



LE LANGAGE SQL

10/12/2020

SOMMAIRE

Objectifs	9
Une donnée	10
Un ensemble de données	11
Une Base de données	12
Base de données	13
SGBD	14
Base de données relationnelle	15
Objectifs	20
Langage SQL	21
Définition d'une table	35
Chargement d'une table : INSERT	41
ORDRE INSERT	43
Chargement d'une table : INSERT	46
SELECT	49
SELECT - exemples	50
SELECT *	51
SELECT avec expression	52
SELECT avec colonnes	53
SELECT avec valeurs calculées	54
Nommer une expression	55
L'option DISTINCT	56
Sélection conditionnelle avec WHERE	57
Sélection conditionnelles avec WHERE	58
SELECT avec prédicat de base	59
SELECT avec condition de recherche	60
L'option GROUP BY	61
Exemple récapitulatif	66
Objectifs	68

Table AUTEURS	69
SELECT et SOUS-SELECT	70
Option UNION.....	72
Les prédicats	74
Le prédicat between	75
Le prédicat NULL.....	76
Le prédicat LIKE	77
Le prédicat EXISTS.....	79
Le prédicat IN	80
Les prédicats quantifiés.....	82
Les fonctions sur colonnes.....	83
La fonction AVG	84
La fonction SUM	85
Les fonctions MAX et MIN.....	86
La fonction COUNT	87
La fonction COUNT	88
Les fonctions scalaires	89
la fonction DECIMAL	91
La fonction SUBSTR.....	92
Les fonctions VALUE et COALESCE.....	93
L'expression CASE	94
La fonction NULLIF	97
La fonction STRIP	98
Imbrication de fonctions	99
Expression de durée	100

Durées	101
Opérations sur données temporelles	102
Additions.....	103
Fonctions scalaires liées au temps	104
La fonction CHAR.....	105
La fonction TIMESTAMP	106
Objectifs du chapitre	108
Table ouvrages	109
INSERT avec sous select.....	110
INSERT : exemples	111
UPDATE.....	113
UPDATE : exemples.....	114
UPDATE : Remarques.....	115
DELETE	116
DELETE : exemples	117
La première forme normale et ses inconvénients	118
Deuxième forme normale	124
Anomalies de mise à jour.....	128
Troisième forme normale et BCNF	129
Les index	131
Les index.....	132
CREATE INDEX.....	133
Les index	136
Rôle et utilisation des index	140
Objectifs.....	142
Exemples de tables.....	143

SOUS Select.....	145
Option FROM avec plusieurs tables	146
Nom de corrélation	148
Sous-requêtes corréliées.....	149
Notion de jointure	153
Produit cartésien.....	154
Equi-jointure	155
Jointure de plusieurs tables	156
Jointure naturelle.....	157
Jointure avec conditions quelconques.....	158
Jointure d’une table sur elle-même	159
Jointure externe	160
Jointure externe	161
Select dans la clause FROM	162
Exemple de synthèse.....	164
Exemple de synthèse.....	165
Intégrité d’entité et de référence	166
Clé primaire et clé étrangère	167
CREATE TABLE avec clé primaire et contrainte de référence	169
Intégrité de référence	173
CREATE TABLE avec contrainte de vérification.....	179

Bases de données

OBJECTIFS

Comprendre les notions relatives aux bases de données relationnelles et notamment les définitions des termes :

- donnée
- base de données

SGBD

- table relationnelle
- base relationnelle

UNE DONNÉE

L'objet de l'informatique, c'est le traitement des données

Qu'est-ce qu'une donnée ?

Deux définitions :

1. Une donnée est la mesure de la réalité physique à laquelle elle se rapporte.
2. Une donnée est une chaîne de caractères à laquelle est attachée une signification.

Définition

- On ne traite pas ces données individuellement, mais en les intégrant dans des ensembles de données.
- On appelle modèle logique la représentation des rapports entre la réalité et les ensembles de données qui y sont associés, ainsi que les liaisons qui existent entre ces ensembles de données.

Définitions

"Ensemble exhaustif, non redondant et structuré des données, fiables et cohérentes, organisées indépendamment de leurs applications, accessibles en temps utile, facilement exploitables et satisfaisant à des normes de confidentialité".

Objectifs

- Description des données (LDD)
- Manipulation des données (LMD)
- Facilité d'accès
- Simultanéité
- Confidentialité
- Sécurité
- Disponibilité
- Évolutivité
- Partageabilité

Définition

Un Système de Gestion de Bases de Données (SGBD) est un ensemble de programmes utilitaires, permettant de décrire et de ranger des ensembles de données, utilisables simultanément, à n'importe quel moment, par de nombreux utilisateurs.

Il garantit la sécurité et l'intégrité des données, et doit faciliter l'évolution des applications existantes.

Il existe trois types de SGBD :

- Hiérarchique
- Réseau
- Relationnel

BASE DE DONNÉES RELATIONNELLE

On peut utiliser DB2 sur différentes plates-formes matérielles, l'ensemble de ces SGBD constituant la famille DB2

DB2 Universal Database version 6.1

DB2 Universal Database version 7

DB2 Universal Database version 8 (Septembre 2003)

DB2 for AIX*

DB2 for HP-UX*

DB2 for Linux (beta) DB2 for OS/2*

DB2 for SCO UnixWare

DB2 for Sun Solaris*

DB2 for Windows NT*, Windows 95 et 98*

DB2 Enterprise-Extended Edition

DB2 Universal Database for AS/400

DB2 for VSE & VM (remplace SQL/DS)

DB2 Universal Database for OS/390 (remplace DB2 for MVS/ESA)

Il existe d'autres produits relationnels non IBM : Oracle, Sybase, Informix, SQL Server ...

Table relationnelle

Expression d'une relation entre des données.

Une table est constituée :

- d'un nombre déterminé de colonnes et d'un nombre variable de lignes
- L'intersection d'une ligne et d'une colonne est une valeur ou l'ensemble vide (NULL : absence de valeur).
- L'ordre des lignes n'est pas significatif.
- L'ordre des colonnes n'est pas significatif sauf si une colonne est définie par sa position relative.
- Même nature des données pour tous les éléments d'une colonne

Base de donnée relationnelle

C'est un **ensemble de données structuré sous forme de tables**.

La forme tabulaire des données ne laisse pas transparaître :

- ⇒ la manière dont est stockée l'information
- ⇒ les techniques d'accès
- ⇒ les liens entre les tables
- ⇒ L'information est sous une forme simple, facile à décrire et à manipuler.

BASE DE DONNÉE RELATIONNELLE

T

Table relationnelle portant sur les ouvrages d'une bibliothèque :

Exemple

Table livres

AUTEUR	TITRE	ANNEE	GENRE	PRIX
HUGO	HERNANI	1830	THEATRE	120.00
HUGO	LES CONTEMPLATIONS	1856	POESIE	78.50
HUGO	LESMISERABLES	1862	ROMAN	148.50
BALZAC	EUGENIE GRANDET	1833	ROMAN	100.00
BALZAC	LE PERE GORIOT	1834	ROMAN	-----
DUMAS	LES TROIS MOUSQUETAIRES	1844	ROMAN	80.00
DUMAS	VINGT ANS APRES	1845	ROMAN	80.00
DUMAS	LA DAME AUX CAMELIAS	1852	THEATRE	110.00
STENDHAL	LE ROUGE ET LE NOIR	1831	ROMAN	98.50
STENDHAL	LA CHARTREUSE DE PARME	1839	ROMAN	110.50
FLAUBERT	MADAME BOVARY	1857	ROMAN	99.50
VERLAINE	POEMES SATURNIENS	1866	POESIE	90.50

- Chaque colonne représente une des propriétés (ou attribut) d'un livre. Elle est définie sur un domaine, contenant un ensemble de valeurs.
- Chaque ligne contient des données relatives à l'un des livres de cette bibliothèque.

3. Définition d'une table relationnelle

OBJECTIFS

- Connaître la syntaxe du langage SQL et son utilisation pour créer et charger une table.
- Caractères et éléments du langage SQL
- Données gérées par SQL
- Définition d'une table : CREATE TABLE
- Chargement d'une table : INSERT

LANGAGE SQL

SQL (**S**tructured **Q**uery **L**anguage) est utilisé pour :

- La définition des données (LDD)
- La manipulation des données (LMD)
- C'est un langage normalisé commun aux différents SGBD
- Relationnels : DB2, ORACLE ...
- La norme en vigueur est SQL92

DDL : data definition language

DML : data manipulation language

Caractères et éléments du langage

Caractères :

- les majuscules de A à Z (les minuscules sont remplacées par les majuscules correspondantes sauf pour les littéraux)
- les chiffres de 0 à 9
- les caractères spéciaux (#, @ et \$) et le caractère blanc souligné (_)

Éléments du langage :

Suite de caractères symbolisant des constantes, des mots-clés, des identificateurs

Identificateurs, Deux types :

- ⇒ identificateur ordinaire : composé de lettres, chiffres et blanc souligné (_), commence par une lettre
- ⇒ identificateur délimité : séquence de caractères entre un couple de délimiteurs (" ou ' selon les options)

Deux longueurs :

- ⇒ identificateur court (ou nom court) : 8 caractères au plus (pas de caractères spéciaux)
- ⇒ identificateur long (ou nom long) : jusqu'à 18 caractères identificateurs qualifiés :
- ⇒ suite d'**identificateurs** séparés par des points (.).

EXEMPLES

```
NOM NOM_DE_TABLE 'NOM DE TABLE'
USER1.NOM_DE_TABLE
NOM_DE_TABLE.NOM_DE_COLONNE
```

Données gérées par SQL

SQL permet de gérer différents types de données :
Numériques

1. Entiers
2. Décimaux
3. Nombres en virgule flottante

- Données non numériques
- Données temporelles

Données gérées par SQL

Données numériques

Entiers

- Entiers longs (**INTEGER** ou **INT**)
 - mot binaire de 31 bits plus le signe
 - valeur entre -2147483648 et +2147483647
- Entiers courts (**SMALLINT**)
 - demi mot binaire de 15 bits plus le signe
 - valeur entre -32768 et +32767

Données gérées par SQL

Données numériques Décimaux

DECIMAL(p,e) ou NUMERIC(p,e)

- p : nombre de chiffres significatifs ($p < 32$)
- e : nombre de chiffres après la marque décimale ($e < p$)
+ DECIMAL(11,2) : nombre de 11 chiffres dont 2 après la virgule
- Format condensé (deux chiffres par octet)
- Valeur entre $10^{31}-1$ et $1-10^{31}$
- Défaut : DECIMAL (ou DEC ou NUMERIC) équivaut à :
DECIMAL(5,0)
- L'échelle peut être omise : DECIMAL(10) équivaut à
DECIMAL(10,0)

Données gérées par SQL

Données numériques

Nombres en virgule flottante

- **Virgule flottante simple précision**

REAL ou FLOAT(n) avec $n < 22$ /

□ occupe 32 bits valeur comprise entre $5.4E-79$ et $7.2E+75$
(soit entre $5.4 \cdot (10^{-79})$ et $7.2 \cdot (10^{+75})$)

- **Virgule flottante double précision**

- DOUBLE PRECISION ou FLOAT(n) avec $21 < n < 54$ occupe 64 bits
valeur comprise entre $5.4E-79$ et $7.2E+75$.

**EN SIMPLE COMME EN DOUBLE PRECISION LA GRANDEUR DE LA VALEUR
POUVANT ETRE STOCKEE
EST STRICTEMENT LA MEME . POURQUOI ?**

Parce que c'est la précision qui change :
en simple précision au maximum 21 chiffres forment la mantisse tandis que
en double précision 54 chiffres forment la mantisse.

- 1,22 chiffres après la virgule en simple précision
- 1,54 chiffres après la virgule en double précision

Données gérées par SQL

Données non numériques

Chaînes de caractères

- ☐ Chaînes de caractères de longueur fixe
 - ☐ CHARACTER(n) ou CHAR(n) ☐ avec $n < 255$
- ☐ CHARACTER (ou CHAR) équivalent à CHARACTER(1)
- ☐ Chaînes de caractères de longueur variable
- ☐ VARCHAR(n) ☐ avec n inférieur à la longueur maximum autorisée (dépendant de la longueur d'une page physique)
- ☐ LONG VARCHAR : chaîne de caractères variable de longueur maximum.

Long Varchar : *Il s'agit d'une colonne qui utilise toute la place disponible dans la page...
C'est un moyen qui était utilisé autrefois pour n'avoir qu'une seule ligne par page.
Technique obsolète.*

Données gérées par SQL

Données non numériques

Chaînes de caractères graphiques

un caractère occupe deux octets (16 bits)
principalement utilisées pour ranger les
caractères orientaux (caractères japonais
Kanji par exemple)

soit de longueur fixe : GRAPHIC(n), (n<128)
soit de longueur variable : VARCHAR(n), n
inférieur à la plus grande valeur permise par
le stockage physique.

LONG VARCHAR : chaîne de caractères
graphiques de la plus grande longueur
possible.

Données gérées par SQL

Données temporelles

- DATE : date (an, mois, jour)
- □ huit chiffres significatifs (AAAAMMJJ) □ occupe 4 octets en format interne condensé (2 pour l'année, 1 pour le mois, 1 pour le jour)
- TIME : heure (heure, minutes, secondes) □ six chiffres significatifs (HHMMSS) □ occupe 3 octets en format interne condensé (1 pour l'heure, 1 pour les minutes, 1 pour les secondes)
- TIMESTAMP : instant (date, heure, microsecondes) □ vingt chiffres significatifs (AAAAMMJJHHMMSSmmmmmm)
- □ occupe 10 octets en format interne condensé

Données gérées par SQL

Constantes entières

Ex: 1 +1234 -234 1234567890

Constantes flottantes

Ex: 10E1 2.2E-2 123.345E-23

Constantes décimales

Ex: 1 +12.34 -234. +1234567890.12345

Constantes chaînes de caractères forme caractères

Ex: 'ABCD' '123456'

Forme hexadécimale

Ex: X'00123c' X'C1C2C3'

Constantes graphiques

Ex: G'<chaîne graphique>'

LANGUAGE SQL

Une constante temporelle doit être incluse entre deux délimiteurs (' ou ") et correspondre à l'un des formats prédéfinis.

Données temporelles

Constantes date et heure

FORMAT	DATE		HEURE
ISO	SSAA-MM-JJ	HH.MM.SS	
USA	MM/JJ/AAAA	HH :MM AM ou PM	
EUR	JJ.MM.AAAA	HH.MM.SS	
JIS		AAAA-MM-JJ	HH :MM :SS
LOCAL	selon installation		

Exemple : '2009-10-20' '12.22.30'

Constante **TIMESTAMP**

Un seul format : AAAA-MM-JJ-HH.MM.SS.mmmmmm





Exemple : '1988-1-1-6.07.25.123456'

Les constantes instants sont souvent appelés **TIMESTAMP**.

Jis : Japanese industrial standard

Notations

Pour décrire la syntaxe des ordres SQL, nous utiliserons les conventions suivantes :

- début d'une instruction : 
- fin d'une instruction : 
- l'instruction se poursuit : 
- suite de l'instruction : 

Mots en majuscules = mots clés

- Mots en minuscules = variables créées par l'utilisateur
- Symboles (.,() etc) = font partie de l'instruction

Délimiteurs d'ordres SQL

Selon le contexte dans lequel on se situe, différents délimiteurs doivent être utilisés :
caractère ";" pour séparer 2 ordres SQL dans l'environnement interactif.

EXEC SQL et **END-EXEC**

encadrant un ordre SQL dans les programmes COBOL.

