

# SQL

S.SAIB

# Introduction

SQL : Structured Query Language :

- Il est capable accéder aux données de la base de données.
- C'est un langage adapté aux bases de données relationnelles.
- Il existe sur tous les SGBD relationnels (Oracle, Access,...),
- Le langage est une norme depuis 1986 qui s'enrichit au fil du temps.
- SQL peut s'interfacer avec tous les langages de troisième génération  
Comme C C++ Java, ...

# Introduction

SQL est un langage de:

- Définition des données
- Manipulation des données
- Contrôle des transactions

Ordres SQL	Aspect du langage
CREATE - ALTER - DROP - COMMENT - RENAME - TRUNCATE - GRANT - REVOKE	Définition des données (DDL : <i>Data Definition Language</i> ).
SELECT - INSERT - UPDATE - DELETE - MERGE - LOCK TABLE	Manipulation des données (DML : <i>Data Manipulation Language</i> ).
COMMIT - ROLLBACK - SAVEPOINT - SET TRANSACTION	Contrôle des transactions (TCL : <i>Transaction Control Statements</i> ).

# Définition de données

# Création de table

```
CREATE TABLE nomTable ( colonne1 type1 [DEFAULT valeur1]  
[NOT NULL], colonne2 type2 [DEFAULT valeur2] [NOT NULL]  
[CONSTRAINT nomContrainte1 typeContrainte1]... ) ;
```

nomTable : peut comporter au maximum 30 caractères (lettres, chiffres et caractères \_, \$, #. le nom est insensible à la casse et sera converti en majuscules dans le dictionnaire de données (il en va de même pour le nom des colonnes).

Colonne i type i : nom de colonne et son type (NUMBER, VARCHAR2, DATE...).

L'option DEFAULT fixe une valeur en cas de non-renseignement (NULL).  
L'option NOT NULL interdit que la valeur de la colonne ne soit pas renseignée.

nomContrainte i typeContrainte i : noms de la contrainte et son type (clé primaire, clé étrangère, etc.).

# Création de table

```
CREATE TABLE nomTable ( colonne1 type1 [DEFAULT valeur1]  
[NOT NULL], colonne2 type2 [DEFAULT valeur2] [NOT NULL]  
[CONSTRAINT nomContrainte1 typeContrainte1]...) ;
```

NB: Le marqueur NULL ne désigne pas une valeur mais une absence de valeur qu'on peut traduire comme non disponible, non affectée. NULL est différent d'une chaîne vide, d'un zéro ou des espaces.

# Création de table

---

## Rappel

La clé primaire (primary key) d'une table est l'ensemble minimal de colonnes qui permet d'identifier de manière unique chaque enregistrement.

Une clé étrangère (foreign key) référence à une clé primaire d'une autre table ... Une table peut contenir plusieurs clés étrangères ou aucune.

# Création de table

---


;  
--  
/\* ... \*/

symbole qui termine une instruction SQL d'Oracle  
Commentaire sur une ligne  
Commentaires sur plusieurs lignes



# Création de table

## Exemple de Création de table

 VOL\_JOUR

NUM_VOL	AERO_DEP	AERO_ARR	COMP	JOUR_VOL	NB_PASSAGERS
---------	----------	----------	------	----------	--------------

### Instruction SQL

```
CREATE TABLE vol_jour
(num_vol      VARCHAR2(6)  NOT NULL,
aero_dep      VARCHAR2(3)  NOT NULL,
aero_arr      VARCHAR2(3)  NOT NULL,
comp          VARCHAR2(4)  DEFAULT 'AF',
jour_vol      DATE          NOT NULL,
nb_passagers  NUMBER(3));
```

### Commentaires

La table contient six colonnes (quatre chaînes de caractères variables, une date et un entier relatif de trois chiffres).

La table inclut cinq contraintes en ligne.

- 4 NOT NULL qui imposent de renseigner quatre colonnes.
- 1 DEFAULT qui fixe un code compagnie à défaut d'être renseigné.

# Création de table

## Contraintes de colonnes

Les contraintes de colonnes ont pour but de programmer des règles de gestion au niveau des colonnes des tables.

