Introduction au langage SQL3 (SQL 99)

Bernard ESPINASSE
Professeur à Aix-Marseille Université (AMU)
Ecole Polytechnique Universitaire de Marseille

Avril 2013

- **La norme SQL3**
- **Les triggers : normalisation**
- **Procédures stockées (PSM)**
- **L'objet dans SQL3**
- **Constructeurs de types et fonctions de SQL3**
- **Conclusion**

Plan

Sources du cours

Livres :

- Gardarin G., « Bases de données objet et relationnel », Ed. Eyrolles, 1999 (ISBN : 2-212-09060-9).
- …

Cours de :

- Georges Gardarin
- Didier Donsez
- Christophides Vassilis
- …

1. Introduction

- **La norme SQL3 et ses composants**
- **Vue d'ensemble de SQL3**
- **Les triggers : normalisation**
- **Spécifications de procédures stockées (PSM)**

SQL : quelques dates...

- **1970: invention du modèle relationnel (CODD)**
- **1981: début des projets Système R et Ingres**
- **1986: normalisation de SQL1**
- **1990: extentions aux objets et règles**
- **1990: normes SQL2**
- **1999: normes SQL3 (SQL 99)**

Normalisation de SQL3

Un groupe international :

- ISO/IEC JTC1/SC 21/WG3 DBL

Pays impliqués :

- Australia, Brazil, Canada, France, Germany, Japan
- Korea, The Netherlands, United Kingdom, United States

ANSI X3H2 (http://www.ansi.org)

Documents :

- ISO/IEC 9075:1992, "Database Languages - SQL"
- ANSI X3.135-1992, "Database Language SQL"

Validation par le NIST (http://ncsl.nist.gov):

- En cours

Les composants de SQL3

- Part 1: Framework : description de la structure du document
- Part 2: Foundation : noyau de spécification, incluant les types de données abstraits (ADTs) (SQL Foundation 846p)
- Part 3: SQL/CLI : l'interface d'appel client (Call Level Interface-SQL/CLI 181p)
- Part 4: SQL/PSM : le langage de spécifications de procédures stockées (Persistent SQL Module - SQL/PSM 143p)
- Part 5: SQL/Bindings : les liens SQL dynamique et "embedded" SQL repris de SQL-92 (Host Language Bindings -SQL Bindings 209p)
- Part 6: SQL/XA : Une spécification de l'interface XA pour moniteur transactionnel (Transaction monitor XA Interface – SQL/XA 51p)
- Part 7: SQL/Temporal : Le support du temps dans SQL3 (Temporal Extension – SQL/Temporal)
- Autres spécifications : ADTs multimédia (SQL/MM), RDA protocol (SQL/RDA)

Vue d'ensemble de SQL3

- **SQL3 présente de multiples facettes en étant un langage :**
  - de définition de types
  - de programmation
  - de requêtes
  - temporel
  - ...
- **SQL3 est adapté à la gestion de données complexes dans des SGBD Objet-Relationnel :**
  - **Nouveaux :**
    - Illustra, UniSQL, ODB II, Versant, …
  - **Relationnels étendus ("universels") :**
    - Ingres, Oracle, DB2 UDB, Informix , …

SQL3 : le modèle objet

Types de données utilisateurs :

- Types abstraits de données (ADTs) avec ou sans OID (encapsulation structure de données + comportement)
- Type tuples nommés avec implicite OID (pas d'encapsulation)
- Types distincts (types scalaires)

Support d'objets complexes : **constructeurs de type**

- pour des types « collection » (set, lists et multisets)
- pour des types « tuples » (Row) et types « références » (REF)

Héritage :

- Définition de sous-types et sous-tables
- Héritage multiple supporté

Fonction et procédures définies par l'utilisateur :

- Interne (dans SQL3) et externe (dans un langage de programmation)

Support de large objets :

- BLOBs et CLOBs)

La base de SQL3

Capacités de base (SQL/CLI) :

- Capacités de base SQL/PSM
- Triggers
- Types de données abstraits (ADT)
- Capacité orientées objet

Capacité prérequises aux objets (SQL/PSM) :

- De définition d'opérations complexes
- De stockage de procédures dans la base
- D'appels de procédures externes

Les triggers : normalisation

Création des triggers

- événement = INSERT, UPDATE, DELETE
- possibilité de déclencher **avant** ou **après l'événement**
- action = opération sur table avec éventuelle **condition**
- possibilité de référencer les valeurs avant ou après mise à jour

Exemple :

```
EMPLOYE (ID int, ancienneté int, salaire float)
CUMUL (ID int, Augmentation float)
CREATE TRIGGER AFTER UPDATE OF salaire ON employé  /* événement */
REFERENCING OLD AS ancien_salaire, NEW AS nouveau_salaire
WHEN (ancienneté > 5)                                /* condition */
    UPDATE CUMUL                                     /* action */
    SET Augmentation =
    Augmentation + nouveau_salaire - ancien_salaire
      WHERE ID = employé.ID
```


Spécifications de procédures stockées (PSM)

Langage de programmation de procédures :

