



1

Objectif

À la fin du module l'apprenti sera en capacité d'utiliser les langages,

DDL : Data Definition Language

DML : Data Manipulation Language

Propres au SQL pour écrire de requêtes afin d'accéder une base de données relationnelle.
Les exemples et exercices seront basés sur un cas d'étude

2

AGENDA



1

Le Cas

- Description du cas
- Rappel Modélisation des données
- MCD
- MLD

2

Langage SQL

- Définition
- Commandes
- Exercices

4

Sécurité de données

- Définition
- Critères
- Risques
- Techniques
- Exemples
- Exercices

3

3

Cas d'étude...



Un club de location de DVD souhaite disposer d'un système d'information pour permettre de mieux gérer son activité commerciale.

- ✓ Le club souhaite informatiser les opérations suivantes :
- ✓ prêt des films
 - ✓ vérifier le droit d'un client d'emprunter un film,
 - ✓ vérifier qu'au titre du film qu'il veut emprunter correspond à un DVD disponible dans le magasin où le prêt est demandé.
 - ✓ renseigner le client de la date de disponibilité d'un DVD.
 - ✓ enregistrer le prêt.
- ✓ retour des films
 - ✓ indiquer le prix à payer pour chaque film rendu
 - ✓ enregistrer le retour
- ✓ enregistrement d'un nouvel adhérent,
- ✓ consulter les films disponibles, les films empruntés,
- ✓ produire de statistiques afin de déterminer les films les plus demandés, les adhérents qui sont les plus gros consommateurs, etc.

4

4

Rappel ...

5

- **Modélisation des données**

Identifier :
 • Les concepts qui font partie du vocabulaire du système
 • Déterminer les identifiants

Déterminer les relations entre les concepts

Identifier les propriétés de ces objets

Définir les cardinalités

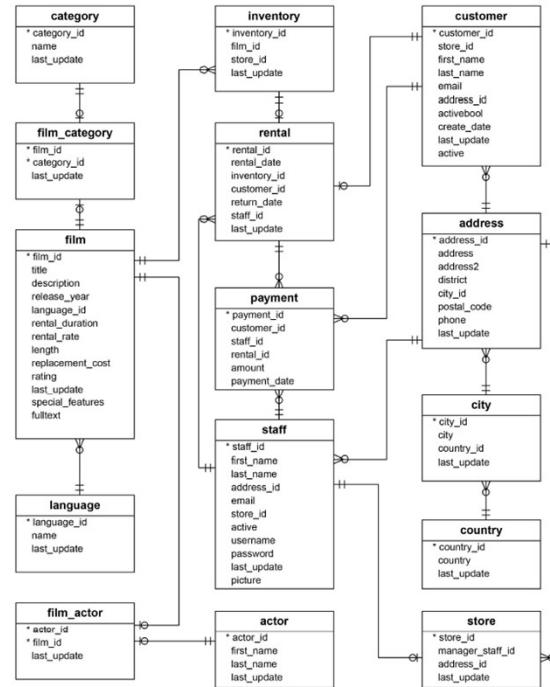
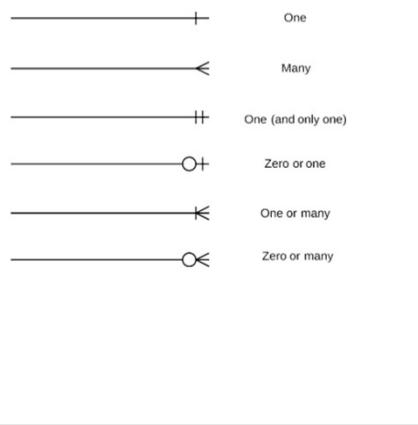
Vérifier la cohérence et la pertinence du modèle

5

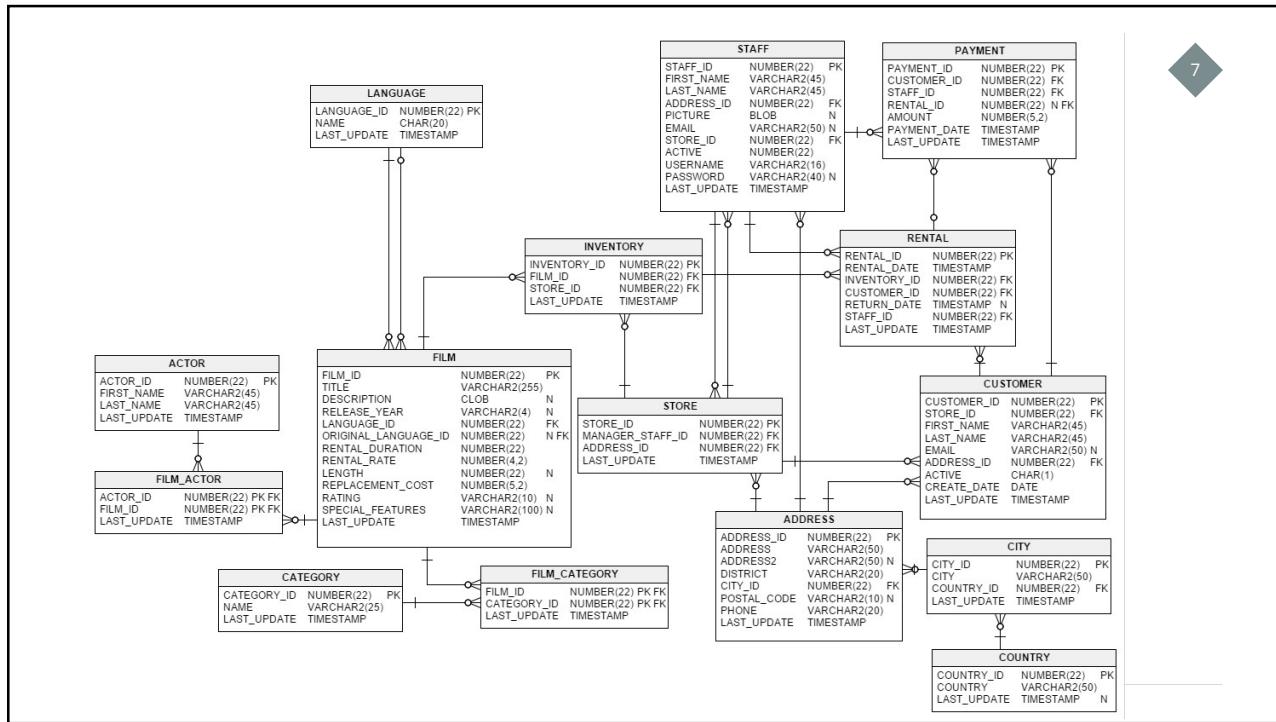
Le modèle ER

6

Rappel de la notation



6



7

7

Base de Données ... (quelques tables)

8

	actor_id	first_name	last_name
1	1	Penelope	Guiness
2	2	Nick	Wahlberg
3	3	Ed	Chase
4	4	Jennifer	Davis
5	5	Johnny	Lollobrigida
6	6	Bette	Nicholson
7	7	Grace	Mostel
8	8	Matthew	Johansson

	category_id	name
1	1	Action
2	2	Animation
3	3	Children
4	4	Classics
5	5	Comedy
6	6	Documentary
7	7	Drama
8	8	Family
9	9	Foreign
10	10	Games
11	11	Horror

	customer_id	store_id	first_name	last_name	email	address_id	activebool	create_date	last_update	active
1	1	1	Mary	Smith	mary.smith@sakilacustomer.org	5	true	2006-02-14	2013-05-26 14:49:45.738	
2	2	1	Patricia	Johnson	patricia.johnson@sakilacustomer.org	6	true	2006-02-14	2013-05-26 14:49:45.738	
3	3	1	Linda	Williams	linda.williams@sakilacustomer.org	7	true	2006-02-14	2013-05-26 14:49:45.738	
4	4	2	Barbara	Jones	barbara.jones@sakilacustomer.org	8	true	2006-02-14	2013-05-26 14:49:45.738	
5	5	1	Elizabeth	Brown	elizabeth.brown@sakilacustomer.org	9	true	2006-02-14	2013-05-26 14:49:45.738	
6	6	2	Jennifer	Davis	jennifer.davis@sakilacustomer.org	10	true	2006-02-14	2013-05-26 14:49:45.738	

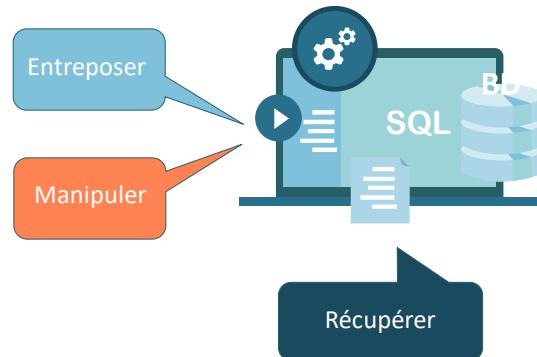
8

SQL ...

9

Structured Query Language :

- Est un langage standard pour communiquer avec une base des données :
 - règles d'écriture,
 - règles de formatage,
 - commentaires.



9

Que peut faire SQL ? ...

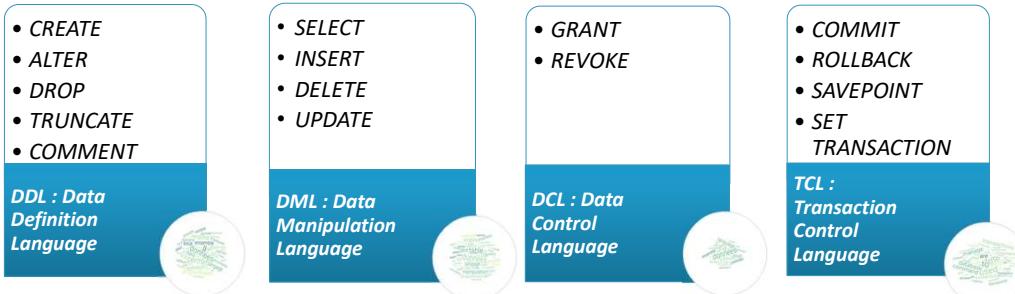
10

- | | |
|---|--|
| 01
exécuter des requêtes contre
une base de données | 06
créer de nouvelles bases
de données |
| 02
récupérer des données à
partir d'une base de données | 07
créer des procédures
stockées dans une base de
données |
| 03
insérer des enregistrements
dans une base de données | 08
créer des vues dans une
base de données |
| 04
mettre à jour les
enregistrements dans une
base de données | 09
définir des autorisations sur les
tables, procédures et vues |
| 05
supprimer des
enregistrements d'une base
de données | 10
créer de nouvelles tables
dans une base de données |

10

Type de commandes SQL ...

11



Est utilisé pour définir la structure ou le schéma de la base de données. DDL est également utilisé pour spécifier des propriétés supplémentaires des données.

Est utilisé pour gérer les données. Il existe deux types de requêtes :

- de sélection (SELECT)
- de modification (INSERT, UPDATE et DELETE)

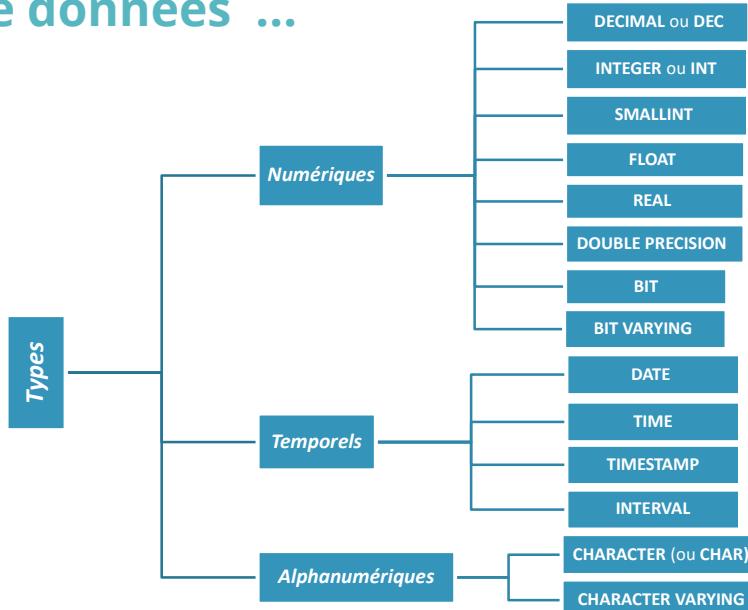
Est utilisé pour contrôler l'accès aux données d'une base de données.

Est utilisé pour gérer les transactions dans la base de données. Permettent de gérer les modifications apportées par les instructions DML.

