

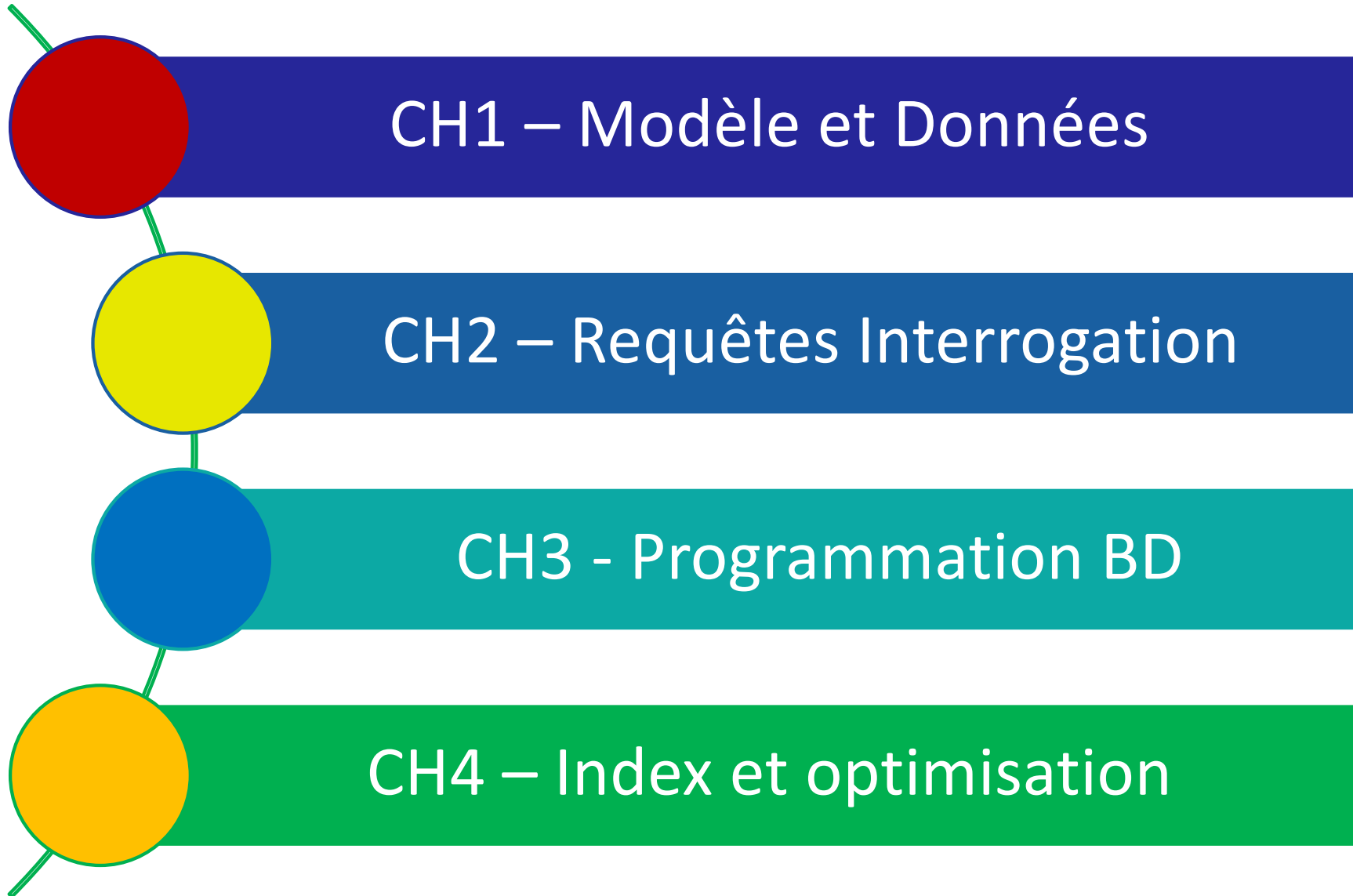
UNIVERSITE DE CORSE
2025-2026
Master DFS-DE 1^{ère} année

Bases de données Rel et optimisation
CH1 - Modèle et données :
Définition et manipulation



Evelyne VITTORI
vittori@univ-corse.fr

Optimisation et Bases de données



CH1- Modèle et Données

Objectifs

- Consolider et approfondir ses connaissances sur les concepts du **modèle** relationnel
- Maîtriser et optimiser **l'implémentation** d'un schéma de données
- Maîtriser le langage DML pour les **mise à jour**
- Savoir créer et manipuler des **vues**
- S'initier à la gestion des privilèges et des transactions



CH1- Modèle et Données

Déroulement du cours

- Ce chapitre contient de nombreux rappels qui ne seront pas présentés intégralement pendant les séances.
- Pour chaque section:
 1. Des questions et des exercices pour tester ses connaissances
 2. Retour sur les définitions et explications si nécessaire



CH1 - Modèle et Données

1. Modèle relationnel et intégrité
2. Gestion des privilèges
3. Langage SQL DDL
4. Langage SQL DML (Mises à jour)
5. Gestion des transactions





Le modèle relationnel

Pourquoi une relation n'est pas une simple table?

Quels sont les concepts à connaître?

A quoi faut-il faire attention?



UNIVERSITÀ DI CORSICA
PASQUALE PAOLI

BD relationnelle: Ensemble de tables?

SKIPERS	SKNUM	SKNOM	SKPORT	SALAIRE
	1	JEAN	AJACCIO	3000
	2	PAUL	AJACCIO	2000
	3	MARINE	ANTIBES	2000

Colonnes
Attributs
champs

Clés
primaires
Clés
étrangères

BATEAUX	BATNUM	BATNOM	BATPORT	CAPACITE
	B001	LIBERTE	ANTIBES	10
	B002	LOUISIANE	BASTIA	12
	B003	KALISTE	BASTIA	6
	B004	INDEPENDANCE	CALVI	6

CROISIERS	CROISNUM	SKNUM	BATNUM	DEPPORT	ARRPORT	DEPDATE	ARRDATE
	C001	1	B002	BASTIA	CALVI	10/07	11/07
	C002	2	B002	ANTIBES	AJACCIO	15/07	16/07
	C003	1	B003	AJACCIO	BASTIA	21/08	22/08
	C004	3	B004	CALVI	MARSEILLE	02/09	03/09



Question N°1

Domaine sémantique



Selon le modèle relationnel, les attributs (colonnes) d'une table devraient être définis sur un domaine dit « sémantique ».

- Qu'est-ce qu'un domaine sémantique par rapport à la notion de type syntaxique?

Une Relation = Une table?

Schéma (ou structure) de la table

<i>Domaines</i> <i>Types syntaxiques</i> <i>Colonnes (Attributs)</i>	Numéro de bateau Chaîne de 5 caractères	Nom de bateau Chaîne de 20 caractères	Nom de port Chaîne de 20 caractères	Nombre de Passagers Nombre Entier
	BATNUM	BATNOM	PORT	CAPACITE
	B002	LOUISIANE	BASTIA	12
	B003	KALISTE	BASTIA	6
	B004	INDEPENDANCE	CALVI	6

Relation BATEAUX

Une Relation = un ensemble

- DOMAINE : ensemble de valeurs caractérisé par un nom

Domaine sémantique \neq type syntaxique

- CAPACITE= nombre entier représentant un nombre de passagers
- PORT= {AJACCIO, MARSEILLE, BASTIA, ANTIBES, CALVI, AJACCIO, BASTIA }

- RELATION : sous-ensemble du produit cartésien de n domaines ($R \subseteq D1 \times D2 \times \dots \times Dn$)

– BATEAUX \subseteq BATNUM \times BATNOM \times PORT \times CAPACITE

n= degré de la relation

Une Relation = un ensemble de tuples

Conséquences?

Les tuples d'une relation ne sont **pas ordonnés** les uns par rapport aux autres.

Tuple= terme anglais désignant un n-uplet

Absence de répétition: les tuples d'une relation sont **deux à deux distincts**.

La plupart des SGBD ne l'assurent pas!

Une vision logique: une Relation = un prédicat

- Un schéma de RELATION est un **prédicat à n variables** écrit sous forme concise
 - BATEAUX(BATNUM, BATNOM, PORT,CAPACITE)
- Un TUPLE est une **proposition** écrite sous forme concise
 - BATEAUX (B001, LIBERTE, ANTIBES, 10)

Conséquences? Langages relationnels

Domaines et attributs

SKIPERS	SKNUM	SKNOM	PORT	SALAIRE
	SKNUM	SKNOM	SKPORT	SALAIRE
	1	JEAN	AJACCIO	3000
	2	PAUL	AJACCIO	2000
	3	MARINE	ANTIBES	2000

Attribut = rôle joué par un domaine dans une relation

Attributs comparables =
définis sur le même
domaine

BATEAUX	BATNUM	BATNOM	PORT	CAPACITE
	BATNUM	BATNOM	BATPORT	CAPACITE
	B001	LIBERTE	ANTIBES	10
	B002	LOUISIANE	BASTIA	12
	B003	KALISTE	BASTIA	6
	B004	INDEPENDANCE	CALVI	6

CROISIERES	CROISNUM	SKNUM	BATNUM	PORT	PORT	DATE	DATE
	CROISNUM	SKNUM	BATNUM	DEPPORT	ARRPORT	DEPDATE	ARRDATE
	C001	1	B002	BASTIA	CALVI	10/07	11/07
	C002	2	B002	ANTIBES	AJACCIO	15/07	16/07
	C003	1	B003	AJACCIO	BASTIA	21/08	22/08
	C004	3	B004	CALVI	MARSEILLE	02/09	03/09



Question N°2


Clés primaires



- 1) Qu'est-ce qu'une clé primaire?
- 2) Quel est l'intérêt de définir une **clé primaire** dans une table?
- 3) L'existence de clés primaires a-t-elle une influence sur la définition et l'exécution:
 - des requêtes de mise à jour de la base de données (insertion, modification de tuples)?
 - des requêtes d'interrogation?

Notion de clé primaire

- Un DOMAINE PRIMAIRE est un domaine de définition d'un attribut clé primaire mono-attribut.