- déclaration de variables
- assignation
- CALL and RETURN statements for SQL procedures
- conditionnels CASE, IF
- LOOP, WHILE and REPEAT statements for repeated execution of a block of SQL statements
- exceptions SIGNAL, RESIGNAL
- possibilité de procédures et fonctions externes
- **...**

2. Les objets dans SQL3

- **Les types abstraits (ADTs)**
- **Types tuple (ROW type)**
- **Les tables et l'héritage dans SQL3**

Les objets dans SQL3

Extensibilité des types de données : les ADTs

- Définition de **types abstraits** (ADT)
- ADT avec ou sans référence (**avec ou sans OID**)

Support d'objets complexes :

- **Constructeurs de types** (tuples, set, list, …)
- Utilisation de **référence** (OID)

Héritage :

- Définition de sous-types
- Définition de sous-tables

Les types abstraits (ADTs)

Définition : types créés par l'utilisateur

Syntaxe :

CREATE TYPE <nom ADT> <corps de l'ADT>

• <corps de l'ADT>

- <OID option> ::= WITH OID [NOT] VISIBLE
  - objets sans OID par défaut
  - les OID peuvent être vus par les requêtes, contraintes et autres ADTs
- <subtype clause> ::= UNDER <supertype clause>
  - possibilité d'héritage multiple avec résolution de conflits explicite
- <member list>
  - <attribute definition> : attributs publics ou privés
  - <function declaration> : opérations publiques
  - <operator name list> : opérateurs surchargés
  - <ordering definition> : ordre (Equals, less, than, relative, hash)
  - <cast clause> : fonction de conversion de types
  - <procedure clause> : procedure avec corps défini dans SQL3 ou externe

Exemples d'ADT

Un type avec référence (avec OID) :

CREATE TYPE WITH OID VISIBLE **Address** (
PUBLIC num INT, street CHAR(20), city CHAR(15), country CHAR(10),
EQUALS DEFAULT, LESS THAN NONE,
PUBLIC FUNCTION **distance**(a Address,b Address) RETURNS FLOAT
PUBLIC FUNCTION **fullAddr**(a Address) RETURNS CHAR(45)

Un type sans référence (sans OID) :

CREATE TYPE **Person** (
PUBLIC name CHAR(50), address: **Address**, Nationality: VARCHAR,
PRIVATE birth-date DATE,
PUBLIC FUNCTION age (DATE, DATE) RETURNS INT) Domains of Attributes

Un sous-type :

CREATE TYPE **Student** UNDER **Person** (PUBLIC style VARCHAR)

Types tuple (ROW type)

Types tuple	Types tuple nommé
CREATE TABLE employees (name CHAR(40), address **ROW**(street CHAR(30), city CHAR(20), zip ROW(original CHAR(5), plus4 CHAR(4)))); INSERT INTO employees VALUES('John Doe', ('2225 Coral Drive', 'San Jose', ('95124', '2347')))); Permet de stocker des tuples complets en variable et de les passer en argument à des fonctions (ici adresse)	**CREATE ROW TYPE** account_t (acctno INT, cust REF(customer_t), type CHAR(1), opened DATE, rate DOUBLE PRECISION, balance DOUBLE PRECISION,); CREATE TABLE account OF account_t (**PRIMARY KEY acctno**); Un type tuple nommé peut être utilisé pour définir un type référence

Les tables dans SQL3

Les tables peuvent avoir des attributs :

- de type ADT
- avec des valeurs complexes (SET, MULTISET, LIST, ROW)
- de type référence (REF <type> ou avec OID)

Possibilité d'utiliser un type prédéfini : ADTs ou ROW types

CREATE TABLE cars OF car ;

Possibilité de définir un nouveau type : le type est celui des tuples de la table

CREATE TABLE Customers OF NEW TYPE Customer (name VARCHAR, Chiffre_affaire MONEY) ;

Possibilité de définir des sous-tables :

CREATE TABLE Paintings UNDER Artifacts (Location REF(Museum))

Note : Location REF fait référence à un type tuple nommé

Héritage dans SQL3

Au niveau des types : sous-types

CREATE TYPE Student UNDER Person (PUBLIC style VARCHAR)

Au niveau des classes : sous-classes

```
CREATE TABLE person
  (name CHAR(20),
   sex CHAR(1),
   age INTEGER);

CREATE TABLE employee UNDER person
  (salary FLOAT);
CREATE TABLE customer UNDER person
  (account INTEGER);
```


3. Types et fonctions dans SQL3

- **Les types BLOBs et CLOBs**
- **Les constructeurs de types dans SQL3**
- **Les fonctions de SQL3**
- **Le parcours de référence**
- **Les tables imbriquées**

Support de large objets : Les types BLOBs et CLOBs)

- **Type BLOB (Binary Large Object)**
- **Type CLOB (Character Large Object)**

- Sont des types définis pour supporter des très gros objets
- Les instances de ces types sont stockées directement dans la BD

```
CREATE TABLE employees
  (id INTEGER,
   name VARCHAR(30),
   salary us_dollar,
...
   resume CLOB(75K),
   signature BLOB(1M),
   picture BLOB(12M));
```