DÉFINITION D'UNE TABLE

CREATE TABLE

La définition d'une table (noms des colonnes, type des données ...) se fait par l'ordre :

```
>>__CREATE TABLE__table-name____>
|
|<_,'____column-definition_____|_)>
|
|_|_unique-constraint_____| | |
|_|_referential-constraint_____|
|_|_check-constraint_____|
|_|_LIKE____table-name_____|
|_|_view-name_____|_|_COLUMN ATTRIBUTES_____|
|_|_INCLUDING IDENTITY_____|_|_|
|
|<_____(1)____>>
|
|_|_IN____table-space-name_____| | |
|_|_|_database-name_____|_|
|_|_IN DATABASE____database-name_____|_|
|_|_EDITPROC____program-name_____|_|
|_|_VALIDPROC____program-name_____|_|
|_|_NONE_____|_|
|_|_AUDIT_____|_CHANGES_____|_|
|_|_|_ALL_____|_|
|_|_OBID____integer_____|_|
|_|_|_NONE_____|_|
|_|_DATA CAPTURE_____|_CHANGES_____|_|
|_|_WITH RESTRICT ON DROP_____|_|
|_|_CCSID____ASCII_____|_|
|_|_|_EBCDIC_____|_|
|_|_|_UNICODE_____|_|
```

Note:
(1) The same clause must not be specified more than once.

DÉFINITION D'UNE TABLE

Description des paramètres

Nom de table

Nom long, identifiant unique de la table pour un utilisateur. Une table appartient à un utilisateur identifié par son idautorisation et à un système dans le cas de systèmes distribués. Un nom de table peut être qualifié :
par l'identifiant de son créateur

➡ USERX.TABLE_EMPLOYE

par l'identifiant de son créateur et le nom du système où réside la table

➡ PARIS.USERX.TABLE_EMPLOYE

Par défaut, le nom de table est qualifié par le système et l'utilisateur courants.

DÉFINITION D'UNE TABLE

Description des paramètres

Nom de colonne

- Nom long, identifiant de manière unique la colonne dans la table.
- Nombre de colonnes d'une table limité (750 au maximum pour DB2 MVS/ESA).

Type de donnée

- INTEGER, DECIMAL, CHAR ...

DÉFINITION D'UNE TABLE

Description des paramètres

Option **NOT NULL**

- ⇒ **NOT NULL** empêche l'introduction de valeurs nulles :
émission d'un message d'erreur si tentative d'insertion de nuls

- ⇒ **NOT NULL WITH DEFAULT** remplace une absence de valeur par une valeur par défaut dépendant du type de donnée défini pour la colonne :
 - ⇒ zéro pour les données numériques.
 - ⇒ blanc pour les chaînes de caractères de longueur fixe.
 - ⇒ chaîne de longueur zéro pour les chaînes de caractères de longueur variable.

- ⇒ date, heure ou instant courant pour les données de type date, heure ou instant.

DÉFINITION D'UNE TABLE

Description des paramètres

Option **WITH DEFAULT**

⇒ Il est possible de définir pour une colonne une valeur par défaut autre que celle correspondant naturellement à son type.

Cette valeur peut s'exprimer par :

⇒ Une constante d'un type compatible avec celui de la colonne

⇒ Les mots-clés USER, CURRENT SQLID, NULL

⇒ Omettre NOT NULL et DEFAULT est équivalent à DEFAULT NULL

DÉFINITION D'UNE TABLE

Exemple

```
CREATE TABLE LIVRES
(AUTEUR CHARACTER(8) NOT NULL,
 TITRE CHARACTER(24) NOT NULL,
 ANNEE SMALLINT,
 GENRE CHAR(8) NOT NULL
 WITH DEFAULT 'DIVERS',
 PRIX DECIMAL(5, 2)
 )
```

Lors du chargement de cette table, les valeurs des colonnes AUTEUR et TITRE doivent impérativement être renseignées, d'autre part la colonne genre, si elle n'est pas spécifiée prendra la valeur « DIVERS ».
Les autres colonnes peuvent prendre la valeur NULL.

CHARGEMENT D'UNE TABLE : INSERT

INSERT

```
>> __INSERT INTO __table-name__>
      |__view-name__| |<_,__column-name_|_|
      |__column-name_|_|
>
|__OVERRIDING USER VALUE_|
> __VALUES__expression__><
| |__DEFAULT__| |
| |__NULL__|
| |<_,__expression_|_|
| |__DEFAULT__|
| |__NULL__|
|__fullselect__
|__WITH__RR__| |__QUERYNO__integer_|
|__RS_|
|__CS_|
```

CHARGEMENT D'UNE TABLE : INSERT

Description des paramètres

Nom de table

Qualifié ou non qualifié.

Nom de colonne

Ordre indifférent.

On peut omettre des colonnes (sauf si NOT NULL).

Défaut : toutes les colonnes dans l'ordre de définition.

VALUES

Correspondance positionnelle.

⇒ constante

⇒ NULL

Description des paramètres (suite)

⇒ registre spécial : USER

- CURRENT DATE
- CURRENT SQLID
- CURRENT SERVER
- CURRENT TIME
- CURRENT TIMESTAMP
- CURRENT TIMEZONE
- CURRENT SERVER
- CURRENT DEGREE
- CURRENT PACKAGESET
- CURRENT RULES

Règles

L'instruction **INSERT** permet de valoriser les colonnes d'une ligne de table.

Des contrôles de validité sont effectués :

- ⇒ si l'ensemble des valeurs correspond à la nature des données, la nouvelle ligne de table est insérée,
- ⇒ si une erreur est détectée, la ligne n'est pas insérée.

A chaque exécution d'une requête est associée un code retour SQL informant l'utilisateur sur le déroulement de sa requête.

ORDRE INSERT

Exemple

```
INSERT INTO LIVRES
VALUES ('HUGO', 'HERNANI',
       1830, 'THEATRE', 120.00)
```

```
INSERT INTO LIVRES
(AUTEUR, TITRE, ANNEE, GENRE)
VALUES ('BALZAC', 'LE PERE GORIOT',
       1834, 'ROMAN')
```

La spécification des colonnes est facultative si toutes les valeurs des colonnes sont fournies.

Attention dans ce cas à vérifier la cohérence avec l'ordre de description des colonnes.

CHARGEMENT D'UNE TABLE : INSERT

Exemple 1: *Insertion de valeurs dans la table **DSN8710.EMP**.*

```
INSERT INTO DSN8710.EMP
VALUES ('000205','MARY','T','SMITH','D11','2866',
       '1981-08-10','ANALYST',16,'F','1956-05-22',
       16345,500,2300);
```

Exemple 2: *Insertion de valeurs dans la table **DSN8710.EMP**
A partir de la table **DSN8710.EMP***

```
INSERT INTO SMITH.TEMPEMPL
SELECT * FROM DSN8710.EMP;
```

Exemple 3: *Autre exemple d'insertion*

```
INSERT INTO SESSION.TEMPEMPL
SELECT *
FROM DSN8710.EMP
WHERE WORKDEPT='D11';
```

4. Les recherches simples de données

Objectifs

Savoir utiliser le langage SQL pour obtenir des résultats à partir des données rangées dans les tables.
Ordre SELECT pour chercher des informations dans une table

- WHERE pour conditionner la recherche
- GROUP BY pour travailler sur des groupes de lignes
- ORDER BY pour trier les résultats

SELECT

```

>> __select-clause__ INTO ____<_,____
                                host-variable_| __from-clause____>
> _____>
|_where-clause_| |_group-by-clause_| |_having-clause_|
> _____>
|_WITH____RR____| |_QUERYNO__integer_|
|_RS_|
|_CS_|
|_UR_|
> _____><
|_FETCH FIRST_____1____| _____ROW_____ONLY_|
|_ROWS_|

```

Fonctionnement

L'instruction SELECT forme conceptuellement, à partir de la table de base dont le nom suit FROM, une table résultante dont les colonnes dépendent des rubriques citées après SELECT (colonnes directement issues de la table d'origine, valeurs calculées, constantes, etc.), dont les lignes satisfont, tant pour leur contenu que pour leur présentation, aux options suivant, le cas échéant, le nom de la table.

```
SELECT rubriques
FROM nom-de-table
[options]
```

SELECT - EXAMPLES

Exemple 1: *Put the maximum salary in DSN8710.EMP into the host variable MAXSALRY.*

```
EXEC SQL SELECT MAX(SALARY)
        INTO :MAXSALRY
        FROM DSN8710.EMP;
```

Exemple 2: *Put the row for employee 528671, from DSN8710.EMP, into the host structure EMPREC.*

```
EXEC SQL SELECT * INTO :EMPREC
        FROM DSN8710.EMP
        WHERE EMPNO = '528671'
END-EXEC.
```

SELECT *

Le caractère * représente l'ensemble des colonnes de la table citée après FROM dans leur ordre de définition.

Exemple : Lister le contenu de la table LIVRES.

SELECT * FROM LIVRES

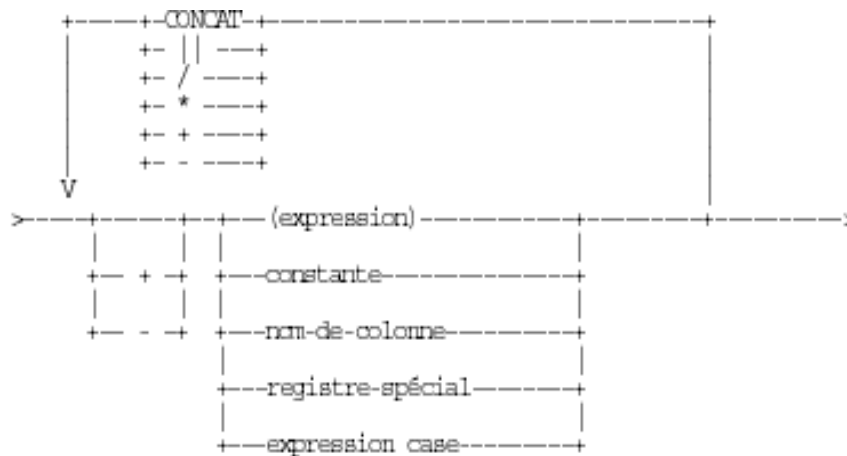
AUTEUR	TITRE	ANNEE	GENRE	PRIX
-----	-----	-----	-----	-----
HUGO	HERNANI	1830	THEATRE	120.00
HUGO	LES CONTEMPLATIONS	1856	POESIE	78.50
HUGO	LES MISERABLES	1862	ROMAN	148.50
BALZAC	EUGENIE GRANDET	1833	ROMAN	100.0
O BALZAC	LE PERE GORIOT	1834	ROMAN	-----
DUMAS	LES TROIS MOUSQUETAIRES	1844	ROMAN	80.00
DUMAS	VINGT ANS APRES	1845	ROMAN	80.00
DUMAS	LA DAME AUX CAMELIAS	1852	THEATRE	110.00
STENDHAL	LE ROUGE ET LE NOIR	1831	ROMAN	98.50
STENDHAL	LA CHARTREUSE DE PARME	1839	ROMAN	110.50
FLAUBERT	MADAME BOVARY	1857	ROMAN	99.50
VERLAINE	POEMES SATURNIENS	1866	POESIE	90.50

*Limiter le select * au strict minimum c'est à dire dans le cadre de sql interactif. Préférer indiquer la liste exhaustive des colonnes. Les performances n'en seront que meilleures (moins de mémoire, index only) et vos programmes seront moins dépendants de l'évolution de vos tables relationnelles*

SELECT AVEC EXPRESSION

SELECT peut être suivi d'une ou plusieurs expressions.

Format simplifié d'une expression :



Exemples d'expressions

```
'NOM : ' TABLE1.COLONNE_1 COLONNE_2 CONCAT ' ET'
      CONCAT COLONNE_3
      COLONNE_4 * COLONNE_5
      (COLONNE_6 * 1.25) / (COLONNE_7 + COL_8)
```

SELECT AVEC COLONNES

Exemple *Lister la date de parution et le titre des ouvrages contenus dans la table LIVRES.*

```
SELECT ANNEE, TITRE FROM LIVRES
```

Résultat

ANNEE	TITRE
-----	-----
1830	HERNANI
1856	LES CONTEMPLATIONS
1862	LES MISERABLES
1833	EUGENIE GRANDET
1834	LE PERE GORIOT
1844	LES TROIS MOUSQUETAIRES
1845	VINGT ANS APRES
1852	LA DAME AUX CAMELIAS
1831	LE ROUGE ET LE NOIR
1839	LA CHARTREUSE DE PARME
1857	MADAME BOVARY
1866	POEMES SATURNIENS

Le nom des colonnes peut être préfixer par le nom de la table (ex : livres.annee).

SELECT AVEC VALEURS CALCULÉES

Exemple *Lister le titre et le prix en centimes de chaque ouvrage de la table LIVRES.*

```
SELECT TITRE, 'PRIX : ', PRIX * 100,  
       'centimes' FROM LIVRES
```

Résultat

TITRE		
-----	-----	-----
HERNANI	PRIX :12000.00	centimes
LES CONTEMPLATIONS	PRIX :7850.00	centimes
LES MISERABLES	PRIX :14850.00	centimes
EUGENIE GRANDET	PRIX :10000.00	centimes
LE PERE GORIOT	PRIX : -----	centimes
LES TROIS MOUSQUETAIRES	PRIX :8000.00	centimes
VINGT ANS APRES	PRIX :8000.00	centimes
LA DAME AUX CAMELIAS	PRIX :11000.00	centimes
LE ROUGE ET LE NOIR	PRIX :9850.00	centimes
LA CHARTREUSE DE PARME	PRIX :11050.00	centimes
MADAME BOVARY	PRIX :9950.00	centimes
POEMES SATURNIENS	PRIX :9050.00	centimes

NB : *le prix de l'ouvrage 'LE PERE GORIOT' est toujours NULL.*

NOMMER UNE EXPRESSION

Il est possible d'attribuer un nom à une expression par l'utilisation de **AS** dans la clause **SELECT**

Ce nom peut être utilisé dans les autres clauses de l'instruction pour faire référence à la colonne résultant de l'évaluation de l'expression (**WHERE, GROUP BY, ORDER BY, etc...**)

Exemple *Lister le titre et le prix en centimes de chaque ouvrage de la table LIVRES.*

```
SELECT TITRE, PRIX * 100 AS CENTIMES
      FROM LIVRES
      ORDER BY CENTIMES
```

Résultat

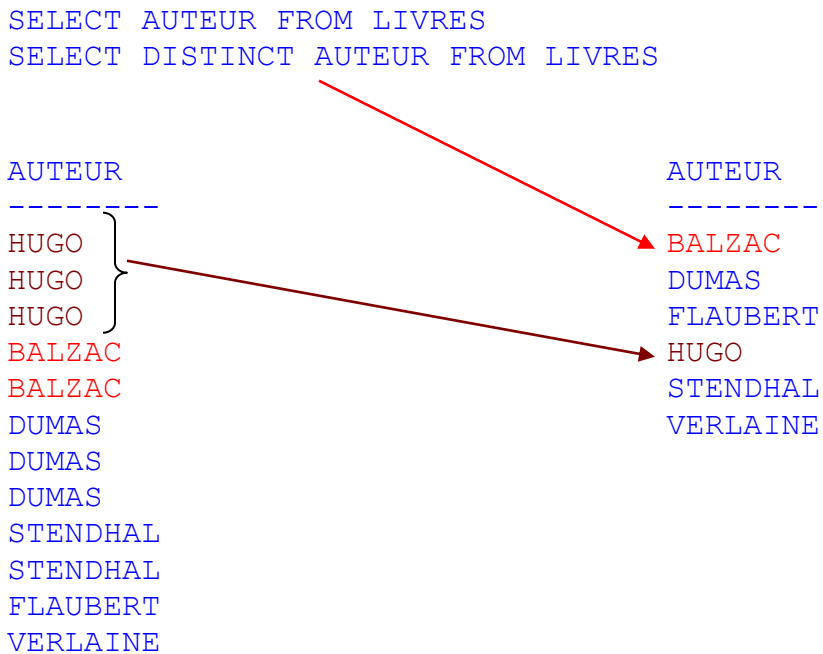
TITRE	CENTIMES
-----	-----
LES CONTEMPLATIONS	7850.00
LES TROIS MOUSQUETAIRES	8000.00
VINGT ANS APRES	8000.00
POEMES SATURNIENS	9050.00
LE ROUGE ET LE NOIR	9850.00
MADAME BOVARY	9950.00
EUGENIE GRANDET	10000.00
LA DAME AUX CAMELIAS	11000.00
LA CHARTREUSE DE PARME	11050.00
HERNANI	12000.00
LES MISERABLES	14850.00
LE PERE GORIOT	-----

L'OPTION DISTINCT

Exemple

Lister les auteurs de la table LIVRES.