	<u>BATNUM</u>	BATNOM	PORT	CAPACITE
BATEAUX	BATNUM	BATNOM	BATPORT	CAPACITE
	B001	LIBERTE	ANTIBES	10
	B002	LOUISIANE	BASTIA	12
	B003	KALISTE	BASTIA	6
	B004	INDEPENDANCE	CALVI	6

- Une CLE PRIMAIRE est un attribut ou groupe d'attributs dont les valeurs permettent de distinguer les tuples les uns des autres

Valeurs uniques et non nulles: Rôle d'identifiant

Clés primaires multi-attributs

Clé Primaire composée de 2 attributs

PASSAGERS	CROISNUM	PASNOM	COUT	COUT
	CROISNUM	PASNOM	PRIX	REMISE
	C001	DUPONT	200	20
	C001	MAC CARTY	150	50
	C001	MAC DOUGLAS	150	10
	C002	LAUREL	300	60
	C002	HARDY	300	60
	C003	DURAND	250	60
	C003	DUBOIS	250	60
	C004	DUPONT	300	60
	C004	LAUREL	300	80
	C004	HARDY	300	60



Question N°3 (1)

Clés étrangères



- 1) Qu'est-ce qu'une clé étrangère?
- 2) Quel est l'intérêt de définir une clé étrangère dans une table?
- 3) Qu'est-ce que la notion d'intégrité référentielle?
- 4) A quel moment l'intégrité référentielle est-elle vérifiée?



Question N°3 (2)

Clés étrangères



- 5) L'existence de clés étrangères a-t-elle une influence sur la définition et l'exécution:
- Des requêtes de mise à jour de la base de données (insertion, modification de tuples)?
 - Des requêtes d'interrogation?
- 6) La présence de clés étrangères induit des contraintes sur l'ordre d'insertion des tuples dans les tables d'une base de données relationnelle. Quel est cet ordre?



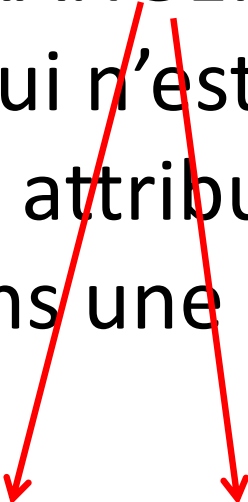
Question N°3 (3) Clés étrangères



7) Il peut parfois être utile de désactiver temporairement la vérification de l'intégrité référentielle. Dans quels cas?

Clés étrangères

- Une CLE ETRANGERE est un attribut (ou groupe d'attribut) qui n'est pas clé primaire mais qui est associé à un attribut (ou groupe d'attributs) clé primaire dans une autre relation.



	CROISNUM	SKNUM	BATNUM	PORT	PORT	DATE	DATE
CROISIERS	CROISNUM	SKNUM	BATNUM	DEPPORT	ARRPORT	DEPDATE	ARRDATE
	C001	1	B002	BASTIA	CALVI	10/07	11/07
	C002	2	B002	ANTIBES	AJACCIO	15/07	16/07
	C003	1	B003	AJACCIO	BASTIA	21/08	22/08
	C004	3	B004	CALVI	MARSEILLE	02/09	03/09

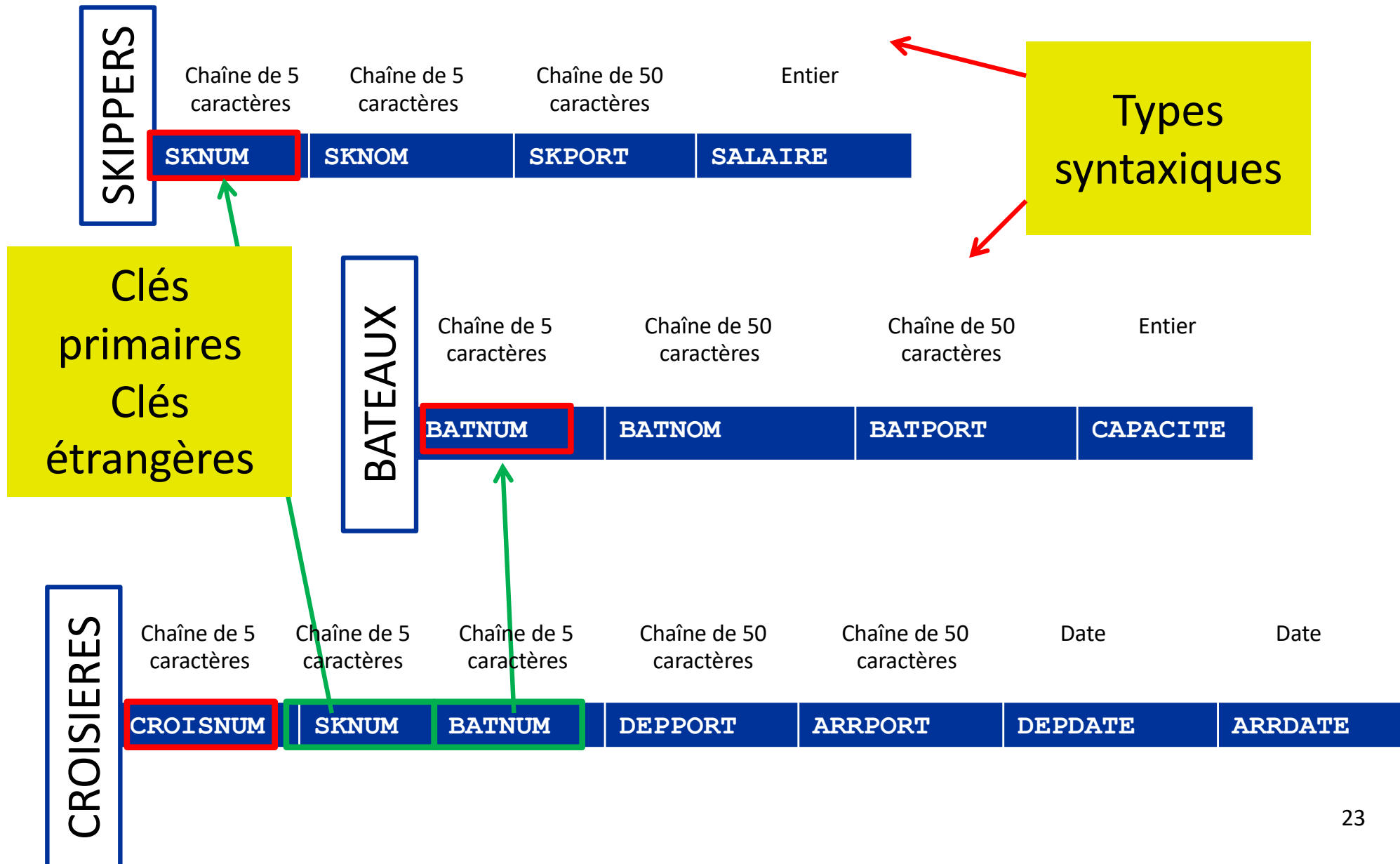
Clés étrangères

- Un attribut peut être clé étrangère et faire partie d'une clé primaire multi-attribut.
- Une relation est dite **statique** si elle ne possède aucune clé étrangère.
- Elle est dite **dynamique** si elle possède des clés étrangères.



Qu'est-ce qu'un schéma relationnel?

Exemple de schéma relationnel



Contraintes d'intégrité

Vérifiées par le SGBD lors des opérations de mises à jour

- Intégrité de domaine
 - Les valeurs d'attributs doivent être conformes à la définition syntaxique (**type**) et (*sémantique*) du domaine associé.
- Intégrité d'entité
 - Les valeurs des attributs clés primaires doivent être **UNIQUES** et **NON NULLES**

Intégrité référentielle

- Les valeurs des attributs clés étrangères doivent être **incluses** dans les valeurs des clés primaires associées

SKIPERS	SKNUM	SKNOM	SKPORT	SALAIRE
	1	JEAN	AJACCIO	3000
	2	PAUL	AJACCIO	2000
	3	MARINE	ANTIBES	2000

CROISIERS	CROISNUM	SKNUM	BATNUM	DEPPORT	ARRPORT	DEPDATE	ARRDATE
	C001	1	B002	BASTIA	CALVI	10/07	11/07
	C002	2	B002	ANTIBES	AJACCIO	15/07	16/07
	C003	1	B003	AJACCIO	BASTIA	21/08	22/08
	C004	3	B004	CALVI	MARSEILLE	02/09	03/09

Missions d'un SGBD relationnel

- Un SGBD Relationnel doit fournir des outils pour:
 - Spécifier le type syntaxique des attributs
 - Définir les clés primaires
 - Définir les clés étrangères
- Il doit assurer le **maintien de l'intégrité** de la base de données lors des opérations de mise à jour.

Data Definition
Language

Insertions, Modifications, Suppressions

Conseils

- Les attributs comparables doivent être définis sur le même type syntaxique.

Attention à la taille des textes!

- Clés primaires « artificielles »
 - Obligatoire si aucune clé candidate
 - En général préférable à une clé « métier »
 - Plus concise et aucune modification envisageable
 - Garder aussi la clé métier avec une contrainte UNIQUE
 - Limiter si possible les clés primaires multi-attributs
- Ne pas oublier les clés étrangères
 - L'ossature fondamentale de la BD





Définition et Mise à jour des données en SQL

Rappels des commandes
SQL

Spécificités Oracle



UNIVERSITÀ DI CORSICA
PASQUALE PAOLI

Langage SQL

- Langage de requête standard des SGDB relationnels
- Commercialisé par **ORACLE** en 1979
- Très facile sur les interrogations basiques
 - Sélection/projection
 - Jointures simples
- Des aspects plus complexes
 - Requêtes imbriquées

Le langage SQL

Principales Commandes

DDL



Administrateur

Commandes de Définition des
structures de données
(Tables, Index, Vues)

- CREATE création
- ALTER modification
- DROP suppression

DCL

Commandes de Gestion du
contrôle des données

Droits d'accès, reprise sur
panne

Administrateur



Utilisateurs



Commandes de Manipulation
des données

DML

INTERROGATION

SELECT recherche

MISES A JOUR

- INSERT insertion
- UPDATE modification
- DELETE suppression

Installation postgreSQL et pgAdmin

■ Installez un serveur postgreSQL sur votre machine

- Téléchargez l'image PostgreSQL depuis Docker Hub (version actuelle 17.6)
- Créer un conteneur docker pour votre serveur

