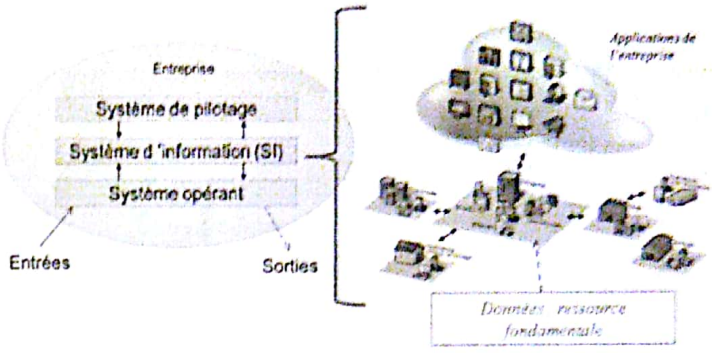
Module SGBD2

(M2106)

Cours 1

Contexte : le Système d'Information

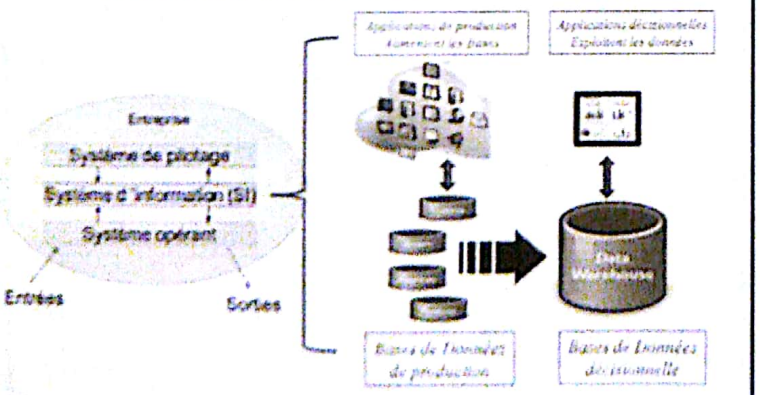


Big data / Data warehouse

Données = ressource fondamental

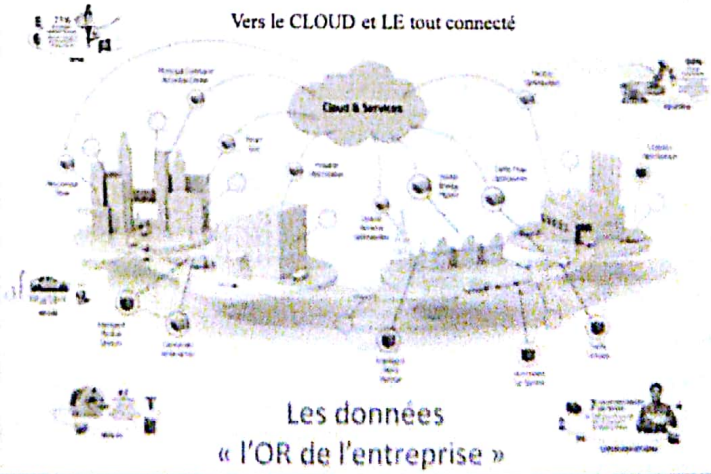
Système d'Information (SI)

Base de données de production versus Bases de données décisionnelles



Evolution du SI

Vers le CLOUD et LE tout connecté



Les données
« l'OR de l'entreprise »

Tendance forte au Cloud
système externalisé

L'importance de la modélisation des données ?

Pourquoi ne pas tout regrouper tout dans une seule table ?

Commandes des clients

N°Commande	DateCommande	NomClient	PortableClient	CodePostal
1	23/01/2017	TARDIF	0634123124	75120
2	24/01/2017	RAOLIA	0712562399	92300
3	30/01/2017	RAOLIA	0612562399	92300

Ajout/ Modification d'une commande

Redondance des données

Erreur de données

L'importance de la modélisation des données ?

N°Commande	DateCommande	NomClient	PortableClient	CodePostal
1	23/01/2017	TARDIF	0634123124	75120
2	24/01/2017	RAOLIA	0712562399	92300
3	30/01/2017	RAOLIA	0612562399	92300

Référentiel des commandes

N°Commande	DateCommande	IdClient
1	23/01/2017	1
2	24/01/2017	2
3	30/01/2017	2

Référentiel des clients

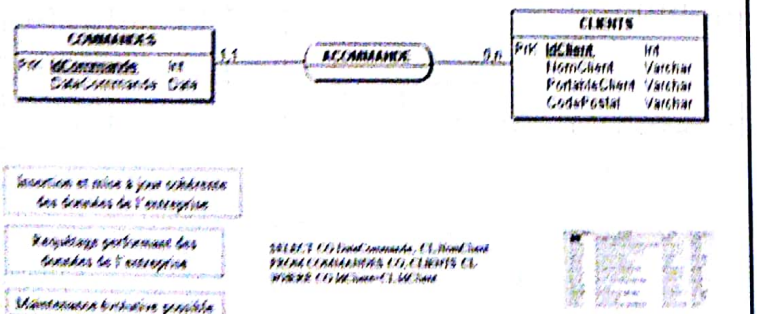
IdClient	NomClient	PortableClient	CodePostal
1	TARDIF	0634123124	75120
2	RAOLIA	0712562399	92300

Ajout de commandes Sans redondances

Appel au référentiel

Modification/Ajout Cohérents

Un modèle normalisé



L'entreprise qui modélise et range ses données de manière structurée et cohérente peut les exploiter dans ses applications

Etapes de passage d'un MCD ->BD ?

Merise

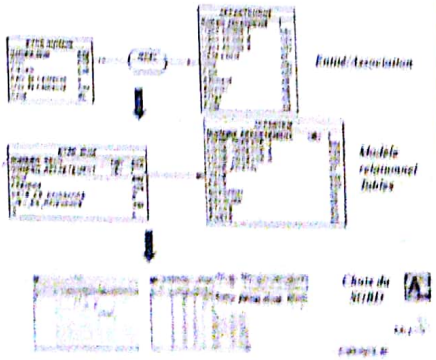
MCD
Modèle Conceptuel de Données

Règles de transformation

MLD
Modèle Logique de Données

Choix d'implantation

MPD
Modèle Physique de Données



Un MCD doit être Cohérent vis-à-vis des données clientes à stocker?

MCD

Les données clientes pourront-elles être stockées dans la base ainsi modélisée ?

Les Ordinateurs

IdOrdi	NomOrdi
100	ZEUS
200	ATHENA

Les données Clientes

Les Personnels

IdPerso	NomPerso
1	DUPONT
2	DURANT

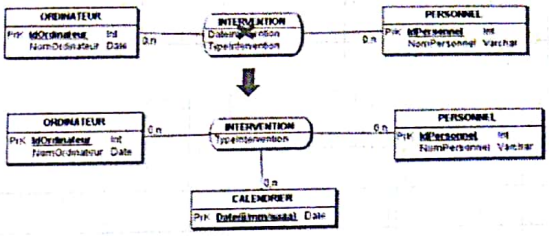
Les Interventions

IdPerso	IdOrdi	DateInterv	TypeInterv
1	100	23/01/18	Installation
1	100	24/01/18	Mise à Jour

Clé primaire (IdPerso, IdOrdi)

Un MCD doit être Cohérent vis-à-vis des données clientes à stocker?

MCD



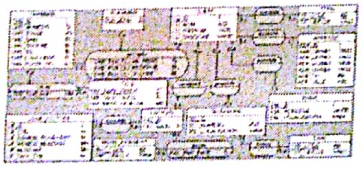
MLD

Les Interventions

IdPerso	IdOrdi	DateInterv	TypeInterv
1	100	23/01/18	Installation
1	100	24/01/18	Mise à Jour

Clé primaire (IdPerso, IdOrdi, DateInterv)

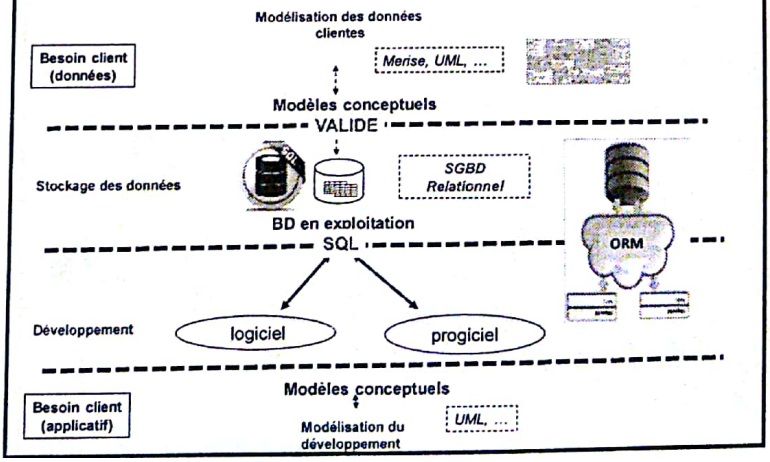
Modèle Conceptuel des Données (MCD)



Un MCD permet de structurer de manière cohérente et non redondante les données de l'entreprise

Outils informatiques : Atelier de Génie Logiciel (AGL) pour MCD ou DC

Etapes d'un projet SI standard



Jérôme FESSY

BD versus SGBD ?

✓ Bases de Données : **ensemble cohérent de données**

✓ SGBD : Système de Gestion de Bases de Données

Logiciel qui permet de gérer les BD en assurant:

- Indépendance Physique
- Indépendance Logique
- Manipulation des données par des non-informaticiens
- Efficacité des accès aux données
- Administration cohérente des données
- Partageabilité des données
- sécurité des données

Jérôme FESSY

Evolution du SI : Générations de SGBD

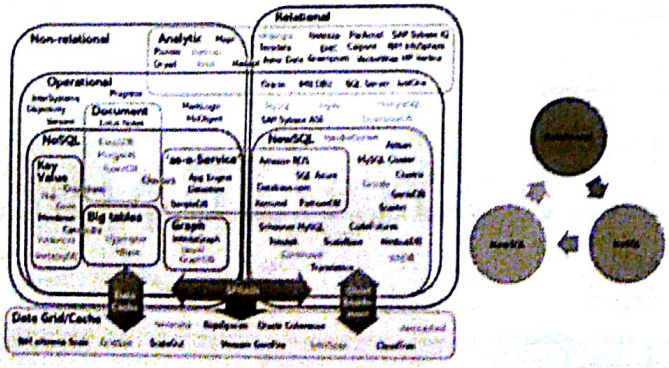
BD centralisées

- ❑ Années 70 : SGBD hiérarchique et réseau
  - LDD / LMD : CODASYL / COBOL
- ❑ Années 80 : SGBD relationnels ⇨ Oracle, Access, Mysql, ...
  - LDD/LMD : SQL
- ❑ Années 90 : SGBD objets ⇨ Ontos, ObjectStore, O2, ...
  - LDD/LMD : SQL3 / OQL

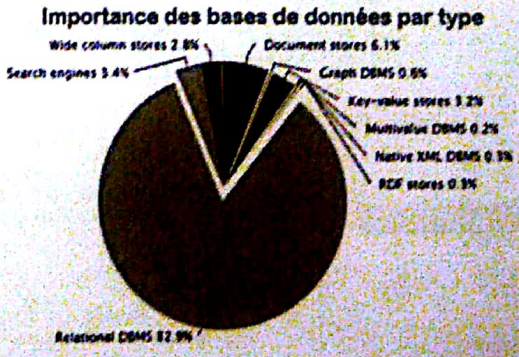
BD distribuées

- ❑ Années 2010 : SGBD NOSQL ⇨ Cassandra, Oracle NoSQL, MongoDB, ...
  - LDD/LMD : LP / LP (Json, Java, ...)