Les constructeurs de types de SQL3

Les constructeurs de base : collections SET(T), MULTISET(T), LIST(T)
CREATE TYPE person (…, prénoms LIST(varchar), tel SET(phone))

Les références d'objets : possibilité de référencer un objet créé "sans OID"
CREATE TYPE car (number CHAR(9), color VARCHAR, owner REF(person))

Types tuples nommé ou non possibilité de référencer un objet créé "sans OID"

- CREATE TYPE Artist UNDER Person (..., influences SET (ROW (name: CHAR(50), date DATE)));
- CREATE ROW TYPE Museum(denomination VARCHAR, addr Address);

Types distincts : déclare que 2 types équivalents doivent être traités comme 2 types se données différents

- CREATE DISTINCT TYPE US_dollar AS DECIMAL(9,2)
- CREATE DISTINCT TYPE Canadian_dollar AS DECIMAL(9,2)

Constructeurs additionnels : stack, queue, array, insertable array (ex : texte)

- non intégrés dans le langage mais peuvent être ajoutés

Les types paramétrés :

- possibilité de types paramétrés (TEMPLATE)

Les fonctions de SQL3

Définies par l'utilisateur, associées à une base, un type, une table, …

Syntaxe :

FUNCTION <function name> <parameter list> RETURNS <type> AS <F body> END FUNCTION
<F body> = <SQL procedure> I <file name>

Exemple :

```
FUNCTION fullAddr (a Address)
RETURNS CHAR(45) AS
BEGIN
:z = findZip (a.street, a.city);
RETURN (a.street ll" "ll a.city ll" "ll z);
END ;
END FUNCTION
```

```
FUNCTION findZip (CHAR(20), CHAR(15))
RETURNS CHAR(10) AS
EXTERNAL NAME './findzip.so'
LANGUAGE C;
END FUNCTION
```


Les fonctions de SQL3 (suite)

Langage de programmation :

- SQL et SQL3 PSM, Langage externe

Appel de fonctions :

```
SELECT r.name
FROM emp j, emp r
WHERE j.name = 'Joe' and distance (j.location,r.location) < 1 ;
```

Nom de l'employé le plus proche de Joe - appel de la fonction « distance «

Note : Address Column est un ADT

Le parcours de référence

Possibilité d'appliquer les fonctions Ref et DeRef (implicite)

- CREATE TABLE cars OF TYPE car
- SELECT c.Owner.name FROM cars c WHERE color = 'red'

Possibilité de cascader la notation pointée

- SELECT dname FROM dept WHERE 1985 IN auto.years

Généralisation possible aux chemins multiples

- SELECT dname FROM dept
  WHERE autos.(year=1985 and name = 'Ford')

Toute collection peut être utilisée en place d'une table

Tables imbriquées

Services

N°	Chef	Adresse	Employés		Dépenses		
			Nom	Age	NDep	Montant	Motif
24	Paul	Versailles	Pierre	45	1	2600	Livres
			Marie	37	2	8700	Mission
					3	15400	Portable
25	Patrick	Paris	Eric	42	5	3000	Livres
			Julie	51	7	4000	Mission

4. Conclusion

- **Comparaison avec le relationnel**
- **Exemple d'application aux Systèmes d'Information Géographiques**
- **Conclusion**

Comparaison avec le relationnel

Accès en relationnel :

```
SELECT effdate, name, vehicleyr
FROM policy, customers, vehicles
WHERE policy.custno = customers.custno
AND policy.vehicleno = vehicles.vehicleno
AND model = 'ferrari'
```

Accès en objet-relationnel :

```
SELECT p.effdate, p.name, p.vehicleyr
FROM policy p
WHERE p.carmodel.make = 'ferrari'
```

Exemple d'application aux Systèmes d'Information Geographiques

(Gardarin)

Type Geométrie :

- Point, ligne, polygone, chemin, rectangle, ellipse, ...

Fonctions :

- distance(geom,geom) returns real
- contained(geom,geom) returns bool
- overlaps(geom,geom) returns bool
- intersection(geom,geom) returns geom
- union(geom,geom) returns geom
- ....

Exemple d'application aux SIG (suite)

(Gardarin)

Images Type Library

Différents formats : TIFF,GIF,FAX,CD,JPEG

Fonctions :

- rotate(image,angle) returns image
- transpose(image) returns image
- flip(image) returns image
- enhance(image), oil_painting(image)
- plus(image,image), minus(image,image)
- intersection(image,image), union(image,image)
- histogram(image) returns(table)
- similarity(image,image)

Conclusion : SQL3 un standard en évolution

Proposition concurrence de l'ODMG : OQL

- Accord entre constructeurs de SGBD Objets
- Support du modèle pur objet de l'OMG
- Variation de SQL traitant des collections imbriquées

Accord ANSI X3 H2 et ODMG :

- Définition d'un langage d'interrogation intégrant relationnel et objet
- Convergence relationnel-objet vers SQL3

De nombreux points restent à fixer :

- Visibilité des OID ?
- Identité des ROW type ?
- Chemins multivalués ?
- Intégrité référentielle ?