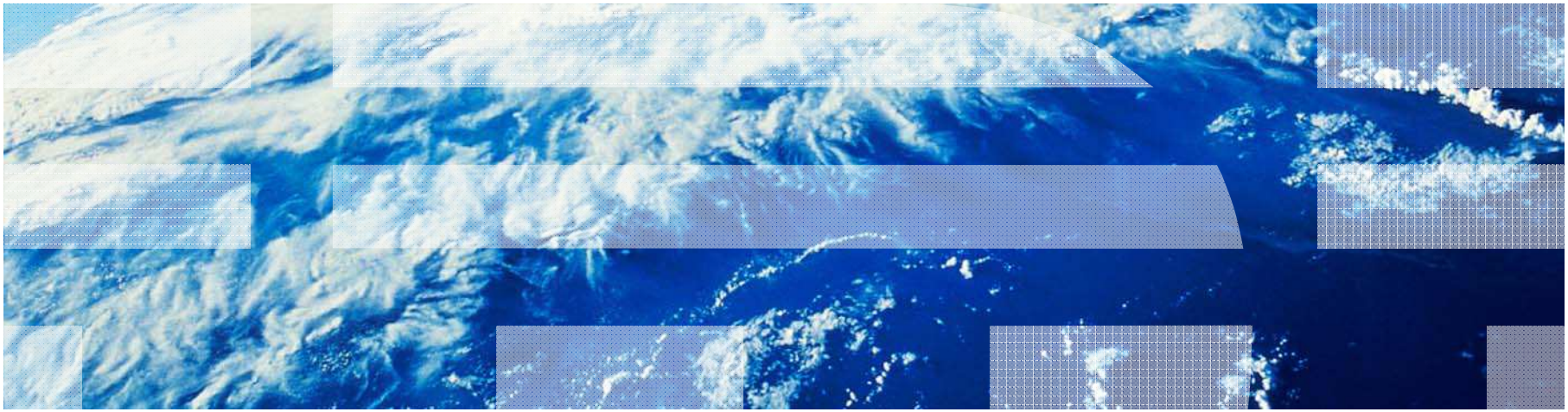
A Data Centric Approach to Application Development - Using SQL to do the Heavy Lifting
Quser January 15, 2013

Dan Cruikshank

dcrank@us.ibm.com



stgls@us.ibm.com
ibm.com/systems/services/labservices

Agenda

- Quser Back in the Day
- Data Centric Development
- Transparent Migration from DDS to DDL

Quser Back in the Day

- I served as an officer of Quser for 4 years from 1983-86
  - I was preceded by Dick Jacobson founder of Help/Systems
- Notable Speakers
  - Ken Kelley President and founder of Advanced System Concepts
    - Makers of Abstract/Probe and SEQUEL (now part of Help/Systems)
    - Topic: Object Oriented Programming
  - Paul Conte Author and DB2 for i Database Guru
    - First professional speaking engagement
    - Topic: Relational Database Design
- Implicit/Explicit Record Format sharing
  - Foundation for transparent DDS to DDL migration
  - Concepts were introduced at Quser as technique to avoid level check errors without recompiling

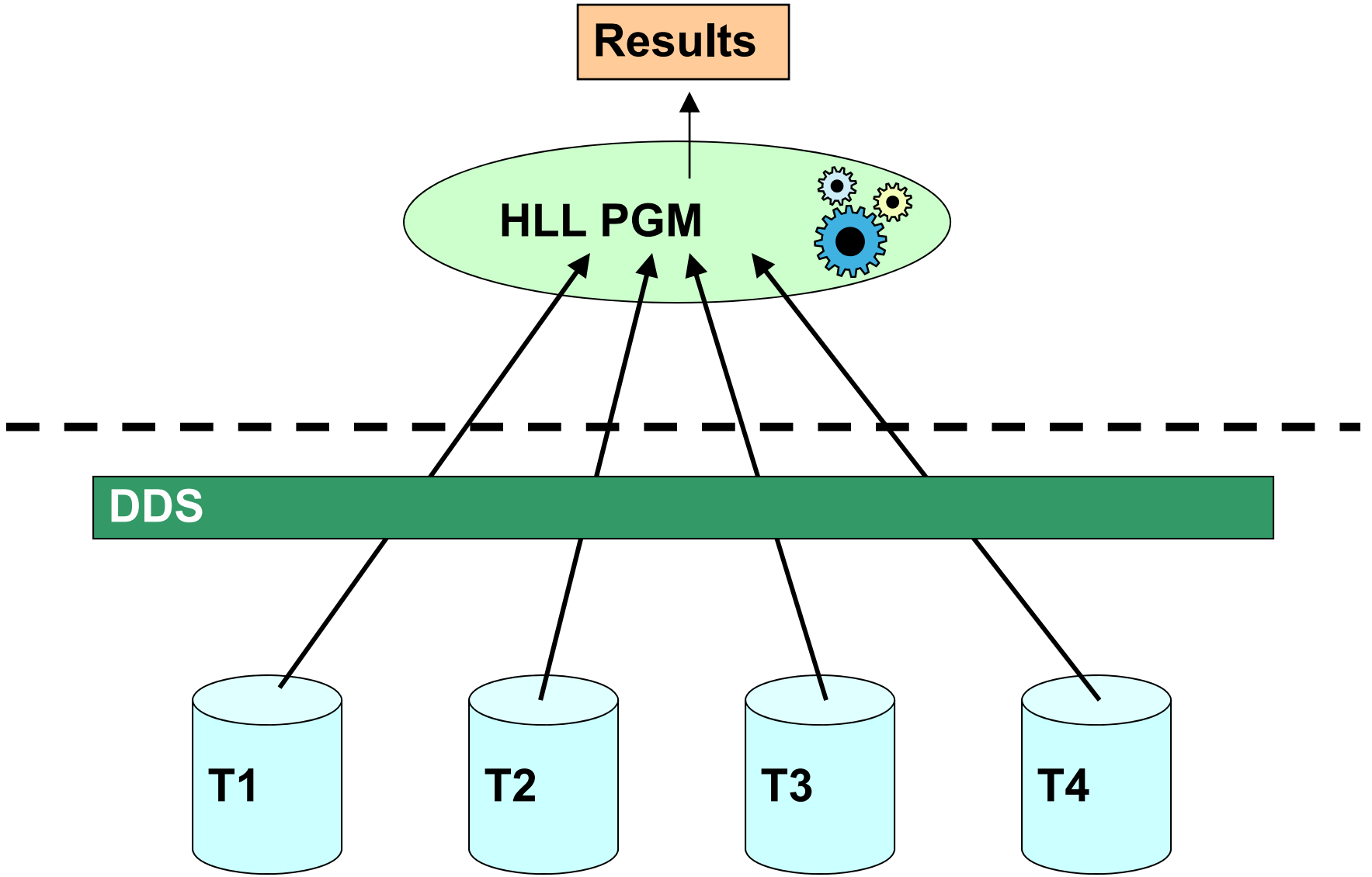
Data Centric Development

- Data Centric Programming and Set Based Processing
- Defining Sets using SQL DDL
- Accessing Sets using SQL DML

What is Data Centric Programming?

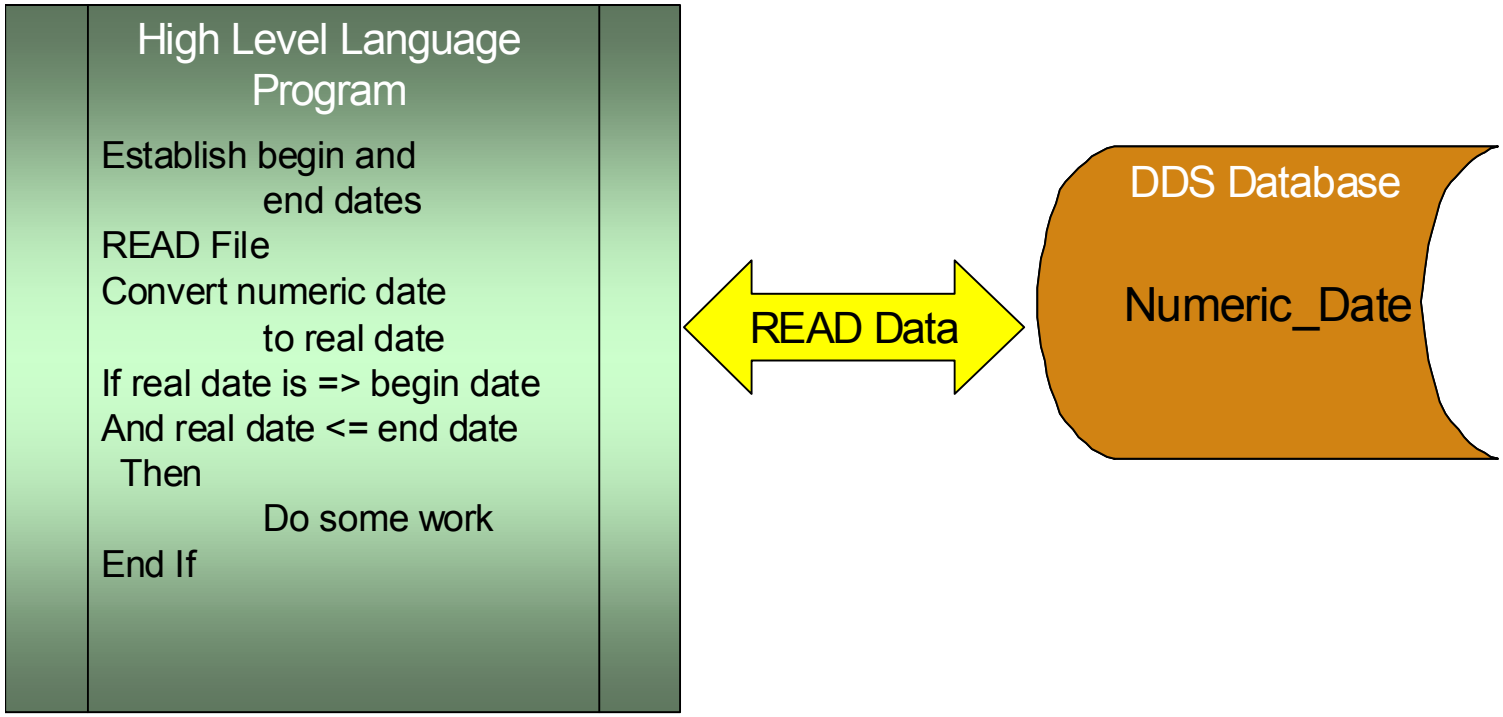
- Definition:
  - Solving business problems using database capabilities
  - Getting the database management system to do more on your behalf
  - Implementing more of the business logic in the database
  - Separating the business logic from the high level application programs

Application Centric Overview

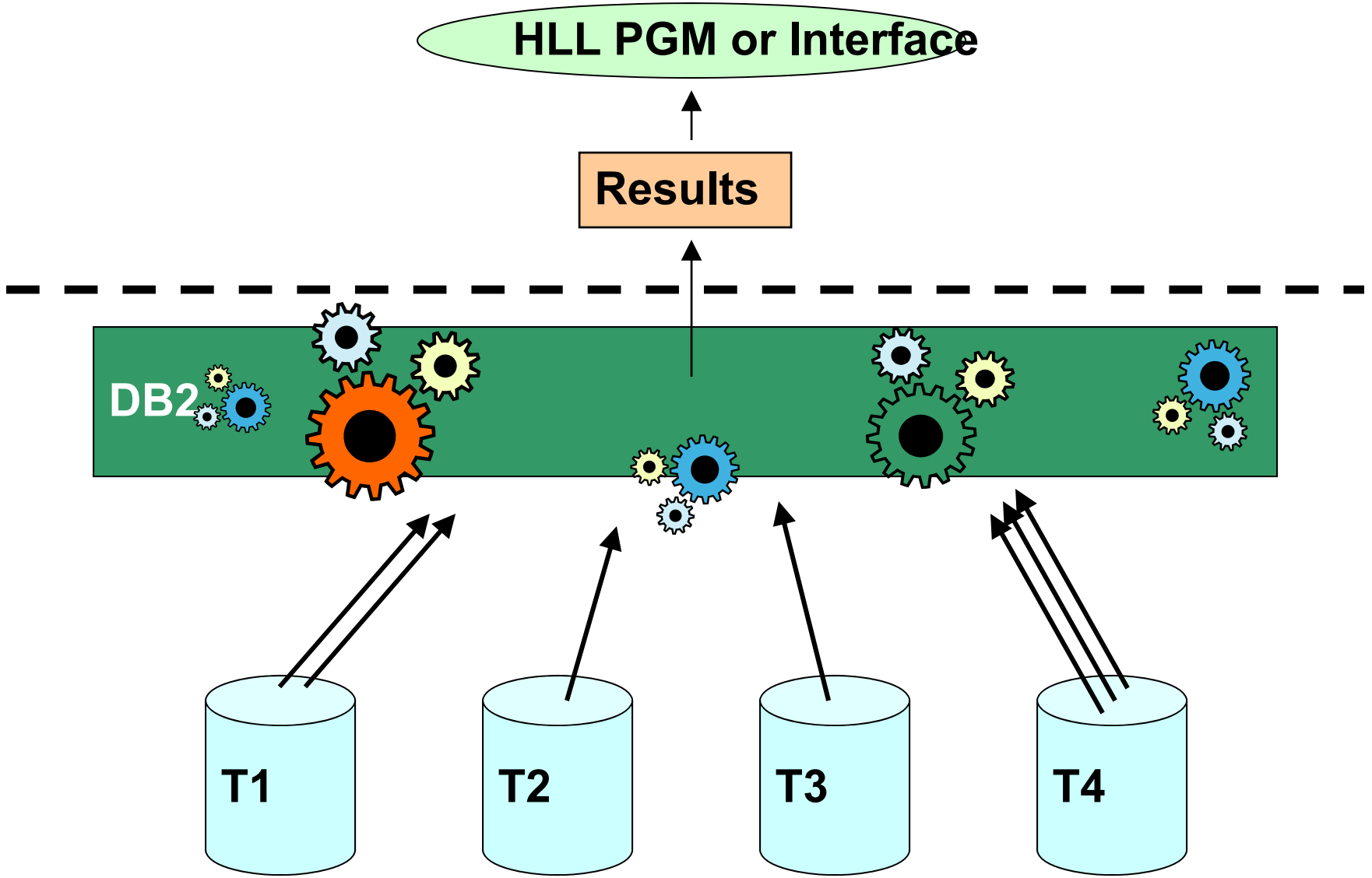


Application Centric Example

- More application logic
- Small pipe (1 row at a time)
- Not using much intelligence in the database

