Application Modernization – DB2 for i Style

Kent Milligan & Mike Cain
IBM STG Lab Services – DB2 Center of Excellence
ISV Enablement – IBM i & DB2 for i

Agenda

- Why?
- Approaches & Options
- Modernizing Database Definitions
- Modernizing Data Access
- Next Steps

Why SQL?

- Strategic database interface for industry
- Portability of code & skills
- Strategic interface for IBM i
- Faster delivery on IT requirements
- Performance & Scalability
- Increased Data Integrity
- Image

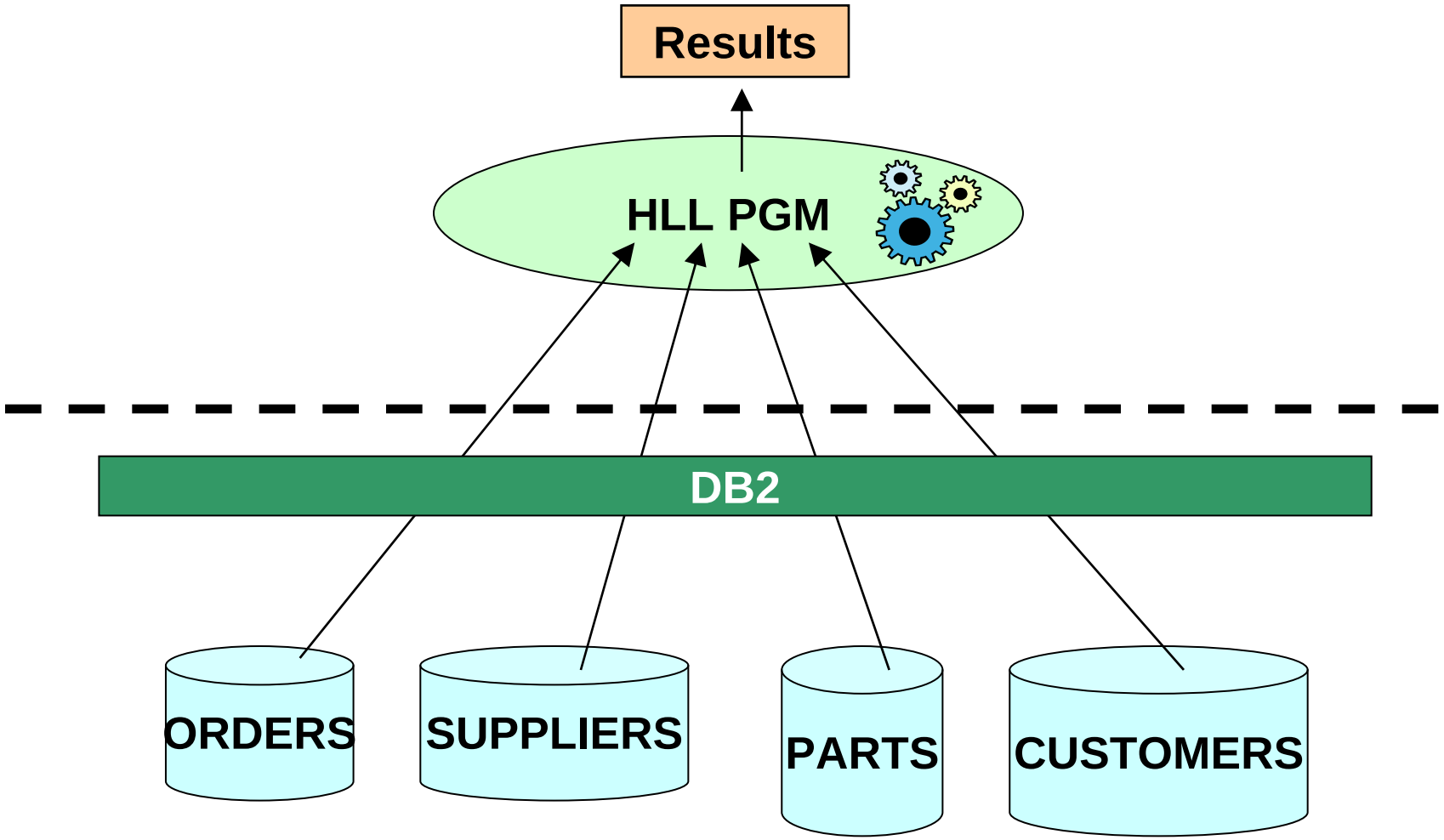
Want More Details…

NEW White Paper on Benefits of Modernizing with SQL

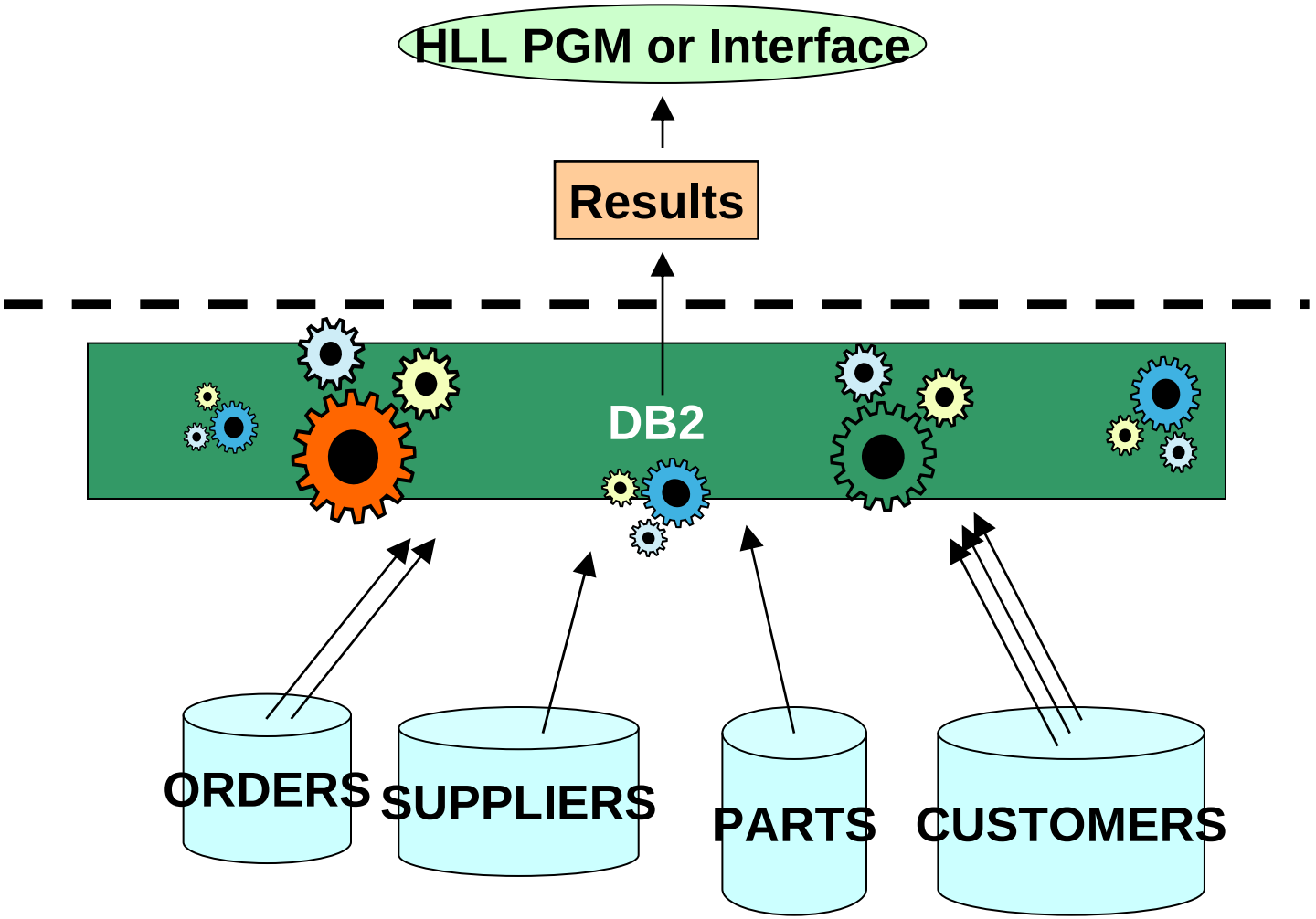
ibm.com/partnerworld/page/whitepaper/ibmi/db2/sql