**CONSTRAINT** *nomContrainte*

- UNIQUE (*colonne1* [,*colonne2*]...)
- PRIMARY KEY (*colonne1* [,*colonne2*]...)
- FOREIGN KEY (*colonne1* [,*colonne2*]...)  
REFERENCES [*schéma.*]*nomTablePere* (*colonne1* [,*colonne2*]...)  
[ON DELETE { CASCADE | SET NULL }]
- CHECK (*condition*)

# Création de table

Les contraintes de colonnes ont pour but de programmer des règles de gestion au niveau des colonnes des tables.

La contrainte UNIQUE impose une valeur distincte sur les colonnes concernées (les NULL font exception)

La contrainte PRIMARY KEY déclare la clé primaire, qui impose une valeur distincte sur les colonnes concernées (NOT NULL est aussi appliqué sur chaque colonne).

La contrainte FOREIGN KEY déclare une clé étrangère pour relier cette table à une autre table mère

La contrainte CHECK impose une condition simple ou complexe entre les colonnes de la table. Par exemple,

CHECK(nb\_passager>0) interdira toute valeur négative

CHECK(aero\_dep!=aero\_arr) interdira la saisie d'un trajet qui part et revient du même aéroport.

Il est possible de disposer de plusieurs contraintes UNIQUE mais seule une clé primaire est autorisée.

# Création de table

Tables	Contraintes
<pre>CREATE TABLE compagnie (comp          VARCHAR2(4), nom_comp      VARCHAR2(15), date_creation DATE <b>CONSTRAINT</b> nn_date_crea <b>NOT NULL</b>, <b>CONSTRAINT</b> pk_compagnie <b>PRIMARY KEY</b>(comp), <b>CONSTRAINT</b> un_nom_comp <b>UNIQUE</b>(nom_comp));</pre>	<p>Une contrainte en ligne et deux contraintes hors ligne.</p>
<pre>CREATE TABLE vol_jour (num_vol      VARCHAR2(6) <b>NOT NULL</b>, aero_dep     VARCHAR2(3) <b>CONSTRAINT</b> nn_depart <b>NOT NULL</b>, aero_arr     VARCHAR2(3) <b>CONSTRAINT</b> nn_arrivee <b>NOT NULL</b>, comp        VARCHAR2(4) <b>DEFAULT</b> 'AF', jour_vol     DATE <b>NOT NULL</b>, nb_passagers NUMBER(3), <b>CONSTRAINT</b> pk_vol_jour <b>PRIMARY KEY</b>(num_vol, jour_vol), <b>CONSTRAINT</b> fk_vol_jour_comp_compagnie <b>FOREIGN KEY</b>(comp) <b>REFERENCES</b> compagnie(comp), <b>CONSTRAINT</b> ck_nb_pax <b>CHECK</b> (nb_passagers&gt;0), <b>CONSTRAINT</b> ck_trajet <b>CHECK</b> (aero_dep != aero_arr));</pre>	<p>Une contrainte en ligne nommée (<b>NOT NULL</b>) et quatre contraintes hors ligne nommées :</p> <ul style="list-style-type: none"> <li>• Clé primaire.</li> <li>• <b>CHECK</b> (nombre d'heures de vol compris entre 0 et 20000).</li> <li>• <b>UNIQUE</b> (homonymes interdits).</li> <li>• Clé étrangère.</li> </ul>

Si vous ne nommez pas une de vos contraintes, un nom sera généré  
 Préfixez par **pk\_** le nom d'une contrainte clé primaire, **fk** une clé étrangère.  
 (exemple **pk\_Pilote**).

- Pour une contrainte clé étrangère, renseignez (ou abrégez) les noms de la table source, de la clé, et de la table cible (exemple **fk\_Pil\_compa\_Comp**).

# Création de table

## Types des colonnes

Pour décrire les colonnes d'une table, Oracle fournit les types prédéfinis suivants

- caractères (CHAR, NCHAR, VARCHAR2, NVARCHAR2, CLOB, NCLOB, LONG).

VARCHAR2 ( 20) : chaîne de caractère de 20 caractères

- valeurs numériques NUMBER.

