition
- ➔ Syntaxe :
  - SELECT champ1, champ2
    FROM table1 AS alias1, table2 AS alias2, ...
- ➔ le résultat est la fusion des 2 tables :
  - chaque ligne de table1 est répétée autant de fois qu'il y a de ligne dans table2

Module SQL

Jointure avec condition

- ➔ Syntaxe non normalisée :
  - SELECT champ1, champ2
    FROM table1, table2
    WHERE table1.colonne = table2.colonne ;
- ➔ Syntaxe normalisée :
  - SELECT champ1, champ2
    FROM table1 JOIN table2 ON table1.colonne = table2.colonne

Module SQL

Jointure externe

- ➔ Cela permet d'afficher les lignes d'une table qui ne correspondent pas à la condition de jointure.
- ➔ Par exemple quand le champ de jointure est vide
- ➔ Syntaxe
  - ◆ SELECT champ1, champ2
    FROM table1 LEFT OUTER JOIN table2 ON table1.colonne = table2.colonne

Module SQL

Exercice 14

➔ Lister tous les livres avec leur auteur

Module SQL

Exercice 15

➔ Lister tous les livres avec leur auteur et leur éditeur

Module SQL

Exercice 16

➔ Afficher la liste des éditeurs avec le nombre de livre de chacun

Module SQL

Exercice 17

➔ Lister tous les livres actuellement empruntés avec leur emprunteur

Module SQL

Exercice 18

➔ Trouver les abonnés ayant emprunté un livre depuis plus de 2 mois et ne l'ayant pas rendu

Module SQL

Exercice 19

➔ Trouver les 10 abonnés ayant emprunté le plus de livres (rendus et non rendus)

Module SQL

Exercice 20

➔ Trouver les abonnés ayant plusieurs livres en leur possession

Module SQL

Exercice 21

➔ Lister les catégories les plus lues

Module SQL

1. Structure d'une base
2. Interrogation de données
3. Requêtes multi-tables
4. **Requêtes complexes**
5. Manipulation de données
6. Langage de définition de données
7. Développement de fonctions
8. Triggers
9. Transactions
10. Administration

- Opérateurs ensemblistes
- Sous-requêtes mono-ligne
- Sous-requêtes multi-lignes
- Sous-requêtes dans la clause from

4. Requêtes complexes

Module SQL

Objectif :

- ➔ Écrire des requêtes utilisant les opérateurs ensemblistes
- ➔ Comprendre la syntaxe des sous-requêtes
- ➔ Écrire des requêtes utilisant des sous-requêtes mono-lignes
- ➔ Écrire des requêtes utilisant des sous-requêtes multi-lignes
- ➔ Écrire des requêtes utilisant des sous-requêtes dans la clause FROM

Module SQL

Les opérateur ensemblistes

➔ UNION

- Permet d'obtenir un ensemble de lignes provenant de 2 requêtes
- requête1 UNION requête2

nom	*prénom*
Chose	Jules
Machin	Pierre
Truc	Patrick

Table1

union

nom	*prénom*
Pouf	Jean
Chose	Jules

Table2

=

nom	*prénom*
Chose	Jules
Machin	Pierre
Pouf	Jean
Truc	Patrick

Résultat

Module SQL

Les opérateur ensemblistes

➔ INTERSECT

- Permet d'obtenir un ensemble de lignes commun à 2 requêtes
- requête1 INTERSECT requête2

nom	*prénom*
Chose	Jules
Machin	Pierre
Truc	Patrick

inter

nom	*prénom*
Pouf	Jean
Chose	Jules

=

nom	*prénom*
Chose	Jules

Module SQL

Les opérateur ensemblistes

- ➔ EXCEPT
  - Permet d'obtenir un ensemble de lignes présent dans le résultat d'une requête, mais pas dans l'autre
  - requête1 EXCEPT requête2

nom	*prénom*
Chose	Jules
Machin	Pierre
Truc	Patrick

Except

nom	*prénom*
Pouf	Jean
Chose	Jules

=

nom	*prénom*
Machin	Pierre
Truc	Patrick

Module SQL

Les opérateur ensemblistes

- ➔ Les différentes requêtes doivent avoir le même nombre de champs
- ➔ Les champs doivent être de même type
- ➔ Les doublons sont supprimé (DISTINCT implicite)
- ➔ La clause Order By doit utiliser les numéros de colonne.

