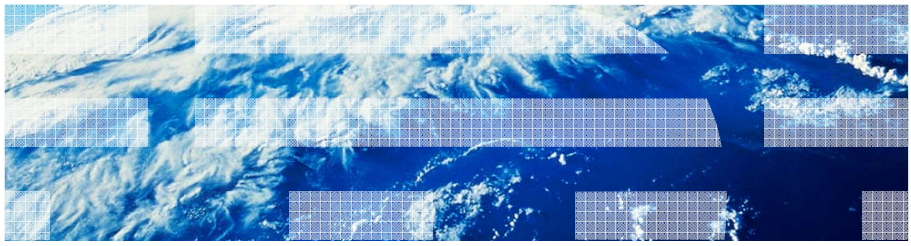
Building a Solid Database Foundation Using SQL DDL

Dan Cruikshank – dcrank@us.ibm.com



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

Agenda

- The Core Fundamental Items
- Database Modeling
- DB2 for i Data Centric Constructs
- Reengineering Existing Databases using the DB2 for i Modernization Strategy

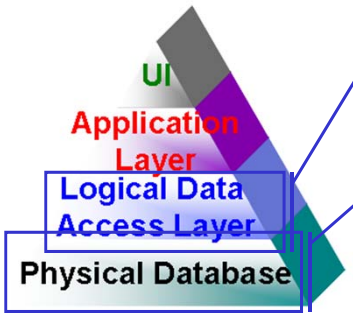
The Core Fundamental Items

Topics

- Core Foundational Layers
- Characteristics of Tables Meeting Fundamental Core Compliance
- Differences between Primary and Unique Key Constraints
- Characteristics of Columns Meeting Fundamental Core Compliance

Core Foundational Layers

- Application Layer
  - Modular, reusable, parameter driven
- Presentation or User Interface (UI)
  - Constantly changing
  - Must be easily replaceable



- Data Centric Design
  - Logical Data Access Layer
    - Virtual representation of the physical database
    - Allows the physical database layer to change without impacting existing applications
  - Physical Database Layer
    - Defined using SQL
    - Most critical foundation
    - Base for other foundations

Characteristics of Tables Meeting Fundamental Core Compliance

- Core requirements
  - The table is defined using SQL Data Definition Language (DDL)
  - The table contains a **unique key constraint**
  - If the table is the parent in a parent-child relationship then it contains a **primary key constraint**
  - The table contains data change timestamp columns
    - One for when a row was created
    - One for when the row was last changed
- Secondary requirements
  - The table contains a single partition (member) if:
    - The total number of rows are less than 4.2 billion and
    - The total size of the partition is less than 1.7TB
  - The table is not regularly cleared, reorganized or copied
  - The table contains a column representing the last user making a change to this table

Differences between Primary and Unique Key Constraints

The following table compares the recommended characteristics of a primary key to a unique key:

Characteristic	*Primary Key*	*Unique Key*
More than one per base table	No	Yes
Can be null	No	Yes
Must be a constraint	Yes	No
Can be known to the outside world	No	Yes
Contains meaningful data	No	Maybe
Needs to encrypted	No	Yes, if meaningful data is sensitive
Can be used in a referential constraint	Yes	Yes, however this should be avoided if data is sensitive
Should be auto generated	Yes	Not always

Characteristics of Columns Meeting Fundamental Core Compliance

- Strong typing
  - Numeric (INTEGER, DECIMAL, NUMERIC, etc)
    - validation at insert
    - Integer and unsigned numeric best for performance
  - Date, Time, Time Stamp
    - Dates cannot be zero, blank or all 9's
- Auto generated columns
  - Identity column
  - ROW CHANGE TIMESTAMP
  - ROW ID
  - SEQUENCE OBJECT
- Capacity to grow
  - VarChar
    - Varying length columns grow with the business
  - Largest BIGINT value = +9 223 372 036 854 775 807
  - Largest decimal precision = 63
- Protection
  - Encryption
  - Implicitly Hidden

SQL Views and DDS LFs can redefine columns for presentation

Database Modeling

Topics

- Data Modeling Definition
- Types of Models
- Building a Logical Data Model
- Building the Physical Data Model