Evolution du SI : les SGBD



Parts de marché des SGBD



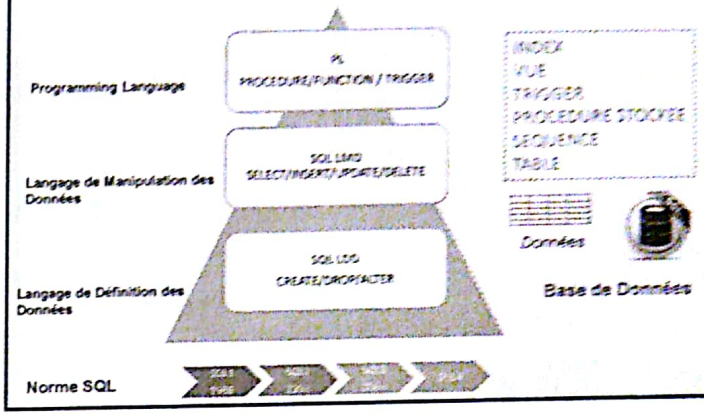
Parts de marché des SGBD

Classement des bases de données
tous types confondus

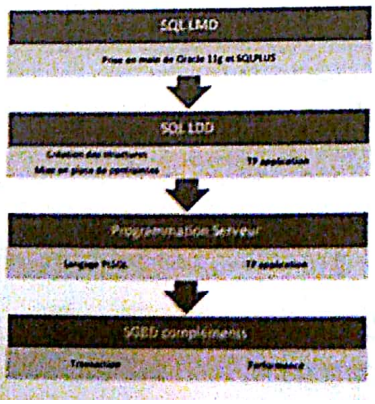
#	DBMS	Database Model
1	Oracle	Relational DBMS
2	MySQL	Relational DBMS
3	Microsoft SQL Server	Relational DBMS
4	MongoDB	Document store
5	PostgreSQL	Relational DBMS
6	DB2	Relational DBMS
7	Microsoft Access	Relational DBMS
8	Cassandra	Wide column store
9	SQLite	Relational DBMS
10	Redis	Key value store

Jérôme FESSY

SQL : Un Langage standardisé d'accès aux données



Le module SGBD2 (M2106)?



Application à
ORACLE 11G

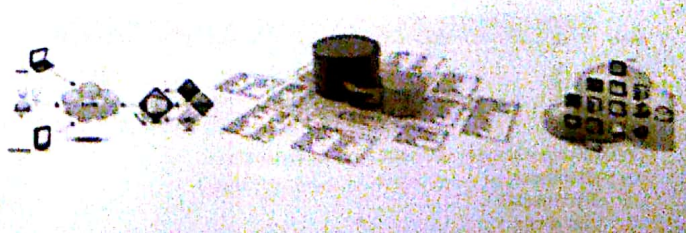
Objectif

Maîtriser la mise en exploitation des données d'une entreprise

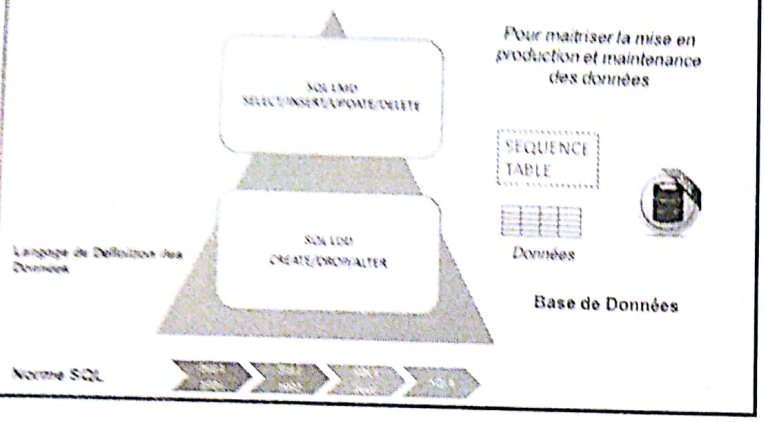
Jérôme FESSY

Partie 1

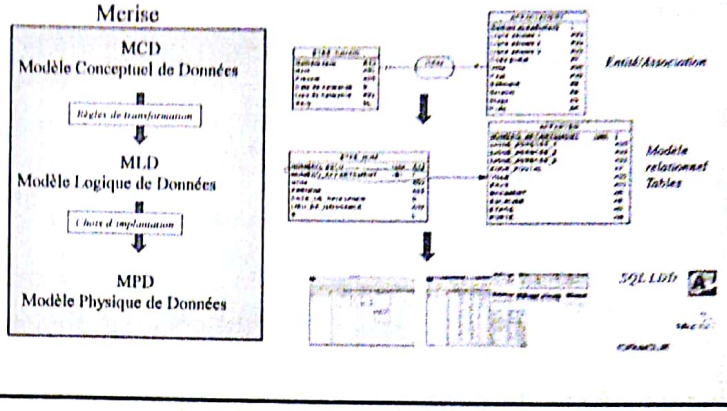
SQL LDD



Objectif : maîtriser le langage SQL LDD



Implantation d'une modélisation



Rappel : le modèle relationnel

Concepts inventé par E.F CODD en 1979. Notion ensembliste

- Une base de données est constituée de tables à deux dimensions appelées relations
- Une table est divisée en colonnes ou attributs
- Les valeurs que peut prendre un attribut est appelé le domaine de cet attribut

Colonne ou Attribut

Pilote

IdPilote	Nom	Prenom	PosteTel
26	PEREZ	Michel	4524
24	ST EXUPERY	Antoine	4522
25	TURNER	Léon	4556
17	PISARINI	Paul	4521

Ligne ou Tuple

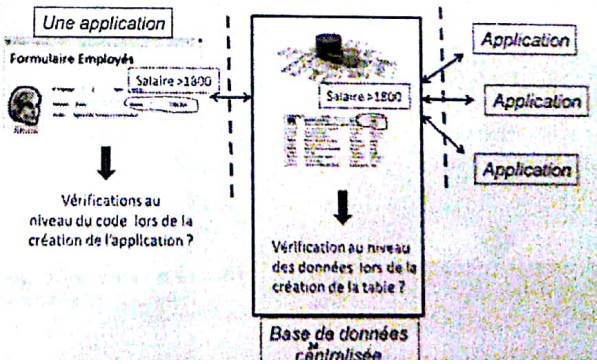
3 concepts importants liés au modèle relationnel :

- Clé primaire
- Clé étrangère
- Contraintes

Cohérence des données

Comment assurer la cohérence des données d'une base de production ?

Solution : mise en place de contrainte d'intégrité (CI) à la création des tables



Contraintes d'intégrités

A la création d'une table, il est possible de mettre en place des contrôles(CI) qui seront vérifiés sur l'ensemble des données de la table à chaque instant.

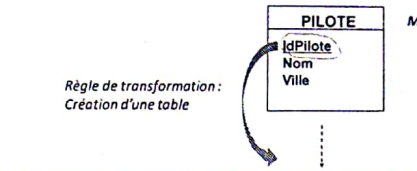
Il existe différents types de contraintes :

1. **Contraintes de données obligatoires :** le nom du pilote est obligatoire
2. **Contrainte booléenne :** le salaire du pilote est > 1800€
3. **Contraintes d'intégrité référentielle :** cohérence des clés étrangères
4. **Contrainte d'unicité de données :** clé primaire ou clé de gestion

Les contraintes plus complexes seront implantées par programmation de TRIGGERS/Déclencheurs

Clé primaire :

Etre capable d'identifier chaque ligne d'une table sans ambigüité



Règle de transformation :
Création d'une table

Base de production

PILOTE (IdPilote, Nom,Ville)

L'identifiant devient la Clé primaire de la table.

Le SGBD doit vérifier que chaque valeur de la colonne « IdPilote » est unique

Clé primaire

Une clé primaire peut être composée d'une ou plusieurs colonnes permettant d'identifier une ligne de manière unique dans la table

RELATION	CLE PRIMAIRE
EMPLOYE	Matricule
OUVRAGE	Réf_ISBN
VEHICULE	Immatriculation
ENTREPRISE	SIRET

- *Chaque table de la base de production a généralement une clé primaire*
- *Le choix de la clé primaire est primordial car il est difficile d'en changer en cours d'exploitation*

Clé primaire par l'exemple

CLE PRIMAIRE

	Nom	Prenom	Tel	Trigramme
10	DUPONT	Jean	06070809	DJE
20	DURANT	Pierre	07080910	DPI

Ajout d'un nouveau client

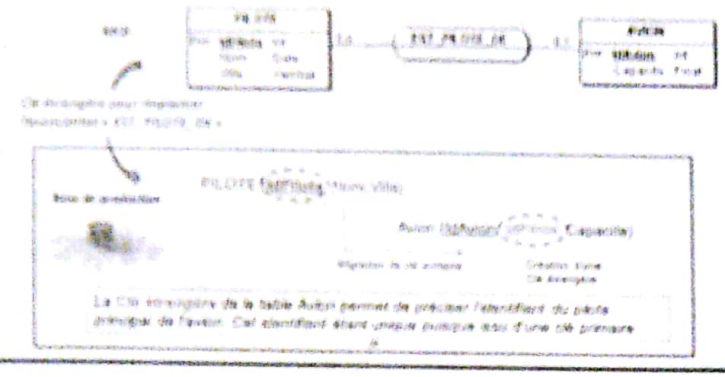
10	ARANU	Marie	0689765643	AMA

Le numéro existe déjà :

- *Le SGBD doit empêcher l'insertion*
- *Il génère une erreur de VIOLATION DE CONTRAINTE UNIQUE*

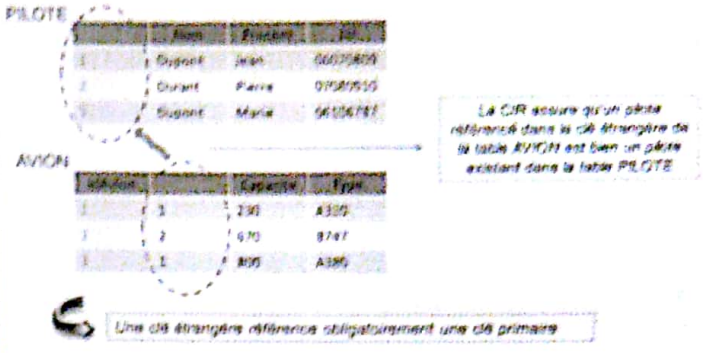
Clé étrangère :

