LE LANGAGE DE REQUETES SQL

- Origines et Evolutions
- SQL1 86: la base
- SQL1 89: l'intégrité

1. Origines et Evolutions

- SQL est dérivé de l'algèbre relationnelle et de SEQUEL
- Il a été intégré à SQL/DS, DB2, puis ORACLE, INGRES, …
- Il existe trois versions normalisées, du simple au complexe :
  - SQL1 86 version minimale
  - SQL1 89 addendum (intégrité)
  - SQL2 (92) langage complet à 3 niveaux
- Une version 3 étendue (objets, règles) est la norme 99.
- La plupart des systèmes supportent SQL2 complet

Opérations

- Opérations de base
  - **SELECT**, **INSERT**, **UPDATE**, **DELETE**
- Opérations additionnelles
  - définition et modification de schémas
  - définition de contraintes d'intégrité
  - définition de vues
  - accord des autorisations
  - gestion de transactions

Organisation du Langage

- SQL comprend quatre parties :
- Le langage de définition de schéma (Tables, Vues, Droits)
- Le langage de manipulation (Sélection et mises à jour)
- La spécification de modules appelables (Procédures)
- L'intégration aux langages de programmation (Curseurs)

SQL1 - 86

- LANGAGE DE DEFINITIONS DE DONNEES
  - **CREATE TABLE**
  - **CREATE VIEW**
- LANGAGE DE MANIPULATION DE DONNEES
  - **SELECT** **OPEN**
  - **INSERT** **FETCH**
  - **UPDATE** **CLOSE**
  - **DELETE**
- LANGAGE DE CONTROLE DE DONNEES
  - **GRANT** et **REVOKE**
  - **BEGIN** et **END TRANSACTION**
  - **COMMIT** et **ROLLBACK**

Base de Données

- Collection de tables et de vues dans un schéma

VITICULTEURS *(NVT, NOM, PRENOM, VILLE, REGION)*
VINS *(NV, CRU, MILLESIME, DEGRE, NVT, PRIX)*
BUVEURS *(NB, NOM, PRENOM, VILLE)*
ABUS *(NV, NB,DATE,QTE)*

GROS_BUVEURS *(NB, NOM, PRENOM)*

2. SELECT: Forme Générale

- **SELECT** <liste de projection>
- **FROM** <liste de tables>
- [**WHERE** <critère de jointure> **AND** <critère de restriction>]
- [**GROUP** BY <attributs de partitionnement>]
- [**HAVING** <citère de restriction>]

- **Restriction** :
  - arithmétique (=, <, >, ≠, ≥,≤)
  - textuelle (**LIKE**)
  - sur intervalle (**BETWEEN**)
  - sur liste (**IN**)
- **Possibilité de blocs imbriqués par :**
  - **IN, EXISTS, NOT EXISTS, ALL, SOME, ANY**

Exemples de Questions (1)

- Q1: Crus des vins sans doubles.
  - **SELECT DISTINCT** CRU
  - **FROM** VINS
- Q2: Noms des buveurs ayant bus des Beaujolais 97 ou 98
  - **SELECT DISTINCT** NOM
  - **FROM** BUVEURS B, VINS V, ABUS
  - **WHERE** B.NB = ABUS.NB
  - **AND** ABUS.NV = V.NV
  - **AND** CRU LIKE '%BEAUJOLAIS%'
  - **AND** MILLESIME IN (1997, 1998)

Exemples de Questions (2)

- Q3 : Noms et prénoms des buveurs de vins dont le cru commence par B, de degré inconnu ou compris entre 11 et 13.
  - **SELECT** NOM, PRENOM
  - **FROM** BUVEURS B, VINS V, ABUS A
  - **WHERE** B.NB = A.NB AND A.NV = V.NV
  - **AND** CRU LIKE "B%"
  - **AND** (DEGRE BETWEEN 11 AND 13 OR DEGRE IS NULL)
- Q4 : Noms des crus bus par au moins un buveurs.
  - **SELECT DISTINCT** CRU
  - **FROM** VINS V
  - **WHERE EXISTS** (SELECT *
  - FROM BUVEURS B, ABUS A
  - WHERE B.NB = A.NB AND A.NV = V.NV)