Enhancements to non-SQL interfaces	SQL enhancements
Unicode support - UTF-16 and UTF-8	Unicode support - UTF-16 and UTF-8
Binary character data type	Binary character data type
CRTLF PAGESIZE parameter	CREATE INDEX PAGESIZE keyword
Larger decimal support	Larger decimal support
SSD enablement for physical & logical files	SSD enablement for tables and Indexes
	XML, National Character, and ROWID data types
	Identity column attribute
	Hidden and Automatic Timestamp column attributes
	Field Procedure column attribute
	Sequence object
	Column-level and Instead-Of triggers
	Merge statement
	OLAP and Super Group expressions
	Create Table from Select and Insert from Select
	SQL functional indexes
	XML Publishing and Decomposition functions
	IBM OmniFind Text Search Server
	SQL Query Engine (SQE)
	SQE Result Set Caching
	SQE Autonomic Indexes
	SQE Self-Learning Query Optimization & Adaptive Query Processing
	SQE Encoded Vector Index fast path for aggregate processing
	SQE In-memory Database Enablement
	IBM i Navigator Plan Cache Tool

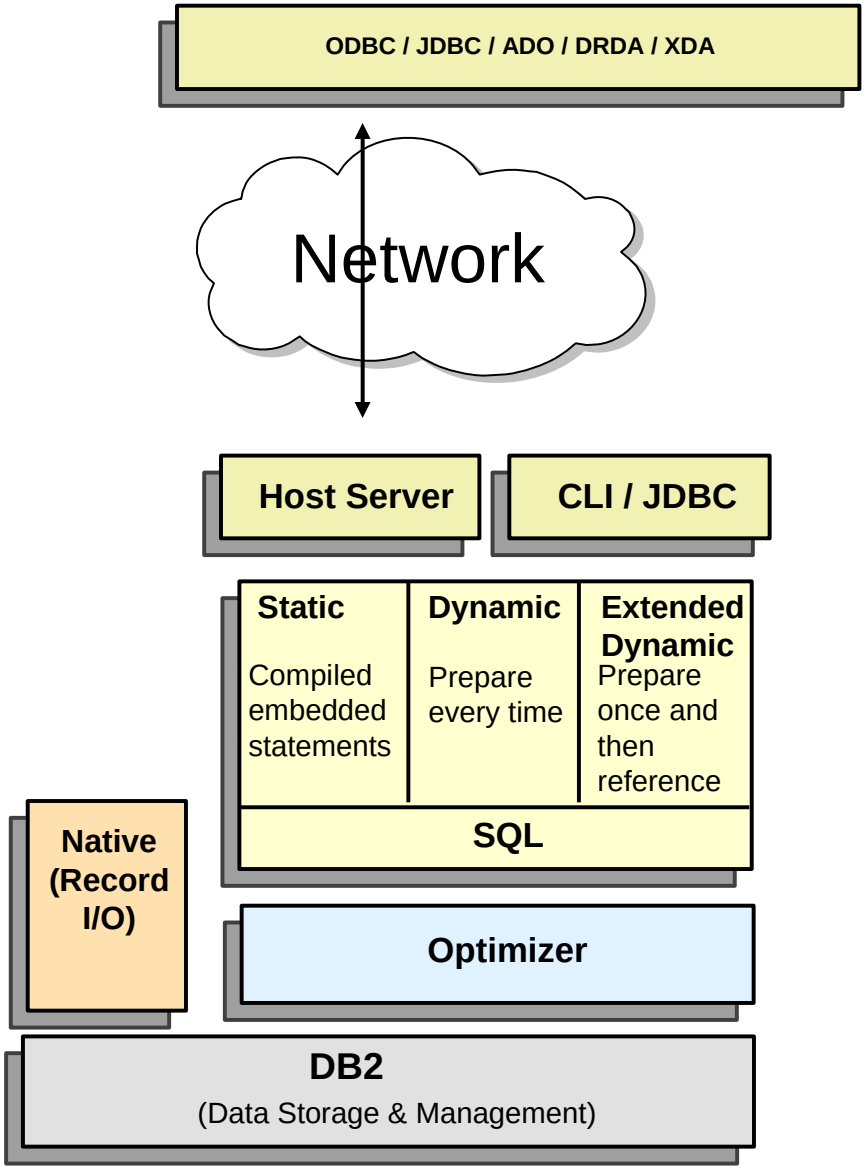
Traditional Record-Level Access



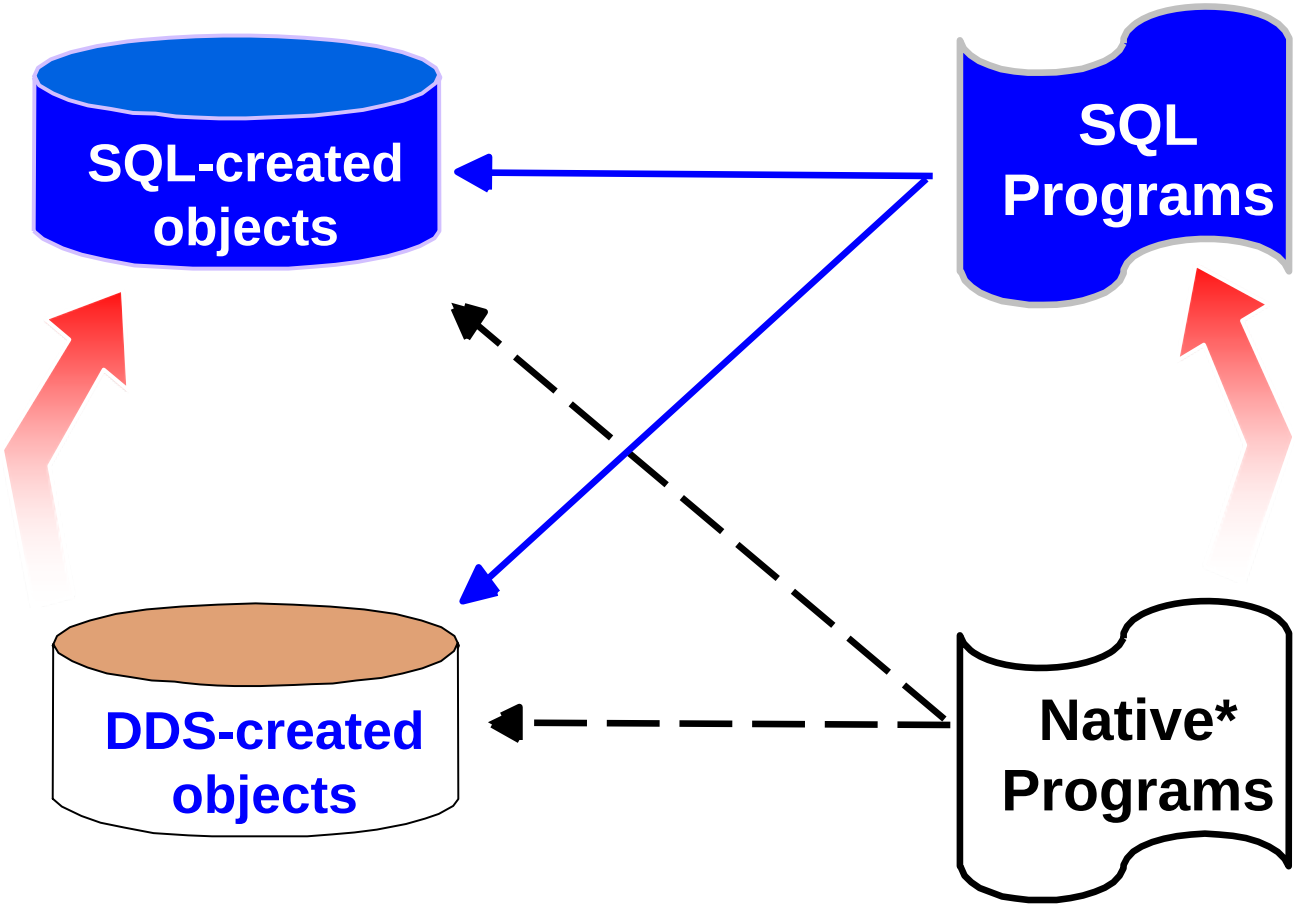
SQL Data-Centric Programming



Approaches & Options



Approaches & Options



***Considerations*:**
Multi-member & multi-format files

