

# Moving from DDS to SQL



Rob Bestgen – [bestgen@us.ibm.com](mailto:bestgen@us.ibm.com)

IBM i – Rochester Development Lab

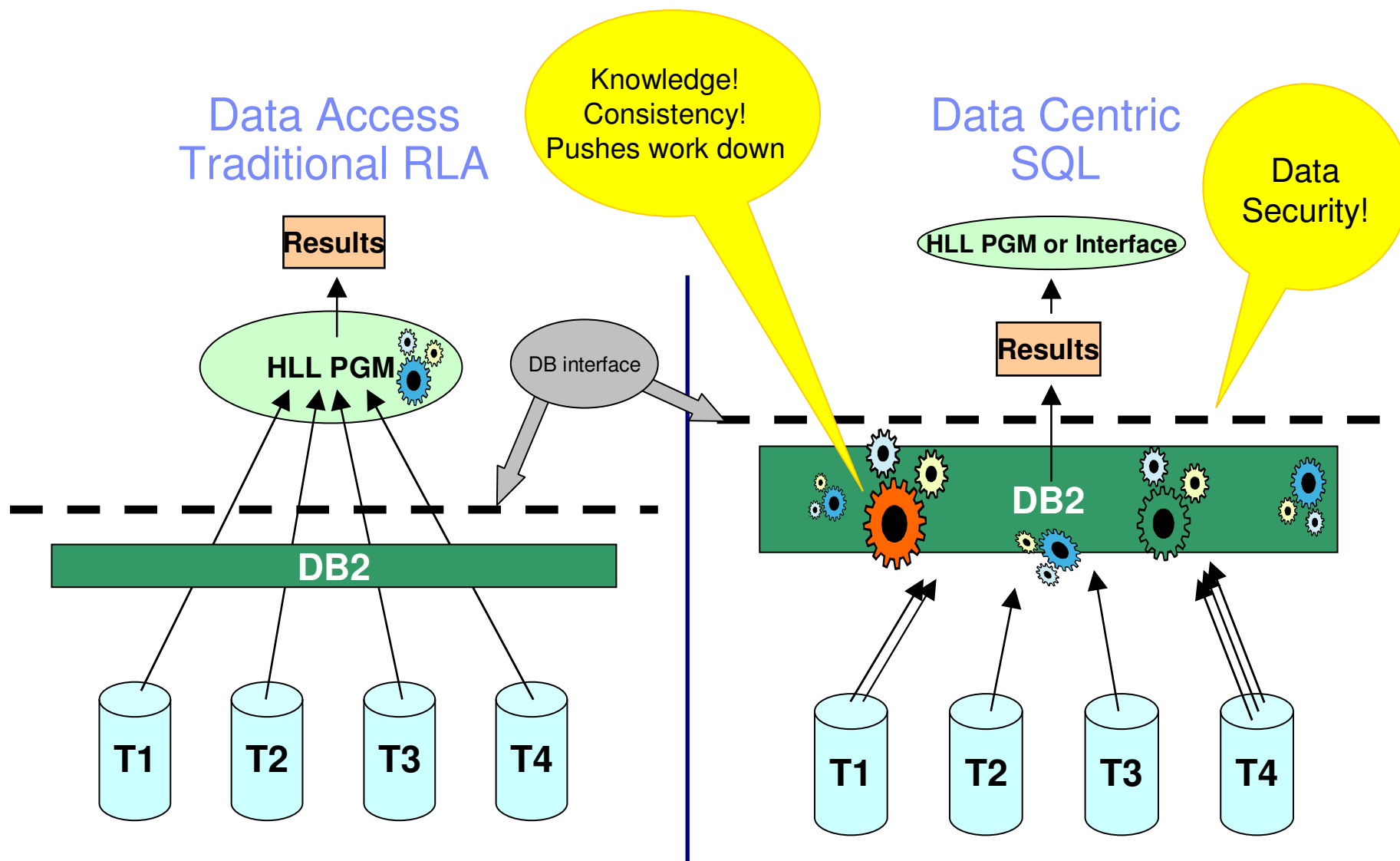


## Why move?

# Why move from DDS to SQL?

- Data-Centric programming
  - Let the Database do more for you!
- Take advantage of the latest DB2 technology
- Drive work into the database instead of the application
  - Improve consistency and efficiency
- Leverage new tools technology
- Open up new ways to access data
  - PHP, JDBC, ODBC, .NET, CLI

## Data Centric Programming – the SQL Difference



## Advantages, A Laundry List...

- Take advantage of features and functions only available via SQL
- Take advantage of modern solutions and tooling based on SQL
- Turn data into information
- Protect sensitive data
- Increase reusability of existing components in both current and future applications
- Increase the life expectancy and extend the value of legacy applications
- Reduce maintenance delays and overhead associated with altering the database
- Easier to embrace modern and/or evolving architectures and designs
- Reusability of data, functions, procedures, across platforms and systems
- Availability of talented SQL programmers
- Update skills of existing programmers

## Data Modeling Best Practices

- Use proper column definitions (i.e. type, length, precision, scale)
- Use only one key column to represent the relationship between any two tables
- Key columns should be of the same type and have the same attributes (i.e. type, length, precision, scale)
- Meaningless keys are acceptable and encouraged
- Define and use constraints
- Define and implement a proper indexing strategy
- Define and implement views to assist the programmers and users

## Protecting Your Investment in IBM i

### Rapid Application Development

- **SQL & RPG Integration**
- **Stored procedure Result Set consumption**
- **FIELDPROC for transparent column-level encryption**
- **XML Integration**
  - XML data type
  - Annotated XML Decomposition
  - SQL XML Publishing functions
- **Three-part Naming**
- **Compatibility with DB2 Family & Oracle**
  - **MERGE** statement
  - Array support & Global Variables
  - **REPLACE** option on **CREATEs**
  - Currently Committed supported
- **JDBC & .NET enhancements**

### Trusted Reliability

- **Enhanced Remote Journal filtering**
- **Library-level Journaling filtering**
- **IASP spanning transactions**

### Performance & Self-Tuning Enhancements

- **SQL Query Engine (SQE) enhancements**
  - Adaptive Query Processing
  - Self-Learning Optimization
  - Inline UDF query rewrite
  - **Logical File on FROM** support
- **Indexing Advancements**
  - SQL Select/Omit Indexes
  - OmniFind Text Indexes
  - EVI Aggregates
- **CPYFRMIMPF performance**
- **SSD & In-Memory Database Enablement**
- **OmniFind Text Search Server enhancements**

### Simplified Management

- **IBM i Navigator Enhancements**
  - Progress Monitors – Alter Table, Index Build
  - Index Advisor improvements
  - Enhanced Generate SQL capability
  - Object Folder content saves

### Data Intelligence & Interoperability

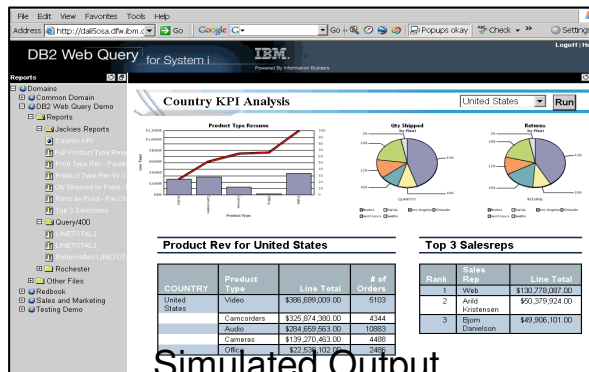
- **DB2 Web Query for System i**
  - Excel client support
  - Microsoft SQL Server adapter

## Modern Sophisticated Applications

The query behind this panel contains 4 unions of approximately 20 tables each.

Average response time 5-7 seconds.

Visual Explain of query providing data for pie chart



# Heavy lifting provided by DB2

## **Simply put:**

- ✓ Modernize your database objects
- ✓ Use SQL



## SQL Philosophy

With high level language record level access:  
You tell DB2 what you want, AND how to get it.

With SQL:  
You tell DB2 what you want, NOT how to get it.

DB2 Optimizer and Database Engine have more tools in the kit

## What is SQL?

- Data Definition Language (DDL)



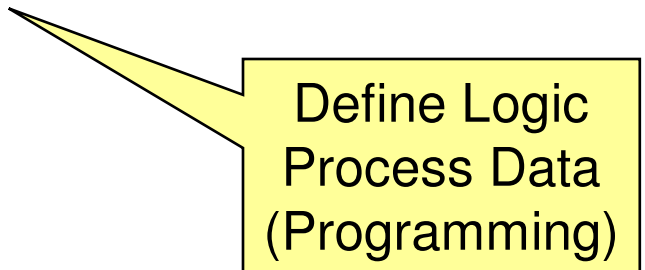
Create  
Alter  
Objects

- Data Manipulation Language (DML)



Identify  
Fetch  
Insert  
Update  
Delete  
Data

- Persistent Stored Modules (PSM)



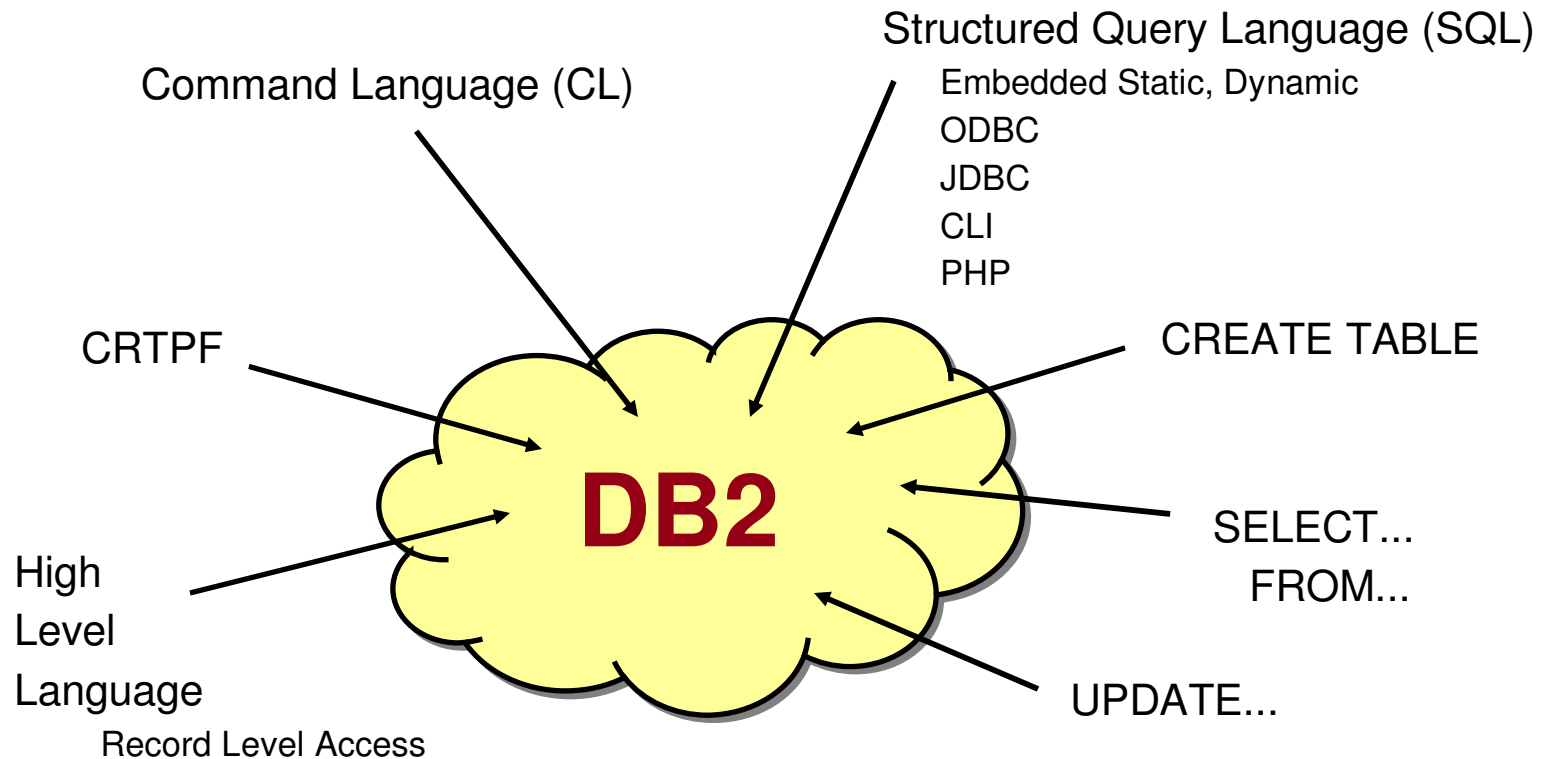
Define Logic  
Process Data  
(Programming)

