

S.Q.L. in SQL

David Andruchuk

Sr. Architect

Computer Systems Design Associates, Inc.

What can *i* do.....*i* can do SQL



SQL(Some Quick Lessons) in SQL

What are we covering today?

SQL Objects (Objects written in SQL and known to the DBMS)

- File type (Tables, Indexes, Views, Aliases)
- Program type (Stored Procedures, Triggers, CTEs, UDFs)

SQL Catalog Tables to assist us

- SYSCOLUMNS
- SYSINDEX
- SYSIXADV
- SYSTABLES
- Etc...

SQL Rules of Thumb

SQL(Some Quick Lessons) in SQL

What is an Internal SQL Object?

An internal SQL object is pure SQL code

What is an External SQL Object?

An external SQL object is a mix of SQL and HLL code

What is the advantage of coding SQL over HLL?

Performance, Flexibility, Development Resources, Interfacing with other languages (.NET, JAVA)

Object Types (SQL vs Traditional)

SQL
Schema / Collection
TABLE
INDEX
VIEW
Row
Column
Log
TRIGGER
STORED PROCEDURE
UDF
UDT

OS/i5
Library
Physical File
Keyed Logical File
Non-keyed Logical File
Record
Field
Jounal

SQL(Some Quick Lessons) in SQL

SQL Object (File Types)

Tables

New Data Types:

- Small Integer
- Integer
- Big Integer
- Decimal
- VarChar
- Vargraphics
- Identity
- Row Change Timestamp

Short Table Name (**CUSTMAST**) and Long Table Name (**Customer_Master**)

- Both reside in SYSTABLES catalog view
- Can be referenced by either name

Short Column Names (**CUSTNUMB**) and Long Column Names (**Customer_Number**)

- Both reside in SYSCOLUMNNS catalog view
- Can be referenced by either name

SQL(Some Quick Lessons) in SQL

SQL Object (File Types)

Tables (continued...)

Auto Generated Values:

Identity Column

- Used as PRIMARY KEY instead of business data
- GENERATED ALWAYS results in the generation of a new value automatically when row is inserted into table
- GENERATED BY DEFAULT only generates value if the auto-generated column is NOT included in the data set being inserted
- Next Identity Column Value is stored in SYSPSTAT system catalog view (Can be changed)

Row Change Time Stamp

- System updated timestamp anytime row is inserted/updated
- Implicitly hidden (You do not need to know the column name to use it in SQL statements)

SQL(Some Quick Lessons) in SQL

SQL Object (File Types)

Indexes

Two Types:

Binary Radix

- Used for most indexes

Encoded Vector Index (EVI)

- Used for “low cardinality” instances (Low number of index occurrences relative to number of rows in table)
 - Can be extremely powerful index
 - Not compatible with some utilities (DBU, etc..)
-
- Use index advisor or search SYSIXADV in System Catalog to determine index needs
 - Indexes are more efficient in maintenance than a LF
 - Do not reference an index in a Select statement or HLL. Let the query optimizer determine the best access for the data

SQL(Some Quick Lessons) in SQL

SQL Object (File Types)

Indexes (continued...)

Create primary key on table

- Use identity column as PK
- Business data should **NOT** be used as PK

➤ Use index to create a unique key based on business data

Can have both long and short name

- If long name < 11 characters, then both are the same value
- If long name specified, short name is not, then system assigns the short name (**not recommended**)

SQL(Some Quick Lessons) in SQL

SQL Object (File Types)

Indexes (continued...)

Indexes can contain:

- Expressions
Create Index TableAI1 on TableA (Qty * Price)
- Functions
Create Index TableAI1 on TableA (Upper(Name))
- Aggregates
Create EVI TableAE1 on TableA (Date) include(Sum(QtySold))
- Non-Keyed Columns
Create Index TableAI1 on TableA (Department) Add Columns ID, Name)
- Row Filtering
Create Index TableAI1 on TableA (Name) where Department='Payroll'

SQL(Some Quick Lessons) in SQL

SQL Object (File Types)

Views

With a View you can:







- Can contain SQL Statements (Joins, select/omit, CTEs, etc..)
- Is basically an SQL Statement(s) that exist as a permanent object
- Are in effect a non-keyed LF that have minimal storage associated with them...
- Provide logical layer between programs and data (2nd phase of Modernization roadmap)
- Perform well since SQL engine develops the execution plan ahead of time
- Require no maintenance like a LF
- Can contain derived columns
- Can define selection criteria that far exceed anything that is remotely possible in DDS
- Can be defined to contain summary information
- Can be used by another View
- Use of views can make any data we have defined on our system that much easier for users to decipher
- Use iSeries Navigator's GUI intuitive interface to define a your views
- Must be accessed via embedded SQL in HLL (sorry no F-Specs)

SQL(Some Quick Lessons) in SQL

SQL Object (File Types)

Views (continued...)