****Restrictions:***
EVIs, LOB columns, XML, UDTs, etc.

Modernizing Definitions & Objects

- Modeling
- Terminology
- Moving from DDS to SQL DDL
- SQL object management
- Embedding business logic into database definitions

Modernizing Definitions & Objects

Data modeling

- Database normalization
  - –Define a separate table for each related set of values
  - –Define the primary key (surrogate or natural)
  - –Eliminate redundant data
  - –Design for Fifth normal form (5NF), performance & storage may drop back to 3NF
  - –Establish RI constraints

- Consider Master Data Management
  - –Services created to retrieve data – what if multiple copies exist?

Enrollment ID	Student ID	Student Name	Course ID	Course Name
200400	3456	Ed Moore	M-1932	Discrete Mathematics
200401	9393	Chuck Sellers	M-3459	Differential Equations
200402	2303	Lisa Gibbs	A-1003	Intro to Anthropology
200403	3456	Ed Moore	A-1003	Intro to Anthropology

Normalization

Student ID (PK)	Student Name
3456	Ed Moore
9393	Chuck Sellers
2303	Lisa Gibbs

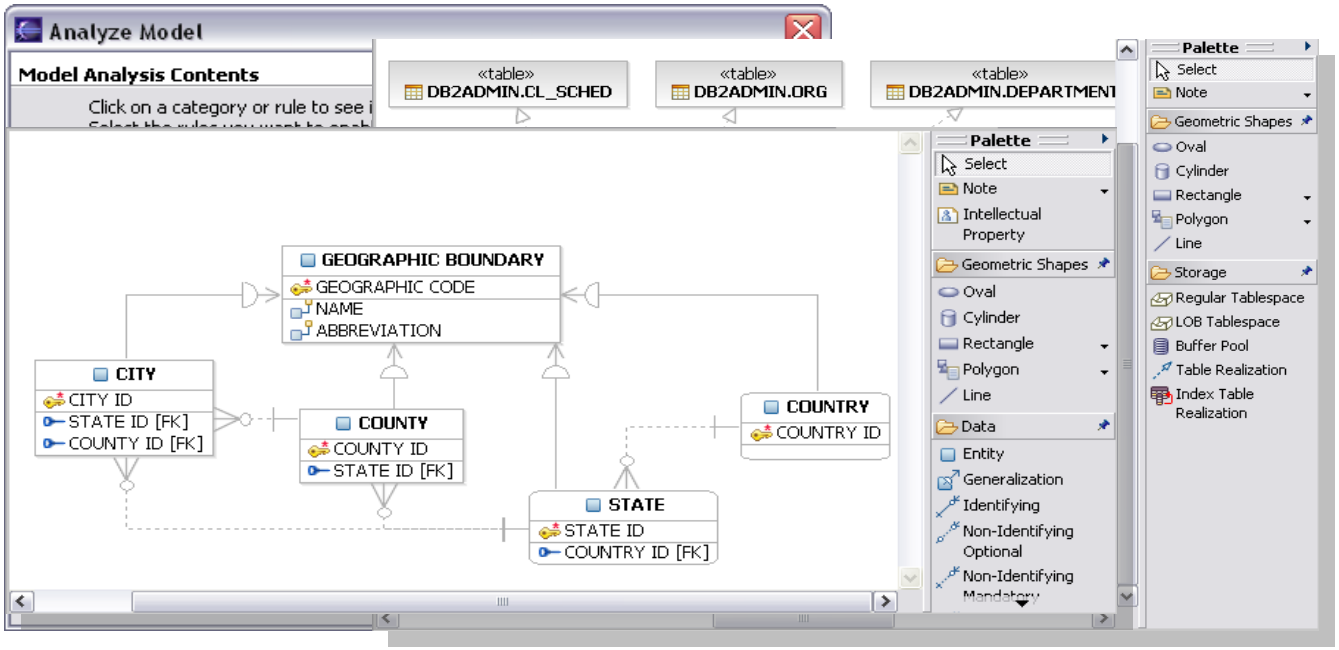
Course ID (PK)	Course Name
M-1932	Discrete Mathematics
M-3459	Differential Equations
A-1003	Intro to Anthropology

Enrollment ID (PK)	Student ID (FK)	Course ID (FK)
200400	3456	M-1932
200401	9393	M-3459
200402	2303	A-1003
200403	3456	A-1003