11

Type de données ...

12



12

Opérateurs ...

13

	+ Addition
	- Soustraction
Arithmétiques	* Multiplication
	/ Division
	% Modulo
	ALL
	AND
Logiques	ANY
	BETWEEN
	EXISTS
	= Égal
	> Supérieur à
Comparateurs	< Inférieur à
	>= Supérieur ou égal à
	<= Inférieur ou égal à
	<> Différent

13

SQL : Quelques-unes des commandes ...

14

- **SELECT** - extrait des données d'une base de données
- **INSERT INTO** - insère de nouvelles données dans une base de données
- **UPDATE** - met à jour les données dans une base de données
- **DELETE** - supprime les données d'une base de données
- **JOINTURES** – combiner plusieurs tables d'une base de données
- **CREATE DATABASE** - crée une nouvelle base de données
- **CREATE TABLE** - crée une nouvelle table
- **ALTER DATABASE** - modifie une base de données
- **ALTER TABLE** - modifie une table
- **DROP TABLE** - supprime une table



Les mots clés **SQL** ne sont PAS sensibles à la casse : **select** est identique à **SELECT**

Le point-virgule est le moyen standard de séparer chaque instruction **SQL** dans les systèmes de base de données

14

SQL : commentaires ...

15

Les commentaires dans les requêtes SQL permettent de mieux s'y retrouver lorsqu'il y a plusieurs façons d'écrire les commentaires dans le langage SQL, qui dépendent notamment du Système de Gestion de Base de Données utilisées (SGBD) et de sa version.

- **Commentaire double tiret (--) et dièse (#)**

Le double tiret et le dièse permettent de faire un commentaire jusqu'à la fin de la ligne.

```
SELECT * FROM `rental`-- ceci est un commentaire
```

```
SELECT * FROM `rental`# ceci est un commentaire
```

- **Commentaire multilignes : /* et */**

Le commentaire multilignes permet d'indiquer où commence et où se termine le commentaire.

```
SELECT * FROM `rental`
```

```
/* ceci est un commentaire qui peut occuper  
plusieurs lignes */
```

15

Création / Suppression



16

16

CREATE ...

17

- **SYNTAXE :**

- **CREATE DATABASE** crée une nouvelle base des données **SQL**
- Il est nécessaire de disposer d'un rôle de **super utilisateur** ou d'un privilège spécial **CREATEDB**.



```
CREATE DATABASE nom_base_de_données ;
```

```
CREATE DATABASE dvdrental ;
```

17

CREATE ...

18

- **SYNTAXE :**

- **CREATE DATABASE** Crée une nouvelle base des données **SQL**
- **CREATE TABLE** Crée une nouvelle table dans la base des données



```
CREATE DATABASE nom_base_de_données ;
```

```
CREATE DATABASE dvdrental ;
```



```
CREATE TABLE table (
    colonne1 type_de_donnée1,
    colonne2 type_de_donnée2,
    colonne3 type_de_donnée3,
    ....);
```

• **colonne_i**, ... sont les noms des colonnes qui contiendra la table.

• **table** est le nom de la table à créer

• **Exemple** : Crée une table avec deux colonnes et définit la clé primaire

```
CREATE TABLE country (
    country_id INTEGER NOT NULL,
    country character VARYING(50) NOT NULL,
    PRIMARY KEY (ID));
```

18

DROP ...

19

- SYNTAXE :

- **DROP DATABASE**
Élimine la base de données
- **DROP TABLE** Élimine la table de la base des données



```
DROP DATABASE nom_base_de_données ;  
DROP DATABASE dev_rental;
```

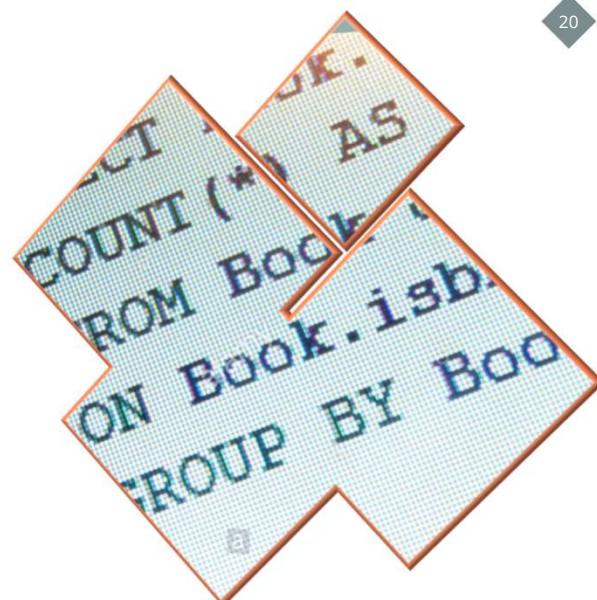


```
DROP TABLE nom_table;  
DROP TABLE film;
```

19

Sélectionner les données **SELECT ...**

20



20

SELECT ...

21

- Utilisée pour **sélectionner** les données des tables d'une base de données.
- Renvoie un tableau de résultats, les données sélectionnées.

SYNTAXE :



```
SELECT *
FROM nom_de_la_table;
```

- “ * ” Permet de sélectionner tous les champs disponibles dans la table.

- Exemple :**

- ```
• SELECT * FROM film;
```

### Résultat :

| film_id<br>[PK] integer | title<br>character varying (255) | description<br>text             | release_year<br>integer | language_id<br>smallint | rental_duration<br>smallint | rental_rate<br>numeric (4,2) | length<br>smallint | replacement_cost<br>numeric (5,2) | rating<br>mpaa_rating | last_update<br>timestamp without time zone | special_features<br>text[]             |
|-------------------------|----------------------------------|---------------------------------|-------------------------|-------------------------|-----------------------------|------------------------------|--------------------|-----------------------------------|-----------------------|--------------------------------------------|----------------------------------------|
| 133                     | Chamber Italian                  | A Fateful Reflection of a ...   | 2006                    | 1                       | 7                           | 4.99                         | 117                | 14.99                             | NC-17                 | 2013-05-26 14:50:58.951                    | {Trailers}                             |
| 384                     | Grosse Wonderful                 | A Epic Drama of a Cat An...     | 2006                    | 1                       | 5                           | 4.99                         | 49                 | 19.99                             | R                     | 2013-05-26 14:50:58.951                    | {"Behind the Scenes"}                  |
| 8                       | Airport Pollock                  | A Epic Tale of a Moose A...     | 2006                    | 1                       | 6                           | 4.99                         | 54                 | 15.99                             | R                     | 2013-05-26 14:50:58.951                    | {Trailers}                             |
| 98                      | Bright Encounters                | A Fateful Yarn of a Lumbe...    | 2006                    | 1                       | 4                           | 4.99                         | 73                 | 12.99                             | PG-13                 | 2013-05-26 14:50:58.951                    | {Trailers}                             |
| 1                       | Academy Dinosaur                 | A Epic Drama of a Femin...<br>i | 2006                    | 1                       | 6                           | 0.99                         | 86                 | 20.99                             | PG                    | 2013-05-26 14:50:58.951                    | {"Deleted Scenes";"Behind the Scenes"} |
| 2                       | Ace Goldfinger                   | A Astounding Epistle of a ...   | 2006                    | 1                       | 3                           | 4.99                         | 48                 | 12.99                             | G                     | 2013-05-26 14:50:58.951                    | {Trailers;Deleted Scenes}              |
| 3                       | Adaptation Holes                 | A Astounding Reflection ...     | 2006                    | 1                       | 7                           | 2.99                         | 50                 | 18.99                             | NC-17                 | 2013-05-26 14:50:58.951                    | {Trailers;Deleted Scenes}              |
| 4                       | Affair Prejudice                 | A Fanciful Documentary o...     | 2006                    | 1                       | 5                           | 2.99                         | 117                | 26.99                             | G                     | 2013-05-26 14:50:58.951                    | (Commentaries;Behind the Scenes)       |
| 5                       | African Egg                      | A Fast-Paced Documenta...       | 2006                    | 1                       | 6                           | 2.99                         | 130                | 22.99                             | G                     | 2013-05-26 14:50:58.951                    | {"Deleted Scenes"}                     |

Les exemples ont été testés avec le SGBD "Postgres", il peut y avoir quelques différences dans la syntaxe

21

## SELECT ...

22

- Utilisée pour **sélectionner** les données d'une base de données.
- Renvoie un tableau de résultats, les données sélectionnées.

### SYNTAXE :



```
SELECT colonne1, colonne2, ...
FROM nom_de_la_table;
```

- colonne1, colonne2, ...** sont les noms de champ de la table à partir de laquelle les données seront sélectionnées.

- Exemple :**

- ```
• SELECT title, length
FROM film
```

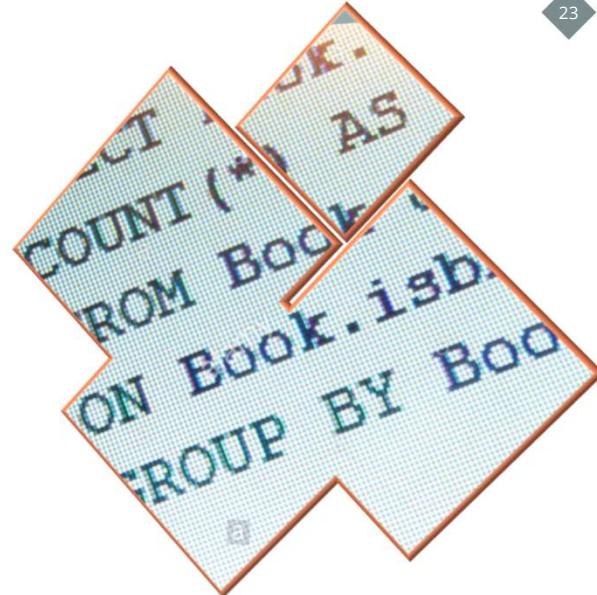
Résultat :

title character varying (255)	length smallint
Chamber Italian	117
Grosse Wonderful	49
Airport Pollock	54
Bright Encounters	73
Academy Dinosaur	86
Ace Goldfinger	48
Adaptation Holes	50
Affair Prejudice	117
African Egg	130

Les exemples ont été testés avec le SGBD "Postgres", il peut y avoir quelques différences dans la syntaxe

22

Plus de variantes pour le SELECT ...



23

SELECT DISTINCT ...

- Utilisée pour **sélectionner** les données d'une base de données.
- Renvoie seulement les valeurs distinctes (différentes)
- Renvoie un tableau de résultats, les données sélectionnées.

SYNTAXE :



```
SELECT DISTINCT colonne1, colonne2, ...
FROM nom_de_la_table;
```

- colonne1, colonne2, ...** sont les noms de champs de la table à partir de laquelle les données seront sélectionnées.

- Exemple :**

- `SELECT DISTINCT first_name
 FROM actor;`

Les exemples ont été testés avec le SGBD "Postgres", il peut y avoir quelques différences dans la syntaxe

first_name	character varying (45)
Adam	
Adam	
Al	
Alan	
Albert	
Albert	
Alec	
Angela	
Angela	
Angelina	
Anne	
Audrey	
Bela	
Ben	
Ben	
Ben	

`SELECT (200)`

first_name	character varying (45)
Adam	
Al	
Alan	
Albert	
Alec	
Angela	
Angelina	
Anne	
Audrey	
Bela	
Ben	
Bette	
Bob	
Burt	
Cameron	
Carmen	

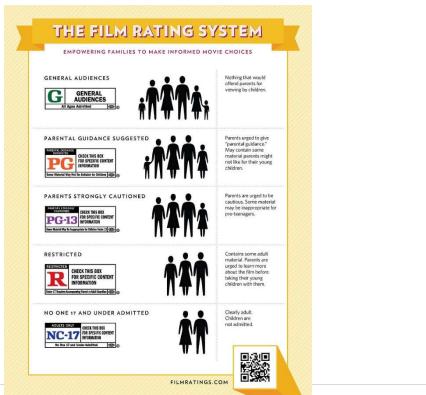
`SELECT DISTINCT (128)`

24

SELECT ...

25

Exercice : Écrire une requête SQL pour connaître les titres et le classement de tous les films (regarder la table afin de connaître le nom de la colonne)



25

WHERE ...

27

- Utilisée pour **filtrer** les enregistrements
- Extrait **uniquement** les enregistrements qui remplissent la **condition spécifiée**.

- SYNTAXE :



```
SELECT colonne1, colonne2, ...
FROM nom_de_la_table
WHERE condition;
```

- colonne1, colonne2, ...** sont les noms de champs de la table à partir de laquelle les données seront sélectionnées.
- WHERE** indique la condition à vérifier
- Exemple :**
 - SELECT title**
FROM film
WHERE film_id = 100

27

WHERE ...

