IBM i/DB2 Modernization to SQL

AXON iVY
digitalize your business

OpenLegacy

ca technologies

cm FIRST
Rethink Modernization

About Your Presenter

IBM i Professional for 16 Years.

Primary Focus is IBM i Engineering / Programming

Well Versed in 2E.

Well Versed in RPG (All Flavors)

Well Versed in CM Products such as Implementer

Up to Date on Newest SQL/RPG Enhancements.

Agenda

What Exactly is SQL (Just in case you don't know. ☺)
Benefits of Moving to SQL Created Objects
SQL Performance Benefits
SQL Objects vs DDS Objects
SQL Source vs DDS Source
Changing a 2E Model to Support SQL
SQL Specific Data Types

What Exactly is SQL?

SQL Stands for Structured Query Language.

Developed at IBM in the Early 1970s.

Consists of a DDL (Data Definition Language)

    IE, CREATE TABLE, CREATE INDEX

Consists of a DML (Data Manipulation Language)

    IE, INSERT INTO, UPDATE, DELETE,

To a Large Extent Consists of Simple English Statements.

Benefits of Moving to SQL Created Objects

Whole World Uses SQL. New resources are more familiar with SQL then DDS.
Long Field Names are Available to Outside World. YAY! ☺
Performance (Further discussion later)
Data is Validated when Written to the Database.
DDL Defined Indexes use a 64K page size vs 8K used by Logical Files.
Null Capable Fields.
New Data Types (Blob, Clob, Lob) Demo of this!

Benefits of Moving to SQL Created Objects

No Enhancements to DDS Language.

Constraints are Built into Database.

Built in Encryption.

Easier Modification of Table Layouts.

RPG Programs that Use SQL Access Don’t Need a Re-compile.

Automatic Identity Columns.

SQL Performance

In DDS Data is Validated When *Read*. In SQL Data is Validated When *Written*.

This is Important as we Normally do many more reads vs. writes. Ratio could be as high as 25 to 1.

Data Can Arrive up to 11 Seconds Faster. Especially Helpful When Loading up Subfiles from Random Keys.

Concurrent Write Support.

Logical Page Size of 64K.

SQL Performance

In DDS Data is Validated When *Read*. In SQL Data is Validated When *Written*.

This is Important as we Normally do many more reads vs. writes. Ratio could be as high as 25 to 1.

Data Can Arrive up to 11 Seconds Faster. Especially Helpful When Loading up Subfiles from Random Keys.

Concurrent Write Support.