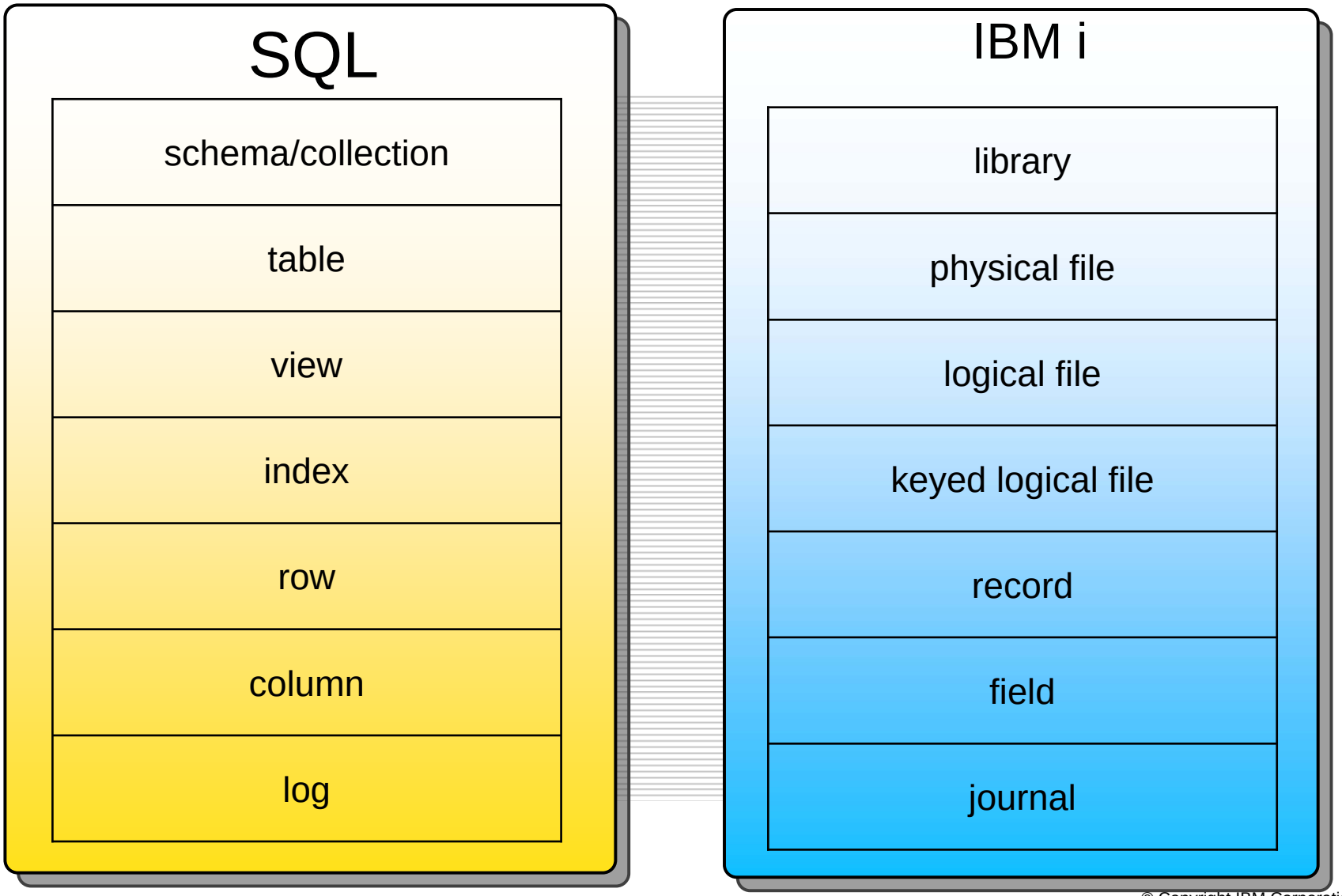
Modernizing Definitions & Objects

Data Modeling - IBM InfoSphere Data Architect

- Enterprise data modeling and management
  - Compare & synchronize
  - Forward & reverse engineering
  - Logical file support – Fixpack 003
  - Model analyzer for enterprise standard conformance
- Database development – SQL Stored Procedures and Function
- Trial Download: http://ibm.com/software/data/optim/data-architect/



Modernizing Database Objects
Terminology



Modernizing Objects: CREATE TABLE vs CRTPF

```
CREATE TABLE EMP_MAST (
  EMP_MAST_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) ,
  EM_ROW_CHANGE_TS TIMESTAMP NOT NULL
     FOR EACH ROW ON UPDATE
        AS ROW CHANGE TIMESTAMP
     IMPLICITLY HIDDEN)
```

```
CRTPF FILE(EMPLOYEE) SRCFILE(QDDSSRC)
        SRCMBR(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)
```

- Wider selection of data types & column attributes
- Longer, more descriptive identifiers
- Data modeling tool support
- Self-contained source statement, can include constraint definitions

- Limited set of data types & attributes
- Format sharing & field attributes (CHECK, RANGE, DATFMT)
- Keyed support, but only 1 key per definition.
- Constraints must be defined separately

Modernizing Objects: CREATE INDEX vs CRTLF (Keyed)

```
CREATE INDEX EMP_LASTNAME_DEPT
  ON EMP_MAST(WORKDEPT, LASTNAME)
    RCDFMT EMPLOYEER1
    ADD COLUMNS
          EMPNO, FIRSTNME, MIDINIT

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

```
CRTLF FILE(EMPLOYEEL1) SRCFILE(QDDSSRC)
      SRCMBR(EMPLOYEEL1)
--Source Data
A         R EMPLOYEER1    PFILE(EMPLOYEE)
A           WORKDEPT
A           LASTNAME
A           EMPNO
A           FIRSTNME
A           MIDINIT
A         K WORKDEPT
A         K LASTNAME
```

- Encoded Vector Index (EVI) structure
- Expressions can be used in the definition of the key columns
- Sparse Indexes with WHERE clause (ie, Select/Omit)
- EVI “Instant” Aggregate support
- Larger default logical page size

- Only Binary Radix Tree structure support – no EVIs
- Limited support for key derivations and expressions
- Key attributes –FCFO, FIFO, LIFO,
- Smaller default logical page size

Modernizing Objects: 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       BONUS
```

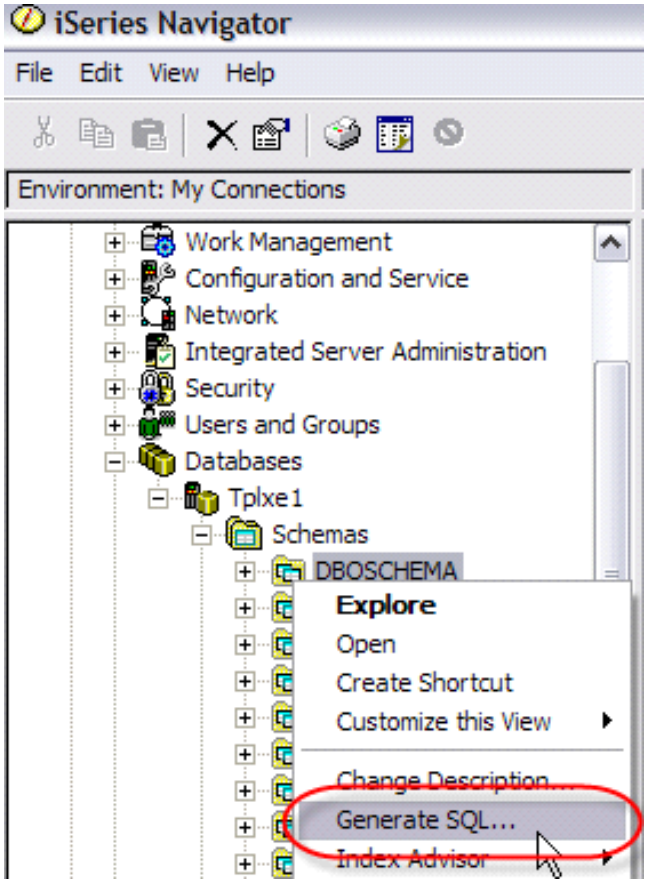
- Full access to advanced query capabilities of SQL
- Can be used as logical files to enhance native functionality
- No support for keying/ordering

- Limited Join support
- No support for Grouping, Case, Subqueries, User-Defined functions, …
- Multiple members & formats