- Clé primaire migrée d'une table à une autre
- Traduit au niveau des tables le lien entre deux entités

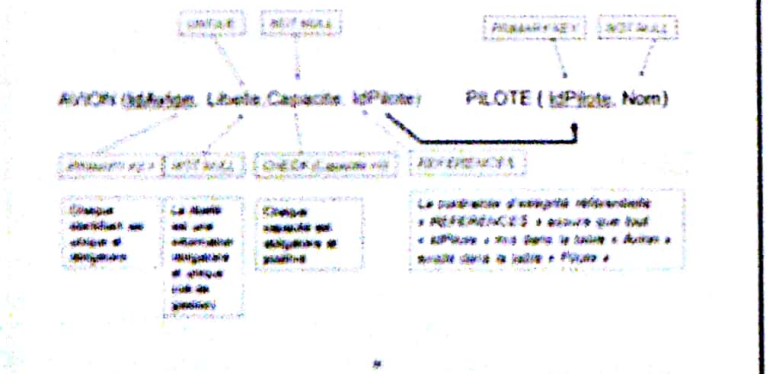


Clé étrangère VS CIR

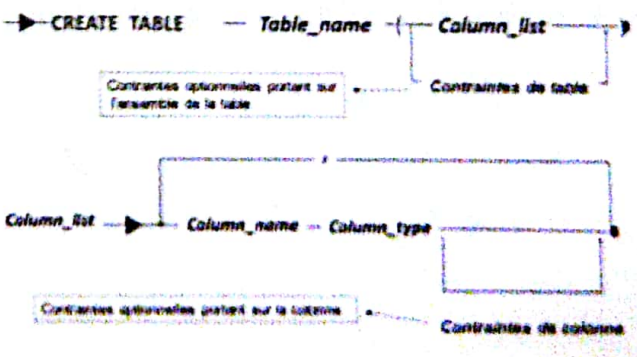
Une Contrainte d'intégrité Référentielle (CIR) est une contrainte portée sur une clé étrangère pour assurer la cohérence des données entre les 2 tables



Contraintes d'intégrité et Mots Clés SQL LDD



Création des tables : syntaxe SQL LDD générale



Les types de données principaux d'ORACLE

CHAR (n) :	chaine de caractères de longueur fixe (0<n<2001)
VARCHAR(n)	chaine de caractères de longueur variable (0<n<4001)
LONG :	chaine de caractères de longueur variable (jusqu'à 2 GO)
CLOB :	Character Large Object ORACLE (jusqu'à 4 GO)
NUMBER(p,s) :	nombre fixe ou flottant (1<p<38, -38<s<38) p : précision - nombre total de chiffres du nombre s : échelle - nombre de chiffres après la virgule
FLOAT(n) :	Nombre Flottant (0<n<126)
DATE :	longueur fixe de 7 octet permettant de gérer le temps et le calendrier
RAW :	données de type binaire de longueur variable (jusqu'à 2000 octets)
LONGRAW :	données de type binaire de longueur variable (jusqu'à 2GO)
BLOB :	Binary Large Object ORACLE (jusqu'à 4 GO)
BFILE:	Binary File ORACLE. Fichier externe dont la taille est limitée par le système d'exploitation ,,

Première implantation de table avec contraintes d'intégrité (1)

```
CREATE TABLE Pilote
(
   IdPilote      INTEGER  PRIMARY KEY,
   NomPilote     VARCHAR(30) NOT NULL
)
```

Contrainte sur colonne : clé primaire de la table

Contrainte sur colonne : donnée obligatoire

34

Première implantation de table avec contraintes d'intégrité (2)

```
CREATE TABLE AVION
(
   IdAvion        INTEGER   PRIMARY KEY,
   LibelleAvion   VARCHAR(30),
   Capacite       INTEGER NOT NULL CHECK (Capacite>0),
   TypeAvion      VARCHAR(10),
   IdPilote       INTEGER REFERENCES PILOTE( IdPilote)
)
```

Clé primaire de la table

Donnée obligatoire

vérification booléenne de la capacité

CIR sur la clé étrangère « IdPilote ». Contrainte de cohérence entre les tables AVION et PILOTE

« La valeur rentrée pour la colonne « IdPilote » doit obligatoirement exister dans la colonne « IdPilote » de la table PILOTE »

35

CIR par un exemple

1	Dupont	Jean	06070809
2	Durant	Pierre	07080910
3	Dupont	Marie	06106787

Contrainte de CIR

REFERENCES

1	1	230	A320
2	2	670	B747
3	100	800	A380

Ajout d'un nouvel avion

Lors de l'ajout d'un nouvel avion, le SGBD générera une erreur : VIOLATION DE LA CONTRAINTE CIR

SQL LMD : compléments

Insertion de données
Modification de données
Suppression de données

Insertion de données dans les tables

Syntaxe SQL de la commande INSERT

INSERT INTO < Nom_Table> [(<Liste_colonnes>)]
{ VALUES (<Liste_valeurs>) | Requête_SQL }

- Exemples :
  - INSERT INTO **AVION (IdAvion, LibelleAvion)** VALUES **(1, 'A380')** → OK
  - INSERT INTO **AVION (LibelleAvion, IdAvion)** VALUES **('A320', 2)** → OK
  - INSERT INTO **AVION** VALUES **(3, 'B747')** → OK mais déconseillé
  - INSERT INTO **AVION** VALUES **('B907', 4)** → KO

N.B. : répéter la commande INSERT pour chaque tuple à insérer

VALUE : une valeur

Modification et Suppression de données des tables

Syntaxe de la commande UPDATE

UPDATE <Nom_Table> SET <Attribut$_1$> = *expr$_1$* [,<Attribut$_2$>= *expr$_2$*] ...

[WHERE *Condition*]

Exemple : *UPDATE AVION SET Capacite = 250 WHERE LibelleAvion= ' A320'*

Syntaxe de la commande DELETE

DELETE FROM <Nom_Table> [WHERE *Condition*]

Exemple : *DELETE FROM Pilote WHERE IdPilote >4*

SQL LMD

VALEUR NULL

Contrainte NOT NULL et valeur NULL

PILOTE NOT NULL NOT NULL

	Nom	Prenom	Salaire
10	Dupont	Jean	32000
20	Durant	Pierre	25600
30	Dupont	Marie	NULL

La donnée n'est pas connue ou connue mais non renseignée

NULL n'est pas une "valeur" comme les autres :

- Elle est valable quel que soit le type de la colonne
- NULL ne correspond à aucune valeur précise

Utilisation de NULL : Exemple

PILOTE

	Nom	Prenom	Salaire
10	Dupont	Jean	32000
20	Durant	Pierre	25600
30	Dupont	Marie	NULL

```
UPDATE Pilote SET Salaire = NULL
WHERE Matricule = 30
```

Ne retourne rien

FAUX

```
SELECT * FROM Pilote WHERE Salaire = NULL;
SELECT * FROM Pilote WHERE Salaire != NULL;
SELECT * FROM Pilote WHERE NULL = NULL;
```

Retourne des valeurs

```
SELECT * FROM Pilote WHERE Salaire IS NULL;
SELECT * FROM Pilote WHERE Salaire IS NOT NULL;
```

Utilisation de NULL : Exemple

PILOTE

	Nom	Prenom	Salaire
10	Dupont	Jean	32000
20	Durant	Pierre	25600
30	Dupont	Marie	NULL

On ajoute 100€ à tous les salaires

```
UPDATE PILOTE SET SALAIRE = SALAIRE + 100;
```

NULL + 100 = NULL

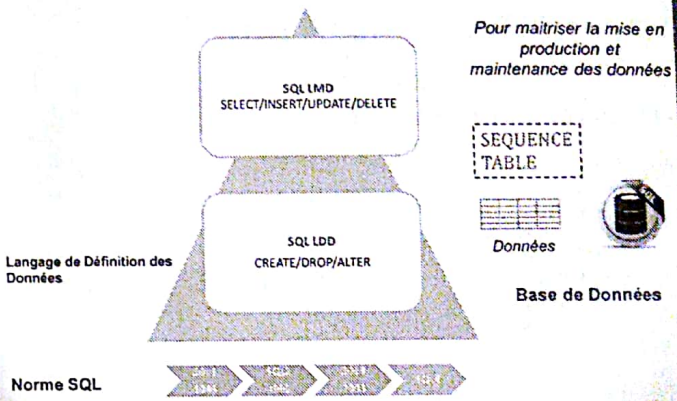
Salaire
32100
25700
NULL

Jérôme FESSY

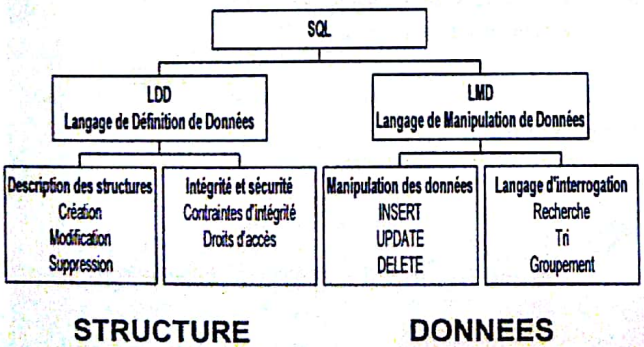
Module SGBD2 (M2106)

Cours 2
SQL LDD (suite)

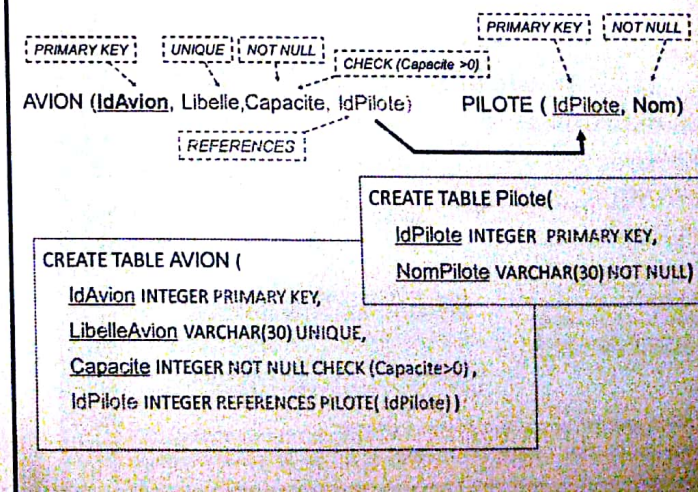
Objectif : maîtriser le langage SQL LDD



RAPPEL : Catégories d'instructions SQL



Rappel : Mots Clés de base du SQL LDD



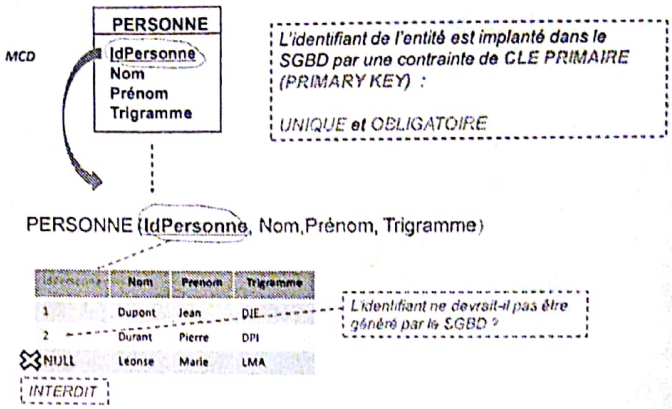
Clé primaire : compléments

Clé physique vs. Clé de gestion

Séquences

Clés primaires : compléments

Les clés numériques de Gestion vs. Physique



PERSONNE (IdPersonne, Nom,Prénom, Trigramme)

IdPersonne	Nom	Prenom	Trigramme
1	Dupont	Jean	DJE
2	Durant	Pierre	DPI
NULL	Leonse	Marie	LMA

INTERDIT

L'identifiant ne devrait-il pas être généré par le SGBD ?