28

Exercice : Écrire une requête pour connaître le titre des films dont la durée est égale à 100 minutes

Table d'origine

title	character varying (255)	length	smallint
Darkness War		99	
Armageddon Lost		99	
Witches Panic		100	
Bilko Anonymous		100	
Gun Bonnie		100	
Behavior Runaway		100	
Happiness United		100	
Pirates Roxanne		100	
Wyoming Storm		100	
Punk Divorce		100	
Divine Resurrection		100	
Deliverance Mulholland		100	
Flatliners Killer		100	
Hyde Doctor		100	
Runner Madigan		101	
Roof Champion		101	

Résultat à obtenir

title	character varying (255)	length	smallint
1 Behavior Runaway		100	
2 Bilko Anonymous		100	
3 Deliverance Mulholland		100	
4 Divine Resurrection		100	
5 Flatliners Killer		100	
6 Gun Bonnie		100	
7 Happiness United		100	
8 Hyde Doctor		100	
9 Pirates Roxanne		100	
10 Punk Divorce		100	
11 Witches Panic		100	
12 Wyoming Storm		100	

28

ORDER BY ...

30

- Utilisé pour trier l'ensemble des résultats dans l'ordre ascendant ou descendant
- Le tri est **ascendant (ASC) par défaut**
- Il faut utiliser le mot clé **DESC** pour trier dans l'ordre descendant

• SYNTAXE



```
SELECT colonne1, colonne2, ...
FROM nom_de_la_table
ORDER BY colonne1,... ASC|DESC;
```

- colonne1, colonne2, ...** sont les noms de champ de la table à partir de laquelle les données seront sélectionnées.
- Colonne1,...** indique la colonne dont les valeurs seront utilisées pour le tri
- Exemple :**

```
SELECT title, length
FROM film
ORDER BY length
```

Les exemples ont été testés avec le SGBD "Postgres", il peut y avoir quelques différences dans la syntaxe

30

SELECT, WHERE, ORDER BY ...

31

Exercice :

Écrire une requête pour connaître la liste des **titres des films** dont la **durée est supérieure à 50 minutes**.

La liste doit être **triée par la durée**.

31

SELECT, WHERE, ORDER BY ...

33

Corrigé : Écrire une requête pour connaître la liste des titres des films dont la durée est supérieure à 50 minutes. La liste doit être triée par la durée

```
SELECT title, length
FROM film
WHERE length > 50
ORDER BY length, title
```

Cette variante trie également alphabétiquement par le titre du film à l'intérieur de chaque sous-groupe

title	length
Champion Flatliners	51
Deep Crusade	51
English Bulworth	51
Excitement Eve	51
Frisco Forrest	51
Hall Cassidy	51
Simon North	51
Caddyshack Jedi	52
Harper Dying	52
Lust Lock	52
Side Ark	52
Spartacus Cheaper	52
Trojan Tomorrow	52
Westward Seabiscuit	52
Beneath Rush	53
Cabin Flash	53

Les exemples ont été testés avec le SGBD "Postgres", il peut y avoir quelques différences dans la syntaxe

33

GROUP BY ...

34

- SYNTAXE



- Utilisé pour **grouper** l'ensemble des résultats
- La sortie est combinée en groupes de lignes qui correspondent sur une ou plusieurs valeurs

```
SELECT colonne1, colonne2, ...
FROM nom_de_la_table
GROUP BY colonne1, ...;
```

- *colonne1, colonne2, ...* sont les noms de champ de la table à partir de laquelle les données seront sélectionnées.
- *Colonne1, ...* indique la colonne dont les valeurs seront utilisées pour le tri
- **Exemple :**

```
SELECT customer_id, amount
FROM payment
GROUP BY customer_id, amount
ORDER BY customer_id
```

Les exemples ont été testés avec le SGBD "Postgres", il peut y avoir quelques différences dans la syntaxe

34

SELECT, WHERE, GROUP BY, ORDER BY ...

35

Exercice :

Écrire une requête pour connaître la liste des clients (id) qui ont retourné les films après le 28/05/2005

35

SELECT, WHERE, GROUP BY, ORDER BY ...

37

Corrigé (syntaxe MySQL):

Écrire une requête pour connaître la liste des clients qui ont retourné les films en retard. On suppose que la date actuelle est le 28/05/2005

```
SELECT customer_id, rental_id, return_date
FROM rental
WHERE return_date > cast(FROM_UNIXTIME('2005-08-28') as datetime)
GROUP BY rental_id, return_date, customer_id
ORDER BY return_date
```

37

TRAITEMENT DES DATES ...

38

SQL fournit plusieurs fonctions qui permettent de manipuler des données à des heures et des dates précises.

```
SELECT customer_id, rental_id, return_date
FROM rental
WHERE return_date > cast(FROM_UNIXTIME('2005-08-28') as
datetime)
GROUP BY rental_id, return_date, customer_id
ORDER BY return_date
```

38

LIKE ...

39

La clause **LIKE** est utilisée dans SQL pour effectuer des recherches de motifs dans une colonne de texte.

- LIKE 'pattern%'
- LIKE '%pattern'
- LIKE '%pattern%'
- LIKE '%p%n'
- LIKE '_pattern'

- SYNTAXE

```
SELECT colonne1, colonne2, ...
FROM nom_de_la_table
WHERE colonne1 LIKE pattern;
```

- **colonne1, colonne2, ...** sont les noms de champ de la table à partir de laquelle les données seront sélectionnées.
- **Colonne1,...** indique la colonne dont les valeurs seront utilisées pour la comparaison
- **Pattern** le motif recherché dans la colonne.
- **Exemple :**

```
SELECT title
FROM film
WHERE title LIKE '%great%';
```

39

COUNT, AVG, SUM ...

40

- SYNTAXE :

- La fonction **COUNT()** renvoie le nombre de lignes correspondant à un critère spécifié.
- La fonction **AVG()** renvoie la valeur moyenne d'une colonne numérique.
- La fonction **SUM()** renvoie la somme totale d'une colonne numérique

COUNT()	SELECT COUNT(colonne) FROM table WHERE condition;
----------------	---

AVG()	SELECT AVG (colonne) FROM table WHERE condition;
--------------	--

SUM()	SELECT SUM (colonne) FROM table WHERE condition;
--------------	--

colonne est le nom du champ (colonne) de la table à utiliser pour le calcul.

table est la table à partir de laquelle les colonnes seront sélectionnées

40

COUNT

Un extrait des résultats est montré

41

Exemple :

Écrire une requête pour compter les films dont la durée est supérieure à 50 minutes

```
SELECT film.length, COUNT(length)
FROM film
WHERE film.length > 50
GROUP BY film.length
ORDER BY film.length DESC
```

Les exemples ont été testés avec le SGBD "Postgres", il peut y avoir quelques différences dans la syntaxe

	length smallint	count bigint
1	185	10
2	184	8
3	183	5
4	182	6
5	181	10
6	180	7
7	179	13
8	178	10
9	177	6
10	176	10
11	175	6
12	174	6
13	173	7
14	172	8
15	171	8
16	170	4
17	169	6

41

COUNT

Exemple :

Écrire une requête pour compter les films dont la durée est supérieure à 50 minutes

```
SELECT film.length, COUNT(length)
FROM film
WHERE film.length > 50
GROUP BY film.length
ORDER BY film.length
```

Les résultats peuvent être triés par la durée

	length smallint	count bigint
51	7	
52	7	
53	9	
54	6	
55	2	
56	5	
57	7	
58	7	
59	9	
60	8	
61	10	
62	6	
63	9	
64	9	
65	7	
66	2	
67	8	
68	5	
69	6	
70	7	

42

42

COUNT ...

43

Exemple : Écrire une requête pour calculer la quantité des paiements du client dont l'id = 10

```
SELECT customer_id, COUNT(amount)
FROM payment
WHERE customer_id = 10
GROUP BY customer_id
```

```
SELECT customer_id, amount, COUNT(amount)
FROM payment
WHERE customer_id = 10
GROUP BY customer_id, amount
```

The screenshot shows a table with three columns: customer_id (smallint), amount (numeric(5,2)), and count (bigint). There are 9 rows of data, each corresponding to a payment from customer_id 10. The count column shows the frequency of payments for each amount.

customer_id	amount	count
1	0.99	4
2	1.99	1
3	2.99	7
4	3.99	2
5	4.99	6
6	5.99	1
7	6.99	1
8	7.99	1
9	8.99	1

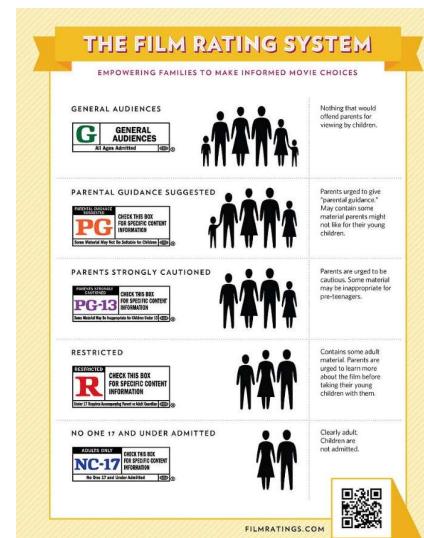
43

COUNT

44

Exercice :

Écrire une requête pour **compter le nombre de films** dans chacune des classifications cinématographiques



44

COUNT, ORDER BY...

46

Exercice :

1. Écrire une requête pour connaitre la quantité de paiements réalisé par chaque client
2. Modifier la requête pour afficher les paiements regroupés par le montant
3. Modifier la requête pour afficher les résultats triés par l'identifiant du client

46

AVG...

49

Exemple : Écrire une requête pour calculer la moyenne des paiements des clients pour les locations des films.

```
SELECT customer_id, AVG(amount)
FROM payment
GROUP BY customer_id
ORDER BY customer_id
```

customer_id	avg
smallint	numeric
1	3.8233333333333333
2	4.7592307692307692
3	5.4483333333333333
4	3.7172727272727273
5	3.8471428571428571
6	3.3900000000000000
7	4.6685714285714286
8	3.7291304347826087
9	3.9400000000000000
10	3.9483333333333333
11	4.3378260869565217
12	3.6053846153846154
13	4.8788888888888889
14	4.2073913043478261
15	4.2087500000000000
16	4.3900000000000000
17	4.8788888888888889
18	4.1478947368421053
19	5.4900000000000000
20	3.8418518518518519

Les exemples ont été testés avec le SGBD "Postgres", il peut y avoir quelques différences dans la syntaxe

49

AVG...

Exemple : Écrire une requête pour calculer la moyenne des paiements des clients pour les locations des films.

```
SELECT customer_id,
       ROUND(AVG(amount), 2)
  FROM payment
 GROUP BY customer_id
 ORDER BY customer_id
```

Les résultats peuvent être arrondis

customer_id	round
smallint	numeric
1	3.82
2	4.76
3	5.45
4	3.72
5	3.85
6	3.39
7	4.67
8	3.73
9	3.94
10	3.95
11	4.34
12	3.61
13	4.88
14	4.21
15	4.21
16	4.39
17	4.88
18	4.15

Les exemples ont été testés avec le SGBD "Postgres", il peut y avoir quelques différences dans la syntaxe

50

50

SUM ...

Exemple : Écrire une requête pour calculer le montant total des paiements du client dont l'id = 341

```
SELECT customer_id, SUM(amount)
  FROM payment
 WHERE customer_id = 341
 GROUP BY customer_id
```

customer_id	sum
smallint	numeric
341	103.78

51

51

HAVING ...

52

- Permet de sélectionner les colonnes de la table "nom_de_la_table" les lignes dont la condition de HAVING est respectée.
- Presque similaire à WHERE à la seule différence que HAVING permet de filtrer en utilisant des fonctions telles que SUM(), COUNT(), AVG(), MIN() ou MAX().

SYNTAXE

```
SELECT colonne1, colonne2, ...
FROM nom_de_la_table
GROUPE BY colonne1, ...
HAVING condition;
```

- colonne1, colonne2, ... sont les noms de champ de la table à partir de laquelle les données seront sélectionnées.
- colonne1,... indique la colonne dont les valeurs seront utilisées pour le tri
- condition,... indique la condition
- Exemple :

```
SELECT customer_id, SUM(amount)
FROM payment
GROUP BY customer_id
HAVING SUM(amount) > 20
ORDER BY customer_id
```

52

HAVING ...

