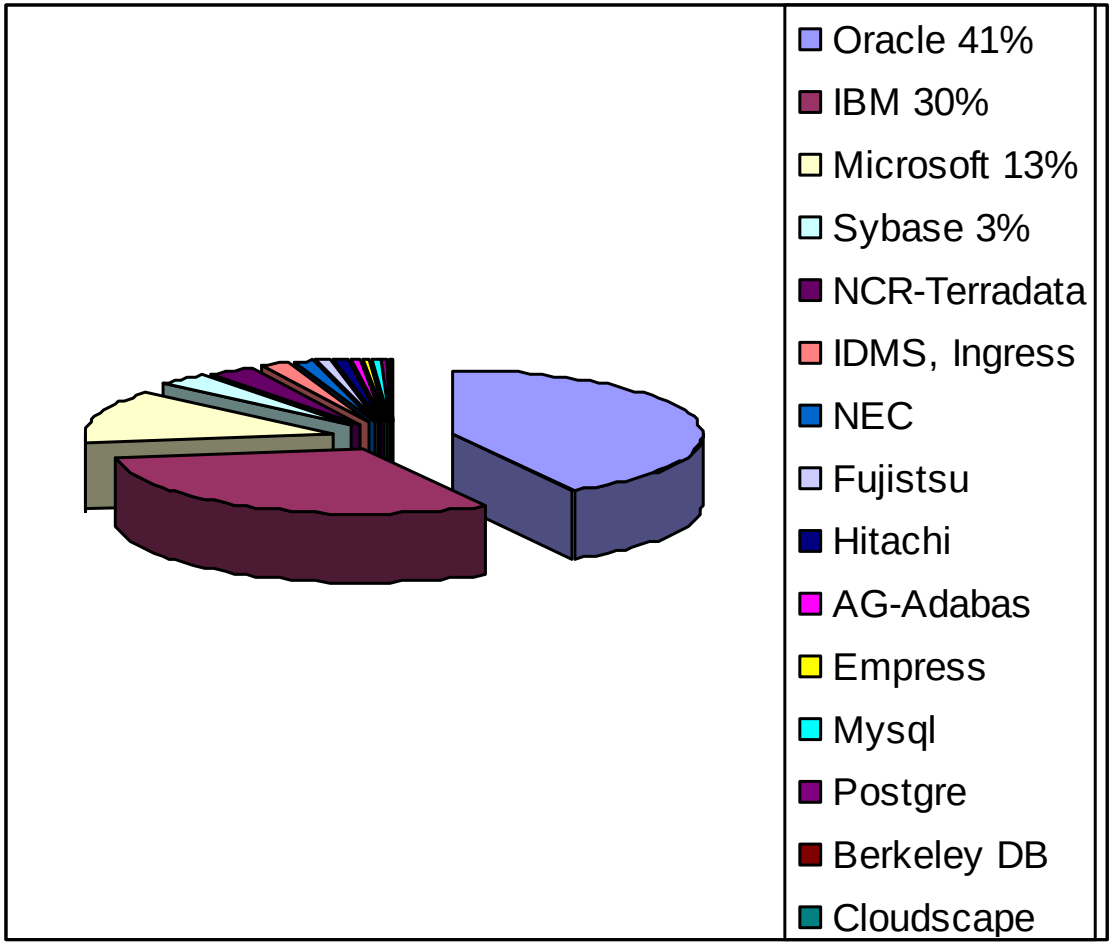
Université du Québec
École de technologie supérieure

Introduction et revue du langage SQL

Survol du cours

- Introduction : le modèle relationnel (4.1)
- Création de schéma d'une table et contraintes d'intégrité (4.1)
- Sélection de données (5.1.1)
- Modification de données (pp.79 à 82)
- Types de données MM en relationnel (2.1.2)

Marché des BD en 2006



Total: $14.9 Billion

Source: IDC, 1995

MySQL

- Fonctionnalités:
  - Pas de librairie multimédia (juste BLOB, Text)
  - Usage général, JDBC, ODBC et .NET

SQL Server - MS

- Solutions: Entreprise, Standard, Workgroup,express, developer, mobile, 64 bits
  - www.microsoft.com/sql
- Fonctionnalités:
  - Pas de librairie multimedia (juste BLOB, Text)
  - XML et Xquery
  - Mobile
  - BI

Access - MS

- Fonctionnalités:
  - Pas de librairie multimedia (juste Blob, Text)
  - Joindre un image
  - Liens à Sharepoint

PostgreSQL – U of Berkeley

- Coût: Logiciel libre, Gratuit
  - www.postgresql.org
- Fonctionnalités:
  - Pas de librairie multimédia (juste LOB)
  - Usage général
  - Relationnel-Objet
  - Géospatial (types géométriques)

DB2 - IBM

- Solutions: DB2-9
  - http://www-306.ibm.com/software/data/db2/udb/
- Librairie Multimédia (SQL2, SQL3,Text, Image, Video, XML, Data Warehouse, Géospatial, RFID,..)