Exemples de Questions (3)

- Q5: Calculer le degré moyen pour chaque cru.
  - SELECT CRU, AVG(DEGRE)
  - FROM VINS
  - GROUP BY CRU
- Q6 : Calculer le degré moyen et le degré minimum pour tous les crus de 94 dont le degré minimum est supérieur à 12.
  - SELECT CRU, AVG(DEGRE), MIN(DEGRE)
  - FROM VINS
  - WHERE MILLESIME = 1994
  - GROUP BY CRU
  - HAVING MIN(DEGRE) > 12

Forme générale de la condition

<**search condition**> ::= **[**NOT**]**
<**nom_colonne**> θ **constante** | <**nom_colonne**>
<**nom_colonne**> LIKE **<modèle_de_chaîne>**
<**nom_colonne**> IN <**liste_de_valeurs**>
<**nom_colonne**> θ **(**ALL | ANY | SOME**)**
<**liste_de_valeurs**>
EXISTS <**liste_de_valeurs**>
UNIQUE <**liste_de_valeurs**>
<**tuple**> MATCH **[**UNIQUE**]** <**liste_de_tuples**>
<**nom_colonne**> BETWEEN **constante** AND **constante**
<**search condition**> AND | OR <**search condition**>

avec
θ:= < | = | > | ≥ |≤ | <>

Remarque: <**liste_de_valeurs**> **peut être dynamiquement déterminée par une requête**

Requêtes imbriquées (1)

- Q7: Donner les crus des vins qui n'ont jamais été commandés

```
SELECT CRU
FROM VINS V
WHERE V.V# NOT IN (
    SELECT C.V#
    FROM COMMANDES C
)
```

```
SELECT CRU
FROM VINS V
WHERE V.V# <> ALL (
    SELECT C.V#
    FROM COMMANDES C )
```

Requêtes imbriquées (2)

- Q8 : Donner le nom des buveurs qui n'ont pas bu tous les vins == A VERIFIER

```
SELECT NOM
FROM BUVEURS B
WHERE EXISTS (
        SELECT *
        FROM  VINS V
        WHERE NOT EXISTS (
                SELECT *
                FROM COMMANDES C
                WHERE V.V# = C.V#
                AND C.B# = B.B#) )
```

Requêtes imbriquées (3)

- Q9: Donner le numéro et le cru des vins commandés exactement une fois

```
SELECT V#, CRU
FROM VINS
WHERE V#  MATCH UNIQUE  (
        SELECT V#
        FROM COMMANDES )
```