53

Résultat : requête pour obtenir les clients qui ont dépensé plus de 20 euros en location

```
SELECT customer_id, SUM(amount)
FROM payment
GROUP BY customer_id
HAVING SUM(amount) > 20
ORDER BY customer_id
```

customer_id	sum
smallint	numeric
1	114.70
2	123.74
3	130.76
4	81.78
5	134.65
6	84.75
7	130.72
8	85.77
9	78.80
10	94.76
11	99.77
12	93.74
13	131.73
14	96.77
15	134.68
16	109.75
17	87.82
18	78.81
19	98.82
20	103.73

Un extrait des résultats est montré

Les exemples ont été testés avec le SGBD "Postgres", il peut y avoir quelques différences dans la syntaxe

53

EXERCICE ...

54

Écrire la requête SQL permettant d'afficher les identifiants des clients qui ont plus de deux retours après la date du "28-05-2005".

54

Combiner plusieurs tables (JOINTURE)...



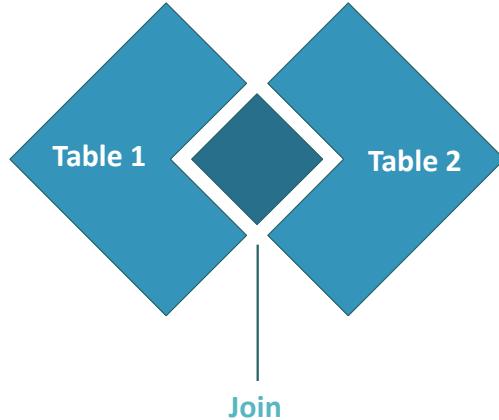
57

57

JOIN ...

58

- Utilisée pour **combiner** des lignes à partir de deux tables ou plus, sur la base d'une colonne commune entre ces tables
- Aussi appelé ***inner join***



58

JOIN ...

59

- Utilisée pour **combiner** des lignes à partir de deux tables ou plus, sur la base d'une colonne commune entre les tables

Une clause de la forme **USING (a, b, ...)** est un raccourci pour **ON**

- SYNTAXE :**

```
SELECT colonne1, colonne2, ...
FROM table1
JOIN table2
ON table1.nom_colonne1 = table2.nom_colonne2 ;
```

- colonne1, colonne2, ...** sont les noms de champ de la table à partir de laquelle les données seront sélectionnées.
- table1, table2**, sont les tables à partir desquelles les colonnes seront sélectionnées
- ON** précise les éléments communs
- Exemple :**

```
SELECT C.first_name, C.customer_id,
       SUM(P.amount) AS total_payment
  FROM customer C
  JOIN payment P
 USING (customer_id)
 GROUP BY C.first_name, C.customer_id
 ORDER BY total_payment DESC
```

Les exemples ont été testés avec le SGBD "Postgres", il peut y avoir quelques différences dans la syntaxe

59

EXERCICE ...

60

- Connaître les adresses mail des 10 meilleurs clients pour les récompenser
- Modifier le code pour connaître les adresses mail des 10 clients moins assidus pour leur envoyer des offres promotionnelles
- Connaître le titre du film loué dont son identifiant dans l'inventaire est "367"

60

JOIN ...

66

Différents types

1 *Left Join*

Renvoie tous les enregistrements de la table de gauche et les enregistrements appariés à partir de la table droite

2 *Right Join*

Renvoie tous les enregistrements à partir de la table de droite, et les enregistrements appariés de la table de gauche

3 *Full Outer Join*

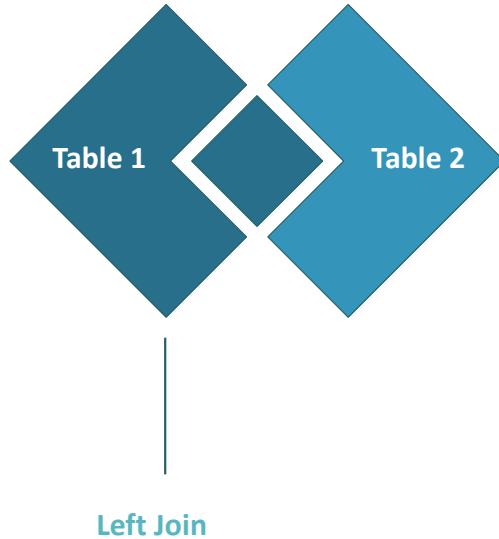
Renvoie tous les enregistrements lorsqu'il y a une correspondance dans la table gauche ou droite

66

LEFT JOIN ...

67

- Renvoie tous les enregistrements de la table gauche (**Table1**) et des enregistrements appariés de la table droite (**Table2**).



67

LEFT JOIN ...

68

- renvoie tous les enregistrements de la table gauche (*table1*) et des enregistrements appariés à partir de la table droite (*table2*).

SYNTAXE :



```
SELECT colonne1, colonne2, ...
FROM table1
LEFT JOIN table2
ON table1.nom_colonne1 = table2.nom_colonne2;
```

• *colonne1, colonne2, ...* sont les noms de champ de la table à partir de laquelle les données seront sélectionnées.

• *table1, table2*, sont les tables à partir desquelles les colonnes seront sélectionnées

• *ON* précise quelle est la colonne en commun

• **Exemple :**

```
SELECT F.title, rental_id
FROM film F
LEFT JOIN inventory
USING (film_id)
LEFT JOIN rental
USING (inventory_id)
ORDER BY title, rental_id DESC
```

68

LEFT JOIN ...

Exemple

```
SELECT F.title, rental_id
FROM film F
LEFT JOIN inventory
USING (film_id)
LEFT JOIN rental
USING (inventory_id)
ORDER BY title, rental_id DESC
```

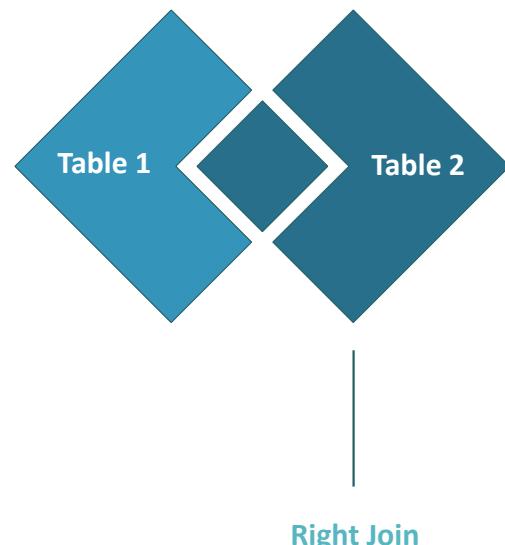
Renvoie tous les films
même s'ils n'ont pas été
loués

title	rental_id
ACADEMY DINOSAUR	16053
ACADEMY DINOSAUR	16052
ACADEMY DINOSAUR	15453
ACADEMY DINOSAUR	15421
ACADEMY DINOSAUR	14798
ACADEMY DINOSAUR	14714
ACADEMY DINOSAUR	14624
ACADEMY DINOSAUR	14098
ACADEMY DINOSAUR	12651
ACADEMY DINOSAUR	11433
ACADEMY DINOSAUR	10883
ACADEMY DINOSAUR	10141
ACADEMY DINOSAUR	10126
ACADEMY DINOSAUR	9449
ACADEMY DINOSAUR	8510
ACADEMY DINOSAUR	7168
ACADEMY DINOSAUR	5766
ACADEMY DINOSAUR	4863
ACADEMY DINOSAUR	4390
ACADEMY DINOSAUR	4187
ACADEMY DINOSAUR	3201
ACADEMY DINOSAUR	2117
ACADEMY DINOSAUR	1210
ACADEMY DINOSAUR	972
ACADEMY DINOSAUR	361
ACADEMY DINOSAUR	NULL

69

RIGHT JOIN ...

- Renvoie tous les enregistrements de la table droite (**Table2**) et des enregistrements appariés de la table gauche (**Table1**).



70

RIGHT JOIN ...

71

- Renvoie tous les enregistrements de la table droite (*table2*) et des enregistrements appariés de la table gauche (*table1*).

- SYNTAXE :



```
SELECT colonne1, colonne2, ...
FROM table1
RIGHT JOIN table2
ON table1.nom_colonne1 = table2.nom_colonne2;
```

- colonne1, colonne2, ...* sont les noms des champs des tables à partir de desquelles les données seront sélectionnées.
- table1, table2*, sont les tables à partir desquelles les colonnes seront sélectionnées
- ON** précise quelle est la colonne en commun
- Exemple :**

```
SELECT rental_id, payment_id
FROM payment
RIGHT JOIN rental
USING (rental_id)
ORDER BY rental_id
```

71

RIGHT JOIN ...

72

Exemple

```
SELECT rental_id, payment_id
FROM payment
RIGHT JOIN rental
USING (rental_id)
ORDER BY rental_id
```

Renvoi toutes les locations même celles qui n'ont pas été payés

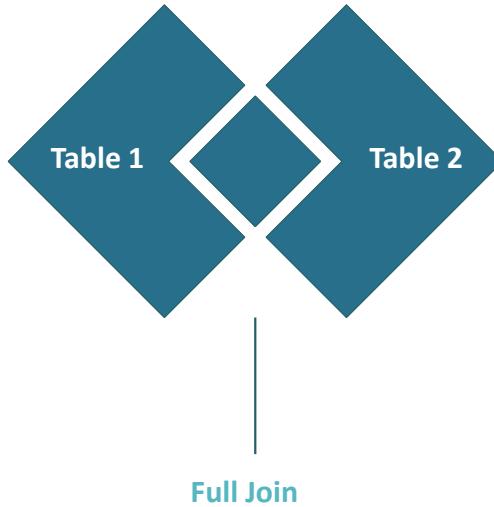
rental_id	payment_id
1	424
1	3504
1	7011
1	10840
2	NULL
3	11032
4	8987
5	6003
6	NULL
7	7274
8	6440
9	3386
10	10785
11	3831
12	7044
13	9014
14	NULL
15	8623
16	8554
17	NULL
18	490
19	NULL
20	5020
21	10499
22	NULL

72

FULL JOIN ...

73

- Renvoie tous les enregistrements en cas de correspondance dans les enregistrements de table gauche (**Table1**) ou droite (**Table2**).



73

FULL JOIN ...

74

- Renvoie tous les enregistrements en cas de correspondance dans les enregistrements de table gauche (**table1**) ou droite (**table2**).



Les FULL JOIN multiples ne sont pas supportés dans MySQL

- SYNTAXE :

```
SELECT colonne1, colonne2, ...
FROM table1
FULL JOIN table2
ON table1.nom_colonne1 = table2.nom_colonne2;
```

• **colonne1, colonne2, ...** sont les noms des champs des tables à partir desquelles les données seront sélectionnées.

• **table1, table2**, sont les tables à partir desquelles les colonnes seront sélectionnées

• **ON** précise quelle est la colonne en commun

• **Exemple :**

```
SELECT title, rental_id, payment_id
FROM film
FULL JOIN inventory
USING(film_id)
FULL JOIN rental
USING (inventory_id)
FULL JOIN payment
USING (rental_id)
GROUP BY title, rental_id, payment_id
```

74

FULL JOIN ...

75

RÉSULTAT (PostgreSQL) :

- Renvoie tous les enregistrements en cas de correspondance dans les enregistrements de table gauche (**table1**) ou droite (**table2**).

260	Opposite Necklace	15780	32055
261	Angels Life	6900	25441
262	Magnolia Forrester	15081	19964
263	Grit Clockwork	7332	30715
264	Frankenstein Stranger	[null]	[null]
265	Titanic Boondock	11782	31942
266	Center Dinosaur	11182	23202
267	Splendor Patton	9103	25293
268	Bright Encounters	3957	26835

Les exemples ont été testés avec le SGBD "Postgres", il peut y avoir quelques différences dans la syntaxe

75

JOINTURE ...

76

Exercice :

Écrire la requête SQL permettant de connaître quel est le prix de location moyen pour chaque genre ? (classés de façon descendante)

76

JOINTURE ...

Écrire la requête SQL permettant de connaître quel est le prix de location moyen pour chaque genre (classés de façon descendante)

Résultat attendu

+ Options	genre	Average_rental_rate	▼ 1
	Games	3.25	
	Travel	3.24	
	Sci-Fi	3.22	
	Comedy	3.16	
	Sports	3.13	
	New	3.12	
	Foreign	3.10	
	Horror	3.03	
	Drama	3.02	
	Music	2.95	
	Children	2.89	
	Animation	2.81	
	Family	2.76	
	Classics	2.74	
	Documentary	2.67	
	Action	2.65	

77

JOINTURE ...