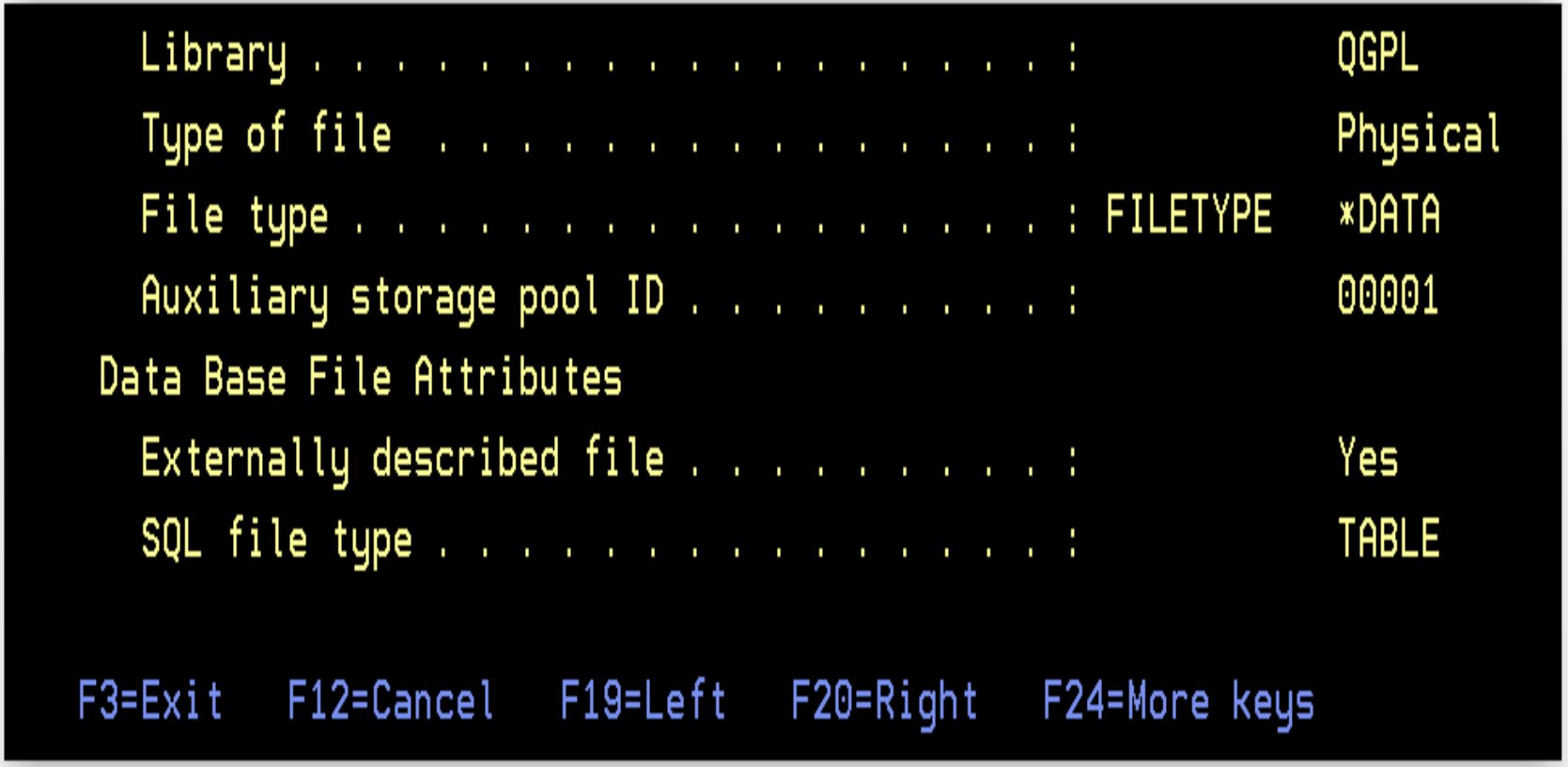
Logical Page Size of 64K.

SQL Objects vs DDS Objects

DDS Objects Create Physical Files & Logical Files
The Object types are *FILE PF and *FILE LF
Most common source is DDS and stored in QDDSSRC.
Source Member type is PF for Physical files, and LF for Logical Files.
Compiled like a program using option 14 from PDM, or using the CRTPF or CRTLF commands.

SQL Objects create Physical Files & Logical Files as well.
The Object types are *FILE PF and *FILE LF
Most common source is SQL and stored in QSQLSRC.
Source Member type can be named anything.

SQL Objects vs DDS Objects DSPFD



SQL Objects vs DDS Objects DSPFD

```
   File type . . . . . . . . . . . . . . . . . . : FILETYPE   *DATA
   Auxiliary storage pool ID . . . . . . . . . . :            00001
 Data Base File Attributes
   Externally described file . . . . . . . . . . :            Yes
   SQL file type . . . . . . . . . . . . . . . . :            VIEW

F3=Exit   F12=Cancel   F19=Left   F20=Right   F24=More keys
```

```
   Auxiliary storage pool ID . . . . . . . . . . :            00001
 Data Base File Attributes
   Externally described file . . . . . . . . . . :            Yes
   SQL file type . . . . . . . . . . . . . . . . :            INDEX
                                                                  Mo
F3=Exit   F12=Cancel   F19=Left   F20=Right   F24=More keys
```

SQL Source vs DDS Source

DDS Source is stored in source physical file QDDSSRC (Most Common)
Source is then compiled using CRTPF or CRTLF commands. (Option 14 in PDM)

```
A*-------------------------------------------------------------------
M* Maintenance   :
A*===================================================================
A          R FFOREW3                    TEXT('AP Transaction')
A*-------------------------------------------------------------------
A            FOABCD          4  0       TEXT('Company Code')
A                                       COLHDG('Company' +
A                                              'Code' )
A                                       EDTCDE(3)
A            FOEICD          6          TEXT('VEN Vendor Code')
A                                       COLHDG('Vendor' +
A                                              ' Code' )
A            FOESCD         14          TEXT('ATR Invoice Number')
A                                       COLHDG('Invoice' +
A                                              'Number' )
A            FONXNB          3  0       TEXT('ATR Activity Sequence')
```

SQL Source vs DDS Source

SQL Source is stored in source file QSQLSRC (Most Common)

Source member type can be named anything.