Oracle

- Fonctionnalités:
  - Relationel-Objet
  - Librairie Multimédia (SQL2, SQL3,Text, Image, Video, XML, 3Dmaps, Data Warehouse, Géospatial, RFID, …)

Sun to Acquire MySQL

- Santa Clara, CA – January 16, 2008
  - *Sun Microsystems, Inc. today announced it has entered into a definitive agreement to acquire MySQL AB, an open source icon and developer of one of the world's fastest growing open source databases for approximately $1 billion in total.*

http://www.mysql.com/news-and-events/sun-to-acquire-mysql.html

Oracle Buys Sun

- Redwood Shores, CA - April 20, 2009
  - *Oracle Corporation (NASDAQ: ORCL) and Sun Microsystems (NASDAQ: JAVA) announced today they have entered into a definitive agreement under which Oracle will acquire Sun common stock for $9.50 per share in cash. The transaction is valued at approximately $7.4 billion, or $5.6 billion net of Sun's cash and debt.*

http://www.oracle.com/us/corporate/press/018363

Incassable?

http://www.liquidmatrix.org/blog/wp-content/uploads/2009/04/unbreakable-right.png

Introduction aux BD relationnelles

Modèle relationnel – concepts

- Types atomiques : littéraux, nombres,dates
- Domaine : sous-ensemble de valeurs d'un type atomique

http://www.crescenzo.nom.fr/CMBasesDeDonnees/003-ModeleRelationnel-Concepts.html

Modèle relationnel – concepts

- Table = ensemble nommé de tuples de même structure (cardinalité identique, domaines identiques deux à deux)
- Colonne : l'un des domaines de la table
- Rangée : l'un des tuples de la table

Modèle relationnel – concepts

- Toutes les données manipulées dans le modèle relationnel doivent être formulées sous forme de tables.
- Opérations relationnelles = manipulations de tables
- SQL = langage déclaratif et ensembliste
- Séparation structure / contenu

Création du schéma d'une table

CREATE TABLE <nom table> (

<nom champ 1> <domaine champ 1>,

<nom champ 2> <domaine champ 2>,

…

<nom dernier champ> <domaine dernier champ>

);

```
CREATE TABLE nomDeLaTable
(spécificationDeColonne,
 [,spécificationDeColonne]...
 [,spécificationDeContrainte]...)
```

Suppression d'une table

DROP TABLE <nom table>;

Syntaxe générale du CREATE TABLE

de Robert Godin

```
CREATE TABLE nomDeLaTable
(spécificationDeColonne,
 [,spécificationDeColonne]...
 [,spécificationDeContrainte]...)
```

Syntaxe de ***spécificationDeColonne***

```
nomColonne [type|domaine] [DEFAULT valeurDeDéfaut]
     [NULL | NOT NULL] [UNIQUE | PRIMARY KEY]
     [REFERENCES nomTable[listeColonnes]]
     [[CONSTRAINT nomContrainte] CHECK (conditionSQL)]
```

Syntaxe de ***spécificationDeContrainte***

```
[CONSTRAINT nomContrainte]
{PRIMARY KEY listeColonnes|
 FOREIGN KEY listeColonnes REFERENCES nomTable[listeColonnes]
  [MATCH {PARTIAL|FULL}]
  [ON DELETE {NO ACTION|CASCADE|SET NULL|SET DEFAULT}]
  [ON UPDATE {NO ACTION|CASCADE|SET NULL|SET DEFAULT}]|
 CHECK (conditionSQL)
}
[[NOT] DEFERRABLE INITIALLY {DEFERRED|IMMEDIATE}]
```

Création du schéma d'une table en SQL (simple)

de Robert Godin

Forme simple

```
CREATE TABLE Client
(noClient       INTEGER,
 nomClient      VARCHAR(15),
 noTéléphone    VARCHAR(15)
)
```

1) vérification interne

2) création de la table

- schéma stocké dans le dictionnaire de données
- allocation des structures physiques
  - clause non standardisée pour organisation primaire

Types SQL (norme SQL2) de Robert Godin

Numérique exact

- INTEGER (ou INT)
  - Entier (précision non standardisée)
  - Exemples : 2, 3, 459
- SMALLINT
  - Petit entier (précision non standardisée)
  - Exemples : 2, 3, 459
- NUMERIC(p, c) (ou DECIMAL(p, c) ou DEC(p, c))
  - Nombre décimal avec *p* chiffres significatifs (excluant le point) et *c* chiffres après le point
  - Exemples : 2.5, 456.342, 6

Types SQL2 (suite) de Robert Godin

Numérique approximatif

- REAL
  - Point flottant (précision non standardisée)
  - Exemples : 3.27E-4, 24E5
- DOUBLE PRECISION
  - Point flottant à double précision (non standardisée)
  - Exemples : 3.27265378426E-4, 24E12
- FLOAT(n)
  - Point flottant
    - précision minimale est de *n* chiffres pour la mantisse
  - Exemples : 3.27E-4, 24E5

Types SQL2 (suite) de Robert Godin

Date et temps (SQL2 intermédiaire; précision *p*: SQL2 complet)

- DATE
  - année (quatre chiffres), mois (2 chiffres) et jour (2 chiffres)
  - Exemple : DATE '1998-08-25'
- TIME[(p)]
  - heure (2 chiffres), minutes (2 chiffres), secondes (2 +*p* chiffres)
  - Exemple : TIME '14:04:32.25'
- TIMESTAMP[(p)]
  - DATE + TIME
  - Exemple : TIMESTAMP '1998-08-25 14:04:32.25'
- INTERVAL
  - Représente un intervalle de temps
  - Exemple : INTERVAL '2' DAY (intervalle de deux jours)