Exercice :

Écrire la requête SQL permettant de connaître la liste des locations en retard (à la date 28/05/2008) afin que les clients puissent être contactés et invités à retourner leurs films en retard.

79

JOINTURE ...

80

customer	email	title	date_prevue
OLVERA, DWAYNE	DWAYNE.OLVERA@sakilacustomer.org	ACADEMY DINOSAUR	2005-08-27 00:30:32
HUEY, BRANDON	BRANDON.HUEY@sakilacustomer.org	ACE GOLDFINGER	2006-02-17 15:16:03
OWENS, CARMEN	CARMEN.OWENS@sakilacustomer.org	AFFAIR PREJUDICE	2006-02-19 15:16:03
HANNON, SETH	SETH.HANNON@sakilacustomer.org	AFRICAN EGG	2006-02-20 15:16:03
COLE, TRACY	TRACY COLE@sakilacustomer.org	ALI FOREVER	2006-02-18 15:16:03
DEAN, MARCIA	MARCIA.DEAN@sakilacustomer.org	ALONE TRIP	2006-02-17 15:16:03
VINES, CECIL	CECIL.VINES@sakilacustomer.org	AMADEUS HOLY	2006-02-20 15:16:03
TURNER, MARIE	MARIE.TURNER@sakilacustomer.org	AMERICAN CIRCUS	2006-02-17 15:16:03
GILLILAND, JOE	JOE.GILLILAND@sakilacustomer.org	AMISTAD MIDSUMMER	2006-02-20 15:16:03
BAUGH, EDWARD	EDWARD.BAUGH@sakilacustomer.org	ARMAGEDDON LOST	2006-02-19 15:16:03
FRANKLIN, BETH	BETH.FRANKLIN@sakilacustomer.org	BAKED CLEOPATRA	2006-02-17 15:16:03
WILLIAMSON, GINA	GINA.WILLIAMSON@sakilacustomer.org	BANG KWAI	2006-02-19 15:16:03
ELLINGTON, MELVIN	MELVIN.ELLINGTON@sakilacustomer.org	BASIC EASY	2006-02-18 15:16:03
GREGORY, SONIA	SONIA.GREGORY@sakilacustomer.org	BERETS AGENT	2006-02-19 15:16:03
WOODS, FLORENCE	FLORENCE.WOODS@sakilacustomer.org	BLADE POLISH	2006-02-19 15:16:03
WREN, TYLER	TYLER.WREN@sakilacustomer.org	BLANKET BEVERLY	2006-02-21 15:16:03
BAILEY, MILDRED	MILDRED.BAILEY@sakilacustomer.org	BOOGIE AMELIE	2006-02-20 15:16:03
HILL, ANNA	ANNA.HILL@sakilacustomer.org	BOULEVARD MOB	2006-02-17 15:16:03
HAYES, ROBIN	ROBIN.HAYES@sakilacustomer.org	BOUND CHEAPER	2006-02-19 15:16:03
BUTLER, LOIS	LOIS.BUTLER@sakilacustomer.org	BUBBLE GROSSE	2006-02-18 15:16:03
ANDERSON, LISA	LISA.ANDERSON@sakilacustomer.org	BULL SHAWSHANK	2006-02-20 15:16:03
STANFIELD, SERGIO	SERGIO.STANFIELD@sakilacustomer.org	CAMELOT VACATION	2006-02-17 15:16:03
HARRIS, HELEN	HELEN.HARRIS@sakilacustomer.org	CANDIDATE PERDITION	2006-02-18 15:16:03
HERNANDEZ, ANGELA	ANGELA.HERNANDEZ@sakilacustomer.org	CANYON STOCK	2006-02-21 15:16:03
SHELTON, DIANNE	DIANNE.SHELTON@sakilacustomer.org	CAT CONEHEADS	2006-02-19 15:16:03

Résultat attendu :

Écrire la requête SQL permettant de connaître la liste des locations en retard (**à la date 28/05/2008**) afin que les clients puissent être contactés et invités à retourner leurs films en retard.

Les dates prévues de retour sont antérieures à la date donnée, ce qui veut dire que les films n'ont pas été rendus

80

JOINTURE ...

84

Exercice :

Connaître le nom et le prénom de l'acteur qui joue dans la majorité des films

84

JOINTURE ...

86

Exercices :

- Quels sont les genres les plus / moins loués (demandés) et quels sont leurs ventes totales?
- Écrivez une requête qui renvoie l'identifiant du magasin, l'année et le mois et le nombre de commandes de location que chaque magasin a traitées pour le mois indiqué.

86

UNION ...

87

- Utilisé pour combiner l'ensemble de résultats de deux ou plusieurs instructions SELECT
- Par défaut, sélectionne uniquement des valeurs distinctes
- Pour autoriser les valeurs en double, utilisez UNION ALL



SYNTAXE :

```
SELECT colonne1, colonne2, ... FROM table1
```

```
UNION
```

```
SELECT colonne1, colonne2, ... FROM table2
```

- **colonne1, colonne2, ...** sont les noms des champs des tables à partir desquelles les données seront sélectionnées.

- **table1 , table2**, sont les tables à partir desquelles les colonnes seront sélectionnées

- **Exemple :**

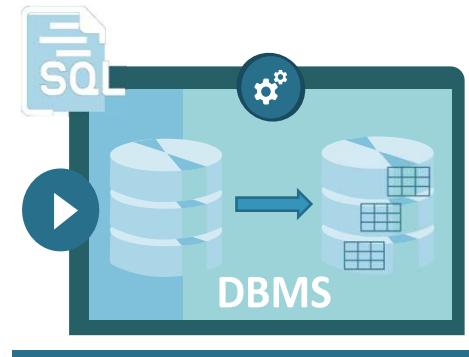
```
SELECT Nom_Region FROM REGION
```

```
UNION
```

```
SELECT Nom_Domaine FROM DOMAINE
```

87

Modification des données tables



88

88

INSERT INTO ...

- SYNTAXE

- Utilisée pour insérer des nouveaux enregistrements dans une table
- Forme 1 :**
 - Spécifier à la fois les **noms des colonnes** et les **valeurs à insérer**



```
INSERT INTO nom_de_la_table
(colonne1, colonne2, ...)
VALUES (value1, value2,...);
```

- colonne1, colonne2, ...* sont les noms de champ de la table où les valeurs seront insérées
- value1, value2, ...* sont les valeurs à insérer
- Exemple :**

```
INSERT INTO rental (rental_date,
inventory_id, customer_id, staff_id)
VALUES (NOW(), 1, 1, 1);
```

89

89

INSERT INTO ...

90

- Utilisée pour insérer des nouveaux enregistrements dans une table
- **Forme 2 :**
 - spécifier seulement les valeurs à insérer
 - ajoute les valeurs à toutes les colonnes de la table
 - s'assurer que l'ordre des valeurs dans la requête est le même que celui des colonnes de la table

SYNTAXE



```
INSERT INTO nom_de_la_table
VALUES (value1, value2,...);
```

- **value1, value2, ...** sont les valeurs à insérer
- Exemple :

```
INSERT INTO rental
VALUES(nextval('public.rental_rental_id_seq
'::regclass),NOW(), 10, 1, '2022-02-10',1)
```



90

INSERT INTO ...

91

Exercice :

- Écrire la requête pour indiquer que le client "Mary Smith" loue le film "*Academy Dinosaur*" de Mike Hillyer aujourd'hui . La transaction a été traité par l'employé dont l'identifiant est 1.
- A quelle date doit être rendu le film ?

91

INSERT INTO ...

98

Exercice : écrire la requête qui permet d'enregistrer le paiement de la location d'un film dont l'identifiant de l'inventaire est 10. Le client qui avait loué le film se nomme "Amber Dixon". L'identifiant de l'employé qui gère la transaction est "1"

Pour enregistrer le paiement d'un film, il faut ajouter un nouvel enregistrement à la table **payment**

- Décrivez les étapes
- Implémentez la requête.

98

INSERT INTO ...

99

Étapes :

- Récupérer l'identifiant du client **customer_id**
- Récupérer l'identifiant de la location **rental_id**
- Calculer le montant à payer (**amount**)
 - Calculer le numéro de jours à partir de la date de location (**rental_date**) et de la date de retour (**return_date**) et les multiplier par la tarif journalière (**rental_rate**) du film loué, les informations du film peuvent être récupérées à partir de l'identifiant dans l'inventaire.
- Insérer le nouvel enregistrement dans la table **payment** avec la date de paiement fixée à aujourd'hui

Structure de la table payment :

payment_id	customer_id	staff_id	rental_id	amount	payment_date	last_update
1	1	1	76	2.99	2022-03-15 10:08:09	2022-03-15 10:08:09

99

UPDATE ...

104

- Utilisée pour **modifier** les enregistrements existants dans une table
- La valeur renvoyée sera le nombre de lignes dans la table qui ont été mises à jour conformément à la condition spécifiée.
- Si la clause **WHERE** est omise, **tous** les enregistrements de la table seront mis à jour!



- SYNTAXE :**

```
UPDATE nom_de_la_table
SET colonne1 = value2, colonne2 = value2,
...
WHERE condition;
```

- colonne1, colonne2, ...** sont les noms de champ dont les valeurs seront mises à jour.
- value1, value2, ...** sont les nouvelles valeurs pour les champs
- WHERE** précise quel (s) enregistrement (s) seront mis à jour

- Exemple :**

```
UPDATE rental
SET return_date = '2022-02-28'
WHERE rental_id = '16053';
```

104

UPDATE ...

105

- Il est possible de modifier plusieurs colonnes de la table



- SYNTAXE :**

```
UPDATE nom_de_la_table
SET colonne1 = value2, colonne2 = value2, ...
WHERE condition;
```

- colonne1, colonne2, ...** sont les noms de champ dont les valeurs seront mises à jour.
- value1, value2, ...** sont les nouvelles valeurs pour les champs
- WHERE** précise quel (s) enregistrement (s) seront mis à jour

- Exemple :**

```
UPDATE rental
SET return_date = '2022-02-28', last_update = now()
WHERE rental_id = '16053';
```

105

UPDATE ...

106

Exercice : écrire la requête qui permet d'enregistrer le retour d'un film, le film a été retourné par un client dont le identifiant est 3, le film retourné à l'identifiant d'inventaire 10.

106

UPDATE ...

108

Exercice : écrire la requête qui permet d'enregistrer le retour du film "Academy Dinosaur" de Mike Hillyer aujourd'hui . Le client qui avait loué le film se nomme "Amber Dixon" .

Pour enregistrer le retour d'un film, il faut mettre à jour la table **rental** afin d'actualiser la date de retour.

- Décrivez les étapes
- Implémentez la requête.

108

DELETE ...

113

- Utilisée pour **supprimer** les enregistrements existants dans une table
- Si la clause **WHERE** est omise, **tous** les enregistrements de la table seront supprimés !



- SYNTAXE :**

```
DELETE FROM nom_de_la_table
WHERE condition;
```

- WHERE** précise quel (s) enregistrement (s) seront supprimés

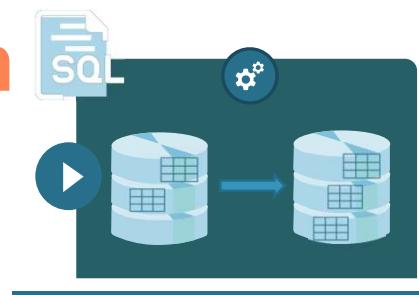
- Exemple :**

```
DELETE FROM CEPAGE
WHERE Couleur_Cepage = 'blanc';
```

113

Modification de la structure

114



114

ALTER TABLE ...

115

- Instruction utilisée pour **ajouter, supprimer ou modifier** des colonnes dans une table existante
- La syntaxe pour **modifier** une colonne peut varier selon le serveur de base de données



- SYNTAXE :

ALTER TABLE nom de La table ADD colonne type_de_données;

```
ALTER TABLE PAYS
ADD NomPays VARCHAR(20);
```

ALTER TABLE nom de La table DROP COLUMN colonne;

```
ALTER TABLE PAYS
DROP COLUMN NomPays;
```

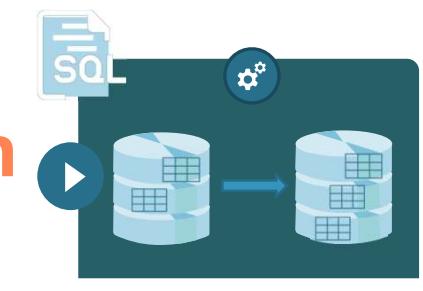
ALTER TABLE nom de La table CHANGE colonne type_de_données;

```
ALTER TABLE PAYS
CHANGE NomPays VARCHAR(10);
```

115

Automatisation

116



116

TRIGGER ...