Modernizing Database Definitions & Objects

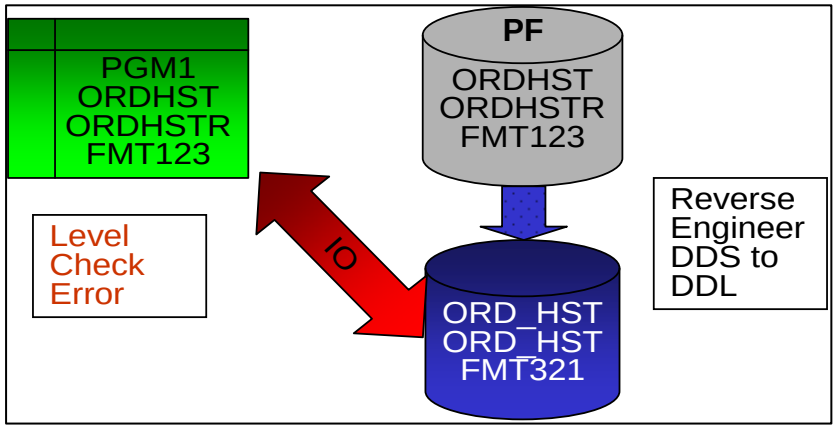
DDS to SQL Conversion Tool

- 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

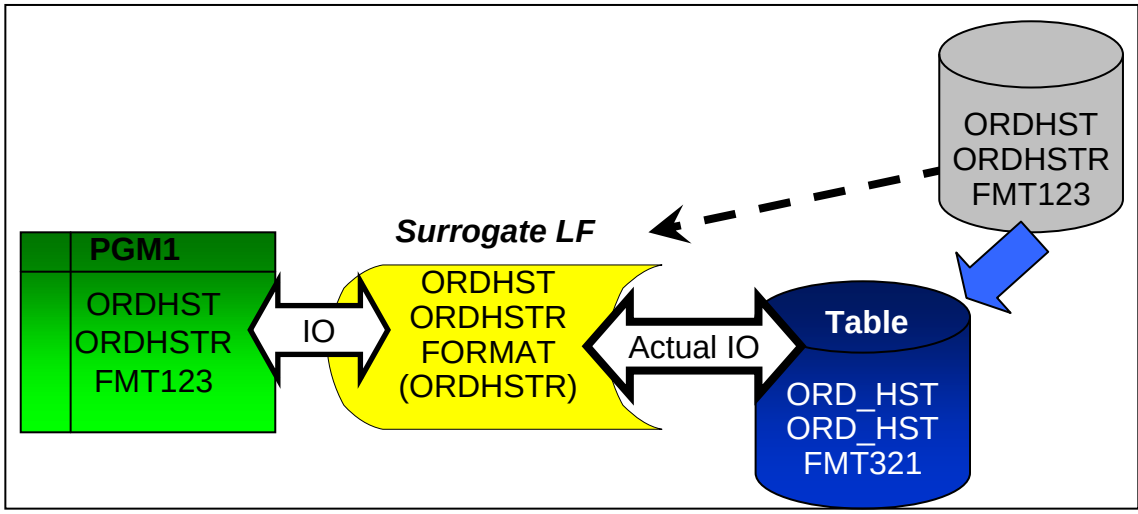


Modernizing Database Definitions - Transparently

- Converting DDS PF to SQL DDL Table results in format identifiers being changed
  - HLL programs accessing the SQL Table will receive a “level check” exception message.
  - Only solutions prior to V5R4
    - •recompile the program or
    - •ignore the exception
    - •(not recommended)



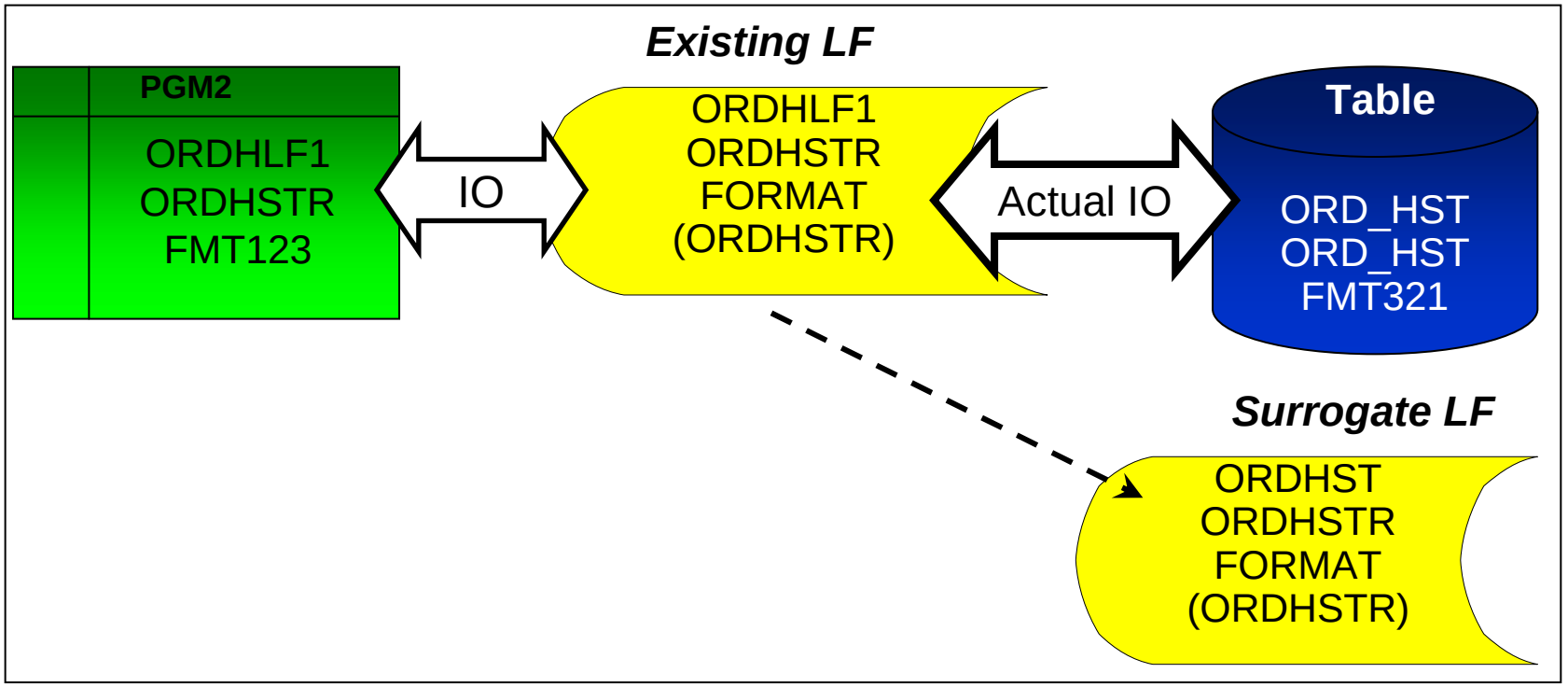
- A surrogate file preserves the original DDS PF format
  - Allows new columns to be added to SQL DDL Table
  - FORMAT keyword used to share surrogate format
    - •Prevents level check IDs for programs accessing original PF or LFs sharing format



- “Best” method for avoiding format id changes!