Sample types of Joins to be used in a View:

Join Type	Description	Picture
Inner Join	Rows from table "A" that exist in table "B"	
Left (Outer)	All rows in table "A" and matching rows in Table "B"	
Right (Outer)	All rows in table "B" and matching rows in Table "A"	
Cross	All rows from table "A" are connected to all rows in table "B"	Cartesian Product
Left Exception	All rows from table "A" that are NOT in table "B"	
Right Exception	All rows in table "B" that are NOT in table "A"	
Full Outer	All rows in table "A" that are not in table "B" and all rows in table "B" that are not in table "A"	

SQL(Some Quick Lessons) in SQL

SQL Object (File Types)

Aliases

Using Aliases you can:

- Can be used to define an alias on a table, partition of a table, view, or member of a database file at the current or remote server
- Define a permanent object (until Dropped) over a member of a multi-member file
- You can use this alias name in an SQL statement in the same way that a table name is used
- Allow applications that use middleware such as ODBC, JDBC, OLEDB or .Net data providers to access multiple member files w/o an OVRDBF
- There are several restrictions on SQL statements that can be used against an SQL alias name that references a member

SQL(Some Quick Lessons) in SQL

SQL Object (Program Types)

Stored Procedures(SP)

Stored Procedure Types:

- Internal (SQL) SP
- External (RPG / JAVA / C##...) SP

Should be used as a middle layer between RPG and Views for data access

Can return a result set that can be read by:

- RPG SQL programs
- JAVA
- .NET
- Etc...

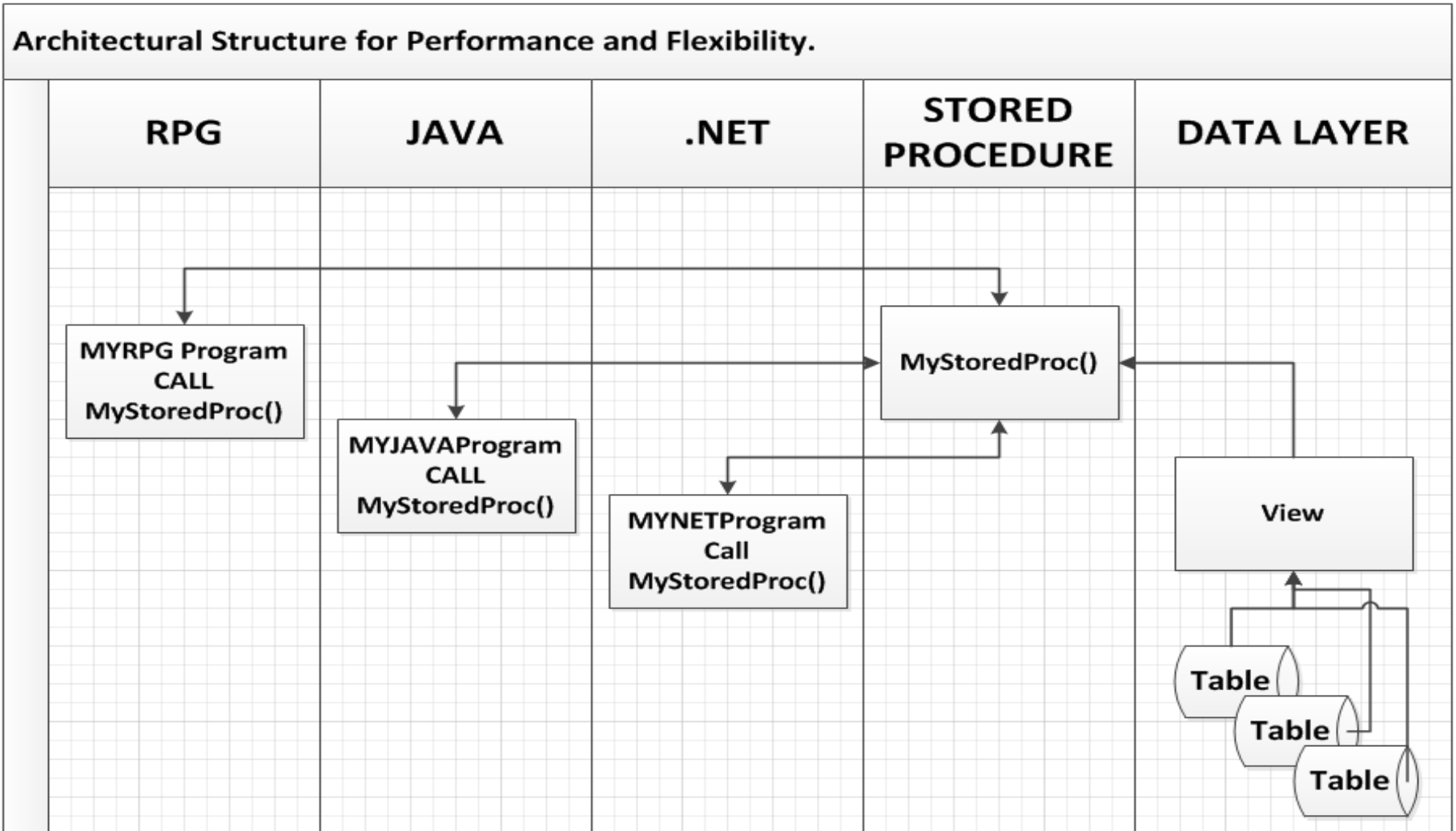
Architectural Structure:

- Significantly more flexibility
- One program can serve data to many different languages and technologies
- Consistency of results
- SQL based SP tend to be faster in execution times than external SP

SQL(Some Quick Lessons) in SQL

SQL Object (Program Types)

Stored Procedures(SP) (continued...)



SQL(Some Quick Lessons) in SQL

SQL Object (Program Types)

Triggers

Triggers are programs attached to DDL Tables / Views or DDS PFs

- You can use DSPFD to display attached triggers
- You can query SYSTRIGGERS in QSYS2 to view triggers

Trigger cause the specified program to fire on database events

- Insert / Delete / Update / Read

Trigger event can be controlled as to when it occurs

- After an event occurs
- Before an event occurs

➤ Journaling must be active

➤ Maximum of 300 triggers per Table / View/PF

Triggers can be used to:

- Record information when master data changes
- Perform function on changed data After / Before written to disk
- Encrypt data automatically

SQL(Some Quick Lessons) in SQL

SQL Object (Program Types)

Triggers (continued...)

- Caution should be heeded and program called thought through for error condition handling as if a trigger program halts the table it is attached to could be unusable until the trigger is fixed
- As of DB2 for i 7.1, the FieldProc technology was added to deliver transparent column-level encryption implementations (Robin Tatam has excellent presentation on this subject)
- As with any Journaling related event, your performance may vary so plan accordingly....

SQL(Some Quick Lessons) in SQL

SQL Object (Program Types)

Common Table Expressions (CTE)

There are two common types of CTEs used:

- Nested
- With Table

Nested appears as a sub-select on the From clause

```
Select field1, field2,  
From (select field1, field2 from TableA where.....)
```

With Table appears as WITH statement prior to SQL statement

```
With CTE_Example as  
(Select field1, field2 From (select field1, field2 from TableA where.....))
```

```
Select * from CTE_Example
```

- Can reduce the need for work files
- Can break SQL statements into more usable forms
- Can be used Recursively in SQL statement

SQL(Some Quick Lessons) in SQL

SQL Object (Program Types)

Common Table Expressions (CTE) (continued...)

Here is another example of a CTE that could be used in an SQL View

```
WITH destinations (origin, departure, arrival, flight_count) AS
  (SELECT a.departure, a.departure, a.arrival, 1
    FROM flights a
    WHERE a.departure = 'Chicago'
  UNION ALL
  SELECT r.origin, b.departure, b.arrival, r.flight_count + 1
    FROM destinations r, flights b
    WHERE r.arrival = b.departure)
SELECT origin, departure, arrival, flight_count
FROM destinations
```

SQL(Some Quick Lessons) in SQL

SQL Object (Program Types)

User Defined Function (UDF)

UDFs allow you to define your own custom SQL Function to be used in any SQL Statements (think *Add, Multiply, Date*, etc...)

UDFs Language Types:

- Internal – SQL
- External – RPG, JAVA, other HLL.....

There are two types of UDFs used:

- Scalar
- Table

Scalar returns one value

Select DateYMD(YY:MM:DD) as WorkDate....

or

Select * From TableA where DateYMD(YY:MM:DD) =

or

WorkDate = DateYMD(YY:MM:DD)

SQL(Some Quick Lessons) in SQL

SQL Object (Program Types)

User Defined Function (UDF) (continued...)