Clés primaires : compléments

comment générer automatiquement des clés primaires numériques ?

```
CREATE TABLE PERSONNE
(
IdPersonne INTEGER PRIMARY KEY
Nom VARCHAR(30),
Prenom VARCHAR(30),
Trigramme CHAR(3)
)
```

① On definit « IdPersonne » comme un entier

② On créé un objet Indépendant
Une SEQUENCE

CREATE SEQUENCE SeqPersonne

③ Chaque insertion ou mise à jour fait appel à la séquence pour générer une nouveau numéro

IdPersonne	Nom	Prenom	Trigramme
1	Dupont	Jean	DJE
2	Durant	Pierre	DPI
3	Léonse	Marie	LMA

INSERT INTO PERSONNE (IdPersonne, Nom, Prenom, Trigramme)
VALUES (SeqPersonne.NEXTVAL, 'Léonce', 'Marie', 'LMA')

LES SEQUENCES

- Gestion de compteurs dans la norme SQL pour les clés primaires numériques
- Une séquence est un objet de la base de données
- On « associe » généralement une séquence par table

EXEMPLE :

```
CREATE SEQUENCE Seq_Pilote;
INSERT INTO Pilote (matricule, nom) Values (Seq_Pilote.nextval, ' Garros ');
```

La colonne doit être de type INTEGER

SeqPilote

2 fonctions associées aux séquences :
Nextval : valeur suivante du compteur
Currval : valeur courante du compteur

La même séquence peut être utilisée pour plusieurs tables

SYNTAXE SQL Norme des SEQUENCES

Création d'une séquence :

CREATE SEQUENCE nom_séquence
[INCREMENT BY integer]
[START WITH integer]
[MINVALUE integer]
[MAXVALUE integer]
[{ CYCLE | NOCYCLE }]

Suppression d'une séquence : DROP SEQUENCE nom_séquence

Modification d'une séquence : ALTER SEQUENCE nom_séquence ....

INCREMENT BY : spécifie l'intervalle d'incrémentation. Cette valeur peut être positive ou négative mais ne peut pas valoir 0. Par défaut, l'incrémentation vaut 1.
START WITH : spécifie la première valeur délivrée par la séquence. Par défaut, cette valeur est la valeur minimum pour les incrémentations positives et la valeur maximum pour les négatives.
MINVALUE :spécifie la valeur minimum de la séquence. Doit être au minimum égal à la valeur spécifiée par START WITH.
MAXVALUE : spécifie la valeur maximum que le compteur peut générer.
CYCLE : permet à la séquence de recommencer lorsqu 'elle atteint le maximum. NOCYCLE interdit à la séquence de générer des valeurs lorsqu 'elle atteint ce seuil.

C'est compliqué! N'y-a-t'il pas plus simple que les séquences pour générer une clé automatiquement ?

CREATE TABLE PERSONNE
(IdPersonne AUTO_INCREMENT PRIMARY KEY,
Nom VARCHAR(30), ...) MYSQL

CREATE TABLE PERSONNE
(IdPersonne NuméroAuto PRIMARY KEY,
Nom VARCHAR(30), ...) ACCESS

Champ auto incrémenté

SI ....

MAIS

Le compteur est dédié à une table et il ne peut pas être partagé entre 2 tables ni utilisé en programmation

INSERT INTO PERSONNE (Nom, Prenom, Trigramme)
VALUES ('Léonce', 'Marie', 'LMA')

	Nom	Prenom	Trigramme
1	Dupont	Jean	DJE
2	Durant	Pierre	DPI
3	Léonse	Marie	LMA

Peut-on faire la même chose sur ORACLE ou Postgres?

CREATE TABLE PERSONNE
(IdPersonne NUMBER GENERAYED ALWAYS AS IDENTITY PRIMARY KEY,
Nom VARCHAR(30), ...) ORACLE 12c

CREATE TABLE PERSONNE
(IdPersonne SERIAL PRIMARY KEY,
Nom VARCHAR(30), ...) PostgreSQL

Champ auto incrémenté

INSERT INTO PERSONNE (Nom, Prenom, Trigramme)
VALUES ('Léonce', 'Marie', 'LMA')

	Nom	Prenom	Trigramme
1	Dupont	Jean	DJE
2	Durant	Pierre	DPI
3	Léonse	Marie	LMA

- Type non normalisé
- Génère automatiquement une séquence associée :

SELECT * FROM USER_SEQUENCES

Clés primaires :

Une clé primaire associée à un compteur garantit-il forcément l'unicité des données ?

On utilise une SEQUENCE

IdClient	Nom	Prenom	Tel	Trigramme
1	Dupont	Jean	06070809	DJe
2	Durant	Pierre	07080910	DPi
3	Dupont	Jean	068976564	DJe

PRIMARY KEY
Toutes les valeurs sont distinctes et obligatoires

UNIQUE(Nom,Prenom,Tel)
Toutes les valeurs sont distinctes mais potentiellement NULL

UNIQUE(Trigramme)
Toutes les valeurs sont distinctes mais potentiellement NULL

Idclient est un compteur mais rien n'empêche d'insérer 2 fois la même personne

NON ! Si la clé primaire physique n'est pas une clé de gestion. Il faut ajouter des contrainte de clés de Gestion (UNIQUE)

Clés primaires :

Une clé primaire associée à un compteur garantit-il forcément l'unicité des données ?

OUI ! Si la clé primaire physique est aussi une clé de gestion

FACTURE

	DateFacture	IdClient
20181	21/04/12	1
20182	22/04/13	102
20183	15/12/13	54

PRIMARY KEY

Toutes les valeurs sont distinctes et obligatoires

Idfacture est un compteur. La clé physique est aussi une clé de gestion. Donc il n'est pas possible d'insérer 2 fois la même facture

Clés primaires : compléments

Clé physique vs. Clé de gestion ?

```
CREATE TABLE PERSONNE                    (1)
( IdPersonne INTEGER PRIMARY KEY
Nom VARCHAR(30),
Prenom VARCHAR(30),
Tel VARCHAR(15),
Trigramme CHAR(3) ,
(2) UNIQUE (Nom,Prenom,Tel),
(3) UNIQUE(Trigramme) )
```

IdPersonne	Nom	Prenom	Trigramme
1	Dupont	Jean	DJE
2	Durant	Pierre	DPI
3	Léonse	Marie	LMA

« IdPersonne » n'est pas une clé de gestion

```
CREATE TABLE FACTURE
( IdFacture INTEGER PRIMARY KEY (1)
Libelle VARCHAR(30),
DateFacture DATE )
```

IdFacture	Libelle	DateFact
20141	Nettoyage	12/01/14
20142	Nettoyage	12/01/14
20143	Nettoyage	13/01/14

« IdFacture » est aussi une clé de gestion

Une table peut-elle avoir plusieurs clés primaires ?

- OUI, logiquement une table peut avoir une clé physique et une ou plusieurs clés de gestion

CLIENT (IdClient, Nom, Prenom, Tel, Trigramme)

PRIMARY KEY — UNIQUE — UNIQUE

- MAIS, une table ne peut avoir qu'une seule PRIMARY KEY
- ATTENTION, les clés primaires peuvent être composées de plusieurs colonnes

PRIMARY KEY (Nom, Prenom, Tel) UNIQUE(Nom, Prenom, Tel)

Ce sont des clés composées

Puis-je définir une table sans clé primaire ?

- OUI

Table sans contrainte

CREATE TABLE Personnel (IdPers Integer, Nom Varchar(30))

- Mais ces tables sont rares en production car l'intérêt du modèle relationnel est de mettre en relation les données de différentes tables par des clés étrangères.

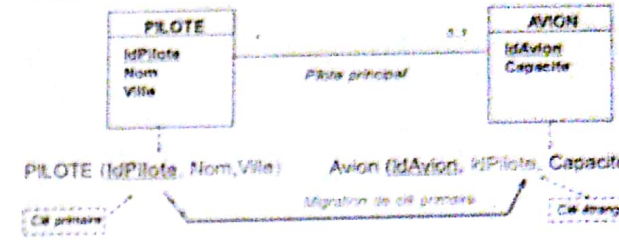
Il faut donc pouvoir les identifier sans ambigüité

Clé étrangère : compléments

Option de la CIR

Clés étrangères : compléments

Rappel



PILOTE (IdPilote, Nom,Ville) Avion (IdAvion, IdPilote, Capacite)

Clé primaire Migration de clé primaire Clé étrangère

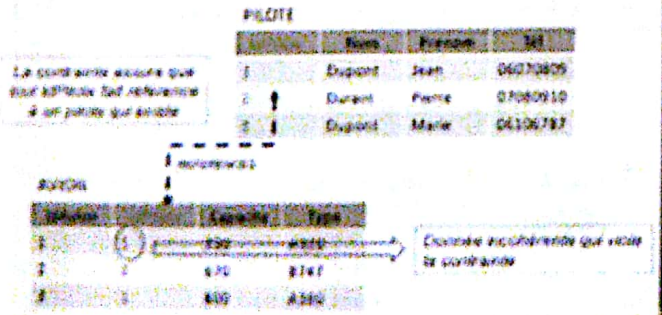
La clé étrangère « IdPilote » de la table « Pilote » permet de retrouver le pilote principal d'un avion

```
CREATE TABLE AVION (
IdAvion INTEGER PRIMARY KEY,
Capacite INTEGER,
IdPilote INTEGER REFERENCES PILOTE( IdPilote))
```

Une contrainte de REFERENCE est associée à une clé étrangère

Clés étrangères :

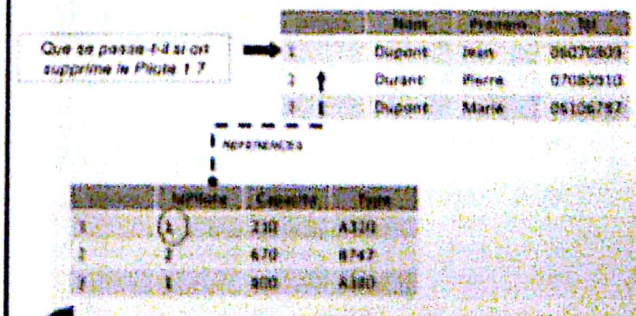
Rappel contrainte d'intégrité référentielle



Le SGBD arrête aussitôt que tout IdPilote fait référence à un pilote qui existe

Exemple incohérence qui viole la contrainte

Que se passe-t-il en cas de suppression ou de modification d'un pilote référencé par un avion ?