Résultat : *Lister les auteurs de la table LIVRES.*



L'option DISTINCT indiquée après SELECT permet l'élimination des lignes en double.

SÉLECTION CONDITIONNELLE AVEC WHERE

Prédicat de base

L'option WHERE permet de préciser une condition de sélection des lignes de la table en utilisant un ou plusieurs prédicats. Prédicat de base.

```
>-----expression-----+---=---+-----expression----->
                                +---^---+
                                +---<---+
                                +--->---+
                                +---^>---+
                                +---<---+
                                +---^<---+
                                +--->=---+
                                +---<=---+
```

Règles de comparaison

- ⇒ Les nombres sont comparés selon leur valeur algébrique.
- ⇒ les chaînes de caractères sont comparées caractère par caractère de gauche à droite, la chaîne la plus courte étant considérée comme complétée par des blancs.

Le résultat d'un prédicat peut être "vrai", "faux" ou "inconnu" si l'une des expressions est nulle.

Seules les lignes donnant le résultat "vrai" sont retenues.

SÉLECTION CONDITIONNELLES AVEC WHERE

Condition de recherche

Elle est constituée d'un ou plusieurs prédicats reliés par les opérateurs logiques **AND**, **OR** et **NOT**.

Notation

```
|> _ | _NOT_ | _ | _predicate_ | _>|
| | _NOT_ | | _ (search-condition) _ |
|
|<
|> _ | _AND_ | _ | _predicate_ | _>|
| | _OR_ | | _NOT_ | | _ (search-condition) _ |
```

SELECT AVEC PRÉDICAT DE BASE

Exemple : *Rechercher, dans la table LIVRES, le titre et le genre de tous les ouvrages écrits par DUMAS.*

```
SELECT TITRE, GENRE
FROM LIVRES
WHERE AUTEUR = 'DUMAS'
```

Résultat

TITRE	GENRE
-----	-----
LES TROIS MOUSQUETAIRES	ROMAN
VINGT ANS APRES	ROMAN
LA DAME AUX CAMELIAS	THEATRE

SELECT AVEC CONDITION DE RECHERCHE

Exemple :

```
SELECT * FROM LIVRES
      WHERE NOT (AUTEUR = 'DUMAS')
      AND ANNEE > 1835
```

Résultat

AUTEUR	TITRE	ANNEE	GENRE	PRIX
-----	-----	-----	-----	-----
HUGO	LES CONTEMPLATIONS	1856	POESIE	78.50
HUGO	LES MISERABLES	1862	ROMAN	148.50
STENDHAL	LA CHARTREUSE DE PARME	1839	ROMAN	110.50
FLAUBERT	MADAME BOVARY	1857	ROMAN	99.50
VERLAINE	POEMES SATURNIENS	1866	POESIE	90.50

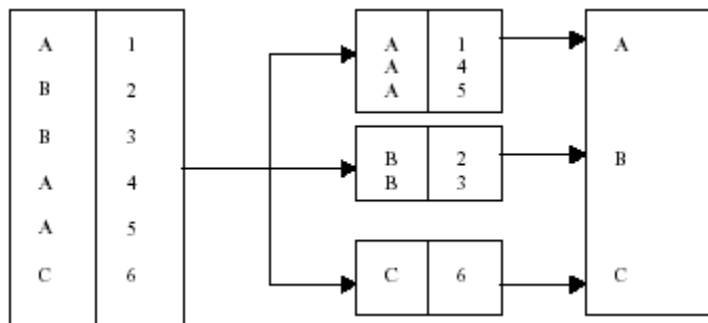
L'OPTION GROUP BY

Définition

L'option GROUP BY permet de traiter des groupes de lignes ayant une ou plusieurs caractéristiques communes.

Fonctionnement

- ⇒ GROUP BY suivi de colonnes de regroupement.
- ⇒ Autant de groupes que de valeurs distinctes des colonnes de regroupement.
- ⇒ Une ligne résultat par groupe.
- ⇒ Le SELECT ne peut comporter que les noms des colonnes de regroupement, ou des noms de fonctions.



L'OPTION GROUP BY

Exemple *Lister, en les regroupant par auteur et par genre, les données auteur et genre des ouvrages de la table LIVRES (ou donner les types d'ouvrages écrits par chaque auteur).*

```
SELECT AUTEUR, GENRE
FROM LIVRES
GROUP BY AUTEUR, GENRE
```

Résultat

AUTEUR	GENRE
-----	-----
BALZAC	ROMAN
DUMAS	ROMAN
DUMAS	THEATRE
FLAUBERT	ROMAN
HUGO	POESIE
HUGO	ROMAN
HUGO	THEATRE
STENDHAL	ROMAN
VERLAINE	POESIE

L'OPTION HAVING

Having

Associée à la clause GROUP BY, l'option HAVING permet d'appliquer une condition de recherche à des groupes.

WHERE : sélection de lignes avant regroupements.

HAVING : sélection de groupes créés par GROUP BY.

Exemple : *Lister en les regroupant par auteur et par genre, les données relatives aux auteurs et aux genres des ouvrages de la table LIVRES, en éliminant les poésies.*

```
SELECT AUTEUR, GENRE FROM LIVRES
      GROUP BY AUTEUR, GENRE
      HAVING GENRE <> 'POESIE'
```

Résultat

AUTEUR	GENRE
-----	-----
BALZAC	ROMAN
DUMAS	ROMAN
DUMAS	THEATRE
FLAUBERT	ROMAN
HUGO	ROMAN
HUGO	THEATRE
STENDHAL	ROMAN

L option HAVING est obligatoirement précédée d'un GROUP BY.

L'OPTION ORDER BY

L'option ORDER BY permet le classement de la table résultante.
Critère de classement : nom ou numéro relatif de colonne.
Séquence de tri : ASC (par défaut) ou DESC.

Exemple *Lister l'auteur, le titre et l'année des ouvrages de la table LIVRES, triés par ordre chronologique.*

```
SELECT AUTEUR, TITRE, ANNEE
FROM LIVRES
ORDER BY ANNEE
```

Résultat

AUTEUR	TITRE	ANNEE
HUGO	HERNANI	1830
STENDHAL	LE ROUGE ET LE NOIR	1831
BALZAC	EUGENIE GRANDET	1833
BALZAC	LE PERE GORIOT	1834
STENDHAL	LA CHARTREUSE DE PARME	1839
DUMAS	LES TROIS MOUSQUETAIRES	1844
DUMAS	VINGT ANS APRES	1845
DUMAS	LA DAME AUX CAMELIAS	1852
HUGO	LES CONTEMPLATIONS	1856
FLAUBERT	MADAME BOVARY	1857
HUGO	LES MISERABLES	1862
VERLAINE	POEMES SATURNIENS	1866

L'OPTION ORDER BY

Exemple Lister par ordre de prix en centimes décroissant et par titre croissant le prix en centimes, l'auteur et le titre des ouvrages de la table LIVRES.

```
SELECT PRIX * 100, AUTEUR, TITRE
FROM LIVRES
ORDER BY 1 DESC, TITRE
```

Résultat

	AUTEUR	TITRE
-----	-----	-----
-----	BALZAC	LE PERE GORIOT
14850.00	HUGO	LES MISERABLES
12000.00	HUGO	HERNANI
11050.00	STENDHAL	LA CHARTREUSE DE PARME
11000.00	DUMAS	LA DAME AUX CAMELIAS
10000.00	BALZAC	EUGENIE GRANDET
9950.00	FLAUBERT	MADAME BOVARY
9850.00	STENDHAL	LE ROUGE ET LE NOIR
9050.00	VERLAINE	POEMES SATURNIENS
8000.00	DUMAS	LES TROIS MOUSQUETAIRES
8000.00	DUMAS	VINGT ANS APRES
7850.00	HUGO	LES CONTEMPLATIONS

« order by 1 » s'exerce sur la première colonne ou expression citée dans le select. D'autre part le prix de l'ouvrage « Le père GORIOT » n'est pas renseigné, il est pourtant pris en compte et apparaît dans la table résultante en première position puisque sa valeur est NULL et que le critère de tri est DESC.

EXEMPLE RÉCAPITULATIF

Exemple : *Lister l'auteur, le libellé "GENRE : " et le genre des ouvrages de la table LIVRES non publiés en 1834, en les regroupant par genre et par auteur, en éliminant les pièces de théâtre, et en triant le résultat par genre croissant et par auteur décroissant.*

```
SELECT AUTEUR, 'GENRE : ', GENRE
FROM LIVRES
WHERE ANNEE <> 1834
GROUP BY GENRE, AUTEUR
HAVING GENRE <> 'THEATRE'
ORDER BY GENRE ASC, AUTEUR DESC
```

Résultat

AUTEUR		GENRE
-----	-----	-----
VERLAINE	GENRE :	POESIE
HUGO	GENRE :	POESIE
STENDHAL	GENRE :	ROMAN
HUGO	GENRE :	ROMAN
FLAUBERT	GENRE :	ROMAN
DUMAS	GENRE :	ROMAN
BALZAC	GENRE :	ROMAN

5. Les possibilités avancées de recherche de données.

OBJECTIFS

Savoir manipuler des formes plus complexes de l'instruction SELECT :

- Sous-select
- Prédicats
- Fonctions incorporées dans SQL
- Traitement des données temporelles

TABLE AUTEURS

Pour illustrer ce chapitre, nous créons une nouvelle table :

Table auteurs

AUTEUR	NE_EN	LIEU	SALLE	RAYON
HUGO	1802	BESANCON	2	3
BALZAC	1799	TOURS	1	1
DUMAS	1802	VILLERS-COTTERETS	1	1
DUMAS	1824	PARIS	1	1
STENDHAL	1783	GRENOBLE	3	5
FLAUBERT	1821	ROUEN	1	2
VERLAINE	1844	METZ	3	6
ZOLA	1840	PARIS	3	6
RIMBAUD	1854	CHARLEVILLE	3	5
SAND	1804	PARIS	3	5

Définition d'un SOUS-SELECT

On appelle sous-select un ordre SELECT comportant le mot-clé FROM et, éventuellement, les options WHERE, GROUP BY et HAVING.

Le sous-select construit une table résultante selon la séquence d'opérations suivante :

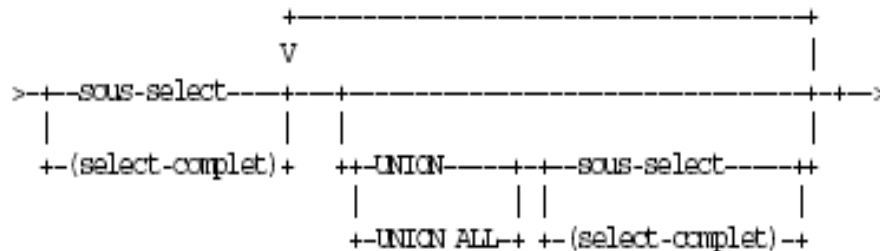
1. recherche de la table définie après le mot clé FROM,
2. application de la ou des conditions explicitées par la clause WHERE,
3. constitution des groupes selon la clause GROUP BY,
4. élimination des groupes ne répondant pas à la clause HAVING,
5. enfin, présentation de la table résultante selon les modalités suivant le mot clé SELECT (liste de rubriques avec, éventuellement, l'option DISTINCT).

SELECT ET SOUS-SELECT

Définition d'un SELECT COMPLET

Un SELECT complet est la combinaison d'une ou plusieurs instructions SELECT (portant sur une ou plusieurs tables).

Format



Après ce SELECT complet, on pourra éventuellement faire figurer l'option ORDER BY pour ordonner le résultat de la requête.

UNION permet de définir une table résultante à partir des lignes de deux autres tables (tables origine) après application des clauses figurant dans chacun des sous-select.

Tables d'origine :

- ☐ même nombre de colonnes
- ☐ deux colonnes de même rang doivent être de même nature

UNION ALL

La table résultante contient toutes les lignes provenant de chacun des sous-select. UNION

UNION

Les lignes en double sont éliminées (peut entraîner un tri).

Cumul des lignes de même nature

OPTION UNION

Exemple Lister par ordre chronologique les événements intervenus entre 1825 et 1850 figurant dans les tables LIVRES et AUTEURS, en précisant la nature de l'événement et l'auteur correspondant.

```
SELECT ANNEE, 'PARUTION ', AUTEUR
  FROM LIVRES WHERE ANNEE >= 1825 AND ANNEE <= 1850
UNION ALL
  SELECT NE_EN, 'NAISSANCE', AUTEUR FROM AUTEURS
    WHERE NE_EN >= 1825 AND NE_EN <= 1850 ORDER BY 1
```

Résultat

1830	PARUTION	HUGO
1831	PARUTION	STENDHAL
1833	PARUTION	BALZAC
1834	PARUTION	BALZAC
1839	PARUTION	STENDHAL
1840	NAISSANCE	ZOLA
1844	NAISSANCE	VERLAINE
1844	PARUTION	DUMAS
1845	PARUTION	DUMAS

LES PRÉDICATS

Dans une condition de recherche, on peut aussi utiliser les **prédicats** suivants :

- **BETWEEN**
- **NULL**
- **LIKE**
- **EXISTS**
- **IN**

prédicats quantifiés :

- **ALL**
- **SOME et ANY**

LE PRÉDICAT BETWEEN

Définition

Le prédicat BETWEEN permet une recherche dans une tranche de valeurs.

Exemple Lister le nom et le lieu de naissance de tous les auteurs nés entre 1802 et 1850.

```
SELECT AUTEUR, LIEU, FROM AUTEURS
      WHERE NE_EN BETWEEN 1802 AND 1850
```

Résultat

AUTEUR	LIEU
-----	-----
HUGO	BESANCON
DUMAS	VILLERS-COTTERETS
DUMAS	PARIS
FLAUBERT	ROUEN
VERLAINE	METZ
ZOLA	PARIS
SAND	PARIS

LE PRÉDICAT NULL

Définition

Le prédicat **NULL** permet de tester la ou les valeurs nulles dans une colonne.

```
>---nom-de-colonne---IS-----+-----+---NULL----->
                        |         |
                        +--NOT--+
```

Exemple

Lister l'auteur et le titre des ouvrages **dont on ne connaît pas le prix**

```
SELECT AUTEUR, TITRE FROM LIVRES
      WHERE PRIX IS NULL
```

Résultat

AUTEUR

BALZAC

TITRE

LE PERE GORIOT

LE PRÉDICAT LIKE

Définition

Le prédicat LIKE permet de rechercher des sous-chaînes de caractères précisées par leur profil.



La colonne doit être de type caractère.

La sous-chaîne de caractères définissant le profil peut contenir :

- ☐ le caractère _ (blanc souligné), remplaçant un caractère quelconque,
- ☐ le caractère % (pour-cent), remplaçant une séquence de zéro à n caractères,
- ☐ une combinaison de ces caractères _ et %,
- ☐ des caractères quelconques.

LE PRÉDICAT LIKE

Exemples Si la sous-chaîne recherchée contient les caractères _ ou %, chacun de ces caractères doit être immédiatement précédé, dans le profil, par un caractère "d'échappement". Le caractère "d'échappement" est indiqué dans une constante chaîne suivant le mot-clé ESCAPE.

AUTEUR LIKE 'B%'

est vrai pour tout auteur dont le nom commence par la lettre B

AUTEUR LIKE '_A_ _ _ _ _'

est vrai pour tout auteur ayant un "A" en deuxième caractère

AUTEUR LIKE '_ _+_%' ESCAPE '+'

est vrai pour tout auteur ayant un blanc souligné en troisième caractère.

Recherche d'un '_' en 3ème position.

