

INTRODUCTION

SQL vs Algèbre Relationnelle

INTRODUCTION

SQL (Structured Query Language) est un langage informatique normalisé qui permet d'interagir avec des bases de données relationnelles et de les exploitées à travers des requêtes;

Quand on souhaite interroger une base de données, on utilise des commandes sous forme de requêtes (récupération, modification, ajout, suppression ...).

INTRODUCTION

Les requêtes SQL sont classifiées selon les langages (sous-langage SQL) suivants :

✓ Langage de Définition de Données LDD (en anglais Data Definition Language DDL) permettant de définir les schémas des BD;

✓ Langage de Manipulation de Données LMD (en anglais Data Manipulation Language DML) permettant de mettre à jour les données d'une BD (Ajout, Modification, Suppression);

✓ Langage d'Interrogation des Données LID (en anglais Data Query Language DQL) permettant d'exprimer toutes les questions que l'on peut poser à une base de données;

✓ Langage de Contrôle de Données LCD (en anglais Data Control Language DCL) permettant de gérer les utilisateurs (création des utilisateurs et affectation de leurs droits) d'une BD et de contrôler l'exécution des transactions.

SQL VS ALGÈBRE RELATIONNELLE

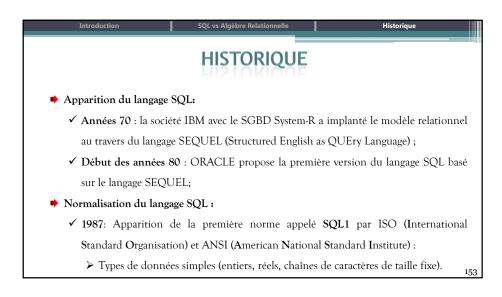
→ L'algèbre relationnelle est un langage formelle (théorique) de manipulation des données qui exprime une succession d'opérations effectuées sur les relations;

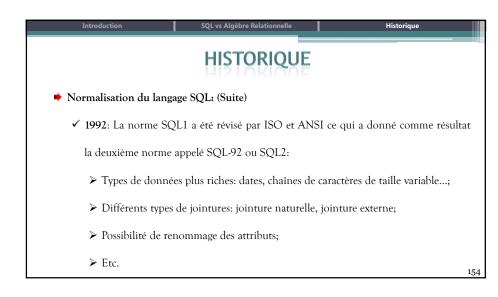
→ L'algèbre relationnelle ne peut pas être compris directement par les SGBDR;

→ SQL est un langage pratique (compris et exécutée directement par les SGBDR) fondé sur l'algèbre relationnelle.

SQL vs Algèbre Relationnelle

38





HISTORIQUE

→ Normalisation du langage SQL: (Suite)

✓ Depuis 1999 : introduction de la norme SQL3 par ISO et ANSI qui comporte de nombreuses fonctionnalités:

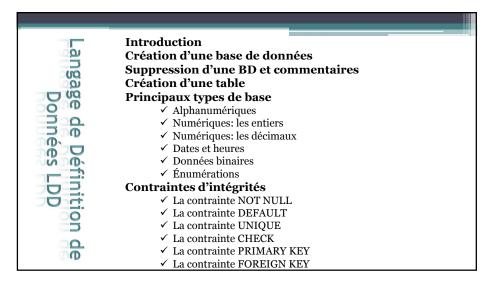
→ Concepts le l'orienté objets;

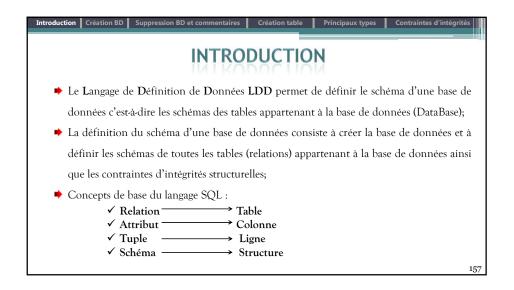
→ Entrepôts de données;

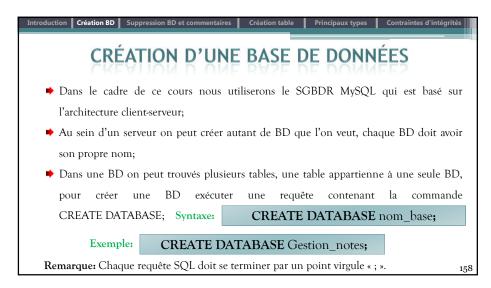
→ Accès à des sources non SQL;

→ Réplication des données;

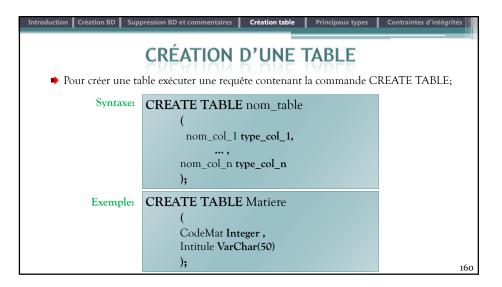
→ etc.