Réponse : Sans option définie sur la CIR, la suppression ou la modification d'un pilote référencé est IMPOSSIBLE si générait une violation de la contrainte de REFERENCE

Clés étrangères :
Options de la contrainte d'intégrité référentielle (CIR)

Syntaxe SQL Norme :
REFERENCES nom_table (liste_colonnes)
[ON DELETE {NO ACTION | CASCADE | SET NULL | SET DEFAULT }]
[ON UPDATE {NO ACTION | CASCADE | SET NULL | SET DEFAULT }]

ON DELETE : action à effectuer sur la table référençante lors d'une suppression dans la table référencée

ON UPDATE : action à effectuer sur la table référençante lors d'une modification dans la table référencée

CASCADE : effacer ou mettre à jour les tuples correspondant dans la table référençante

SET NULL : tuple cible supprimé ou modifié et tuples clé étrangère associés initialisés à NULL

SET DEFAULT :tuple cible supprimé ou modifié et tuples clé étrangère associés initialisés à leur valeur par défaut

NO ACTION : équivalent à l'omission des clauses ON DELETE, ON UPDATE

Clés étrangères : compléments
Options de la contrainte d'intégrité référentielle (CIR)

OPTION : ON DELETE CASCADE

PILOTE

	Nom	Prenom	Tél
1	Dupont	Jean	06070809
2	Durant	Pierre	07080910
3	Dupont	Marie	06106787

① *Suppression du pilote 1 ?*

REFERENCES ON DELETE CASCADE

AVION

IdAvion		Capacite	Type
~~1~~	~~1~~	~~230~~	~~A320~~
2	2	670	B747
~~3~~	~~1~~	~~800~~	~~A380~~

② *Suppression en cascade de tous les avions référençant le pilote 1*

⇨ *La cohérence des données est assurée*

Clés étrangères : compléments
Options de la contrainte d'intégrité référentielle (CIR)

OPTION : ON DELETE SET NULL

PILOTE

	Nom	Prenom	Tél
1	Dupont	Jean	06070809
2	Durant	Pierre	07080910
3	Dupont	Marie	06106787

① *Suppression du pilote 1 ?*

REFERENCES ON DELETE SET NULL

AVION

IdAvion		Capacite	Type
1	NULL	230	A320
2	2	670	B747
3	NULL	800	A380

② *Mise à NULL des klPilote de tous les avions référençant le pilote 1*

⇨ *La cohérence des données est assurée*

C'est moi qui décide de mettre ON DELETE CASCADE ?

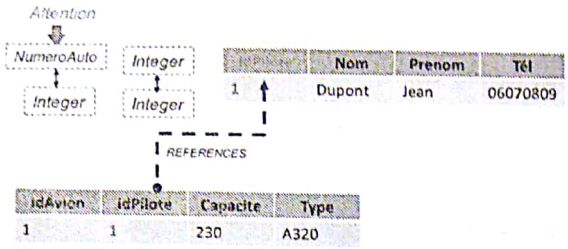
Réponse : NON. Tout dépend de la modélisation. C'est elle qui implique cette option dans certains cas

Exemple d'entité FORTE/FAIBLE (MERISE) ou COMPOSITION (UML)

Une salle est une entité relative par rapport à un cinéma.
La suppression d'un cinéma entraine la suppression de ses salles

Le type de donnée d'une clé étrangère peut-il être différent du type de la clé primaire qu'elle référence ?

- NON, une clé étrangère est une clé primaire migrée. C'est donc la même colonne avec exactement le même type



Une clé étrangère doit-elle nécessairement référencer une clé primaire ?

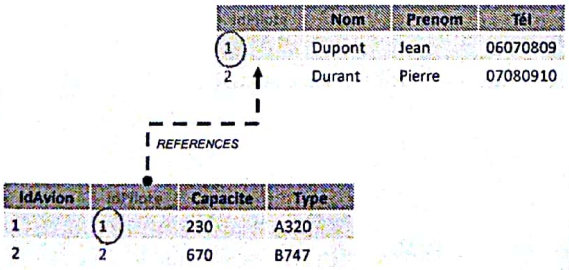
- OUI sinon cela n'a pas de sens!
- On référence une donnée qui peut être identifiée sans ambiguïté
- Techniquement :

REFERENCES PILOTE(IdPilote);

Génère une erreur si IdPilote n'est pas défini comme PRIMARY KEY ou UNIQUE dans la table PILOTE

Une clé étrangère assure-t-elle la cohérence des données entre les deux tables concernées ?

- NON, une clé étrangère est une colonne de lien et non une vérification !
- C'est la CIR portée sur cette colonne qui vérifie cette cohérence



SQL LMD

Compléments

Valeurs par défaut

Insertion de données

Usage de la valeur par défaut

```
CREATE TABLE Commande
( IdCom INTEGER,
LibelleCom VARCHAR(30) NOT NULL,
DateCom DATE DEFAULT SYSDATE
MontantCom INT DEFAULT 0 )
```

La valeur par défaut de la colonne est la valeur NULL

La valeur par défaut de la colonne : date système

La valeur par défaut de la colonne : 0

```
INSERT INTO Commande (IdCom, LibelleCom, DateCom, MontantCom)
VALUES (1, 'Commande Ballon', '17/01/2018', 173)
```

Toutes les valeurs sont spécifiées

```
INSERT INTO Commande (IdCom, LibelleCom, DateCom, MontantCom)
VALUES (1, 'Commande Ballon',NULL,NULL)
```

La date et le montant ne sont pas connus et sont explicitement mis à NULL
On se prive alors des valeurs par défaut spécifiées à la création

Insertion de données

Usage de la valeur par défaut

```
CREATE TABLE Commande
( IdCom INTEGER,
LibelleCom VARCHAR(30) NOT NULL,
DateCom DATE DEFAULT SYSDATE
MontantCom INT DEFAULT 0 )
```

La valeur par défaut de la colonne est la valeur NULL

La valeur par défaut de la colonne : date système 0

```
INSERT INTO Commande (IdCom, LibelleCom)
VALUES (1, LibelleCom)
```

Le montant et la date ne sont pas explicitement spécifiés. Ils prendront leurs valeurs par défaut

1	REF123	10/01/2018	0

```
INSERT INTO Commande (IdCom, LibelleCom,DateCom)
VALUES (1, LibelleCom, DEFAULT)
```

SYSDATE

On peut utiliser le mot Clé DEFAULT pour explicitement montrer qu'on utilise la valeur par défaut

1	REF123	10/01/2018	0

Mise à jour de données avancée

Mise à jour entre tables

PILOTE

IdPilote	Nom	Prenom	AnneeNais
1	Dupont	Jean	NULL
2	Durant	Pierre	NULL
3	Dupont	Marie	NULL

Mise à jour ?

T_REF_ANNEE

Idpilote	AnneeNais
1	1925
2	1997
3	1958

```
UPDATE PILOTE SET AnneNaiss = ?
```

Comment mettre à jour la table PILOTE à partir d'une autre table de référence ?

Mise à jour de données avancée

Mise à jour entre tables

PILOTE

IdPilote	Nom	Prenom	AnneeNais
1	Dupont	Jean	NULL
2	Durant	Pierre	NULL
3	Dupont	Marie	NULL

Mise à jour ?

T_REF_ANNEE

Idpilote	AnneeNais
1	1925
2	1997
3	1958

jointure

```
UPDATE PILOTE P
SET AnneeNais = (   SELECT AnneeNais
                    FROM T_REF_ANNEE T
                    WHERE T.IdPilote = P.IdPilote
                )
```

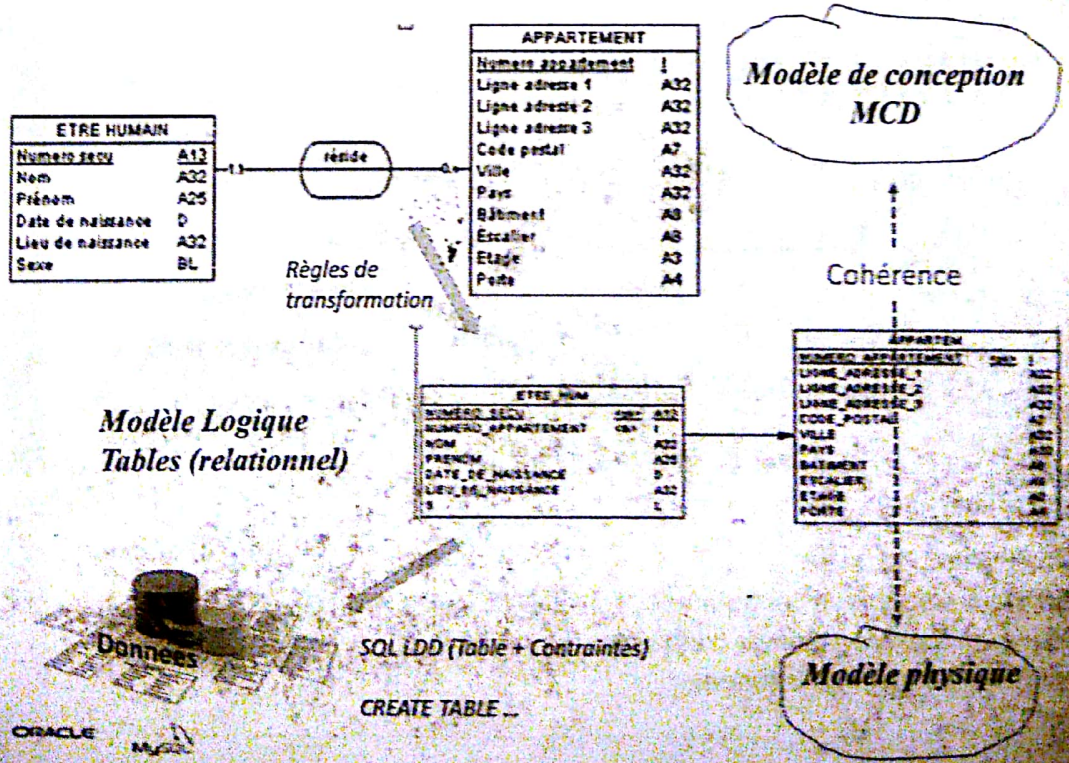
Jérôme FESSY

Module SGBD2 (M2106)
Cours 3
SQL LDD (suite)