Expressions de DATE Oracle

de Robert Godin

```
SQL> SELECT SYSDATE FROM DUAL;

SYSDATE
--------
02-02-05
```

```
SQL> ALTER SESSION SET NLS_DATE_FORMAT = 'DD-MM-YYYY HH24:MI:SS';

Session altered.

SQL> SELECT SYSDATE FROM DUAL;

SYSDATE
--------------------
05-02-2002 09:08:26
```

```
SQL> SELECT TO_DATE('05/02/2000', 'DD/MM/YYYY') FROM DUAL;

TO_DATE('05/02/2000
--------------------
05-02-2000 00:00:00
```

```
SQL> SELECT TO_CHAR(SYSDATE,'DD/MM/YYYY') FROM DUAL;

TO_CHAR(SY
----------
22/01/2002
```

Types SQL (suite) de Robert Godin

- *Booléen* (SQL2)
  - BIT (n)
    - Vecteur de *n* bits.
    - Exemples : B'00100110', X'9F'
  - BIT VARYING (n)
    - taille variable (max = *n*)
- ***Données de grande taille*** (LOB SQL3:1999)
  - BINARY LARGE OBJECT (n) (BLOB(n))
    - *n* : taille en octets (ex: 1024, 5K, 3M, 2G)
    - Exemple : X '52CF4 ' (hexadécimal)
  - CHARACTER LARGE OBJECT (n) (CLOB(n))
  - NATIONAL CHARACTER LARGE OBJECT (n) (NCLOB(n))

Dialecte Oracle de Robert Godin

- NUMBER(*p*,[*c*])
  - numérique exact; *p* entre 1 et 38
  - *c* doit être entre -84 et +127 (défaut, c =0)
    - valeur négative signifie un arrondissement.
- VARCHAR2(*n*) : $n \leq 4000$
- RAW(*n*)
  - Binaire de taille *n* octets ($n \leq 2000$).
- LONG(*n*)
  - Chaîne de caractères de taille variable ($n \leq 2G$)
  - Maximum une colonne LONG par table
- LONG RAW(*n*)
  - Binaire de taille variable ($n \leq 2G$).
  - Maximum une colonne de type LONG RAW par table

Dialecte Oracle (suite) de Robert Godin

- ROWID : identifiant de ligne composé de
  - identificateur de fichier
  - identificateur de bloc relatif au fichier
  - identificateur de ligne relatif au bloc
- UROWID
  - identificateur universel de ligne (à partir de la version 8.1).
  - distingue index primaire (ORGANIZATION INDEX)
- Conversions implicites

Type SQL2	Type Oracle
CHARACTER (*n*), CHAR (*n*)	CHAR (*n*)
NUMERIC (*p*,*s*), DECIMAL (*p*,*s*), DEC (*p*,*s*)	NUMBER (*p*,*s*)
INTEGER, INT, SMALLINT	NUMBER (38)
FLOAT (*p*)	FLOAT (*p*)
REAL	FLOAT (63)
DOUBLE PRECISION	FLOAT (126)
VARCHAR(*n*), CHARACTER VARYING(*n*)	VARCHAR2 (*n*)

Dialecte Oracle(suite et fin)

de Robert Godin

- DATE
  - ~TIMESTAMP SQL2
- Mécanisme d ’internationalisation
  - Paramètre de configuration NLS_LANG
    - CHARACTER SET
    - DATE_FORMAT
    - ...
- ALTER SESSION
  - pour modifier

```
ALTER SESSION SET NLS_DATE_FORMAT = 'DD/MM/YYYY'
```

- LOB : taille max 4G en 9i, beaucoup + en 11g
- BFILE : fichier externe

Notion de clé

- Clé candidate : clé de taille minimale
- Clé primaire : de préférence non évolutive et sans structure interne implicite, ni signification externe
- Utilisation de la clé primaire pour référencer les enregistrements
- Clé étrangère: référence à un enregistrement d'une autre table

Contraintes d'intégrité

- Contraintes NOT NULL
  - spécifie si un champ est obligatoire ou non
  - défaut Oracle : NULL (sauf clé primaire)

```
CREATE TABLE Affectations (
    id_projet           INTEGER,
    id_employe          INTEGER,
    taux                NUMERIC(5,2) NOT NULL CHECK (taux >
    0)
);
```

Contraintes d'intégrité

- Contraintes de clé primaire
  - unique
  - renseignée (NOT NULL)

```
CREATE TABLE Affectations (
    id_projet           INTEGER,
    id_employe          INTEGER,
    taux                NUMERIC(5,2) NOT NULL CHECK (taux > 0),
    PRIMARY KEY (id_projet, id_employe)
);
```