However, common examples are

SQLT for tables, SQLV for views and SQLI for indexes.

SQL Objects are created using the RUNSQLSTM command.

SQL Source vs DDS Source

Example SQL Table Source.

```
EXEC SQL
  CREATE TABLE AAAQCPP (
      SPOOLFILE FOR COLUMN AQAQVN       CHAR(10)
      NOT NULL WITH DEFAULT
     ,SPOOLFILE_NUMBER FOR COLUMN AQA3NB       DECIMAL(10,0)
      NOT NULL WITH DEFAULT
     ,DATE_CREATED FOR COLUMN AQADDZ       DATE
      NOT NULL WITH DEFAULT
     ,TIME_CREATED FOR COLUMN AQADTZ       TIME
      NOT NULL WITH DEFAULT
     ,USER_DATA FOR COLUMN AQATTX       CHAR(25)
      NOT NULL WITH DEFAULT
```

SQL Source vs DDS Source

Example SQL Table Source.

```
   RCDFMT FAQCPB2
 END-EXEC
```

```
EXEC SQL
  LABEL ON TABLE AAAQCPP IS
     'Spool File Details        Physical file'
END-EXEC
```

```
LABEL ON AAANCPP (
    ID TEXT IS 'ID'
   ,STATUS TEXT IS 'Status'
   ,FIRST_NAME TEXT IS 'First Name'
   ,LAST_NAME TEXT IS 'Last Name'
```

SQL Source vs DDS Source

Example SQL View Source.

```
CREATE VIEW AAAOCPL0 (
     PROGRAM_NAME FOR COLUMN AOAIVN
    ,USER_ID FOR COLUMN AOAJVN
    ,AUTHORIZED_TO_ADD FOR COLUMN AOAGST
    ,AUTHORIZED_TO_CHANGE FOR COLUMN AOAHST
    ,AUTHORIZED_TO_VIEW FOR COLUMN AOAIST
    ,AUTHORIZED_TO_DELETE FOR COLUMN AOAJST
```

SQL Source vs DDS Source

Example SQL View Source.

```
AS SELECT
    PROGRAM_NAME
   ,USER_ID
   ,AUTHORIZED_TO_ADD
   ,AUTHORIZED_TO_CHANGE
   ,AUTHORIZED_TO_VIEW
```

```
    FROM   AAAOCPP
    RCDFMT FAOCPBN
  END-EXEC
  EXEC SQL
    COMMENT ON TABLE AAAOCPL0 IS
       'Security                    Update index'
  END-EXEC
```