Nommage de constraintes
ALTER TABLE
Bonnes pratiques

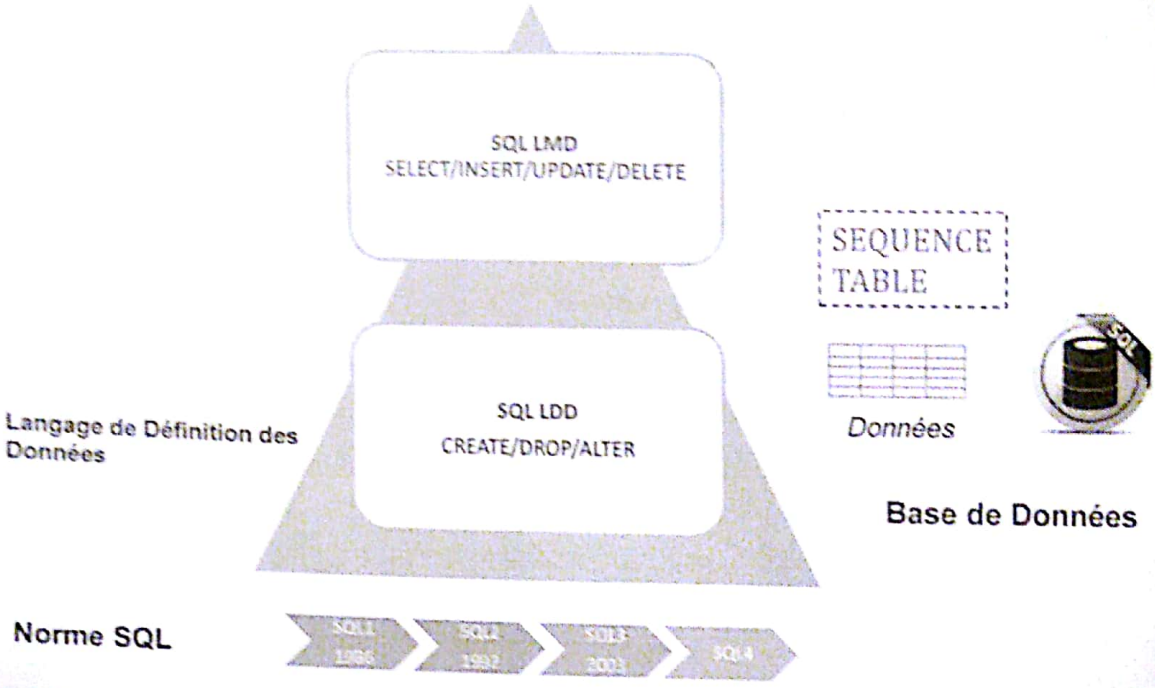
Jérôme FESSY

Objectif : Implanter exactement le modèle de l'analyste

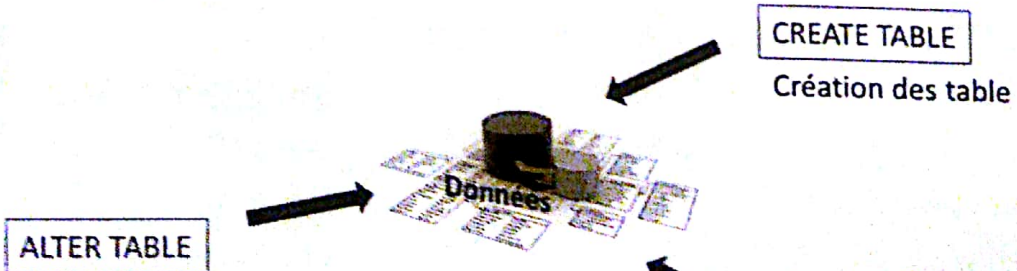


Fessy Mesnil Florence

Cela passe par la maîtrise du langage SQL LDD



GESTION DES TABLES



Création des tables

Types SQL

Types ORACLE

CREATE TABLE

Création des table

Rappel : *CREATE TABLE*

PRIMARY KEY *UNIQUE* *NOT NULL* *CHECK (Capacite >0)* *PRIMARY KEY* *NOT NULL*

AVION (**IdAvion**, Libelle,Capacite, IdPilote) PILOTE (IdPilote, Nom)

REFERENCES

```
CREATE TABLE Pilote(
    IdPilote INTEGER  PRIMARY KEY,
    NomPilote VARCHAR(30) NOT NULL)
```

```
CREATE TABLE AVION (
    IdAvion INTEGER PRIMARY KEY,
    LibelleAvion VARCHAR(30) UNIQUE,
    Capacite INTEGER NOT NULL CHECK (Capacite>0) ,
    IdPilote INTEGER REFERENCES PILOTE( IdPilote) )
```

Création de Tables : Syntaxe SQL étoffée

CREATE TABLE <Nom_table> (<def_colonne>*, [def_contrainte_table>*])

attributs et contraintes — *Contraintes niveau table*

<def_colonne> ::=
<nom_colonne> <type>
[<**NOT NULL** | **UNIQUE** | **DEFAULT** valeur |
[**CONSTRAINT** nom_contrainte
CHECK (condition) |
| **PRIMARY KEY** |
REFERENCES nom_table (liste_colonnes)
[ON DELETE {NO ACTION | CASCADE | SET NULL | SET DEFAULT }]
[ON UPDATE {NO ACTION | CASCADE | SET NULL | SET DEFAULT }] >]

<def_contrainte_table> ::= **CONSTRAINT** nom_contrainte
<**PRIMARY KEY (liste_colonnes)** |
CHECK (condition) |
FOREIGN KEY (liste_colonnes) **REFERENCES** nom_table(liste_colonnes) >

Types de données SQL (1)

Types booléens et binaires

Type	Description	Normalisation
Boolean, bool	**Une valeur unique true ou false**	**SQL99**
Bit(n)	**Chaîne de bit de longueur n fixe**	**SQL92**
Bit varying(n), varbit(n)	**Chaîne de bit de longueur n variable**	**SQL92**

Types caractères

Type	Description	Normalisation
Character(n), char(n)	**Chaîne de caractères de longueur n fixe**	**SQL89**
text	**Chaîne de caratères de longueur variable et illimitée**	**SQL92**
caracter varying(n), varchar(n)	**Chaîne de caratères de longueur n variable**	**SQL99**

Types de données SQL (2)

Types numériques

Type	Description	Normalisation
Smallint, int2	**Entier signé sur 2 octets**	**SQL89**
Integer, int, int4	**Entier signé sur 4 octets**	**SQL92**
real, float4	**Nombre à virgule flottante sur 4 octets**	**SQL89**
Double precision, float8, float	**Nombre à virgule flottante sur 8 octets**	**SQL89**
Numeric(p,s), decimal(p,s)	**Type numérique exact avec une précision p quelconque et un facteur d'échelle s.**	**SQL99**

Types de données SQL (3)

Types date et heure

Type	Description	Normalisation
date	**Date du calendrier (jour, mois, année)**	**SQL92**
time	**heure**	**SQL92**
Time with time zone	**Heure avec les informations sur la zone horaire**	**SQL92**
Timestamp with time zone	**Date et heure**	**SQL92**
interval	**Délai quelconque**	**SQL92**

SQL propose également d'autres types plus spécifiques

Type FLOAT

- FLOAT (p)
  - "p" est la précision binaire
    - Valeur entre 1 et 126 (défaut)
- Le type FLOAT est stocké comme un type NUMBER
- La tentative d'insérer, dans une colonne, un nombre ayant une précision qui dépasse celle de la colonne provoque une erreur
- Les types ANSI convertis en FLOAT
  - REAL est converti en FLOAT(63)
  - DOUBLE PRECISION est converti en FLOAT(126)

Chaînes de caractères

- CHAR (n [BYTE | CHAR])
  - Longueur fixe avec un maximum absolu de 2000 octets
    - Si la valeur a une longueur inférieure à celle de la définition de la colonne alors des espaces sont ajoutés
- VARCHAR2 (n [BYTE | CHAR])
  - Longueur variable avec un maximum absolu de 4000 octets
  - VARCHAR est un synonyme de VARCHAR2 mais il est susceptible de changer de définition ultérieurement
- NCHAR (n) / NVARCHAR2 (n)
  - NCHAR est pour les chaînes de longueur fixe
  - NVARCHAR2 est pour les chaînes de longueur variable
  - Dans les deux cas, Oracle utilise le jeu de caractères national

Type DATE

- Description
  - Précision jusqu'aux secondes près
    - Pas de fractions de secondes ni fuseau horaire
  - Intervalle 1er janvier 4712 av. J.-C. .. 31 décembre 9999 apr. J.-C.
- Toute date est, par défaut, interprétée/affichée en fonction du format désigné par un paramètre de session NLS_DATE_FORMAT
- Les constantes de type DATE peuvent être spécifiées comme une chaîne de caractères (préfixée eventuellement par "DATE«)
  - Exemple
    - '6/02/2018'
    - DATE'2018-02-06'
    - DATE'2018/02/06' -- Erreur
    - DATE'06-02-2018' -- Erreur

Données binaires brutes

- RAW(n)
  - Suite de bits de longueur variable avec une valeur maximale de 2000 pour n
  - Une constante de type RAW peut être exprimée par une chaîne de caractères en HEXA
- Si la longueur des données dépasse 2000 octets
  - Avant Oracle 8
    - Le type LONG RAW peut être utilisé
      - Ce type peut aller jusqu'à 2GO
      - Le type LONG RAW a les mêmes restrictions que le type LONG
  - À partir d'Oracle 8, le type LONG RAW doit être abandonné
    - Il est recommandé d'utiliser le type BLOB
      - La taille maximale des données dans le cas du type BLOB est de 4 GO

Type ROWID

- ROWID
  - Permet de représenter l'emplacement physique du stockage d'une ligne dans une table
  - Peut être affiché sous SQL*Plus comme une chaîne de 18 caractères
- UROWID [(n)]
  - Une chaîne hexadécimale représentant l'adresse d'une ligne
    - La longueur maximale, qui est d'ailleurs utilisée par défaut, vaut 4000 octets
  - Ce type permet de représenter
    - L'adresse logique d'une ligne dans une table-index
    - L'adresse d'une ligne dans une base de données non Oracle
    - L'adresse d'une ligne quelconque de la base Oracle

Les types de données principaux d'ORACLE

CHAR (n) :	**chaîne de caractères de longueur fixe (0<n<2001)**
VARCHAR(n) :	**chaîne de caractères de longueur variable (0<n<4001)**
LONG :	**chaîne de caractères de longueur variable (jusqu'à 2 GO)**
CLOB :	**Character Large Object ORACLE (jusqu'à 4 GO)**
NUMBER(p,s) :	**nombre fixe ou flottant (1<p<38, -38<s<38)** **p : précision - nombre total de chiffres du nombre** **s : échelle - nombre de chiffres après la virgule**
FLOAT(n) :	**Nombre Flottant (0<n<126)**
DATE :	**longueur fixe de 7 octet permettant de gérer le temps et le calendrier**
RAW :	**données de type binaire de longueur variable (jusqu'à 2000 octets)**
LONGRAW :	**données de type binaire de longueur variable (jusqu'à 2GO)**
BLOB :	**Binary Large Object ORACLE (jusqu'à 4 GO)**
BFILE:	**Binary File ORACLE. Fichier externe dont la taille est limitée par le système d'exploitation**

Le choix du type d'une colonne est IMPORTANT

```
CREATE TABLE PERSONNE (
    IdPersonne INTEGER PRIMARY KEY,
    NomPersonne VARCHAR(30),
    DateNaissance DATE,
    CodePostal CHAR(5),
    EstFumeur CHAR(1) )
```

Quelles tailles max auront les données ?