Contraintes d'intégrité

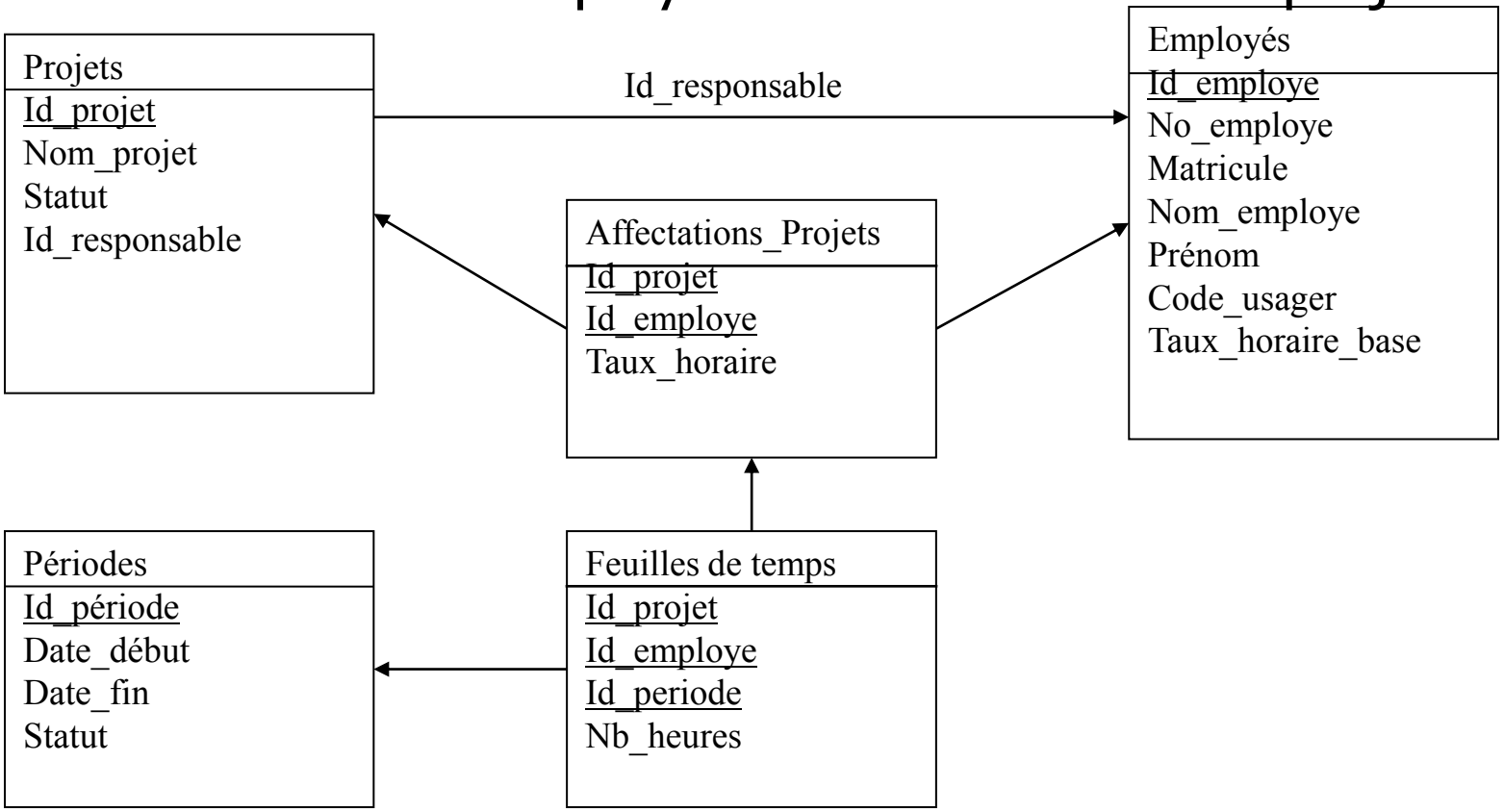
- Contraintes de clé étrangère (suite)
  - NO ACTION (restriction - défaut)
  - CASCADE (effet domino)
  - SET NULL (mise à blanc)
  - SET DEFAULT (mise à la valeur par défaut)

Contraintes d'intégrité

```
CREATE TABLE Affectations (
        id_projet           INTEGER REFERENCES Projets,
        id_employe          INTEGER REFERENCES Employes(id_employe)
                              ON DELETE CASCADE,
        taux                NUMERIC(5,2) NOT NULL CHECK (taux > 0),
        PRIMARY KEY (id_projet, id_employe));
```

Contraintes d'intégrité de Robert Godin

Peut-on détruire un employé encore affecté à un projet ?



Modification du schéma (ALTER TABLE)

de Robert Godin

Syntaxe

```
ALTER TABLE nomTable
{ADD COLUMN spécificationColonne|
DROP COLUMN nomColonne [RESTRICT|CASCADE]|
ADD spécificationContrainte|
DROP nomContrainte [RESTRICT|CASCADE]|
ALTER nomColonne SET DEFAULT valeurDéfaut|
ALTER nomColonne DROP DEFAULT}
```

```
ALTER TABLE Client
ADD COLUMN age INTEGER CHECK(age >0)
```

Exemples de VIEWS du INFORMATION_SCHEMA de Robert Godin

- SCHEMATA
  - les SCHEMA créés par CURRENT_USER
- DOMAINS
  - les DOMAIN accessibles par CURRENT_USER ou PUBLIC
- TABLES
  - les noms des tables accessibles par CURRENT_USER ou PUBLIC
- VIEWS
  - les vues accessibles par CURRENT_USER ou PUBLIC
- COLUMNS
  - les colonnes des TABLE accessibles par CURRENT_USER ou PUBLIC
- TABLE_CONSTRAINTS
  - contraintes des TABLE créées par CURRENT_USER
- CHECK_CONSTRAINTS
  - contraintes CHECK des TABLE créées par CURRENT_USER
- ASSERTIONS
  - ASSERTION créées par CURRENT_USER
- TABLE_PRIVILEGES
  - privilèges accordés par CURRENT_USER, à CURRENT_USER, ou à PUBLIC