Récupérez le
fichier
Installation
postgresql.pdf
sur l'ENT

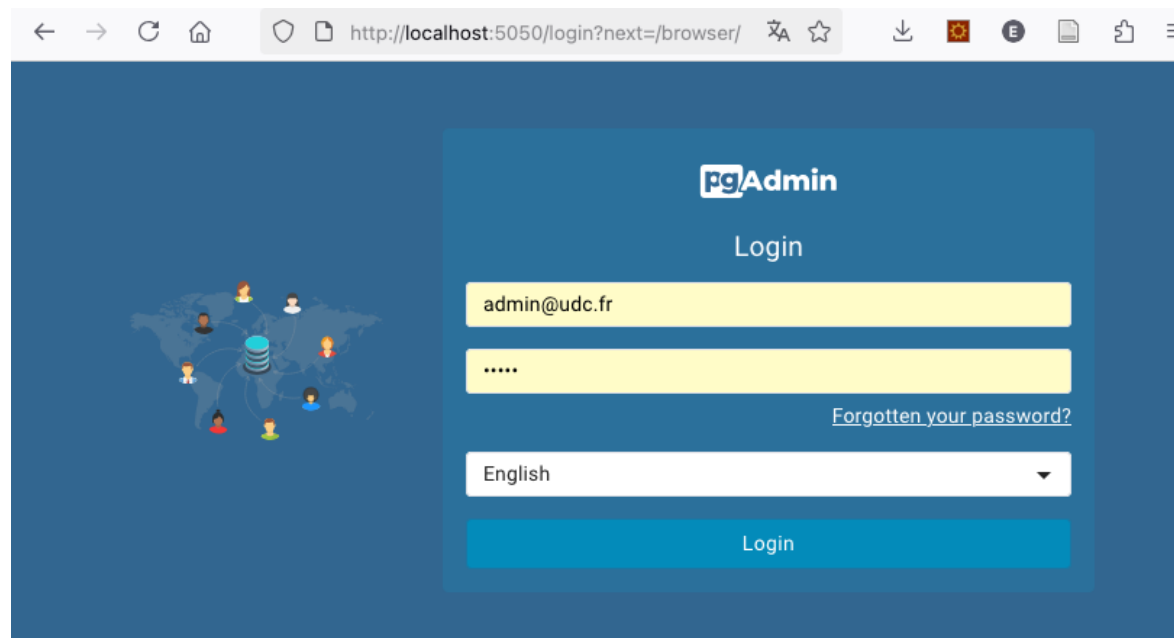
■ Installez l'interface graphique pgadmin pour interagir avec le serveur

- Téléchargez l'image PostgreSQL depuis Docker Hub (version pgadmin 4 version 9.8)
- Créer un conteneur docker associé

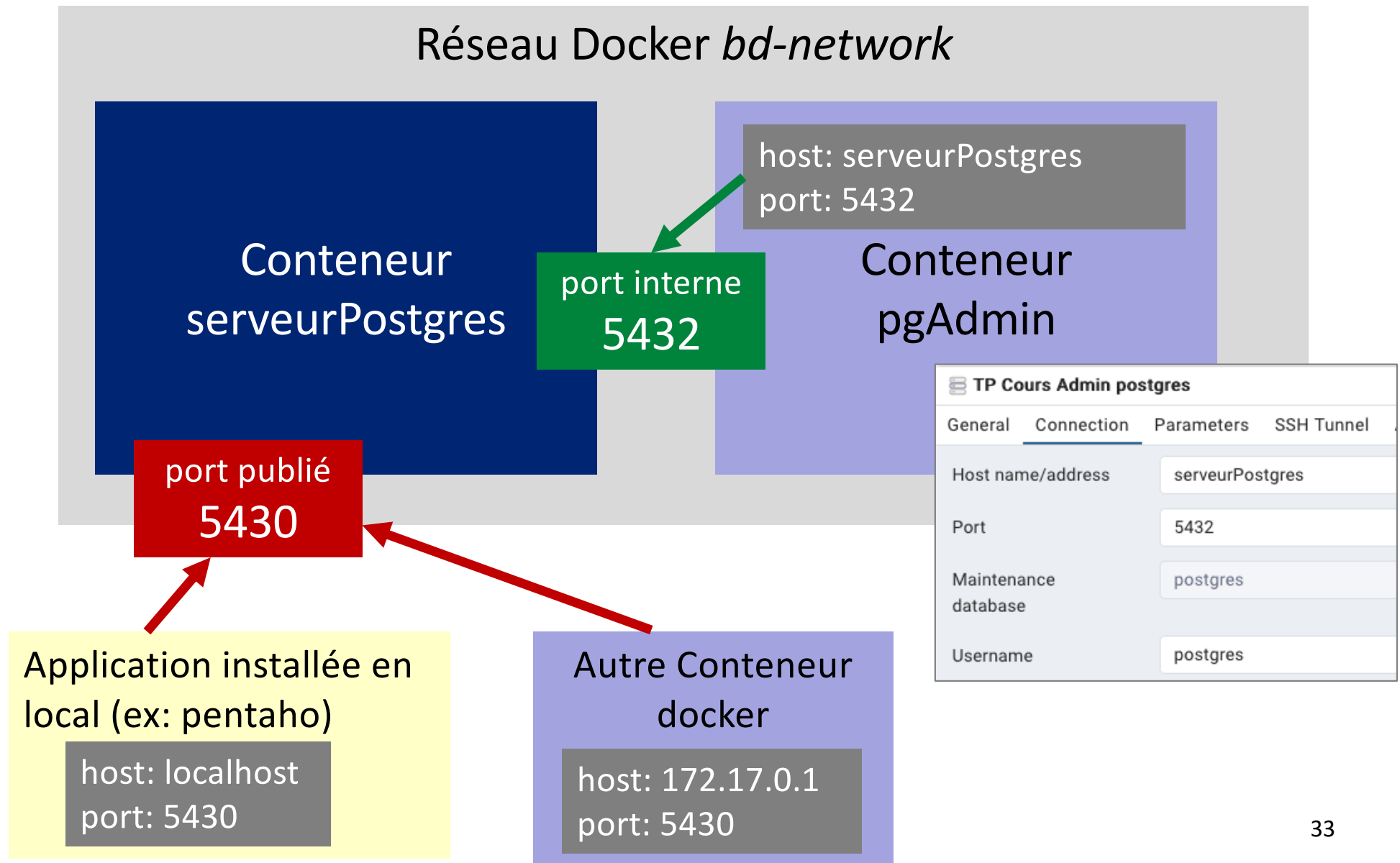
Attention au mot de
passe de l'utilisateur
admin que vous créerez
à l'installation !
Notez le bien

Installation et lancement pgadmin

- Télécharger les images docker
- Créer un réseau docker db_network
- Lancer le conteneur PostgreSQL (5432 → 5430)
- Lancer le conteneur pgAdmin (80 → 5050)
- Accédez à l'interface pgadmin dans votre navigateur
 - `http://localhost:5050/`



Organisation des conteneurs dockers



Comment accéder à PostgreSQL en conteneur Docker

Contexte	Linux natif	Windows / macOS (Docker Desktop)	Notes
Depuis un autre conteneur attaché au même réseau Docker user-defined (ex: postgres-network)	serveurPostgres:5432	serveurPostgres:5432	Utiliser le nom du conteneur (: port interne) si Postgres et pgAdmin partagent un réseau user-defined (recommandé).
Depuis l'hôte (outil installé en local, ex. Pentaho)	localhost:5430	localhost:5430	Utiliser le port publié (-p 5430:5432) → même commande sur tous les OS.
Depuis un autre conteneur via l'hôte (port externe publié : 5430)	172.17.0.1:5430	host.docker.internal:5430	Gateway Docker : sur Linux c'est 172.17.0.1, sur Win/Mac il faut utiliser host.docker.internal.

Vérifier l'adresse de la passerelle à utiliser avec la commande
docker network inspect *postgres-network*(*nom du reseau docker*)
Gateway ← l'adresse de passerelle à utiliser (ex. 172.21.0.1).

Instance PostgreSQL (dans un conteneur docker)

- Processus postgres qui tourne sur une machine (ou ici dans un conteneur docker)
- Défini par une **adresse (host) + port**
- Peut contenir **plusieurs bases de données**

```
# Accès depuis un  
conteneur dans le même  
reseau docker
```

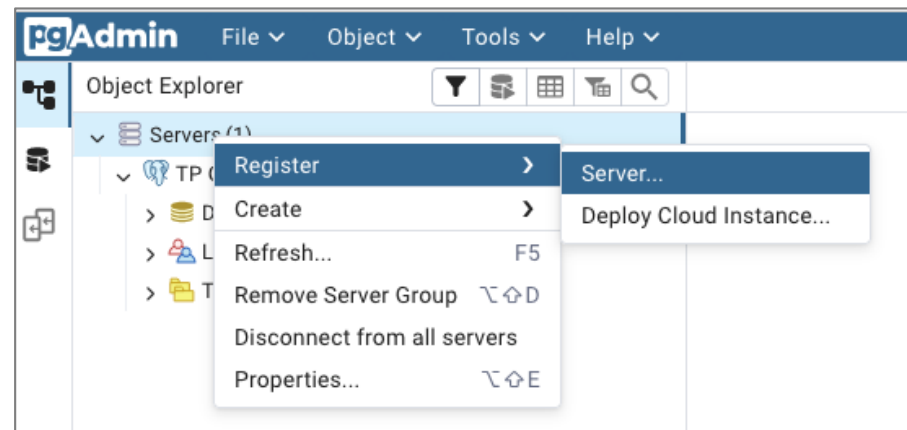
```
host name : serveurPosgres  
port : 5432
```

```
# Accès via l'hôte
```

```
host name : 172.17.0.1  
port : 5430
```

Créer une Connexion (PgAdmin)