The DB2 for i SQL Reference manual is your friend

# Moving to SQL

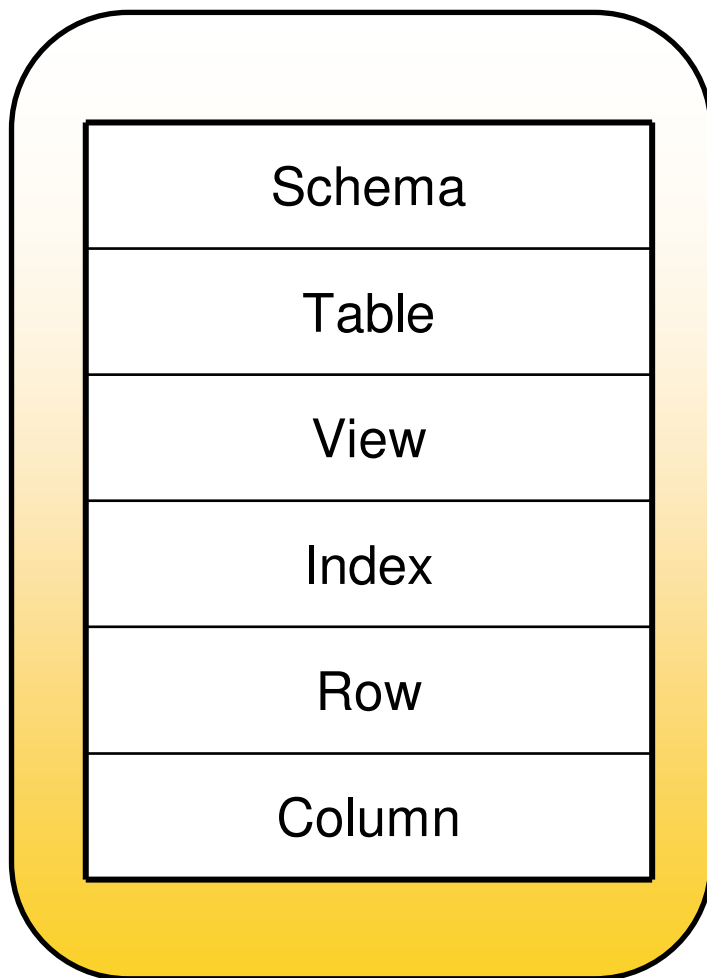
## Where to begin?

## One Database Management System with multiple interfaces

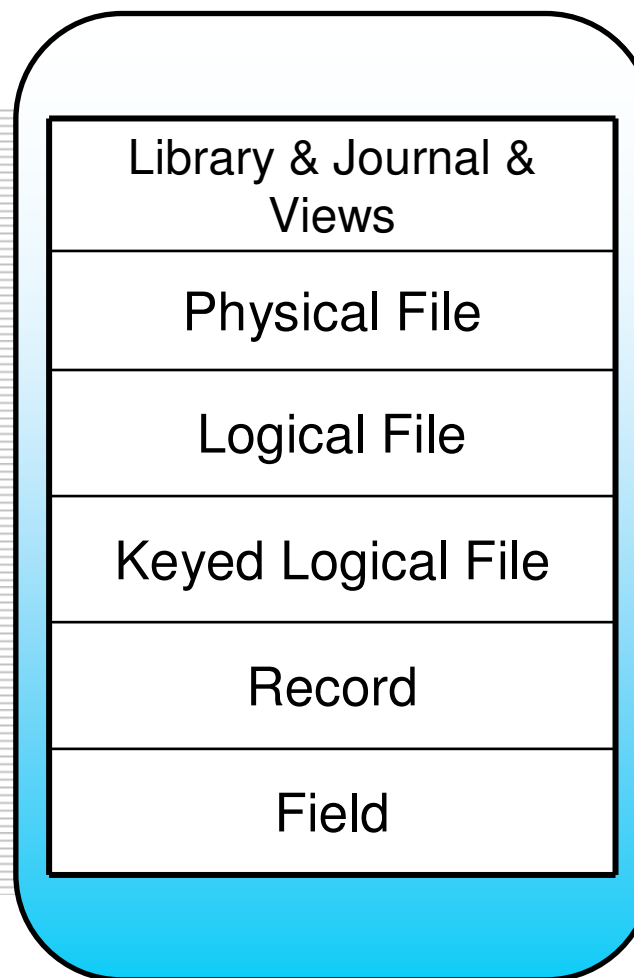


# Terminology

## SQL



## IBM i Native



## Moving to SQL

Moving to SQL can proceed in any of several ways

- Change files from native DDS to SQL DDL
- Rewrite existing applications to use SQL
- Develop new applications based on SQL

Where to begin?

They are all valid!

Let's focus on DDS to DDL

## Moving to SQL

Moving to SQL can be proceed in any of several ways

- Change files from native DDS to SQL DDL
  - Identify and Exploit SQL DDL Enhancements
  - Minimizes impacts to existing applications
  - Leverages SQL for additional features
- Rewrite existing applications to use SQL
- Develop new applications based on SQL

## Why SQL - Identify and Exploit DDL Enhancements

- Adding new columns takes advantage of data centric capabilities
  - Auto-generation fields
    - Identity Columns (Primary key)
    - Row change TIMESTAMP (optimistic locking, LCFO)
    - Sequence objects (Unique keys)
  - Implicitly hidden Columns
- Numerous additional table options
  - NOT LOGGED, VOLATILE, LIKE, partition tables, field procedures, ...
- Additional object types
  - Views, MQTs, EVIs, Sequence objects, Global Variables...
- Future enhancements



## CREATE TABLE (DDL) vs CRTPF (DDS)

```
CREATE TABLE EMP_MAST (
  EMP_MAST_PK FOR COLUMN EM_PK
  BIGINT GENERATED BY DEFAULT AS IDENTITY IMPLICITLY HIDDEN
  PRIMARY KEY,
  EMPNO CHAR(6) UNIQUE,
  FIRSTNME VARCHAR(12),MIDINIT CHAR(1), LASTNAME VARCHAR(15),
  EMP_PICTURE BLOB(102400) ,
  EMP_ROWID ROWID GENERATED ALWAYS,
  DL_PICTURE DATALINK(1000) DEFAULT
  EM_ROW_CHANGE_TS FOR COLUMN EMROWCHGTS TIMESTAMP
  NOT NULL FOR EACH ROW ON UPDATE AS ROW CHANGE
  TIMESTAMP IMPLICITLY HIDDEN)
```

```
CRTPF FILE(EMPLOYEE) SRCFILE(QDDSSRC)
  SRCMBR(EMPLOYEE)
ADDPFM FILE(QDDSSRC) MBR(EMPLOYEE)
--Source Data
A                               UNIQUE
A      R EMPLOYEE
A      EMPNO      6
A      FIRSTNME   12  VARLEN
A      MIDINIT    1
A      LASTNAME   15  VARLEN
A      K EMPNO
ADDPFCST FILE(EMPLOYEE) TYPE(*PRIKEY) KEY(EMPNO)
```

Many new data types and functions

Long names

Multiple constraints defined within statement

Self contained source statement

- store as IBM i source member or PC file

No new data types

Only 1 key per definition. Constraints must be manually added

Requires separate source member

Source member must exist on IBM i to be compiled

## Why SQL - CREATE TABLE Table-level Attributes

- NOT LOGGED INITIALLY clause
  - Temporary disable journaling
  - Can improve the performance of initial population of work or summary table
  - Requires the usage of commitment control
- Table volatility – VOLATILE or NOT VOLATILE (default value)
  - Describe to query optimizer on the size fluctuation of table
  - VOLATILE influences optimizer to use access method that performs consistently - regardless of table size
    - Good candidate: work tables with rows added & deleted continually
- Media preference for explicit SSD usage
  - UNIT SSD clause to recommend DB2 object be placed on SSD
  - Best object candidates:
    - Large amount of random data access and...
    - Data that is read many times, but written less frequently

## CREATE TABLE (& SQL) Naming Considerations

- SQL Column & Object names have maximum lengths of 128, but many IBM i utilities, commands and interfaces only support a 10-character length. How does that work?!?
  - System automatically generates a short 10 character name
    - First 5 chars with unique 5 digit number  
CUSTOMER\_MASTER >> CUSTO00001
- Might be different each time a specific table is created, depending on creation order and what other objects share the same 5 character prefix
- Use IBM i SQL syntax to specify your own short name
  - RENAME TABLE (tables & views) & RENAME INDEX
  - FOR COLUMN clause for columns
  - SPECIFIC clause for procedures, functions

## CREATE TABLE LIKE

- LIKE clause creates a new table with duplicate column definitions, no data copied
  - Closest to column duplication of CPYF CRTFILE(\*YES)
    - SQL Data copy support: INSERT INTO newtab SELECT \* FROM oldtab
  - Similar to CRTDUPOBJ DATA(\*NO) function, but LIKE does NOT duplicate constraints & triggers
  - When source object is DDS created, all non-SQL attributes are removed
- Parentheses make a difference
  - NO Parentheses, duplicates following “extra” column attributes
  - CREATE TABLE newtab LIKE oldtab
    - Default value, null attribute, column heading & text (ie, LABEL)
    - Identity attribute, hidden column attribute, row change attribute
  - Parentheses, ignores the “extra” column attributes  
CREATE TABLE newtab (LIKE oldtab)
  - Copy Options to explicitly control “extra” column attributes  
CREATE TABLE newtab LIKE oldtab EXCLUDING COLUMN DEFAULTS  
CREATE TABLE newtab (LIKE oldtab INCLUDING IDENTITY COLUMN ATTRIBUTES)