NUMBER(5, 2): 5 chiffres avant la virgule et 2 après

- date/heure (DATE, INTERVAL DAY TO SECOND, INTERVAL YEAR TO MONTH, TIMESTAMP, TIMESTAMP WITH TIME ZONE, TIMESTAMP WITH LOCAL TIME ZONE)

Format des dates jj/mm/aaaa

# Création de table

DESC (raccourci de DESCRIBE) est une commande qui permet d'extraire la structure brute d'une table.

```
SQL> desc vol_jour
```

Nom	NULL ?	Type
NUM_UOL	NOT NULL	VARCHAR2(6)
AERO_DEP	NOT NULL	VARCHAR2(3)
AERO_ARR	NOT NULL	VARCHAR2(3)
COMP		VARCHAR2(4)
JOUR_UOL	NOT NULL	DATE
NB_PASSENGERS		NUMBER(3)

# Suppression de table

---

**DROP TABLE** nomTable

# **Manipulation de données**



## Manipulation des données

- L'insertion d'enregistrements : INSERT
- La modification de données : UPDATE
- La suppression d'enregistrements : DELETE

# Insertion de données

**INSERT INTO** nomTable **VALUES** (valeur1 | DEFAULT, valeur2 | DEFAULT...);

**INSERT INTO** nomTable( colonne1, colonne2, ...) **VALUES** (valeur1, valeur2, ...);

Instructions SQL	Commentaires
<pre>INSERT INTO compagnie VALUES ('SING', 'Singapore AL', TO_ DATE('19470101', 'YYYYMMDD'));</pre>	Les valeurs sont renseignées dans l'ordre de la structure de la table.
<pre>INSERT INTO compagnie (nom_comp, comp, date_creation) VALUES ('Air France', 'AF', TO_DATE('19330101', 'YYYYMMDD'));</pre>	Les valeurs sont renseignées dans l'ordre de la liste.

Si des valeurs ne respectent pas des contraintes alors les messages renvoyés pour chaque erreur font apparaître le nom de la contrainte.

# Modification des valeurs de colonnes

L'instruction **UPDATE** permet la mise à jour des colonnes d'une table.

```
UPDATE nomTable SET colonne1 = expression WHERE condition  
;
```

Exemple : Modifions la compagnie de code 'AN1' en affectant la valeur 50 à la colonne nrue.

```
UPDATE Compagnie SET nrue = 50 WHERE comp = 'AN1';  
;
```

# Suppression des valeurs de colonnes

L'instruction DELETE permet de supprimer un ou plusieurs enregistrements d'une table.

```
DELETE [FROM] nomTable [WHERE (condition)] ;
```

```
DELETE FROM Pilote WHERE compa = 'AF';  
DELETE Compagnie WHERE comp = 'AF';
```

La première commande détruit tous les pilotes de la compagnie de code 'AF'.  
La deuxième, détruit la compagnie de code 'AF'.

La condition sélectionne les lignes à supprimer dans la table. Si aucune condition n'est précisée, toutes les lignes seront supprimées. Si la condition ne sélectionne aucune ligne, aucun enregistrement ne sera supprimé.

# Renommer une table

---

```
RENAME ancienNom TO nouveauNom;
```

L'instruction RENAME renomme une table.

# Modification structurelle (ALTER TABLE)

## Ajout de colonnes:

La directive ADD de l'instruction ALTER TABLE permet d'ajouter une nouvelle colonne à une table. Cette colonne est initialisée à NULL pour tous les enregistrements (à moins de spécifier une contrainte DEFAULT, auquel cas tous les enregistrements de la table sont mis à jour avec une valeur non nulle).

```
ALTER TABLE Pilote ADD (nbHVol NUMBER(7,2));  
ALTER TABLE Pilote ADD (compa VARCHAR2(4) DEFAULT 'AF',  
                           ville VARCHAR2(30) DEFAULT 'Paris' NOT NULL);
```

# Modification structurelle (ALTER TABLE)

## Modifier le type des colonnes

La directive MODIFY de l'instruction ALTER TABLE modifie le type d'une colonne existante.

Il est possible d'augmenter la taille d'une colonne numérique (largeur ou précision) – ou d'une chaîne de caractères (CHAR et VARCHAR2) – ou de la diminuer si toutes les données présentes dans la colonne peuvent s'adapter à la nouvelle taille. Les contraintes en ligne peuvent être aussi modifiées par cette instruction (DEFAULT, NOT NULL, UNIQUE, PRIMARY KEY et FOREIGN KEY). Une fois la colonne changée, les nouvelles contraintes s'appliqueront aux mises à jour ultérieures de la base

```
ALTER TABLE Pilote MODIFY compa VARCHAR2(6) DEFAULT 'SING';
```

-- Augmente la taille de la colonne compa et change la contrainte de valeur par défaut.

