

Building Database Applications with SQL

ISO/IEC JTC1 SC32 Opening Plenary
22 June 2009
32N1892

Keith W. Hare
Convenor, ISO/IEC JTC1 SC32 WG3
JCC Consulting, Inc.
600 Newark Road, P.O. Box 381
Granville, OH 43023 USA
+1.740.587.0157
Keith@jcc.com

Introduction

We have been developing the SQL Standard for over twenty-five years.

- Large number of SQL implementations
 - More-or-less adhere to the standard
 - Some companies have several implementations
 - SQL is a tool to implement databases and database applications
- My background is database administration and implementing database applications

Topics

- Standard SQL
- What distinguishes SQL?
- Building Applications
 - Data Definitions
 - Stored Procedures and Functions
 - Access Control
- Summary

Standard SQL

The following is a short, incomplete history of the SQL Standards – ISO/IEC 9075

- 1987 – Initial ISO/IEC Standard
- 1989 – Referential Integrity
- 1992 – SQL2
 - 1995 SQL/CLI (ODBC)
 - 1996 SQL/PSM – Procedural Language extensions
- 1999 – User Defined Types
- 2003 – SQL/XML
- 2008 – Expansions and corrections

What Distinguishes SQL?

Several factors distinguish SQL from other query and data storage technologies:

- Persistent data accumulated over long periods
- Complex update and read-only transactions
- Concurrent update and read-only access
- Null values and three-valued logic
- Integration with other technologies

SQL is a tool for building
databases
AND
database applications!

Building Applications

The simplified steps in building a database application are:

- Data Definitions
- Stored Procedures and Functions
- Access Control

Data Definitions

SQL Schema Definition Language statements are used to implement a data design.

- Data modeling and design is a critical step in any database application
 - Not in the scope of this presentation
- Domains and Types
- Tables
- Constraints

Domains and Types

- Built-in Data types
 - Character
 - Fixed Length
 - Variable Length
 - National Character Set
 - Numeric
 - Fixed Precision
 - Floating
 - Large Objects (LOBS)
 - Character Large Object (CLOB)
 - Binary Large Object (BLOB)
- Derived Types

Derived Types

- Domains
 - Based on built-in types
 - Domain constraints
 - Not widely implemented
- User Defined Types
 - Potentially Complex
 - Can include methods

Domain Examples

```
create domain test_domain char(1)
    check (test_domain in ('A', 'B', 'C') or
test_domain is null)
    not deferrable;
```

```
Create domain Last_name_domain char(30)
    default ' ';
```

UDT Example

```
create type interval_t is object
(
  start_number number,
  end_number number,
  member function
    is_within(val number)
return number,
  member function
    num_overlaps(val interval_t)
return number
);
```

```
create type body interval_t is
member function is_within(val number) return number is
begin
    if val between self.start_number and self.end_number
    then
        return 1;
    else
        return 0;
    end if;
end is_within;
member function num_overlaps(val interval_t) return number
is
begin
    if val.start_number < self.end_number
    and self.start_number < val.end_number
    then
        return 1;
    else
        return 0;
    end if;
end num_overlaps;
end;
```

Tables

- Columns & Rows
- Number of rows is limited by storage space
- Actual details of table storage are beyond the scope of the standard
- Example – Three tables from an application

Example Partners

```
CREATE TABLE Partners
  (PartnerID                int Auto_increment NOT NULL
  ,PartnerName              nvarchar(50) NOT NULL
  ,StartDate                 datetime NOT NULL
  ,EndDate                   datetime NULL
  ,FederalTaxID             nvarchar(12) NOT NULL
  ,PrimaryAddressType       nvarchar(12) NULL
  ,PrimaryTelephoneNumberType nvarchar(12) NULL
  ,PrimaryEmailAddressType  nvarchar(12) NULL
  ,PartnerType              nvarchar(12) NULL
  ,BirthDate                 datetime NULL
  ,SQLUsername               nvarchar(40)
  ,CONSTRAINT PartnersPrimaryKey PRIMARY KEY
  (
    PartnerID
  )
)
;
```