Module SQL

Sous requête mono-ligne

- ➔ Utilisation d'un Select renvoyant une ligne unique
- ➔ Peut être utilisé dans
  - ◆ Comme champ dans la clause SELECT
  - ◆ Comme valeur de condition dans la clause WHERE
- ➔ Syntaxe :
  - ◆ SELECT champ1, champ2, (sous-requête)
    FROM table
  - ◆ SELECT champ1, champ2
    FROM table
    WHERE champ opérateur(sous-requête)

Module SQL

Exercice 22

➔ Lister tous les abonnés avec le dernier livre qu'ils ont empruntés même s'ils n'ont jamais emprunté de livre (sous select dans le SELECT)

Module SQL

Exercice 23

➔ Lister les 10 livres les plus empruntés (avec le nombre d'emprunt)

Module SQL

Sous requête multi-lignes

- ➔ Utilisation d'un Select renvoyant plusieurs ligne dans la clause WHERE
- ➔ Utilisation des opérateur
  - IN
  - NOT IN
- ➔ Syntaxe :
  - SELECT champ1, champ2
    FROM table
    WHERE champ [NOT] IN (sous-requête)

Module SQL

Sous requête dans la clause FROM

- ➔ Permet d'imbriquer un SELECT dans une clause FROM
- ➔ Syntaxe :
  - ◆ SELECT champ1, champ2
    FROM (sous-requête)
    WHERE condition

Module SQL

Exercice 24

➔ Lister tous les abonnés avec le dernier livre qu'ils ont empruntés (sous-select dans la jointure)

Module SQL

Exercice 25

➔ Lister tous les livres de l’auteur "GUDULE" ou UBAC Claire

Module SQL

1. Structure d'une base
2. Interrogation de données
3. Requêtes multi-tables
4. Requêtes complexes
5. **Manipulation de données**
6. Langage de définition de données
7. Développement fonctions
8. Triggers
9. Transactions
10. Administration

- La commande INSERT
- La commande UPDATE
- La commande DELETE

5. Manipulation de données

Module SQL

Objectif :

- ➔ Savoir insérer des données dans la base
- ➔ Savoir modifier des données dans la base
- ➔ Savoir supprimer des données de la base

Module SQL

La commande INSERT

- ➔ Permet d'insérer des données dans la base
- ➔ Toutes les colonnes
  - ◆ INSERT INTO table VALUES(valeur_col1, valeur_col2, …)
- ➔ Colonne spécifiques
  - ◆ INSERT INTO table (col1, col2, …) VALUES(valeur_col1, valeur_col2, …)
  - ◆ Les colonnes non alimentées ne doivent pas être obligatoire (NOT NULL)
- ➔ Plusieurs lignes
  - ◆ INSERT INTO table (col1, col2, …) VALUES
    (valeur1_col1, valeur1_col2, …),
    (valeur2_col1, valeur2_col2, …)

Module SQL

Insertion à partir d'une requête

- ➔ Permet d'insérer des lignes dans une table à partir de données provenant d'une requête
- ➔ Syntaxe :
  - ◆ INSERT INTO table (col1, col2, …) SELECT col1b, col2b, ... FROM table2 ;

Module SQL

Exercice 26

➔ Ajouter une ligne dans la table abonné avec votre nom

Module SQL

Exercice 27

➔ Ajouter 3 personnes dans la table abonné en une seule requête

Module SQL

La commande UPDATE

- ➔ Permet de modifier les données d'une table
- ➔ Syntaxe :
  - ◆ UPDATE table SET col1 = valeur1, col2 = valeur2, …
    WHERE condition ;
- ➔ Les valeurs peuvent être des valeurs fixes ou des valeurs issues de sous requêtes

Module SQL

Exercice 28

➔ Mettre à jour la date de fin de votre abonnement à la date du jour + 1 an

Module SQL

La commande DELETE

- ➔ Permet de supprimer des lignes d'une table
- ➔ Syntaxe :
  - ◆ DELETE FROM table
    WHERE condition ;