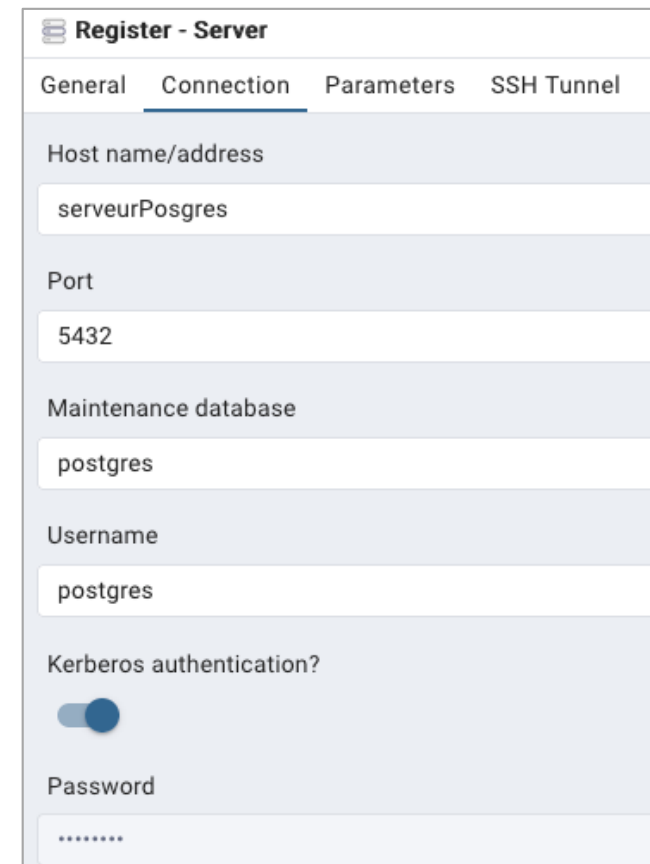
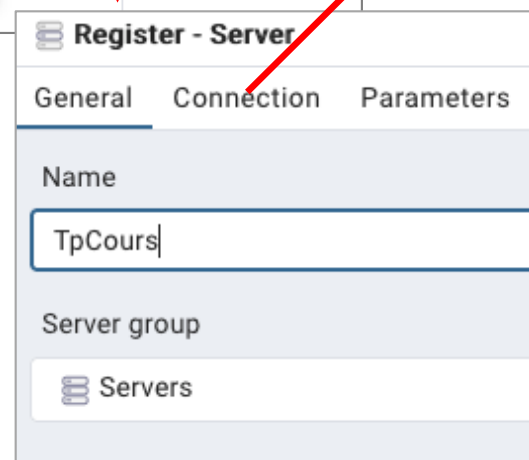
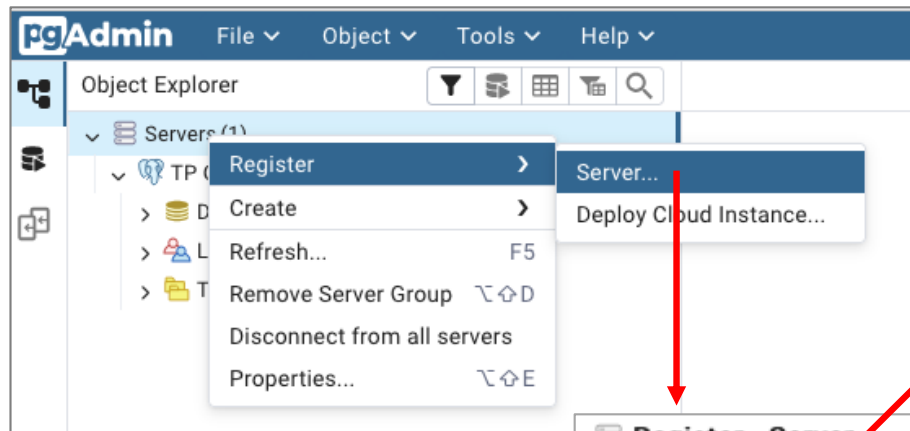
- Un « serveur » dans pgAdmin =
 - une connexion enregistrée vers une instance
- Paramètres : host, port, utilisateur, mot de passe
- Une même instance peut être représentée par plusieurs connexions (utilisateurs/droits différents)



Créez une connexion au serveur



- Créez une nouvelle connexion BD nommée TPCOURS en vous connectant avec l'utilisateur postgres (utilisateur par défaut de PostgreSQL).



Créez une première BD



Object Explorer

Servers (1)

TP Cours Admin postgres

Databases (17)

Create Database... F5

Refresh F5

Create - Database

General Definition Security Parameters Advanced SQL

Database PremiereBD

OID

Owner postgres

Comment

Close Reset Save

Servers (1)

TP Cours Admin postgres

Databases (18)

PremiereBD

Casts

Catalogs

Event Triggers

Extensions

Foreign Data Wrappers

Languages

Publications

Schemas (1)

public

Aggregates

Collations

Domains

FTS Configurations

FTS Dictionaries

FTS Parsers

FTS Templates

Foreign Tables

Functions

Materialized Views

Operators

Procedures

Sequences

Tables

Trigger Functions

Types

Views

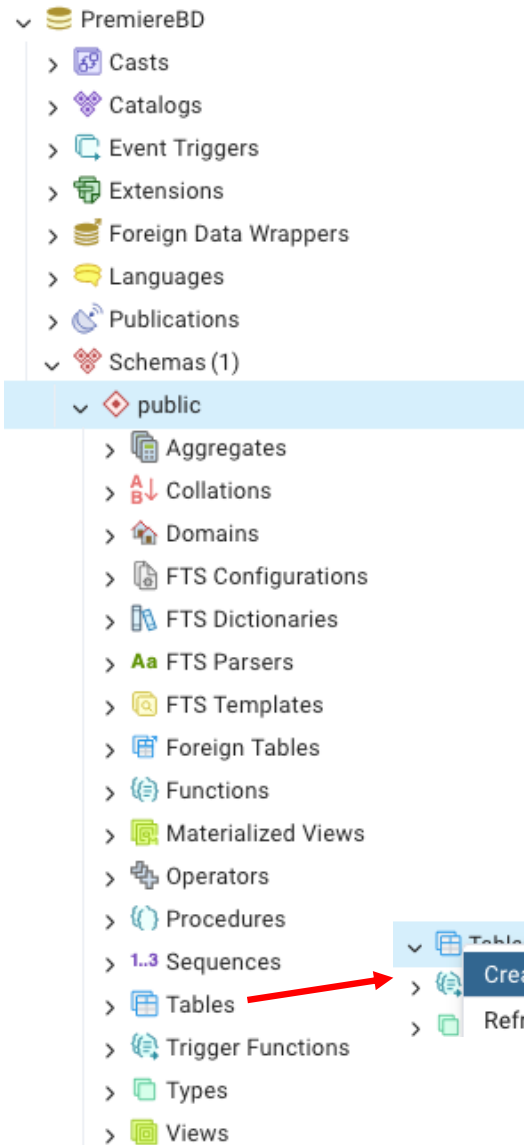
Diagram illustrating the steps to create a database in PostgreSQL:

- Right-click on the **Databases (17)** folder in the Object Explorer.
- Select **Create** > **Database...**
- The **Create - Database** dialog box appears. Enter **PremiereBD** in the **Database** field.
- Click the **Save** button.

The resulting database structure is shown in the Object Explorer:

- Servers (1)**
 - TP Cours Admin postgres**
 - Databases (18)**
 - PremiereBD**
 - Casts**
 - Catalogs**
 - Event Triggers**
 - Extensions**
 - Foreign Data Wrappers**
 - Languages**
 - Publications**
 - Schemas (1)**
 - public**
 - Aggregates**
 - Collations**
 - Domains**
 - FTS Configurations**
 - FTS Dictionaries**
 - FTS Parsers**
 - FTS Templates**
 - Foreign Tables**
 - Functions**
 - Materialized Views**
 - Operators**
 - Procedures**
 - Sequences**
 - Tables**
 - Trigger Functions**
 - Types**
 - Views**

Créez une table (avec l'interface graphique)

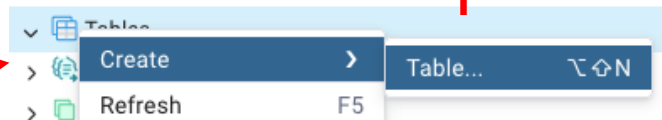


cours

General Columns Advanced Constraints Partitions Parameters Security SQL

Inherited from table(s)

Columns							
	Name	Data type	Length/Precision	Scale	Not NULL?	Primary key?	Default
	id	integer			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	nextval('c
	nom	character varying	100		<input type="checkbox"/>	<input type="checkbox"/>	
	description	text			<input type="checkbox"/>	<input type="checkbox"/>	



Accès à la fenêtre SQL



The image shows a database management interface. On the left, a tree view displays the database structure under 'PremiereBD'. A red arrow points to the 'Query Tool' option in the context menu. On the right, the 'Query Tool' window is open, showing a query editor with a red arrow pointing to the query input area.

Database Structure (Left Panel):

- ▼ PremiereBD
 - > Cas
 - > Cat
 - > Eve
 - > Ext
 - > For
 - > Lan
 - > Pub
 - ▼ Sch

Context Menu (Right-click on PremiereBD):

- Create
- Drop
- Drop (Force)
- Refresh... (F5)
- Restore...
- Backup...
- Export Data Using Query...
- CREATE Script
- Disconnect from database
- ERD For Database
- Maintenance...
- Grant Wizard...
- Search Objects... (Ctrl+S)
- Query Tool (Ctrl+Q)**
- Properties... (Ctrl+E)

Query Tool Window (PremiereBD/postgres@TP Cours Admin postgres):

- Query History
- 1



Gestion des utilisateurs et privilèges

- SQL Data Control Language

Qu'est ce que le DCL?

- Le Data Control Language (Langage de contrôle des données) est destiné au DBA.
- Il s'agit d'un sous-ensemble de SQL dédié à la gestion des utilisateurs et des droits de manipulation des objets de la base de données.



On parle de gestion
des privilèges

Gestion des utilisateurs (version ORACLE – MySQL)

- Création
 - **CREATE USER** nom1 **IDENTIFIED BY** motPasse;
- Changement de nom
 - **RENAME USER** nom1 TO nom2;
- Suppression
 - **DROP USER** nom1;
- Modification mot de passe
 - **SET PASSWORD FOR** nom1=
PASSWORD('motPasse');

ORACLE

Gestion des utilisateurs (version Postgres)

- Création
 - `CREATE USER nom1 WITH PASSWORD motPasse;`
- Changement de nom
 - `ALTER USER nom1 RENAME TO nom2;`
- Suppression
 - `DROP USER nom1;`
- Modification mot de passe
 - `ALTER USER nom1 WITH PASSWORD 'motPasse';`



Gestion des privilèges

- Permettent de contrôler l'accès à la base de données
- Privilèges octroyés à un utilisateur sur un ou plusieurs objets de la base
- Gestion réalisée par un administrateur ou un utilisateur disposant des droits requis
- Types de privilèges:
 - création, MAJ, suppression

Donner/Supprimer des privilèges

■ **GRANT** privilege1, privilege2, ...