## CREATE TABLE AS SELECT

- Duplicates column definitions and optionally copy the data
  - Similar to CRTDUPOBJ DATA(\*YES) and OPNQRYF/CPYFRMQRYF combo
    - Constraints & triggers NOT duplicated
  - Same copy options available as CREATE TABLE LIKE
  - Simplifies creation of work & summary tables
  - Can enable the usage of DDS Field Reference files
- Example:
  - Field Reference File Usage:  
**CREATE TABLE customer AS**  
**(SELECT id cust\_id, lname cust\_lastname, fname cust\_firstname,**  
**city cust\_city FROM RefFile)**  
**WITH NO DATA**
  - General Usage:  
**CREATE TABLE Rpart\_summary AS**  
**(SELECT o.year, p.part, SUM(o.quantity) AS total\_quantity**  
**FROM orders o, parts p**  
**WHERE o.partkey = p.partkey AND o.year >= 2009 AND p.parttype='R'**  
**GROUP BY o.year, p.part ) WITH DATA**

## CREATE TABLE AS (<Materialized Query Table>)

```
CREATE TABLE EMP_MAST_SUM AS (
  SELECT WORKDEPT, SUM(SALARY)
  SAL_TOTAL
  FROM EMPLOYEE
  GROUP BY WORKDEPT
  ORDER BY SALARY_TOTAL DESC)
DATA INITIALLY DEFERRED
REFRESH DEFERRED
MAINTAINED BY USER
```

```
CRTPF FILE(EMMASTSUM)
  SRCFILE(QDDSSRC)
  SRCMBR(EMMASTSUM)
ADDPFM FILE(QDDSSRC) MBR(EMMASTSUM)
--Source Data
  A      R DEPTTOTAL
  A      WORKDEPT
  A      SALARY
OPNQRYF FILE(EMPLOYEE)
  FORMAT(EMMASTSUM)
  GRPFLD(WORKDEPT) MAPFLD
  (SAL_TOTAL '%SUM(SALARY)')
CPYFRMQRYP...
```

MQT Definition part of DDL  
DB2 for i Optimizer is aware of table  
Table must be manually  
refreshed/updated

Requires CL program for OPNQRYF  
and CPYFRMQRYP to do summary  
Table must be manually  
refreshed/updated

## Why SQL - CREATE TABLE AS (<Partitioned Table>)

```
CREATE TABLE PAYROLL_HIST
  (EMPNO INT, FIRSTNAME CHAR(15),
   LASTNAME CHAR(15), CHECK_AMT
   DEC(9,2),
   CHECK_TAX_YEAR SMALLINT)
PARTITION BY RANGE(CHECK_YEAR)
(STARTING FROM (MINVALUE) ENDING AT
(2007) INCLUSIVE,
STARTING FROM (2008) ENDING AT (2008)
INCLUSIVE,
STARTING FROM (2009) ENDING AT
(MAXVALUE)
```

**Partitions (members) are defined as part of DDL  
DB2 for i determines proper partition  
Only recommended for tables exceeding 1.7TB  
in size or 4.2 billion rows**

```
CRTPF FILE(PRHIST) SRCFILE(QDDSSRC)
  SRCMBR(PRHIST) MAXMBRS(n)
ADDPFM FILE(QDDSSRC) MBR(PAYROLLHST)
--Source Data
  A      R PRHISTR
  A      EMPNO
  A      FIRSTNAME
  A      LASTNAME
  A      CHECK_AMT
  A      CHECK_YEAR
ADDPFM FILE(PRHIST) MBR(YEAR2009)
OPNDBF FILE(PRHIST) OPTIONS(*BOTH)
  MBR(YEAR2009)
OVRDBF FILE(PRHIST) MBR(YEAR2008)
CLOF FILE(PRHIST)
```

Requires CL program to select proper member  
(programmer responsibility)

## CREATE VIEW vs CRTLF (non-keyed)

```
CREATE VIEW
  EMPLOYEE_BONUSES_BY_DEPARTMENT_WITHIN
    _STATE
AS
SELECT EA.STATE, DM.DEPTNAME, SUM(EM.BONUS)
FROM EMAST EM
JOIN EADDR EA USING (EM_PK)
JOIN DMAST DM ON WRKDPT = DPTNO
GROUP BY EA.STATE, DM.DEPTNAME
```

```
CRTLF FILE(EMPLOYEEJ1) SRCFILE(QDDSSRC)
  SRCMBR(EMPLOYEEJ1)
--Source Data
A    R EMPLOYEEJA JFILE(EMAST EADDR +
A                                DMAST)
A    J                                JOIN(1 2)
A                                JFLD(EM_PK EM_PK)
A    J                                JOIN(1 3)
A                                JFLD(WRKDPT DPTNO)
A    STATE
A    DEPTNAME
A    BPNUS
```

Full access to advanced query capabilities  
of SQL

No support for keying/ordering

Limited Join support

No support for Grouping, Case,  
Subqueries, User-Defined functions, ...



## CREATE INDEX vs CRTLF (Keyed)

```
CREATE INDEX EMP_LASTNAME_DEPT
ON EMP_MAST(WORKDEPT, LASTNAME)
RCDFMT EMPLOYEEER1
ADD COLUMNS EMPNO,FIRSTNME,MIDINIT

CREATE ENCODED VECTOR INDEX RegionIX
ON SALES(REGION)
```

```
CRTLF FILE(EMPLOYEEEL1)
      SRCFILE(QDDSSRC) SRCMBR(EMPLOYEEEL1)
--Source Data
A      R EMPLOYEEER1  PFILE(EMPLOYEE)
A      WORKDEPT
A      LASTNAME
A      EMPNO
A      FIRSTNME
A      MIDINIT
A      K WORKDEPT
A      K LASTNAME
```

Encoded Vector Index(EVI) support

Expressions can be used in the definition of the key columns

Sparse Indexes with WHERE clause (ie, Select/Omit)

EVI "Instant" Aggregate support

Only Binary Radix Tree structure support – no EVIs

Limited support for key derivations and expressions

Smaller default logical page size

## Why SQL – Example - DB2 i7.1 – Field Procedures

### ▪ Field Procedures

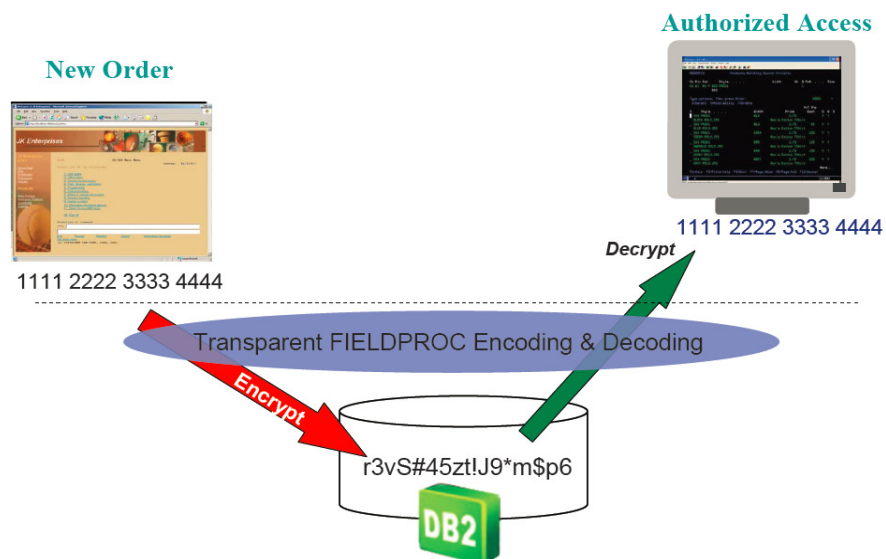
- Register a program that is called **anytime** a table column is read or written
- Allows control of both ‘at rest’ and viewed data

### ▪ Column Level Encryption

- Extension of field procedures
- Allows for transparent (no application changes) encryption of a specific column in a database table accessed through SQL or native
- Solutions from tool providers including Patrick Townsend, Linoma Software & nuBridges supply encryption algorithms

#### Note:

Be extremely careful of using CHGPF after altering SQL Table or PF



First Name	Last Name	City	State	Credit Card#
Megan	Jones	Minneapolis	Minnesota	*&^%\$*
Casey	Smith	Ames	Iowa	\$%@^

## Why SQL - CREATE INDEX – Expressions & Selection

- SQL key definitions support expressions, functions and operators enabling more usage of indexes by query optimizer on complex queries

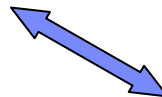
- Fully supported by SQE optimizer

- EXAMPLES:

```
CREATE INDEX ix_TotalSalary ON employees (Sales + Bonus AS TotalSalary)  
CREATE INDEX ix_FullName ON employees (CONCAT(CONCAT(FName, ' '), LName))  
CREATE ENCODED VECTOR INDEX yearqry ON dates_tab  
  (YEAR(ORDERDATE), QUARTER(ORDERDATE));
```

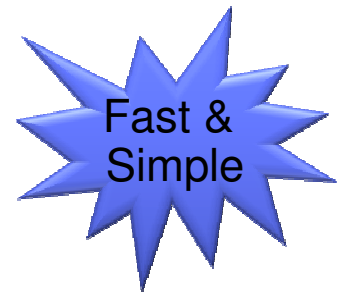
- Great for improving performance of case-insensitive searches

```
SELECT cust_id, cust_phone FROM customers  
  WHERE UPPER(company_name) = 'ACME'
```



```
CREATE INDEX ix_uCompName ON customers(UPPER(company_name))
```

- Sparse indexes – optimizer awareness added with 7.1 release
  - EXAMPLE: CREATE INDEX cust\_ix1 ON customers(cust\_id) **WHERE activCust='Y'**



## Why SQL – Example - CREATE INDEX – Encoded Vector Index

- Complementary indexing technology for boosting performance in analytical query & reporting environments
  - Patented technology that advances traditional bitmapped indexing
  - Best fit – columns with low cardinality (type, color, etc)

Example: `CREATE ENCODED VECTOR INDEX idx1 ON sales(region)`

- Encoded Vector Index (EVI) Aggregate Support (7.1)

```
CREATE ENCODED VECTOR INDEX idx1 ON sales(region)
```

```
    INCLUDE ( SUM(saleamt),  COUNT(*) )
```

```
CREATE ENCODED VECTOR INDEX idx2
```

```
    ON sales(territory)
```

```
    INCLUDE (SUM(saleamt + promoamt))
```

**EVI aggregates  
maintained as underlying  
table changes**

```
SELECT territory, SUM(saleamt+promoamt) FROM sales  
GROUP by territory
```

```
SELECT region, SUM(saleamt) FROM sales GROUP BY region
```

## Why SQL - CREATE SEQUENCE - Sequence Object

- Another DB2 construct that supports the automatic generation of column values
  - Viewed as a superset of Identity columns
  - Generated values easily shared across tables
  - Can support alpha-numeric key generation
  - Can create constant sequence to be used as Global DB2 variables (pre 7.1)

- Example:

```
CREATE SEQUENCE order_seq  
START WITH 1 INCREMENT BY 1 NO MAX VALUE
```

```
INSERT INTO orders(ordnum,custnum)  
VALUES (NEXT VALUE FOR order_seq, 123)
```

```
VALUES NEXT VALUE FOR order_seq INTO :hostvar
```

```
UPDATE orders SET ordnum = :hostvar  
WHERE custnum = 123
```