Dictionnaire de données Oracle avec SQL*plus

de Robert Godin

```
SQL> CREATE TABLE Client
  2  (noCLIENT   INTEGER,
  3   nomClient   VARCHAR(15),
  4   noTéléphone  VARCHAR(15))
  5  /

Table créée.

SQL> SELECT Table_Name
  2  FROM   USER_TABLES
  3  /

TABLE_NAME
------------------------------
CLIENT

SQL> SELECT Column_Name, Data_Type
  2  FROM   USER_TAB_COLUMNS
  3  WHERE  Table_Name = 'CLIENT'
  4  /

COLUMN_NAME                    DATA_TYPE
------------------------------ ------------------
------------
NOCLIENT                       NUMBER
NOMCLIENT                      VARCHAR2
NOTÉLÉPHONE                    VARCHAR2
```

Recherche d'une table du dictionnaire de données

de Robert Godin

```
SQL> SELECT  Table_Name
  2  FROM  DICTIONARY
  3  WHERE  Table_Name like '%TABLE%'
  4  /

TABLE_NAME
------------------------------
ALL_ALL_TABLES
ALL_NESTED_TABLES
ALL_OBJECT_TABLES
ALL_PART_TABLES
ALL_TABLES
ALL_UPDATABLE_COLUMNS
USER_ALL_TABLES
USER_NESTED_TABLES
USER_OBJECT_TABLES
USER_PART_TABLES
USER_QUEUE_TABLES
USER_TABLES
USER_TABLESPACES
USER_UPDATABLE_COLUMNS
TABLE_PRIVILEGES

15 ligne(s) sélectionnée(s).
```

Sélection de données

- Syntaxe:

  SELECT …
  FROM …
  [WHERE …]
  [GROUP BY …]
  [HAVING …]
  [ORDER BY …]

- FROM: Table
- Résultat: Table

Requêtes SQL (SELECT) de Robert Godin

Syntaxe de *requêteSQL*

```
selectSQL |
(requêteSQL) {UNION|INTERSECT|EXCEPT} (requêteSQL)
```

Syntaxe du *selectSQL*

```
SELECT      {[ALL|DISTINCT] expression [AS nomColonne]
                        [,expression [AS nomColonne]]…}|*
FROM        table [AS nomTable [(nomColonne[,nomColonne])]]]
           [,table [AS nomTable [(nomColonne[,nomColonne])]]]…
[WHERE      conditionSQL]
[GROUP BY   nomColonne [,nomColonne]…
[HAVING     conditionSQL]
[ORDER BY   nomColonne [ASC|DESC] [,nomColonne[ASC|DESC]]…]
```

- …

Sélection de données

Id_client	Nom	Naissance	NAS
1	Sanlessou	01/01/1950	111111111
2	Chaiquenboi	01/01/1960	121
3	Senzintéré	01/01/1950	

- **Projection**

```
SELECT nom
FROM Clients;
```

Nom
Sanlessou
Chaiquenboi
Senzintéré

```
SELECT *
FROM Clients ;
```

Id_client	Nom	Naissance	NAS
1	Sanlessou	01/01/1950	111111111
2	Chaiquenboi	01/01/1960	121
3	Senzintéré	01/01/1950	

Sélection de données condition égal

Id_client	Nom	Naissance	NAS
1	Sanlessou	01/01/1950	111111111
2	Chaiquenboi	01/01/1960	121
3	Senzintéré	01/01/1950	

- **Sélection**

```
SELECT *
FROM Clients
WHERE naissance = '01/01/1950';
```

Id_client	Nom	Naissance	NAS
1	Sanlessou	01/01/1950	111111111
3	Senzintéré	01/01/1950	

Autres opérateurs de comparaison : >, <, >=, <=, <>, BETWEEN, NOT BETWEEN, IN, NOT IN, LIKE, IS NULL, IS NOT NULL

Combinaisons de comparaisons par AND, OR, NOT

ConditionSQL – BETWEEN de Robert Godin

Sélectionner les *Commandes* du mois de juin de l'année 2000

```
SELECT    *
FROM      Commande
WHERE     dateCommande BETWEEN '01/06/2000' AND '30/06/2000'
```

```
SELECT    *
FROM      Commande
WHERE     dateCommande >= '01/06/2000' AND
          dateCommande <='30/06/2000'
```

ConditionSQL – IN de Robert Godin

Sélectionner les *Commandes* du *Client* dont le *noClient* est 10 ou 40 ou 80

```
SELECT      *
FROM        Commande
WHERE       noClient IN (10, 40, 80)
```

```
SELECT      *
FROM        Commande
WHERE       noClient = 10 OR noClient = 40 OR noClient = 80
```

ConditionSQL – LIKE de Robert Godin

Sélectionner les *Clients* dont le *nomClient* contient le mot *Le*

```
SELECT *
FROM      Client
WHERE     nomClient LIKE '%Le%'
```