Modernizing Database Definitions – Transparently

- Logical files also need to re-engineered to reference the SQL table
  - For each logical file which shared the physical file format (FMT123):
    - •PFILE modified to point at SQL table (FMT321)
    - •FORMAT keyword specifies surrogate LF (FMT123)
  - Some LFs don't require re-engineering
    - •DDS LF with unique format name
    - •DDS Join Logical Files have unique format IDs



Modernizing Database Definitions - Transparently

1. Convert PF to SQL Table (with new name)

2. Create SQL indexes to replace any implicitly created keyed access paths that exist for DDS files (use “Show Indexes”)

3. Create “Surrogate” LF with same name as original PF name

4. Modify existing LFs to reference SQL table

Transparent SQL Migration - Example

- **Existing PF – INVENTORY**

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

- **Existing LF - INVLF**

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

- **Converted SQL Table:**

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

- **Surrogate LF – INVENTORY**

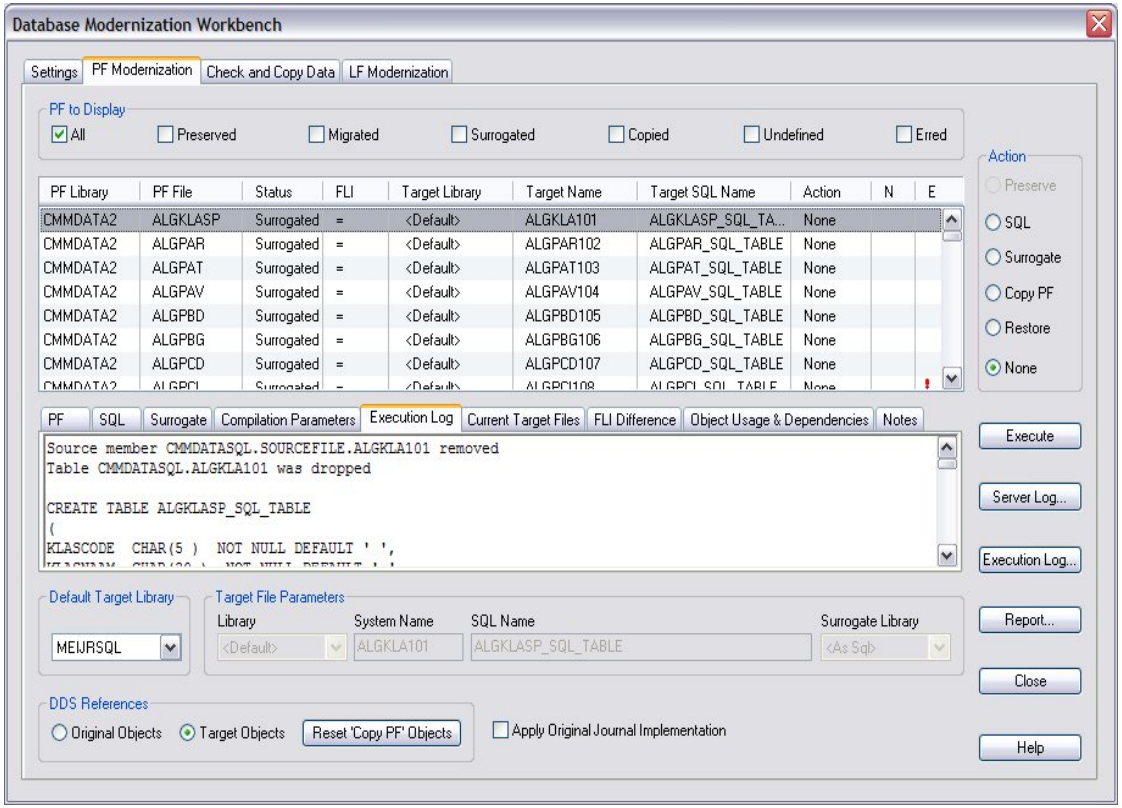
```
A R INVFMTR PFILE(SQL_INVENT)
A   ITEM     15A
A   ORDER    10A
A   SUPPLY   15A
A   QTY       5P
A   QTYDUE    5P
```

- **Existing LF - INVLF**

```
A R INVFMTR PFILE(SQL_INVENT)
            FORMAT(INVENTORY)
A K ITEM
A K ORDER
```

Transparent SQL Migration - Tooling

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



Modernizing Database Definitions & Objects

SQL Object management

- SQL Source Management best practices:
  - Just like DDS SQL source can be stored in source physical file members just and referenced with the RUNSQLSTM CL command instead of CRTPF/CRTLF
    - If change management tools are not IBM i specific, store SQL scripts in IFS or PC
    - If SQL source misplaced, Generate SQL can be used to retrieve the SQL source from System Catalogs (SYSIBM & QSYS2)
    - Navigator Run SQL Scripts in 6.1 can store and retrieve SQL from source members
  - SQL Table definitions can use Field Reference File

```
CREATE TABLE customer AS
  (SELECT id cust_id, lname cust_lastname, fname cust_firstname,
     city cust_city FROM RefFile)
  WITH NO DATA
```

- May need to adjust process for moving from development to production
  - Best practice is to re-execute SQL creation script
  - Save/Restore process for SQL databases documented at: ibm.com/developerworks/db2/library/techarticle/0305milligan/0305milligan.html

Modernizing Database Definitions & Objects

SQL Object Management

- 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

Modernizing Database Definitions & Objects

SQL Object Management

- 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
```

Modernizing Database Definitions & Objects

SQL Object Management

- RPG requires system name and record format name to be different
  - SQL defaults record format name to the system name
  - RCDFMT keyword can be used to override default behavior

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

Modernizing Definitions & Objects

SQL & Non-relational data

- User-Defined Table Functions
  - – Allows non-relational & legacy data to be virtualized as an SQL table

```
SELECT * FROM TABLE(myudtf('Part XYZ'))
```

  - – Both SQL & External Table Functions supported
    - External UDTFs can be easily written to access multi-format files, S/36 files, and stream files
    - Table functions need to be invoked from SQL-based interfaces or SQL view
    - External UDTF Examples: http://ibm.com/systems/i/db2/db2code.html

- LOBs
  - – Allows you to keep non-relational data along with all the other business data

Modernizing Definitions & Objects

Moving Business Logic into DB2 - Automatic Key Generation

- 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')**
- Sequence Object
  - Separate object that can be shared across multiple tables
  - Generated value to be part of non-numeric keys

    CREATE SEQUENCE order_seq START WITH 10 INCREMENT BY 10

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

Modernizing Definitions & Objects

Moving Business Logic into DB2 - Triggers

- Triggers allow you initiate business policies & processes whenever new data comes in or existing data is changed
  - DB2 responsible for always invoking the trigger program
  - Execution is independent of the user interface
  - Can be used to transform data before it gets into DB2
- DB2 for i Trigger Support
  - Before & After: Insert, Update, & Delete events (up to 300 triggers)
  - SQL & External(ADDPFTRG) Triggers
    - Column-level, Statement-level, and Instead Of triggers only available with SQL Triggers

```
CREATE TRIGGER audit_salary
  AFTER UPDATE ON employee(salary)
  REFERENCING NEW AS n
  REFERENCING OLD AS o
  FOR EACH ROW
  WHEN (n.salary - o.salary >= 5000)
      INSERT INTO audit
           VALUES(n.empno, n.deptno, n.salary,current timestamp)
```

Modernizing Data Access

- Programming Interfaces
- Native I/O to SQL Comparison

Modernizing Data Access – Programming Interfaces

Static SQL	Dynamic SQL	Extended Dynamic SQL
Embedded Static	Embedded Dynamic	QSQPRCED
SQL Procedures, Functions, Triggers	SQL Procedures, Functions, Triggers	Toolbox JDBC driver
	JDBC, SQLJ	IBM i Access ODBC & OLE DB
	ADO.NET, OLE DB	
	CLI, ODBC	
	PHP ibm_db2	
	RUNSQLSTM	

*****DB2 SQL Development Kit only required if embedded SQL (& STRSQL) is going to be used***

Modernizing Data Access

Native I/O to SQL Example