## CREATE SEQUENCE - Sequence Object

- Sequence values can be used to generate non-numeric key  
**CREATE SEQUENCE s START WITH 1001; ...**  
**SET alphakey = 'N' || CAST(NEXT VALUE FOR s AS CHAR(4))**
- Customizable Sequence Attributes
  - START WITH & INCREMENT BY
  - MINVALUE & MAXVALUE
  - CYCLE & NO CYCLE
  - CACHE & NO CACHE - To improve performance, DB2 allocates a block of sequence values at the job/connection level.
  - ORDER & NO ORDER - ORDER ensures that values are returned in the actual order that they are requested independent of the job/connection. NO ORDER is the default. ORDER also disables caching.
- Sequence attributes can be changed with the ALTER SEQUENCE statement

**CREATE SEQUENCE s1 NO CACHE ORDER**

**Job1: NEXT VALUE FOR s1 => VALUE = 1**

**Job 2: NEXT VALUE FOR s1 => VALUE = 2**

**Job 1: NEXT VALUE FOR s1 => VALUE = 3**

**Job 1: NEXT VALUE FOR s1 => VALUE = 4**

**Job 2: NEXT VALUE FOR s1 => VALUE = 5**

**CREATE SEQUENCE s1 CACHE 20 NO ORDER**

**Job1: NEXT VALUE FOR s1 => VALUE = 1**

**Job 2: NEXT VALUE FOR s1 => VALUE = 21**

**Job 1: NEXT VALUE FOR s1 => VALUE = 2**

**Job 1: NEXT VALUE FOR s1 => VALUE = 3**

**Job 2: NEXT VALUE FOR s1 => VALUE = 22**

## Why SQL - Identity Column

- Identity Column Attribute

- Attribute that can be added to any “whole” numeric columns
- Not guaranteed to be unique - primary key or unique index must be defined
- Only available for SQL tables, BUT identity column value generated for non-SQL interfaces (eg, RPG)

```
CREATE TABLE emp( empno INTEGER GENERATED ALWAYS AS IDENTITY  
                    (START WITH 10 , INCREMENT BY 10),  
                    name CHAR(30), dept# CHAR(4))
```

```
INSERT INTO employee(name,dept) VALUES('MIKE','503A') or...  
INSERT INTO employee VALUES(DEFAULT,'MIKE', '503A')
```

## Why SQL - CREATE VARIABLE – SQL Global Variables

- Enables simpler sharing of values between SQL statements and SQL objects (Triggers, Views, etc) across the life of a job/database connection

- Example #1 – Cache User Information

```
CREATE VARIABLE gvdept INTEGER DEFAULT
```

```
(SELECT deptno FROM employee WHERE empuserid = USER);
```

```
CREATE VIEW filtered_employee AS (
```

```
  SELECT firstname, lastname, phoneno FROM employee WHERE deptno = gvdept);
```

```
...
```

```
SELECT firstname, phoneno FROM filtered_employee;
```



## CREATE VARIABLE – SQL Global Variables

- Example #2 – Conditional Trigger Behavior

```
CREATE VARIABLE batch_run CHAR(1);
```

```
CREATE TRIGGER track_expenses AFTER INSERT ON expenses  
  REFERENCING NEW AS n FOR EACH ROW
```

```
WHEN (batch_run='N')
```

```
  BEGIN
```

```
    DECLARE emplname CHAR(30);
```

```
    SET emplname = (SELECT lastname FROM employee WHERE empid=n.empno);
```

```
    IF n.totalamount < 10000 THEN
```

```
      INSERT INTO travel_audit
```

```
        VALUES(n.empno, emplname, n.deptno, n.totalamount, n.enddate);
```

```
    ELSE
```

```
      SIGNAL SQLSTATE '38001' SET MESSAGE_TEXT='Exceeded Maximum';
```

```
    END IF;
```

```
  END
```

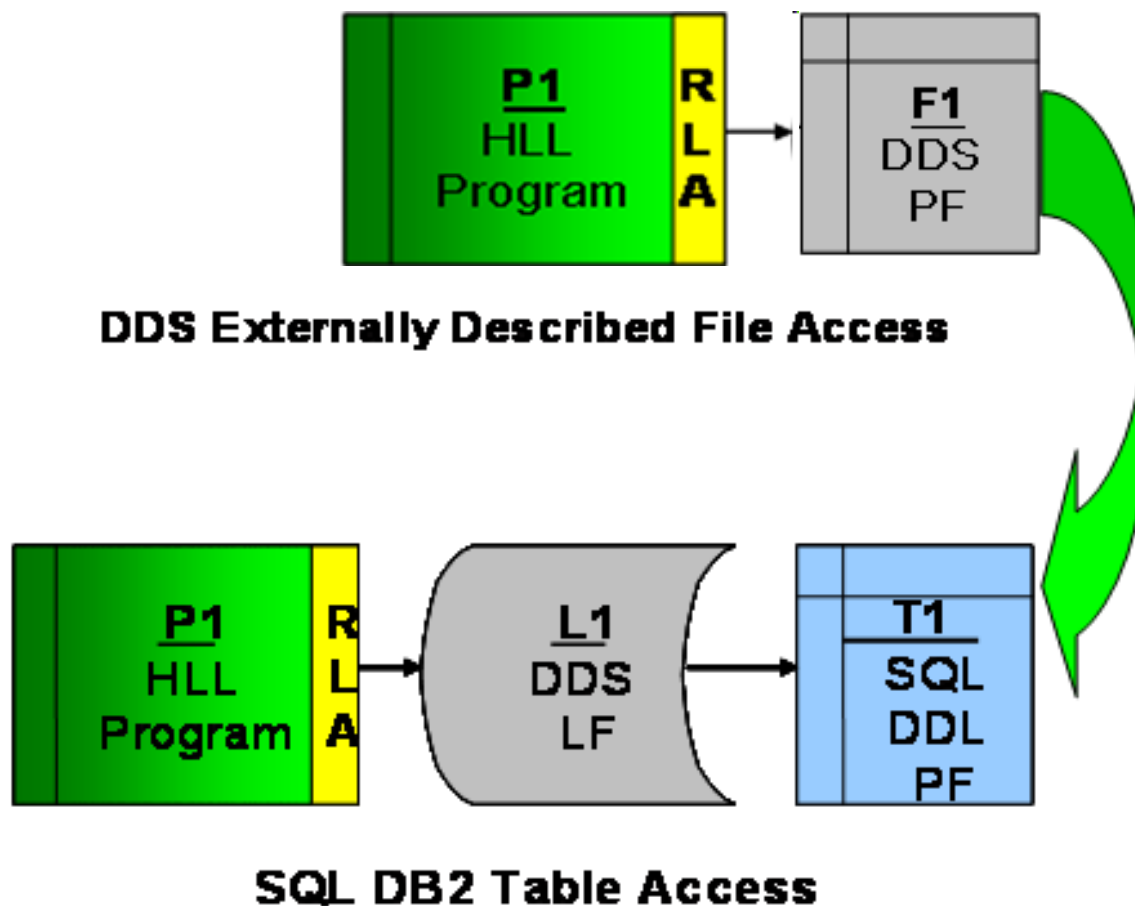
```
  ...
```

```
VALUES 'Y' INTO batch_run;
```