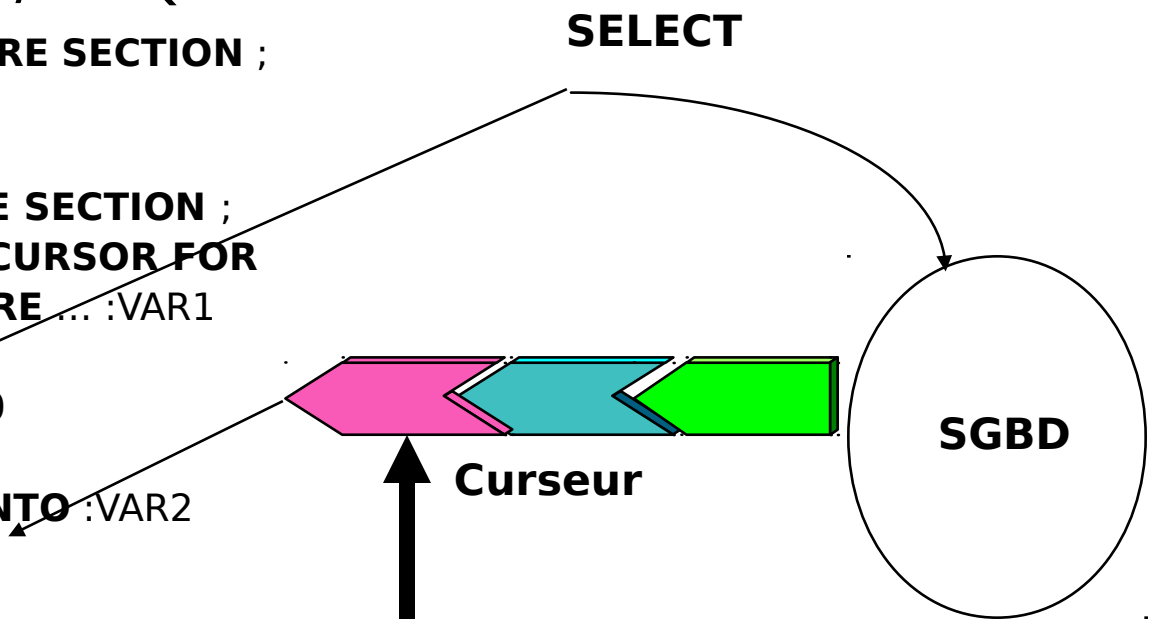
Requête Union

- Q10 :Donner le numéro et le cru des vins commandés plus de 100 fois ou bien jamais commandés

```
( SELECT V.V#, V.CRU
FROM VINS V, COMMANDES C
WHERE V.V# = C.V#
GROUP BY V.V#
HAVING COUNT(C.C#) > 100 )
UNION
( SELECT V#, CRU
FROM VINS
WHERE V# NOT IN (SELECT V# FROM COMMANDES) )
```

Utilisation de SQL depuis un langage de prog.

- Intégration de deux systèmes de types
  - utilisation d'un pré-compilateur et d'une librairie
- Passage de l'ensembliste au tuple à tuple
  - utilisation de curseurs et Fetch
- Exemple Program PL/1-SQL
  - **EXEC SQL BEGIN DECLARE SECTION** ;
  - **DCL** VAR1 CHAR(20) ;
  - **DCL** VAR2 INT ;
  - **EXEC SQL END DECLARE SECTION** ;
  - **EXEC SQL DECLARE** C1 **CURSOR FOR**
  - **SELECT** ...**FROM** ... **WHERE** ... :VAR1
  - **EXEC SQL OPEN** C1 ;
  - **DO WHILE SQLCODE** = 0
  - **BEGIN**
  - **EXEC SQL FETCH** C1 **INTO** :VAR2



3. Les Mises à Jour

- INSERT
  - Insertion de lignes dans une table
  - Via formulaire où via requêtes
- UPDATE
  - Modification de lignes dans une table
- DELETE
  - Modification de lignes dans une table

Commande INSERT

- **INSERT INTO** <relation name>
  [(attribute [,attribute] ...)]
  {**VALUES** <value spec.> [, <value spec.>] ...| <query spec.>}
- Exemples
  - **INSERT INTO** VINS (NV, CRU, MILLESIME)
  - **VALUES** 112, "JULIENAS", NULL

  - **INSERT** INTO BUVEURS (NB,NOM,PRENOM)
  - **SELECT** NVT, NOM, PRENOM
  - **FROM** VITICULTEURS
  - **WHERE** VILLE **LIKE** '%DIJON%'

Commande UPDATE

UPDATE <relation name>
SET <attribute = {value expression | **NULL**}
[<attribute> = {value expression | **NULL**}] ...
[**WHERE** <search condition>]

- **EXEMPLE**
  - **UPDATE** ABUS
  - **SET** QTE = QTE * 1.1
  - **WHERE** ABUS.NV IN
  - **SELECT** NV
  - **FROM** VINS
  - **WHERE** CRU = 'VOLNAY' **AND** MILLESIME = 1990