[ON objet1, objet2,]

TO user1, user2, ...

[WITH GRANT OPTION];

ou ROLES

option de délégation (le droit de transmettre à son tour)

■ **REVOKE** privilege1, privilege2, ...

[ON objet1, objet2,]

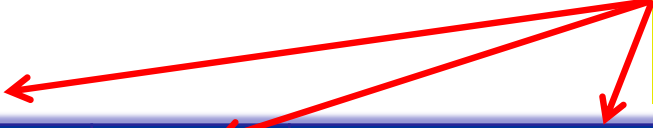
TO user1, user2, ...

ALL PRIVILEGES

Pour donner/supprimer tous les privileges

Exemples de Privilèges

Objets



Privilèges	Table	Vue	Séquence
ALTER			×
DELETE	×	×	
INDEX	×		
INSERT	×	×	
SELECT	×	×	×
UPDATE	×	×	

Gestion des privilèges/rôles : exemples

- **ALTER USER patron WITH SUPERUSER;**
 - confère les privilèges d'administrateur (rôle prédéfini SUPERUSER) à l'utilisateur patron
- **CREATE ROLE chef;**
 - crée le rôle chef
- **GRANT INSERT ON TABLE** skippers **TO** chef;
 - confère un privilège au rôle chef
- **GRANT** chef **TO** Pierre, Jean;
 - confère les privilèges attribués au rôle chef aux utilisateurs Pierre et Jean

Gestion des Privilèges : exemples

- **GRANT** INSERT,UPDATE (SKPORT) **ON TABLE** SKIPPERS **TO** JEAN, PIERRE;
 - confère les droits d'insertion et de maj de la colonne skport de la table Skippers aux utilisateurs Jean et Pierre
- **GRANT** DELETE, UPDATE , INSERT **ON TABLE** SKIPPERS **TO** MICHEL **WITH GRANT OPTION**;
- **REVOKE** UPDATE (SKPORT) **ON TABLE** SKIPPERS **FROM** JEAN ;

Gestion des privilèges: Rôles

- Les rôles sont définis pour regrouper un ensemble de privilèges et faciliter leur gestion
- Un rôle peut être attribué à un utilisateur
 - Création d'un rôle *exemple* ayant l'autorisation de créer des bases de données et des rôles
 - `CREATE ROLE exemple WITH CREATEDB
CREATE ROLE;`

Gestion des privilèges: Rôles

- Quelques rôles prédéfinis
 - **SUPERUSER**: Tous les privilèges (équivalent au rôle DBA dans Oracle)
 - **CONNECT**: Connexion à la base
 - GRANT CONNECT ON DATABASE nom_de_la_base TO utilisateur;
 - *accorde à utilisateur le privilège de se connecter à la base de données nom_de_la_base.*

Exercice 1.1: Utilisateurs et droits

- Dans votre BD PremièreBD
- Ajoutez un nouvel utilisateur **utilcours** (mot de passe utilcours) ayant uniquement le droit d'insérer des données dans une table **cours**.
- Deconnectez vous du serveur et créez une nouvelle connection **TestRoleUtilCours** en utilisant l'utilisateur utilcours (et non postgres).



Exercice 1.2: Utilisateurs et droits



Depuis cette nouvelle connexion **TestRoleUtilCours**

- Essayez d'exécuter les commandes suivantes:
 - Insérer une ligne dans la table Cours avec une commande Insert

```
INSERT INTO cours (id,nom, description)
VALUES (1,'Introduction à PostgreSQL', 'Un cours
pour débutants sur PostgreSQL');
```

- Créer une table TestTable :

```
CREATE TABLE TestTable (
  id SERIAL PRIMARY KEY,
  nom VARCHAR(50)
);
```

Est-ce possible?

SQL

Requêtes de définition des structures de données

- CREATE création
- ALTER modification
- DROP suppression
- Objets **SEQUENCE** (Oracle)

DDL
Data Definition
Language

Requêtes de création de table

```
CREATE TABLE Nom_Table
(
  colonne1          type1,
  colonne2          type2,
  .....
  CONSTRAINT nom_constraint1
    type_constraint1,
  .....
);
```

Trois Types de
Contraintes

PRIMARY KEY

FOREIGN KEY

CHECK (condition)

Principaux types de données Oracle

optimisation
mémoire mais MAJ
plus lourdes

Type	description
VARCHAR2(taille)	Chaînes de caractères de longueur variable
CHAR(<i>taille</i>)	Chaînes de caractères de longueur fixe
NUMBER(<i>nbChiffres</i> , <i>nbDecim</i>)	Numériques de longueur variable
DATE	dates et heures

nbChiffres= nombre total de chiffres
nbDecim=nombre de chiffres après la virgule

Principaux types de données postgres



PostgreSQL

Type	description
VARCHAR(<i>taille</i>) ou TEXT	Chaînes de caractères de longueur variable
CHAR(<i>taille</i>)	Chaînes de caractères de longueur fixe
NUMERIC(<i>nbChiffres</i> , <i>nbDecim</i>)	Numériques de longueur variable
INTEGER ou int	Nombres entiers (32 bits)
SMALLINT	Nombres entiers (16 bits)
BIGINT	Nombres entiers (64 bits)
DATE	Dates (sans heure)
TIMESTAMP	Dates et heures

optimisation
mémoire mais MAJ
plus lourdes

nbChiffres= nombre total de
chiffres
nbDecim=nombre de chiffres
après la virgule

Types de données (Autres SGBD)

Type de données	Access	SQLServer	Oracle	MySQL	PostgreSQL
boolean	Yes/No	Bit	Byte	-	Boolean
integer	Number (integer)	Int	Number	Int Integer	Int Integer
float	Number (single)	Float Real	Number	Float	Numeric
currency	Currency	Money	-	-	Money
string (fixed)	-	Char	Char	Char	Char
string (variable)	Text (<256) Memo (65k+)	Varchar	Varchar2	Varchar	Varchar
binary object	OLE Object Memo	Binary (fixed up to 8K) Varbinary (<8K) Image (<2GB)	Long Raw		

source : http://www.w3schools.com/sql/sql_datatypes_general.asp
plus de détails sur http://docs.oracle.com/cd/E12151_01/doc.150/e12155/oracle_mysql_compared.htm#BABGACIF

Bonnes pratiques sur le choix des types

- Toujours choisir le type le plus spécifique -> intégrité, performance (important pour indexation), clarté

Types Numériques

•integer / bigint :

→ adaptés à des entiers, rapides en stockage et en calcul
→ bigint évite les dépassements pour de très gros volumes

•numeric pour valeurs monétaires :

→ garantit une précision exacte (pas d'erreur d'arrondi comme avec float)
→ essentiel pour des montants financiers ou mesures sensibles

•float :

→ utile pour des calculs scientifiques approximatifs, mais pas pour des montants précis

Chaînes de caractères

text vs varchar(n) :

→ PostgreSQL gère text et varchar de la même manière côté performance
→ varchar(n) n'ajoute que la contrainte de longueur

char(n) :

→ remplit avec des espaces
→ coûteux et source d'erreurs de comparaison.

Booléens et énumérations

boolean :

→ plus lisible et léger qu'un int (0/1)

enum :

→ pratique pour un petit ensemble stable
→ plus compact et plus rapide qu'une FK dans ce cas

Requêtes de création de table

clé primaire mono-attribut

```
CREATE TABLE BATEAUX (  
  BATNUM VARCHAR(5) CONSTRAINT pk_bateaux  
    PRIMARY KEY,  
  BATNOM VARCHAR(50),  
  BATPORT VARCHAR(50),  
  CAPACITE INTEGER);
```

Définition des
types syntaxiques

Définition de la
clé primaire
(contrainte
colonne)

Requêtes de création de table clé primaire multi-attribut

```
CREATE TABLE PASSAGERS (  
    CROISNUM VARCHAR(5),  
    NOMPASSAGER VARCHAR(50),  
    PRIX NUMERIC(6),  
    REMISE NUMERIC(6) ,  
CONSTRAINT pk_passagers  
    PRIMARY KEY( CROISNUM, NOMPASSAGER) );
```




Définition d'une contrainte table
(obligatoire pour les multi-attributs)

Requêtes de création de table

Contrainte de vérification

```
CREATE TABLE SKIPPERS(  
    SKNUM VARCHAR(5) CONSTRAINT pk_skippers  
        PRIMARY KEY  
    SKNOM VARCHAR(50) NOT NULL,  
    SKPORT VARCHAR(50),  
    SALAIRE NUMERIC(6),  
    CONSTRAINT ck_skippers_salaire CHECK(salaire >=  
        1200) ) ;
```



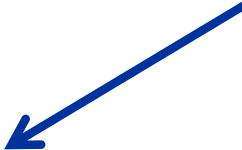
Définition d'une
contrainte de vérification
des valeurs

Requêtes de création de table

Définition des clés étrangères

```
CREATE TABLE CROISIERES (  
  CROISNUM VARCHAR(5) CONSTRAINT pk_croisieres  
  PRIMARY KEY,  
  DEPPORT VARCHAR(50),  
  ARRPORT VARCHAR(50),  
  DEPDATE DATE,  
  ARRDATE DATE,  
  BATNUM VARCHAR(5) REFERENCES BATEAUX (BATNUM),  
  SKNUM VARCHAR(5),  
  
  CONSTRAINT fk_croisieres_sknum FOREIGN KEY (sknum)  
  REFERENCES SKIPPER(SKNUM) );
```

Définition d'une clé étrangère (contrainte de colonne)



Définition d'une clé étrangère (contrainte de table)



Résumé contraintes

NULL / NOT NULL : niveau colonne

UNIQUE (colonne1 [, colonne2] ...)

PRIMARY KEY (colonne1 [, colonne2] ...)

FOREIGN KEY (colonne1 [, colonne2] ...)

REFERENCES nomTableLieu (colonne1 [, colonne2] ...)

[ON DELETE {CASCADE | SET NULL}]

CHECK {condition}

Options lors des
suppression

Contraintes définissables au niveau
colonne (ligne de définition de l'attribut)
ou au niveau table.