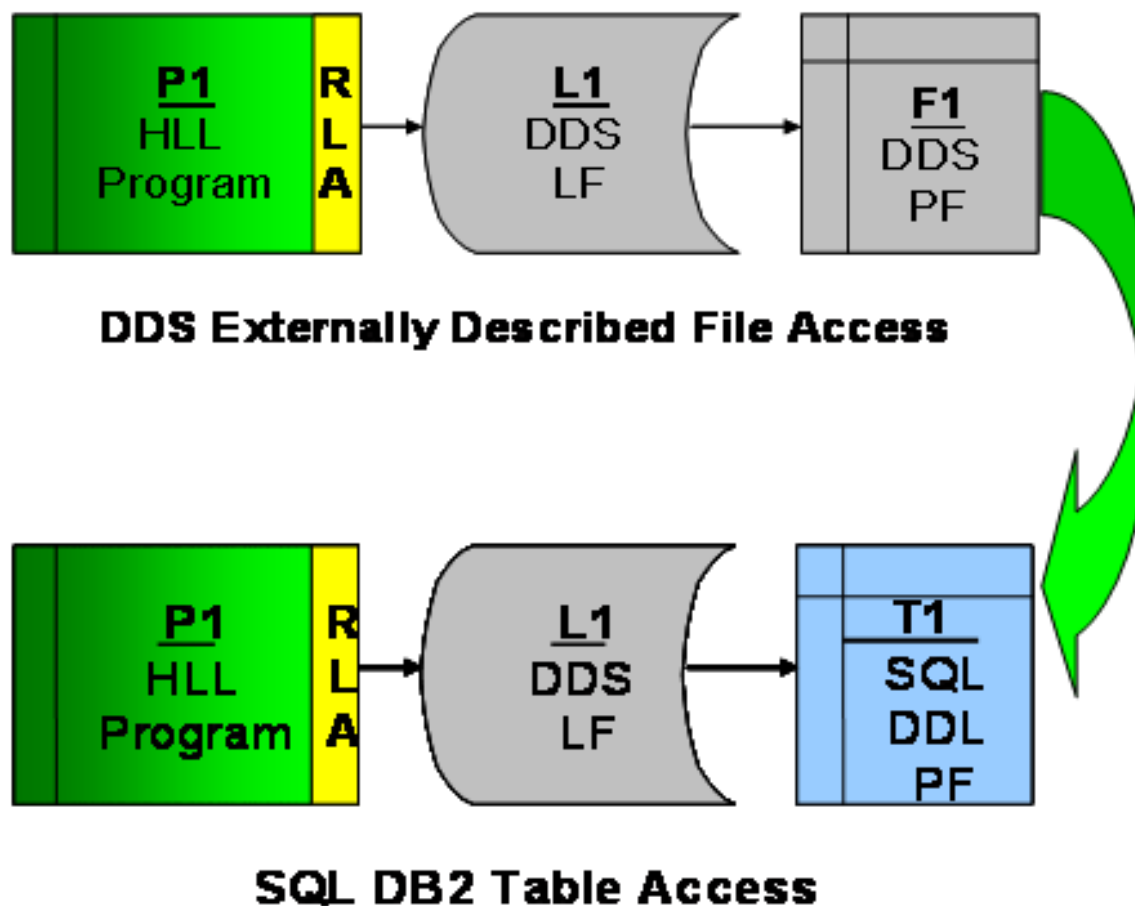
```
  ...
```

# Moving DDL into an existing environment

## Transparent Migration to SQL – Options

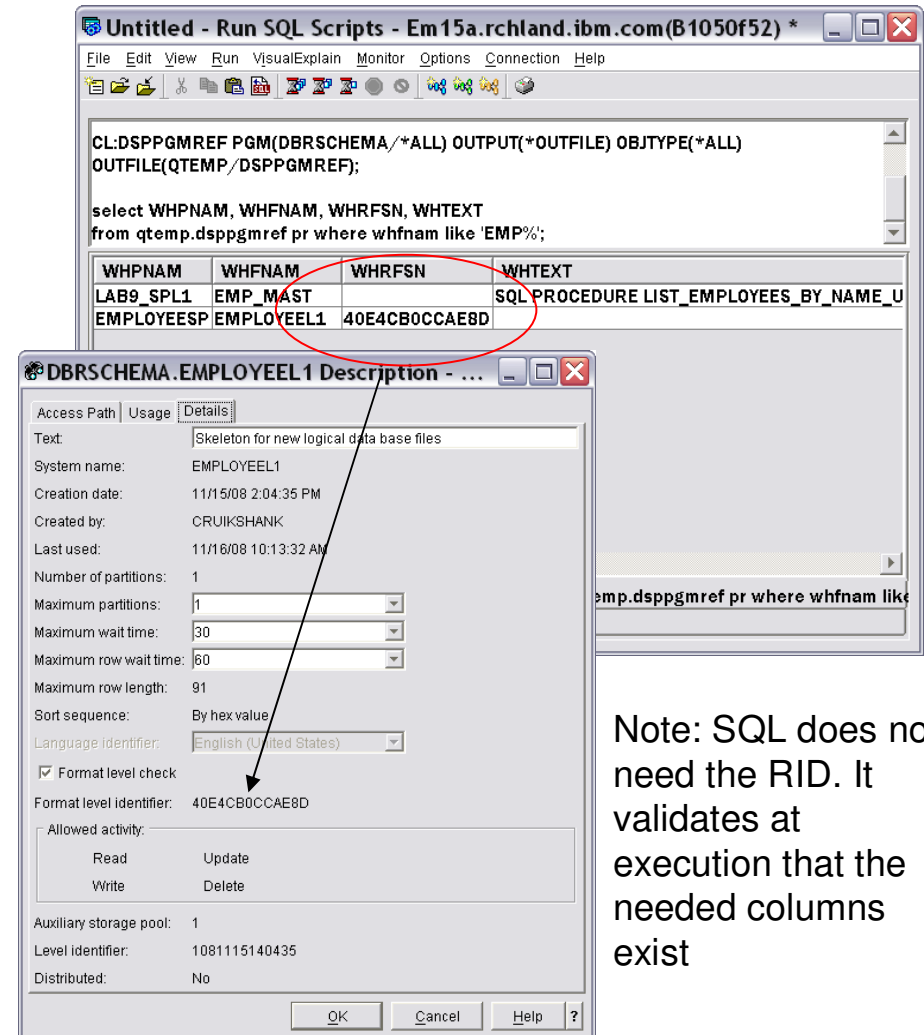


## Transparent Migration to SQL – Options...



## Beware the Format Level ID!

- A database object contains a **Record Format Level Identifier (RID)**
  - The RID is captured within a program object using Record Level Access (native)
  - Note: SQL does not care about RID
- The RID establishes integrity between the file and programs using native access
  - When the RID changes (i.e. column added or dropped) the program will break unless:
    - The program is created with Level Check = \*NO (**Not recommended**)
    - The program is recreated



Can we handle RID **AND** leverage new DDL support?

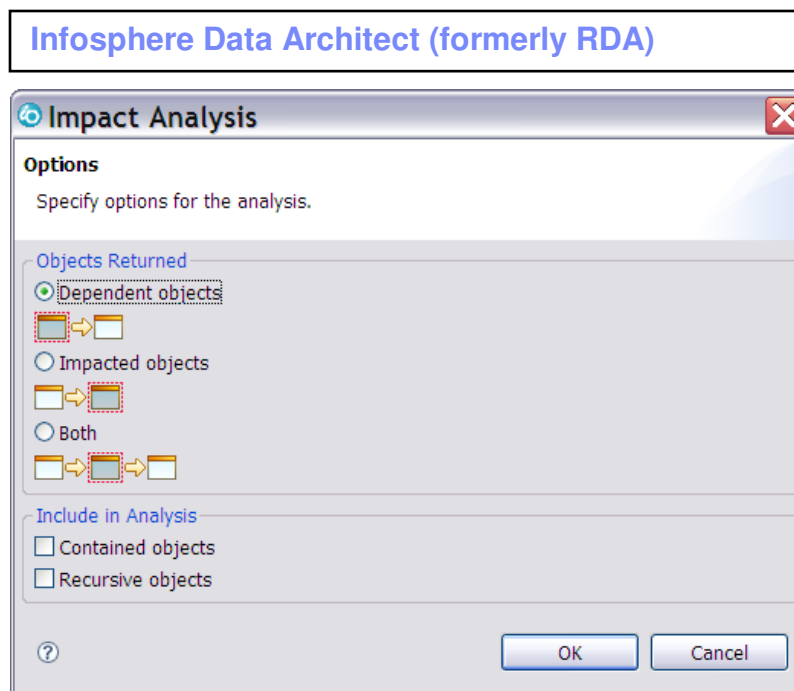
## CREATE TABLE Table-level Attributes

- RCDFMT keyword to alter default SQL behavior of having record format name be identical to the object name
  - RPG requires record format name to be different
  - Example:

```
CREATE TABLE dbtest/customer_master  
    (customer_name FOR COLUMN cusnam CHAR(20),  
     customer_city FOR COLUMN cuscty CHAR(40))  
RCDFMT cmfmt
```

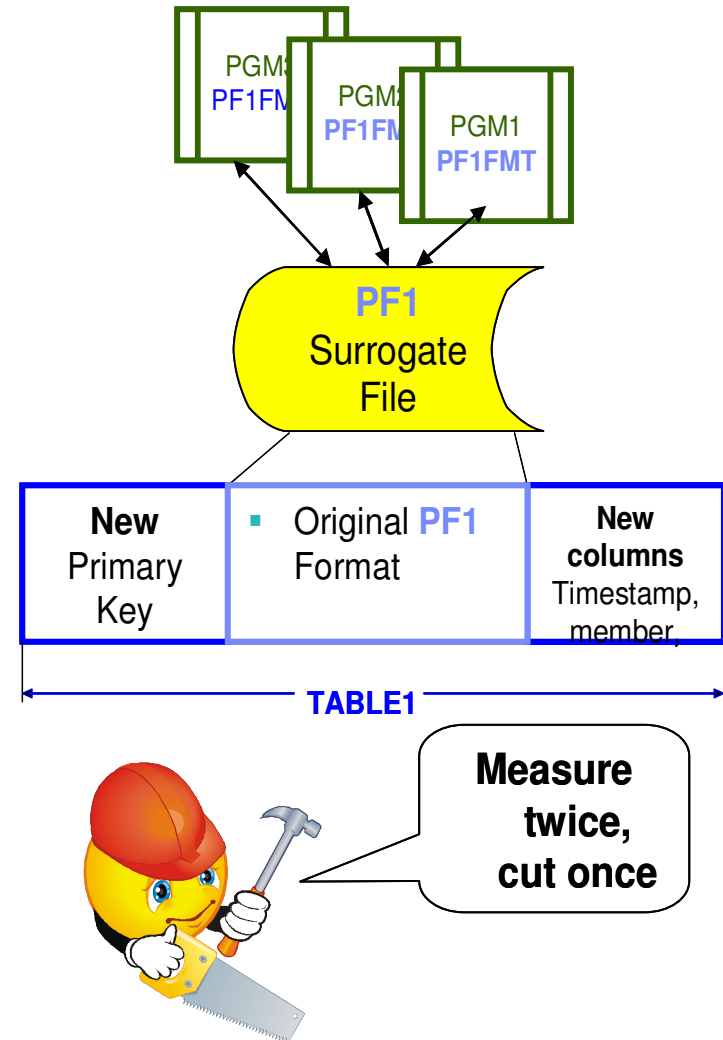
## Record Formats

- The format of the columns associated with a DDS file
- Not required for SQL
  - RCDFMT added to CREATE TABLE, VIEW and INDEX DDL for compatibility purposes
- Typically used for impact analysis
  - How many programs need to be changed if a column in record format 1 is changed?
- Used in conjunction with IBM i DSPPGMREF command
- Rational, combined with DB2 for i system catalogs provide equivalent support



## Adding New Columns to Re-engineered Table

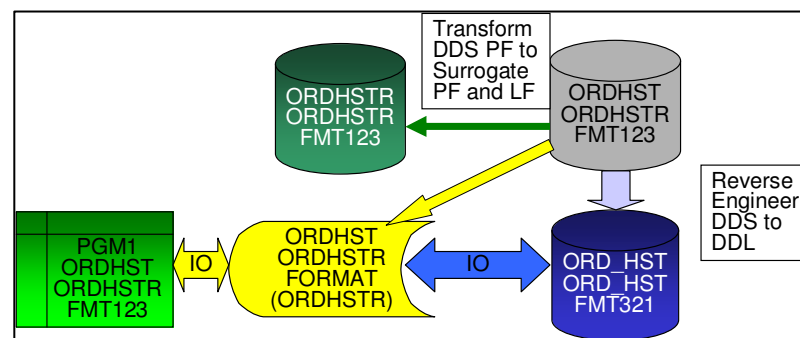
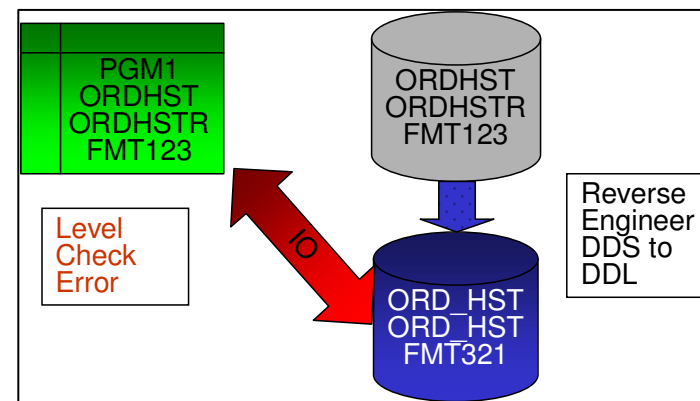
- Surrogate LF methodology enables converted SQL table to be enhanced with new features...
  - WITHOUT** changing ID of Surrogate!
    - New columns can be added before or after the original columns
      - Add Identity columns
      - Add Implicitly Hidden columns
    - Original column definitions can be altered





## Creating and sharing the “surrogate” file

- Reengineering DDS PF to DDL PF results in format identifiers being changed
  - HLL programs accessing the DDL PF will receive a “level check” exception message.
  - Only solutions prior to 5.4
    - recompile the program or
    - ignore the exception (**not recommended**).
- A **surrogate file** preserves the original DDS PF format.
  - Allows new columns to be added to DDL PF
  - FORMAT keyword used to share surrogate format
    - Prevents level check IDs for programs accessing original PF or LFs sharing format.
- This is the “best” method for avoiding format id changes



## Transparent SQL Migration - Example

### ■ Existing PF - INVENTORY

```
A R INVMTR
A   ITEM      15A
A   ORDER     10A
A   SUPPLY    15A
A   QTY        5P
A   QTYDUE     5P
A K ITEM
```

### ■ Existing LF - INVLF

```
A R INVMTR PFILE (INVENTORY)
A K ORDER
A K ITEM
```

### ■ Converted SQL Table –

```
CREATE TABLE sq_invent (
  item CHAR(15),
  order CHAR(10),
  supply CHAR(15),
  qty DECIMAL(5,0),
  qtydue DECIMAL (5,0))
```

### ■ Surrogate LF - INVENTORY

```
A R INVMTR PFILE (SQ_INVENT)
A   ITEM
A   ORDER
A   SUPPLY
A   QTY
A   QTYDUE
A K ITEM
```