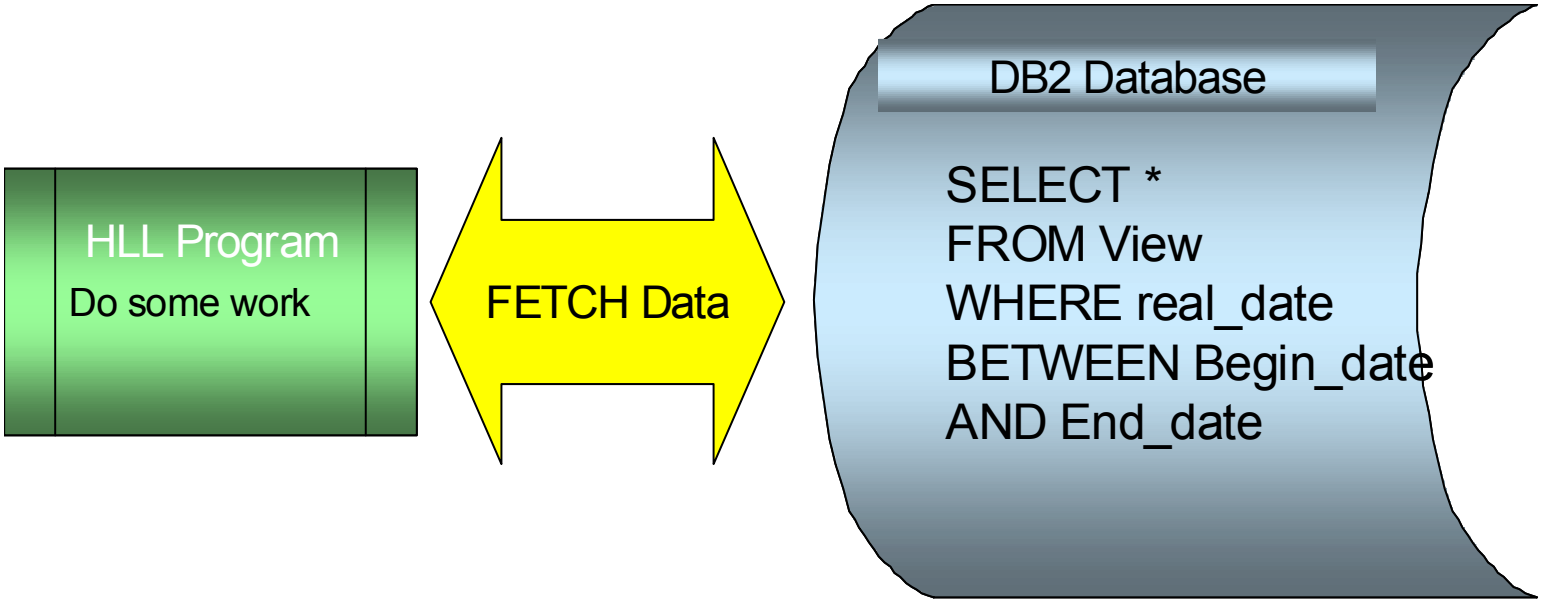


Data Centric Overview



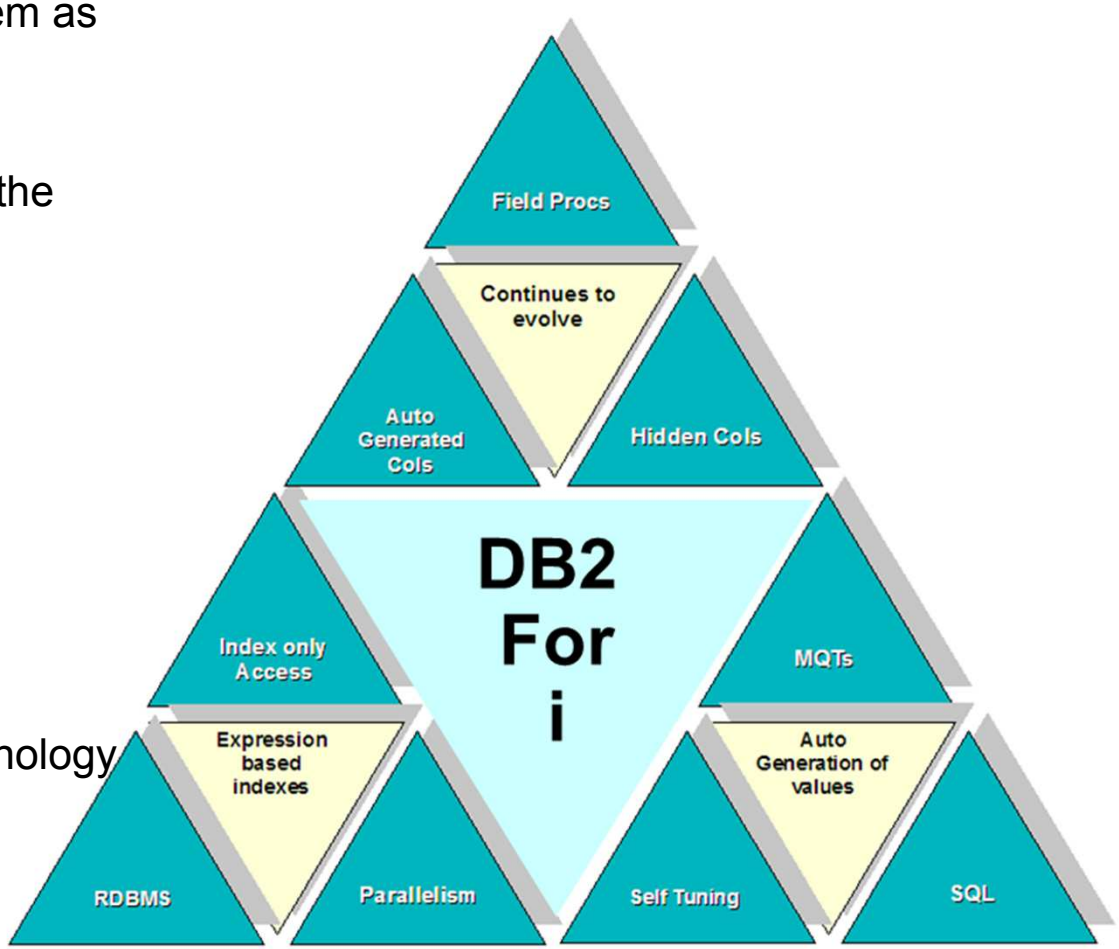
Data Centric Programming Example

- Less application logic
  - Less application development and testing
  - You can run and test SQL without any programming
- Big pipe (sets of many rows processed at a time)
- Taking advantage of more intelligence in the database



Data Centric Programming Scalability

- Goal is to drive as much work down into the database management system as possible.
- Define business rules as part of the database
  - Rules apply to all application interfaces
- Take advantage of SQL only capabilities
- Database evolves to:
  - Meet new requirements
  - Take advantage of new technology



SQL and Data Centric Programming

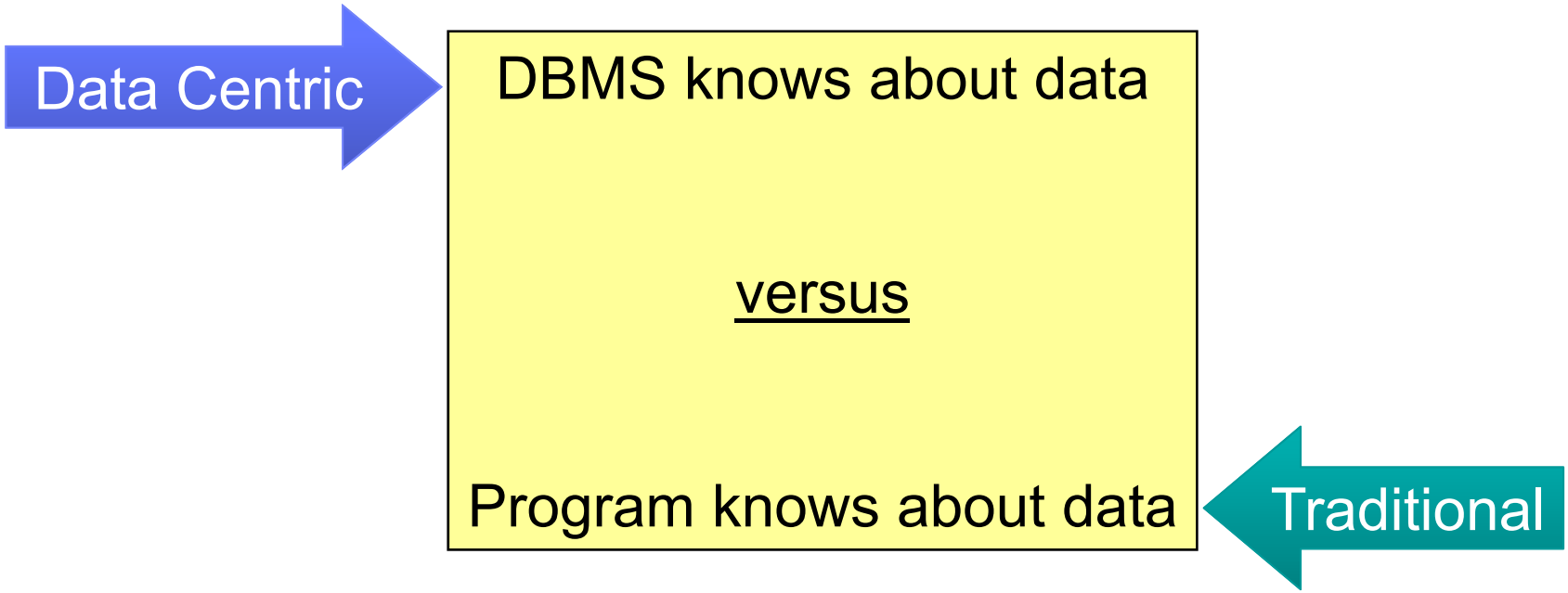
SQL works very well for Data Centric Programming

SQL works best for processing sets of data

Set Based Thinking and SQL

- SQL is declarative and not procedural
  - Think in terms of defining the sets and the operations on those sets
- DDL (Data Definition Language)
  - Defines the sets
- DML (Data Manipulation Language)
  - Operates upon those sets

Enabling Sets



SQL and Data Centric Programming

Proper *data modeling* results in good "set" definitions

Proper SQL usage results in good *set based* operations

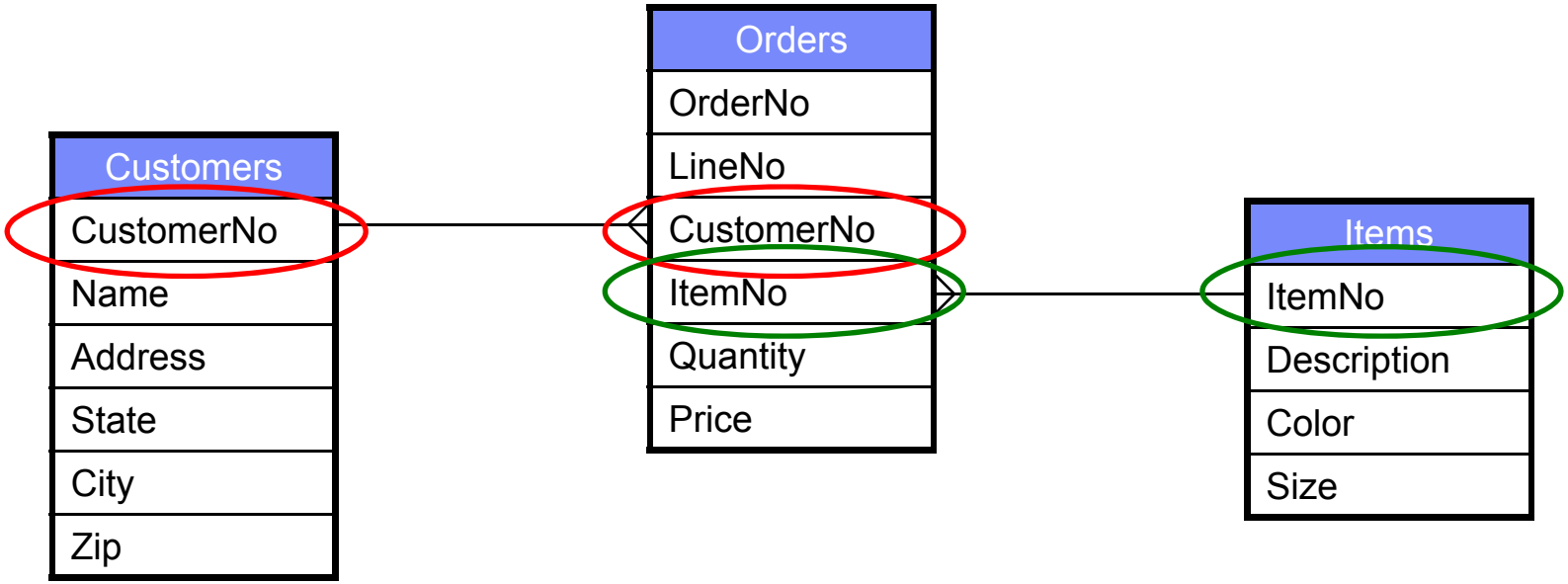
Agenda

- Data Centric Programming and Set Based Processing
- **DEFINING SETS USING SQL DDL**
- Accessing Sets using SQL DML

Data Centric Constructs Used When Defining the Database

- Constraints
  - The basis for a solid foundation
- Indexes
  - The plumbing and wiring
- SQL Views
  - The walls, roof, doors and windows
- DB2 for i Auto Generated Column Support
  - The “must have” appliances

What are the Data Integrity Rules?



What happens if we are updating Orders but have specified an invalid CustomerNo value?

What happens if we drop a customer row from Customers?