Data Modeling Definition

- A method used to define and analyze data requirements needed to support the business processes of an organization
- Is used to communicate the business rules and processes
- It is the process of creating a blueprint to visually represent data, its organization and the relationships between structures

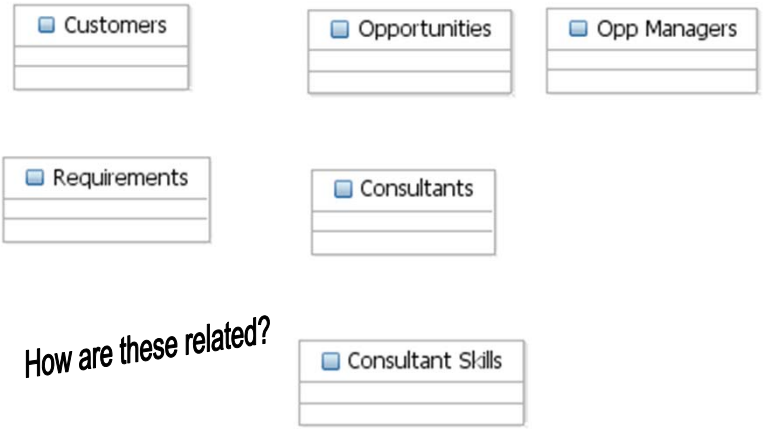
Types of Models

- Conceptual Data Model
  - Describes data from a high level
  - Includes entities and their relationships
  - Typically developed first
- Data Dictionary Models
  - Domain Model
    - collection of named data types for consistency and reuse
  - Glossary Model
    - Collection of column names and abbreviations
- Logical Data Model
  - Describes business information and defines business rules
  - Provides attributes
  - Typically developed after data dictionary
- Physical Data Model
  - Describes the implementation in a database
  - Is the blueprint for the database implementation
  - Typically developed after LDM

Conceptual Data Model

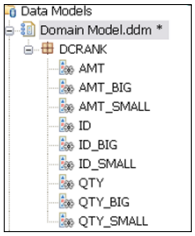
What are the business entities?

What are the attributes?

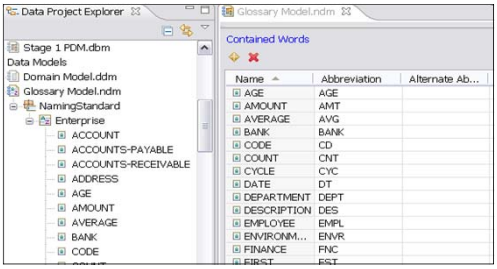


Data Dictionary Models

- Domain Model
  - Abstract data type
  - Based on base data type
    - Integer, decimal, date, etc.
  - Typically used for standardization of data types

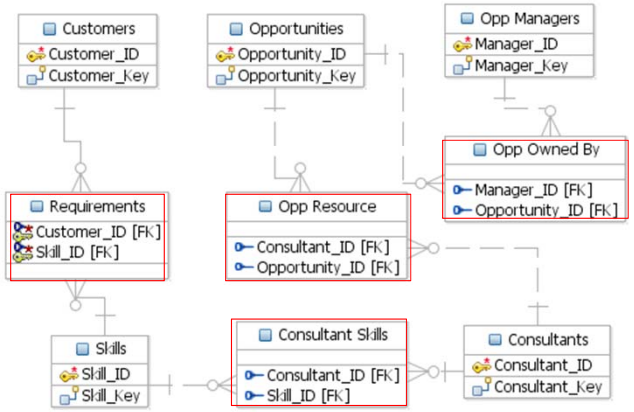


- Glossary Model
  - Typically used for standardization of data names and abbreviations