LE PRÉDICAT EXISTS

Format

Le prédicat EXISTS permet de tester si le sous-select placé après EXISTS retourne au moins une ligne.

```
>---EXISTS---(sous-select)----->
```

Exemple

```
SELECT * FROM LIVRES
WHERE EXISTS
  (SELECT 1 FROM AUTEURS WHERE NE_EN = 1830)
```

produit, comme résultat, une table vide, car il n'y a pas un seul auteur né en 1830.

☞ ☐ : *Il n'est pas nécessaire de mentionner des noms de colonnes dans la clause SELECT de la sous-requête. Le prédicat EXISTS est surtout utilisé avec des sous-requêtes corrélées.*

LE PRÉDICAT IN

Format

Le prédicat IN permet une recherche dans une collection de valeurs définie par une énumération ou par un sous-select

Exemple Lister l'auteur, le titre et l'année des ouvrages de la table LIVRES écrits en 1839, 1866 ou 1857.

```
SELECT AUTEUR, TITRE, ANNEE FROM LIVRES
WHERE ANNEE IN (1839, 1866, 1857)
```

Résultat

AUTEUR	TITRE	ANNEE
-----	-----	-----
STENDHAL	LA CHARTREUSE DE PARME	1839
FLAUBERT	MADAME BOVARY	1857
VERLAINE	POEMES SATURNIENS	1866

Equivalent à (ANNEE = 1839 OR ANNEE = 1866 OR ANNEE = 1857) mais préférable.

LE PRÉDICAT IN

Exemple Lister le nom et le lieu de naissance des auteurs de la table AUTEURS figurant dans la table LIVRES.

```
SELECT AUTEUR, LIEU FROM AUTEURS
WHERE AUTEUR IN
      (SELECT AUTEUR FROM LIVRES)
```

Fonctionnement La requête la plus interne est exécutée en premier, ce qui permet de définir une liste d'auteurs.

- La requête de niveau supérieur est ensuite exécutée, ce qui permet de ne sélectionner que les auteurs figurant dans la liste.

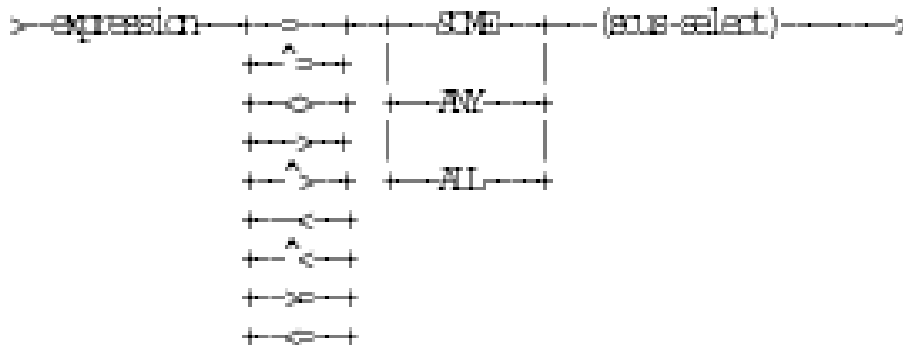
Résultat

AUTEUR	LIEU
HUGO	BESANCON
BALZAC	TOURS
DUMAS	VILLERS-COTTERETS
DUMAS	PARIS
STENDHAL	GRENOBLE
FLAUBERT	ROUEN
VERLAINE	METZ

LES PRÉDICATS QUANTIFIÉS

SOME, ANY, ALL

Un prédicat quantifié permet d'effectuer une comparaison entre une expression et le résultat d'un sous-select avec l'une des options **SOME**, **ANY** ou **ALL**.



ALL : « vrai »

- ☐ si la condition est satisfaite pour toutes les valeurs retournées par le sous-select condition.
- ☐ ou si le sous-select ne renvoie aucune valeur

SOME (ou ANY)

- ☐ « vrai » si la condition est satisfaite pour au moins une valeur retournée par le sous-select.
- ☐ « faux » si aucune valeur retournée par le sous select

Définition

Une fonction sur colonne renvoie une valeur unique.
5 fonctions sur colonne :

- AVG
- SUM
- MAX et MIN
- COUNT

Le résultat d'une fonction sur colonne dépend du type de SELECT :

- sans GROUP BY :
une seule valeur pour toute la table
- avec GROUP BY :
une valeur par groupe distinct

LA FONCTION AVG

Format La fonction AVG permet de calculer la moyenne des valeurs d'une collection de nombres.



Le type de donnée résultant est identique au type de données origine.

La valeur nulle est neutre.

Le résultat est la valeur nulle si la collection de valeurs est vide.

Exemple Lister la moyenne des prix de chaque genre d'ouvrage de la table LIVRES.

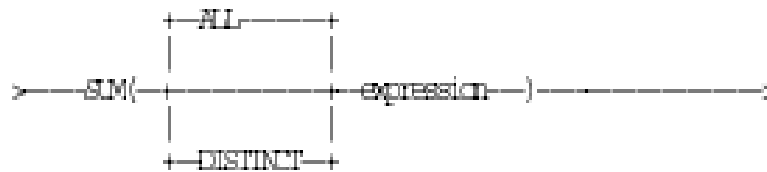
```
SELECT GENRE, AVG(PRIX) FROM LIVRES
GROUP BY GENRE
```

Résultat

GENRE	
POESIE	84.500000000000
ROMAN	102.42857142857
THEATRE	115.000000000000

LA FONCTION SUM

Format La fonction SUM permet de calculer la somme des valeurs d'une collection de nombres.



Format Lister la somme des prix des romans de la table LIVRES.

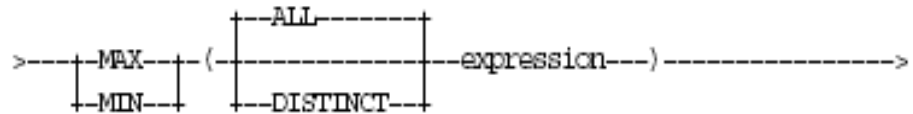
```
SELECT SUM(PRIX) FROM LIVRES
      WHERE GENRE = 'ROMAN'
```

Résultat

```
-----
      717.00
```

LES FONCTIONS MAX ET MIN

Format Les fonctions MAX et MIN ont le même format :



MAX et MIN s'appliquent sur tous les types de données. Les valeurs nulles sont ignorées. DISTINCT n'a aucune incidence sur le résultat obtenu. ***Le résultat est la valeur nulle si la collection de valeurs est vide.***

Exemples Donner le prix maximum des ouvrages de la table LIVRES.

```
SELECT MAX(PRIX) FROM LIVRES
```

Donner le prix maximum des ouvrages de chaque genre de la table LIVRES.

```
SELECT GENRE, MAX(PRIX) FROM LIVRES  
GROUP BY GENRE
```

Résultat

GENRE	
POESIE	90.50
ROMAN	148.50
THEATRE	120.00

LA FONCTION COUNT

Format La fonction COUNT permet de dénombrer une collection de lignes ou de valeurs.

```
>-----COUNT(---+---DISTINCT---expression-----+---)----->
                |                                     |
                +---*-----+
```

Le résultat est un entier long

COUNT(*) : nombre de lignes d'un groupe ou d'une table résultante après application des conditions de sélection

COUNT (DISTINCT nom-de-colonne) : nombres de valeurs distinctes présentes dans une colonne, en éliminant les valeurs nulles

Le résultat est égal à 0 si la collection de valeurs est vide.

LA FONCTION COUNT

Exemple

Compter le nombre d'auteurs figurant dans la table LIVRES.

```
SELECT COUNT(DISTINCT AUTEUR)
FROM LIVRES
```

Résultat :

```
-----
6
```

Exemple

Lister les auteurs figurant dans la table LIVRES ayant écrit plus d'un ouvrage.

```
SELECT AUTEUR FROM LIVRES
GROUP BY AUTEUR
HAVING COUNT (*) > 1
```

Résultat

```
AUTEUR
-----
BALZAC
DUMAS
HUGO
STENDHAL
```


LES FONCTIONS SCALAIRES

Format Les fonctions scalaires effectuent une opération sur une valeur (ou sur un nombre fixe de valeurs). Une fonction scalaire peut être utilisée dans une expression.

fonctions scalaires de format simple

```
>>- nom-de-fonction ( expression ) --><
```

Fonctions ayant ce format :

- DIGITS
- FLOAT
- HEX
- INTEGER
- LENGTH
- VARGRAPHIC

LES FONCTIONS SCALAIRES

DIGITS convertit la valeur absolue d'un nombre entier décimal ou binaire en chaîne de caractères de longueur fixe

FLOAT permet d'obtenir la représentation en virgule flottante double précision d'un nombre.

HEX donne la représentation en hexadécimal d'une chaîne de caractères.

Si l'argument est une date, une heure ou un instant, on obtient la représentation hexadécimale interne de cet argument

INTEGER permet d'obtenir, sous forme d'un entier long, la représentation d'un nombre.

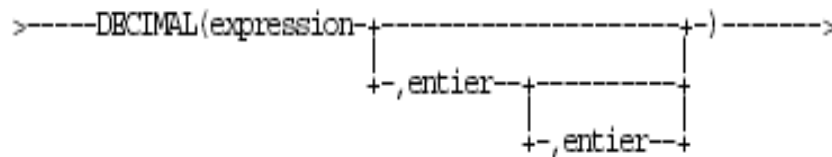
LENGTH donne la longueur de la représentation interne d'un argument.

VARGRAPHIC permet de convertir une chaîne de caractères en chaîne de caractères graphique de longueur variable.

Length est surtout intéressant pour les VARCHAR

LA FONCTION DECIMAL

Format La fonction **DECIMAL** permet d'obtenir la représentation décimale d'une valeur numérique.



Premier argument : nombre à convertir.

Deuxième argument : précision du résultat

Troisième argument : échelle

Exemple

```
DECIMAL (AVG (PRIX) , 6, 2)
```

```
donne 0101.45.
```

LA FONCTION SUBSTR

Format La fonction **SUBSTR** permet d'extraire une sous-chaîne à partir d'une chaîne de caractères.

>-----SUBSTR(expression,position--+-----+---)>
 | |
 +--,longueur--+

Expression : chaîne de caractères de départ

Position : position de départ de la sous-chaîne

Longueur : longueur de la sous-chaîne à extraire

EXEMPLES

SUBSTR ('HUGO', 2) donne 'UGO',

SUBSTR ('HUGO', 2, 2) donne 'UG'.

LES FONCTIONS VALUE ET COALESCE

Format La fonction **VALUE** permet de substituer une valeur définie à une valeur nulle.

+-----+
V |
>-----VALUE(expression---,expression---+)->

Les expressions entre parenthèses doivent être compatibles.

Le résultat de la fonction sera la première valeur non nulle.

Le résultat ne sera nul que si toutes les expressions ont la valeur nulle.

Exemple `VALUE (PRIX, 0)`

le résultat sera :

- ☐ la valeur de PRIX si elle est non nulle
- ☐ zéro si PRIX a la valeur nulle

La fonction **COALESCE** est synonyme de la fonction **VALUE**

Utiliser COALESCE pour être en conformité avec le standard SQL-92.

L'EXPRESSION CASE

Définition

Une expression CASE permet :

- d'évaluer une condition avec la clause WHEN
- D'affecter une valeur de sortie avec la clause

ELSE si aucun cas n'est vérifié.

L'expression CASE peut-être utilisée dans un prédicat IN

La fonction CASE est plus performante en terme de temps de traitement et d'autre part moins lourde à écrire et donc à relire.

L'expression CASE est déjà fréquemment utilisée dans de nombreux langages de programmation

La valeur est une expression résultat ou NULL

Si l'évaluation du CASE n'est jamais VRAI et si la clause ELSE est spécifiée alors le résultat est soit l'expression résultat, soit NULL

Si l'évaluation du CASE n'est jamais VRAI et si la clause ELSE n'est pas spécifié alors le résultat est NULL


Les différents cas sont déterminés en fonction de certains critères.

Chaque cas à son propre résultat qui dépend de la vérification du critère de sélection

Table 8. Equivalent CASE Expressions	
CASE Expression	Equivalent Expression
CASE WHEN e1=e2 THEN NULL ELSE e1 END	NULLIF(e1,e2)
CASE WHEN e1 IS NOT NULL THEN e1 ELSE e2 END	COALESCE(e1,e2)
CASE WHEN e1 IS NOT NULL THEN e1 ELSE COALESCE(e2,...,eN) END	COALESCE(e1,e2,...,eN)

L'EXPRESSION CASE

Exemples

<pre>SELECT ACCOUNT_NUM, 'NORD' FROM ACCOUNT_TABLE WHERE REGION = 'N' UNION ALL SELECT ACCOUNT_NUM, 'SUD' FROM ACCOUNT_TABLE WHERE REGION = 'S' UNION ALL SELECT ACCOUNT_NUM, 'OUEST' FROM ACCOUNT_TABLE WHERE REGION = 'W' UNION ALL SELECT ACCOUNT_NUM, 'EST' FROM ACCOUNT_TABLE WHERE REGION = 'E' UNION ALL SELECT ACCOUNT_NUM, 'NON ASSIGNE' FROM ACCOUNT_TABLE WHERE REGION NOT IN ('N', 'S', 'W', 'E')</pre>	 <pre>SELECT ACCOUNT_NUM, CASE REGION WHEN 'N' THEN 'NORD' WHEN 'S' THEN 'SUD' WHEN 'W' THEN 'OUEST' WHEN 'E' THEN 'EST' ELSE 'NON ASSIGNE' END FROM ACCOUNT_TABLE</pre>
---	--

Exemples

```
UPDATE table_employe
  SET SALAIRE = CASE NOTE
    WHEN 15 THEN SALAIRE * 1.15
    WHEN 16 THEN SALAIRE * 1.16
    WHEN 17 THEN SALAIRE * 1.17
    WHEN 18 THEN SALAIRE * 1.18
    WHEN 19 THEN SALAIRE * 1.19
    WHEN 20 THEN SALAIRE * 1.20
  ELSE SALAIRE
END ;
```

```
SELECT EMPNO, NO_DEPT, SALAIRE+COMM
  FROM EMPLOYE
 WHERE
   CASE WHEN SALAIRE = 0 THEN NULL
        ELSE COMM/SALAIRE
   END
   > 0.25 ;
```

Cette requête permet de retrouver les employés qui gagnent plus de 25% en commission par rapport à leur salaire

LA FONCTION NULLIF

Format

```
>--NULLIF--(--expression--,--expression--)----->
```

Règles Cette fonction permet de retourner une valeur NULL si les deux arguments sont égaux.

Si les deux arguments sont différents, alors la valeur du premier argument est renvoyée.

Les deux arguments doivent être de type compatible.(comparaison de deux expressions caractères ou de deux expressions numériques ou de deux dates,durée...).

NULLIF(e1,e2) donne le même résultat que l'expression CASE ci dessous :

```
CASE WHEN e1=e2 THEN NULL ELSE e1 END
```

NULLIF retourne la valeur du premier arguments dans le cas ou une ou les deux expressions sont nulles.

LA FONCTION STRIP

Règles

La fonction STRIP permet à une application de retirer les blancs ou tout autre caractère spécial de la fin, du début, voire des deux, d'une chaîne de caractères.

- ☞ Le premier argument doit être une chaîne de caractères
- ☞ Le deuxième est optionnel, il indique si les caractères sont à retirer du début ou de la fin de la chaîne. S'il n'est pas spécifié les caractères sont à retirer au début et à la fin de la chaîne
- ☞ Le troisième est une constante d'un caractère qui indique soit SBCS soit DBCS.

Exemples

⇒ L'exemple suivant retourne le résultat 123.45 si la valeur de COST de type CHAR(10) est 0000123.45 :

```
STRIP (:COST, L, '0')
```

⇒ L'exemple suivant retourne le résultat 'HUGO' si la valeur de AUTEUR de type CHAR(9) est 'HUGO'

```
STRIP (:AUTEUR)
```

STRIP retourne ...