Module SQL

Exercice 29

➔ Supprimer la ligne correspondant à votre nom dans la table abonné

Module SQL

La commande TRUNCATE

- ➔ Permet de supprimer TOUTES les lignes d'une table
- ➔ Syntaxe :
  - ◆ TRUNCATE table ;

Module SQL

Exercice 30

➔ Essayer d'insérer une ligne dans la table livre avec un id_auteur qui n’existe pas dans la table auteur

Module SQL

Exercice 31

➔ Remplir la table genre à partir des différents genre présents dans la table livre

Module SQL

Exercice 32

➔ Mettre à jour la colonne id_genre de la table livre en fonction de la valeur contenue dans la colonne genre

Module SQL

Exercice 33

➔ Essayer de supprimer une ligne de la table genre

Module SQL

Exercice 34

- ➔ Créer un nouveau genre
- ➔ Créer plusieurs livres ayant ce nouveau genre
- ➔ Supprimer tous les livres ayant ce genre

Module SQL

1. Structure d’une base
2. Interrogation de données
3. Requêtes multi-tables
4. Requêtes complexes
5. Manipulation de données
6. **Langage de définition de données**
7. Développement de fonctions
8. Triggers
9. Transactions
10. Administration

- Création de table
- Modification de table
- Suppression de table
- Les vues
- Les séquences
- Les indexes

6. Langage de définition de données (LDD)

Module SQL

Objectif :

➔ Connaître les commandes de manipulation de structure de table

Module SQL

Créer une table

- ➔ Utilisation de la commande CREATE TABLE
- ➔ Syntaxe :
  - ◆ CREATE TABLE table
    (
      col1 <type_de_la_colonne> <contrainte>,
      ...
) ;

Module SQL

Les contraintes de colonne

➔ Se placent après le type d'une colonne

- NOT NULL
  - *interdit la valeur nulle*
- DEFAULT valeur
  - *valeur par défaut du champ*
- CONSTRAINT nom PRIMARY KEY
  - *clé primaine*
- CONSTRAINT nom REFERENCES table (refcol)
  - *Clé étrangère*

Module SQL

Les contraintes de table

- ➔ Le placent après la liste des colonnes
  - CONSTRAINT nom PRIMARY KEY (col, ...)
  - CONSTRAINT nom FOREIGN KEY (col) REFERENCES table (col)

Module SQL

Les tables temporaires

- Une table temporaire jusqu'à ce que la session expire.
- Elle est visible uniquement à partir de la session qui l'a crée.
- Syntaxe :
  - CREATE TEMP TABLE table
    (
      col1 <type_de_la_colonne> <contrainte>,
      ...
) ;

Module SQL

Exercice 35

- Créer une table categorie
  - comportant un identifiant nommé id de type serial et une colonne nom_categorie
  - l'identifiant sera la clé primaire de la table

Module SQL

Modifier une table

- ➔ Ajout d'une colonne
  - ◆ ALTER TABLE table ADD col <type_de_la_colonne>;
- ➔ Modification d'une colonne
  - ◆ ALTER TABLE table ALTER COLUMN col TYPE <nouveau_type>;
- ➔ Suppression d'une colonne
  - ◆ ALTER TABLE table DROP col ;
- ➔ Ajout d'un contrainte
  - ◆ ALTER TABLE table ADD CONSTRAINT ...

Module SQL

Exercice 36

➔ Ajouter une colonne id_categorie dans la table livre et créer une contrainte d'intégrité vers la table categorie

Module SQL

Exercice 37

➔ Supprimer la colonne genre de la table livre

Module SQL

Supprimer une table

➔ DROP TABLE table ;

Module SQL

Les vues

- Objet logique auquel est rattaché une requête SELECT
- A chaque utilisation de la vue, la requête associée est exécutée
- Les données de la vue ne sont pas stockées
- Syntaxe :
  - CREATE VIEW nom_vue
    (col1, col2, …)
    AS SELECT …
- Suppression d'une vue :
  - DROP VIEW nom_vue ;

Module SQL

Exercice 38

➔ Créer une vue renvoyant les livres, leur éditeur et leur genre
- id
- titre
- nom_editeur
- nom_genre

Module SQL

Les séquences

- ➔ Compteur permettant de générer des numéros uniques
- ➔ Syntaxe :
  - ◆ CREATE SEQUENCE nom_sequence
    START WITH valeur_initiale
    INCREMENT val ;
- ➔ Utilisation :
  - ◆ SELECT nextval(‘nom_sequence’)