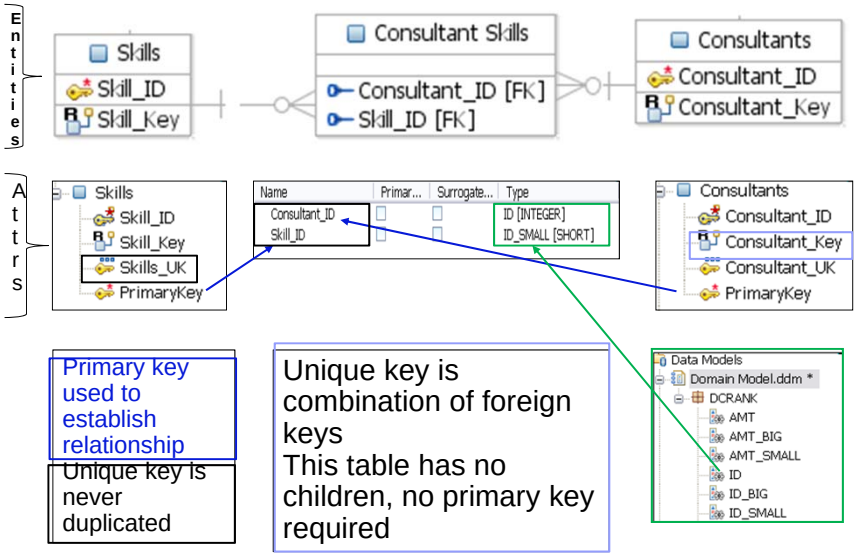


Building a Logical Data Model Step 1 (ERD)

- Entity Relationship Diagram (ERD)
  - First step in LDM creation
- Focus on relationships
  - Attributes come later
- Normalize to nth degree
  - Consider impact of change
  - E.g. What happens if manager leaves or consultant retires?



Entities and Key Attributes



Primary key used to establish relationship

Unique key is never duplicated

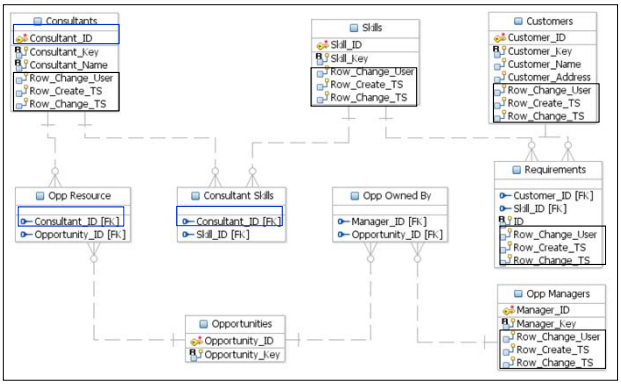
Unique key is combination of foreign keys
This table has no children, no primary key required

Logical Data Model with Column Attributes Step 2

- Ensure that only **primary key** columns are duplicated
- Model is database agnostic
- Some columns will have to be tuned in Physical Data Model
  - **Row change timestamps**, VARCHAR(ALLOCATE), etc.

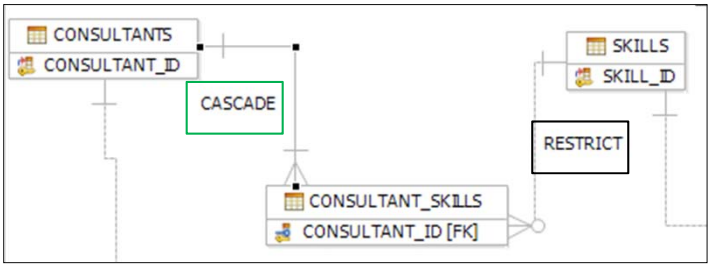
Primary Key

Data Centric columns

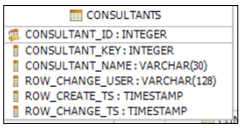


Physical Data Model and Referential Integrity

- Referential Integrity is a powerful data centric development tool
- **RESTRICT** - Eliminates the need for existence checking
  - DB2 performs check based on RI constraint
  - E.g. do not delete skill if foreign key exists
- **CASCADE** - Eliminates need to delete dependents using loop technique
  - E.g. Delete all consultant skills when consultant deleted