- ⇒ Le résultat de cette fonction sera sous la forme d'une chaîne de longueur variable avec la même longueur maximum que la variable d'origine.
- ⇒ La taille du résultat sera de longueur :
expression – nombre d'octets supprimés
- ⇒ Tous les arguments peuvent être supprimés : le résultat sera une chaîne de caractères variables de longueur zéro.
- ⇒ Si le premier argument est null le résultat sera aussi null.

IMBRICATION DE FONCTIONS

Exemples

```
SUM (INTEGER (PRIX))
```

somme les valeurs entières des prix, soit 1113.

```
INTEGER (SUM (PRIX))
```

donne la valeur entière de SUM (PRIX), soit 1116.

EXPRESSION DE DURÉE

Format Une expression de durée représente une unité de temps exprimée selon le format



expression : addition ou soustraction

premier opérande : de type DATE, TIME ou TIMESTAMP

Unité

un des mots de la liste

YEAR	YEARS
MONTH	MONTHS
DAY	DAYS
HOURL	HOURS
MINUTE	MINUTES
SECOND	SECONDS
MICROSECOND	MICROSECONDS

Exemple

Ajout de deux expressions de durée d'unités différentes à une date.

```
DATE_1 + 5 MONTHS + 6 DAYS
```

DURÉES

Durée date

- nombre de format AAAAMMMJJ
- obtenue par la soustraction de deux dates

ex : `DATE_FIN - DATE_DEBUT`

Durée heure

- nombre de format HHMMSS
- obtenue par la soustraction de deux heures

ex : `HEURE_FIN - HEURE_DEBUT`

Durée instant

- nombre de format AAAAMMMJJHHMMSSmmmmmm
- obtenue par la soustraction de deux instants

ex : `INSTANT_FIN - INSTANT_DEBUT`

Opérations sur données temporelles

Affectations

Une donnée de type DATE, TIME ou TIMESTAMP peut être affectée

- Soit à une colonne de même type
- Soit à une variable ou une colonne représentée par une chaîne de caractères

Dans ce cas, il y a conversion en constante de type date, heure ou instant

Comparaison

Une donnée de type DATE, TIME ou TIMESTAMP peut être comparée à une donnée de même type ou à une constante de son type

Les comparaisons s'effectuent selon la séquence chronologique

Exemple :

HEURE-1 > '20.15.10'

DATE-1 > DATE-2

ADDITIONS

Additions

Format

On ne peut additionner qu'une durée à une donnée temporelle :

DATE + durée → DATE

HEURE + durée → HEURE

INSTANT + durée → INSTANT

Les additions se font en tenant compte du calendrier

Exemple :

Si DATE-1 contient une date au 31 Janvier 2009

DATE-1 + 31 DAYS

Contiendra le 3 mars 2009 puisque l'année 2009 n'est pas bissextile.

FONCTIONS SCALAIRES LIÉES AU TEMPS

DATE	permet d'obtenir une <i>date</i>
DAY	donne la partie <i>JJ</i> d'une date ou le nombre de jours écoulés entre 2 dates
DAYOFMONTH	donne la partie <i>JJ</i> d'une date
DAYOFWEEK	donne le numéro du jour dans la semaine (de 1 pour dimanche à 7 pour samedi) correspondant à une date
DAYOFWEEK_ISO	donne le numéro du jour dans la semaine (de 1 pour lundi à 7 pour dimanche) correspondant à une date
DAYOFYEAR	donne le numéro du jour dans l'année (de 1 pour le 1er janvier à 366)
DAYS	donne le <i>nombre de jours</i> entre le 1er janvier 0001 et une date
HOUR	donne la partie <i>HH</i> d'une heure
JULIAN_DAY	donne le nombre de jours écoulés depuis le 1er janvier du calendrier julien
LAST_DAY	donne la date correspondant au dernier jour du mois de la date fournie en argument
MICROSECOND	donne la partie <i>microsecondes</i> d'un instant
MIDNIGHT_SECONDS	donne le nombre de <i>microsecondes</i> écoulées depuis minuit et l'instant fourni en argument
MINUTE	donne la partie <i>MM</i> d'une heure
MONTH	donne la partie <i>MM</i> d'une date
NEXT_DAY	donne la date qui est postérieure à la date précisée en argument et qui correspond au jour de la semaine précisé en argument
QUARTER	donne le numéro du trimestre correspondant à la date précisée en argument
SECOND	donne la partie <i>SS</i> d'une heure
TIME	permet d'obtenir une <i>heure</i>
TIMESTAMP	permet d'obtenir un <i>instant</i>

LA FONCTION CHAR

La fonction CHAR

Temps à caractère

La fonction **CHAR** permet de convertir une date, une heure ou un instant en une chaîne de caractères de longueur fixe

```
>-----CHAR(expression--+-----+)->
                        +---, ISO-----+
                        +---, USA-----+
                        +---, EUR-----+
                        +---, JIS-----+
                        +---, LOCAL---+
```

Le second paramètre de la fonction indique le format désiré.
(par défaut : selon option de format de l'installation)

CHAR(CURRENT DATE, USA)
peut donner '10/26/2009'

Format

```
>-----TIMESTAMP (expression1+-----+)------>
                        |               |
                        +- , expression2 -+
```

sont équivalents

6. Mise à jour - Normalisation et index

OBJECTIFS DU CHAPITRE

Connaître les opérations de mise à jour de données :

- ⇒ Ajout
- ⇒ Modification
- ⇒ Annulation

Etre sensibilisé aux notions de normalisation des tables:

- ⇒ Trois premières formes normales
- ⇒ Clé primaire

Savoir utiliser les index :

- ⇒ Création
- ⇒ Rôle et utilisation

TABLE OUVRAGES

Pour illustrer ce chapitre nous créons une nouvelle table.

AUTEUR	TITRE	NE_EN	SALLE	RAYON
HUGO	HERNANI	1802	2	3
HUGO	LES CONTEMPLATIONS	1802	2	3
HUGO	LES MISERABLES	1802	2	3
BALZAC	EUGENIE GRANDET	1799	1	1
BALZAC	LE PERE GORIOT	1799	1	1
DUMAS	LES TROIS MOUSQUETAIRES	1802	1	1
DUMAS	VINGT ANS APRES	1802	1	1
DUMAS	LA DAME AUX CAMELIAS	1824	1	1
STENDHAL	LE ROUGE ET LE NOIR	1783	3	5
STENDHAL	LA CHARTREUSE DE PARME	1783	3	5
FLAUBERT	MADAME BOVARY	1821	1	2
VERLAINE	POEMES SATURNIENS	1844	3	6
ZOLA	GERMINAL	1840	3	6
RIMBAUD	UNE SAISON EN ENFER	1854	3	5
SAND	LA MARE AU DIABLE	1804	3	5

INSERT : EXEMPLES

Exemple 1 Initialiser la table OUVRAGES (considérée comme vide) à partir de la table LIVRES et lister la table après initialisation.

```
INSERT INTO OUVRAGES (AUTEUR, TITRE)
SELECT AUTEUR, TITRE FROM LIVRES
SELECT * FROM OUVRAGES
```

Résultat

AUTEUR	TITRE	NE_EN	SALLE	RAYON
HUGO	HERNANI	0		
HUGO	LES CONTEMPLATIONS	0		
HUGO	LES MISERABLES	0		
BALZAC	EUGENIE GRANDET	0		
BALZAC	LE PERE GORIOT	0		
DUMAS	LES TROIS MOUSQUETAIRES	0		
DUMAS	VINGT ANS APRES	0		
DUMAS	LA DAME AUX CAMELIAS	0		
STENDHAL	LE ROUGE ET LE NOIR	0		
STENDHAL	LA CHARTREUSE DE PARME	0		
FLAUBERT	MADAME BOVARY	0		
VERLAINE	POEMES SATURNIENS	0		

INSERT : EXEMPLES

```
INSERT INTO OUVRAGES
(AUTEUR, NE_EN, SALLE, RAYON)
SELECT AUTEUR, NE_EN, SALLE, RAYON FROM AUTEURS
SELECT * FROM OUVRAGES
```

Résultat

AUTEUR	TITRE	NE_EN	SALLE	RAYON
-----	-----	-----	-----	-----
HUGO		1802	2	3
BALZAC		1799	1	1
DUMAS		1802	1	1
DUMAS		1824	1	1
STENDHAL		1783	3	5
FLAUBERT		1821	1	2
VERLAINE		1844	3	6
ZOLA		1840	3	6
RIMBAUD		1854	3	5
SAND		1804	3	5

UPDATE

L'instruction **UPDATE** permet de modifier la valeur des données des colonnes d'une ou de plusieurs lignes d'une table.

Format général simplifié

```
>>--UPDATE--nom-de-table-->
      +-----+
      | +,-----+
      | V         |
>--SET--nom-de-colonne=+expression--+>
                        |             |
                        +--NULL-----+

>-----><
      |-----|
      +--WHERE-condition-de-recherche--+
```

UPDATE est suivi du nom de la table dans laquelle on veut modifier des données.

SET définit le nom de la ou des colonnes que l'on désire mettre à jour, ainsi que le mode d'obtention de la nouvelle valeur.

WHERE permet de sélectionner les lignes à mettre à jour. En cas d'erreur, aucune valeur n'est modifiée.

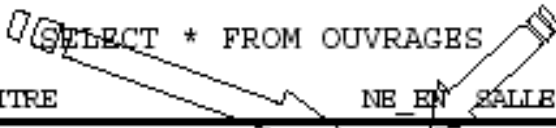
UPDATE : EXEMPLES

Exemple 3 Mettre à jour la table OUVRAGES obtenue après l'exemple 1 en remplaçant l'année de naissance par la constante 1800.

```
UPDATE OUVRAGES  
SET NE_EN = 1800
```

```
UPDATE OUVRAGES  
SET NE_EN = NE_EN + 1800
```

Résultat



AUTEUR	TITRE	NE_EN	SALLE	RAYON
HUGO	HERNANI	1800	-----	-----
HUGO	LES CONTEMPLATIONS	1800	-----	-----
HUGO	LES MISERABLES	1800	-----	-----
BALZAC	EUGENIE GRANDET	1800	-----	-----
BALZAC	LE PERE GORIOT	1800	-----	-----
DUMAS	LES TROIS MOUSQUETAIRES	1800	-----	-----
DUMAS	VINGT ANS APRES	1800	-----	-----
DUMAS	LA DAME AUX CAMELIAS	1800	-----	-----
STENDHAL	LE ROUGE ET LE NOIR	1800	-----	-----
STENDHAL	LA CHARTREUSE DE PARME	1800	-----	-----
FLAUBERT	MADAME BOVARY	1800	-----	-----
VERLAINE	POEMES SATURNIENS	1800	-----	-----

*La colonne NE_EN a été mise à jour pour toutes les lignes de la table. Les modifications apparaissent en **caractères gras** et sont encadrés*

UPDATE : REMARQUES

Trois remarques :

1. la date de naissance de DUMAS est incorrecte
 2. dans la table AUTEURS, il existe deux DUMAS avec deux dates de naissance différentes : DUMAS père et DUMAS fils
 3. les tables AUTEURS et OUVRAGES présentent des incohérences (exemple : la date de naissance de VERLAINE est 1844 dans la table AUTEURS et 1800 dans la table OUVRAGES)
- L'opération de mise à jour n'assure donc pas implicitement l'intégrité des données.

DELETE

Format L'instruction DELETE permet de supprimer des lignes d'une table. Format général simplifié

```
>>--DELETE FROM---nom-de-table----->

>-----+-----+-----><
|                                     |
+---WHERE-condition-de-recherche---+
```

Fonctionnement :

- ☐ **pas d'option WHERE** : suppression de toutes les lignes de la table.
- ☐ en cas d'erreur, aucune ligne n'est supprimée.

DELETE : EXEMPLES

Exemple 7 Annuler, dans la table OUVRAGES obtenue après l'exemple 6, les lignes correspondant à des auteurs ayant leur date de naissance à la valeur 1800.

```
DELETE FROM OUVRAGES
WHERE NE_EN = 1800

SELECT * FROM OUVRAGES
```

Résultat

AUTEUR	TITRE	NE_EN	SALLE	RAYON
HUGO	HERNANI	1802	2	3
HUGO	LES CONTEMPLATIONS	1802	2	3
HUGO	LES MISERABLES	1802	2	3
BALZAC	EUGENIE GRANDET	1799	1	1
BALZAC	LE PERE GORIOT	1799	1	1
DUMAS P	LES TROIS MOUSQUETAIRES	1802	1	1
DUMAS P	VINGT ANS APRES	1802	1	1
DUMAS F	LA DAME AUX CAMELIAS	1824	1	1

LA PREMIÈRE FORME NORMALE ET SES INCONVÉNIENTS

Définition Une table est en première forme normale quand elle possède uniquement des données élémentaires pour chaque colonne dans chaque ligne (pas de données répétitives).

Inconvénients :

- ☐ anomalies au DELETE
- ☐ anomalies à l'INSERT
- ☐ anomalies à l'UPDATE

Toutes les tables présentées en exemples sont en première forme normale

LA PREMIÈRE FORME NORMALE ET SES INCONVÉNIENTS

Anomalies au DELETE

Exemple Annuler, dans la table OUVRAGES, les lignes relatives aux ouvrages écrits par l'auteur HUGO.

```
DELETE FROM OUVRAGES
WHERE AUTEUR = 'HUGO';
SELECT * FROM OUVRAGES
```

Résultat

AUTEUR	TITRE	NE_EN	SALLE	RAYON
-----	-----	-----	-----	-----
BALZAC	EUGENIE GRANDET	1799	1	1
BALZAC	LE PERE GORIOT	1799	1	1
DUMAS	LES TROIS MOUSQUETAIRES	1802	1	1
DUMAS	VINGT ANS APRES	1802	1	1
DUMAS	LA DAME AUX CAMELIAS	1824	1	1
STENDHAL	LE ROUGE ET LE NOIR	1783	3	5
STENDHAL	LA CHARTREUSE DE PARME	1783	3	5
FLAUBERT	MADAME BOVARY	1821	1	2
VERLAINE	POEMES SATURNIENS	1844	3	6
ZOLA	GERMINAL	1840	3	6
RIMBAUD	UNE SAISON EN ENFER	1854	3	5
SAND	LA MARE AU DIABLE	1804	3	5

En annulant les informations relatives aux ouvrages écrits par Victor HUGO, on perd également la date de naissance de cet auteur, information indépendante des ouvrages.

LA PREMIÈRE FORME NORMALE ET SES INCONVÉNIENTS

Anomalies à l'insert

Introduire, dans la table OUVRAGES, le livre "La Cousine Bette" de l'auteur BALZAC.

```
INSERT INTO OUVRAGES (AUTEUR, TITRE)
VALUES ('BALZAC', 'LA COUSINE BETTE')
```

```
SELECT * FROM OUVRAGES
```

INSERT
n'oblige
pas à
valoriser
toutes
les
données
de toutes
les
colonnes

AUTEUR	TITRE	NE_EN	SALLE	RAYON
-----	-----	-----	-----	-----
BALZAC	EUGENIE GRANDET	1799	1	1
BALZAC	LE PERE GORIOT	1799	1	1
DUMAS	LES TROIS MOUSQUETAIRES	1802	1	1
DUMAS	VINGT ANS APRES	1802	1	1
DUMAS	LA DAME AUX CAMELIAS	1824	1	1
STENDHAL	LE ROUGE ET LE NOIR	1783	3	5
STENDHAL	LA CHARTREUSE DE PARME	1783	3	5
FLAUBERT	MADAME BOVARY	1821	1	2
VERLAINE	POEMES SATURNIENS	1844	3	6
ZOLA	GERMINAL	1840	3	6
RIMBAUD	UNE SAISON EN ENFER	1854	3	5
SAND	LA MARE AU DIABLE	1804	3	5
BALZAC	LA COUSINE BETTE	0	----	-----

LA PREMIÈRE FORME NORMALE ET SES INCONVÉNIENTS

Anomalies à l'UPDATE

Mettre à jour la date de naissance, la salle et le rayon du livre "La Cousine Bette" de l'auteur BALZAC.

```
UPDATE OUVRAGES
  SET NE_EN = 1800, SALLE = 2, RAYON = 2
  WHERE AUTEUR = 'BALZAC'
    AND TITRE LIKE '%BETTE%'
```

```
SELECT * FROM OUVRAGES
```

AUTEUR	TITRE	NE_EN	SALLE	RAYON
BALZAC	EUGENIE GRANDET	1799	1	1
BALZAC	LE PERE GORIOT	1800	1	1
DUMAS	LES TROIS MOUSquetaIRES	1844	1	1
DUMAS	VINGT ANS	1820	1	1
DUMAS	LE COMTE DE CRISTO	1828	1	1
STENDHAL	LE REDOUTABLE	1829	5	5
STENDHAL	LE VERTIGINEUX	1828	5	5
FLAUBERT	MADAME BOLAUGNE	1857	2	2
VERLAINE	POESIES	1852	6	6
ZOLA	GERMINIUM	1865	6	6
RIMBAUD	UNE SAISON	1872	3	5
SAND	LA MARE AU DIABLE	1804	3	5
BALZAC	LA COUSINE BETTE	1800	2	2

LA PREMIÈRE FORME NORMALE ET SES INCONVÉNIENTS

Causes et solutions

Pourquoi ces anomalies ?