SQL Source vs DDS Source

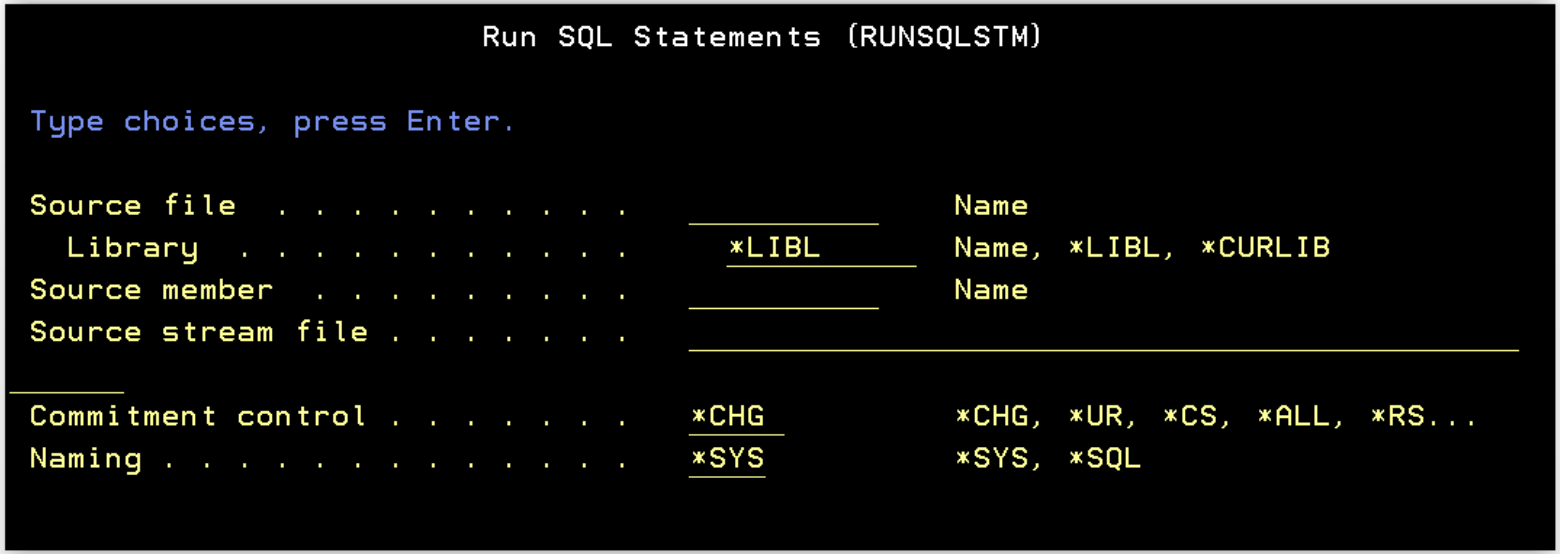
Example SQL Index Source.

```
EXEC SQL
  CREATE UNIQUE INDEX OPS01I02I
         ON OPS01T01
        (SECURITY_CODE ASC
        )
  RCDFMT FAKCPA8
END-EXEC
```

SQL Source vs DDS Source

Items to Note!

- SQL objects are created using the RUNSQLSTM Command.
- SQL View And Indexes produce separate IBM i Objects. (LFs)
- No problem combining SQL Created tables and DDS Created LFs. (Common)



DML (Data Manipulation Language)

DML is used to manipulate the database.

Examples are,

- SELECT (to read data)
- INSERT (to insert data)
- UPDATE (to update data)
- DELETE (to delete data)

Big advantage is the use of the DB2 Database Engine for automatic index selection!

- If an index exists that matches the search criteria then that index will be used automatically. In standard record level access from RPG this is not the case.

SQL DML Statements can be incorporated directly into an RPG program! There is no longer a need to use the traditional record level access provided by RPG.

DML (Data Manipulation Language)

Select (Statement Example)

```
C+      DECLARE AAAOCPL4XCSR     CURSOR FOR
C+        SELECT * FROM AAAOCPL4
C+            WHERE (AOAIVN     =   :AOAIVN
C+            AND    AOAJVN     =   :AOAJVN
C+                  )
C+        ORDER BY  AOAIVN         ASC,
C+                  AOAJVN         ASC
```

DML (Data Manipulation Language)

Select (Statement Example)

```
C/EXEC SQL
C+   OPEN AAAOCPL4XCSR
C/END-EXEC
```

```
C/EXEC SQL
C+   FETCH AAAOCPL4XCSR
C+     INTO :QAAAO1
C/END-EXEC
```

```
D QAAAO1          E DS                  EXTNAME(AAAOCPL4)
```

DML (Data Manipulation Language)

INSERT (Statement Example)

```
C/EXEC SQL
C+      INSERT   INTO AAAOCPL0
C+        VALUES(  :AOAIVN
C+               ,  :AOAJVN
C+               ,  :AOAGST
C+               ,  :AOAHST
C+               ,  :AOAIST
C+               ,  :AOAJST
C+               ,  :AOAFST
C+               ,  :AOAGVN
C+               ,  :AOABDZ
C+               ,  :AOABTZ
C+               ,  :AOAHVN
C+               ,  :AOACDZ
C+               ,  :AOACTZ
C+               )
C/END-EXEC
```

DML (Data Manipulation Language)

UPDATE (Statement Example)