Module SQL

Les index

- ➔ S’applique à une ou plusieurs colonnes
- ➔ Permettent un accès plus rapide aux données.
- ➔ Permettent de forcer l’unicité
- ➔ Syntaxe :
  - ◆ CREATE [UNIQUE] INDEX nom_index
    ON table (col1, …) ;

Module SQL

Démonstration

La base formation0 contient 3,3 M de lignes

- ➔ Faire une recherche sur un abonné par son nom et noter le temps d'exécution
  - ◆ 1,2 seconde
- ➔ Refaire la même recherche
  - ◆ 800 ms
  - ◆ Cela va plus vite car les données ont été mises en cache par le moteur de base
- ➔ Créer un index sur la colonne nom de la table abonne
  - ◆ Cela va prendre environ 2 mn
- ➔ Refaire la recherche et comparer le temps d'exécution
  - ◆ 35 ms

Module SQL

Exercice 39

- ➔ Créer un index unique sur le nom, prénom, date de naissance de la table abonne
- ➔ Essayer d'ajouter une donnée déjà existante

Module SQL

1. Structure d’une base
2. Interrogation de données
3. Requêtes multi-tables
4. Requêtes complexes
5. Manipulation de données
6. Langage de définition de données
7. **Développement de fonctions**
8. Triggers
9. Transactions
10. Administration

- Notions sur les fonctions

7. Procédures stockées et fonctions

Module SQL

Objectif :

➔ Avoir des notions sur le développement de fonctions

Module SQL

Les fonctions personnalisées

- PostgreSQL permet la création de fonctions personnalisées
  - Equivalent aux procédures stockées avec d'autres bases de données
- Reçoivent des arguments en entrée
- Retournent une valeur ou un ensemble de valeur
- Permettent d'effectuer des opérations complexes en un seul appel
- plusieurs langages de développement sont disponibles (SQL, PL/pgsql, C)
- Elles s'utilisent comme les fonctions natives (left, date_part, …)

Module SQL

1. Structure d’une base
2. Interrogation de données
3. Requêtes multi-tables
4. Requêtes complexes
5. Manipulation de données
6. Langage de définition de données
7. Développement de fonctions
8. **Triggers**
9. Transactions
10. Administration

- Qu’est-ce qu’un trigger
- Types de triggers

8. Triggers

Module SQL

Objectif :

➔ Avoir des notions sur les triggers

Module SQL

Principes

- ➔ Un trigger la possibilité d'exécuter une fonction lors d'un évènement sur la table :
  - insert
  - update
  - delete
- ➔ Le déclenchement peut être effectué avant ou après l'événement
- ➔ Le déclenchement peut se faire une fois pour l'ensemble des lignes modifiées, ou une fois pour chaque ligne modifiée

Module SQL

1. Structure d’une base
2. Interrogation de données
3. Requêtes multi-tables
4. Requêtes complexes
5. Manipulation de données
6. Langage de définition de données
7. Développement de fonctions
8. Triggers
9. **Transactions**
10. Administration

- Qu’est qu’une transaction
- Gestion du rollback
- Verrous

9. Transactions

Module SQL

Objectif :

- ➔ Connaître le principe des transactions
- ➔ Savoir les gérer
- ➔ Connaître l’impact sur les verrous

Module SQL

Notion de transaction

- ➔ Une transaction permet de regrouper plusieurs ordre sql de mise à jour en un ensemble cohérent
- ➔ Toutes les mises à jour effectuées sont validées ensemble à la fin de la transaction (Commit)
- ➔ Si la transaction est annulée les actions déjà effectué sont annulées (Rollback)

Module SQL

Utilisation

➔ Syntaxe :

- BEGIN ;
  <sql>
  <sql>
  [COMMIT | ROLLBACK] ;

Module SQL

Verrouillage

- ➔ Lorsqu'un enregistrement est mis à jour, il est verrouillé (Lock) jusqu'à ce que la transaction soit terminée
- ➔ Les transactions doivent être courtes