# Modifications structurelles (ALTER TABLE)

## Supprimer des colonnes

La directive DROP COLUMN de l'instruction ALTER TABLE permet de supprimer une colonne.

Il n'est pas possible de supprimer avec cette instruction :

- Des clés primaires référencées par des clés étrangères.
- Toutes les colonnes d'une table.

```
ALTER TABLE Pilote DROP COLUMN adresse;
```



# Modification structurelle (ALTER TABLE)

## Ajouter des colonnes

ALTER TABLE ADD permet d'ajouter une colonne à la table

```
ALTER TABLE Avion ADD heurePax NUMBER(10,2) AS (nbhVol/age)  
CHECK (heurePax BETWEEN 0 AND 2000) NOT NULL;
```

-- permet d'ajouter s à la table Avion la colonne virtuelle qui détermine le ratio du nombre d'heures de vol par passager en y ajoutant deux contraintes .

# Interrogation des données

# Interrogation de données

---

- Toute interrogation se fait par la commande « SELECT »
- Commande déclarative : décrit ce que l'on cherche sans décrire le moyen de le réaliser

# Interrogation de données

## Forme complète de la commande SELECT

- l'instruction SELECT doit inclure :
  - 1/ une clause SELECT : qui détermine les colonnes à afficher
  - 2/ une clause FROM : détermine la table contenant les colonnes
  - 3/ une clause WHERE : détermine la condition

```
SELECT [DISTINCT]  
FROM liste de (nom de table [[AS] nom]) | (requête AS nom)  
WHERE condition  
[GROUP BY liste de nom de colonne]  
[HAVING condition]  
[ORDER BY liste de ((nom de colonne | rang de colonne) (ASC  
DESC))];
```

# Interrogation de données

```
SELECT [ { DISTINCT | UNIQUE } | ALL ] listeColonnes  
FROM nomTable ;  
[ clauseOrdonnancement ] ;
```

- DISTINCT et UNIQUE jouent le même rôle : ne pas prendre en compte les duplicatas.
- ALL : prend en compte les duplicatas (option par défaut).
- ListeColonnes : peut être remplacé par :
  - { \* | expression1 [[AS] alias1 ] [, expression2 [[AS] alias2 ]...},
  - \* : extrait toutes les colonnes de la table
  - **expression** : nom de colonne, fonction, constante ou calcul.
  - **alias** : renomme l'expression (nom valable pendant la durée de la requête).
- FROM : désigne la table (qui porte un alias ou non) à interroger
- ClauseOrdonnancement : tri sur une ou plusieurs colonnes ou expressions.

# Ordonnancement

Les clauses d'ordonnancement permettent de trier le résultat d'une requête.

Elles précisent également l'ordre dans lequel la liste des lignes sélectionnées sera donnée.

```
ORDER BY { expression1 | position1 | alias1 } [ASC | DESC] [NULLS FIRST | NULLS LAST] [, {expression2 | position2 | alias2} [ASC | DESC] [NULLS FIRST | NULLS LAST]]...
```

- expression : nom de colonne, fonction, constante, calcul.
- position : entier qui désigne l'expression (au lieu de la nommer) dans son ordre d'apparition dans la clause SELECT.
- ASC ou DESC : tri ascendant ou descendant (par défaut ASC).
- NULLS FIRST ou NULLS LAST : position des valeurs nulles (au début ou à la fin du résultat). NULLS LAST par défaut pour l'option ASC, NULLS FIRST par défaut pour l'option DESC.

```
SELECT brevet, nom FROM Pilote ORDER BY nom;
```

```
SELECT brevet,nbHVol FROM Pilote ORDER BY nbHvol ASC NULLS FIRST;
```

# La Clause WHERE

Les éléments de la clause WHERE d'une requête permettent de programmer l'opérateur de restriction. Cette clause limite la recherche aux enregistrements qui respectent une condition simple ou complexe.

```
SELECT [ { DISTINCT | UNIQUE } | ALL ] listeColonnes  
FROM nomTable  
[ WHERE condition ] ;
```

condition : est composée de colonnes, d'expressions, de constantes liées deux à deux entre des opérateurs :

- de comparaison (>, =, =, <=, <>) ;
- logiques (NOT, AND ou OR) ;
- intégrés (BETWEEN, IN, LIKE, IS NULL).