Physical Data Model and Deployment



CONSULTANTS

Name	Primar...	Domain	Data Type	Length	Scale	Not Null	Genera...	Default Value/Genera...
CONSULTANT_ID	☑	ID [INTEGER]	INTEGER			☑	☑	As Identity with Speci...
CONSULTANT_KEY	☐	ID [INTEGER]	INTEGER			☑	☐	
CONSULTANT_NAME	☐	Name [VARCHAR(30)]	VARCHAR	30		☑	☐	
ROW_CHANGE_USER	☐		VARCHAR	128		☐	☐	USER
ROW_CREATE_TS	☐		TIMESTAMP			☐	☐	CURRENT_TIMESTAMP
ROW_CHANGE_TS	☐		TIMESTAMP			☑	☐	

```
CREATE TABLE CONSULTANTS (
CONSULTANT_ID INTEGER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY IMPLICITLY HIDDEN,
CONSULTANT_KEY INTEGER NOT NULL UNIQUE,
CONSULTANT_NAME VARCHAR(30) NOT NULL,
ROW_CHANGE_USER VARCHAR(128) ALLOCATE(10) DEFAULT USER IMPLICITLY HIDDEN,
ROW_CREATE_TS TIMESTAMP DEFAULT CURRENT_TIMESTAMP IMPLICITLY HIDDEN,
Consultants_RCTS NOT NULL GENERATED BY DEFAULT FOR EACH ROW ON UPDATE AS ROW CHANGE
TIMESTAMP  IMPLICITLY HIDDEN);
```


Data Modeling Tools

- Characteristics of a good data modeling tool:
  - Provides Data Dictionary, Logical and Physical Data Modeling
  - Generates DDL
  - Imports DDL (DDS is icing on the cake)
  - Analyzes model (normalization, domain and glossary validation)

InfoSphere Data Architect	IBM Corporation
Xcase	Resolution Software LTD
ERwin	Computer Associates
ER/Studio	Embarcadero Technologies
PowerDesigner	Sybase Corporation

Best Practices for Relational Database Design

- Use only one key column to represent the relationship between any two tables (Primary Key)
- Key columns should be of the same type and have the same attributes (i.e. type, length, precision, scale)
  - Primary-Foreign key columns should have the same name
- Meaningless keys are acceptable and encouraged (identity column)
- Define and use constraints (RI, UNIQUE, CHECK)
- Define and implement views to assist the programmers and users

DB2 for i Data Centric Constructs

Topics

- Constraints
- Auto-generated Columns
- Partitioned Tables

Constraints

- Database constraints define business rules
- DB2 provides methods to enforce the rules
  - Indexes are created to support the enforcement
- Constraints can assist the query optimizer and DB engine
  - Rules enforced by the DB2 provide guarantees
  - Rules enforced by programs do not
- Example of data centric programming to minimize coding
  - Let the DB2 server do the work!

✓ Unique key constraint
✓ Primary key constraint
✓ Referential constraint
✓ Check constraint

Constraints

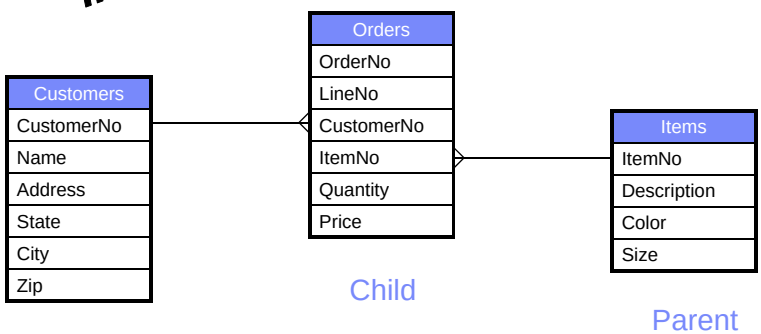
- Unique key constraint
  - the rule that the values of the key are valid only if they are unique
- Primary key constraint
  - the rule that the values of the key are valid only if they are unique and not null-able
- Referential constraint
  - the rule that the values of the "foreign key" are valid only if one of the following conditions is met:
    - foreign key appear as values of a parent key
    - some component of 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.

Constraints Example

How might they work?

- ✓Unique key constraint
- ✓Primary key constraint
- ✓Referential constraint
- ✓Check constraint



Parent

Child

Parent

A constraint is a rule enforced by the database manager to limit the values that can be inserted, deleted, or updated in a table.