Sample Scalar UDF using SQL code:

```
CREATE FUNCTION DATEYMD
(YR DECIMAL(2,0), MO DECIMAL(2,0), DY DECIMAL(2,0))
  RETURNS DATE
  LANGUAGE SQL
  RETURNS NULL ON NULL INPUT
BEGIN
  DECLARE EXIT HANDLER FOR SQLEXCEPTION
  BEGIN
    RETURN CAST(NULL AS DATE);
  END;
  RETURN DATE((2000+YR) || '-' || MO || '-' || DY);
END
```

SQL(Some Quick Lessons) in SQL

SQL Object (Program Types)

User Defined Function (UDF) (continued...)

Table returns a Table (used in From clause)

```
Select A.Schema, A.Journal
```

```
From Table(GetJournalInfo('Table_Name', '*LIBL')) as A
```

Sample Table UDF using SQL code:

```
CREATE FUNCTION GETJOURNALINFO  
(inTable varchar(128), inLibrary varchar(128) )  
returns table  
(Journal char(10), Library char(10) )  
Language RPGLE  
Specific GETJRN001  
Not Deterministic  
Disallow Parallel  
Reads Sql Data  
Returns Null on Null Input  
External Name GETJRN001  
Parameter Style DB2SQL
```

SQL(Some Quick Lessons) in SQL

System Catalog Tables

The *system catalog* consists of tables and views that describe the structure of the database.

Sometimes called the *data dictionary*, these table objects contain everything that the database knows about itself.

- Each system catalog table contains information about specific elements in the database.
- Each database has its own system catalog.

The system catalog tables maintain information about the database, including the following categories of database objects:

- Tables, views, synonyms, and table fragments
- Columns, constraints, indexes, and index fragments
- Distribution statistics for tables, indexes, and fragments
- Triggers on tables, and INSTEAD OF triggers on views
- Procedures, functions, routines, and associated messages
- Authorized users, roles, and privileges to access database objects
- LBAC security policies, components, labels, and exemptions
- Data types and casts
- User-defined aggregate functions
- Access methods and operator classes
- Sequence objects

SQL(Some Quick Lessons) in SQL

System Catalog Tables (continued...)

- Storage spaces for BLOB and CLOB objects
- External optimizer directives
- Inheritance relationships
- XA data sources and XA data source types
- Trusted user and surrogate user information

SYSCOLUMNS

- The syscolumns system catalog table describes each column in the database

SYSINDEXES

- The sysindexes table is a view on the sysindices table. It contains one row for each index in the database.

SYSIXADV

- The sysixadv system catalog table contains “logging” information about required access paths for each table in the database

SYSTABLES

- The systables system catalog table contains a row for each table object (a table, view, synonym, or in IBM Informix, a sequence) that has been defined in the database, including the tables and views of the system catalog

SQL(Some Quick Lessons) in SQL

SQL Rules of Thumb

- Do not use “SELECT * FROM” in SQL Statements
- No DDL objects opened in “F” specs (or use USROPN and Faux open)
- The optimum number of files accessed in a program is 1 (use Views to bring data together into your programs)
- Use views between your programs and the physical table. Do Not Access Physical Tables Directly
- Use indexes to support your views
- Every SQL statement (in views or programs) should be analyzed by the index advisor to determine index needs
- Compile recursively called SQLRPGLE modules / programs with *ENDACTGRP or *ENDJOB
- Use both short names and long names for columns names and Table names
- Use short and long names for Views and Indexes

SQL(Some Quick Lessons) in SQL

SQL Rules of Thumb (continued...)

- Use INTEGER data type when zero precision is used instead of NUMERIC()
- Every table should have a primary key. The PK should be an identity column
- Create Business key as Unique but not PK
- Do not reference access paths (LF / Indexes) in a SQL query. Always access Table or View. (Preferably View)
- Use DATE data type instead of numeric(8,0)
- Use defaults when possible. i.e. CURRENT_DATE, CURRENT_TIMESTAMP, etc...
- Always GET DIAGNOSTICS at the end of every fetch. Interrogate SQLCODE at the end of every EXE SQL SELECT
- Save DDL / SQL source in Source Files just like DDS and RPG
- Embedded SQL statements should be as simple as possible. Complex SQL statements should be in views and accessed in HLL

THiNK

An exploration into making the world work better