Can update multiple records at once!

```
C/EXEC SQL
C+      UPDATE AAAOCPL0
C+             SET AOAGST        = :AOAGST
C+               ,  AOAHST       = :AOAHST
C+               ,  AOAIST       = :AOAIST
C+               ,  AOAJST       = :AOAJST
C+               ,  AOAFST       = :AOAFST
C+               ,  AOAGVN       = :AOAGVN
C+               ,  AOABDZ       = :AOABDZ
C+               ,  AOABTZ       = :AOABTZ
C+               ,  AOAHVN       = :AOAHVN
C+               ,  AOACDZ       = :AOACDZ
C+               ,  AOACTZ       = :AOACTZ
C+         WHERE  ( AOAIVN       = :AOAIVN      AND
C+                  AOAJVN       = :AOAJVN    )
C/END-EXEC
```

DML (Data Manipulation Language)

DELETE (Statement Example)

Can delete multiple records at once!

```
C/EXEC SQL
C+     DELETE FROM AAAOCPL0
C+       WHERE ( AOAIVN        = :WAAIVN      AND
C+               AOAJVN        = :WAAJVN     )
C/END-EXEC
```

Change Model to use SQL Support.

Several different approaches can be taken to change a model to support SQL.

You can change the model to use the 2E generated table name, but use longer field name support. This allows you to expose the longer field names to the outside world. No change to RPG.

You can change the model to use both the longer table name, and also the longer field names. Again, easier for the outside world. No change to RPG.

You can change the generated function to use SQL record level access regardless of the database type.

Change Model to use SQL Support.

Any change to support SQL is done though the changing of model values.

Name	Description	Values
YSQLVNM	SQL naming	*DDS
YDDLDBA	Database Access Method	*RLA / *SQL
YSQLLEN	SQL naming length	10 to 25
YSQLOPT	Generate SQL OPTIMIZE clause	*NO
YSQLFMT	Generate SQL RCDFMT clause	*NO
YSQLSTM	SQL statement generation type	*EXC
YSQLCOL	Generate SQL Col/Library Name	*YES
YLVLCHK	Generate IDX with LVLCHK(*YES)	*NO
YDBFGEN	Method for database file generation	DDS / SQL / DDL
YSQLWHR	specifies whether to use OR or NOT logic when generating SQL WHERE clauses. The default is *OR.	
YSQLLCK		

Change Model to use SQL Support.

YSQLVNM relates to the naming scheme used for fields and files.

Values

- *DDS – Uses the DDS name
- *SQL – Use the names of the objects in the model.
- *LNG – Use long names for fields and files
- *LNF – Use long field names
- *LNT – Use long table names

Change Model to use SQL Support.

YDDLDBA

specifies a method of accessing the database

Values

*RLA

External function generates with RLA access.

*SQL

External function generates with SQL access.

Change Model to use SQL Support.

YSQLLEN – The length of field names up to 25 characters.

Values

Any number between 10 and 25.

Change Model to use SQL Support.

YSQLFMT

Specifies whether the RCDFMT keyword must be generated for SQL tables, views, and indexes.

Values

*YES

the RCDFMT value is calculated using the same rules as are used when DDS files are generated.

*NO

the record format is the same as the table, index, or view name.

Change Model to use SQL Support.

YDBFGEN – Specifies the language used in creating the logical/view over a physical file/table.

- *DDS
- *SQL
- *DDL

Change Model to use SQL Support.

YDBFGEN – Specifies the language used in creating the logical/view over a physical file/table.

- *DDS
- *SQL
- *DDL

Change existing table to SQL.

Generating SQL files

Extended Names?

Generating SQL Functions?

Generating RLA Functions over SQL Files

Change existing table to SQL.

There are options to consider when modernizing a file name to SQL in 2E.

Keep the file and field names the same.

Keep the file name the same, give the fields enhanced names.

Change the file and fields to enhanced names.

SQL Record access in Functions

Functions can be changed through model values, or one at a time to SQL access by the use of function options.

SQL Specific Data Types

Use BLOB data type of store binary data.
SQL Only Data Type! (No DDS Support)
BLOBs can be up to 2GB in size! YAY!
Created As Column In Table Definition.
Can Store ANY Binary Data (Files, Audio, Emails, .EXE Files)
Cannot Be Modified Outside of Program. (NOT in IFS)
Takes Advantage of IBM i Security.
Automatic Replication to Target HA System.

Wrap Up

Questions? Comments?