117

- Est ce une fonction appelée automatiquement chaque fois qu'un événement associé à une table se produit.
- Un événement peut être l'un des suivants : `INSERT, UPDATE, DELETE`
- La différence entre un déclencheur et une fonction définie par l'utilisateur est qu'un déclencheur est automatiquement appelé lorsqu'un événement déclencheur se produit.
- Il faut spécifier si le déclencheur est appelé avant ou après l'événement.
- Permet de faciliter et d'automatiser des actions au sein du SGBD.

• SYNTAXE :

```
CREATE TRIGGER nom_du_trigger
{BEFORE | AFTER} {INSERT | UPDATE| DELETE}
ON nom_table FOR EACH ROW
corps_du_trigger;
```

117

TRIGGER ...

118

Exemple :

- Le trigger (déclencheur) `last_updated` met à jour la colonne `last_update` de la table en question avec l'heure et la date actuelles au fur et à mesure que les lignes sont modifiées.

```
CREATE TRIGGER last_updated
BEFORE UPDATE ON public.film
FOR EACH ROW EXECUTE PROCEDURE public.last_updated();
```

Le trigger a été défini pour la table film

	actor_id [PK] integer	first_name character varying (45)	last_name character varying (45)	last_update timestamp without time zone
1	1	Penelope	Guiness	2013-05-26 14:47:57.62
2	2	Nick	Wahlberg	2013-05-26 14:47:57.62

Dans `postgresql` il faut créer une fonction

118

TRIGGER ...

119

Exemple 2 :

- Le *trigger* (déclencheur) rental_date met à jour la colonne **rental_date** de la table rental avec l'heure et la date actuelles avant qu'une nouvelle location soit enregistrée.

```
CREATE TRIGGER rental_date
BEFORE INSERT ON rental
FOR EACH ROW SET NEW.rental_date = NOW();
```

119

Sécurité des données



120

120

Sécurité des données ...

121

... protéger les données numériques, telles que celles d'une base de données, contre les forces destructrices et contre les actions indésirables d'utilisateurs non autorisés telles que la cyberattaque ou une violation de données.

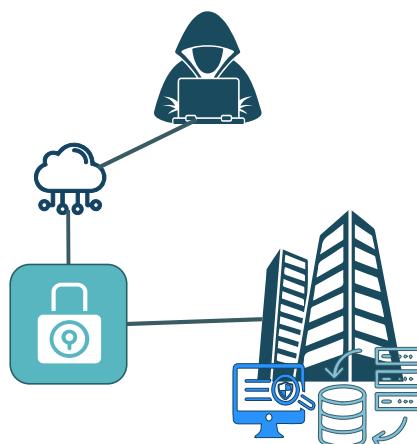
121

Sécurité des données

122

La sécurité d'une application peut être compromise de multiples manières.

- Les applications web sont plus particulièrement vulnérables du fait de leur architecture
- La compromission d'une application peut engendrer la compromission en cascade des sources de données auxquelles elle a accès.

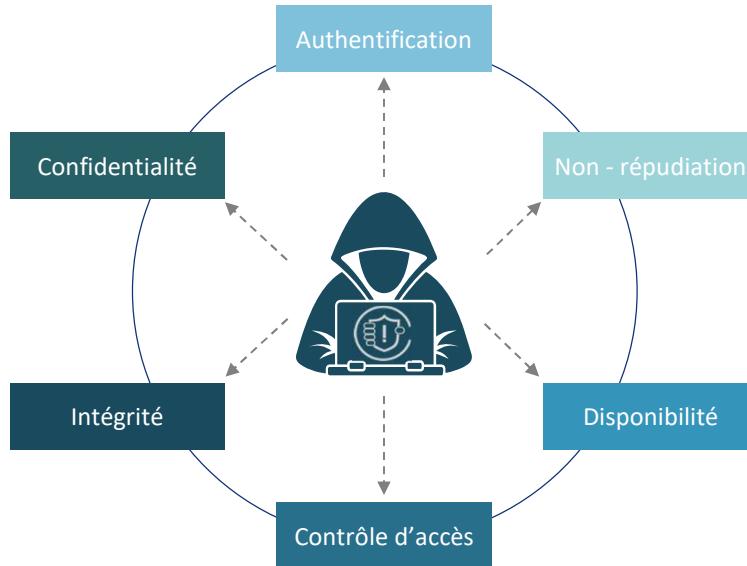


La sécurité de la base des données comprend l'ensemble de mécanismes et dispositions protégeant la base de données contre les effets des menaces accidentnelles ou intentionnelles

122

Facteurs de la sécurité de données....

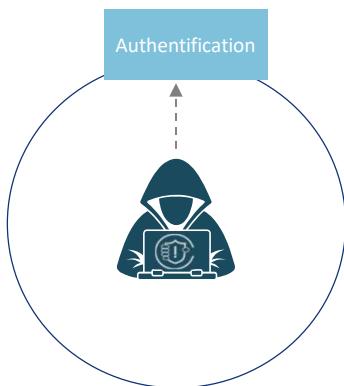
123



123

Facteurs de la sécurité de données....

124

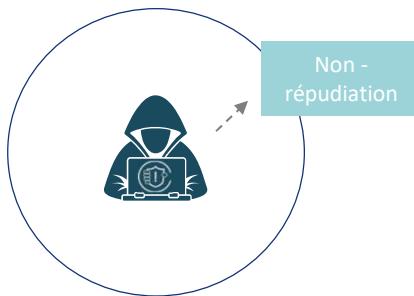


- Processus de vérification de l'identité d'un utilisateur ou d'un système qui demande l'accès à une ressource ou à des données dans une base de données.
- Est utilisée pour s'assurer que seules les personnes ou les systèmes autorisés peuvent accéder aux données stockées dans la base de données.
- Consiste à vérifier que les informations d'identification fournies sont valides.
- Peut être réalisée en utilisant divers mécanismes, tels que **l'authentification par mot de passe**, **l'authentification à deux facteurs**, **l'authentification par certificat**, **l'authentification biométrique**, etc. Dans le contexte des bases de données, l'authentification est généralement réalisée en utilisant des noms d'utilisateur et des mots de passe.

124

Facteurs de la sécurité de données....

125



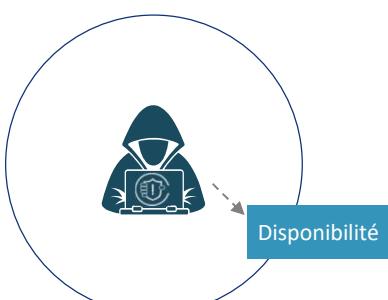
Non -
répudiation

- Garantit qu'une personne ne peut pas nier avoir créé ou modifié une donnée ou une information dans une base de données. Cela signifie que si une action est effectuée dans la base de données, il est impossible pour la personne qui l'a réalisée de prétendre ultérieurement qu'elle n'a pas été impliquée dans cette action.
- La **non-répudiation** est importante dans le contexte de la sécurité des données car elle permet de garantir l'exactitude et l'intégrité des données stockées dans la base de données.
- Par exemple, les **signatures électroniques** peuvent être utilisées pour garantir l'authenticité d'un document ou d'une donnée, tandis que les **journaux d'audit** peuvent être utilisés pour enregistrer toutes les actions effectuées dans la base de données, permettant ainsi de retracer les activités et de prouver qui a effectué quoi.

125

Facteurs de la sécurité de données....

126



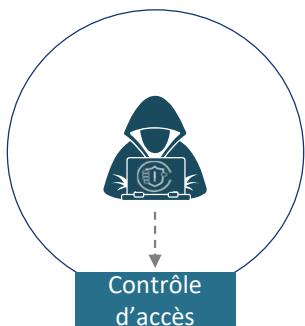
Disponibilité

- Capacité d'un système de base de données à être disponible et fonctionnel pour les utilisateurs autorisés à tout moment, **sans interruption ou perturbation**.
- Les utilisateurs doivent être en mesure d'accéder aux données stockées dans la base de données et de les utiliser de manière efficace et efficiente sans être confrontés à des temps d'arrêt ou à des temps de réponse lents.
- Pour garantir la disponibilité des données dans une base de données, des mesures telles que la mise en place de **serveurs de base de données redondants**, la **sauvegarde régulière des données**, la mise en place de **politiques de sécurité et de maintenance adéquates**, et la **surveillance continue de la performance** du système sont nécessaires.

126

Facteurs de la sécurité de données....

127

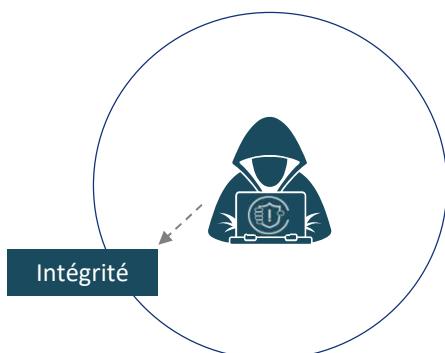


- Garantit que les utilisateurs ne peuvent accéder qu'aux données pour lesquelles ils ont les autorisations appropriées et empêchent les utilisateurs non autorisés d'accéder à ces données.
- Fait référence aux méthodes et outils utilisés pour gérer et restreindre l'accès aux données stockées dans la base de données.
- Le contrôle d'accès peut être mis en œuvre en utilisant divers mécanismes tels que **l'identification et l'authentification** des utilisateurs, la mise en place de **rôles et de privilèges d'accès**, la mise en place de **politiques de sécurité**, la **journalisation** et **l'audit** des activités de l'utilisateur, et **la gestion des droits d'accès**.

127

Facteurs de la sécurité de données....

128



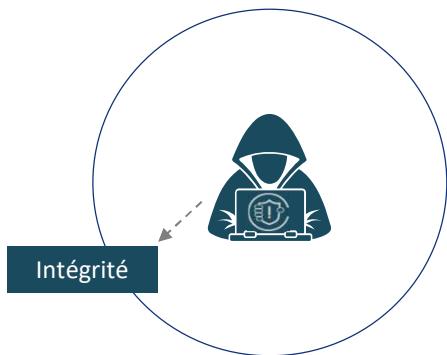
- L'intégrité des données se réfère à la qualité et la fiabilité des données stockées dans la base (ex. précision, la cohérence, la validité et la conformité aux règles et contraintes définies).
- Pour assurer l'intégrité des données, les bases de données utilisent des **contraintes d'intégrité**, qui sont des règles qui garantissent que les données stockées dans une base de données sont cohérentes et conformes aux attentes de l'utilisateur. Les contraintes d'intégrité peuvent inclure des règles pour les clés primaires, les clés étrangères, les valeurs uniques, les règles de validation de données, etc.

128

Facteurs de la sécurité de données....

129

Il existe plusieurs types de contraintes d'intégrité :



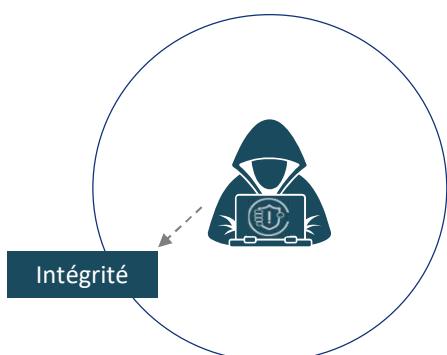
- **Contraintes de clé primaire** : Elles définissent une colonne ou un ensemble de colonnes qui identifient de manière unique chaque enregistrement dans une table. **Aucune valeur en double ou nulle** n'est autorisée dans cette colonne ou cet ensemble de colonnes.
- **Contraintes de clé étrangère** : Elles définissent une relation entre deux tables en spécifiant que **les valeurs d'une colonne dans une table doivent correspondre aux valeurs d'une colonne dans une autre table**. Cela garantit l'intégrité référentielle entre les tables.
- **Contraintes d'unicité** : Elles garantissent qu'une colonne ou un ensemble de colonnes **ne peut contenir aucune valeur en double**. Chaque valeur dans la colonne doit être unique.

129

Facteurs de la sécurité de données....

130

Il existe plusieurs types de contraintes d'intégrité :



- **Contraintes de vérification** : Elles spécifient des conditions qui doivent être remplies pour qu'une valeur soit valide. Par exemple, une contrainte de vérification peut exiger que l'âge d'une personne soit supérieur à zéro.
- **Contraintes de domaine** : Elles définissent les types de données et les plages de valeurs autorisées pour une colonne.