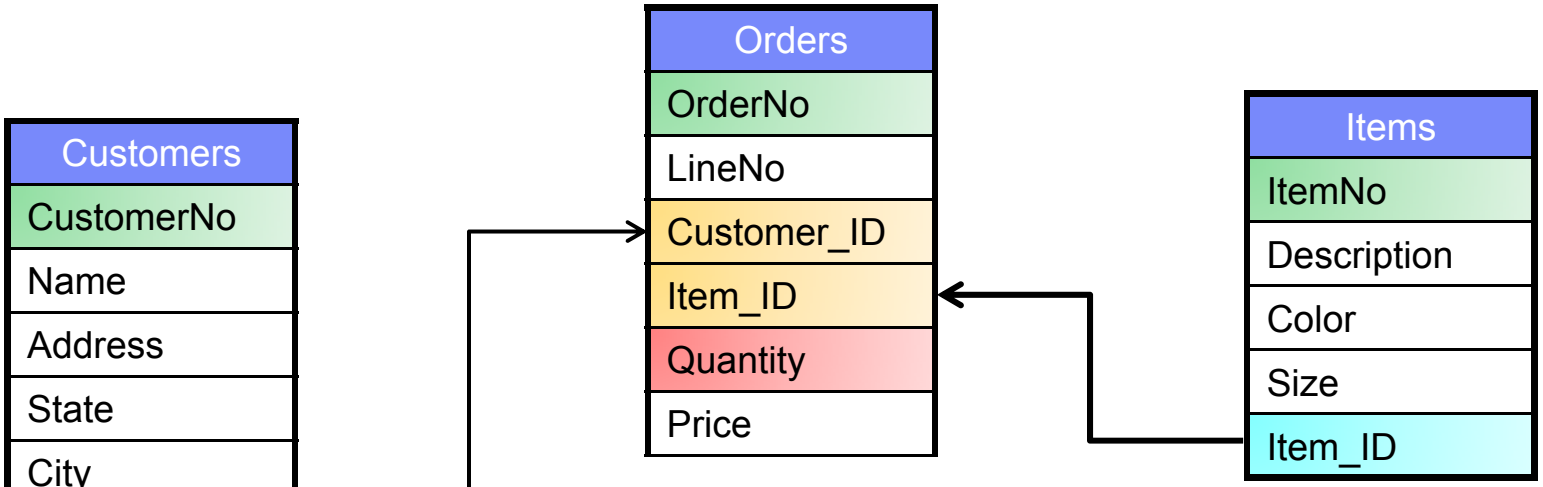
What happens if we add an element to Items with an invalid size?

Constraints

- Unique key constraint
  - the values of the key are valid only if they are unique
  - More than 1 per table
- Primary key constraint
  - the values of the key are valid only if they are unique and not null-able
  - Only 1 per table
- Referential constraint
  - the values of the “foreign key” are valid only if one of the following conditions is met:
    - foreign key appear as values of a parent key
    - the foreign key is null
- Check constraint
  - the rule that limits the values allowed in a column or group of columns

A constraint is a rule enforced by the database manager to limit the values that can be inserted, deleted, or updated in a table. It is defined at CREATE time or later with an ALTER.

Enforcing the Rules



Constraints	Rules
UNIQUE	Identifies unique public key
PRIMARY KEY	Identifies parent key
FOREIGN KEY	Matches parent key. Defines actions for insert, update and delete events
CHECK	Ensures proper values, e.g. Quantity > 0

All applications must follow the rules

Indexes

- Indexes are needed for optimal performance
  - 1 Index can service several SQL requests
- Indexes are not referenced directly in SQL statements like SELECT, UPDATE, etc.
  - That's the job of the query optimizer, it decides if and when and how to use indexes
- Indexes can contain:
  - Expressions: CREATE INDEX X1 ON T1 (**QTY * PRICE**)
  - Functions: CREATE INDEX X2 ON T1 (**UPPER(NAME**))
  - Aggregates: CREATE EVI E1 ON T1 (NAME) **INCLUDE(SUM(QTY))**
  - Non-keyed columns: CREATE INDEX X3 ON T1 (DEPT) **ADD COLUMNS ID, NAME**
  - Row filtering: CREATE INDEX X4 ON T1 (NAME) **WHERE DEPT = 'D11'**

INDEX Usage Examples

```
SELECT …. FROM T1 ORDER BY DEPT;

SELECT …. FROM T1 WHERE DEPT =
  'D11' ORDER BY NAME;
```

One index servicing two queries

```
CREATE INDEX X3 ON T1 (DEPT, NAME);
```

Multiple Indexes servicing 1 query

```
SELECT …. FROM T1 WHERE DEPT =
  'D11'  AND SUBSTR(NAME,1,3) = 'ABC'
  ORDER BY NAME;
```

```
CREATE INDEX X3 ON T1 (DEPT, NAME);
```

+

```
CREATE INDEX X7 ON T1 (SUBSTR(NAME,1,3));
```

An aggregate Index supporting SUM

```
SELECT DEPT, SUM(QTY) FROM T1
  GROUP BY DEPT
  ORDER BY DEPT;
```

```
CREATE INDEX E7 ON T1 (DEPT)
  INCLUDE SUM(QTY);
```

SQL Views

- SQL views are used to show some subset of the data or to capture and isolate some complex operation
- SQL views do not contain data or access paths
- Views have no implied order
- Views are referenced just like tables in SQL statements
- Views can be joined to other views
- Views can be used to externally describe data structures

VIEW Usage Examples

Complex Query

```
SELECT … FROM T1, T2, T3,T4…
GROUP BY… ORDER BY…
```

Masking Complexity

```
CREATE VIEW COMPLEX_QUERY
AS SELECT … FROM T1, T2, T3,T4… GROUP BY…;

SELECT * FROM COMPLEX_QUERY ORDER BY…;
```

Program Described Structure

```
T1_RESULT DS
  A  CHAR(10)
  B  DECIMAL(11,0)
  C  DATE
SELECT A, B, C INTO  :T1_RESULT FROM T1
```

Externally Described Structure

```
CREATE VIEW  T1_VIEW
AS SELECT A, B, C FROM T1;

T1_RESULT DS EXTNAME(T1_VIEW)

SELECT *  INTO  :T1_RESULT FROM T1_VIEW
```

Multiple Views and Data Structures

```
T1_RESULT DS EXTNAME T1_VIEW
T2_RESULT DS EXTNAME T2_VIEW

SELECT * INTO  :T1_RESULT FROM T1_VIEW
SELECT * INTO  :T2_RESULT FROM T2_VIEW
WHERE T2.A = :T1_RESULT.A
```

Joined View One Structure

```
CREATE VIEW  JOIN_VIEW AS
SELECT A, B, C, D, E, F FROM T1 JOIN T2 USING(A);

JOIN_RESULT DS EXTNAME(JOIN_VIEW)

SELECT * FROM JOIN_VIEW INTO  :JOIN_RESULT
```

Auto-generated Values

- DB2 for i can automatically generate the value for a column
- A value can be auto-generated in 1 of following ways:
  - Defined as a column attribute
  - Defined as a column type (ROWID)
  - Extracted from a external object
- The following SQL column attributes allow auto-generation:
  - Row Change Timestamp
  - Identity
- A Sequence object contains a system generated value
  - Is external from a table
  - Can be used on INSERT statements to assign the value to a column

Auto-Generated Column Examples

Row Change Timestamp

```
CREATE TABLE ORDERS
  (ORDER_KEY BIGINT …,
  ORDER_CHANGE_AUDIT TIMESTAMP
NOT NULL
  FOR EACH ROW ON UPDATE AS ROW
CHANGE TIMESTAMP)
IMPLICITLY HIDDEN;
```

Notes

IMPLICITLY HIDDEN **prevents column from being included in SQL * statements**

ROW CHANGE TIMESTAMP function can be used without column name knowledge

SELECT …. ROW CHANGE TIMESTAMP FOR T1 AS RCTS FROM T1

UPDATE T1 ….. WHERE ORDER_KEY = :v_ORDER_KEY AND ROW CHANGE TIMESTAMP FOR T1 = :v_RCTS

Identity Column

```
CREATE TABLE ORDERS
  (ORDER_ID BIGINT NOT NULL
  GENERATED BY DEFAULT AS IDENTITY
PRIMARY KEY IMPLICITLY HIDDEN,
```

ID Column Notes

Only 1 IDENTITY column per table

- Ideal as meaningless PRIMARY KEY

GENERATED ALWAYS results in the generation of a new value automatically.

- Existing values are ignored
- Can be an issue when copying or duplicating data

GENERATED BY DEFAULT only generates a value if the auto-generated column is not included in the data

Agenda

- Data Centric Programming and Set Based Processing
- Defining Sets using SQL DDL
- **ACCESSING SETS USING SQL DML**

Data Centric Constructs Used When Accessing the Database

- Joins
- Table Expressions
- Accessing Result Sets From HLL Programs
- Dynamic SQL

Join Concepts

- Joins allow columns from more than one table to be returned as a single row
- DB2 for i supports the following types of joins:
  - INNER, LEFT, RIGHT and FULL OUTER, LEFT and RIGHT EXCEPTION, CROSS (Cartesian product)
- A single join can reference 1000 tables
  - Combination of base (physical), virtual (logical), derived (expression) and non-relational (table functions)
  - Only 256 allowed within a view
- Joins can be updated via Instead of Triggers
- Any column from any table can be used in GROUP BY or ORDER BY
  - Including derived columns

Join Usage Examples

INNER JOIN 2 or more base tables

```
SELECT a.col1, b.col2, c.col3
FROM T1 a JOIN T2 b USING (ID)
  JOIN T3 c ON a.ID = c.ID
```

INNER JOIN notes

USING shorthand if join column names are same

ON if names are different

FROM T1, T2 WHERE T1.ID = T2.ID (also works)

No difference in implementation of the above

OUTER JOIN examples

```
SELECT a.col1, b.col2, c.col3
FROM T1 a LEFT JOIN T2 b USING (ID)
  JOIN T3 c ON a.ID = c.ID

SELECT a.col1, b.col2, c.col3
FROM T1 a RIGHT JOIN T2 b USING (ID)
```

OUTER JOIN notes

Unmatched columns from LEFT or RIGHT tables returns NULL values

IFNULL or COALESCE can be used to return value

Example: SELECT a.col1, IFNULL(b.col2, 'No Match')

FROM T1 a LEFT JOIN T2 b USING (ID

JOINing/ORDERing Multiple Table Types

```
SELECT a.col1, b.col2, c.col3
FROM Table a LEFT JOIN View b USING (ID)
  JOIN TABLE(SELECT ID, MAX(val) col3 FROM T3) c
    ON a.ID = c.ID
ORDER BY a.col1, b.col2
```

Multiple Table Type notes

DB2 can sort the result set to satisfy ORDER BY requirement

TABLE is optional (required for table functions)

LATERAL can be used instead of TABLE

Common Table Expressions (CTEs)

- Two types: Nested and Common Table Expressions
  - Nested appears on FROM clause (best for single use)
  - Common is part of WITH clause (named expression can be recursive)
- Possible applications
  - Breaking a report into logical steps
    - Improve readability
    - Reduce usage of physical work tables
  - Recursive SQL
  - Enabling a query to be re-used within a query

Eliminating Multiple Step Queries

Application Centric - 3 steps

```
DECLARE GLOBAL TEMPORARY TABLE
Step1 AS
(SELECT shipdate, customer, phone,
orderkey, linenumber
  FROM item_fact i, cust_dim c
  WHERE c.custkey=i.custkey AND
discount=0.08) WITH DATA;

DECLARE GLOBAL TEMPORARY TABLE
Step2 AS
(SELECT customer, phone, orderkey,
linenumber, year, quarter
  FROM Step1 , star1g.time_dim t
  WHERE t.datekey=shipdate ) WITH DATA;

SELECT * FROM Step2 ;
```

Data Centric – 1 step

```
WITH EXP1 AS
 (SELECT shipdate, customer, phone,
orderkey, linenumber
  FROM item_fact i, cust_dim c
  WHERE c.custkey = i.custkey AND
discount=0.08),

 EXP2 AS
  (SELECT customer, phone, orderkey,
linenumber, year, quarter
   FROM EXP1 , star1g.time_dim t
   WHERE t.datekey = shipdate)

SELECT * FROM EXP2 ;
```

Result Sets and Procedures

- Stored procedures in combination with result sets can drastically reduce network trips by returning blocks of results
- Stored procedures that return result sets can only be called by the following interfaces
  - System i Access ODBC, OLE DB & ADO.NET middleware
  - SQL CLI
  - Toolbox JDBC driver
  - Native JDBC driver
  - DB2 Connect
  - IBM i DRDA Connections
  - New with 7.1 - Embedded SQL & SQL Routines
- Result sets are returned via open SQL cursors
  - Multiple result sets can be returned by a single stored procedure
  - OPEN CURSOR or SET RESULT SETS
    - SET RESULT SETS CURSOR can be used to control order of result sets
    - Only 1 SET RESULT SETS ARRAY can be returned
- 7.1 TR5 – Named Arguments and Defaults for Parameters

Accessing Result Sets From High Level

RPG SQL Program

```
D RS_LOC1   S
    SQLTYPE(RESULT_SET_LOCATOR)
EXEC SQL CALL RS1_PROC (E11);
EXEC SQL ASSOCIATE RESULT SET
   LOCATORS (:RS_Loc1)WITH PROCEDURE
   RS1_PROC;
EXEC SQL ALLOCATE C1 CURSOR FOR
   RESULT SET :RS_Loc1;
EXEC SQL FETCH NEXT FROM C1 FOR n
   ROWS INTO:RS_Array;
```

External Application (Java, PHP, .net…

```
SQL CALL RS1_PROC (E11);
```

SQL Stored Procedure

```
CREATE PROCEDURE RS1_PROC (
  IN P_WDEPT CHAR(3))
RESULT SETS 1
DECLARE RS1_PROC_C1 CURSORFOR
   SELECT * FROM VEMPDPT3
   WHERE WORKDEPT = p_WDEPT;
OPEN RS1_PROC_C1;
```

EMPNO	LASTNAME	WORK...	DEPTNAME	PROJ...	ACT...	EMPT...	EMSTDATE
000090	HENDERSON	E11	OPERATIONS	OP1010	10	1.00	1982-01-01
000290	PARKER	E11	OPERATIONS	OP1010	130	1.00	1982-01-01
000280	SCHNEIDER	E11	OPERATIONS	OP1010	130	1.00	1982-01-01
000310	SETRIGHT	E11	OPERATIONS	OP1010	130	1.00	1982-01-01
000300	SMITH	E11	OPERATIONS	OP1010	130	1.00	1982-01-01

Named Arguments and Defaults for Parameters

- New with 7.1 Technology Refresh 5
  - Parameters may be omitted if the routine was defined with a default value
  - Parameters may be specified in any order by specifying the parameter name in the call
  - Works with LANGUAGE SQL and EXTERNAL procedures
  - Same type of support as CL commands
- Examples

```
CREATE PROCEDURE Add_New_Hire (Name CHAR(40),
ID int DEFAULT (select NEXT VALUE from
EmployeeIDs), Dept int DEFAULT 123, Date_Hired
DEFAULT Current Date)
```

Omitting parameters – defaults used

```
CALL Add_New_Hire ('John Doe')
```

Using a named parameter

```
CALL Add_New_Hire('John Doe',
Date_Hired=>'06/23/2012')
```

Advantages of Dynamic SQL

- Offers a high degree of application flexibility
- Can create/build SQL statement
  - Based on parameters received from
    - Interactive user interface
    - List selection techniques
    - Application control file
- Two types:
  - Fixed list (does not require a descriptor)
  - Varying List (requires a descriptor)
- Use any programming language

SQL Descriptor Areas

What are they?