# La Clause WHERE

---

Notez l'utilisation de simples guillemets pour comparer des chaînes de caractères.



# La Clause WHERE

## L'opérateur BETWEEN

BETWEEN limiteInf AND limiteSup teste l'appartenance à un intervalle de valeurs.

```
SELECT brevet, nom, nbHVol  
FROM Pilote  
WHERE nbHVol BETWEEN 399 AND 1000 ;
```

# La Clause WHERE

## L'opérateur IN

IN (listeValeurs) compare une expression avec une liste de valeurs.

```
SELECT brevet, nom, compa  
FROM Pilote  
WHERE compa IN ('CAST', 'SING');
```

# La Clause WHERE

## L'opérateur LIKE

```
SELECT brevet, nom, compa  
FROM Pilote  
WHERE compa LIKE('"%A%"') ;
```

LIKE (expression) compare de manière générique des chaînes de caractères à une expression.

- Le symbole % remplace un ou plusieurs caractères.
- Le symbole \_ remplace un caractère.

Ces symboles peuvent se combiner.

# La Clause WHERE

## Les opérateurs NULL et IS NOT NULL

```
SELECT nom, prime, nbHVol, compa  
FROM Pilote  
WHERE prime IS NULL OR nbHVol IS NULL ;
```

IS NULL compare une expression (colonne, calcul, constante) à la valeur NULL.

La négation s'écrit soit « expression IS NOT NULL » soit « NOT (expression IS NULL) ».

# Les fonctions

Oracle propose un grand nombre de fonctions qui s'appliquent dans les clauses SELECT ou WHERE d'une requête. La syntaxe générale d'une fonction est la suivante :

```
nomFonction(colonne1 | expression1 [,colonne2 | expression2 ...]);
```

## Exemples de fonctions

- MOD(m,n) : Division entière de m par n.
- POWER(m,n) : m puissance n.
- ROUND(m,n) : Arrondi à une ou plusieurs décimales. donne 17,57.
- SIGN(n) : Retourne le signe d'un nombre.

# Les fonctions

Les principales fonctions pour les dates sont:

ADD_MONTHS	Ajoute des mois à une date
SYSDATE:	Date courante du système
MONTHS_BETWEEN(d1,d2)	Retourne le nombre de mois entre deux dates (d1 et d2 avec d1>d2).
LAST_DAY(d):	Retourne le dernier jour du mois d

Exemple: Rendez-vous dans 4 mois.

-- Ajoute 4 mois à la date en cours

```
SELECT ADD_MONTHS(SYSDATE, 4) AS "RDV" FROM DUAL;
```

-- Numéro du mois d'il y a 65 jours

```
SELECT EXTRACT(MONTH FROM (SYSDATE-65) "Mois" FROM DUAL;
```

# Regroupement

Il s'agit des regroupements de lignes

```
SELECT [ { DISTINCT | UNIQUE } | ALL ] listeColonnes  
FROM nomTable  
[ WHERE condition ]  
[ clause Regroupement ]  
[ HAVING Condition ]  
[ clause Ordonnancement ] ;
```

listeColonnes : peut inclure des expressions (présentes dans la clause de regroupement) ou des fonctions de groupe.

- clause Regroupement : **GROUP BY** (expression1[,expression2]...) permet de regrouper des lignes selon la valeur des expressions (colonnes, fonction, constante, calcul).
- **HAVING** condition : pour inclure ou exclure des lignes aux groupes (la condition ne peut faire intervenir que des expressions du **GROUP BY**).

# Fonctions de groupe

AVG([DISTINCT   ALL] expr)	Moyenne de expr (nombre).
COUNT({ *   [DISTINCT   ALL] expr })	Nombre de lignes (* toutes les lignes, expr pour les colonnes non nulles).
MAX([DISTINCT   ALL] expr)	Maximum de expr (nombre, date, chaîne)
MIN([DISTINCT   ALL] expr)	Minimum de expr (nombre, date, chaîne)
SUM([DISTINCT   ALL] expr)	Somme de expr (nombre).

L'option DISTINCT évite les duplicatas alors que ALL les prend en compte (par défaut).

À l'exception de COUNT, toutes les fonctions ignorent les valeurs NULL (il faudra utiliser NVL pour contrer cet effet).

Utilisées sans GROUP BY, ces fonctions s'appliquent à la totalité ou à une seule partie d'une table