Commande DELETE

- **DELETE FROM** <relation name>
- [**WHERE** <search condition>]

- EXEMPLE
  - **DELETE FROM** ABUS
  - **WHERE** NV IN
  - **SELECT** NV
  - **FROM** VINS
  - **WHERE** DEGRE **IS NULL**

4. Contraintes d'intégrité

- Contraintes de domaine
  - Valeurs possibles pour une colonne
- Contraintes de clés primaires
  - Clé et unicité
- Contraintes référentielles(clé étrangères)
  - Définition des liens inter-tables

SQL1 - 89 : INTEGRITE

- **VALEURS PAR DEFAUT**
  - **CREATE TABLE** VINS
  - (NV INT **UNIQUE**,
  - CRU CHAR(10),
  - ANNEE INT,
  - DEGRE FIXED (5,2) ,
  - NVT INT,
  - PRIX FIXED(7,2) **DEFAULT** 40)
- **CONTRAINTES DE DOMAINES**
  - SALAIRE INT **CHECK BETWEEN** 6000 **AND** 100000

SQL1 - 89 : Contrainte référentielle

- Clé primaire et contrainte référentielle
  - **CREATE TABLE** VINS
  - (NV INT **PRIMARY KEY,**
  - CRU CHAR(10),
  - ANNEE INT,
  - DEGRE FIXED (5,2) ,
  - NVT INT **REFERENCES** VITICULTEURS,
  - PRIX **DEFAULT** 40)
- Référence en principe la clé primaire
  - celle de VITICULTEURS

SQL1 – 89 : Création de table

CREATE TABLE <nom_table>
(<def_colonne> *
[<def_contrainte_table>*]) ;

< def_colonne > ::=
<nom_colonne> < type | nom_domaine >
[**CONSTRAINT** nom_contrainte
< **NOT NULL** | **UNIQUE** | **PRIMARY KEY** |
CHECK (condition) | **REFERENCES** nom_table (liste_colonnes) >]

< def_contrainte_table > ::= **CONSTRAINT** nom_contrainte
< **UNIQUE** (liste_colonnes) | **PRIMARY KEY** (liste_colonnes) |
CHECK (condition) |
FOREIGN KEY (liste_colonnes) **REFERENCES** nom_table (liste_colonnes) >
[**NOT**] **DEFERRABLE**

Autre création de tables

```
CREATE TABLE EXPEDITIONS
  ( numExp INTEGER PRIMARY KEY
    date_exp DATE,
    qte QUANTITE,
    CONSTRAINT refCom FOREIGN KEY numExp
       REFERENCES  COMMANDES (numCom)
    DEFERRABLE
) ;
```

L'association d'un nom à une contrainte est optionnelle.

Ce nom peut être utilisé pour référencer la contrainte (ex: messages d'erreurs).

5. CONCLUSION

- SQL1 est un standard minimum
- Les versions étendues:
  - SQL2 = Complétude relationnelle
  - SQL3 = Support de l'objet
- Sont aujourd'hui intégrées dans les grands SGBD

LA NORMALISATION DE SQL

- Groupe de travail ANSI/X3/H2 et ISO/IEC JTC1/SC2
- Documents ISO :
  - SQL1 - 86 : Database Language SQL X3.135 ISO-9075-1987)
  - SQL1 - 89 : Database Language SQL with Integrity Enhancement X3.168 ISO-9075-1989
  - SQL2 - 92 : Database Language SQL2 X3.135 ISO-9075-1992
- Arguments pour :
  - Réduction des coûts d'apprentissage
  - Portabilité des applications
  - Longévité des applications
  - Langage de communication inter-systèmes
- Arguments contre :
  - Manque de rigueur théorique
  - Affaiblit la créativité

POSITION DES VENDEURS

- Problèmes :
  - SQLCODE (0 ou <0 si erreur)
  - Requêtes imbriquées
  - Dynamique SQL (Prepare, Execute)
  - Méta-base normalisée
  - Modèles internes (Index, Espaces,...)