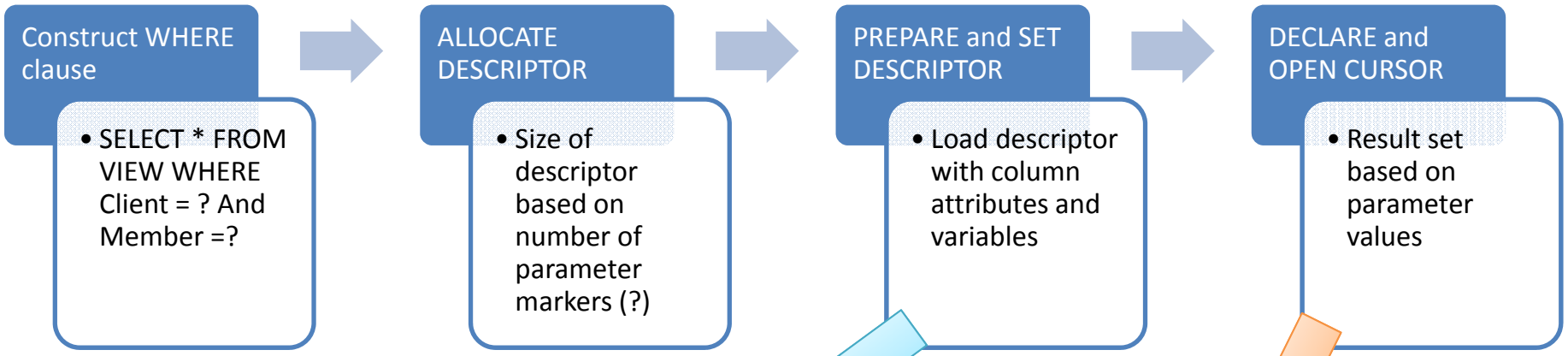
- An SQL descriptor area is used to contain information about a dynamic SQL statement
- A set of variables are used to communicate this information between SQL and your program
  - Think externally described data structure with a variable file name
- The meaning of the information in the descriptor area depends on the type of statement
  - SELECT or non-SELECT (UPDATE, INSERT, etc.)

Where could they be used?

- Eliminate DDS Select/Omit Logical Files or SQL Sparse Indexes
- Replace OPNQRYF or embedded RUNQRY commands
- Create Generic SQL Open Access Handlers
- Provide single stored procedure for all data access to/from a view
- Minimize SQL coding

Dynamic SQL – Variable parameter lists using Descriptors

CALL PROC1 ('ABC', 'X127B89');



Descriptor Size	Max Nbr of Vars	Actual nbr of vars
96 bytes	2	2

Type	Length	Name	Data
CHAR	6	Client	'ABC'
VARCHAR	15	Member	'X127B89'

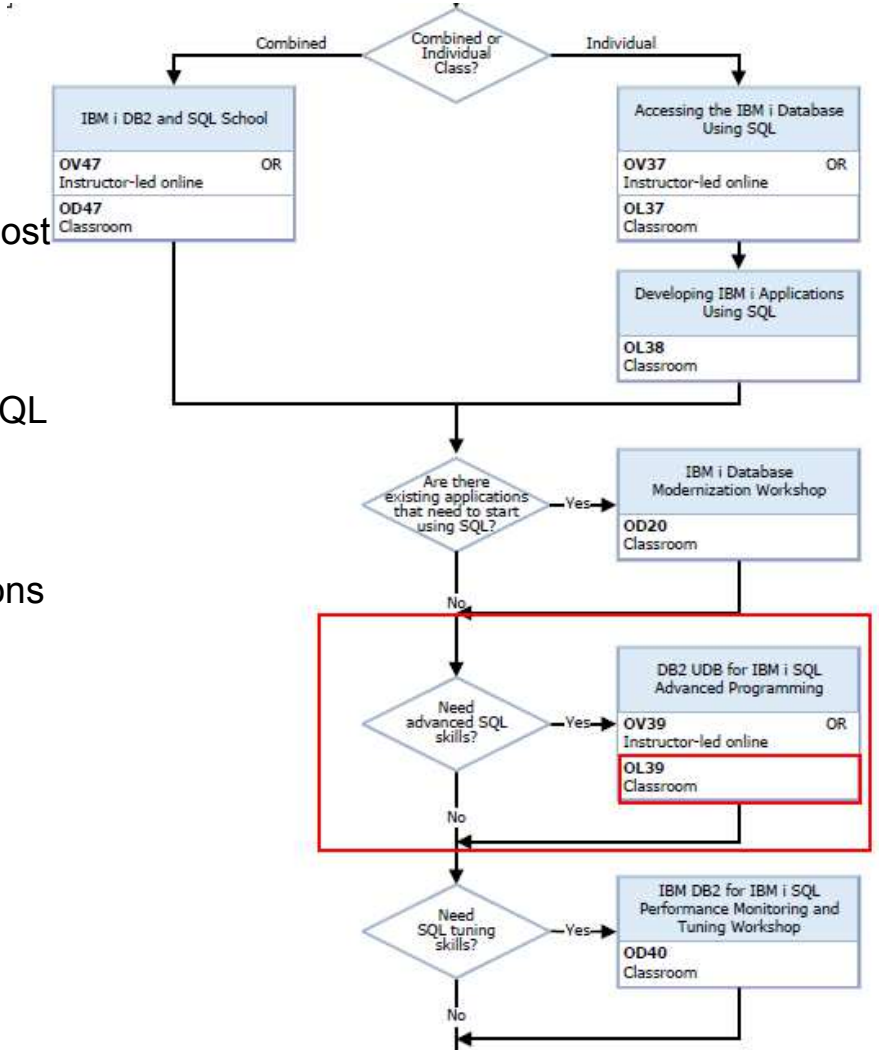
Client	Member	…	…
ABC	X127B89	1	A
ABC	X127B89	2	B
ABC	X127B89	3	C

Summary

- Data Centric Programming and Set Based Processing
  - Think in Sets (i.e. what is the logical unit of work?)
  - Use SQL to define and access data
  - Code data integrity rules and business logic at the database level
- Defining Sets using SQL DDL
  - Use constraints (UNIQUE, PRIMARY KEY, FOREIGN KEY and CHECK)
  - Use SQL indexes to improve performance and provide index access methods
  - Use SQL Views to mask the complexity of the database
  - Use auto-generated columns (IDENTITY, ROW CHANGE TIMESTAMP)
- Accessing Sets using SQL DML
  - Use SQL Joins to return data from multiple tables
  - Use table expressions to enhance and minimize the views
  - Use common SQL Procedures to access data across applications
  - Use dynamic SQL
- Let DB2 do the heavy lifting!!!

IBM Education Roadmap for DB2 & SQL

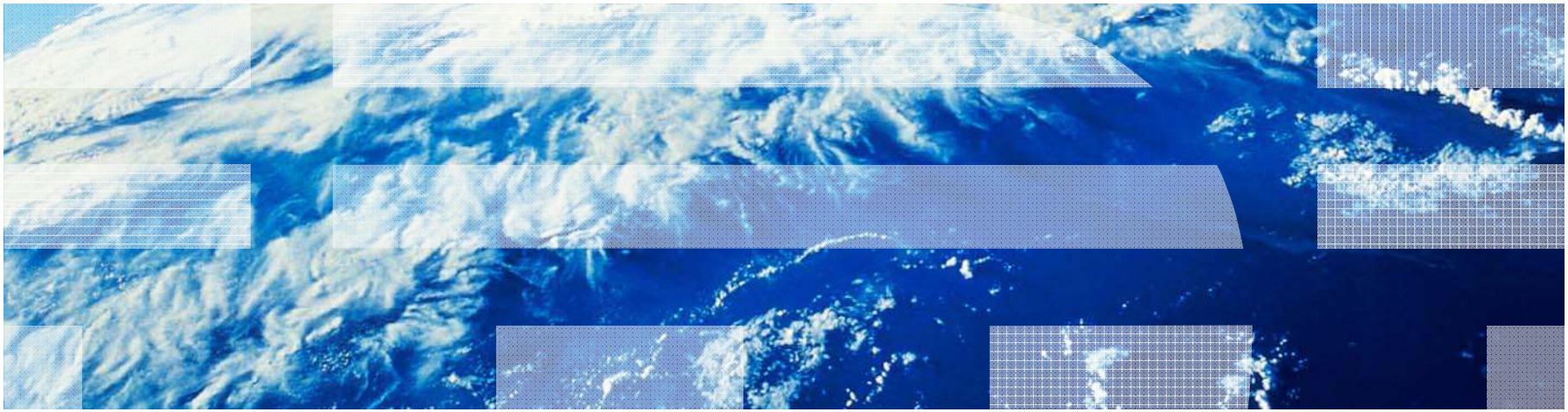
- **DB2 for i Advanced Programming**
  - Data Modeling Best Practices
  - DB2 for i Architecture and Types of SQL
  - Data Cenric Development DDL and DML basics
  - Embedded SQL within an HLL program – use of host variables and indicator variables
  - Thinking in Sets and Processing Data Sets
  - Error Detection Dynamic SQL
  - Debugging Tools – using the debugger, running SQL Scripts
  - Advanced Table and Index Definition
  - Sequence and Variable Objects
  - Joins, views, subqueries, common table expressions
  - Instead of Triggers, Online Analytical Processing (OLAP)
  - SQL Procedures, User Defined Functions, and Triggers
  - And more…
  - Labs
    - SQL only via System I Navigator
    - SQL and Embedded SQL via iNav and RDP and IBM Data Studio



Transparent Migration From DDS to DDL

Dan Cruikshank

dcrank@us.ibm.com



stgls@us.ibm.com
ibm.com/systems/services/labservices

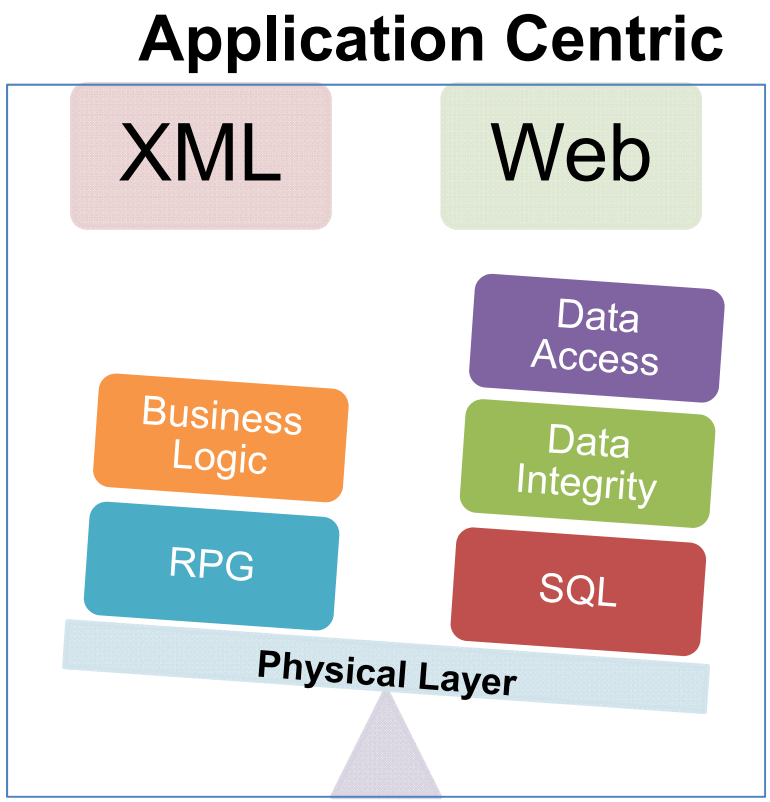
Agenda

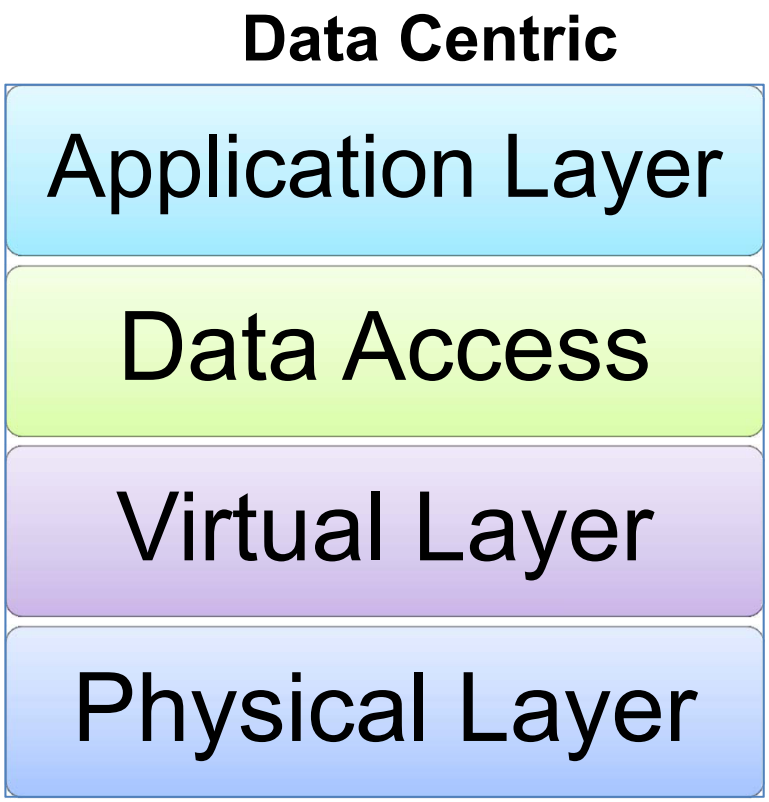
- Getting started
  - Why Data Centric, Why SQL?
  - Training, Teaming and Tools
- The Phased Approach
  - Discovery (Phase 0)
  - Migration (Phase 1)
  - Isolation (Phase 2)
  - Integrity (Phase 3)

Getting Started

- Establish good business reasons for going Data Centric
- Some good examples
  - Expansion into New Markets
  - **Provide advanced Query capabilities**
  - Provide 24/7 Data Access
  - **Overcoming Limits to Growth**
  - **Lack of data integrity**
  - Consolidation of databases or physical file members
  - **Eliminating or reducing batch runs**
- The above are just a few
- Performance is a benefit of Data Centric development