Lorsqu'une contrainte d'intégrité est définie sur une base de données, le système de gestion de base de données (SGBD) vérifie automatiquement que toutes les opérations de modification des données (insertion, mise à jour ou suppression) respectent ces contraintes. Si une opération viole une contrainte, le SGBD la rejette et empêche la modification des données, garantissant ainsi l'intégrité et la cohérence des données stockées dans la base de données.

130

Facteurs de la sécurité de données....

131

Exemple : contraintes d'intégrité table "rental"

Foreign key constraints

Actions	Constraint properties	Column	Foreign key constraint (INNODB)	Database	Table	Column	
Drop	rental_ibfk_1	ON DELETE RESTRICT	ON UPDATE NO ACTION	inventory_id	dvdrental	inventory	inventory_id
Drop	rental_ibfk_2	ON DELETE RESTRICT	ON UPDATE NO ACTION	customer_id	dvdrental	customer	customer_id
Drop	rental_ibfk_3	ON DELETE RESTRICT	ON UPDATE RESTRICT	staff_id	dvdrental	staff	staff_id

Le SGBD empêchera toute suppression de la ligne parente (table référencée) si des lignes enfants (table faisant référence) existent.

si une mise à jour est effectuée sur la valeur de la clé primaire dans la table parente, aucune action n'est entreprise sur les valeurs correspondantes dans la table enfant.

Intégrité

131

Facteurs de la sécurité de données....

132

Exemple : contraintes d'intégrité table "rental"

Foreign key constraints

Actions	Constraint properties	Column	Foreign key constraint (INNODB)	Database	Table	Column	
Drop	rental_ibfk_1	ON DELETE RESTRICT	ON UPDATE RESTRICT	inventory_id	dvdrental	inventory	inventory_id
Drop	rental_ibfk_2	ON DELETE RESTRICT	ON UPDATE RESTRICT	customer_id	dvdrental	customer	customer_id
Drop	rental_ibfk_3	ON DELETE RESTRICT	ON UPDATE RESTRICT	staff_id	dvdrental	staff	staff_id

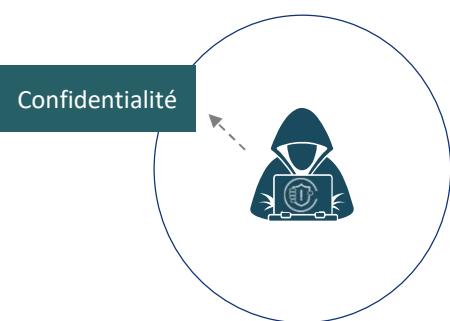
```
ALTER TABLE rental
ADD CONSTRAINT rental_ibfk_2
FOREIGN KEY (customer_id)
REFERENCES customer(customer_id);
```

Intégrité

132

Facteurs de la sécurité de données....

133



- Fait référence à la **protection des données sensibles** contre toute divulgation non autorisée. Cela implique de garantir que les informations stockées dans la base de données ne sont accessibles qu'aux **utilisateurs autorisés et que leur accès est limité à des fins spécifiques**.
- Pour assurer la confidentialité des données dans une base de données, des mesures telles que **l'identification et l'authentification** des utilisateurs, la mise en place de **contrôles d'accès**, la **cryptage des données**, la **gestion des droits d'accès**, la mise en place de politiques de sécurité et la surveillance continue sont nécessaires.

133

Les risques

134

Les risques propres à une source de données sont les suivants :

- Vol de données (perte de **confidentialité**) ;
- Altération de données (perte **d'intégrité**) ;
- Destruction de données (remise en cause de la **continuité d'activité**) ;
- Augmentation du niveau de priviléges d'un utilisateur d'une application pour réaliser des actions malveillantes, telles que l'accès aux données sensibles (**sécurité, espionnage**) ;
- Utilisation abusive des ressources système (**dénie de service**).
- **Divulgation** non autorisée de données



134

La protection des données intègre

135



135

Sécurité de données (composants) ...

136

- Le **contrôle d'accès** permet de vérifier qui a le droit de consulter et d'utiliser les données. Via une **authentification** et une **autorisation**, les stratégies de contrôle d'accès vérifient que les utilisateurs sont bien ceux qu'ils disent être et qu'ils disposent d'un accès adapté aux données.
- La **protection des données** : protection des informations sensibles contre les dommages, la perte ou la corruption.
 - Protection contre les dommages malveillants ou accidentels
 - Restauration rapide des données en cas de dommage ou de perte
- La **surveillance** de l'activité sur les données ainsi que la surveillance des comportements suspects, permettent de découvrir rapidement les failles de sécurité
- La **gestion des utilisateurs** permet de définir la façon dont les autres utilisateurs accèdent à la base de données en leur accordant ou supprimant des priviléges. Chaque utilisateur ne doit avoir accès qu'aux applications et aux données dont il a besoin dans le cadre de ses fonctions.



136

Contrôle d'accès

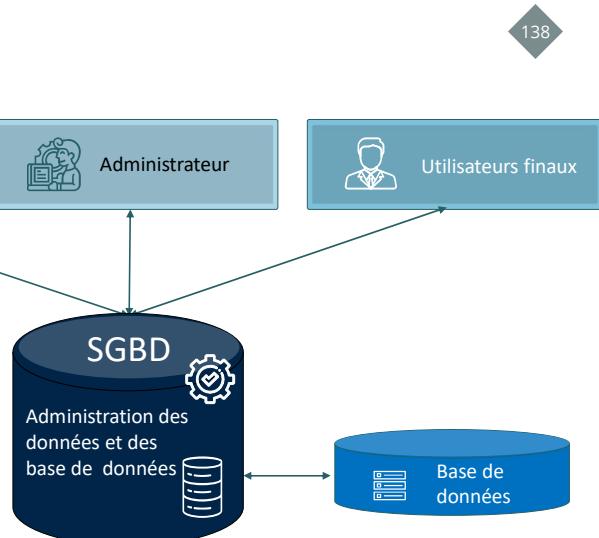


137

137

Gestion des utilisateurs ...

- Ajouter de nouveaux utilisateurs
- Supprimer des utilisateurs
- Gérer l'accès des utilisateurs
- Définir le privilège de connexion utilisateur
- Modifier les autorisations utilisateur
- Afficher les autorisations utilisateur existantes
- Modifier les mots de passe des utilisateurs



138

138

Type d'utilisateur ...

139

Les utilisateurs ayant besoin d'un accès à la base de données, ils peuvent être de différents types :



Programmes
d'application



Administrateur



Utilisateurs finaux

- L'**administrateur** est une personne physique ayant tous les droits sur le SGBD, mais pas forcément sur le contenu des bases de données : il peut réaliser des opérations de gestion des droits d'accès et des ressources système, mais on pourra choisir d'exclure ou non les droits d'accès en lecture et/ou écriture au contenu des bases de données.
- Une **application** est tout programme accédant pour lui-même à la base de données.
- L'**utilisateur** est une personne physique se connectant directement à la base de données en ligne de commande ou via une interface graphique ou utilisant une application qui va se connecter à la base de données sous l'identité de l'utilisateur.

139

Création des utilisateurs ...

140

- **SYNTAXE (MySQL) :**

- Crée de nouveaux comptes.
- La syntaxe peut varier selon le Système de Gestion de Base de Données (SGBD).



```
CREATE USER [IF NOT EXISTS]
    user [auth_option] [, user [auth_option]] ...
    DEFAULT ROLE role [, role ] ...
    [REQUIRE {NONE | tls_option [[AND] tls_option] ...}]
    [WITH resource_option [resource_option] ...]
    [password_option | lock_option] ...
    [COMMENT 'comment_string' | ATTRIBUTE 'json_object']
```

`CREATE USER 'mgonzalez'@'localhost' IDENTIFIED BY 'password';`

140

Suppression des utilisateurs ...

141

- **SYNTAXE (MySQL) :**

- Supprime un ou plusieurs comptes MySQL et leurs priviléges



```
DROP USER [IF EXISTS] user [, user ] ...
```

141

Gestion de privilèges (rôles et permissions)...

142

- Pour réaliser certaines actions (consulter, exécuter, modifier...) sur les objets de la base de données (tables, vues, procédures stockées, fonctions...) l'utilisateur doit avoir les priviléges (aussi appelés droits) nécessaires.
- L'utilisateur peut obtenir ces priviléges de manière directe ou indirecte (via les rôles, permettent de définir de profils).
- Les priviléges peuvent être accordés ou révoqués.
- Les priviléges peuvent être classés dans deux familles :
 - **Privilèges système** - permettent à l'utilisateur de CREATE, ALTER ou DROP des objets de base de données.
 - **Privilèges d'objet** - permettent à l'utilisateur de EXECUTE, SELECT, INSERT, UPDATE ou DELETE des données d'objets de base de données.

En SQL :

- GRANT
- REVOKE

DCL : Data Control Language



142

Octroyer des privilèges ...

143

- Utilisé pour fournir aux utilisateurs des accès ou des privilèges sur les objets de base de données.
- Nom du privilège : ALL, EXECUTE, SELECT, ...

- SYNTAXE (MySQL) :

```
GRANT nom du privilege [, nom du
privilege]
ON nom de l'objet
TO user_ou_role, [user_ou_role]
[WITH GRANT OPTION];
```

Example:

GRANT SELECT ON film TO user1;

table

Cette commande accorde une autorisation pour l'utilisation du SELECT sur la table film à user1.

143

Octroyer des privilèges ...

144

- Il est possible d'accorder des privilèges valables seulement à partir d'un poste particulier

```
GRANT SELECT ON dvdrental.film TO 'mgontier'@'192.168.10.27'
```

L'utilisateur mgontier pourra consulter la table "film" à partir du poste dont l'adresse IP est 192.168.10.27

144

Octroyer des privilèges ...

145

- Il est possible de consulter les privilèges accordés sur un objet de la base de données

```
SELECT grantee, privilege_type
FROM information_schema.role_table_grants
WHERE table_name='film'
```

Cette commande fonctionne pour Postgresql.

	grantee name	privilege_type character varying
1	adm_ticket	INSERT
2	adm_ticket	SELECT
3	adm_ticket	UPDATE
4	adm_ticket	DELETE
5	adm_ticket	TRUNCATE
6	adm_ticket	REFERENCES
7	adm_ticket	TRIGGER

145

Exercice ...

146

- Créer un nouvel utilisateur (dvd_user)
- Accorder le privilège `SELECT` sur la table `film`
- Lister les privilèges sur la table `film`

146

Corrigé ...

147

- Créer un nouvel utilisateur (dvd_user)
- Accorder le privilège **SELECT** sur la table **film**
- Lister les privilèges sur la table **film**

	grantee name	privilege_type character varying
1	adm_ticket	INSERT
2	adm_ticket	SELECT
3	adm_ticket	UPDATE
4	adm_ticket	DELETE
5	adm_ticket	TRUNCATE
6	adm_ticket	REFERENCES
7	adm_ticket	TRIGGER
8	dvd_user	SELECT

147

Révoquer des privilèges ...

148

- SYNTAXE (MySQL) :

- Supprime les droits ou privilèges d'accès utilisateur aux objets de base de données.
- Nom du privilège : **ALL**, **EXECUTE**, **SELECT**, ...

`REVOKE nom du privilege [, nom du privilege]
ON nom de l'objet
FROM user_ou_role, [user_ou_role]`

Example:

`REVOKE SELECT ON film FROM user1;`

Cette commande révoque l'autorisation pour l'utilisation du **SELECT** sur la table **film** à **user1**.

148

Révoquer des privilèges ...

149

- Révoquer le droit de **SELECT** à l'utilisateur **adm_ticket**
- Quel sera le résultat de la requête ?
SELECT * FROM film

149

Révoquer des privilèges ...

150

Corrigé :

REVOKE SELECT ON film FROM adm_ticket

REVOKE

Requête exécutée avec succès en 111 msec.

SELECT * FROM film

ERREUR : ERREUR: droit refusé pour la table film

État SQL : 42501

150

Contrôle d'accès via une application



151

151

Protection des identifiants ...

- Quelques approches couramment utilisées :
 - **Utilisation de variables d'environnement** : Stockez les identifiants d'authentification dans des variables d'environnement plutôt que dans le code source directement. L'application peut ensuite accéder à ces variables d'environnement pendant son exécution.

```
<?php
// Récupérer les informations d'authentification à partir des variables d'environnement
$db_user = getenv('DB_USER');
$db_password = getenv('DB_PASSWORD');
$db_name = getenv('DB_NAME');
$db_host = getenv('DB_HOST');

try
{
    // Établir la connexion à la base de données
    $pdo = new PDO("mysql:host=$db_host;dbname=$db_name", $db_user, $db_password);
```

152

152

Protection des identifiants ...