Auto Generated and Implicitly Hidden Columns

- Automatically generating values for columns
- Auto Generation Considerations
- Implicitly Hidden Columns

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

Examples

- Example of defining an identity column

```
CREATE TABLE ORDERS
   (ORDER_ID BIGINT NOT NULL
    GENERATED ALWAYS AS IDENTITY PRIMARY KEY,…
```

- Example of defining a row change timestamp

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

- Example of defining a row id column

```
CREATE TABLE ORDERS
   (ORDER_ID BIGINT NOT NULL
   GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
   ORDER_ROW_ID ROWID,…
```

- Defining and using a sequence

```
CREATE SEQUENCE ORDER_LINE_SEQ;

CREATE TABLE ORDER_DETAILS
   (ORDER_ID BIGINT,
    ORDER_LINE_SEQ INTEGER,…);

SELECT * FROM FINAL TABLE (
INSERT INTO ORDER_DETAILS (ORDER_ID, ORDER_LINE_SEQ)
VALUES (1, NEXT VALUE FOR ORDER_LINE_SEQ),
       (1, NEXT VALUE FOR ORDER_LINE_SEQ));
```

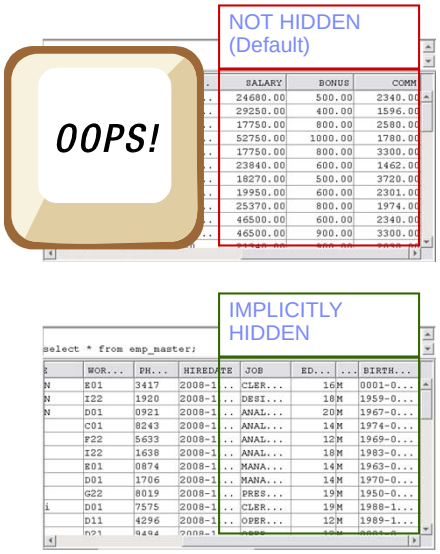
ORDER ID	ORDER LINE SEQ	ORDER DETAIL CHANGE AUDIT
1	1	13:25:39
1	2	13:25:39

GENERATED Considerations

- 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
- Table definition can be overridden
  - OVERRIDING SYSTEM VALUE uses the value in the data
  - OVERRIDING USER VALUE uses the system generated value
- Generated by default is better suited for data propagation, in which the contents of an existing table are copied, or for the unloading and reloading of a table.

Hiding Columns

- IMPLICITLY HIDDEN
  - Specified on CREATE TABLE or ALTER TABLE ALTER COLUMN statements
  - Column is not visible in SQL * statements unless it is referred to explicitly by name.
  - Attribute ignored when table accessed directly using traditional IO
- NOT HIDDEN (Default)
  - Column is included in SQL * statements.
- The result sets on the right show the differences between HIDDEN and NOT HIDDEN

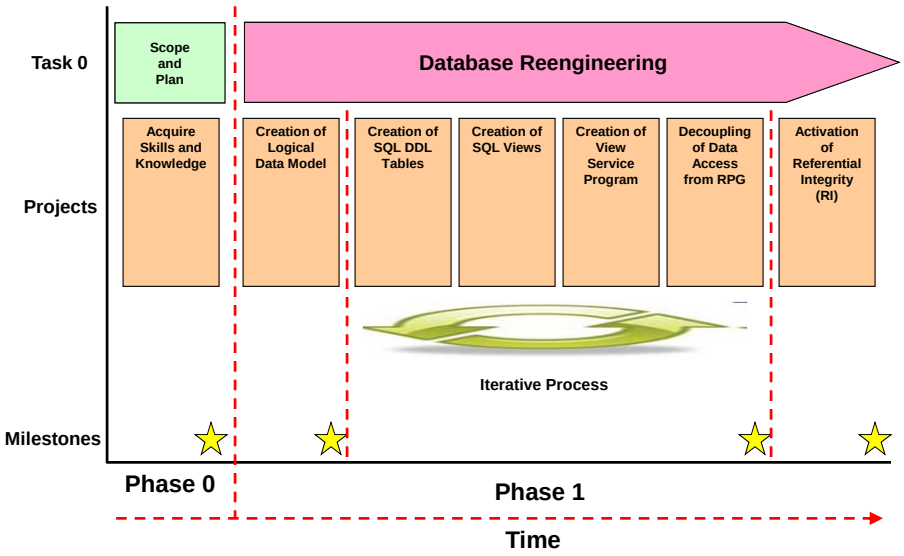


Reengineering Existing Databases

Topics

- Database Reengineering Sequence of Events
- Phase 0
  - Skills and Tool Inventory Assessments
  - Database Statistical Analysis
- Phase 1
  - Overview of the Database Reengineering Stages

