CA IDMS™ Modernization Best Practices: A Detailed Look

IUA Workshop

October 2007

Transforming IT Management

Abstract

This session expands upon the topics described in session "CA IDMS™ Database Modernization Techniques" by focusing on field experiences at CA IDMS customers, with particular emphasis on how to use views, table procedures and called procedures effectively in CA IDMS modernization projects. Issues encountered and approaches evaluated will be discussed, along with suggestions on Best Practices for CA IDMS modernization based on our practical customer experiences.

Agenda

> How do you maximize business value?

> Modernizing CA IDMS™

> Network to relational mapping

> Techniques

How Do You Maximize Business Value?

> Decisions, decisions
  - How to SOA?
  - Convert?
  - Modernize?

> Conversion
  - Costly
  - Risky

> You can **modernize** CA IDMS ...

... at a fraction of the cost of conversion

Modernizing CA IDMS

Modernizing CA IDMS

How You Can Do It

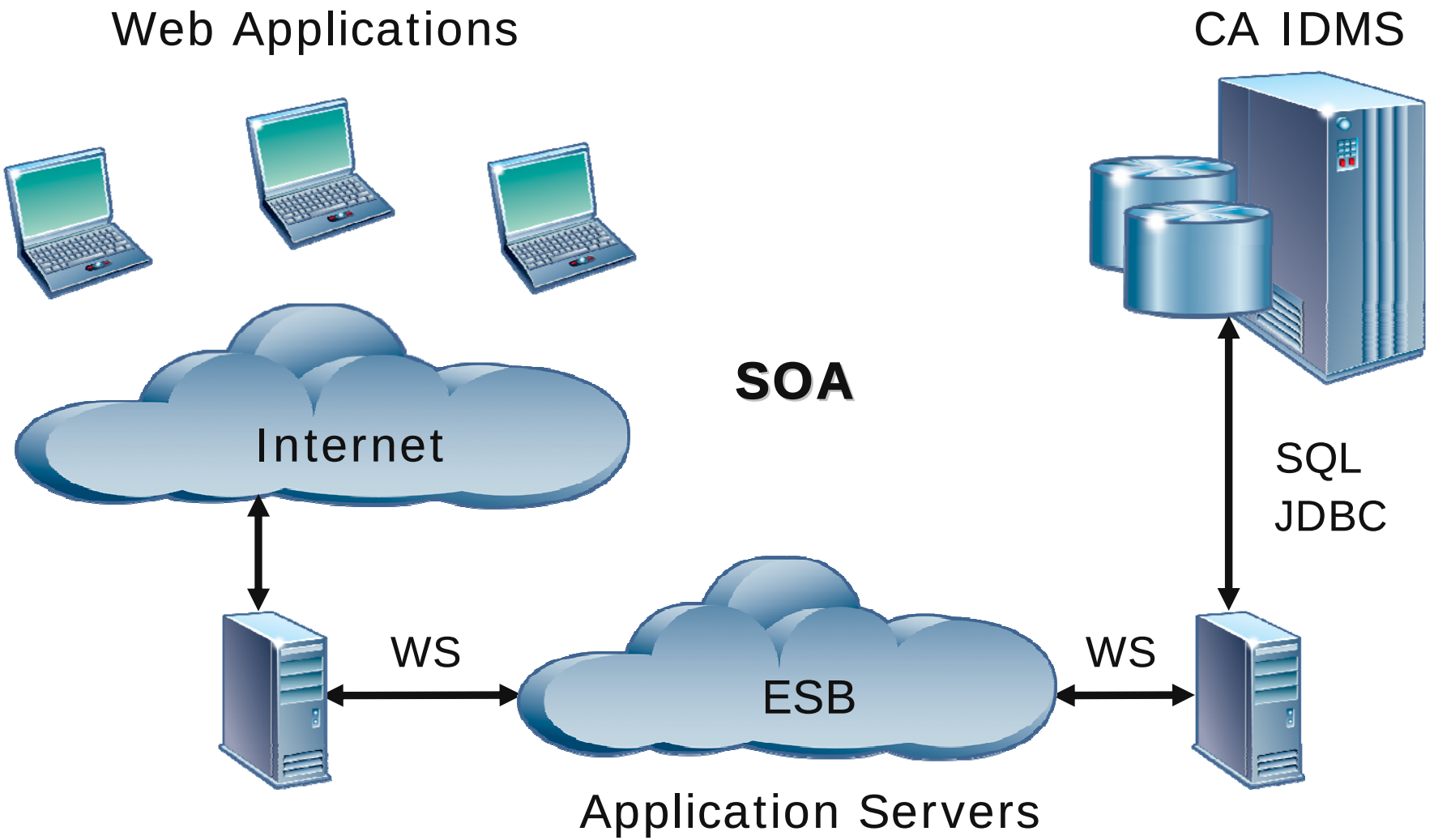
> Establish Standards

- SQL
- ODBC, JDBC, and .NET
- Web Services

> Define Appropriate Access Methods

- CA IDMS™ SQL
- CA IDMS™ Server
- Views, Table Procedures, Call Procedures, Foreign Keys