```
...
C/EXEC SQL
C+ DECLAREsql_jn CURSOR FOR SELECT
C+   t.year,t.month,i.orderdt,c.country,c.cust
C+   p.part,s.supplier,i.quantity,i.revenue
C+ FROM item_fact i
C+  INNER JOIN part_dim p  ON (i.partid =p.partid)
C+  INNER JOIN time_dim t ON (i.orderdt=t.datekey)
C+  INNER JOIN cust_dim c ON (i.custid=c.custid)
C+  INNER JOIN supp_dim s ON (i.suppid=s.suppid)
C+ WHERE year=2008 AND month=6
C/END-EXEC

C/EXEC SQL
C+ OPEN sql_jn
C/END-EXEC

C/EXEC SQL
C+ FETCH NEXT FROM sql_jn FOR :RowsReq ROWS
C+    INTO :result_set
C/END-EXEC
C               If      SQLCOD = 0   and
C                                    SQLER5 = 100 and
C                               SQLER3 > 0
C               Eval            RowsRd = SQLER3
...
```

```
C   SearchKey  KList
C                       Kfld            SearchYear
C                  Kfld                 SearchMonth
...
C   Times      Occur    Result_Set
C   SearchKey           Setll       TIME_DIML1
C                       If          %FOUND
C                  DOU      RowsReq = Rows Rd
C                  READ     TIME_DIML1
C                  If       %EOF
C                           Leave
C                  Endif
C   DATEKEY                 Setll       ITEMFACTL1
C                  If       %FOUND
C                           DOU      RowsReq = RowsRd
C   DATEKEY                 READE            ITEMFACTL1
C                  If                %EOF
C                           Leave
C                  Endif
C   PARTKEY     CHAIN       PART_DIML1
C                  If       Not %FOUND
C                  Iter
C                  Endif
C   CUSTKEY                 CHAIN CUST_DIML1
C                  If       Not %FOUND
C                  Iter
C                  Endif
C   SUPPKEY                 CHAIN SUPP_DIML1
C                  If       Not %FOUND
C                  Iter
C                  Endif  ...
```

Modernizing Data Access

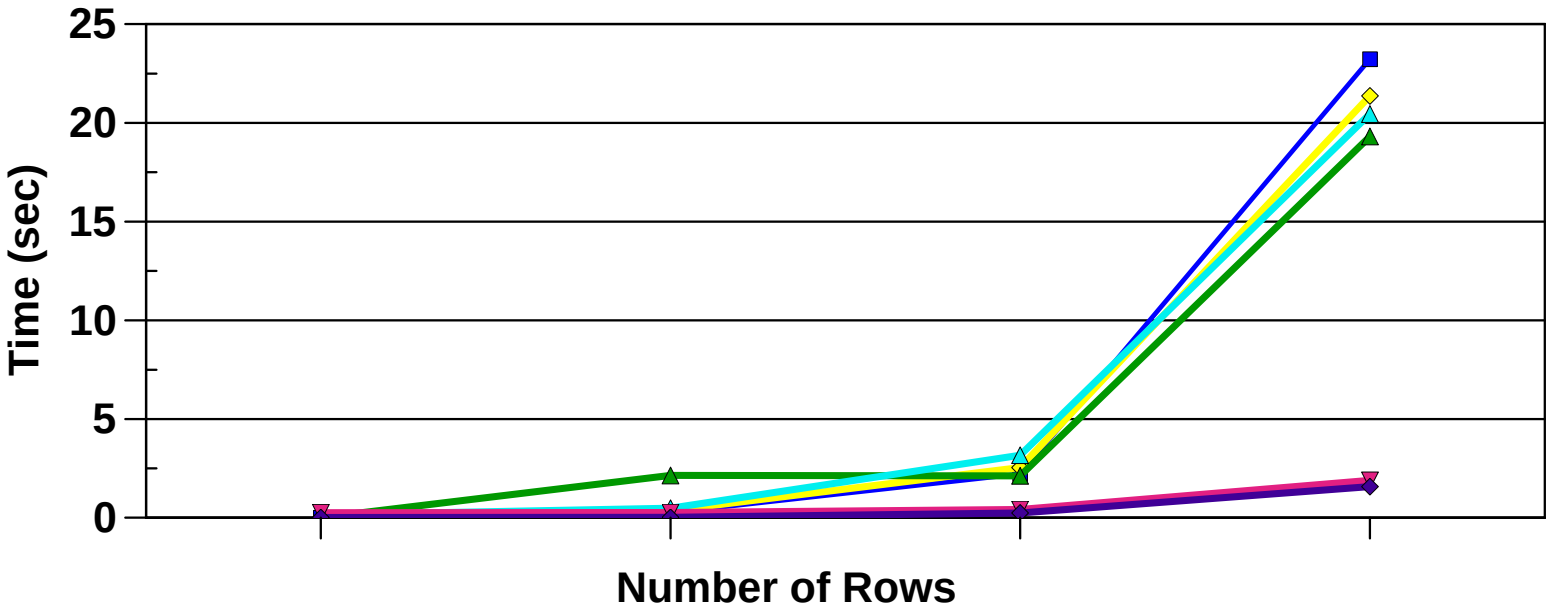
Native I/O to SQL Example - Joined LFs & Views

```
...

C/EXEC SQL
C+ DECLARE sql_jn CURSOR FOR
C+     SELECT *  FROM JoinView
C+       WHERE year=2008 AND month=6
C/END-EXEC

C/EXEC SQL
C+ OPEN sql_jn
C/END-EXEC

C/EXEC SQL
C+ FETCH NEXT FROM sql_jn FOR
C+      :RowsReq ROWS INTO :result_set
C/END-EXEC

C                 If        SQLCOD = 0   and
C                                        SQLER5 = 100 and
C                                 SQLER3 > 0
C                 Eval              RowsRd = SQLER3
```

```
..
C    SearchKey        KList
C                     Kfld            SearchYear
C                Kfld            SearchMonth
...

C    SearchKey        SETLL           NTVJOIN002
C                If                %FOUND
C                     DO             RowsReq Times
C    Times            Occur          Result_Set
C                     READ           NTVJOIN002
C                     If             %EOF
C                     Leave
C                     Endif