Database Reengineering Sequence of Events

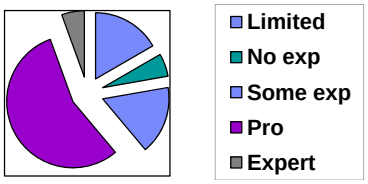


Phase 0 - Skills and Tool Inventory Assessments

- 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

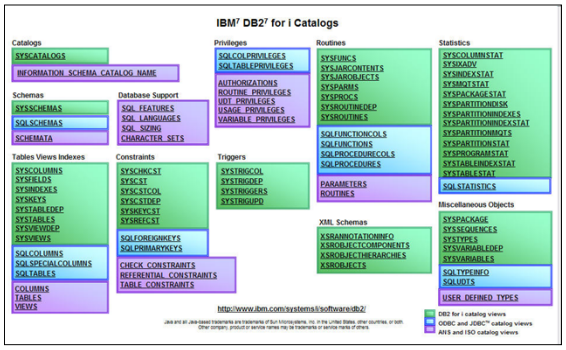
Skill / Tool	Description
Infosphere Data Architect	Reverse Engineering, Transforming Physical Data Models to Logical Data Models, Defining relationships, Generating DDL, Deploying DDL
Rational Team Concert (Jazz)	Establish team repositories, check out, check in, deployment
Rational Open Access	Handler basics, using the QRN data structure, developing generic handler
Rational Developer for Power	Remote System Explorer, IBM i Projects, importing and exporting, verification, snippets, templates, synchronizing
Rational Business Developer	Data Perspective, Data Projects, Data Source Explorer
IBM Web Query	
Ecase	Implementation, reviewing diagnostic, reverse engineering, surrogate file
System i Navigator	Basics, Databases, reverse engineering, IFS

Rational Developer for Power



Phase 0 - Database Statistical Analysis

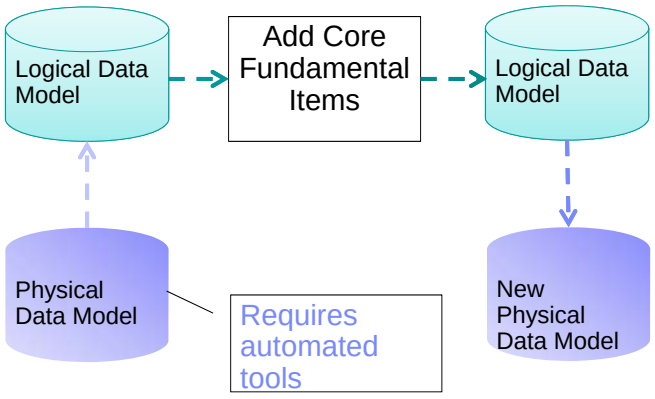
- The DB2 for i system catalogs provide the answers to the following questions:
  - Which files are in use?
  - Which files are core files? Work files?
  - Which files are DDS defined?
  - What are the relationships between core files?



Phase 1 - Overview of the Database Reengineering Stages

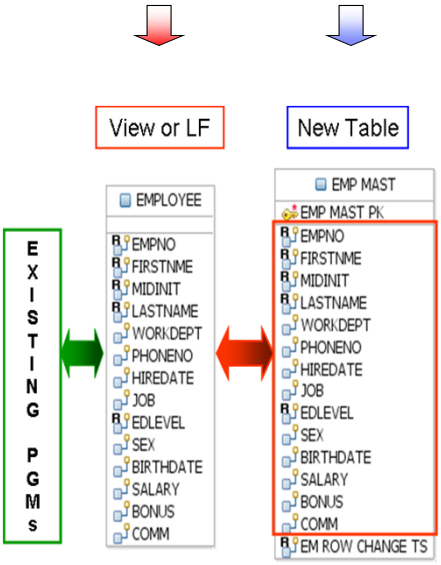
- Reconstruction
  - ***Reverse engineering*** the DDS defined database objects to a Logical Data Model, adding ***core fundamental items*** and then deploying the database using SQL constructs while retaining the original format.
- Isolation
  - Accessing the new SQL defined database via SQL views and IO Data Access modules utilizing embedded SQL
- Correction and Exploitation
  - Restructure the database to take advantage of advanced and new functions and features
  - Secure the new SQL defined database from unauthorized access
  - Enhance the new SQL defined database with advanced capabilities