### ■ Modified Existing LF - INVLF

```
A R INVMTR PFILE (SQ_INVENT)
A           FORMAT (INVENTORY)
A K ORDER
A K ITEM
```

## CREATE TABLE (& SQL) Naming Considerations

- Short & Long Name Co-existence Example
  - Specify the short name at creation:

```
CREATE TABLE dbtest/cusmst  
(customer_name FOR COLUMN cusnam CHAR(20),  
customer_city FOR COLUMN cuscty CHAR(40))
```

- Specify a long name for existing short-name:

```
RENAME TABLE dbtest/cusmst TO customer_master  
FOR SYSTEM NAME cusmst
```

- If long name specified on SQL Table definition, can also add/control the short name after table created:

```
RENAME TABLE dbtest/customer_master TO SYSTEM NAME cusmst
```

- RCDFMT keyword to alter default SQL behavior
  - RPG requires record format name to be different

```
CREATE TABLE dbtest/customer_master  
  (customer_name FOR COLUMN cusnam CHAR(20),  
   customer_city FOR COLUMN cuscty CHAR(40))  
RCDFMT cmfmt
```

## CREATE INDEX – Logical File Equivalency

- SQL indexes (& views) can be used by traditional native record-level access interfaces
  - By default, SQL indexes include all columns in Logical File record format
  - ADD clause provides ability to simulate logical file definitions
    - ADD ALL COLUMNS (default)
    - ADD KEYS ONLY
    - ADD col-name1, ...
  - ADD must be used in conjunction with RCDFMT keyword
  - SQL field ordering & format level identifier may be different than equivalent LF

**CREATE INDEX EMP\_LASTNAME\_DEPT**

**ON EMP\_MAST(WORKDEPT, LASTNAME)**

**RCDFMT** employeer1

**ADD COLUMNS** empno, firstname, middle\_initial

## Enhanced DDL TABLE and Surrogate DDS **LF**

```
CREATE TABLE CUST_MAST 1 (
  CUST_MAST_ID FOR COLUMN 2
  CUSTMASTID BIGINT GENERATED BY
  DEFAULT AS IDENTITY PRIMARY KEY,
  CUSTKEY INTEGER NOT NULL UNIQUE 3,
  CUSTOMER CHAR(25) NOT NULL ,
  ADDRESS CHAR(40) NOT NULL ,
  CITY CHAR(30) NOT NULL ,
  STATE CHAR(2) NOT NULL ,
  ZIPCODE NUMERIC(10, 0) NOT NULL ,
  PHONE CHAR(15) NOT NULL ,
  CM_LAST_CHANGED FOR COLUMN
  CMLASTCHG TIMESTAMP NOT NULL
  FOR EACH ROW ON UPDATE
  AS ROW CHANGE TIMESTAMP);
```

```
CRTLF CUSTMAST
A      R CUSTMASTR PFILE(CUST_MAST 1)
A      CUSTKEY      R
A      CUSTOMER     R
A      ADDRESS      R
A      CITY         R
A      STATE        R
A      ZIPCODE      R
A      PHONE        R
A      K CUSTKEY 3
```

### Notes

1. Original PF is now LF and references new SQL table CUST\_MAST
2. New SQL only columns are not part of surrogate file
3. CUSTKEY is now unique key constraint (if appropriate)
4. FIELDREF no longer used, R in REF column ignored for LFs

## Enhanced DDL TABLE and Surrogate DDS **PF**

```
CREATE TABLE CUST_MAST (
  CUST_MAST_ID FOR COLUMN CUSTMASTID
  BIGINT GENERATED BY DEFAULT AS
  IDENTITY PRIMARY KEY,
  CUSTKEY INTEGER NOT NULL UNIQUE ,
  CUSTOMER CHAR(25) NOT NULL ,
  ADDRESS CHAR(40) NOT NULL ,
  CITY CHAR(30) NOT NULL ,
  STATE CHAR(2) NOT NULL ,
  ZIPCODE NUMERIC(10, 0) NOT NULL ,
  PHONE CHAR(15) NOT NULL ,
  CM_LAST_CHANGED FOR COLUMN
  CMLASTCHG TIMESTAMP NOT NULL
  FOR EACH ROW ON UPDATE
  AS ROW CHANGE TIMESTAMP);
```

```
SELECT * FROM CUSTMAST1;
CREATE VIEW CUSTMAST2 AS SELECT *
  FROM CUST_MAST;

CRTPF FILE(CUSTMASTS3) MBR(*NONE)3
```

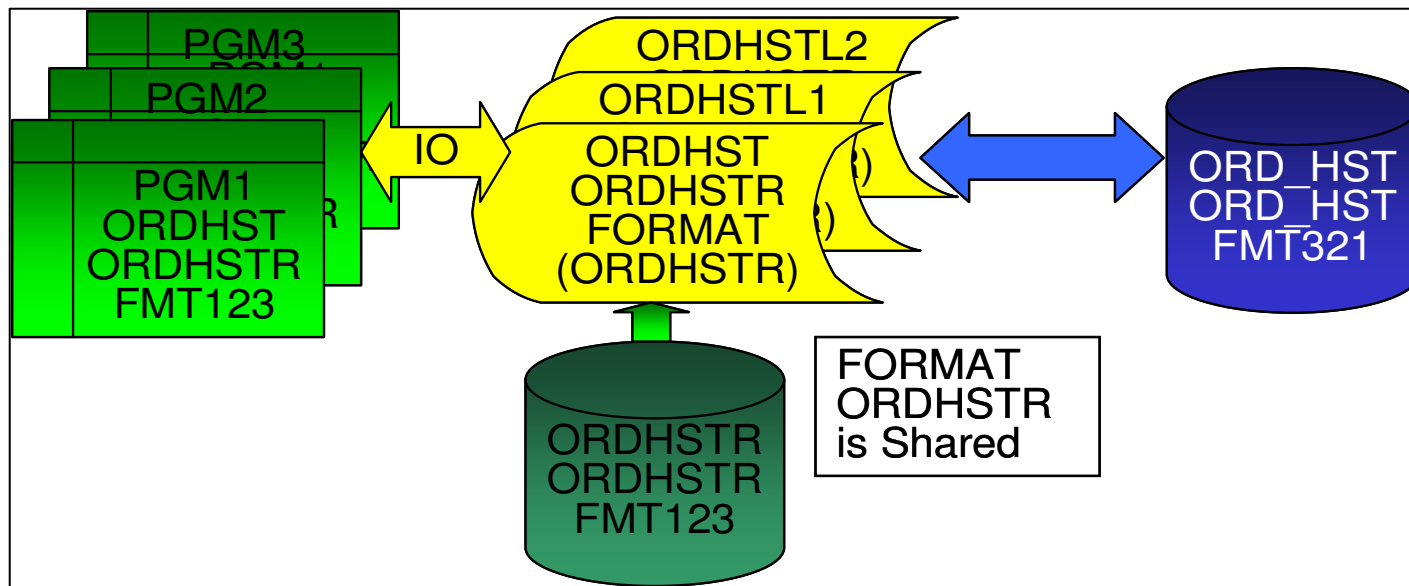
A		REF(FIELDREF)
A	R CUSTMASTR	
A	CUSTKEY	R
A	CUSTOMER	R
A	ADDRESS	R
A	CITY	R
A	STATE	R
A	ZIPCODE	R
A	PHONE	R

### Notes

1. Current PF is being referenced by SQL
2. Original PF becomes SQL View
3. PF created with no members-it is used for format references only

## Sharing the Format

- For each logical file which shared the physical file format (FMT123):
  - PFILE now points to SQL table (FMT321)
  - FORMAT keyword specifies surrogate (FMT123)
- Reasons a format will not be shared:
  - DDS Join Logical Files have unique format IDs
  - Existing DDS LF has unique format name



## Updated LFs Sharing Surrogate LF Format

CRTLF CUSTMAST <sup>2</sup>			CRTLF CUSTMASTL1		
A*		REF(FIELDREF)	A*	R CUSTMASTR	PFILE(CUSTMAST)
A	R CUSTMASTR <sup>2</sup>	PFILE(CUST_MAST <sup>1</sup> )	A	R CUSTMASTR <sup>2</sup>	PFILE(CUST_MAST <sup>1</sup> )
A	CUSTKEY	R	A		FORMAT(CUSTMAST <sup>2</sup> )
A	CUSTOMER	R	A	K CUSTOMER	
A	ADDRESS	R	CRTLF CUSTMASTL2		
A	CITY	R	A*	R CUSTMASTR	PFILE(CUSTMAST)
A	STATE	R	A	R CUSTMASTR <sup>2</sup>	PFILE(CUST_MAST <sup>1</sup> )
A	ZIPCODE	R	A		FORMAT(CUSTMAST <sup>2</sup> )
A	PHONE	R	A	K STATE	
A	K CUSTKEY		A	K CITY	
			A	K CUSTOMER	

### Notes

1. All DDS LFs are built over the new SQL table CUST\_MAST
2. The format CUSTMASTR is part of the surrogate LF CUSTMAST



## Updated LFs Sharing Surrogate PF Format

CRTPF FILE(**CUSTMASTSF**) MBR(\*NONE)

A REF(FIELDREF)

A R **CUSTMASTR<sup>2</sup>**

A CUSTKEY R

A CUSTOMER R

A ADDRESS R

A CITY R

A STATE R

A ZIPCODE R

A PHONE R

CRTL F CUSTMASTL1

A\* R CUSTMASTR PFILE(CUSTMAST)

A R **CUSTMASTR<sup>2</sup>** **PFILE(CUST\_MAST<sup>1</sup>)**

A **FORMAT(CUSTMASTSF<sup>2</sup>)**

A K CUSTOMER

CRTL F CUSTMASTL2

A\* R CUSTMASTR PFILE(CUSTMAST)

A R **CUSTMASTR<sup>2</sup>** **PFILE(CUST\_MAST<sup>1</sup>)**

A **FORMAT(CUSTMASTSF<sup>2</sup>)**

A K STATE

A K CITY

A K CUSTOMER

### Notes

1. All DDS LFs are built over the new SQL table **CUST\_MAST**
2. The format **CUSTMASTR** is part of the surrogate file **CUSTMASTSF**

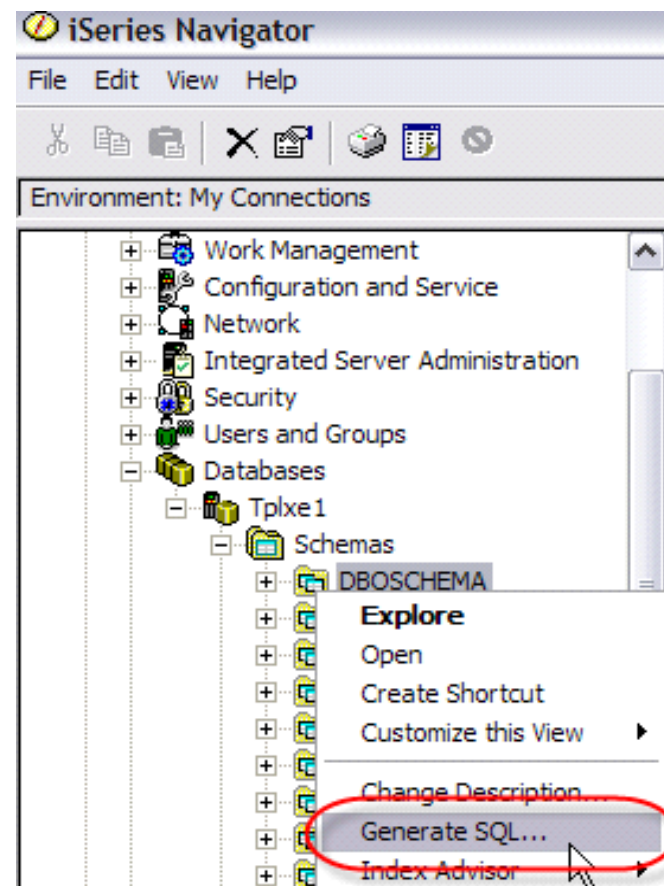
## Transparent SQL Migration - Considerations