La taille sera toujours fixe ?

Booléen

Pour quel usage stocke-t-on cette date ?

- *SELECT DateNaissance FROM PERSONNE*
- *SELECT TO_CHAR(DateNaissance, 'YYYY') FROM PERSONNE*

Création des tables
Nommage des contraintes

Intérêt ?

Syntaxe

Nommage des contraintes

- Toute contrainte est unique
- Le SGBD attribut un nom à chaque contrainte
- Lors d'une violation de contrainte, le nom de la contrainte est précisé

```
CREATE TABLE AVION (
    IdAvion INTEGER PRIMARY KEY,
    LibelleAvion VARCHAR(30) UNIQUE,
    Capacite INTEGER CHECK (Capacite>0) ,
    IdPilote INTEGER REFERENCES PILOTE( IdPilote) )
```

INSERT INTO AVION(IdAvion, LibelleAvion, Capacite, IdPilote)
VALUES (12, 'Airbus A320', 250, 100)

⟶ *Erreur : Violation de la contrainte ORASYS3003*

Nommage des contraintes : CONSTRAINT

Mettre en place une nomenclature de nommage de ses contraintes permet :

- Une maintenance simplifiée
- Facilite l' Activation/ Désactivation des contraintes en cours d'exploitation

Nom de la contrainte

```
CREATE TABLE AVION (
IdAvion  INTEGER CONSTRAINT PK_Avion_IdAvion PRIMARY KEY,
LibelleAvion VARCHAR(30) CONSTRAINT U_Avion_Libelle UNIQUE,
Capacite INTEGER CONSTRAINT CK_Avion_Capacite CHECK (Capacite>0),
IdPilote INTEGER CONSTRAINT FK_Avion_Pilote REFERENCES PILOTE( IdPilote) )
```

Nomenclature possible:
PK : Primary Key
CK : CheCK
FK : Foreign Key
U : Unique

Nommage des contraintes : CONSTRAINT

Mettre en place une nomenclature de nommage de ses contraintes permet :

- Une maintenance simplifiée
- Facilite l' Activation/ Désactivation des contraintes en cours d'exploitation

```
CREATE TABLE AVION (
IdAvion INTEGER CONSTRAINT PK_Avion_IdAvion PRIMARY KEY,
LibelleAvion VARCHAR(30) CONSTRAINT U_Avion_Libelle UNIQUE,
Capacite INTEGER CONSTRAINT CK_Avion_Capacite CHECK (Capacite>0),
IdPilote INTEGER CONSTRAINT FK_Avion_Pilote REFERENCES PILOTE( IdPilote) )
```

```
INSERT INTO AVION(IdAvion, LibelleAvion, Capacite, IdPilote)
VALUES ( 12, 'Airbus A320', 250, 100)
```

⟶ *Erreur : Violation de la contrainte FK_AVION_PILOTE*

MODIFICATION DES TABLES
ALTER TABLE

ALTER TABLE

Modification des tables :

Ajout/Modification/suppression de colonnes
Ajout/suppression de contraintes
Activation/Désactivation de contraintes

Ajout de colonnes
Modification de colonnes
Suppression de colonnes

Ajout de colonnes

Syntaxe

ALTER TABLE *table*
ADD
(
column datatype [**DEFAULT** *expr*] [*column_constraint*] ...
[, *column datatype* [**DEFAULT** *expr*] [*column_constraint*] ...] ...
)

Exemples

```
ALTER TABLE Depart
ADD (PlacesReservees NUMBER DEFAULT 0)

ALTER TABLE Pilote
ADD (Sexe CHAR(1) CONSTRAINT CK_SEXE CHECK (UPPER(Sexe) IN ('F', 'M')))
```

Modification de colonnes

Syntaxe

ALTER TABLE *table*
MODIFY
(
column [*datatype*] [**DEFAULT** *expr*] [*column_constraint*] ...
[, *column* [*datatype*] [**DEFAULT** *expr*] [*column_constraint*] ...] ...
)

Exemples

```
ALTER TABLE Pilote
MODIFY ( Nom VARCHAR(50) );
```

Impossible si des données stockées ne correspondent pas à la modification

```
ALTER TABLE Pilote
MODIFY ( salaire DEFAULT 2000)
```

Suppression de colonnes

Syntaxe

ALTER TABLE *table*
DROP { (*liste_columns*) | **COLUMN** *column* }
[**CASCADE CONSTRAINTS**]

Exemples

```
ALTER TABLE Pilote
DROP (Sexe);
```

Impossible de supprimer une colonne si c'est une clé primaire référencée

```
ALTER TABLE Pilote
DROP COLUMN Sexe;
```

Ajout de contraintes
Suppression de contraintes

Jérôme HESSY

Ajout et suppression d'une contrainte NOT NULL

Ajout de la contrainte

```
ALTER TABLE table
MODIFY
(
  colonne [ CONSTRAINT contrainte ]
  NOT NULL
)
```

```
ALTER TABLE table
ADD ( [ CONSTRAINT contrainte ]
  CHECK (colonne IS NOT NULL )
)
```

Exemples

```
ALTER TABLE Pilote
MODIFY (
    Nom CONSTRAINT Nom_not_nul NOT NULL );
```

A privilégier

```
ALTER TABLE Pilote
ADD CONSTRAINT Nom_not_nul
CHECK (Nom IS NOT NULL );
```

Suppression de la contrainte

```
ALTER TABLE table
DROP CONSTRAINT contrainte
```

```
ALTER TABLE Pilote
DROP CONSTRAINT Nom_not_null;
```

Ajout et suppression d'une clé UNIQUE

Ajout de la contrainte

```
ALTER TABLE table
ADD [ CONSTRAINT contrainte ]
UNIQUE (Liste_colonnes)

ALTER TABLE table
ADD (
    [ CONSTRAINT contrainte ]
    UNIQUE (Liste_colonnes)
)
```

Exemples

```
ALTER TABLE Pilote
ADD CONSTRAINT UC_Pilote
UNIQUE (Matricule);

ALTER TABLE Pilote
ADD (
    CONSTRAINT UC_Pilote
    UNIQUE (Matricule) );
```

Suppression de la contrainte

```
ALTER TABLE table
DROP { CONSTRAINT contrainte |
       UNIQUE (Liste_colonnes) }
[ CASCADE ]
```

```
ALTER TABLE Pilote
DROP UNIQUE (Matricule) CASCADE;
```

Ajout et suppression d'une clé primaire

Ajout de la contrainte

```
ALTER TABLE table
ADD [ CONSTRAINT contrainte ]
PRIMARY KEY (Liste_colonnes)

ALTER TABLE [ schéma. ] table
ADD (
  [ CONSTRAINT contrainte ]
  PRIMARY KEY (Liste_colonnes)
)
```

Exemples

```
ALTER TABLE Pilote
ADD CONSTRAINT PK_Pilote
PRIMARY KEY (Matricule);

ALTER TABLE Pilote
ADD (
    CONSTRAINT PK_Pilote
    PRIMARY KEY (Matricule) );
```

Suppression de la contrainte

```
ALTER TABLE table
DROP { CONSTRAINT contrainte |
       PRIMARY KEY }
[ CASCADE ]
```

```
ALTER TABLE Pilote
DROP PRIMARY KEY CASCADE;
```

Ajout et suppression d'une contrainte CHECK

Ajout de la contrainte

```
ALTER TABLE table
MODIFY
(
   colonne [ CONSTRAINT contrainte ]
   CHECK ( condition )
);
ALTER TABLE table
ADD
(
   [ CONSTRAINT contrainte ]
   CHECK ( condition )
)
```

Exemples

```
ALTER TABLE Pilote
MODIFY
(
   age CONSTRAINT ck_age_valide
   CHECK (age BETWEEN 25 AND 50)
);

ALTER TABLE Pilote
ADD
(
   CONSTRAINT ck_age_valide
   CHECK (age BETWEEN 25 AND 50)
);
```

Suppression de la contrainte

```
ALTER TABLE table
DROP CONSTRAINT contrainte
```

```
ALTER TABLE Pilote
DROP CONSTRAINT ck_age_valide;
```

Ajout et suppression d'une contrainte référentielle

Ajout de la contrainte

```
ALTER TABLE table
MODIFY
(
 colonne [ CONSTRAINT contrainte ]
 REFERENCES [schéma.]table(colonne)
 ON DELETE { CASCADE | SET NULL }
)

ALTER TABLE  table
ADD
(
 CONSTRAINT contrainte
 FOREIGN KEY (liste_colonnes)
 REFERENCES [schéma.]table(liste_colonnes)
 ON DELETE { CASCADE | SET NULL } )
```

Exemples

```
ALTER TABLE Depart
MODIFY
(
   Matricule CONSTRAINT fk_matricule
   REFERENCES Pilote (Matricule)
   ON DELETE SET NULL
);
ALTER TABLE Depart
ADD
(
   CONSTRAINT fk_matricule
   FOREIGN KEY (Matricule)
   REFERENCES Pilote (Matricule)
   ON DELETE SET NULL );
```

Suppression de la contrainte

```
ALTER TABLE [ schéma. ] table
DROP CONSTRAINT contrainte
```

```
ALTER TABLE Pilote
DROP CONSTRAINT fk_matricule;
```

Activation / Désactivation d'une contrainte

Activation de la contrainte

ALTER TABLE *table*
ENABLE
CONSTRAINT *contrainte*;

Exemples

ALTER TABLE Pilote
ENABLE CONSTRAINT pk_Pilote;

Désactivation de la contrainte

ALTER TABLE *table*
DISABLE
CONSTRAINT *contrainte*;

ALTER TABLE Pilote
DISABLE CONSTRAINT pk_Pilote;

Les contraintes d'une table peuvent être désactivées ou réactivées en cours d'exploitation

ALTER TABLE :

Ajout de contraintes en cours d'exploitation

- Si la contrainte que l'on souhaite ajouter viole les données déjà présentes dans la base, la contrainte ne peut pas être ajoutée
- Il faut résoudre l'incohérence des données avant de pouvoir faire cet ajout

PILOTE

IdPilote	Nom	Prenom	Tél
1	Dupont	Jean	06070809
2	Durant	Pierre	07080910
1	Dupont	Marie	06106787

ALTER TABLE PILOTE ADD CONSTRAINT PK_Pilote PRIMARY KEY(IdPIlote)

SUPPRESSION DES TABLES

DROP TABLE :

Suppression des tables

La suppression d'une table entraine la suppression des données de la table ainsi que des contraintes associées à la table

SYNTAXE SQL : DROP « nom de table » [CASCADE CONSTRAINT]

DROP TABLE PILOTE ⇨

PILOTE

idPilote	Nom	Prenom	Tél
1	Dup…		06070809
2	Du…		07080910
3	Dupont	Marie	06106787

Contrainte PK_Pilote

Contrainte U_Pilote

Oracle : Une table de sauvegarde est mis en place lors de la suppression. Elle pourra être restaurée en cas de problème. L'option PURGE empêche cette sauvagarde : DROP TABLE pilote PURGE

DROP TABLE :
Option CASCADE

- Si une contrainte de référence pointe sur PILOTE, elle ne peut pas être supprimée.
- L'option CASCADE force la suppression. Elle entraine la suppression de la contrainte de référence

SYNTAXE SQL : DROP « nom de table » CASCADE

DROP TABLE PILOTE CASCADE

PILOTE

IdPilote	Nom	Prénom	Tél
1	Dup...		06070809
2	D...		07080910
3	Dupont	Marie	06106787

CONSTRAINT FK_PILOTE_AVION REFERENCES PILOTE (IdPilote)

AVION

IdAvion	IdPilote	Capacite	Type
1	5	230	A320
2	2	670	B747
3	1	800	A380

DROP TABLE :
Comment supprimer toutes les tables de mon compte ORACLE ?