Parce que la table n'est qu'une juxtaposition de colonnes, alors que nous nous intéressons à la réalité que cette table décrit.

La table OUVRAGES contient :

- □ des données relatives à des auteurs (AUTEUR, NE_EN, SALLE, RAYON) car on range les ouvrages par auteur,
- □ des données relatives à des ouvrages (AUTEUR, TITRE).

De plus, les auteurs ne sont pas correctement identifiés: au nom DUMAS correspond deux auteurs différents (Alexandre DUMAS père et Alexandre DUMAS fils), chacun ayant sa propre date de naissance.

LA PREMIÈRE FORME NORMALE ET SES INCONVÉNIENTS

Identifiant (clé primaire : Définition

Après changement de **DUMAS** par **DUMAS F** ou **DUMAS P** on obtient un **identifiant unique** pour chaque auteur : c'est la **clé primaire** (ou identifiant ou critère d'identification).

On appelle clé primaire une donnée (ou une suite de données) permettant de définir de manière unique un élément dans un ensemble.

Clé candidate et clé primaire

Si plusieurs données peuvent être utilisables comme identifiant, on donne le nom de

clé candidate

à chacune de ces clés potentielles, et on choisit

la clé primaire

parmi les clés candidates.

DEUXIÈME FORME NORMALE

Pour mettre en évidence la deuxième forme normale, nous créons, par projection, deux nouvelles tables à partir de la table

OUVRAGES initiale :

- □ la table LIVRES1 contenant les données relatives aux livres
- □ la table AUTEURS1 contenant les données relatives aux auteurs

DEUXIÈME FORME NORMALE




Table LIVRES1	AUTEUR	TITRE	AUTEUR	NE_EN	SALLE	RAYON
Table AUTEURS1	BALZAC	EUGENIE GRANDET	BALZAC	1799	1	1
	BALZAC	LE PERE GORIOT	DUMAS F	1824	1	1
	DUMAS F	LA DAME AUX CAMELIAS	DUMAS P	1802	1	1
	DUMAS P	LES TROIS MOUSQUETAIRES	FLAUBERT	1821	1	2
	DUMAS P	VINGT ANS APRES	HUGO	1802	2	3
	FLAUBERT	MADAME BOVARY	RIMEAUD	1854	3	5
	HUGO	HERNANI	SAND	1804	3	5
	HUGO	LES CONTEMPLATIONS	STENDHAL	1783	3	5
	HUGO	LES MISERABLES	VERLAINE	1844	3	6
	RIMEAUD	UNE SAISON EN ENFER	ZOLA	1840	3	6
	SAND	LA MARE AU DIABLE				
	STENDHAL	LA CHARTREUSE DE PARME				
	STENDHAL	LE ROUGE ET LE NOIR				
	VERLAINE	POEMES SATURNIENS				
	ZOLA	GERMINAL				

La colonne TITRE constitue la clé primaire de la table LIVRES1.

La colonne AUTEUR constitue la clé primaire de la table AUTEURS1.

DEUXIÈME FORME NORMALE

Définition Les tables LIVRES1 et AUTEURS1 remédient aux inconvénients de mise à jour de la première forme normale.

Ces tables sont en deuxième forme normale.

Une relation est en deuxième forme normale si et seulement si elle est en première forme normale et si chaque attribut non clé dépend de la totalité de la clé.

LIVRES1 ne contient plus de redondances. C'est un ensemble de données relatives aux livres de la bibliothèque.

Cependant, des anomalies peuvent encore se produire au cours d'opérations de mise à jour de la table AUTEURS1, car elle contient des **dépendances fonctionnelles**.

DEUXIÈME FORME NORMALE

Dépendance fonctionnelle

Chaque auteur possède une date de naissance, une salle et un rayon dans lequel on trouve ses ouvrages.

Les données NE_EN, SALLE et RAYON dépendent fonctionnellement de la donnée AUTEUR dans la table AUTEURS1.

Mais il existe aussi des dépendances fonctionnelles entre attributs de la table AUTEURS1 :

un rayon appartient à une salle et une seule.

La connaissance du rayon implique celle de la salle (dépendance transitive entre AUTEUR et SALLE via RAYON).

Le fait de mélanger les données relatives aux rayons et celles relatives aux salles peut aussi être générateur d'erreurs de mise à jour.

Anomalies au DELETE

L'annulation dans la table AUTEURS1 des ouvrages écrits par l'auteur HUGO entraîne la perte de l'information : "la salle 2 contient le rayon 3", information indépendante de l'auteur HUGO.

Anomalies à l'INSERT

Il est impossible de créer une nouvelle salle ou un nouveau rayon si on n'a pas d'auteurs à y ranger (à moins d'introduire des données sans signification dans les colonnes AUTEUR et NE_EN d'une ligne).

Anomalies à l'UPDATE

L'instruction

```
UPDATE AUTEURS1
  SET SALLE = 4
  WHERE AUTEUR = 'ZOLA'
```

introduit une incohérence : le rayon 6 est
en salle 4 pour ZOLA
en salle 3 pour VERLAINE.

TROISIÈME FORME NORMALE ET BCNF

La solution consiste à créer deux tables à partir de la table AUTEURS1 :

- □ table AUTEURS2 comprenant les données AUTEUR, NE_EN et RAYON.

cette table ne contient que des données dépendant fonctionnellement d'AUTEUR

- □ une table RAYONS comprenant les données RAYON et SALLE.

cette table matérialise la dépendance fonctionnelle de SALLE par rapport à RAYON

Les deux tables sont en troisième forme normale.

TROISIÈME FORME NORMALE ET BCNF

Définitions Une relation est en troisième forme normale si et seulement si elle est en deuxième forme normale et si toutes les données non clé ne dépendent fonctionnellement que de la clé primaire.

Autre définition :

Forme normale de Boyce/Codd (ou BCNF pour Boyce/Codd Normal Form) .

Si on appelle déterminant une donnée ou une suite de données dont dépend fonctionnellement un attribut, on peut fournir la définition suivante:

Une relation est en forme normale de Boyce/Codd si et seulement si chaque déterminant est une clé candidate.

Troisième forme normale = minimum requis pour la conception des tables.

Définition

Tables = ensemble de lignes non ordonnées

Une instruction de lecture ou de mise à jour (SELECT, UPDATE, DELETE ou INSERT) comportant une condition de recherche peut entraîner un balayage complet de la table.

Index = moyen d'accélérer la recherche par un accès plus rapide aux données

De plus :

- possibilité de contrôler l'unicité de certaines valeurs
- possibilité de demander un ordre de rangement physique

Définitions

Index :

- structure d'accès bâtie sur une table à partir des données contenues dans une ou plusieurs colonnes de cette table
- ensemble ordonné de postes

Poste d'index :

- établit la correspondance entre une valeur et une ou plusieurs lignes de table contenant cette valeur
- **contient** :
 1. la valeur indexée (provenant d'une ou de plusieurs colonnes)
 2. un pointeur vers la ligne de table correspondante.

CREATE INDEX

Format

L'instruction CREATE INDEX permet de définir un index sur une table.

Format général simplifié

```
>> CREATE +-----+>
      +-TYPE+-1++    +-UNIQUE+-----++
              +-2-+                + WHERE NOT NULL +
>--INDEX--nom-d'index-CN--nom-de-table----->

      +-----+
      |         |
      |         |
      V         |
>--(---nom-de-colonne+-----+----)----->
                        |         |
                        +---DESC--+
```

Format

- Fonctionnement :

si des lignes existent dans la table,

construction immédiate ou différée (selon l'option DEFER) des entrées de l'index

index maintenu à chaque mise à jour (INSERT, UPDATE, DELETE)

présence d'index transparente pour les programmes (sauf si option UNIQUE) : seules les performances peuvent être affectées

CREATE INDEX

Exemples Créer un index sur la table LIVRES par ordre croissant d'auteur et de titre.

```
CREATE TYPE 2 INDEX I1LIVRES ON LIVRES  
(AUTEUR, TITRE)
```

Créer un index sur la table LIVRES par ordre décroissant de prix, par ordre croissant d'auteur et par ordre décroissant d'année de parution.

```
CREATE TYPE 2 INDEX I2LIVRES ON LIVRES  
(PRIX DESC, AUTEUR ASC, ANNEE DESC)
```

Préciser type 2 si vous ne connaissez pas l'option par défaut qui dépend de l'installation de DB2.

L'option UNIQUE

L'option UNIQUE permet de garantir que deux lignes ne contiennent pas des valeurs identiques dans une ou plusieurs colonnes précises.

contrôle effectué à chaque opération de chargement ou de mise à jour

s'applique également sur une table ayant des lignes : pas de constitution d'index si doublons

Si l'option WHERE NOT NULL est omise :

NULL se comporte comme toute autre valeur

Si l'option WHERE NOT NULL est utilisée :

Deux occurrences de la valeur NULL sont considérées comme des valeurs différentes

On ne peut spécifier WHERE NOT NULL qu'avec les INDEX de type 2 définis comme tel explicitement ou par défaut.

- Le type de l'index par défaut dépend des paramètres d'installation de DB2 (DSNTIPE) et du paramètre LOCKSIZE de création du TABLESPACE

LES INDEX

Exemples Créer, sur la table LIVRES, un index assurant l'unicité des couples auteur-titre.

```
CREATE TYPE 2 UNIQUE INDEX I3LIVRES  
ON LIVRES  
(AUTEUR ASC, TITRE ASC)
```

Résultat : **l'index est construit.**

Créer, sur la table LIVRES, un index assurant l'unicité du prix lorsque celui ci est connu.

```
CREATE TYPE 2 UNIQUE WHERE NOT NULL  
INDEX I4LIVRES  
ON LIVRES  
(PRIX)
```

Résultat : *la colonne PRIX contient deux fois la valeur 80.00 :*

l'index n'est pas construit.

L'Option CLUSTER

L'option CLUSTER permet d'indiquer la séquence physique de rangement des lignes de la table (objectif : performances)

Ainsi, dans un ordre CREATE INDEX avec l'option CLUSTER, le nom de la ou des colonnes définissent la séquence de rangement choisie

Pas plus d'un index CLUSTER par table

CLUSTER ne garantit pas l'ordre : si pas de place disponible à l'INSERT, ligne placée le plus près possible de son emplacement idéal

Si création d'un index CLUSTER sur une table contenant des données, reclassement au rechargement ou à la réorganisation.

Exemple Créer, sur la table LIVRES, un index assurant à la fois l'unicité des couples auteur-titre et l'ordre de rangement selon cette même séquence.

```
CREATE TYPE 2 UNIQUE INDEX I5LIVRES  
ON LIVRES  
(AUTEUR ASC, TITRE ASC)  
CLUSTER
```

Résultat :

L'ordre de rangement physique des lignes de la table s'effectuera par auteur et titre croissants à la prochaine réorganisation.

RÔLE ET UTILISATION DES INDEX

Possibilités Les index ont une importance considérable dans un système relationnel :

caractéristiques logiques :

option UNIQUE seul moyen d'éviter les doubles, et de garantir qu'une donnée est un identifiant; à utiliser pour les clés primaires

caractéristiques physiques

:

option CLUSTER pour demander un ordre de rangement, et minimiser les entrées-sorties

utilisation par le système pour choisir la stratégie d'accès

Mais

chaque index entraîne un coût supplémentaire de gestion.

Importance du choix de bons index (dans la conception et le suivi des bases)

Si tentative d'insertion d'une clef en double vous serez avertit par un SQLCODE négatif.

7. Manipulation de plusieurs tables.

OBJECTIFS

Savoir utiliser l'ordre SELECT pour obtenir des résultats à partir d'informations issues de différentes tables.

- Format complet de l'ordre SELECT
- Notion de jointure de tables
- Nom de corrélation
- Notion de sous-requêtes corrélées
- Notion de jointure externe
- Expressions de tables imbriquées
- Règles d'intégrité

EXEMPLES DE TABLES

Table LIVRES

AUTEUR	TITRE	ANNEE	GENRE	PRIX
HUGO	HERNANI	1830	THEATRE	120.00
HUGO	LES CONTEMPLATIONS	1856	POESIE	78.50
HUGO	LES MISERABLES	1862	ROMAN	148.50
BALZAC	EUGENIE GRANDET	1833	ROMAN	100.00
BALZAC	LE PERE GORIOT	1834	ROMAN	-
DUMAS	LES TROIS MOUSQUETAIRES	1844	ROMAN	80.00
DUMAS	VINGT ANS APRES	1845	ROMAN	80.00
DUMAS	LA DAME AUX CAMELIAS	1852	THEATRE	110.00
STENDHAL	LE ROUGE ET LE NOIR	1831	ROMAN	98.50
STENDHAL	LA CHARTREUSE DE PARME	1839	ROMAN	110.50
FLAUBERT	MADAME BOVARY	1857	ROMAN	99.50
VERLAINE	POEMES SATURNIENS	1866	POESIE	90.50

EXEMPLES DE TABLES

Table ECRIVAINS

Clé primaire :
AUTEUR

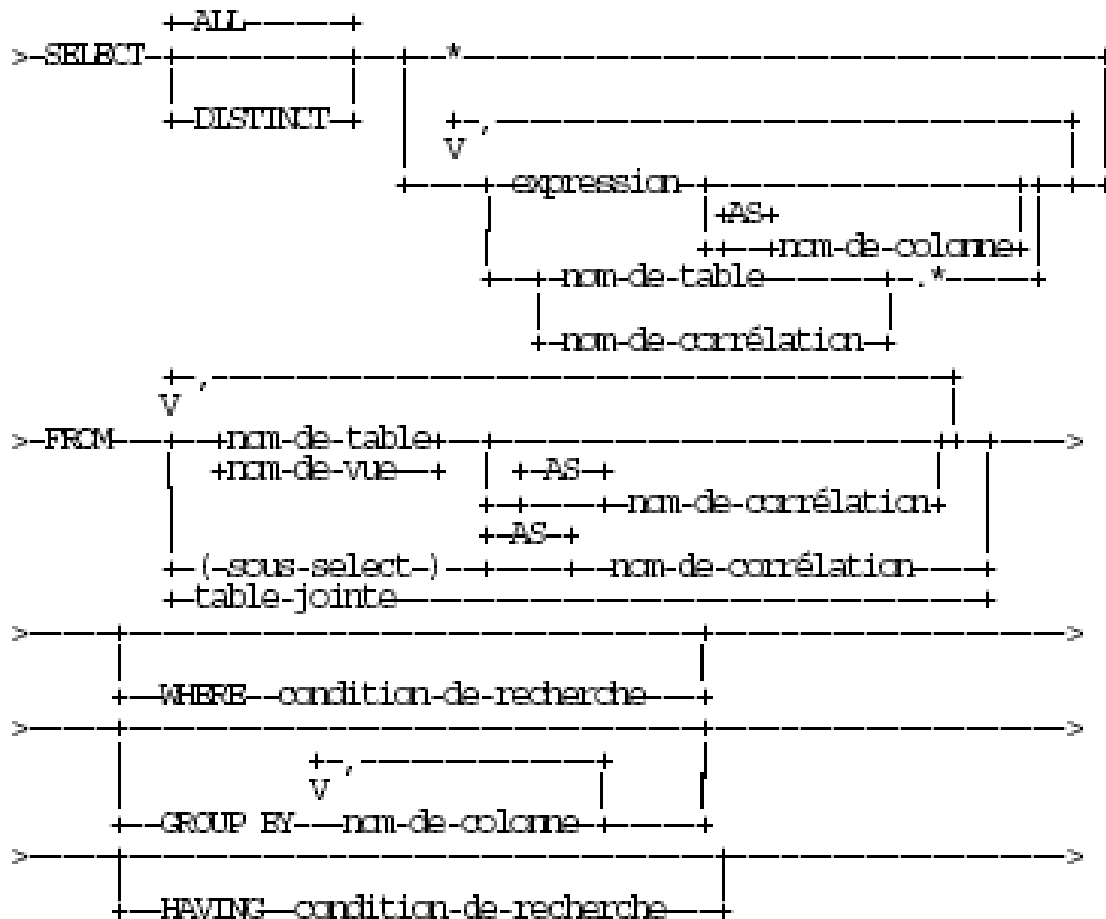
AUTEUR	NE_EN	LIEU	RAYON
HUGO	1802	BESANCON	3
BALZAC	1799	TOURS	1
DUMAS	1802	VILLERS-COTTERETS	1
DUMAS	1824	PARIS	1
STENDHAL	1783	GRENOBLE	5
FLAUBERT	1821	ROUEN	2
VERLAINE	1844	METZ	6
ZOLA	1840	PARIS	6
RIMBAUD	1854	CHARLEVILLE	5
SAND	1804	PARIS	5

TABLE RAYONS

Clé primaire :
RAYON

RAYON	SALLE
1	1
2	1
3	2
4	2
5	3
6	3

SOUS SELECT



OPTION FROM AVEC PLUSIEURS TABLES

L'instruction **SELECT** permet d'indiquer plusieurs tables après le mot clé **FROM**.

```
SELECT TITRE, GENRE, RAYON  
FROM LIVRES, ECRIVAINS
```

TITRE et **GENRE** : table **LIVRES**
RAYON : table **ECRIVAINS**

Mais il y a ambiguïté si des colonnes ont le même nom dans différentes tables.