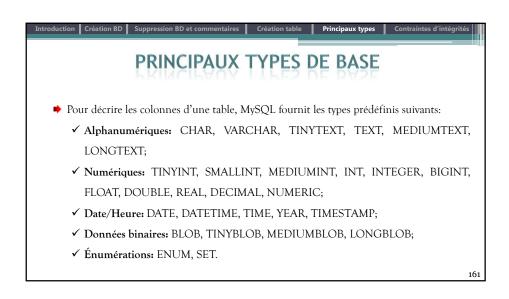










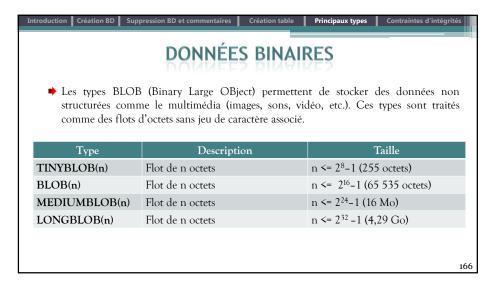


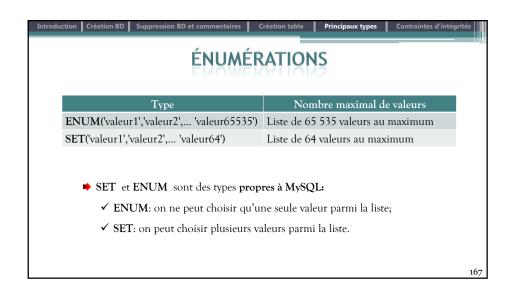
Introduction Création BD Su	ppression BD et commentaires Création table P	rincipaux types Contraintes d'intégrités		
ALPHANUMÉRIQUES				
Type	Description	Longueur Maximale		
CHAR(n)	Chaîne fixe de n caractères (n octets)	n <=28-1 (255 octets)		
VARCHAR(n)	Chaîne variable de n caractères (n octet	s) $n \le 2^{16}-1 (65 535 \text{ octets})$		
TINYTEXT(n)	Chaîne fixe de n caractères (n octets)	$n \le 2^8 - 1$ (255 octets)		
TEXT(n)	Chaîne fixe de n caractères (n octets)	n <= 2 ¹⁶ -1 (65 535 octets)		
MEDIUMTEXT(n)	Chaîne fixe de n caractères (n octets)	n <= 2 ²⁴ -1 (16 Mo)		
LONGTEXT(n)	Chaîne fixe de n caractères (n octets)	$n \le 2^{32} - 1$ (4,29 Go)		
→ Pour les chaînes fixe les valeurs sont stockées en ajoutant des blancs si sa taille est inférieure à n. Ces blancs ne seront pas considérés après extraction à partir de la table;				
▶ Pour les chaînes variable les valeurs sont stockées sans l'ajout de blancs, on stocke la				
chaîne plus la longueur de la chaîne stockées.				

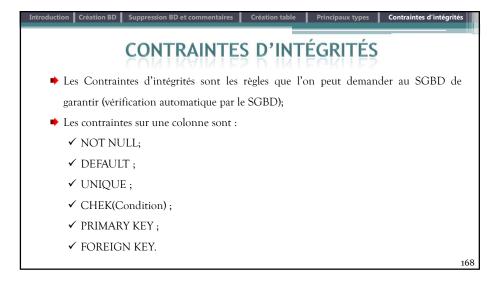
Introduction	res Création table Principaux types Contraintes d'intégrités		
NUMÉRIQUES: LES ENTIERS			
Type	Valeurs minimales et maximales		
INTEGER[(n)] [UNSIGNED] [ZEROFILL]	Entier (sur 4 octets): signé de -2^{31} à 2^{31} -1, non signé de 0 à 2^{32} -1.		
TINYINT [(n)] [UNSIGNED] [ZEROFILL]	Entier (sur 10ctet): signé de -2^7 à 2^7-1 , non signé de 0 à 2^8-1 .		
SMALLINT[(n)] [UNSIGNED] [ZEROFILL]	Entier (sur 2 octets): signé de -2^{15} à $2^{15}-1$, non signé de 0 à $2^{16}-1$.		
MEDIUMINT[(n)] [UNSIGNED] [ZEROFILL]	Entier (sur 3 octets): signé de -2^{23} à $2^{23}-1$, non signé de 0 à $2^{24}-1$.		
BIGINT[(n)] [UNSIGNED] [ZEROFILL]	Entier (sur 8 octets): signé de -2^{63} à 2^{63} –1, non signé de 0 à 2^{64} –1.		
 ▶ La directive UNSIGNED permet de considérer seulement des valeurs positives; ▶ La directive ZEROFILL complète par des zéros à gauche une valeur (par exemple : soit un INTEGER(3) contenant valeur 4, si ZEROFILL est appliqué, la valeur extraite sera «004»). 16 			