153

1. Sur Apache, ces variables peuvent être définies dans un fichier .env

```
DB_USER=votre_utilisateur  
DB_PASSWORD=votre_mot_de_passe  
DB_NAME=votre_base_de_donnees  
DB_HOST=votre_hote
```

2. Configurer Apache : s'assurer que la configuration Apache permet la lecture des fichiers d'environnement. La configuration se fait en utilisant le fichier de configuration Apache (*httpd.conf*),

153

Protection des identifiants ...

154

- Voici quelques approches couramment utilisées :
 - **Utilisation de certificats et d'authentification basée sur des clés** : Si possible, utilisez des méthodes d'authentification plus avancées, telles que l'authentification basée sur des clés, en utilisant des certificats ou d'autres mécanismes de sécurité.

154

Protection des identifiants ...

155

- Voici quelques approches couramment utilisées :
 - Utilisation de variables d'environnement : Stockez les identifiants d'authentification dans des variables d'environnement plutôt que dans le code source directement. Votre application peut ensuite accéder à ces variables d'environnement pendant son exécution.
 - Services de gestion des secrets : Utilisez des services spécialisés tels que AWS Secrets Manager, HashiCorp Vault ou Azure Key Vault pour stocker et gérer vos secrets. Votre application récupérera les informations d'authentification au moment de l'exécution à partir de ces services.
 - Fichiers de configuration sécurisés : Stockez les informations d'authentification dans des fichiers de configuration spécifiques, mais assurez-vous que ces fichiers ne sont pas accessibles publiquement. Utilisez des règles de sécurité pour restreindre l'accès aux fichiers de configuration contenant des informations sensibles.
 - Protégez l'accès au code source : Limitez l'accès au code source aux seules personnes autorisées. Utilisez des outils de gestion de code source avec des

155

Protection des données



156

156

Protection des données ...

157

Les Views (vues)

- Une "view" (ou "vue" en français) est une **représentation virtuelle** d'une partie ou de la totalité d'une ou plusieurs tables d'une base de données.
- Elle est créée en utilisant une requête SQL et agit comme une table virtuelle qui affiche une sélection de données provenant des tables de la base de données sous une forme spécifique. Les vues n'ont pas de données physiques associées, mais plutôt une définition de la requête utilisée la créer.
- Les vues peuvent être utilisées pour **restreindre** l'accès aux données sensibles, pour regrouper les données d'une manière logique ou pour simplifier les requêtes complexes.

157

Protection des données ...

158

Les Views (exemple) : cette vue permet d'obtenir les informations des clients qui ont des films en retard à la date d'aujourd'hui

```

CREATE VIEW en_retard AS
SELECT CONCAT(customer.last_name, ' ', customer.first_name) AS customer,
customer.email, film.title, DATE_ADD(rental_date, INTERVAL rental_duration DAY)
AS date_prevue
FROM rental INNER JOIN customer ON rental.customer_id = customer.customer_id
INNER JOIN inventory ON rental.inventory_id = inventory.inventory_id
INNER JOIN film ON inventory.film_id = film.film_id
WHERE rental.return_date IS NULL
AND DATE_ADD(rental_date, INTERVAL film.rental_duration DAY) < CURRENT_DATE
ORDER BY title

```

158

Protection des données ...

159

Les Views (exemple) : cette vue permet d'obtenir les informations des clients qui ont des films en retard.

Une fois la vue créée, elle est ajoutée à la base de données.



Ensuite, il sera possible de créer un utilisateur et de lui accorder des droits sur la vue. Seulement de droits de lecture son accordés.

```
CREATE USER 'mgontier'@'192.168.10.27' IDENTIFIED BY 'password@123Y';
GRANT SELECT ON dvdrental.en_retard TO 'mgontier'@'192.168.10.27'
```

159

Protection des données ...

160

Les Views (exemple) : la vue peut ensuite être consultée comme une autre table

```
SELECT * FROM `en_retard`
```

L'utilisateur pourra accéder cette table selon les permissions accordées.

En général, les vues sont généralement considérées comme des objets de base de données en lecture seule.

customer	email	title	date_prevue
OLVERA, DWAYNE	DWAYNE.OLVERA@sakilacustomer.org	ACADEMY DINOSAUR	2005-08-27 00:30:32
SMITH, MARY	MARY.SMITH@sakilacustomer.org	ACADEMY DINOSAUR	2022-04-27 17:17:02
HUEY, BRANDON	BRANDON.HUEY@sakilacustomer.org	ACE GOLDFINGER	2006-02-17 15:16:03
OWENS, CARMEN	CARMEN.OWENS@sakilacustomer.org	AFFAIR PREJUDICE	2006-02-19 15:16:03
HANNON, SETH	SETH.HANNON@sakilacustomer.org	AFRICAN EGG	2006-02-20 15:16:03
COLE, TRACY	TRACY COLE@sakilacustomer.org	ALI FOREVER	2006-02-18 15:16:03
DEAN, MARCIA	MARCIA.DEAN@sakilacustomer.org	ALONE TRIP	2006-02-17 15:16:03
VINES, CECIL	CECIL.VINES@sakilacustomer.org	AMADEUS HOLY	2006-02-20 15:16:03
TURNER, MARIE	MARIE.TURNER@sakilacustomer.org	AMERICAN CIRCUS	2006-02-17 15:16:03
GILLILAND, JOE	JOE.GILLILAND@sakilacustomer.org	AMISTAD MIDSUMMER	2006-02-20 15:16:03
BAUGH, EDWARD	EDWARD.BAUGH@sakilacustomer.org	ARMAGEDDON LOST	2006-02-19 15:16:03
FRANKLIN, BETH	BETH.FRANKLIN@sakilacustomer.org	BAKED CLEOPATRA	2006-02-17 15:16:03
WILLIAMSON, GINA	GINA.WILLIAMSON@sakilacustomer.org	BANG KWAI	2006-02-19 15:16:03
ELLINGTON, MELVIN	MELVIN.ELLINGTON@sakilacustomer.org	BASIC EASY	2006-02-18 15:16:03
GREGORY, SONIA	SONIA.GREGORY@sakilacustomer.org	BERETS AGENT	2006-02-19 15:16:03
WOODS, FLORENCE	FLORENCE.WOODS@sakilacustomer.org	BLADE POLISH	2006-02-19 15:16:03
WREN, TYLER	TYLER.WREN@sakilacustomer.org	BLANKET BEVERLY	2006-02-21 15:16:03

160

Protection de la base de données ...

161

1

Héberger les données dans des environnements sécurisés, climatisé et monitorés

2

Duplicer et conserver de manière sécurisée les données afin d'assurer leur disponibilité

3

Répondre aux demandes de récupération en cas d'incident

Garantir la pérennité des données

161

Sauvegarde de la base de données ...

162

La fréquence de vos sauvegardes dépend du type de données à traiter.

En règle générale, le délai entre deux sauvegardes ne doit pas excéder le temps nécessaire pour refaire le travail éventuellement perdu.

Les sauvegardes doivent être fréquentes :

- sauvegardes incrémentales quotidiennes
- sauvegardes complètes à intervalles réguliers

02

À quelle fréquence?

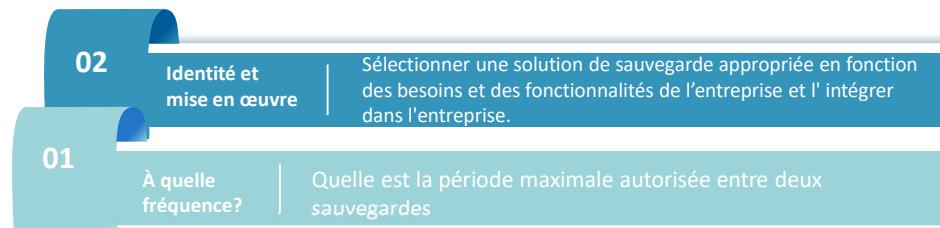
Quelle est la période maximale autorisée entre deux sauvegardes

162

Sauvegarde de la base de données ...

163

- Supports de sauvegarde stockés dans un endroit sécurisé et extérieur à l'entreprise (si possible dans des coffres ignifugés et étanches)
- Sous-traiter, par exemple, une prestation d'externalisation des sauvegardes
- Prévoir une redondance matérielle des dispositifs de stockage
- Définir une procédure de rétablissement adéquate.



163

Sauvegarde de la base de données ...

164

- Surveiller que les sauvegardes sont effectuées correctement,
 - Protéger les données sauvegardées au même niveau de sécurité que celles stockées sur les serveurs d'exploitation (par exemple en chiffrant les sauvegardes).
 - Encadrer la sous-traitance externe
 - Lorsque les sauvegardes sont transmises par le réseau, il convient de chiffrer le canal de transmission si celui-ci n'est pas interne à l'entreprise.
- Définir une procédure de rétablissement adéquate.



164

Stratégie de sauvegarde ...

165

Règle 3 - 2 - 1:



au moins
trois copies
des données



stocker les
copies sur
deux
supports
différents



conserver
une copie de
la
sauvegarde
hors site

165

Stratégie de sauvegarde ...

166

Automatisation

- Planification de tâches : Utilisez le planificateur de tâches intégré au système d'exploitation pour exécuter un script de sauvegarde à intervalles réguliers.
 - Planificateur de tâches de Windows : pour exécuter un fichier batch contenant les commandes SQL nécessaires pour sauvegarder la base de données.
- Outils de sauvegarde intégrés : Certains SGBD fournissent des outils intégrés pour automatiser les sauvegardes (exemple : *mysqldump*, *pg_dump*)
- Outils tiers : Il existe également des outils tiers
 - SQL Server Management Studio (*Microsoft SQL Server*)
 - phpMyAdmin (*MySQL*)
- Scripts personnalisés : Il est possible de créer des scripts personnalisés dans un langage de programmation tel que Python, PowerShell ou Bash

166

Sauvegarde ...

167

Exemple



Sauvegarder une base de données via SQL

```
mysqldump -u [utilisateur] -p [mot_de_passe] [nom_de_la_base_de_données] > sauvegarde.sql
```

l'utilitaire mysqldump qui peut être utilisé pour sauvegarder une base de données MySQL

Fichier de sauvegarde qui sera générée (appelé « fichier dump »),

167

Sauvegarde ...

168

Exemple

Sauvegarder une base de données MySQL via PHPMyAdmin

1. Sélectionnez la base de données à sauvegarder (menu de gauche)
2. Cliquez sur l'onglet "Exporter" situé en haut de PHPMyAdmin
3. Cochez la case "Transmettre" puis cliquez sur "Exécuter"

La capture d'écran montre la interface de PHPMyAdmin avec la base de données 'dvrent' sélectionnée. À gauche, une liste arborescente des tables ('dvrent') est visible, dont les noms sont classés par préfixe ('actor', 'address', etc.). À droite, une liste détaillée des tables ('classmodels') est affichée avec des colonnes pour 'Table', 'Action', 'Rows', 'Type', 'Collation', 'Size' et 'Overhead'. L'onglet 'Export' est actif et encadré en rouge. En bas de l'écran, une barre d'outils Windows et une barre de tâches sont visibles.

168

Sauvegarde ...

Exemple

Sauvegarder une base de données via PgAdmin

1. Sélectionnez la base de données à sauvegarder (menu de gauche)
2. Clique droit sur la base de données
3. Sélectionnez l'option "Sauvegarder..."

The screenshot shows the pgAdmin 4 interface. On the left, the 'Servers' tree view shows three servers: PostgreSQL 13, PostgreSQL 14, and localhost_demo. Under localhost_demo, there are three databases: 'Bases de données (3)', 'Cat', and 'dvd_rental'. The 'dvd_rental' database is selected and has a context menu open over it. The menu items include: Actualiser..., Créer, Supprimer, CREATE Script, Désconnecter la base de données..., Générer le diagramme entité-association, Maintenance, Sauvegarder..., Restaurer..., Assistant de gestion des droits..., Recherche d'objets..., Éditeur de requêtes, Outil PSQL, Propriétés..., pg_execute_server_program, pg_read_all_data, and pg_monitor. The 'Sauvegarder...' option is highlighted with a red box.

169

Sauvegarde ...

Exemple

Sauvegarder une base de données via SQL

PostgreSQL



```
pg_dump base_de_donnees > fichier_sauvegarde
```

Est un programme client PostgreSQL classique, il doit avoir un accès en lecture à toutes les tables que à sauvegarder

Fichier texte de commandes SQL (appelé « fichier dump »), qui, si on le renvoie au serveur, recrée une base de données identique à celle sauvegardée.

170