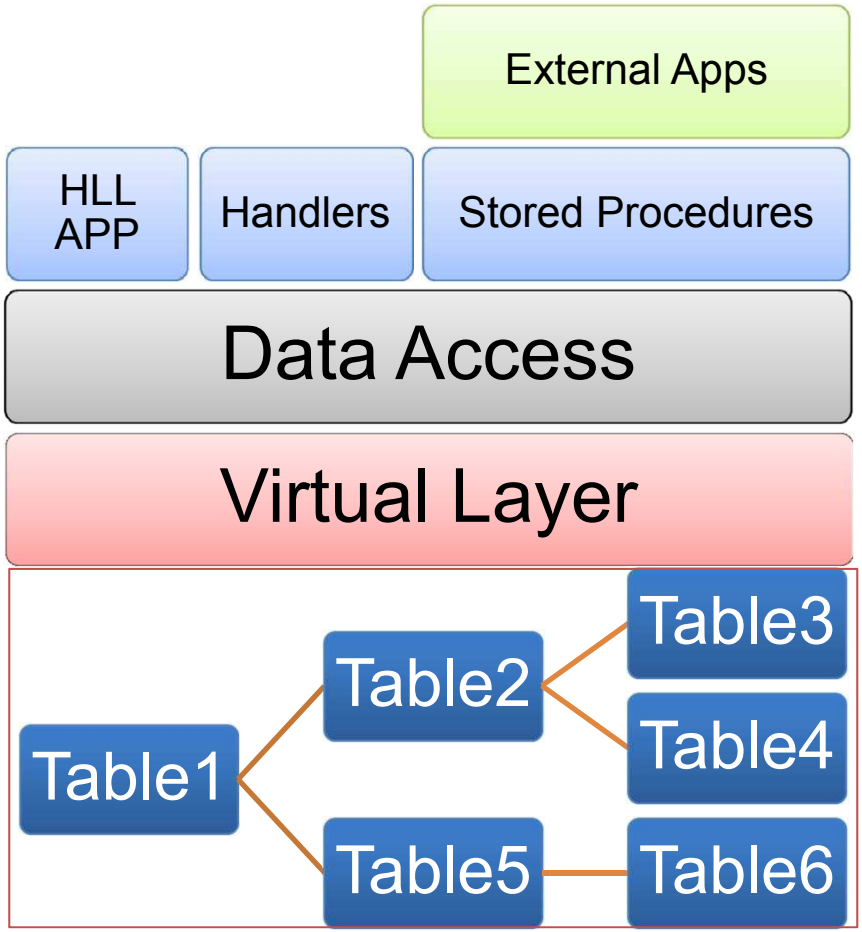
So why SQL?





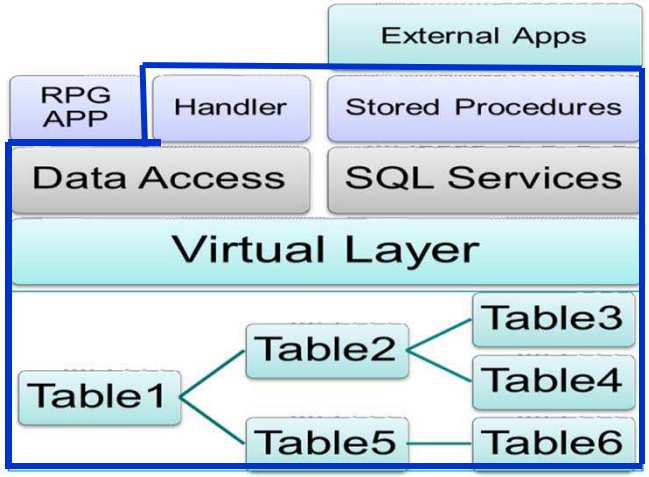
Suggested Framework

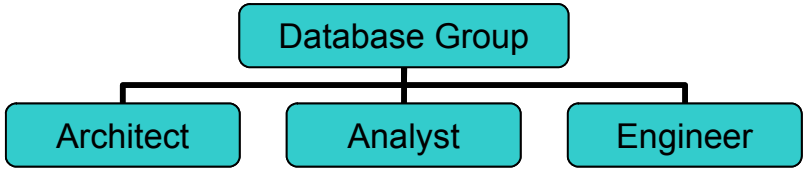
- Application Layer
  - Host programs
  - External access via Stored Procedures
- Data Access Layer
  - SQL DML
  - SQL Routines (procedures,functions,etc)
- Virtual or Data Abstract Layer
  - SQL Views
    - Instead of Triggers
  - Mask complexity
- Physical Data Model
  - DB2 for i
  - Highly normalized
  - Indexes, triggers, constraints



Establish a Database Group

- Many IBM I shops do not have database administrators
  - Database plays second fiddle to other technologies
  - System i professionals are unaware of advances in DB2 for i technology
    - SQL has been available on this platform for over 20 years
- A dedicated database group can stay on top of new changes
  - SQL enhancements
  - Database enhancements





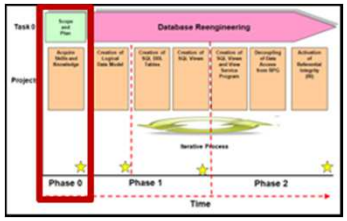
New skill positions emerging

Arm the Database Team with Advanced Tools

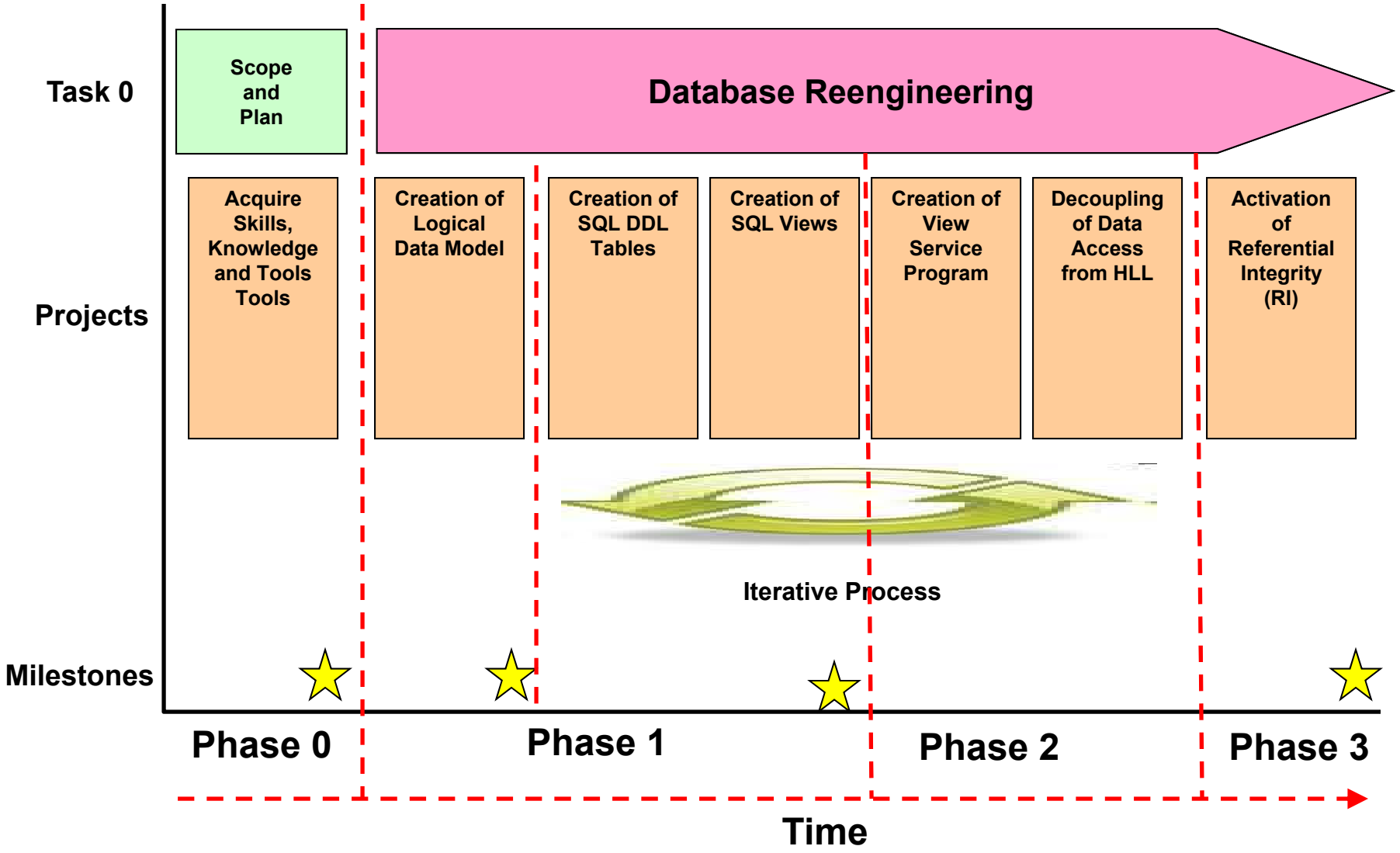
- System i Navigator
- Rational Development Suite which includes:
  - Rational Developer for Power Systems Software for IBM i (RDP)
  - IBM Data Studio (No charge download)
  - http://www.ibm.com/developerworks/downloads/im/data/
- RPG Open Access (No charge RPG enhancement)
  - http://publib.boulder.ibm.com/infocenter/iseries/v7r1m0/topic/books/rzasm.pdf
- DB2 Express-C (No charge download)
  - http://www.ibm.com/software/data/db2/express/download.html
- Third Party Automation Tools
  - Xcase, X-Analysis, AO Foundation, etc.
- Review existing tools (i.e. PM, SCM, IDE, etc.)
  - upgrade or replace as needed with Eclipse compliant equivalents
- Establish training schedules to augment necessary skill levels.

Agenda

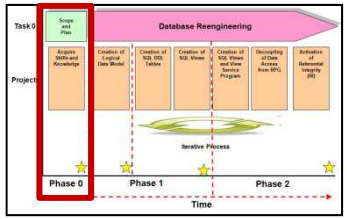
- Getting started
  - Why Data Centric, Why SQL?
  - Training, Teaming and Tools
- **THE PHASED APPROACH**
  - **DISCOVERY (PHASE 0)**
  - Migration (Phase 1)
  - Isolation (Phase 2)
  - Integrity (Phase 3)



Database Reengineering Sequence of Events



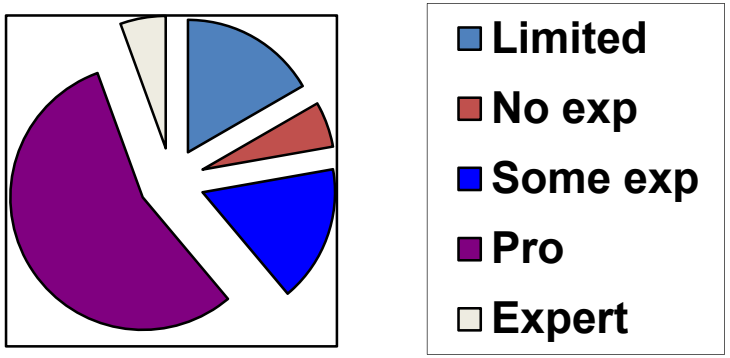
Phase 0 - Skills and Tool Inventory Assessments



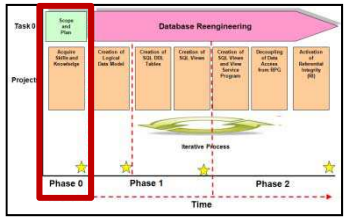
- A skills and inventory assessment must be done to identify current strengths and weaknesses
  - How much do we know?
  - What tools do we have?
    - Do we use them?
  - What do we need?
    - Education
    - Tools
    - Additional resources

Skill / Tool	NA- Not Applicable	1- Limited Knowledge	2- Acquired Knowledge, no practical experience	3- Acquired Knowledge w/ some practical experience	4- Applied Knowledge, multiple experience	5- Mastered, expert	Descriptions
Infosphere Data Architect							Reverse Engineering, Transforming Physical Data Models to Logical Da Models, Defining relationships, Generating DDL, Deploying DDL
Rational Team Concert (Jazz)							Establish team repositiories, check out, check in, deployment
Rational Open Access							Handler basics, using the QRN data stucture, developing generic handler
Rational Developer for Power							Remote System Explorer, IBM i Projects, importing and exporting, verfiication, snippets, templates, synchronizing
Rational Business Developer							Data Perspective, Data Projects, Da Source Explorer
DB2 Web Query							
Xcase							Implementation, reviewing diagnostic reverse engineering, surrogate file
System i Navigator							Basics, Databases, reverse engineering, IFS,

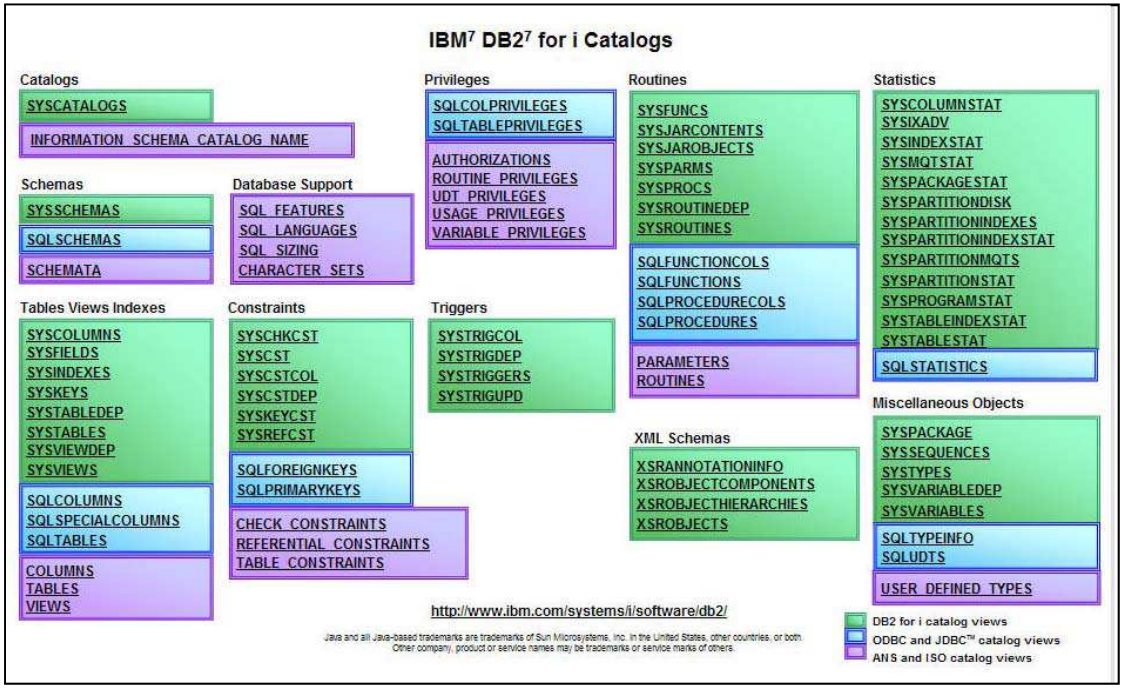
Rational Developer for Power



Phase 0 - Database Statistical Analysis



- The DB2 for i system catalogs provide the answers to the following questions:
  - Which files are in use?
  - Which files are core files? Work files?
  - Which files are DDS defined?
  - What are the relationships between core files?