Valeurs par défaut

- **default** *valeur* (sur la ligne de définition de l'attribut):
 - DEPPORT **VARCHAR(50)** default 'AJACCIO'
 - DATEDEPART DATE default **CURRENT_DATE**



date du jour
ou CURRENT_TIMESTAMP

Remarque : **sysdate** sous Oracle

Requêtes de suppression/modification de table

```
DROP TABLE Nom-Table ;
```

Supprimer la table BATEAUX.

```
DROP TABLE BATEAUX;
```

DROP TABLE IF
EXISTS BATEAUX
permet d'éviter les
erreurs

Modification de table

Ajout d'un attribut

```
ALTER TABLE Nom-Table  
ADD Attribut type;
```

Suppression d'un
attribut

```
ALTER TABLE Nom-Table  
DROP COLUMN Attribut;
```

Requêtes de modification de table

Ajouter une colonne (attribut) Cap de type entier à la table BatAJACCIO.

```
ALTER TABLE      BatAJACCIO  
ADD  Cap INTEGER ;
```

Supprimer la colonne Cap de la table BatAJACCIO.

```
ALTER TABLE      BatAJACCIO  
DROP COLUMN      Cap;
```

Requêtes d'ajout/suppression de contrainte

Ajout d'une contrainte

```
ALTER TABLE Nom-Table  
ADD CONSTRAINT nom_contrainte  
.....définition;
```

Ajouter une contrainte spécifiant qu'il ne peut y avoir qu'un seul départ de croisière par jour pour chaque port

```
ALTER TABLE CROISIERES ADD CONSTRAINT  
UK_DEPART UNIQUE (DEPPORT, DEPDATE);
```

Suppression d'une contrainte

```
ALTER TABLE Nom-Table  
DROP CONSTRAINT nom_contrainte;
```

Requêtes de modification de table

Définition des clés étrangères

Ajout des clés étrangères
après la création de la table

```
ALTER TABLE croisiere ADD CONSTRAINT Cbatnum FOREIGN KEY  
(batnum) REFERENCES bateau (batnum);
```

```
ALTER TABLE croisiere ADD CONSTRAINT Csknum FOREIGN KEY  
(sknum) REFERENCES skipper (sknum);
```

Requêtes de modification de table

Renommer une table

- ALTER TABLE ancienNom **RENAME** nouveauNom;

- Ou plus simplement

RENAME TABLE *t1 TO newt1*
[, *t2 TO newt2*] ...

Colonne auto incrémentée (Postgres)

```
CREATE TABLE BATEAUX (  
  BATNUM SERIAL PRIMARY KEY,  
  BATNOM VARCHAR(50),  
  BATPORT VARCHAR(50),  
  CAPACITE INT);
```



type spécial INTEGER (ou INT)
avec une séquence de
génération automatique.

Colonne auto incrémentée (MySQL)

```
CREATE TABLE BATEAUX (  
  BATNUM INT(5) AUTO_INCREMENT PRIMARY  
  KEY,  
  BATNOM VARCHAR(50),  
  BATPORT VARCHAR(50),  
  CAPACITE INT(4));
```



Non disponible
sous ORACLE Il
faut utiliser un
objet SEQUENCE

Numérotation automatique
(valable pour les types
numériques)



Les objets SEQUENCE

- Un objet **SEQUENCE** définit un générateur de nombres.
- Il permet de générer des numérotations automatiques en particulier pour la création des valeurs de clé primaire.
- Une séquence est un objet à part entière qui peut être utilisé par plusieurs tables.
- Commandes
 - CREATE SEQUENCE, ALTER SEQUENCE, DROP SEQUENCE

pour information

Création d'un objet SEQUENCE

CREATE SEQUENCE nom

START WITH n

valeur initiale de la
numérotation

INCREMENT BY p

pas d'incrémentation
(positif ou négatif)

[**MINVALUE** min (ou **NOMINVALUE**)]

[**MAXVALUE** max (ou **NOMAXVALUE**)]

[**CYCLE** ou **NOCYCLE**]

[**CACHE** c ou **NOCACHE**];



pour information

anticipation des c valeurs suivantes en mémoire

Manipulation d'un objet SEQUENCE

- nomseq.**CURRVAL**
 - valeur courante de la séquence
- nomseq.**NEXTVAL**
 - incrémente la séquence et retourne la nouvelle valeur

Les Objets Sequence sont utilisés dans les commandes d'ajout de tuples

```
CREATE SEQUENCE C_NOCLI START WITH 1000  
MAXVALUE 9999 NOCYCLE;
```

```
INSERT INTO CLIENTS(NOCLI, NOMCLI)  
VALUES (C_NOCLI.NEXTVAL, 'PAOLI');
```



pour information

SQL

Requêtes de Mise à jour des données

- INSERT insertion
- UPDATE modification
- DELETE suppression

DML
Data
Manipulation
language (suite)

Exercice 2 : insertion de tuples



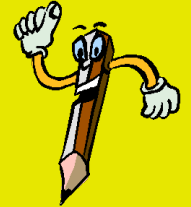
- Définir la commande de création d'une table SORTIE dont un exemple est donné ci-dessous.
- La clé NUMERO doit être incrémentée automatiquement lors des insertions.
- Définir la commande permettant d'insérer le tuple suivant dans la table sortie
 - Attention sous Postgres les littéraux de type date doivent être de la forme 'YYYY-MM-DD'

SORTIE

numero [PK] integer	lieu character varying (80)	datesortie date
1	Ajaccio	2024-09-22

Requêtes d'ajout de tuples

Toujours préférer lister explicitement les attributs
Même si cela semble lourd!



```
INSERT INTO Nom-Table (liste attributs)
VALUES
( expression1, expression2, ....);
```

- Ajouter à la table BATEAUX, le bateau numéro B22 de nom Toto localisé à AJACCIO et ayant une capacité de 15 passagers.

```
INSERT INTO BATEAUX (BATNUM,BATNOM, BATPORT, CAPACITE)
VALUES ('B22', 'Toto', 'AJACCIO', 15);
```


Requêtes d'ajout de tuples

- Ajouter à la table BATEAUX, le bateau numéro B23 de nom Evasion dont on ignore encore le port d'attache et la capacité.

```
INSERT INTO BATEAUX (BATNUM,BATNOM)  
VALUES ('B23', 'Evasion');
```

Requêtes d'ajout de tuples

```
INSERT INTO Nom-Table (liste attributs)
  SELECT liste attributs
  FROM Table
  WHERE Condition ;
```



- Ajouter à la table BATAJACCIO ayant pour attributs Num et Nom, les numéros et noms des bateaux de la table BATEAUX localisés à AJACCIO.

```
INSERT INTO    BATAJACCIO (Num, Nom)
  SELECT      BATNUM, BATNOM
  FROM        BATEAUX
  WHERE       BATPORT='AJACCIO';
```


Requêtes d'ajout de tuples

- Soit GrandBATEAUX une table ayant la structure suivante
GrandBATEAUX(BATNUM, BATNOM, ETAT)

Bon, Moyen, Mauvais

Ajouter à la table GrandBATEAUX, les bateaux de la table BATEAUX ayant une capacité supérieure ou égale à 10 en les définissant en « bon » état.

```
INSERT INTO GrandBATEAUX (BATNUM, BATNOM, ETAT)
SELECT BATNUM, BATNOM, 'Bon'
FROM BATEAUX
WHERE CAPACITE >= 10 ;
```

Toujours lister
explicitement les
attributs



Requêtes de suppression de tuples

```
DELETE FROM Nom-Table  
WHERE Condition ;
```

Condition définissant les tuples
devant être supprimés

- Supprimer tous les bateaux localisés à AJACCIO.

```
DELETE FROM      BATEAUX  
WHERE    BATPORT='AJACCIO';
```

Requêtes de modification de tuples

```
UPDATE Nom-Table  
SET  Attribut1 = expression1 ,  
      Attribut2 = expression2, ....  
WHERE Condition ;
```

Condition définissant les tuples
devant être modifiés

- Augmenter de 200 euros le salaire du Skipper numéro 1.

```
UPDATE    SKIPERS  
SET       SALAIRE = SALAIRE + 200  
WHERE     SKNUM=1 ;
```

Insertion massive de données

Pour accélérer l'exécution d'un script de chargement massif de données , il peut être intéressant de désactiver temporairement la vérification de contraintes de clés étrangères.

SOLUTION sous ORACLE uniquement
(commande non disponible sous postgresql)

```
ALTER TABLE croisiere DISABLE CONSTRAINT fk_sknun;  
..insertions massives  
ALTER TABLE croisiere ENABLE CONSTRAINT fk_sknun;
```

contrainte



A n'utiliser que si les insertions respectent bien l'intégrité

Désactivation temporaire des contraintes de clés étrangères

Solutions sous Postgres



■ Supprimer la contrainte puis la recréer

- `ALTER TABLE croisieres DROP CONSTRAINT fk_sknum;`
-
- `ALTER TABLE croisieres ADD CONSTRAINT fk_sknum FOREIGN KEY (sknum) REFERENCES skippers(sknum);`

■ Utiliser les options de validation différée

- Permet de différer la vérification des contraintes à la fin d'une transaction
- La contrainte doit avoir été définie comme **deferrable** :

```
ALTER TABLE croisieres ADD CONSTRAINT fk_sknum FOREIGN  
KEY (sknum) REFERENCES skippers(sknum) DEFERRABLE  
INITIALLY DEFERRED;
```

SQL

Notion de Vue

- Définition
- Vues et indépendance logique

DML
Data
Manipulation
language (suite)