C                Eval            RowsRd = RowsRd + 1
C                ENDDO

C                     Endif
```

Modernizing Data Access

Native I/O to SQL Example - Performance Comparison



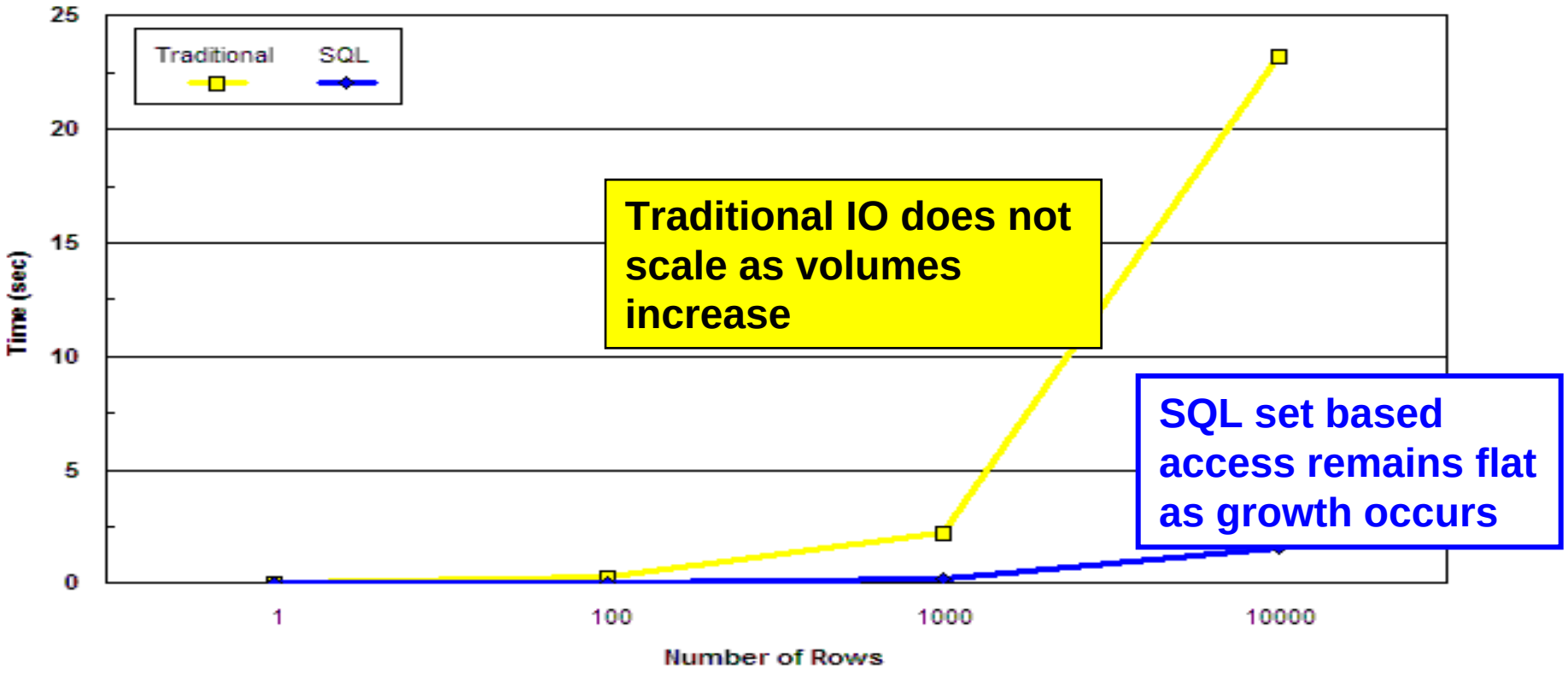
Note: Tests run on Model 720 w/1600 CPW & 2 GB Memory - your performance results may vary

	1	100	1000	10000
Native File Join	0.002512	0.260248	2.219504	23.228176
Native JoinLF	0.002304	0.362128	2.544608	21.366480
Native JoinLF w	0.002400	2.144288	2.125032	19.311464
SQL - No IOA	0.145160	0.489136	3.166704	20.452984
SQL IOA	0.251168	0.267208	0.417800	1.898800
SQL SQE IOA	0.013536	0.019320	0.250160	1.576536

Modernizing Data Access

SQL and Scalability

- The issue is throughput not response time
  - As growth occurs, programs with Record Level Access (RLA) have a harder time scaling on IBM POWER Systems
  - Throwing hardware at the problem no longer an option
  - Application changes will be inevitable



Modernizing Data Access

Native to SQL Considerations

- ORDER BY clause is the **only way** to guarantee the sequencing of results when using SQL - no clause, means ordering by chance

- SQL Precompilers do not always support all the latest features in the high-level language, still missing from RPG SQL Precompiler:
  - Support for qualified names with more than one level of qualification

- Consider impact of SQL isolation level & journaling on native applications

- Critical Performance Success Factors
  - Sound Indexing & Statistics Strategy
    ibm.com/partnerworld/wps/servlet/ContentHandler/servers/enable/site/bi/strategy/index.html
  - Maximize Open Data Path (ODP) Reuse
    - Prepare Once, Execute Many
    - Connection Pooling
    - Keep Connections & Jobs active as long as possible
    - Reference:
      ibm.com/partnerworld/wps/servlet/ContentHandler/servers/enable/site/education/ibp/4fa6/
  - Use Blocked Fetches & Inserts
  - Attend SQL Performance Workshop – ibm.com/systems/i/db2/db2performance.html

Next Steps

1) Identify First Project
   - – Write a new function/program component using SQL
   - – Rewrite an existing component using SQL (eg, reporting)
     - OPNQRYF to SQL
     - Query/400 to DB2 Web Query
   - – Port SQL-based program to DB2 for i
     - Porting guides & conversion tools at:
     - http://ibm.com/partnerworld/i/db2porting

Next Steps

2) Get Education
- *IBM i Database Modernization Workshop*
  - http://ibm.com/systems/i/support/itc/educ/lsdb2mod.html

- *Modernizing iSeries Application Data Access* Redbooks document
  www.redbooks.ibm.com/abstracts/sg246393.html?Open
- *Case Study: Modernizing a DB2 for iSeries Application* white paper
  ibm.com/partnerworld/wps/servlet/ContentHandler/servers/enable/site/education/wp/9e5a/index.html

- DB2 for i SQL Performance Workshop
  - ibm.com/systems/i/db2/db2performance.html
  - ibm.com/partnerworld/wps/training/i5os/courses
- Indexing & Stats Strategy White Paper
  ibm.com/partnerworld/wps/servlet/ContentHandler/servers/enable/site/bi/strategy/index.html
- Database modernization roadmaps
  - **Modernizing DB2 definitions and usage**
    https://www.ibm.com/partnerworld/wps/servlet/ContentHandler/SOX_TwGV47Qq9ppycFAT
  - **Modernizing data access with SQL**
    https://www.ibm.com/partnerworld/wps/servlet/ContentHandler/SOX_JUGV47Q9cz7ycFAT
  - **Optimizing SQL performance**
    https://www.ibm.com/partnerworld/wps/servlet/ContentHandler/SOX_G7FV47QlUAiycFAT

Conclusion

- DDS and Native Record-Level Access are
- not sustainable

- Must migrate both Native to SQL, and your Mind to SQL

- There is no reason not to keep your business data in DB2 for i

Additional Information

- DB2 for i Websites
  - Homepage: ibm.com/systems/i/db2
  - developerWorks Zone: ibm.com/developerworks/db2/products/db2i5OS
- Newsgroups
  - DeveloperWorks: https://www.ibm.com/developerworks/forums/forum.jspa?forumID=292
  - System i Network DB2 Forum - http://forums.systeminetwork.com/isnetforums/
- Education Resources - Classroom & Online
  - http://ibm.com/systems/i/db2/db2educ_m.html
  - http://ibm.com/partnerworld/wps/training/i5os/courses
- DB2 for i Publications
  - Online Manuals: http://ibm.com/systems/i/db2/books.html
  - White Papers: ibm.com/partnerworld/wps/whitepaper/i5os
  - Porting Help: http://ibm.com/partnerworld/i/db2porting
  - DB2 for i5/OS Redbooks (http://ibm.com/systemi/db2/relredbooks.html)
    - Stored Procedures, Triggers, & User-Defined Functions on DB2 for iSeries (SG24-6503)
    - DB2 for AS/400 Object Relational Support (SG24-5409)
    - Advanced Functions & Administration on DB2 for iSeries (SG24-4249)
    - Getting Started with DB2 Web Query for System i (SG24-7214)
  - SQL for DB2 by Conte & Cooper
    - http://www.amazon.com/SQL-James-Cooper-Paul-Conte/dp/1583041230/

DB2 Modernization Assistance

DB2 for i Modernization Workshop
http://ibm.com/systems/i/support/itc/educ/lsdb2mod.html

➔Need help?

IBM DB2 for i Consulting and Services

- ✓ Database modernization
- ✓ DB2 Web Query
- ✓ 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
IBM Systems and Technology Group
Rochester, MN USA