Example Catalog Analysis Queries

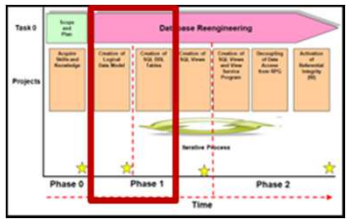
```
WITH Member_Counts AS (
SELECT table_schema, table_name, T1.*,
CASE Number_Partitions WHEN 1 THEN 1 ELSE 0 END
one_member,
CASE WHEN Number_Partitions BETWEEN 2 AND 255
THEN 1 ELSE 0 END many_members,
CASE WHEN Number_Partitions > 255 THEN 1 ELSE 0
END too_many_members
FROM QSYS2.SYSTABLESTAT T1
JOIN QSYS2.SYSTABLES T2
USING(table_schema, table_name)
where T2.file_type <> 'S')
SELECT Table_Schema, COUNT (DISTINCT
table_name) total_files,
SUM(one_member) AS One_Partition,
SUM(many_members) AS LT_256_Partitions,
SUM(too_many_members) AS Too_Many_Partitions
FROM Member_Counts
GROUP BY ROLLUP (Table_Schema);
```

```
-- Work files
SELECT table_type, Table_Schema,
COUNT (DISTINCT table_name) total_files,
max(number_rows) max_rows,  max(data_size) max_size,
SUM(OPEN_OPERATIONS) Opens,
SUM(cLEAR_OPERATIONS) Clears
FROM QSYS2.SYSTABLESTAT T1
JOIN QSYS2.SYSTABLES T2
USING(table_schema, table_name)
where T2.file_type <> 'S' AND clear_operations  > 0
GROUP BY ROLLUP (table_type, table_schema);
```

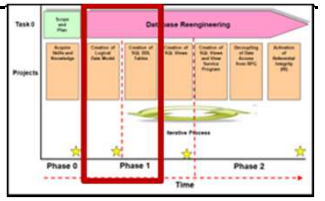
```
--Indexing
SELECT  table_type, Table_Schema, COUNT (DISTINCT
table_name) total_files, max(number_rows) max_rows,
max(data_size) max_size,
MAX(maintained_temporary_index_size) max_MTI_Size,
SUM(Index_builds) Ix_Builds
FROM QSYS2.SYSTABLESTAT T1
JOIN QSYS2.SYSTABLES T2
USING(table_schema, table_name)
where T2.file_type <> 'S'
GROUP BY ROLLUP (table_type, Table_Schema);
```

Agenda

- Getting started
  - Why Data Centric, Why SQL?
  - Training, Teaming and Tools
- **THE PHASED APPROACH**
  - Discovery (Phase 0)
  - **MIGRATION (PHASE 1)**
  - Isolation (Phase 2)
  - Integrity (Phase 3)

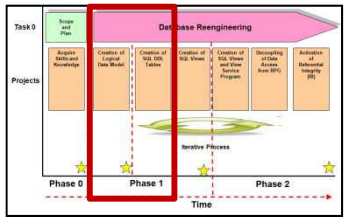


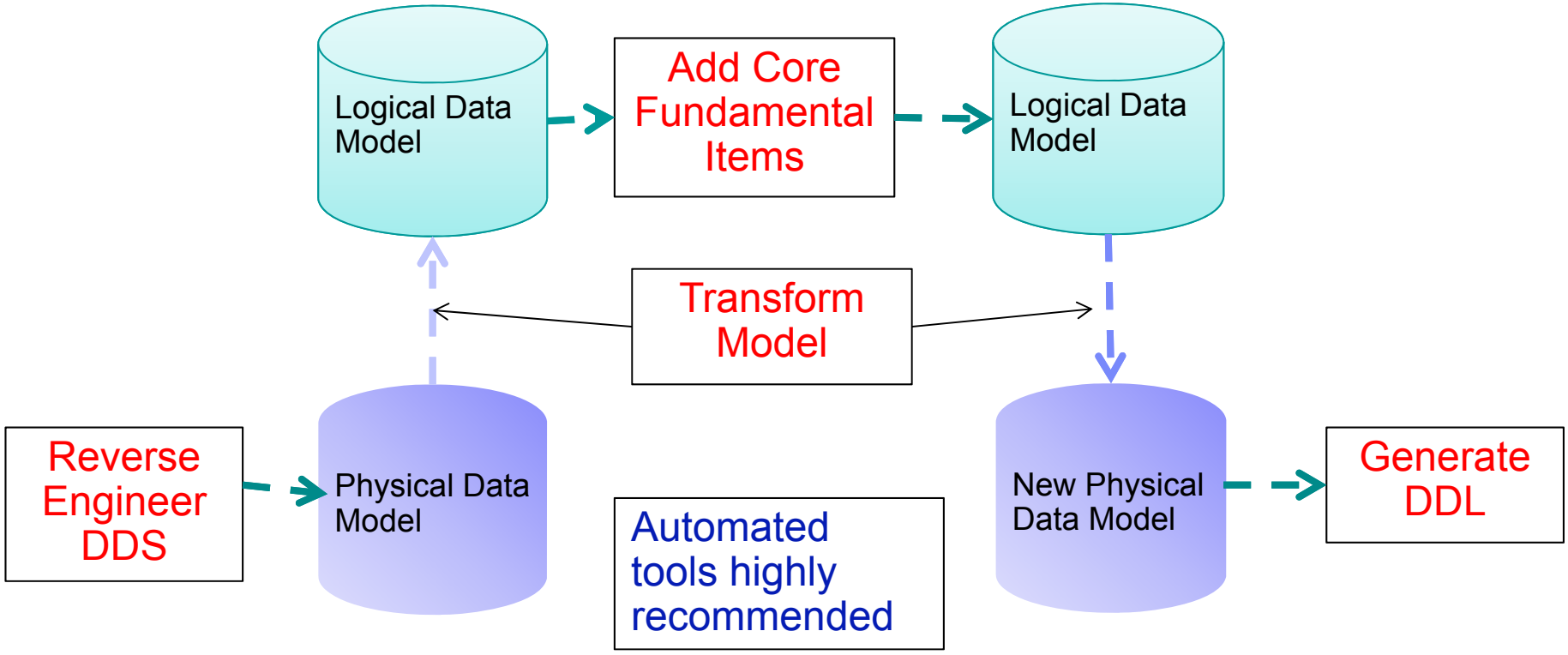
Database Core Fundamental Items



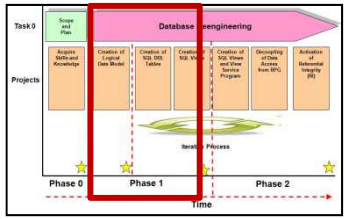
- The following core items must be present in the current physical data model:
  - All database objects are defined using SQL DDL.
  - All base tables (physical files) have a ***unique key***
  - Parent or master tables have a ***primary key***
  - Dependent files have ***foreign key constraints***
  - All ***key*** fields are named the same across tables
  - Columns (fields) are defined appropriately (i.e. date and time data is defined as date or time types, long text fields are defined as varchar, etc)
  - There is a minimal use of work files
  - Properly normalized database structures (3NF)
- Correcting the above issues is a critical success factor to addressing such requirements as **flexibility, agility and scalability**

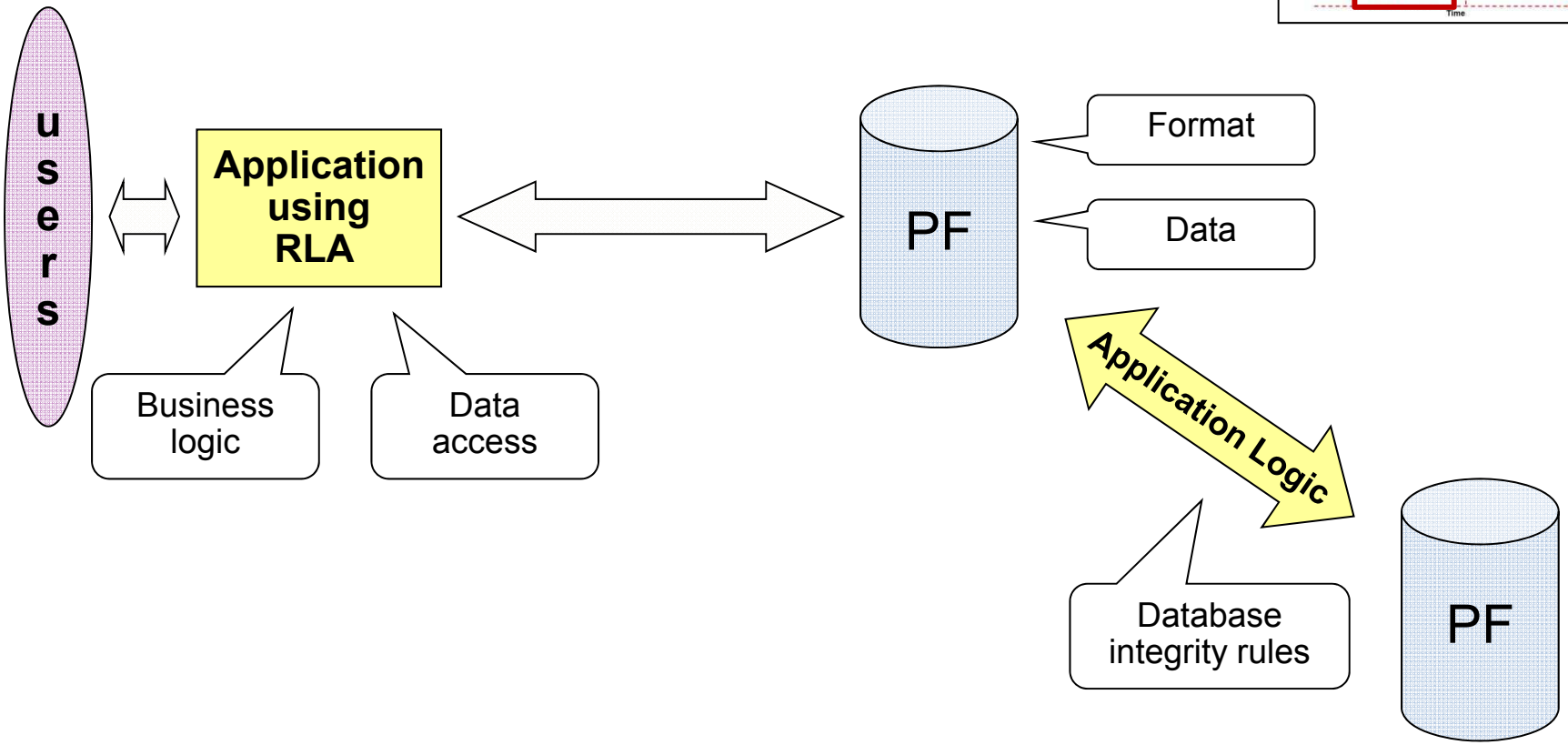
Creating an LDM from an Existing Physical Data Model



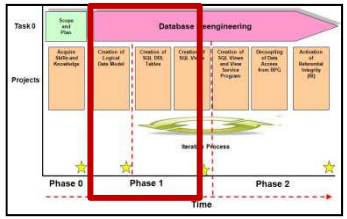


Phase 1 – Current state





Phase 1 Reverse Engineering and Transformation



users

Application using RLA

Business logic

Data access

LF

Original format

Data
New format

SQL PROC

SQL View

SQL Table

Application Logic

SQL Table

Initial modernization - Foundation for leveraging SQL

Database integrity rules

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*                          REF(FIELDREF) [4]
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**

Updated LFs Explicitly Sharing Surrogate LF Format

```
CRTLF CUSTMAST[2]
A*                          REF(FIELDREF)
A       R CUSTMASTR[2]   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
```

```
CRTLF CUSTMASTL1
A*      R CUSTMASTR   PFILE(CUSTMAST)
A       R CUSTMASTR[2]  PFILE(CUST_MAST[1])
A                       FORMAT(CUSTMAST[2])
A       K CUSTOMER

 CRTLF CUSTMASTL2
A*      R CUSTMASTR   PFILE(CUSTMAST)
A       R CUSTMASTR[2]   PFILE(CUST_MAST[1])
A                       FORMAT(CUSTMAST[2])
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 LF CUSTMAST

Updated LFs – No Explicit Format Sharing