Introduction Création BD Suppression BD et commentaires Création table Principaux types Contraintes d'intégrités NUMÉRIQUES: LES DÉCIMAUX			
Type	Valeurs minimales et maximales		
FLOAT[(n[,p])] [UNSIGNED] [ZEROFILL]	Flottant en précision simple: signé de $-3.4\ 10^{+38}$ à $-1.1\ 10^{-38}$, 0. non signé de $1.1\ 10^{-38}$ à $3.4\ 10^{+38}$.		
DOUBLE[(n[,p])] [UNSIGNED][ZEROFILL]	Flottant en précision double: signé de $-1.7\ 10^{+308}$ à $-2.2\ 10^{-308}$, 0 . non signé de $2.2\ 10^{-308}$ à $1.7\ 10^{+308}$.		
DECIMAL(n,p) [UNSIGNED] [ZEROFILL]	Décimal à virgule fixe (par défaut n vaut 10, p vaut 0). Stocké sous frome de chaîne: chaque chiffre, le signe «-», la virgule «,» occupe un caractère (octet), mais le signe «-» et la virgule «,» ne sont pas comptés dans n .		
▶ Dans toute instruction SQL, écrivez l	a virgule avec un point (7/2 retourne 3.5).		



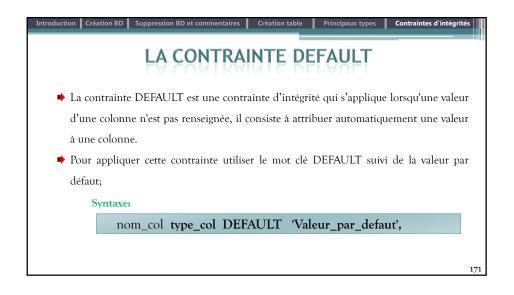


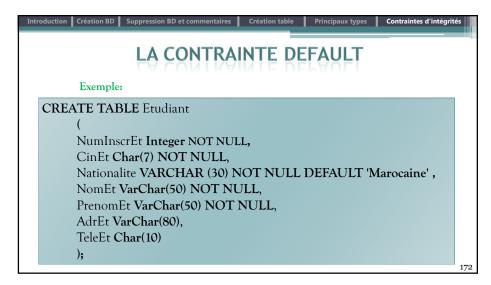


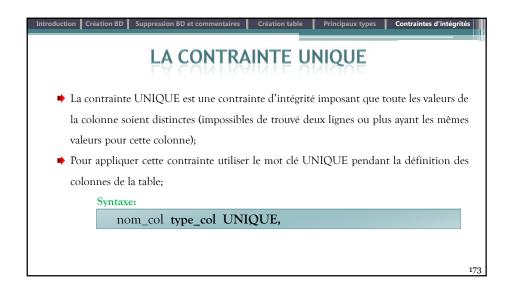


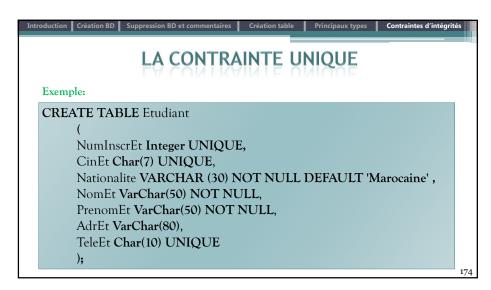












LA CONTRAINTE CHECK

La norme SQL2 comprend une contrainte d'intégrité CHECK (condition) imposant qu'une colonne d'une table doit comporter des valeurs vérifiant la condition;

Pour appliquer cette contrainte utiliser le mot clé CHECK puis exprimer la condition entre parenthèse pendant la définition des colonnes de la table, ou créer une contrainte avec le mot clé CONSTRAINT et exprimer la condition avec CHECK;

Syntaxes:

CHECK (condition sur une colonne)

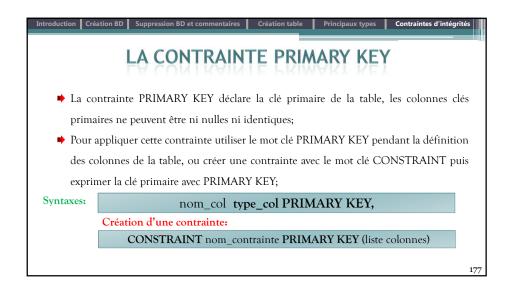
Création d'une contrainte:

CONSTRAINT nom_contrainte CHECK (condition sur une colonne)

Pour MySQL Cette contrainte est prise en charge au niveau de la déclaration mais n'est pas encore opérationnelle.

175





```
Exemple:

Création BD Suppression BD et commentaires Création table Principaux types Contraintes d'intégrités

LA CONTRAINTE PRIMARY KEY

Exemple:

CREATE TABLE Etudiant

(
NumInscrEt Integer PRIMARY KEY,
CinEt Char(7) UNIQUE NOT NULL,
Nationalite VARCHAR (30) NOT NULL DEFAULT 'Marocaine',
NomEt VarChar(50) NOT NULL,
PrenomEt VarChar(50) NOT NULL,
AdrEt VarChar(80),
TeleEt Char(10) UNIQUE
);
```