VUES et Requêtes administrateur

VUE = Table virtuelle définie à partir d'autres tables grâce à une requête

Définition d'une vue

```
CREATE VIEW Nom-Vue AS Requete SFW;
```

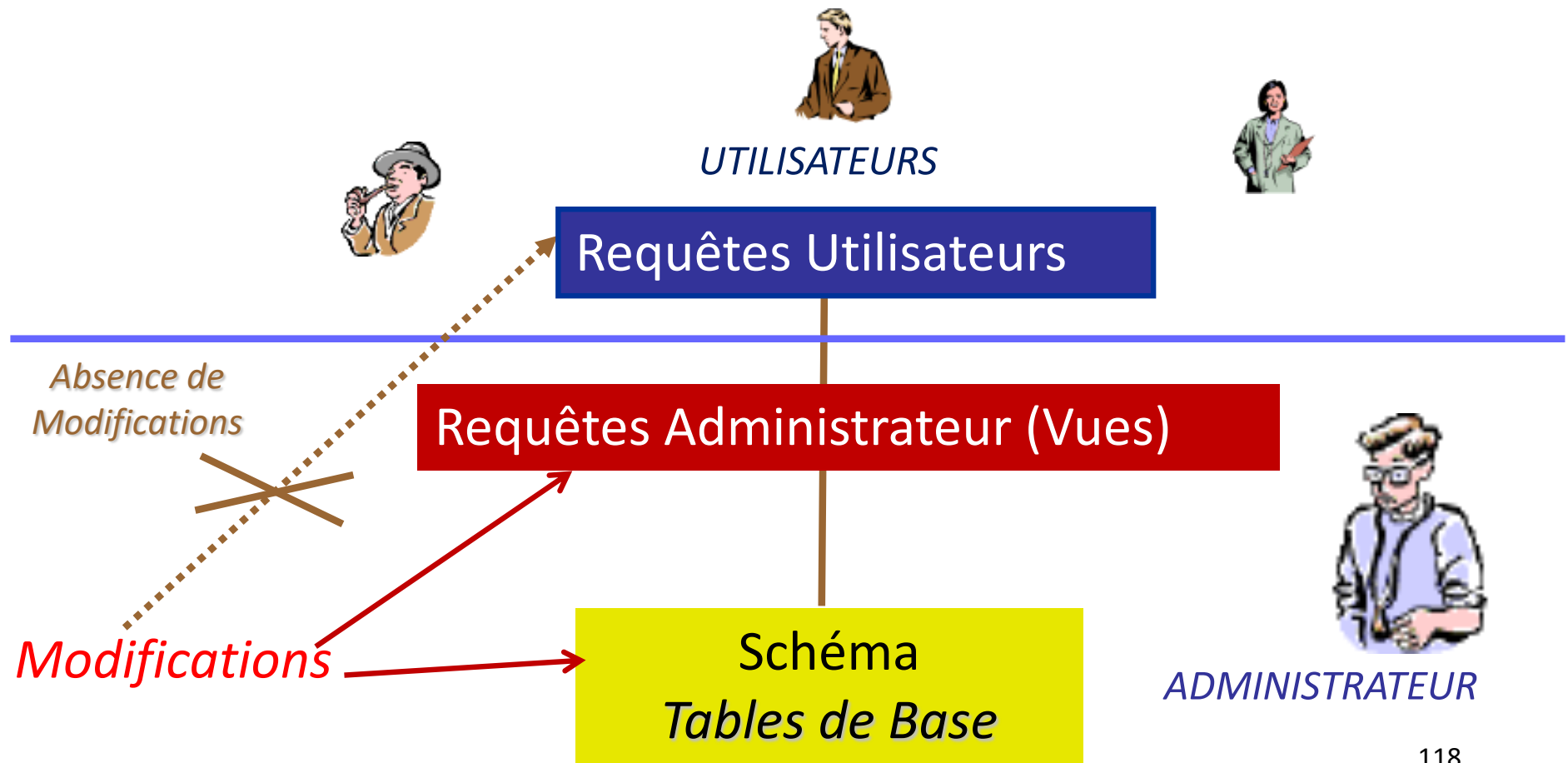
- Créer une vue contenant tous les skippers localisés à Nice

```
CREATE VIEW SKIPNICE AS  
SELECT *  
FROM SKIPPERS  
WHERE SKPORT= 'NICE' ;
```

VUES et Requêtes administrateur

Indépendance Logique

Les utilisateurs ne manipulent les tables de base qu'à travers les requêtes (ou vues) définies par l'administrateur.



Limites des VUES

- Les possibilités de mises à jour sur les vues sont souvent très limitées :
 - Vue mono-table uniquement
 - Vue ne comportant pas de clauses GROUP BY
 - Vue ne comportant pas d'attribut calculé
 - Vue ne comportant pas de clause ensembliste (UNION, MINUS, INTERSECT)

Exercice 3 (1) : Vues et indépendance logique



- Définissez une nouvelle BD BDCroisieres.
- Récupérez sur l'ENT le script de définition de la Base de données Croisières utilisée dans les chapitres 1 et 2 du cours.

bdcroisieresCours.sql

- 1) Définissez une vue SKIPMOY contenant les numéros, noms et ports d'attache des skippers ayant un salaire supérieur à la moyenne des salaires des skippers.
- 2) Définissez une requête basée sur la vue SKIPMOY et renvoyant la liste des ports d'attache des skippers dont le salaire est supérieur à la moyenne.



Exercice 2 (2) : Vues et indépendance logique

Suite à un changement dans l'organisation de l'entreprise, un Skipper peut à présent être attaché à plusieurs ports de localisation.

- Ainsi la table SKIPPERS initiale ne permet pas d'insérer les informations suivantes:
 - Le skipper N° 1, JEAN a pour ports d'attache AJACCIO et PROPRIANO
 - Le skipper N°3, LAURA a pour ports d'attache ANTIBES et NICE.

sknum	skno	skport
1	JEAN	AJACCIO
2	PAUL	AJACCIO
3	LAURA	ANTIBES
4	PIERRE	BASTIA

Comment modifier le schéma pour prendre en compte ce changement?

Exercice 3 (3) : Vues et indépendance logique



- 3) Proposez une solution et définissez les requêtes nécessaires à la prise en compte de vos modifications (ajout et modification de tables, mises à jour de données, vue)
- 4) Pouvez-vous réexécuter la requête de la question 2 dans le nouveau schéma ? Que pouvez-vous en conclure sur l'intérêt de l'utilisation des vues ?

VUES MATERIALISEES

- Vues dont le résultat est stocké physiquement dans la BD

Contrairement à une vue classique

- **Avantage** : Performance

les requêtes sur une vue matérialisée sont souvent plus rapides car les données sont précalculées

- **Contraintes**

- Nécessité d'un rafraichissement
 - manuel ou automatique(possible uniquement sous oracle)
- Cout d'occupation en termes d'espace

Disponible sous Postgresql uniquement depuis la version 9.3

VUES MATERIALISEES POSTGRESQL

```
CREATE MATERIALIZED VIEW nom_vue_materialisee  
AS  
requete SFW
```

- En PostgreSQL, toutes les vues matérialisées sont construites immédiatement. Il n'y a pas d'option pour différer la construction comme en Oracle.

Le Rafraichissement
automatique n'est
pas possible



VUES MATERIALISEES



- Rafraichissement manuel

```
REFRESH MATERIALIZED VIEW nom_vue_materialisee;
```

2 options possibles :

- **WITH DATA** (par défaut si rien n'est précisé)
 - rafraîchit la vue matérialisée et remplit ses données en exécutant à nouveau la requête sous-jacente.
- **WITH NO DATA**
 - vide la vue matérialisée en attendant un rafraichissement ultérieur.

VUES MATERIALISEES :exemple

- Vue matérialisée pour afficher le nombre total de croisières par bateau.
- Cette vue doit se rafraîchir automatiquement lors de chaque mise à jour de la table croisiere.

```
CREATE MATERIALIZED VIEW mv_croisieres_par_bateau  
BUILD IMMEDIATE  
REFRESH FORCE ON COMMIT  
AS  
SELECT BATNUM, COUNT(CROISNUM) as NB_CROISIERES  
FROM CROISIERES  
GROUP BY BATNUM;
```


VUES MATERIALISEES ORACLE

```
CREATE MATERIALIZED VIEW nom_vue_materialisee  
BUILD [IMMEDIATE | DEFERRED]  
REFRESH [FAST | COMPLETE | FORCE]  
[START WITH date]  
[NEXT expr]  
[ON COMMIT | ON DEMAND]
```

Rafraichissement
synchrone ou
asynchrone (sur
demande)

AS

requete SFW



- FAST : rafraîchissement rapide si possible
- COMPLETE : Effectue toujours un rafraîchissement complet.
- FORCE : Oracle décide du type de rafraîchissement (rapide ou complet) en fonction de la disponibilité des journaux.

VUES MATERIALISEES

(sous ORACLE)

■ Rafraîchissement manuel

```
BEGIN  
    DBMS_MVIEW.REFRESH(list => nomVue,  
    method => 'C');  
END; /
```



- 'C': Rafraîchissement complet.
- 'F': Rafraîchissement rapide (si possible). Seules les modifications (insertions, mises à jour, suppressions) depuis le dernier rafraîchissement sont prises en compte.
- '?': Oracle décide de la meilleure méthode (complet ou rapide).

JOURNAL DE VUES MATERIALISEES

- Pour pouvoir effectuer des rafraichissements rapides (seules les lignes modifiées depuis le dernier rafraîchissement sont mises à jour)
- Il faut définir un journal de vue matérialisée sur la table et les colonnes concernées.

Fast



```
CREATE MATERIALIZED VIEW LOG ON TABLE  
WITH ROWID,  
SEQUENCE(nomCol1, nomCol2,...)  
INCLUDING NEW VALUES;
```

Non
disponible
sous
postgres

VUES CLASSIQUES et VM

Caractéristiques	Vue classique	Vue Matérialisée
Stockage des données	Aucun	Stockage physique
Performance	Recalcul à chaque exécution	Plus rapide car exécution unique
Mise à jour des données	Automatique	Nécessite un rafraîchissement
Utilisation d'espace	Aucun espace de stockage	Requiert de l'espace de stockage
Flexibilité et performance	Peut représenter n'importe quelle requête SQL sans coûts supplémentaires en termes de stockage ou de performance.	Idéale pour les requêtes fréquemment utilisées et coûteuses, mais moins flexible

Exercice 4 : Vue Matérialisée



- Définissez une vue matérialisée qui donne la liste des numéros et noms de bateaux avec le nombre de skippers qui les barrent et le total de leurs salaires.

batnum character varying (5) 🔒	batnom character varying (50) 🔒	nb_skippers bigint 🔒	total_salaire bigint 🔒
B004	INDEPENDANCE	2	5000
B001	LIBERTE	1	2500
B003	KALISTE	1	3000
B002	LOUISIANE	4	11000

SQL

Notion de dictionnaire de données

- Principales vues
- Requêtes sur le DD



Notion de Dictionnaire de données

- **Méta-données**= informations sur les objets de la base de données (tables, index, ...)
- **Dictionnaire de données** (DD) = ensemble de tables définissant les métadonnées d'une BD
- Les tables du DD sont en général manipulées par l'intermédiaire d'un ensemble de vues




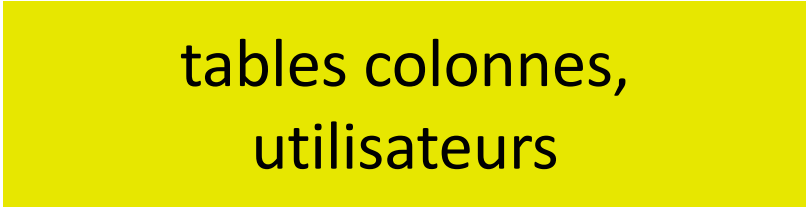

Existe dans tous les SGBD avec des noms de tables différents

Pourquoi utiliser les DD?