2ième lettre du *nomClient* = *o* et dernière lettre est un *k*

```
SELECT *
FROM      Client
WHERE     nomClient LIKE '_o%k'
```

ConditionSQL - IS NOT NULL

de Robert Godin

Sélectionner les *Articles* dont la description n'est pas une valeur nulle

```
SELECT      *
FROM        Article
WHERE       description IS NOT NULL
```

Expression sur colonne du WHERE

de Robert Godin

Les *Articles* dont le *prixUnitaire* incluant la taxe de 15% est inférieur à $16.00

```
SELECT noArticle, prixUnitaire, prixUnitaire*1.15 AS prixPlusTaxe
FROM        Article
WHERE       prixUnitaire*1.15 < 16
```

noArticle	prixUnitaire	prixPlusTaxe
10	10.99	12.64
20	12.99	14.94
70	10.99	12.64

Sélection de données

Id_client	Nom	Naissance	NAS
1	Sanlessou	01/01/1950	111111111
2	Chaiquenboi	01/01/1960	121
3	Senzintéré	01/01/1950	

Id_compte	No_compte	Type	Id_client	Solde
10	A90SJ	Chèques	1	2300.00
21	XC1234	Épargne	2	210.00
30	X1390	Épargne	1	2100.00

- **Produit Cartésien**

```
SELECT *
FROM Client, Compte;
```

Id_client	Nom	Naissance	NAS	Id_compte	No_compte	Type	Id_client	Solde
1	Sanlessou	01/01/1950	111111111	10	A90SJ	Chèques	1	2300.00
1	Sanlessou	01/01/1950	111111111	21	XC1234	Épargne	2	210.00
1	Sanlessou	01/01/1950	111111111	30	X1390	Épargne	1	2100.00
2	Chaiquenboi	01/01/1960	121	10	A90SJ	Chèques	1	2300.00
...	...	...	...	...	...	...	...	...

Sélection de données

Id_client	Nom	Naissance	NAS
1	Sanlessou	01/01/1950	111111111
2	Chaiquenboi	01/01/1960	121
3	Senzintéré	01/01/1950	

Id_compte	No_compte	Type	Id_client	Solde
10	A90SJ	Chèques	1	2300.00
21	XC1234	Épargne	2	210.00
30	X1390	Épargne	1	2100.00

- **Jointure** : combinaison sélection/produit cartésien

```
SELECT *                                    (syntaxe traditionnelle)
FROM Client, Compte
WHERE Client.id_client = Compte.id_client;
```

Id_client	Nom	Naissance	NAS	Id_compte	No_compte	Type	Id_client	Solde
1	Sanlessou	01/01/1950	111111111	10	A90SJ	Chèques	1	2300.00
2	Chaiquenboi	01/01/1960	121	21	XC1234	Épargne	2	210.00
1	Sanlessou	01/01/1950	111111111	30	X1390	Épargne	1	2100.00

Sélection de données

- **Jointure** : syntaxe SQL99

```
SELECT *
FROM  Client NATURAL JOIN Compte ;

SELECT *
FROM Client JOIN Compte ON Client.id_client = Compte.id_client;

[LEFT | RIGHT] OUTER JOIN
```

Sélection de données

Id_client	Nom	Naissance	NAS
1	Sanlessou	01/01/1950	111111111
2	Chaiquenboi	01/01/1960	121
3	Senzintéré	01/01/1950	

Id_compte	No_compte	Type	Id_client	Solde
10	A90SJ	Chèques	1	2300.00
21	XC1234	Épargne	2	210.00
30	X1390	Épargne	1	2100.00

- **TRI** des résultats

```
SELECT Nom, No_compte
FROM Client, Compte
WHERE Client.id_client = Compte.id_client
ORDER BY Nom, No_compte;
```

Nom	No_compte
Chaiquenboi	XC1234
Sanlessou	A90SJ
Sanlessou	X1390

Sélection de données

Id_client	Nom	Naissance	NAS
1	Sanlessou	01/01/1950	111111111
2	Chaiquenboi	01/01/1960	121
3	Senzintéré	01/01/1950	

Id_compte	No_compte	Type	Id_client	Solde
10	A90SJ	Chèques	1	2300.00
21	XC1234	Épargne	2	210.00
30	X1390	Épargne	1	2100.00

- **Utilisation de fonctions**

```
SELECT Nom, year(naissance) AS an
FROM Client
ORDER BY Nom;
```

Nom	An
Chaiquenboi	1960
Sanlessou	1950
Senzintéré	1950

- **Utilisation de fonctions d'agrégation** :

  MAX, MIN, COUNT, SUM, AVERAGE

```
SELECT COUNT(*) AS nb_clients,
            MIN(naissance) AS nais_min
FROM Client;
```

Nb_clients	Nais_min
3	01/01/1950

Sélection de données

Id_client	Nom	Naissance	NAS
1	Sanlessou	01/01/1950	111111111
2	Chaiquenboi	01/01/1960	121
3	Senzintéré	01/01/1950	

Id_compte	No_compte	Type	Id_client	Solde
10	A90SJ	Chèques	1	2300.00
21	XC1234	Épargne	2	210.00
30	X1390	Épargne	1	2100.00

- **Définition de groupes**

```
SELECT id_client, count(*) AS nb_comptes
FROM Compte
GROUP BY id_client;
```