OPTION FROM AVEC PLUSIEURS TABLES

Exemple

L'exemple suivant n'est pas syntaxiquement correct.

```
SELECT AUTEUR, TITRE, GENRE  
FROM LIVRES, ECRIVAINS
```

TITRE et **GENRE** : table **LIVRES**

AUTEUR : table **LIVRES** ou table **ECRIVAINS** ???

Il faut qualifier chaque nom de colonne susceptible de provoquer une ambiguïté.

```
SELECT LIVRES.AUTEUR, TITRE, GENRE  
FROM LIVRES, ECRIVAINS
```

La colonne **AUTEUR** est celle de la table **LIVRES**

NOM DE CORRÉLATION

Définition

On peut utiliser un **nom de corrélation** pour préfixer les noms de colonnes à la place du nom de table.

Identificateur long suivant chaque nom de table après FROM.

Valable uniquement dans l'instruction où il est cité.

Exemple

```
SELECT A.AUTEUR, B.AUTEUR,  
       B.TITRE, B.GENRE  
FROM LIVRES A, ECRIVAINS B
```

SOUS-REQUÊTES CORRÉLÉES

Rappel

On peut utiliser une sous-requête à l'intérieur d'une condition de recherche (WHERE).

Il y a d'abord exécution de la sous-requête puis son résultat est exploité dans la condition de recherche.

On parle de **sous-requête corrélée** quand la sous-requête fait référence à une colonne figurant à un niveau supérieur de requête.

Dans ce cas, la sous-requête est exécutée plusieurs fois
Une fois par ligne de la requête supérieure.

SOUS-REQUÊTES CORRÉLÉES

Exemple Lister l'auteur, le titre et l'année de parution des ouvrages de la table LIVRES écrits après l'année maximum de naissance des auteurs de la table ECRIVAINS, dont le nom est inférieur au nom de l'auteur courant.

```
SELECT AUTEUR, TITRE, ANNEE
FROM LIVRES
WHERE ANNEE >
      (SELECT MAX(NE_EN) FROM ECRIVAINS
       WHERE AUTEUR < LIVRES.AUTEUR )
```



MAX(NE_EN) calculé autant de fois qu'il y a de lignes dans la table LIVRES.

AUTEUR	TITRE	ANNEE
VERLAINE	POEMES SATURNIENS	1866
FLAUBERT	MADAME BOVARY	1857
DUMAS F	LA DAME AUX CAMELIAS	1852
DUMAS P	VINGT ANS APRES	1845
DUMAS P	LES TROIS MOUSQUETAIRES	1844
BALZAC	LE PERE GORIOT	1834
BALZAC	EUGENIE GRANDET	1833
HUGO	LES MISERABLES	1862
HUGO	LES CONTEMPLATIONS	1856
HUGO	HERNANI	1830

SOUS REQUÊTES CORRÉLÉES

Avec l'option **GROUP BY**

Exemple Lister le nom et l'année de parution du dernier ouvrage des auteurs de la table LIVRES ayant écrit leur dernier ouvrage après la moyenne de la date de naissance des auteurs de la table ECRIVAINS hormis cet écrivain.

```
SELECT AUTEUR, MAX(ANNEE)
  FROM LIVRES X
   GROUP BY AUTEUR
        HAVING MAX(ANNEE) >
        (SELECT AVG(NE_EN) FROM ECRIVAINS
         WHERE NOT AUTEUR = X.AUTEUR )
```

Résultat

AUTEUR	
BALZAC	1834
DUMAS F	1852
DUMAS P	1845
FLAUBERT	1857
HUGO	1862
STENDHAL	1839
VERLAINE	1866

SOUS-REQUÊTES CORRÉLÉES

Avec EXISTS

Exemple Lister le nom des auteurs et le titre des ouvrages de la table LIVRES écrits la même année que l'année de naissance d'un des auteurs de la table ECRIVAINS.

```
SELECT AUTEUR, TITRE
FROM LIVRES X
WHERE EXISTS
  (SELECT 1 FROM ECRIVAINS
   WHERE NE_EN = X.ANNEE)
```

AUTEUR	TITRE
-----	-----
DUMAS P	LES TROIS MOUSQUETAIRES

La fonction EXISTS

Est évaluée pour chaque valeur de l'année de parution
Ne renvoie la valeur "vrai" que lorsqu'elle trouve au moins
une ligne satisfaisant la condition de recherche

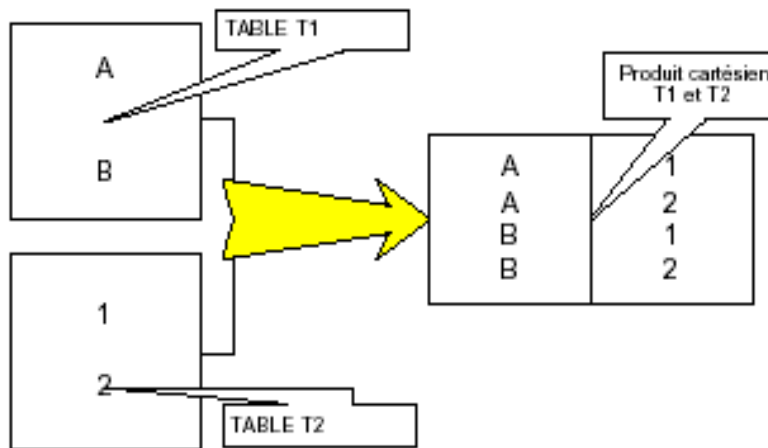
NOTION DE JOINTURE

Jointure

Sélection, par un seul ordre **SELECT**, de colonnes provenant de plusieurs tables.

Produit CARTESIEN

Le produit cartésien s'obtient par une jointure sans condition de recherche.



PRODUIT CARTÉSIEN

Exemple

Effectuer une jointure entre les auteurs de la table LIVRES et ceux de la table ECRIVAINS

```
SELECT A.AUTEUR, B.AUTEUR
FROM LIVRES A, ECRIVAINS B
```

Résultat

AUTEUR	AUTEUR
-----	-----
HUGO	HUGO
HUGO	BALZAC
HUGO	DUMAS P
HUGO	DUMAS F
.....	
VERLAINE	RIMBAUD
VERLAINE	SAND

Peut être extrêmement pénalisant en performances.
Appliquer des conditions de recherche exploitant des données communes aux tables jointes pour réduire le nombre de lignes résultantes.

EQUI-JOINTURE

Exemple

La mise en correspondance de plusieurs tables sur l'égalité d'un critère est appelée équi-jointure.

Lister le titre, l'auteur et le rayon de rangement des ouvrages de la bibliothèque.

```
SELECT A.AUTEUR, A.TITRE, B.RAYON
FROM LIVRES A, ECRIVAINS B
```

Résultat

AUTEUR	TITRE	RAYON
-----	-----	-----
BALZAC	EUGENIE GRANDET	1
BALZAC	LE PERE GORIOT	1
DUMAS F	LA DAME AUX CAMELIAS	1
DUMAS P	LES TROIS MOUSQUETAIRES	1
DUMAS P	VINGT ANS APRES	1
FLAUBERT	MADAME BOVARY	2
HUGO	HERNANI	3
HUGO	LES CONTEMPLATIONS	3
HUGO	LES MISERABLES	3
STENDHAL	LE ROUGE ET LE NOIR	5
STENDHAL	LA CHARTREUSE DE PARME	5
VERLAINE	POEMES SATURNIENS	6

JOINTURE DE PLUSIEURS TABLES

Exemple

On peut réaliser la jointure de plus de deux tables. jointure, de façon interne, de deux tables puis, à partir de la table intermédiaire ainsi obtenue, jointures successives, jusqu'à aboutir à la table résultante finale.

Reconstituer la table OUVRAGES à partir des tables LIVRES, ECRIVAINS et RAYONS.

```
SELECT X.AUTEUR, X.TITRE, Y.NE_EN, Z.SALLE, Z.RAYONS
FROM LIVRES X, ECRIVAINS Y, RAYON Z
WHERE X.AUTEUR = Y.AUTEUR
AND Y.RAYON = Z.RAYON
```

Résultat

AUTEUR	TITRE	NE_EN	SALLE	RAYON
-----	-----	-----	-----	-----
BALZAC	EUGENIE GRANDET	1799	1	1
BALZAC	LE PERE GORIOT	1799	1	1
DUMAS F	LA DAME AUX CAMELIAS	1824	1	1
DUMAS P	LES TROIS MOUSQUETAIRES	1802	1	1
DUMAS P	VINGT ANS APRES	1802	1	1
FLAUBERT	MADAME BOVARY	1821	1	2
HUGO	HERNANI	1802	2	3
HUGO	LES CONTEMPLATIONS	1802	2	3
HUGO	LES MISERABLES	1802	2	3
STENDHAL	LE ROUGE ET LE NOIR	1783	3	5
STENDHAL	LA CHARTREUSE DE PARME	1783	3	5
VERLAINE	POEMES SATURNIENS	1844	3	6

JOINTURE NATURELLE

Définition

Une **jointure naturelle** est une équi-jointure dans laquelle la table résultante ne comporte qu'une seule fois le domaine commun (et pas de lignes en double)

Exemple

Effectuer la **jointure naturelle** de la table LIVRES et de la table ECRIVAINS sur l'égalité de la donnée AUTEUR en listant l'auteur et le rayon.

```
SELECT DISTINCT A.AUTEUR, B.RAYON
FROM LIVRES A, ECRIVAINS B
WHERE A.AUTEUR = B.AUTEUR
```

Résultat

AUTEUR	RAYON
-----	-----
BALZAC	1
DUMAS F	1
DUMAS P	1
FLAUBERT	2
HUGO	3
STENDHAL	5
VERLAINE	6

L'option DISTINCT permet d'éliminer les doubles.

JOINTURE AVEC CONDITIONS QUELCONQUES

Exemple : Lister les auteurs nés après l'écriture des pièces de théâtre, ou, plus précisément, lister les données GENRE, ANNEE, NE_EN, AUTEUR de la table LIVRES et AUTEUR de la table ÉCRIVAINS obtenues par jointure sur la condition année de naissance supérieure à année d'écriture en restreignant au genre THEATRE.

```
SELECT A.GENRE, A.ANNEE, B.NE_EN,  
A.AUTEUR, B.AUTEUR  
FROM LIVRES A, ECRIVAINS B  
WHERE B.NE_EN > A.ANNEE  
AND A.GENRE = 'THEATRE'
```

Résultat

GENRE	ANNEE	NE_EN	AUTEUR	AUTEUR
THEATRE	1830	1844	HUGO	VERLAINE
THEATRE	1830	1840	HUGO	ZOLA
THEATRE	1830	1854	HUGO	RIMBAUD
THEATRE	1852	1854	DUMAS	F RIMBAUD

JOINTURE D'UNE TABLE SUR ELLE-MÊME

Exemple : Pour rapprocher plusieurs lignes d'une table entre elles, on peut faire figurer plusieurs fois la même table dans un ordre SELECT.

Dans ce cas, les noms de corrélation sont indispensables. Lister le nom et la date de naissance des écrivains nés la même année.

```
SELECT A.AUTEUR, A.NE_EN, B.AUTEUR
FROM ECRIVAINS A, ECRIVAINS B
WHERE A.NE_EN = B.NE_EN
```

Résultat :

AUTEUR	NE_EN	AUTEUR
STENDHAL	1783	STENDHAL
BALZAC	1799	BALZAC
HUGO	1802	HUGO
HUGO	1802	DUMAS P
DUMAS P	1802	HUGO
DUMAS P	1802	DUMAS P
SAND	1804	SAND
FLAUBERT	1821	FLAUBERT
DUMAS F	1824	DUMAS F
...

La table précédente comporte des doubles et des informations redondantes que l'on peut éliminer en écrivant :

```
SELECT A.AUTEUR, A.NE_EN, B.AUTEUR
FROM ECRIVAINS A, ECRIVAINS B
WHERE A.NE_EN = B.NE_EN
AND A.AUTEUR < B.AUTEUR
```

Résultat :

AUTEUR	NE_EN	AUTEUR
DUMAS P	1802	HUGO

JOINTURE EXTERNE

Exemple : On souhaite écrire une requête qui permette d'associer à chaque auteur, les ouvrages ayant été écrits l'année de sa naissance.
Une jointure entre les tables LIVRES et ECRIVAINS laisse apparaître la seule valeur commune aux colonnes NE_EN et ANNEE :

```
SELECT A.AUTEUR, A.NE_EN, B.ANNEE, B.TITRE
FROM ECRIVAINS A, LIVRES B
WHERE A.NE_EN = B.ANNEE
```

Résultat :

AUTEUR	NE_EN	ANNEE	TITRE
VERLAINE	1844	1844	LES TROIS MOUSQUETAIRES

On appelle ce type d'opération une **jointure interne**.

JOINTURE EXTERNE


Le principe de la jointure exclut certains ouvrages, parce qu'aucun auteur n'est né au cours de leur année de parution. Si on veut voir apparaître ces ouvrages, il faut associer la jointure avec une requête sur LIVRES qui exclut cette fois les ouvrages pris en compte par la jointure.