Creating an LDM from an Existing Physical Data Model



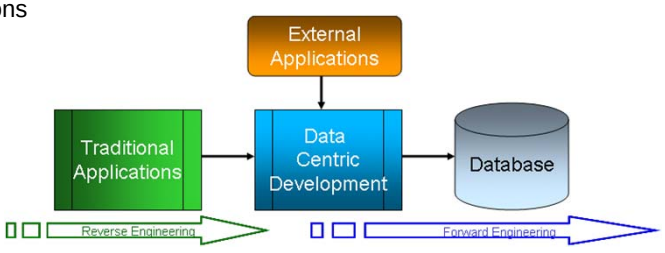
Reconstructing the Database

- Goal
  - ***Reverse engineer*** existing DDS DB objects to SQL DDL DB objects with no impact on existing applications
- Strategy
  - Identify key files and relationships
  - Replace DDS PF with SQL PF but retain original record format ID
  - Add missing attributes to SQL table
- Benefits
  - No program changes required
  - Existing programs continue to work



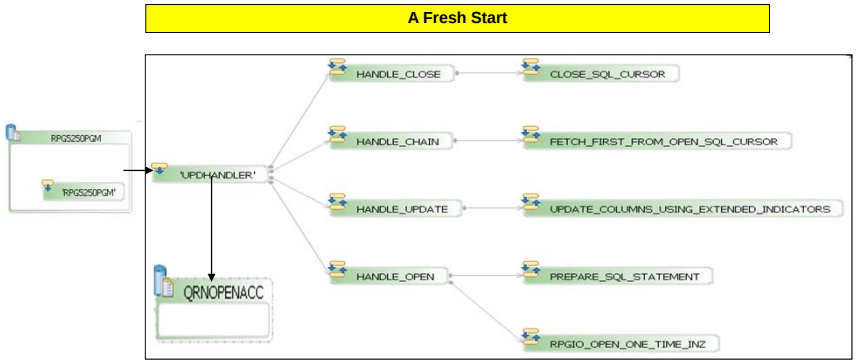
Isolating the Database

- Goal
  - Minimize the impact of change
- Strategy
  - Access the database using **SQL Views** (Virtual tables),
  - Common service programs using **embedded SQL** and
  - **Bridging** to these new services from existing applications
- Benefits
  - Reuse existing components
  - Minor changes to existing programs
    - **No SQL rewrite required**
  - Exploit SQL Data Centric Programming concepts
  - Common interface for both traditional and new development



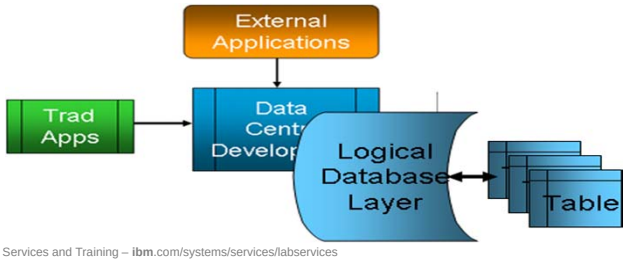
Bridging to SQL using Rational Open Access: RPG Edition

- Rational Open Access
  - Handlers intercept traditional IO operations
  - Minimal change to existing RPG programs
- Handlers:
  - Transform data access to SQL
    - Utilize advanced database feature and function



Correcting and Exploiting the Database

- Goal
  - Minimize the impact of adding new feature and function
- Strategy
  - Enable Data Centric Programming constructs
- Benefits
  - Normalized database
    - Eliminate redundancy, exposures
    - Reduced index maintenance, journal overhead



Summary and Questions

Summary

- Building a solid database foundation begins with data modeling
- Following good relational database design concepts allows agility, flexibility and scalability
- Using data centric development constructs key to minimizing impact of change
- Database modernization does not require extensive database and application modification

DB2 for IBM i Consulting and Services

- ✓ Database modernization
- ✓ DB2 Web Query
- ✓ Database design, features and functions
- ✓ DB2 SQL performance analysis and tuning
- ✓ Data warehousing review and assessment
- ✓ DB2 for IBM i education and training

Contact: Dan Cruikshank dcrank@us.ibm.com
IBM Systems and Technology Group
Rochester, MN USA