```
CRTLF CUSTMAST[2]
A*                          REF(FIELDREF)
A       R CUSTMASTR[2]   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
```

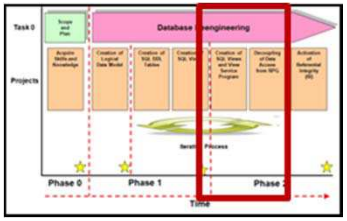
```
CRTLF CUSTMASTL1
A*      R CUSTMASTR   PFILE(CUSTMAST)
A       R CUSTMASTR[2]  PFILE(CUST_MAST[1])
A         CUSTKEY
A         CUSTOMER
A         ADDRESS
A         CITY
A         STATE
A         ZIPCODE
A         PHONE
A       K CUSTOMER
```

Notes

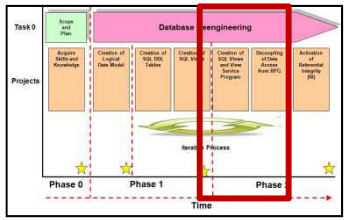
1. All DDS LFs are built over the new SQL table CUST_MAST
2. The format CUSTMASTR is still used for impact analysis
3. Format IDs are the same

Agenda

- Getting started
  - Why Data Centric, Why SQL?
  - Training, Teaming and Tools
- **THE PHASED APPROACH**
  - Discovery (Phase 0)
  - Migration (Phase 1)
  - **ISOLATION (PHASE 2)**
  - Integrity (Phase 3)



Phase 2 Isolation –View Access



users

Application using RLA

I/O Handlers

SQL View

New format

Business logic

Data access Commitment control

Data New format

SQL PROC

SQL View

SQL Table

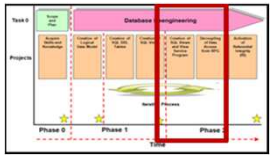
Application Logic

SQL Table

Initial modernization - Foundation for leveraging SQL

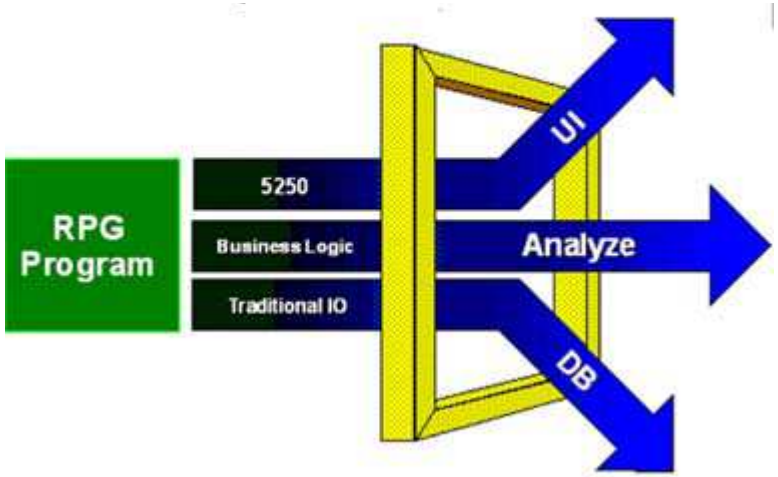
Database integrity rules

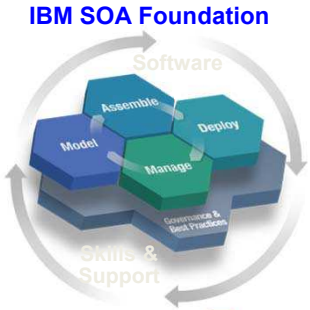
Handlers to the Rescue!!!



- **Handler transforms 5250 to any UI device**
  - **3rd party tools available**

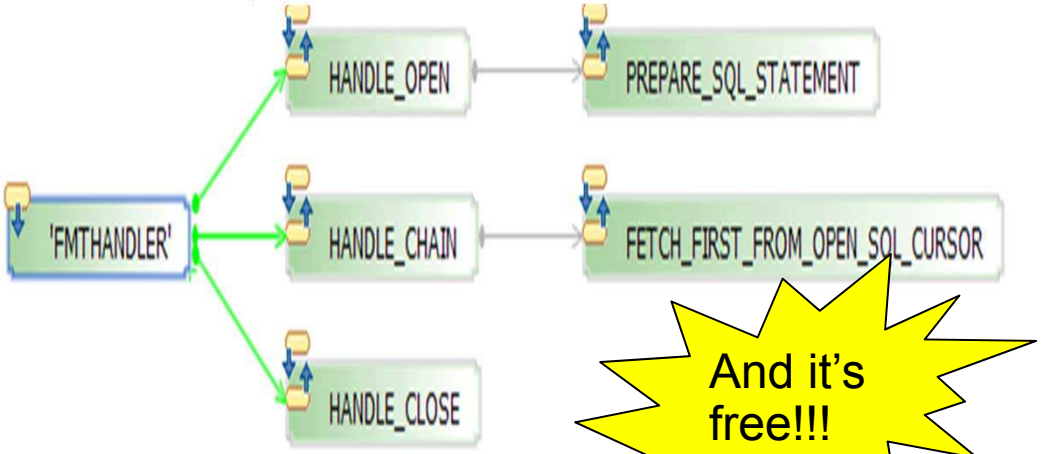
Handlers intercept traditional IO operations



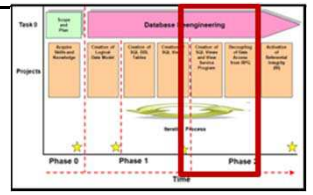


- **Minimal change to existing RPG programs**
  - **Create or acquire a handler**
  - **Add at least one line of code to an existing program**

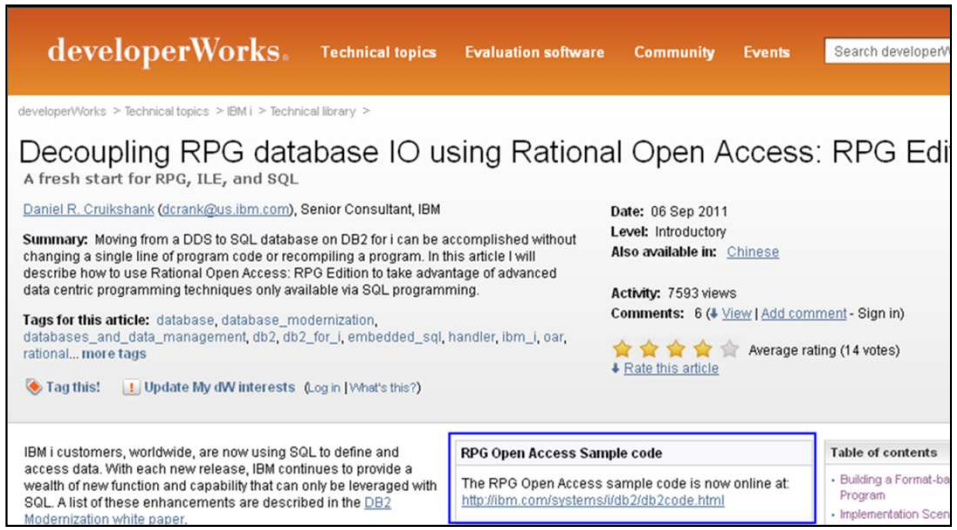
- **Handler transforms data access to SQL procedure calls**
  - **Utilize modern tools and advanced database feature and function**
- **Start with a few key programs**
  - **No need to change all files**



Getting Started with SQL Handlers



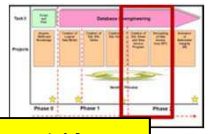
- Tools and resources are available to help build handlers
  - Visit the RPG Cafe for articles, tutorials, podcasts and forums about RPG, RDi and RTCi
  - Visit developerWorks and download a sample SQL handler



http://www.ibm.com/developerworks/ibmi/library/i-roaforsql/index.html?ca=drs-

www.ibm.com/rational/cafe

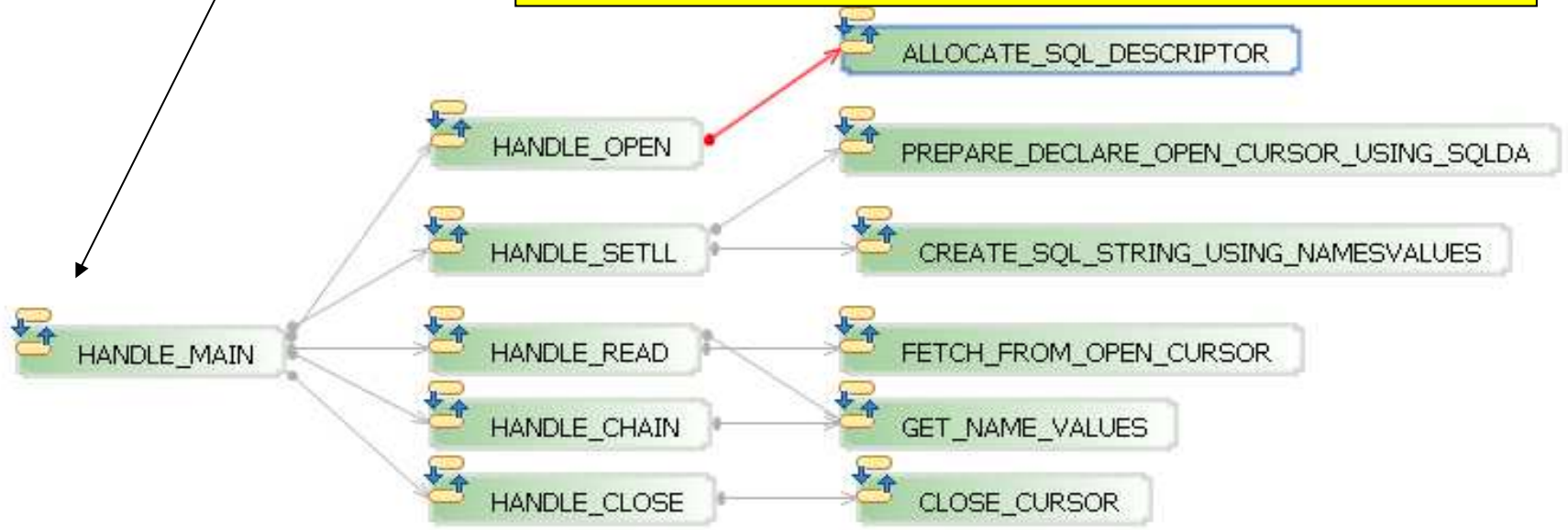
Handling RPG Using a Statement-based Handler



```
FDDSFILE1  UF   E     K DISK
F          handler('STMHANDLER')
```

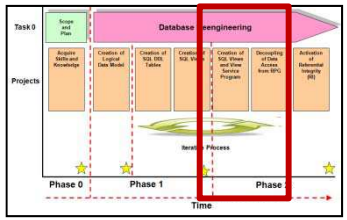
No external definitions required when using namesValues = '1'

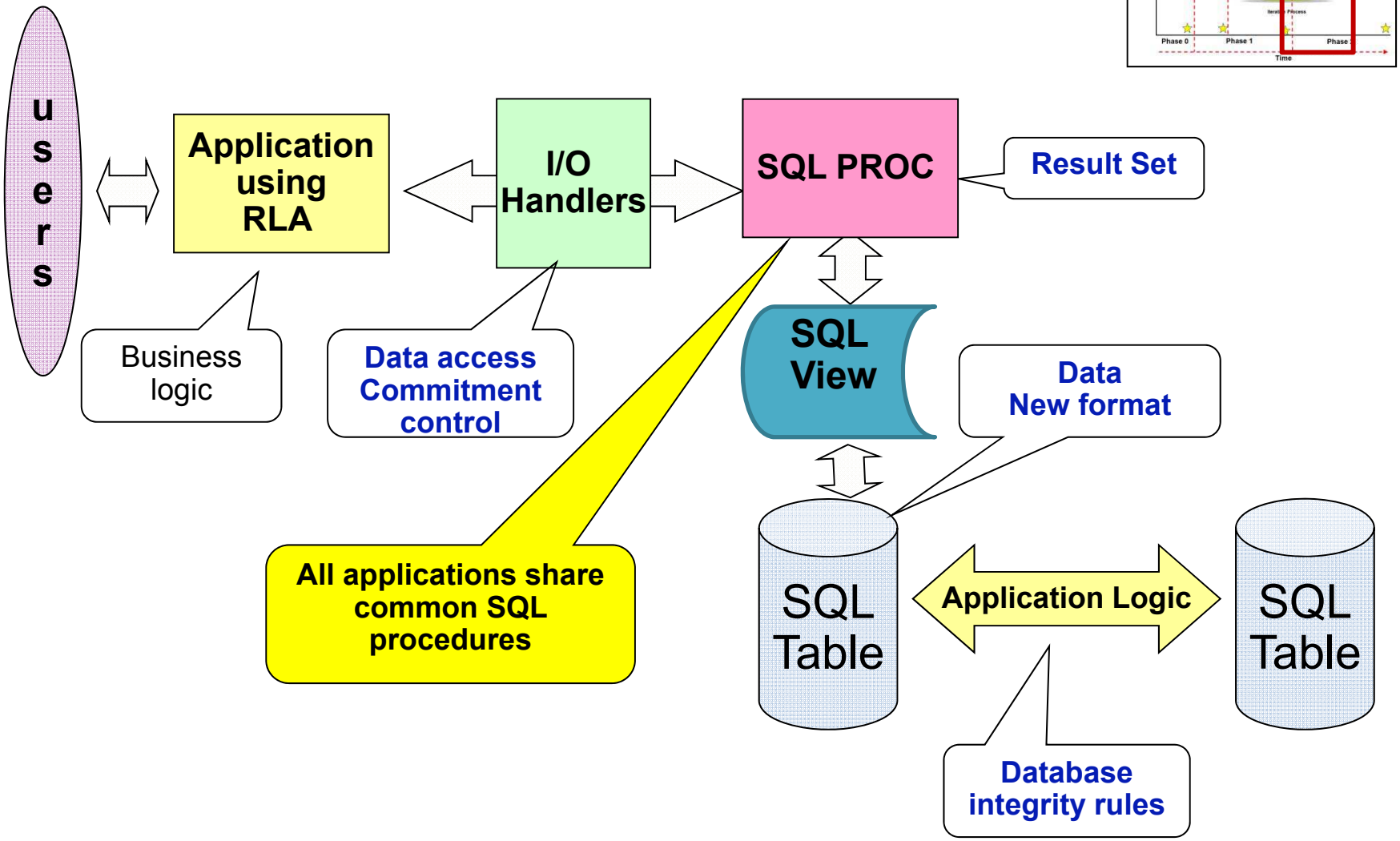
PREPARE INTO describes table in SQLDA



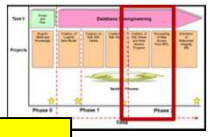