Id_client	Nb_comptes
1	2
2	1

- **Sélection de groupes**

```
SELECT id_client, count(*) AS nb_comptes
FROM Compte
GROUP BY id_client
HAVING count(*) > 1;
```

Id_client	Nb_comptes
1	2

Sélection de données
Sous-requête (dans le WHERE)

Id_client	Nom	Naissance	NAS
1	Sanlessou	01/01/1950	111111111
2	Chaiquenboi	01/01/1960	121
3	Senzintéré	01/01/1950	

Id_compte	No_compte	Type	Id_client	Solde
10	A90SJ	Chèques	1	2300.00
21	XC1234	Épargne	2	210.00
30	X1390	Épargne	1	2100.00

```
SELECT id_client
FROM Client
WHERE id_client IN (SELECT id_client FROM Compte);
```

Id_client
1
2

```
SELECT id_client
FROM Client
WHERE EXISTS (SELECT *
              FROM Compte
              WHERE Compte.id_client = Client.id_client);
```

Id_client
1
2

Sélection de données
Sous-requête (dans le FROM)

Id_client	Nom	Naissance	NAS
1	Sanlessou	01/01/1950	111111111
2	Chaiquenboi	01/01/1960	121
3	Pasforfor	01/01/1950	212111323

Id_employe	Nom_emp	NAS
10	Travayen	121333313
21	Pasforfor	212111323

Nombre d'individus qui sont clients et/ou employés

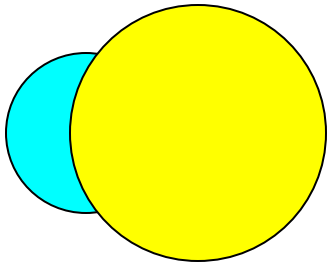
```
SELECT COUNT(*)
FROM   (( SELECT NAS
            FROM Client)
          UNION
           (SELECT NAS
            FROM Employe)) Individu;
```

4

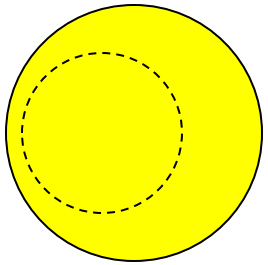
Sélection de données
Division ensembliste

- $T_1 \subseteq T_2$?

```
NOT EXISTS (T1 EXCEPT T2)
```



$T_1 \not\subset T_2$



$T_1 \subset T_2$

n $\rightarrow T_2 \div T_1$

Sélection de données
Division ensembliste de Robert Godin

Quelles commandes incluent
au moins les articles 10 et 70 ?

noArticle
10
70

$\subseteq$

noCommande	noArticle
1	10
1	70
1	90
2	40
2	95
3	20
4	40
4	50
5	70
5	10
5	20
6	10
6	40
7	50
7	95

```
SELECT noCommande
FROM   Commande
WHERE NOT EXISTS
    ((SELECT noArticle FROM Article
      WHERE noArticle IN (10, 70))
     EXCEPT
     (SELECT noArticle FROM LigneCommande
      WHERE  noCommande =
                  Commande.noCommande));
```

noCommande
1
5

Modifications de données

- Insertion de rangées

```
INSERT INTO Clients
VALUES (5, "Bibi", "1934/10/10", 123111434);

INSERT INTO Clients (id_client, nom)
VALUES (50, "Bibi2");

INSERT INTO Clients2(id_client, nom, NAS)
   SELECT id_client, nom, NAS
   FROM Clients;
```

Modifications de données

- Mise à jour de rangées

```
UPDATE Comptes
SET solde = solde + 100
WHERE solde > 1000;

UPDATE Comptes
SET solde = solde + 30, No_compte = 'XYZ99'
WHERE id_client IN (    SELECT id_client
                        FROM Clients
                        WHERE nom like 'A%');
```

Modifications de données

- Suppression de rangées

```
DELETE FROM Comptes
WHERE solde = 0;

DELETE FROM Clients
WHERE id_client NOT IN (SELECT id_client
                        FROM Comptes);

DELETE FROM Clients;
```

- Différence entre DELETE FROM Clients et DROP Clients ?

Types de données MM en relationnel

- BFILE : pointeur vers un fichier externe
  - taille maximale limitée par le o/s
  - fichier géré par le système d'exploitation
  - équivaut à un LOB externe
  - accessible en lecture seulement
- (B/C/NC)LOB : large objects
  - 3 versions : CLOB (caractères d'1 octet), NCLOB (Unicodes), BLOB (binaire)
  - taille maximale 4GB (9i) **et 8 TB-128 TB(10g)**
  - fait partie de la table logiquement
  - pointeur vers l'objet stocké dans la table
  - objet lui-même stocké à part des autres champs physiquement
  - LOB interne à la BD
  - peut être mis à jour par des INSERT ou des UPDATE

Table 2.1

BLOB	Binary	8 TB – 128 TB	Random access Transactions Needs locator
CLOB	Character	8 TB – 128 TB	Random access Transactions Needs locator
NCLOB	National	8 TB – 128 TB	Random access Transactions Needs locator
BLOB	Character sets	8 TB – 128 TB	Random access Transactions Needs locator
BFILE	Binary	O/s limited	Read-only External file

BLOB ou BFILE ?