Module SQL

Exercice 40

- ➔ Effectuer plusieurs mises à jour de la table abonne après avoir démarré une transaction
- ➔ Dans une autre session regarder les valeurs avant de valider la transaction, puis après
- ➔ Effectuer la même chose en annulant la transaction

Module SQL

1. Structure d’une base
2. Interrogation de données
3. Requêtes multi-tables
4. Requêtes complexes
5. Manipulation de données
6. Langage de définition de données
7. Développement de fonctions
8. Triggers
9. Transactions
10. **Administration**

- Installation de PostreSQL
- Utilisateurs
- Création / suppression de base
- Sauvegarde / Restauration

10. Administration

Module SQL

Objectif :

- ➔ Savoir installer PostgreSQL
- ➔ Connaître les actions d'administration de base

Module SQL

Installation

- ➔ L'installer windows est disponible sur le site de postgresql
  - ◆ http://www.postgresql.org/
- ➔ Installation :
  - ◆ Choix du répertoire d'installation
  - ◆ Choix du répertoire de stockage de données
  - ◆ Choix du mot de passe du compte postgres
  - ◆ Choix du port d'écoute
  - ◆ Choix du jeu de caractères par défaut

Module SQL

Exercice 41

- ➔ Installer PostreSQL en local
- ➔ S’y connecter avec pgadmin

Module SQL

Gestion des utilisateurs

- ➔ Via pgAdmin
  - ◆ Dans le groupe “Rôle de connexion”
- ➔ En SQL :
  - ◆ CREATE ROLE formation LOGIN ENCRYPTED PASSWORD 'abcdewxyz' ;

Module SQL

Création d'une base

- ➔ Via pgAdmin
  - ◆ Dans le groupe "Bases de données"
  - ◆ Une base possède un propriétaire
- ➔ En SQL :
  - ◆ CREATE DATABASE <ma_base>
    WITH OWNER = <login_propriétaire>
    ENCODING = 'UTF8'
    LC_COLLATE = 'fr_FR.UTF-8'
    LC_CTYPE = 'fr_FR.UTF-8' ;

Module SQL

Suppression d’une base

- ➔ Il ne doit pas y avoir de connexion active sur une base pour pouvoir la supprimer
- ➔ Via pgAdmin
  - ◆ Dans le groupe “Bases de données”
- ➔ En SQL :
  - ◆ DROP DATABASE <ma_base> ;

Module SQL

Exercice 42

➔ Créer une base “formation”, accessible par un utilisateur “formation”

Module SQL

Sauvegarde de base

- Il existe plusieurs formats de sauvegarde :
  - PLAIN (fichier plat) : fichier sql contenant la structure et les données
  - TAR : archive contenant un fichier sql pour la structure et un fichier de données par table
  - CUSTOM : fichier compressé spécifique à postgresql
- Format Custom le plus adapté pour pouvoir restaurer facilement
- Via pgAdmin
  - Clic droit sur la base, puis Sauvegarder.
  - Sélectionner les options de sauvegarde
- En ligne de commande
  - pg_dump -h localhost -F p|t|c -U <user> -W -f <nom_fichier> <nom_base>
  - format : p : plain, t : tar, c : custom

Module SQL

Exercice 43

- Sauvegarde la base formation du serveur distant.
  - avec pg_admin

Module SQL

Restauration de base

- ➔ Restauration des formats TAR ou CUSTOM
- ➔ Via pgAdmin
  - ◆ Clic droit sur la base, puis Restaurer.
  - ◆ Sélectionner les options de restauration
- ➔ En ligne de commande
  - ◆ pg_restore -h localhost -F t|c -U <user> -W -f <nom_fichier> -d <nom_base>
  - ◆ format : t : tar, c : custom
  - ◆ Options possible :
    - *-O : ne pas restaurer le propriétaire des objets*
    - *-c : Supprimer les objets de la base avant restauration*

Module SQL

Exercice 44

- Restaurer la base formation locale à partir d'un backup.
  - avec pg_admin

Restons en contact.

DIGINAMIC
Lionel Cabon, Directeur
contact@diginamic.fr

IOCEAN
Les projets informatiques dont les clients parlent avec plaisir

Nos coordonnées : 04 34 09 04 60
contact@iocean.fr - www.iocean.fr

N° Déclaration OF : 91 34 08867 34
Nantes, Paris, Montpellier
www.diginamic.fr