```
//Allocate descriptor. sqlN = nbr of cols from OA*max nbr of rows to fetch
SQLDA.sqln = rpgOA.flds.num * number of Rows;
SQLDA_p = %Alloc(((SQLDA.sqlN) * 80) + 16);
//PREPARE INTO same as PREPARE and DESCRIBE
EXEC SQL PREPARE SQL_Descriptor_S1 INTO :SQLDA FROM :SQLString;
//Varying list dynamic SQL uses row storage area associated with SQLDA
EXEC SQL FETCH NEXT FROM SQL_Statement_C1 FOR :numberOf ROWS
         USING DESCRIPTOR :sqlda INTO :rowStorageArea:sqlIndAry ;
```

Phase 2 Isolation – Shared Result Set



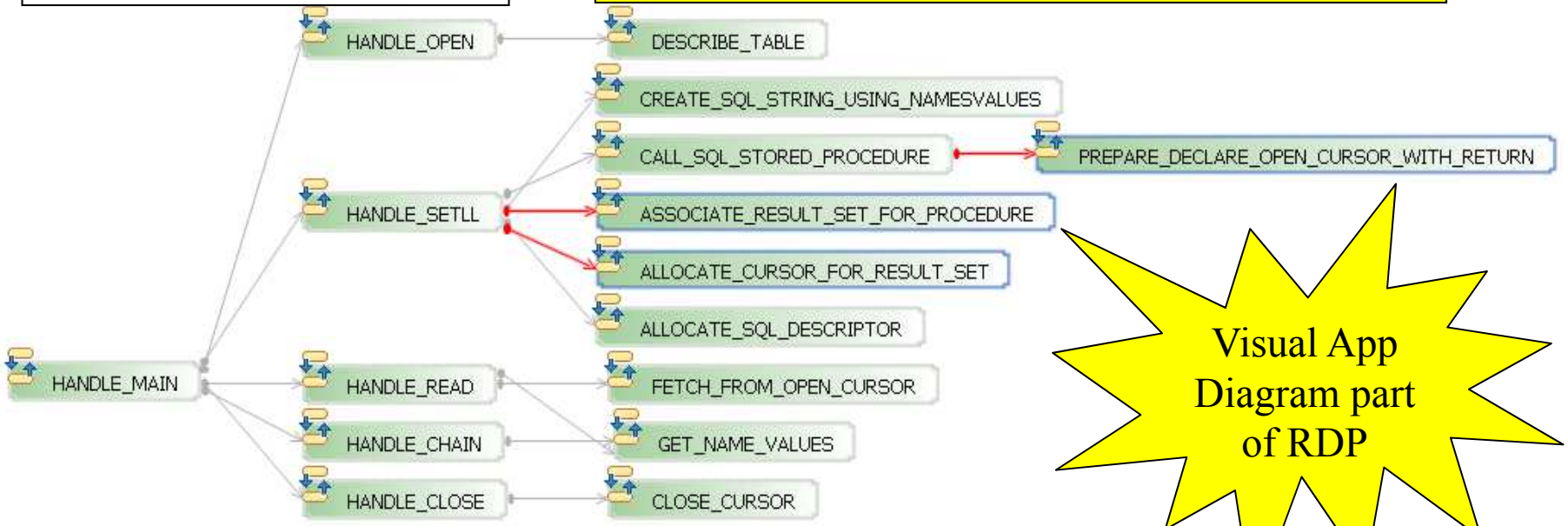


Handling RPG Using a Procedure-based Handler



```
FDDSFILE1  UF   E     K DISK
F          handler('PRCHANDLER')
```

DESCRIBE TABLE provides view format when PREPARE is external



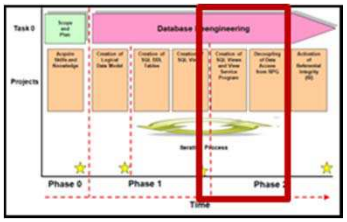
Visual App Diagram part of RDP

```
//Result set consumption (7.1)
EXEC SQL ASSOCIATE RESULT SET LOCATORS (:a) WITH PROCEDURE P1;
EXEC SQL ALLOCATE C1 CURSOR FOR RESULT SET :a;
//Get Name Values
For i = 1 to Flds.num; //Flds provided by RPG using namesValues
 // set the length and value information for this field
  Flds.field(i).valueLenBytes = sqlvar.sqlLen;
  Flds.field(i).value = sqlvar.sqldata;
EndFor;
```

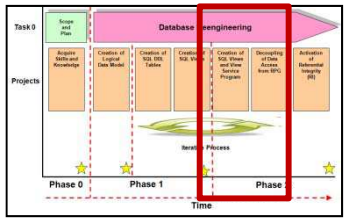
sqlvar part of SQL descriptor

Agenda

- Getting started
  - Why Data Centric, Why SQL?
  - Training, Teaming and Tools
- **THE PHASED APPROACH**
  - Discovery (Phase 0)
  - Migration (Phase 1)
  - Isolation (Phase 2)
  - **INTEGRITY (PHASE 3)**



Phase 3 Integrity



users

Application using RLA

I/O Handlers

SQL PROC

CRUD

Data New format

Business logic

Data access Commitment control

SQL Join View

Instead of Trigger

All applications share common SQL procedures

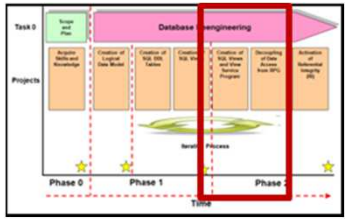
SQL Table

SQL Constraints

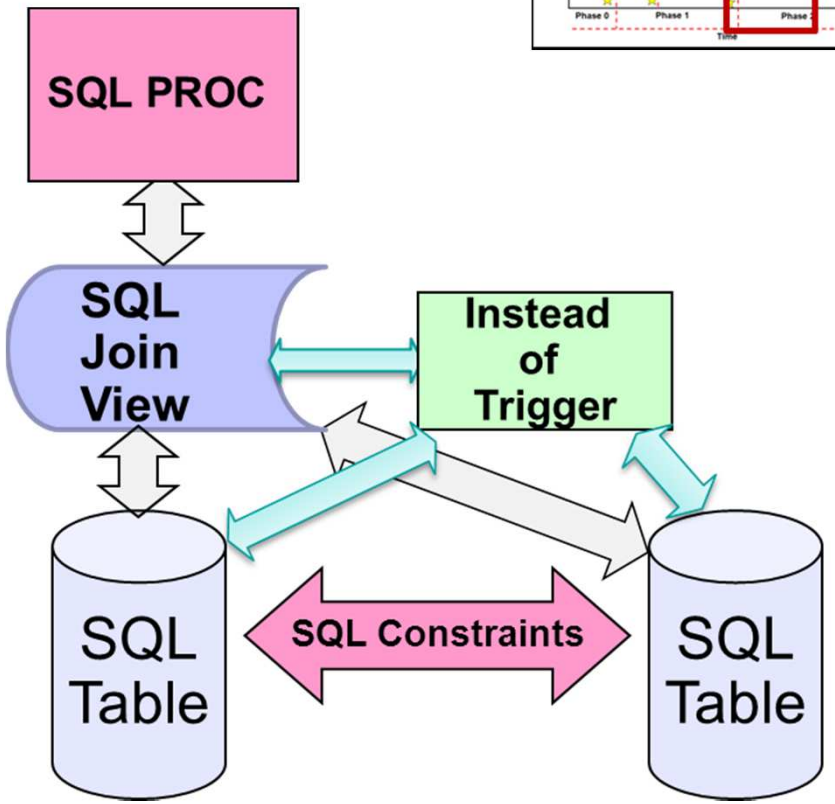
SQL Table

Database integrity rules

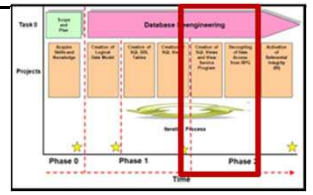
Instead of Triggers



- Enables updates/deletes or inserts for SQL views which are not considered updateable, delete-able or insert-able
  - A join would be one such view
- Allows database changes without impacting existing Stored Procedures
  - Insert/Update/Delete events are redirected to appropriate table
- Developer creates IOT logic



Instead of Triggers and Constraints



```
INSTEAD OF INSERT ON JOIN_V1
REFERENCING  NEW AS N FOR EACH ROW MODE
   DB2SQL
BEGIN
      INSERT INTO Parent
      (Col1, Col2…)
      VALUES (N.Val1, N.Val2…);
      INSERT INTO Child
       (Parent_ID, Col1a, Col2a…)
      VALUES (IDENTITY_VAL_LOCAL(), N.Val1a,
      N.Val2a…);
END;
```

- Parent contains PK as IDENTITY
- IDENTITY_VAL_LOCAL returns generated IDENTITY value from prior INSERT

```
INSTEAD OF DELETE ON JOIN_V1
REFERENCING OLD AS O FOR EACH ROW MODE
   DB2SQL
BEGIN
      DELETE FROM Parent
      WHERE Parent_Key= O.Parent_Key;
END
```

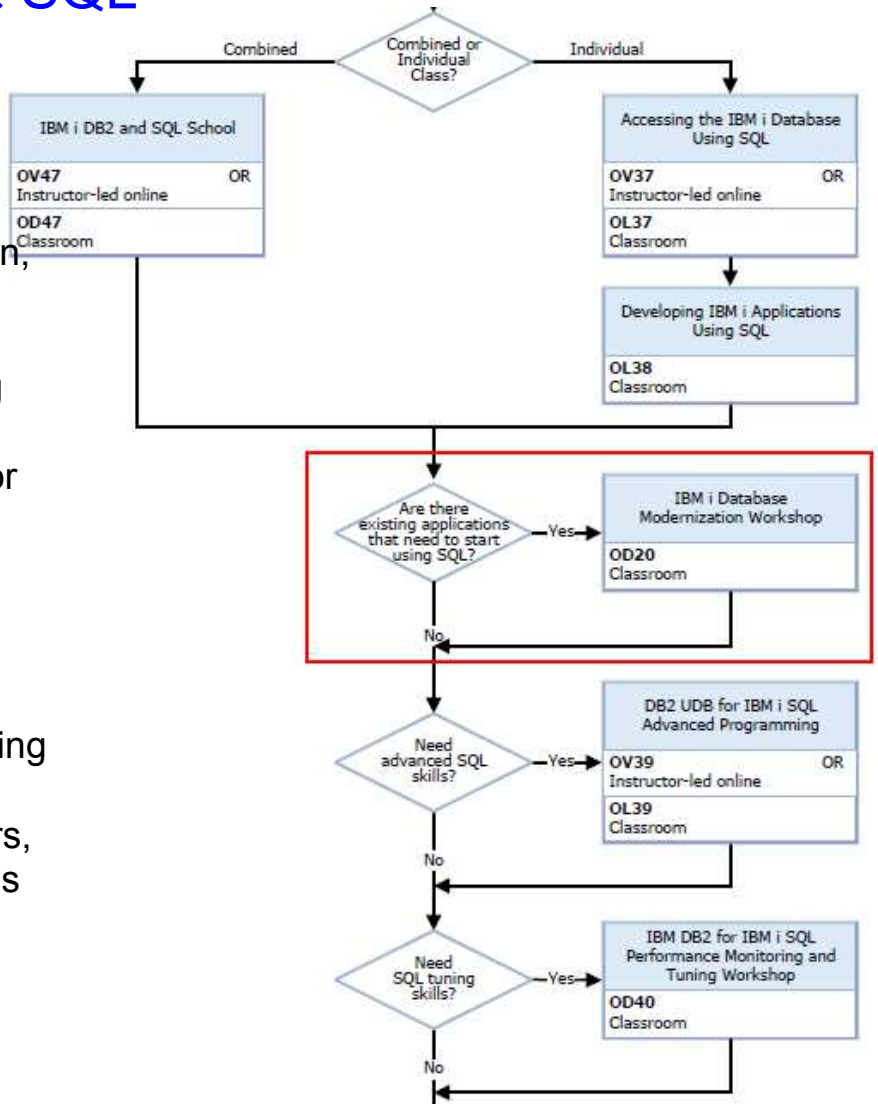
- **FOREIGN KEY constraint rule CASCADE**
- **Child rows deleted by DB2**

Summary

- Migrating legacy databases to modern DB2 databases requires planning and additional skill set
  - Understanding query optimization
  - Relational database design knowledge
  - Acquire a good tool
- You should have good business reasons for migrating
  - Need for enhanced feature and function, lack of skills in marketplace
- Start small, get some experience
  - Identify a pilot application which would benefit from modernization
- Performance improvement is a benefit, not the main reason for reengineering
  - The amount of performance gain will be determined by the level of data centric development
    - DDL and DML

IBM Education Roadmap for DB2 & SQL

- **Database Modernization Workshop**
  - Understand the reasons for DB reengineering.
  - Understand the benefits of using SQL.
  - Understand RDMS concepts such as normalization, referential integrity, etc.
  - Learn the methodology and steps involved for reengineering legacy databases without impacting the legacy programs
  - Understand the methodology and tools required for identifying high usage files
  - Learn conversion of Record level access to SQL access through the use of embedded SQL and external SQL Views
  - Understand the concepts of using bridge and wrapper programs as productivity tools for accessing SQL IO modules including RPG OA
  - Understand the use of Triggers, Instead of Triggers, User Defined Functions, and Stored Procedures as database modernization techniques
  - Labs are based on Rational Developer for Power (RDP) and IBM Data Studio
  - Source is provided



Where to Start

- Review the following Redbook:
  - Modernizing iSeries Application Data Access (SG24-6393)
- You can find more information about Rational Open Access at the following websites:
  - http://www.ibm.com/software/rational/products/openaccess/
  - http://publib.boulder.ibm.com/infocenter/iseries/v7r1m0/index.jsp and then expand ibm i 7.1 Information Center->Programming->Programming Languages->RPG
- To learn more about the Rational Developer for Power Systems tool used to create the Visualizer Application Diagrams within this presentation, and to download an evaluation copy, visit the developerWorks Rational Tools download website at: http://www.ibm.com/developerworks/downloads/r/rdp/
- To get the most up to date, in depth knowledge of the DB2 for i Data Access modernization strategy, including hands on experience with Rational Open Access, Rational Developer for Power and Infosphere Data Architect, enroll in the IBM i Database Modernization Workshop at the following website: http://www.ibm.com/systems/i/software/db2/db2educ_m.html and choose DB2 for i Modernization Workshop from the list of courses.

Additional Information

- DB2 for i Websites
  - Home Page: **ibm.com/systems/i/db2**
  - DeveloperWorks Zone: ibm.com/developerworks/db2/products/db2i5OS
  - Porting Zone: ibm.com/partnerworld/i/db2porting
- Newsgroups & Forums
  - USENET: comp.sys.ibm.as400.misc, comp.databases.ibm-db2
  - DevloperWorks: https://www.ibm.com/developerworks/forums/forum.jspa?forumID=292
  - System i Network DB2 Forum: http://forums.systeminetwork.com/isnetforums/
- Education Resources - Classroom & Online
  - ibm.com/systemi/db2/gettingstarted.html
  - ibm.com/partnerworld/wps/training/i5os/courses
- DB2 for i Publications
  - White Papers: ibm.com/partnerworld/wps/whitepaper/i5os
  - Online Manuals: ibm.com/systems/i/db2/books.html
  - DB2 for i Redbooks (http://ibm.com/redbooks)
    - Getting Started with DB2 Web Query for System i (SG24-7214)
    - OnDemand SQL Performance Analysis … in V5R4 (SG24-7326)
    - Preparing for and Tuning the SQL Query Engine on DB2 for i5/OS (SG24-6598)