- BFILEs n'est pas sous le contrôle du SGBD. Les utilisateurs peuvent détruire les fichiers ou changer la localisation sur le disque;
- Il n'est pas possible d'utiliser le BFILE dans des requêtes SQL3 et des transactions;
- Un BFILE sert pour enregistrer temporairement un fichier.

Types de données MM en relationnel

- Utilisation de BFILE
  - Au niveau du Système d'exploitation:
    - création d'un répertoire contenant les fichiers;
    - création des fichiers;
    - assignation des droits d'accès pour que les processus d'Oracle puisse lire les fichiers.
  - Au niveau d'Oracle:
    - déclaration du répertoire contenant les fichiers (CREATE OR REPLACE DIRECTORY) (par un utilisateur autorisé)
    - assignation d'un droit d'accès en lecture à ce répertoire aux usagers autorisés (par celui qui a déclaré le répertoire)
    - création de la table contenant la colonne BFILE
    - création de la référence au fichier via la fonction BFILENAME

Types de données MM en relationnel

- Utilisation de BFILE : exemple
  - Au niveau d'Oracle
    - utilisateur dba/system ou autre utilisateur autorisé

```
CREATE OR REPLACE DIRECTORY photo_dir AS
  'C:\PICTURES'
GRANT READ ON DIRECTORY photo_dir TO scott
```

    - propriétaire de la table

```
CREATE TABLE grape
(grape_name      VARCHAR(2) PRIMARY KEY,
picture          BFILE)

INSERT INTO grape
VALUES ('chardonnay',
BFILENAME('photo_dir','chardonnay.jpg'))
```

Types de données MM en relationnel

- Utilisation de CLOB/BLOB : Création de la table

```
CREATE TABLE wine_list
(  wine_code          CHAR(6),
   wine_name          VARCHAR2(30) NOT NULL,
   region             VARCHAR2(20) NOT NULL,
   year               NUMBER(4),
   category           VARCHAR2(20),
   grape              VARCHAR2(20),
   price              NUMBER(5,2),
   bottle_size        NUMBER(4),
   character          VARCHAR2(50),
   note               CLOB DEFAULT EMPTY_CLOB(),
   pronunciation      BLOB DEFAULT EMPTY_BLOB(),
   picture            BFILE,
CONSTRAINT prim_wine PRIMARY KEY (wine_code))
```

Types de données MM en relationnel

- Insertion de données dans un LOB en 2 étapes
  - Insertion des données classiques

```
INSERT INTO wine_list (wine_code, wine_name,
  region, year) VALUES ('CHPA00', 'Chateau
  Parizeau', 'Languedoc', 2000)
```

  - Mise à jour des CLOB

```
UPDATE wine_list
SET note = 'Excellent vin avec les grillades de
  bœuf ou d''agneau'
WHERE wine_code = 'CHPA00'
```

    - Cette seconde étape doit être mise en œuvre différemment pour les BLOB

Types de données MM en relationnel

- Insertion BLOB (méthodes de InterMedia)

```
BEGIN
   INSERT INTO wine_list (wine_code, wine_name,....)
   VALUES (....., ORDSYS.ORDAudio.init('FILE',
        'FILE_DIR','speaker.au'), ...);
 COMMIT;
 END; /
```

- Pour des images = ORDSYS.ORDImage.init
- Pour des vidéos = ORDSYS.ORDVideo.init

Types de données MM en relationnel

- Suppression des données d'un CLOB/d'un BLOB

```
UPDATE wine_liste
SET note = EMPTY_CLOB()
WHERE wine_code = 'CHPA00';
```

Types de données MM en relationnel

- Autres manipulations de CLOB
  - via la plupart des opérateurs et fonctions définies sur des CHAR(n) ou VARCHAR(n)
  - Opérateur de concaténation : ||

```
UPDATE wine_liste
SET note = note || '(Toto Tremblay,
  23/02/2002)'
WHERE wine_code = 'CHPA00'
```

  - Fonctions concat, lower, upper, replace, substr, length …

Types de données MM en relationnel

- Manipulations limitées de CLOB en SQL
- Aucune manipulation de BLOB en SQL
  - Il faut pour cela utiliser une extension procédurale à SQL : PL/SQL, JDBC, InterMedia, dbms_lob…

Types SQL

User-Defined Types :

- Attributes
- Methods

Oracle

- Objects
- Arrays
- Nested tables
- Scalar types

Built-in data types :

- Character e.g. CHAR
- Numeric
- Date
- BLOB
- Type operations
- **scalar**
- atomic or encapsulated

Travaux Personnels et labo

- Regardez les exercices du Chapitre 4
- Révisez la syntaxe du langage SQL
- Débutez la création de votre BD pour le laboratoire 1

Exemples

LitSearch
http://infolab.stanford.edu/~ullman/fcdb/ito/index.html

Database Project: Episode Guide
http://infolab.stanford.edu/~ullman/fcdb/silverberg/home.html

Oracle en bref…
(Database systems: the complete book)

http://infolab.stanford.edu/~ullman/fcdb/oracle.html

Type d'Oracle 11g (en bref)

http://ss64.com/ora/syntax-datatypes.html

Sommaire

- On a introduit le modèle relationnel
- On a fait le tour des fonctionnalités de base de SQL (dans sa variante Oracle)
  - création de tables
  - mise à jour de données dans les tables
  - sélection de données à partir des tables
- On a introduits les types multimédias