Exemple

```
SELECT A.AUTEUR, A.NE_EN, B.ANNEE, B.TITRE
  FROM ECRIVAINS A, LIVRES B
    WHERE A.NE_EN = B.ANNEE
  UNION ALL
    SELECT '-----', 0, ANNEE, TITRE
  FROM LIVRES A
  WHERE NOT EXISTS
    (SELECT 1 FROM ECRIVAINS B
     WHERE B.NE_EN=A.ANNEE)
```

Résultat

AUTEUR	NE_EN	ANNEE	TITRE
VERLAINE	1844	1844	LES TROIS MOUSQUETAIRES
-----	0	1830	HERNANI
-----	0	1856	LES CONTEMPLATIONS
-----	0	1862	LES MISERABLES
-----	0	1833	EUGENIE GRANDET
-----	0	1834	LE PERE GORIOT
-----	0	1845	VINGT ANS APRES
-----	0	1852	LA DAME AUX CAMELIAS
-----	0	1831	LE ROUGE ET LE NOIR
-----	0	1839	LA CHARTREUSE DE PARME
-----	0	1857	MADAME BOVARY
-----	0	1866	POEMES SATURNIENS

 pour le deuxième sous-select de l'UNION, il faut utiliser une valeur de substitution ('-----',0) pour les colonnes AUTEUR et NE_EN.

SELECT DANS LA CLAUSE FROM

Définition

Les opérandes de la clause FROM peuvent être plus complexes qu'un simple nom de table.

Une expression de table imbriquée correspond à l'utilisation d'un sous-select dans la clause FROM.

Il est obligatoire d'associer un nom de corrélation à chaque sous select.

Exemple : Lister le prix moyen de chaque genre d'ouvrage de la table LIVRE

```
SELECT AVG(PRIX) AS PRIXMOYEN, GENRE
FROM LIVRES
GROUP BY GENRE
```

PRIXMOYEN	GENRE
84,5000000000000	POESIE
102,428571428571	ROMAN
115,0000000000000	THEATRE

SELECT DANS LA CLAUSE FROM

Exemple Trouver la valeur la plus élevée du prix moyen par genre d'ouvrage

```
SELECT MAX(TEMP.PRIXMOYEN) AS MAXPRIXMOYEN
FROM (SELECT AVG(PRIX) AS PRIXMOYEN, GENRE
FROM LIVRES
GROUP BY GENRE) AS TEMP
```

Résultat :

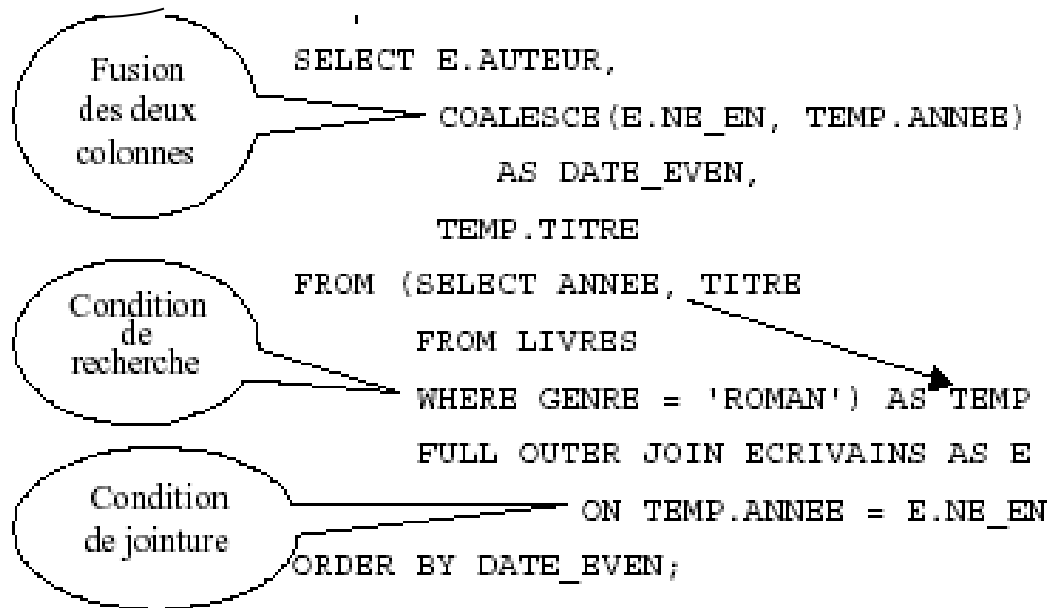
```
MAXPRIXMOYEN
-----
115,000000000000
```

1. AVG(PRIX) devient PRIXMOYEN
2. PRIXMOYEN devient TEMP.PRIXMOYEN
3. TEMP.PRIXMOYEN devient MAXPRIXMOYEN

EXEMPLE DE SYNTHÈSE

Lister le nom des auteurs de la table ECRIVAINS, le titre des romans de la table LIVRES, en associant les lignes de chacune de ces deux tables sur l'égalité de l'année de naissance et de l'année de parution.

On présentera le résultat trié suivant le critère de jointure.



EXEMPLE DE SYNTHÈSE

Résultat

AUTEUR	DATE Even	TITRE
STENDHAL	1783	
BALZAC	1799	
HUGO	1802	
DUMAS P	1802	
SAND	1804	
FLAUBERT	1821	
DUMAS F	1824	
	1831	LE ROUGE ET LE NOIR
	1833	EUGENIE GRANDET
	1834	LE PERE GORIOT
	1839	LA CHARTREUSE DE PARME
ZOLA	1840	
VERLAINE	1844	LES TROIS MOUSQUETAIRES
	1845	VINGT ANS APRES
RIMBEAU	1854	
	1857	MADAME BOVARY
	1862	LES MISERABLES

👉 la fonction COALESCE permet de fusionner
NE_EN et ANNEE et d'obtenir ainsi une jointure naturelle.

INTÉGRITÉ D'ENTITÉ ET DE RÉFÉRENCE

Définition

Pour que les informations obtenues en rapprochant plusieurs tables soient valides, il faut que la cohérence des données soit assurée.

Deux règles d'intégrité remplissent cette fonction.

Intégrité d'entité

- ☐ Assurée quand il n'est pas possible d'accepter la valeur NULL pour une clé primaire.

Intégrité de référence

- ☐ Assurée quand toute clé étrangère correspond à une clé primaire existant dans la table de référence.

Ces règles d'intégrité sont prises en charge par DB2 grâce à des options du CREATE TABLE.

CLÉ PRIMAIRE ET CLÉ ÉTRANGÈRE

Définitions

Clé primaire

Identifie une ligne de table de manière **unique**.

Il existe une seule clé primaire par table.

On appelle **Table Parente**, une table comportant une clé primaire référencée par une clé étrangère.

Référence la clé primaire d'une table.

Clé étrangère

Il peut exister plusieurs clés étrangères pour une table.

On appelle **Table Dépendante**, une table comportant une clé étrangère.

Clé étrangère = foreign key

Clé primaire = primary key

CLÉ PRIMAIRE ET CLÉ ÉTRANGÈRE

Clé primaire

Table RAYONS

RAYON	SALLE
1	1
2	1
3	2
4	2
5	3
6	3

Clé Primaire

Clé étrangère

Table écrivains

AUTEUR	NE_EN	LIEU	RAYON
HUGO	1802	BESANCON	3
BAI ZAC	1799	TOURS	1
DUMAS	1802	VILLERS-COTTERETS	1
DUMAS	1824	PARIS	1
STENDHAL	1783	GRENOBLE	5
FLAUBERT	1821	ROUEN	2
VERLAINE	1844	METZ	6
ZOLA	1840	PARIS	6
RIMBAUD	1854	CHARLEVILLE	5
SAND	1804	PARIS	5

CREATE TABLE AVEC CLÉ PRIMAIRE ET CONTRAINTE DE RÉFÉRENCE

Format général

```

>>--select-complet----->>
|          +-,------+ |
|          |          +-ASC--+ | |
|          V          |      | |
+-ORDER BY+-nom-de-colonne+-+-----+--+
|          |      |      |
+-entier-----+ +-DESC+

```

Select-complet

```

          +-----+-----+
          V                                     |
>+---scus-select-----+--+-----+--+>
|                       |   |                       |
+- (select-complet)-+ ++-UNION-----+--scus-select-----++
|                       |   |                       |
      +-UNION ALL--+ +- (select-complet)-+

```

CREATE TABLE AVEC CLÉ PRIMAIRE ET CONTRAINTE DE RÉFÉRENCE

Option Primary Key

- désigne la clé primaire de la table
- constituée de 1 à 64 colonnes
- chaque colonne de la clé primaire doit être définie avec l'option NOT NULL
- pour que la définition de la table soit complète, il faut créer un index unique associé à la clé primaire.

```
CREATE TABLE RAYONS
(RAYON SMALLINT NOT NULL,
SALLE SMALLINT,
PRIMARY KEY(RAYON));
CREATE UNIQUE INDEX IXRAY
ON RAYONS (RAYON)
```

CREATE TABLE AVEC CLÉ PRIMAIRE ET CONTRAINTE DE RÉFÉRENCE

FOREIGN KEY

- désigne une clé étrangère de la table
- la liste des colonnes doit correspondre à la définition de la clé primaire de la table parente (nombre de colonnes, type, longueur)
- les colonnes de la clé étrangère peuvent admettre des nuls
- le nom de contrainte est optionnel (s'il est omis, il sera déterminé par DB2)
- la construction d'un index associé à la clé étrangère est conseillée.

REFERENCES

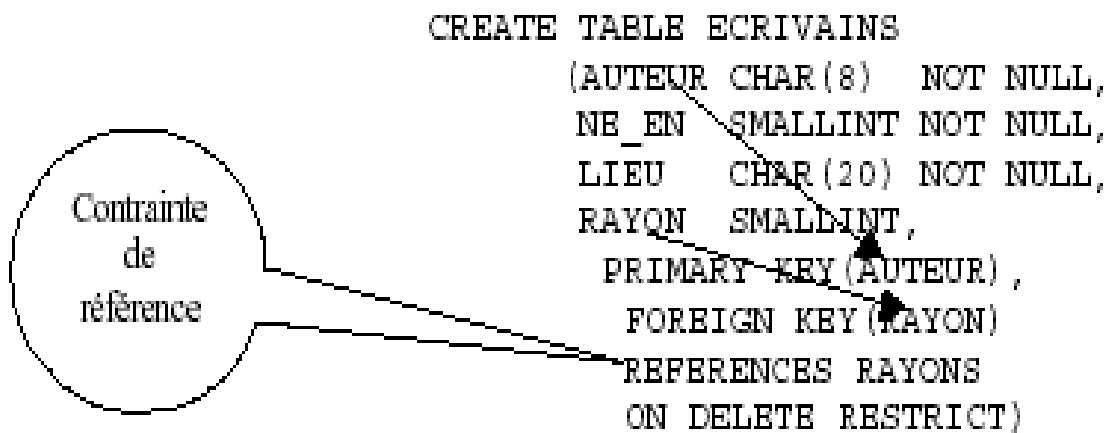
- désigne la table parente
- désigne les colonnes référencées dans la clé étrangère. Ces colonnes ne sont pas nécessairement clé primaire de la table parente. Seul un index unique est requis sur les colonnes contenues dans le mot clé REFERENCES. S'il n'y a pas ou plus d'index unique le statut de la table est changé à « incomplete »

ON DELETE

- précise la règle à appliquer en cas de suppression d'une valeur de clé primaire dans la table parente : que faire des lignes dépendantes ?

CREATE TABLE AVEC CLÉ PRIMAIRE ET CONTRAINTE DE RÉFÉRENCE

EXEMPLE



Fonctionnement

En cas d'ajout d'une valeur de clé étrangère dans la table dépendante (par INSERT ou UPDATE) :

- la nouvelle valeur de clé étrangère (si elle n'est pas nulle) doit correspondre à une clé primaire de la table parente.

C'est une règle implicite.

L'ordre INSERT suivant ne sera pas réalisé car le rayon 7 n'existe pas dans la table RAYONS.

Exemple

```
INSERT INTO ECRIVAIN  
(AUTEUR, NE_EN, LIEU, RAYON)  
VALUES  
( 'LEROUX', 1862, 'PARIS', 7)
```

En cas de suppression d'une valeur de clé primaire dans la table parente.

Si la clé primaire supprimée correspond à une ou des clés étrangères dans une table dépendante :

La règle à respecter se définit de manière explicite dans l'ordre CREATE TABLE de chaque table dépendante.

quatre options possibles

- **RESTRICT** (Option par défaut)

Une valeur définie en tant que clé primaire ne pourra être supprimée que lorsqu'elle n'aura plus de dépendants.

- **CASCADE**

La suppression d'une clé primaire se propage sur tous ses dépendants.

- **SET NULL**

La suppression d'une clé primaire entraîne le positionnement des clés étrangères des dépendants à la valeur NULL. (Les nuls doivent être autorisés)

NO ACTION

Dans tous les cas, à une différence près cette règle de suppression fournit le même résultat que l'option

RESTRICT

La différence entre RESTRICT et NO ACTION se situe lors de la mise en œuvre de la règle de suppression.

L'option NO ACTION est mise en œuvre à la fin de l'instruction, tandis que RESTRICT est mis en œuvre immédiatement.

Cas où il existe une différence en terme de résultat :
Lorsque un DELETE met en jeu une contrainte auto référencée (i-e : la table parente et la table dépendante sont les mêmes).

NO ACTION autorise la suppression tandis que RESTRICT l'empêche.

exemples On désire supprimer le rayon 6 de la table RAYONS.

```
DELETE FROM RAYONS  
WHERE RAYON = 6
```

- ☐ Avec l'option RESTRICT ou No action:

```
CREATE TABLE ECRIVAINS  
(AUTEUR CHAR(8) NOT NULL,  
NE_EN SMALLINT NOT NULL,  
LIEU CHAR(20) NOT NULL,  
RAYON SMALLINT,  
PRIMARY KEY(AUTEUR),  
FOREIGN KEY(RAYON)  
REFERENCES RAYONS  
ON DELETE RESTRICT)
```

L'ordre DELETE n'est pas réalisé car le Rayon 6 est référencé dans la table ECRIVAINS.

Exemples Avec l'option **CASCADE** :

```
CREATE TABLE ECRIVAINS  
...  
...  
ON DELETE CASCADE)
```

L'ordre DELETE est réalisé :
Le rayon 6 est supprimée de la table RAYONS.
Les lignes de la table ECRIVAINS faisant référence au rayon 6 sont aussi supprimées.

Avec l'option **SET NULL** :

```
CREATE TABLE ECRIVAINS  
...  
...  
ON DELETE SET NULL)
```

L'ordre DELETE est réalisé :
Le rayon 6 est supprimée de la table RAYONS.
Les colonnes RAYON de la table ECRIVAINS faisant référence au rayon 6 sont mises à la valeur NULL.

INTÉGRITÉ DE RÉFÉRENCE : ORDRE DE CRÉATION DE TOUS LES OBJETS

Si l'on définit des règles d'intégrité référentielle, les objets doivent être décrits dans l'ordre suivant :

1. Table Parente
2. Index Primaire
3. Table Dépendante
4. Eventuellement, index sur la clé étrangère

Le chargement des Tables doit impérativement se faire dans l'ordre suivant :

1. Table Parente
2. Table Dépendante

CREATE TABLE AVEC CONTRAINTE DE VÉRIFICATION

Contrainte de vérification

```
+--CONSTRAINT--nom-de-contrainte--+  
>--CHECK-(condition de vérification)--><
```

Clause CHECK

- permet de définir une règle de contrôle sur la valeur des colonnes.
- Contrôle effectué par DB2 lors des insertions et des mises à jour de la table

CREATE TABLE AVEC CONTRAINTE DE VÉRIFICATION

Définition

La condition de vérification s'exprime sous la forme d'une condition de recherche et :

- ne se réfère qu'aux colonnes de la table
- ne peut contenir :
 - ni sous-select
 - ni fonction sur colonne
 - ni variable hôte
 - ni colonne associée à une FIELDPROC
 - ni prédicat quantifié (SOME, ANY ou ALL)
 - ni prédicat EXISTS

Exemple de table avec contrainte de vérification

```
CREATE TABLE ECRIVAINS
(AUTEUR CHAR(8) NOT NULL,
  NE_EN SMALLINT NOT NULL,
  LIEU CHAR(20) NOT NULL,
  RAYON SMALLINT)
  CONSTRAINT VERIFNE_EN (Contrainte de vérification)
    CHECK (NE_EN BETWEEN 1600 AND 2000)
```