- Creating a Surrogate LF can be challenging for certain Physical File (PF) types
  - Multi-member physical files
    - Require usage of partitioned tables
    - “Modernized” partitioned tables not advised until 7.1 when identity columns and RI supported for partitioned tables
    - Partitioned tables only support 256 partitions (members)
  - Usage of ALWNULL keyword may cause non-compatible record formats
  - Source file on CLRPFM or CPYF commands – most likely these are work files
- Not all files need to be converted to SQL DDL – especially work files!!!
- SQL Statements that reference PF can have different performance behavior
  - Surrogate LF conversion will cause SQL statements to reference an LF instead of PF
  - Prior to 7.1, all SQL statements referencing LF processed by Classic Query Engine (CQE)
    - IBM i 6.1 release now has Test PTF to enable simple LF references to be processed by SQL Query Engine (SQE)
  - NET: Upgrade to a newer release!!!
- DB2 for i SQL Modernization Workshop  
[ibm.com/systems/i/support/itc/educ/lsdb2mod.html](http://ibm.com/systems/i/support/itc/educ/lsdb2mod.html)

## Transparent SQL Migration - Tooling

### *DDS to SQL Conversion Tool*

- System i Navigator Generate SQL Task (QSQGNDDL API)
  - Useful in converting object definitions from DDS to SQL
  - Supports physical & logical files
    - Not all DDS features can be converted, tool will convert as much as possible and generate warnings for unconvertible options (e.g., EDTCDE)
    - Logical files converted to SQL Views
    - SQL Field Reference File support not used
  - Can convert a single object or a group of objects
  - Output can be edited & saved directly into source file members



- **XCase for System i** tooling that automates and manages this migration process ([www.xcaseforsystemi.com](http://www.xcaseforsystemi.com))
  - Free Diagnostic Modernization download
  - Data modeling tool also available

## Reengineering Surrogate Considerations

### Non-SQL access

- Referencing PF directly can hinder leveraging new DDS features
  - Future changes may result in unnecessary recompiles

### SQL access

- **Pre 7.1 only**
  - Avoid referencing surrogate logical file
    - Prevents SQE access, requires CQE
    - Performance and functionality can suffer
  - rewrite to reference new physical table
- The SQE optimizer can use DDS LFs as of **7.1**

# Beyond DDS conversion

## Moving to SQL

Moving to SQL can be proceed in any of several ways

- Change files from native DDS to SQL DDL
- Rewrite existing applications to use SQL
  - Leverage the power of SQL
  - Open up new ways to access data
- Develop new applications based on SQL

## Setting the stage

- The art and science of writing and deploying SQL is different than traditional record level access
  - Writing SQL like native access is the wrong approach!
- Proper training, support and governance is required
- A suitable SQL migration and use strategy must be defined and implemented
- Modernization requires modern tools and methods



## Setting the stage....

### Examples. SQL:

- Is answer set oriented, not record oriented
  - Worry about the answer set you want, not the steps to get it
- Is meant to join tables
  - Don't chain them in the application program!
- Can help hide complexity
  - Views can simplify definitions for application developers
- Lets application developers focus on the application
  - And database developer focus on the database
- Has its own very rich programming language
  - With procedures, functions and triggers

.  
. .  
. .

SQL is a language

Like any language,  
you have to learn  
the syntax, proper usage, and coding best practices.



You must learn the science, and develop the art.

## SQL DML example, join

```
SELECT t.year,t.month,i.orderdt,c.country,c.cust  
       p.part,s.supplier,i.quantity,i.revenue  
FROM item_fact i  
     INNER JOIN part_dim p ON (i.partid = p.partid)  
     INNER JOIN time_dim t ON (i.orderdt = t.datekey)  
     INNER JOIN cust_dim c ON (i.custid = c.custid)  
     INNER JOIN supp_dim s ON (i.suppid = s.suppid)  
WHERE t.year > 2005
```

## SQL DDL/DML example, hiding complexity with a view

```
CREATE VIEW JoinView(orderyear, ordermonth, orderdate, country,  
    last_name, part_name, supplier_name, quantity, revenue)  
AS SELECT t.year,t.month,i.orderdt,c.country,c.cust  
    p.part,s.supplier,i.quantity,i.revenue  
FROM item_fact i  
    INNER JOIN part_dim p ON (i.partid = p.partid)  
    INNER JOIN time_dim t ON (i.orderdt = t.datekey)  
    INNER JOIN cust_dim c ON (i.custid = c.custid)  
    INNER JOIN supp_dim s ON (i.suppid = s.suppid)  
WHERE  
SELECT * FROM JoinView  
    WHERE orderyear = 2009 AND ordermonth IN (10,11,12)
```



Simplification!

## SQL Example, SQL procedure

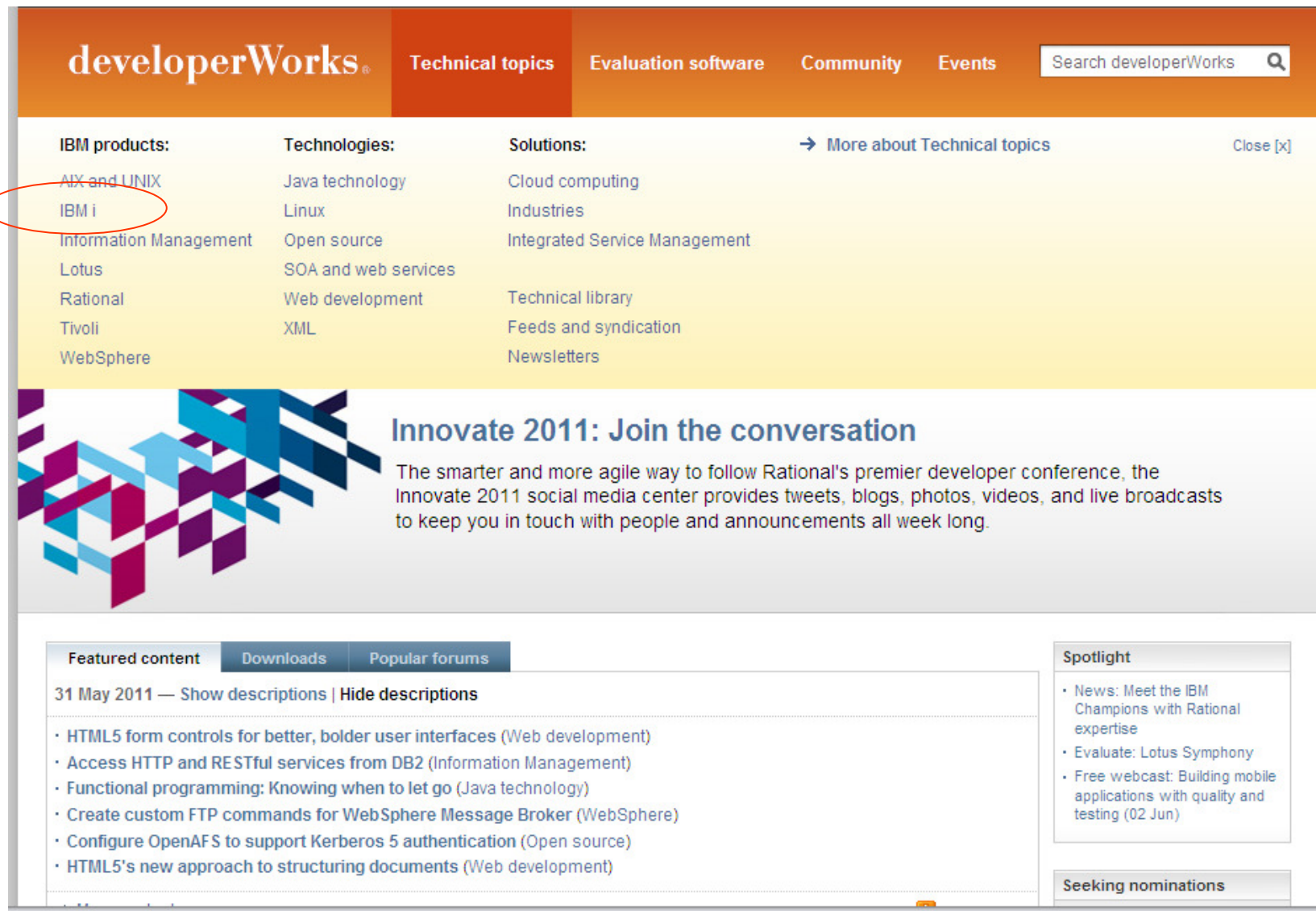
```
Create procedure justice_for_all(out o_number_of_raises int, out o_cost_of_raises decimal(9,2))
language sql
proc_body:
begin
  declare v_avg_tenure int;
  declare v_avg_compensation decimal(9,2);
  declare v_number_of_raises int;
  declare v_cost_of_raises decimal(9,2);
  set v_avg_tenure = 0;
  set v_avg_compensation = 0.0;
  select avg(year(current_timestamp) - year(hiredate)), decimal(avg(salary + bonus + comm), 9, 2)
    into v_avg_tenure, v_avg_compensation
    from employee;

  set v_number_of_raises = 0;
  set v_cost_of_raises = 0.0;
  for_loop:
    FOR each_row AS c1 CURSOR FOR
      SELECT year(current_timestamp) - year(hiredate) as tenure,
             salary+ bonus + comm as compensation
      FROM employee
      DO
        IF tenure > v_avg_tenure and compensation < v_avg_compensation
        THEN
          UPDATE employee SET salary = salary + (v_avg_compensation - compensation)
          WHERE CURRENT OF c1;
          SET v_number_of_raises = v_number_of_raises + 1;
          SET v_cost_of_raises = v_cost_of_raises + (v_avg_compensation - compensation);
        END IF;
      END FOR;
  SET o_number_of_raises = v_number_of_raises;
  SET o_cost_of_raises =v_cost_of_raises;
END proc_body;
```

## Practical advise

- Migrating legacy databases requires planning plus additional knowledge and skills
  - Understanding query optimization
  - Relational database design knowledge
  - Acquire and use good tooling
  
- You should have good business reasons for migrating
  - New or changing requirements
  - Need for enhanced features and functions
  - New applications accessing legacy data
  
- Start small, get some experience
  - Identify a pilot application which would benefit from modernization
  - Get educated on SQL and DB2 for i
  
- Performance improvement is a benefit, not the main reason for reengineering

## IBM i is now on developerWorks!!



The screenshot shows the IBM developerWorks website. The top navigation bar includes 'developerWorks', 'Technical topics', 'Evaluation software', 'Community', and 'Events', along with a search bar. Below this, a yellow banner displays three columns of links: 'IBM products:', 'Technologies:', and 'Solutions:'. The 'IBM products:' column lists 'AIX and UNIX', 'IBM i' (circled in red), 'Information Management', 'Lotus', 'Rational', 'Tivoli', and 'WebSphere'. The 'Technologies:' column lists 'Java technology', 'Linux', 'Open source', 'SOA and web services', 'Web development', and 'XML'. The 'Solutions:' column lists 'Cloud computing', 'Industries', 'Integrated Service Management', 'Technical library', 'Feeds and syndication', and 'Newsletters'. To the right of these columns are links for 'More about Technical topics' and a 'Close [x]' button. Below the banner is a section for 'Innovate 2011: Join the conversation', featuring a graphic of colorful geometric shapes and text about the conference. At the bottom, there are tabs for 'Featured content', 'Downloads', and 'Popular forums'. The 'Featured content' tab is active, showing a list of articles dated '31 May 2011'. To the right of the featured content is a 'Spotlight' section with links to news, evaluation, and a free webcast. At the very bottom right is a 'Seeking nominations' button.