Table Example

```
CREATE TABLE Fund
  (FundID          int Auto_increment NOT NULL
  ,FundName        nvarchar(32) NOT NULL
  ,FundDescription nvarchar(255) NOT NULL
  ,GeneralPartnerID int NOT NULL
  ,FundInceptionDate datetime NOT NULL
  ,FundTerminationDate datetime NULL
  ,CONSTRAINT FundPrimaryKey PRIMARY KEY
  (
    FundID
  )
)
```

Example Tables

```
CREATE TABLE FundPartners
  (FundID          int NOT NULL
  ,PartnerID      int NOT NULL
  ,FundPartnerStartDate datetime NOT NULL
  ,FundPartnerEndDate  datetime NULL
  ,GPAssociateID  int NULL
  ,AccountNumber  nvarchar(16) NOT NULL
  ,CONSTRAINT FndPrtnrPrimaryKey PRIMARY KEY
    (FundID
    ,PartnerID
    )
  )
```

Constraints

Constraints are rules that describe what it means for the data to be logically consistent.

- Enforced by the database
- Primary Key – unique identifier
- Foreign Key – referential integrity
 - Column in one table must exist as a primary key in another table
 - Bi-directional
- Referential Actions

Primary Key Example

```
CREATE TABLE Fund
  (FundID          int Auto_increment
  , ...
  , CONSTRAINT FundPrimaryKey PRIMARY KEY
  (
    FundID
  )
)
```

Foreign Key Example

```
ALTER TABLE FundPartners  
  ADD CONSTRAINT FK_FundPartners_Fund  
    FOREIGN KEY(FundID)  
    REFERENCES Fund (FundID)
```

;

```
ALTER TABLE FundPartners  
  ADD CONSTRAINT FK_FundPartners_Partners  
    FOREIGN KEY(PartnerID)  
    REFERENCES Partners (PartnerID)
```

;

Stored Procedures and Functions

- Stored in the SQL database
- Functions return a value, procedures do not
- Procedural Constructs
 - Begin – End blocks
 - IF – Then – Else – End if
 - Loops
 - Exception Handling
- SQL is a programming language

Function Example

```
create function CalculateMonthlyBasis(currentFund int,  
    SummaryMonth date)  
returns int not deterministic  
begin  
    declare done int default 0;  
    ...  
    DECLARE FundCsr CURSOR FOR  
        SELECT FundID  
            ,PartnerID  
        FROM FundPartners  
        where FundId = currentFund;  
  
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;  
  
set SummaryMonthDay1  
    = cast(concat(extract(year from SummaryMonth) ,"-"  
                ,Extract(month from SummaryMonth), "-1")  
        as date);
```

Function Example (Cont.)

```
Set Last_Month_date = SummaryMonthDay1 - interval 1 month;  
set NextMonthDay1 = SummaryMonthDay1 + interval 1 month;
```

```
open FundCsr;
```

```
  repeat
```

```
    fetch FundCsr into CurFundID, CurPartnerID;
```

```
    if not done
```

```
    then
```

```
      set counter_loop = counter_loop + 1;
```

```
      -- do work here...
```

```
    end if;
```

```
until done end repeat;
```

```
close FundCsr;
```

```
return Counter_Loop;
```

```
end
```

Access Control

- User Authentication
- Grant & Revoke
- Roles
- Views

Access Control – User Authentication

- Largely left to the SQL Implementation
- Implemented either in the database or as a part of the Operating System environment
- No external standard for authentication when SQL originally published

Access Control – Grant & Revoke

```
grant select, insert, update, delete  
  on Partners  
  to Keith;
```

```
revoke delete  
  on partners  
  from Keith;
```

Access Control – Roles

- Roles allow access control to be managed at a higher level

```
Create Role Fund_Administrator;
```

```
Grant Read, Write, Modify, Delete  
on Partners  
to Fund_Administrator;
```

```
Grant Role Fund_Administrator  
to Keith;
```

Access Control – Views

- A view is essentially a stored query
- Can grant a user access to a view while denying the user access to the underlying table

View Example

```
create view View_PartnerInformation as
  select current_user
         ,p.PartnerName
         ,...
  from partners p
 where p.SQLUsername = current_user
;
```

```
Grant read on View_PartnerInformation to public;
```

SQL supports database applications

- Define and manipulate tables
- Build application logic as functions and procedures
- Apply security to encapsulate access

Questions?