- Interroger la structure de la base de données
 - Savoir quelles tables, vues, colonnes, et types de données existent.
- Surveiller les performances et l'activité
 - Obtenir des statistiques sur l'utilisation des tables, des index, et des connexions.
- Gérer les utilisateurs et les privilèges
 - Vérifier quels rôles ont accès à quels objets de la base de données.
- Diagnostiquer et résoudre les problèmes
 - Identifier les verrous bloquants ou les requêtes longues.

Principales vues du DD sous PostgreSQL

4 catégories:

- Schéma pg_catalog
 - tables et vue systemes principales de PostgreSQL
- Vues de schéma d'information (information_schema) 
 - conforme au standard SQL
- Vues de statistiques et de surveillance (pg_stat_*) 
- Vues de description des objets (pg_description)

Principales vues du DD

Nom de la vue	Objets décrits	Colonnes
information_schema.tables	Tables	table_schema, table_name, table_type
information_schema.columns	Colonnes des tables	table_schema, table_name, column_name, data_type
information_schema.views	Vues	table_schema, table_name, view_definition
information_schema.table_constraints	Contraintes	constraint_schema, constraint_name, table_name, constraint_type
information_schema.key_column_usage	Utilisation des colonnes clés	constraint_name, table_name, column_name
information_schema.routines	Fonctions/Procédures	routine_schema, routine_name, routine_type
pg_roles	roles	
information_schema.role_table_grants	utilisateurs	

Noms standards communs à d'autres SGBD

Requêtes sur le DD

- Donner la liste des noms des tables de la base

```
SELECT TABLE_NAME  
FROM information_schema.tables  
WHERE table_schema = 'public' AND  
       table_type = 'BASE TABLE';
```

table_name	
name	🔒
bateaux	
croisieres	
skippers	

- Donner la liste des noms de colonnes de la table BATEAUX et leur type

```
SELECT COLUMN_NAME, DATA_TYPE  
FROM information_schema.columns  
WHERE TABLE_NAME='bateaux';
```

column_name	data_type
name	character varying
capacite	integer
batnum	character varying
batnom	character varying
batport	character varying

Exercice 5 : Manipulation du DD



- 1) Définissez une requête qui donne la liste des noms de colonnes de la table CROISIERES
- 2) Définissez une requête qui donne le type de la colonne SKPORT de la table SKIPPER
- 3) Définissez une requête qui donne pour chaque table, son nombre de colonnes
- 4) Définissez une requête qui donne le nombre moyen de colonnes dans les tables de la base de données Croisieres



SQL Transaction Control Language

- Transactions

Qu'est ce que le TCL?

- Le Transaction Control Language (Langage de contrôle des transactions) est destiné au DBA.
- Il s'agit d'un sous-ensemble de SQL dédié à la gestion des **mises à jour** de la base de données faites par le DML.
- Il contient les commandes de manipulation des **transactions**.



Qu'est ce qu'une Transaction?

- Une transaction est une série de requêtes de **mises à jour** qui doivent obligatoirement être exécutées ensembles.

- Lorsque toutes les requêtes de la transaction se sont exécutées correctement la transaction est dite « **validée** » et les mises à jour sont effectivement réalisées sur la BD

« Commit » de la transaction

- Si un problème est survenu pendant l'exécution d'une requête, la transaction est « **annulée** » et aucune mise à jour n'est réalisée

« Rollback » de la transaction

Propriétés des transactions

- Dans les SGBD relationnels, les transactions doivent respecter les principes dits '**ACID**' :
 - **Atomicity**: transactions « atomiques »: le tout ou rien!
 - **Consistency**: maintien de l'intégrité référentielle
 - **Isolation**: tant qu'une transaction n'est pas validée, les autres utilisateurs ne voient pas ses effets
 - **Durability**: après validation d'une transaction, les mises à jour sont conservées de manière durable même en cas de panne

Pourquoi les transactions?

- Le meilleur exemple: un virement bancaire
 - Un virement implique deux mises à jour:
 1. Débit sur le compte demandeur
 2. Crédit sur le compte destinataire
 - C'est la loi du « tout ou rien »: si une panne survient après la mise à jour 1 et que la mise à jour 2 ne peut pas être réalisée, la BD est incohérente
 - Ces deux opérations doivent être dans la **même transaction.**

Instructions de manipulation des transactions

- Pour démarrer la définition d'une transaction:

- **START TRANSACTION;**
- ou **BEGIN;**

- Pour valider une transaction:

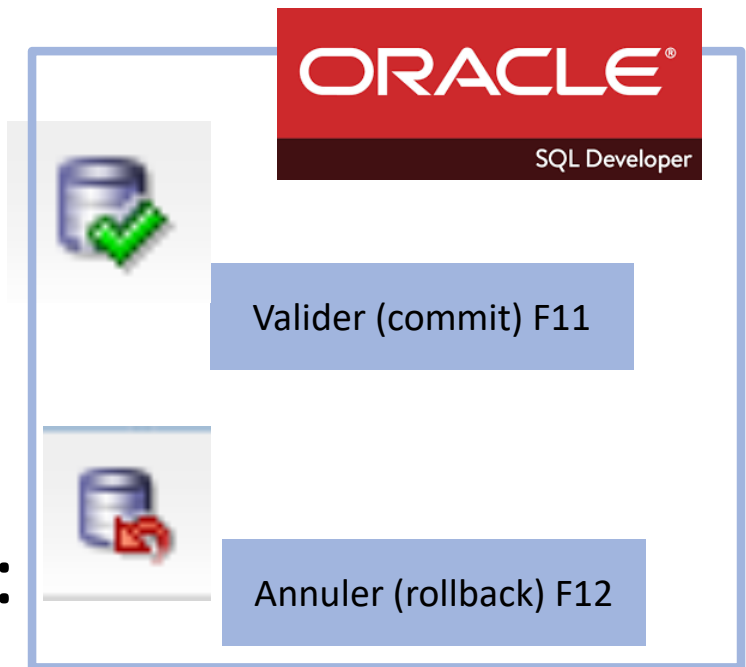
- **COMMIT;**

- Pour annuler une transaction:

- **ROLLBACK;**

- Le «commit » peut être automatique : mode **autocommit**

- **SET AUTOCOMMIT=0**



Exemple de transaction: un virement bancaire

START TRANSACTION;

Instructions de mise à jour

//UPDATE sur le compte du demandeur

//UPDATE sur le compte du receveur

COMMIT;

Tant que le « commit » n'a pas été exécuté, les modifications ne sont pas visibles dans la base de données.

Remarque

MySQL et les transactions

- Par défaut, MySQL n'est pas en mode transactionnel. Il est en mode "autocommit" : toutes les requêtes de mises à jour sont validées automatiquement
- Seules les tables InnoDB supportent le transactionnel



Exercice 6 : Test de transaction



- 1) Définissez deux requêtes d'insertion de tuples groupées dans une transaction:
 - Un tuple dans la table Bateau
 - Un tuple dans la table Croisiere impliquant le bateau que vous venez de définir.
- 2) Effectuer un rollback de la transaction
- 3) Vérifiez que les deux insertions n'ont pas été réalisées



Verrous

- Gestion des accès multi-utilisateurs aux objets de la base de données

Qu'est ce qu'un verrou?

- Dans un contexte d'utilisateurs multiples (**concurrency**), un verrou permet de **restreindre voire interdire l'accès à un élément** de la BD aux autres utilisateurs.
- Il peut être « posé » sur une table ou sur certaines lignes seulement.
- Il peut limiter:
 - L'accès en **lecture** (READ) uniquement
 - L'accès en **écriture** (mise à jour) (WRITE) dans ce cas la lecture est également impossible

Définition d'un verrou sur une table

- **LOCK TABLES** nomT [AS aliasT]

[READ | WRITE] [, ...];

Pose un verrou READ ou WRITE sur la table nomT

Exemple de pose d'un verrou de mise à jour sur la table skipper

LOCK TABLES skippers WRITE

- La présence d'un alias lors de la définition du verrou contraint les utilisateurs à l'utiliser

- **UNLOCK TABLES** : déverrouille toutes les tables

Exemple

BEGIN;

-- Verrouille la table BATEAUX en mode ROW
EXCLUSIVE

LOCK TABLE BATEAUX IN ROW EXCLUSIVE MODE;

-- Effectue des opérations sur la table

UPDATE BATEAUX SET CAPACITE = CAPACITE + 10
WHERE BATNUM = 'B001';

-- Valide les modifications et libère le
verrou

COMMIT;

Remarque

- PostgreSQL utilise un système de verrouillage automatique très performant : MVCC - Multi-Version Concurrency Control).
- Le verrouillage explicite des tables n'est donc pas obligatoire et n'est que rarement nécessaire.
- A noter :
 - Le verrouillage explicite des tables avec LOCK TABLE peut bloquer les autres transactions et affecter les performances.

A utiliser avec parcimonie

Liens



- Les cours de Witold LITWIN (Professeur Université paris Dauphine)

<http://www.lamsade.dauphine.fr/~litwin/cours98/BD-wl-11.htm>

- Bases de données relationnelles, Normalisation, algèbre
- SQL, SQL avancé

- L'introduction aux BD de Georges Gardarin (Professeur Université Paris6): Une référence dans le monde des BD!

http://georges.gardarin.free.fr/Cours_Total/1-IntroductionR.ppt