**developerWorks®** Technical topics Evaluation software Community Events Search developerWorks

**IBM products:** **Technologies:** **Solutions:** → More about Technical topics Close [x]

AIX and UNIX  
**IBM i**  
Information Management  
Lotus  
Rational  
Tivoli  
WebSphere

Java technology  
Linux  
Open source  
SOA and web services  
Web development  
XML

Cloud computing  
Industries  
Integrated Service Management  
Technical library  
Feeds and syndication  
Newsletters

**Innovate 2011: Join the conversation**  
The smarter and more agile way to follow Rational's premier developer conference, the Innovate 2011 social media center provides tweets, blogs, photos, videos, and live broadcasts to keep you in touch with people and announcements all week long.

**Featured content** Downloads Popular forums  
31 May 2011 — Show descriptions | Hide descriptions

- HTML5 form controls for better, bolder user interfaces (Web development)
- Access HTTP and RESTful services from DB2 (Information Management)
- Functional programming: Knowing when to let go (Java technology)
- Create custom FTP commands for WebSphere Message Broker (WebSphere)
- Configure OpenAFS to support Kerberos 5 authentication (Open source)
- HTML5's new approach to structuring documents (Web development)

**Spotlight**

- News: Meet the IBM Champions with Rational expertise
- Evaluate: Lotus Symphony
- Free webcast: Building mobile applications with quality and testing (02 Jun)

**Seeking nominations**

## Introducing:

# developerWorks for IBM i!

## IBM i technology information wiki

- Launched as part of the dW “i zone” in April, 2011  
Wiki URL: [www.ibm.com/developerworks/ibmi/techupdates](http://www.ibm.com/developerworks/ibmi/techupdates)  
DB2 URL: [www.ibm.com/developerworks/ibmi/techupdates/db2](http://www.ibm.com/developerworks/ibmi/techupdates/db2)

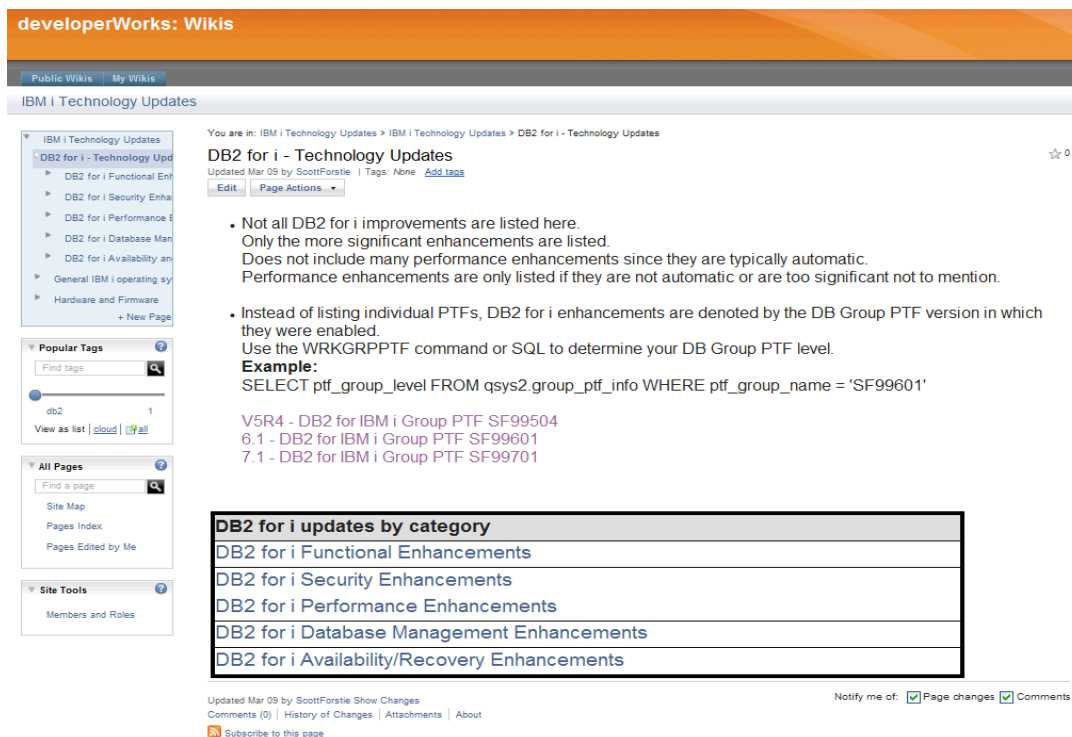
- Centralized location for **communicating DB2 for i enhancements** made in between major IBM i releases

- Organized by subject matter and category of enhancement

- Notify me of Page Changes* option

- comment or recommend specific pages

- Info Center remains the site for technical documentation, however, sometimes the information will appear on dW first



**developerWorks: Wikis**

Public Wikis | My Wikis

IBM i Technology Updates

You are in: IBM i Technology Updates > IBM i Technology Updates > DB2 for i - Technology Updates

### DB2 for i - Technology Updates

Updated Mar 09 by ScottForstie | Tags: None | [Add tags](#)

[Edit](#) | [Page Actions](#)

- Not all DB2 for i improvements are listed here. Only the more significant enhancements are listed. Does not include many performance enhancements since they are typically automatic. Performance enhancements are only listed if they are not automatic or are too significant not to mention.
- Instead of listing individual PTFs, DB2 for i enhancements are denoted by the DB Group PTF version in which they were enabled. Use the WRKGRPPTF command or SQL to determine your DB Group PTF level.  
**Example:**  
`SELECT ptf_group_level FROM qsys2.group_ptf_info WHERE ptf_group_name = 'SF99601'`

V5R4 - DB2 for IBM i Group PTF SF99504  
6.1 - DB2 for IBM i Group PTF SF99601  
7.1 - DB2 for IBM i Group PTF SF99701

DB2 for i updates by category
<a href="#">DB2 for i Functional Enhancements</a>
<a href="#">DB2 for i Security Enhancements</a>
<a href="#">DB2 for i Performance Enhancements</a>
<a href="#">DB2 for i Database Management Enhancements</a>
<a href="#">DB2 for i Availability/Recovery Enhancements</a>

Updated Mar 09 by ScottForstie | [Show Changes](#)  
Comments (0) | [History of Changes](#) | [Attachments](#) | [About](#)  
[Subscribe to this page](#)

Notify me of: ☒ Page changes ☒ Comments

## On the Web

### developerWorksDB2 URL:

- [www.ibm.com/developerworks/ibmi/techupdates/db2](http://www.ibm.com/developerworks/ibmi/techupdates/db2)

### DB2 for i Home Page

- <http://www.ibm.com/systems/i/software/db2/>

### System i Advantages

- <http://www-1.ibm.com/systems/i/advantages/>

### System i Access

- <http://www.ibm.com/systems/i/access/>

### DB2 for i Java

- <http://www.ibm.com/systems/i/software/db2/javadb2.html>

### Education and Publications

- <http://www.ibm.com/systems/i/> - Click on Education
- <http://publib.boulder.ibm.com/series/>

### Newsgroups and Forums

- [comp.databases.ibm-db2](http://comp.databases.ibm-db2)
- [comp.sys.ibm.as400.misc.groups](http://comp.sys.ibm.as400.misc.groups)

### Questions can be sent to:

- [rchudb@us.ibm.com](mailto:rchudb@us.ibm.com)





**Thank You!**

**è Need help using the latest DB2 for i technologies?**

**è Are you getting the most out DB2 for i?**



### IBM DB2 for i Consulting and Services

- ✓ Database modernization
- ✓ DB2 WebQuery
- ✓ Database design, features and functions
- ✓ DB2 SQL performance analysis and tuning
- ✓ Data warehousing and Business Intelligence
- ✓ DB2 for i education and training

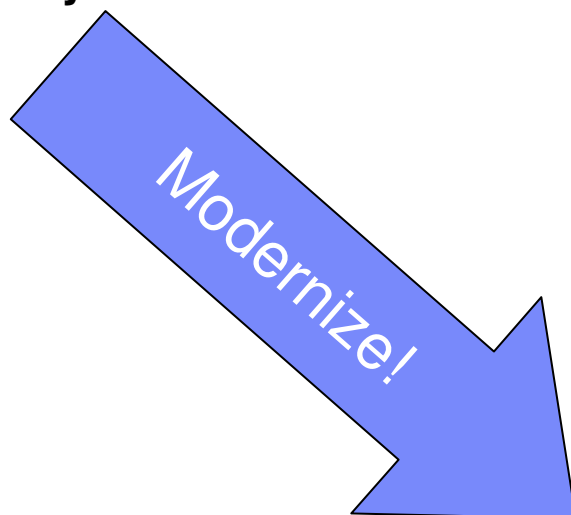
Contact: Mike Cain [mcain@us.ibm.com](mailto:mcain@us.ibm.com)  
IBM Systems and Technology Group  
Rochester, MN USA

## An Example of Why Modernize

## Store and Manage Documents in DB2

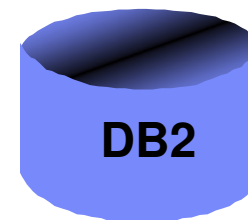
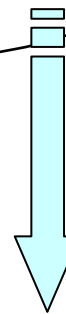
### Integrated XML support (7.1)

- New XML column type
- Store document in DB2
- Manage and manipulate XML
- Search and reference XML documents
- **SQL only**



### XML document

```
<booking unitCharge="50" units="2"  
        currency="USD"  
        status="confirmed">  
  <item>  
    <room hotelName="White Palace"  
          type="suite"  
          bookedFrom="2011-05-25"  
          bookedTo="2011-05-29" />  
  </item>  
</booking>
```



Row ID	XML	Timestamp
--------	-----	-----------

## IBM OmniFind Text Search Server for DB2 for i

- New IBM i product offering: 5733-OMF
  - *No-charge* offering
  - Requires IBM i 6.1 or greater
  
- Delivers common DB2 Family text search technology
  - Advanced, linguistic high-speed searches
  - Support enabled for any character-based column
  - Search technology also supports Rich Text document formats
    - Example: LOB columns containing XML, PDF or Microsoft® Word documents
    - IFS documents can be indexed with extra programming
  - Includes support for 26 different languages
  - **SQL only**
    - CONTAINS and SCORE functions

# Fuzzy Search