- Connaitre les tables du compte :

SELECT * FROM TAB

```
TNAME                          TABTYPE  CLUSTERID
------------------------------ -------  ----------
ACHAT                          TABLE
AFFECTATION                    TABLE
ANIMATEUR                      TABLE
ARBITRE_SPORT                  TABLE
ARCHIVAGE_PRODUIT              TABLE
```

SOLUTION 1 : table par table

DROP TABLE ACHAT CASCADE CONSTRAINT PURGE ;

SOLUTION 2 : Génération des requêtes de suppression par SQL

SELECT 'DROP TABLE ' || TNAME || ' CASCADE CONSTRAINT PURGE;'
FROM TAB;

```
DROP TABLE ACHAT CASCADE CONSTRAINT PURGE;
DROP TABLE AFFECTATION CASCADE CONSTRAINT PURGE;
DROP TABLE ANIMATEUR CASCADE CONSTRAINT PURGE;
DROP TABLE ARBITRE_SPORT CASCADE CONSTRAINT PURGE;
DROP TABLE ARCHIVAGE_PRODUIT CASCADE CONSTRAINT PURGE;
```

Gestion des tables

Ecriture de scripts

Règles de bon usage

Jérôme TESSY
Fessy Mesaïla Florence

Création de Tables : règles de bon usage

Définition des tables :

- Toutes les tables de la base doivent avoir des noms différents
- Le nom des tables doivent être choisis avec précision
- Associer le plus de contraintes d'intégrité possible lors de sa création
- Définir une clé primaire et si besoin une clé logique

Définition des colonnes :

- Donner un nom **non ambigüe** à la colonne
- Choisir très **précisément** le type de la colonne
- Préciser la valeur par défaut de la colonne

ECRITURE DE SCRIPTS SQL de mise en exploitation
Où définir les contraintes ?

Pour créer des tables avec contraintes, on peut :

1. Mettre les contraintes dans le CREATE TABLE lors de la définition des colonnes :

les contraintes sont définies au niveau des colonnes

```
CREATE TABLE Pilote
(
IdPilote INTEGER  CONSTRAINT PK_Pilote PRIMARY KEY,
AgePilote INTEGER CONSTRAINT CK_AgePilote CHECK (AgePilote BETWEEN 18 AND 70),
)
```

Remarque : Une clé primaire composée ne peut pas être définie comme une contrainte de colonne

ECRITURE DE SCRIPTS SQL de mise en exploitation
Où définir les contraintes ?

Pour créer des tables avec contraintes, on peut :

2. Mettre les contraintes dans le CREATE TABLE en contraintes de table :

les contraintes sont définies au niveau de la table après la définition des colonnes

```
CREATE TABLE Pilote
(
IdPilote INTEGER,
AgePilote INTEGER,
CONSTRAINT PK_Pilote PRIMARY KEY(IdPilote),
CONSTRAINT CK_AgePilote CHECK (AgePilote BETWEEN 18 AND 70)
)
```

ECRITURE DE SCRIPTS SQL de mise en exploitation
Où définir les contraintes ?

③ **Pour créer des tables avec contraintes, on peut :**

Mettre les contraintes en dehors du CREATE TABLE :

```
CREATE TABLE Pilote
(
IdPilote INTEGER,
AgePilote INTEGER)
/
ALTER TABLE PILOTE
ADD CONSTRAINT PK_Pilote PRIMARY KEY(IdPilote)
ADD CONSTRAINT CK_AgePilote CHECK (AgePilote BETWEEN 18 AND 70)
/
```

Les contraintes ne sont pas définies dans le CREATE TABLE mais son ajoutées après par un ALTER TABLE

ECRITURE DE SCRIPT : EXEMPLE
Définition de contrainte de clé primaire

①
```
CREATE TABLE Pilote(
IdPilote INTEGER  CONSTRAINT PK_Pilote PRIMARY KEY,
NomPilote VARCHAR(30) )
```

Définition de la contrainte lors de la définition de la colonne

②
```
CREATE TABLE Pilote(
IdPilote INTEGER,
NomPilote VARCHAR(30),
CONSTRAINT PK_Pilote PRIMARY KEY(idPilote) )
```

Définition de la contrainte comme une contrainte de table

③
```
CREATE TABLE Pilote(
IdPilote INTEGER,
NomPilote VARCHAR(30) )
/
ALTER TABLE PILOTE ADD CONSTRAINT PK_Pilote PRIMARY KEY(idPilote)
```

Définition de la contrainte en dehors du CREATE TABLE

ECRITURE DE SCRIPT : EXEMPLE

Définition de contraintes de référence

```
CREATE TABLE AVION (
IdAvion INTEGER, ...,
IdPilote INTEGER CONSTRAINT FK_Pilote_Avion REFERENCES PILOTE( IdPilote) )
```

① Définition de la contrainte lors de la définition de la colonne

```
CREATE TABLE AVION (
IdAvion INTEGER, ...,
IdPilote INTEGER,
CONSTRAINT FK_Pilote_Avion FOREIGN KEY(idPilote) REFERENCES PILOTE( IdPilote) )
```

② Définition de la contrainte comme une contrainte de table

```
CREATE TABLE AVION (
IdAvion INTEGER, ...,
IdPilote INTEGER )
/
ALTER TABLE AVION
ADD CONSTRAINT FK_Pilote_Avion FOREIGN KEY(idPilote) REFERENCES PILOTE( IdPilote)
/
```

③ Définition de la contrainte en dehors du CREATE TABLE

ECRITURE DE SCRIPT : Bon usage d'écriture

Objectif : Ecrire un script homogène facile à lire et à maintenir

```
CREATE TABLE AVION
( IdAvion        INTEGER ,
  Libelle        VARCHAR(30) ,
  Capacite       INTEGER CONSTRAINT NN_Capacite  NOT NULL,
  TypeAvion      VARCHAR(10) ,
  IdPilote       INTEGER,
)
/
ALTER TABLE AVION ADD CONSTRAINT PK_Avion_IdAvion  PRIMARY KEY (IdAvion);
ALTER TABLE AVION ADD  CONSTRAINT CK_CapaciteSup0 CHECK (Capacite>0) ;
ALTER TABLE AVION ADD  CONSTRAINT FK_Avion_Pilote FOREIGN KEY (IdPIlote)
                                                  REFERENCES PILOTE( IdPilote);
ALTER TABLE AVION ADD  CONSTRAINT CU_Capacite UNIQUE (Libelle);
```

① Nomenclature de nommage des : TABLE; COLONNES; CONTRAINTES

② Création de la table sans contraintes

③ Ajout de l'ensemble des contraintes par table

SQL LMD

VALEUR NULL

VALEUR NULL

PILOTE NOT NULL NOT NULL

IdPilote	Nom	Prenom	Salaire
10	Dupont	Jean	32000
20	Durant	Pierre	25600
30	Dupont	Marie	NULL

UPDATE Pilote SET Salaire = NULL
WHERE Matricule = 30

La donnée n'est pas connue ou connue mais non renseignée

NULL n'est pas une "valeur" comme les autres :

- Elle est valable quel que soit le type de la colonne
- NULL ne correspond à aucune valeur précise. Elle précise que la valeur est inconnue « UNKNOWN »

NULL est UNKNOW

```
SELECT * FROM Pilote  WHERE  Salaire <> 32000;
```

PILOTE

IdPilote	Nom	Prenom	Salaire
10	Dupont	Jean	32000
20	Durant	Pierre	25600
30	Dupont	Marie	NULL

→ FALSE
→ TRUE
→ UNKNOW

20	Durant	Pierre	25600

IS NULL / IS NOT NULL

```
SELECT * FROM Pilote  WHERE  Salaire <> 32000 OR Salaire IS NULL;
```

PILOTE

IdPilote	Nom	Prenom	Salaire
10	Dupont	Jean	32000
20	Durant	Pierre	25600
30	Dupont	Marie	NULL

→ FALSE
→ TRUE
→ TRUE

20	Durant	Pierre	25600
30	Dupont	Marie	

IS NULL / IS NOT NULL

PILOTE

idPilote	Nom	Prenom	Salaire
10	Dupont	Jean	32000
20	Durant	Pierre	25600
30	Dupont	Marie	NULL

Ces expressions ne retournent pas TRUE

Ne retourne rien

SELECT * FROM Pilote WHERE Salaire = NULL;

SELECT * FROM Pilote WHERE Salaire != NULL;

SELECT * FROM Pilote WHERE NULL = NULL;

Retourne des valeurs

SELECT * FROM Pilote WHERE Salaire IS NULL;

30	Dupont	Marie	NULL

SELECT * FROM Pilote WHERE Salaire IS NOT NULL;

10	Dupont	Jean	32000
20	Durant	Pierre	25600

MISE A JOUR VALEUR NULL / COALESCE

PILOTE

idPilote	Nom	Prenom	Salaire
10	Dupont	Jean	32000
20	Durant	Pierre	25600
30	Dupont	Marie	NULL

NULL + 100 => NULL
UNKNOW + 100 => UNKNOW

On ajoute 100€ à tous les salaires

UPDATE PILOTE SET SALAIRE = SALAIRE + 100;

Salaire
32100
25700
NULL

UPDATE PILOTE SET SALAIRE = COALESCE(SALAIRE,0) + 100;

SELECT COALESCE(SALAIRE,0) FROM Pilote;

Valeur de salaire. Mais si salaire est NULL, valeur 0