Modern Application Architecture



Modernizing CA IDMS Databases

Access Existing Network Databases

- CA IDMS SQL
- CA IDMS Server
- Views
- Table Procedures
- Called Procedures
- Foreign Keys

Network to Relational Mapping

Schema
Records
Elements
Sets

Schema

SQL Schema

- Defined in SQL catalog
- References network schema
- Optional DBNAME
  - If not specified, CONNECT DBNAME must include necessary catalog and database segments

```
CREATE SCHEMA EMPSQL (Non-SQL)
  FOR NONSQL SCHEMA
    APPLDICT.EMPSCHM V 100
  DBNAME EMPDEMO;
```


Schema

Additional SQL Schema

- Defined in SQL catalog
- References Table Procedures, Views, Pure SQL
- Establish a Standard for DBNAME Segments

```
CREATE SCHEMA xxxTBP;    (Table Procedure)

CREATE SCHEMA xxxVW;     (View)

CREATE SCHEMA xxxPR;     (Procedure)

CREATE SCHEMA xxxSQL
     DBNAME IDMSSQL;     (Native SQL)
```


Schema Table Procedure

> Separate Schema to manage Table Procedure tables

> Separate Schema to map to specific DBNAME

> Establish Naming Conventions

- Easy to identify
- Name Procedure and COBOL Program the same
- Easy to DROP if necessary
- Easy to create batch scripts to ADD or move

Schema for Views

- Separate Schema to map views
- Separate Schema to map DBNAMES
- Template to allow non DBA's to build VIEWS
- Validate VIEWS using OCF
- Easy to DROP and ADD with Batch commands
- Hide Navigation
- Control which ELEMENTs and SETs are used

VIEW Template

> Create a VIEW Source Member

> Format the template

- DROP view_schema_name.view
- ADD view_schema_name.view

```
        (sql_column,
         sql_column,
         sql_column)
AS SELECT element,
                element,
                element
FROM schema_name.record
WHERE ‘where selection criteria’;
```


VIEW Case Study

- Multiple Plants
- Data Download accuracy and time
- SQL Only CV
- Customize a small number of VIEWs
- Customize DBNAMEs to support Plant specific access
  - Catalog Segment
  - Dictionary Segment
  - Plant Specific Data Segments
- Customize SYSCA_ACCESSIBLE_TABLES

SQL and Elements

Network to Relational Mapping

> Some structures not mapped

- Group items
- Occurs depending
- OM SETs
- BOM Structures
- Multi Member SETs

Elements

Columns

- Lowest level elements
- Type mapped to SQL type
- SQL synonym can override name
- Hyphens translated to underscores
- Not all element definitions mapped
  - Redefines
  - Occurs depending on

SQL and ELEMENTS CASE STUDY

- Very large critical data record
- Large number of Elements
- Long Element Names
- Use of CA IDMS SQL Quickbridge and multiple Table Procedures
- Secure Table Procedure Programs to restrict Users
- Deal with Redefines Elements
- Limited access to all Elements in the record

CA IDMS Defined SQL Databases

> CA IDMS Defined SQL Database Considerations

- Use for Data Warehousing
- Use for Extending Non-SQL Record Structures
- Use for Non-SQL data summarization
- Use for developing pure SQL Applications

➢ Trade off Considerations

- ➢ Could require Data Replication
- ➢ Could require Two-Phase Commit
- ➢ Data Synchronization

> Case Study

CA IDMS SQL Run Time Considerations

> CA IDMS Defined SQL Only Central Version

- Backup Copy from night before in Retrieval
- Optimized for SQL and batch Reporting
- Use for Non-SQL data summarization
- SYSGEN considerations

➢ Trade off Considerations

- ➢ May not provide timely access to data
- ➢ Will not support SQL or Native Updates

CA IDMS SQL Run Time Considerations

> CA IDMS SQL Only Central Version

- Consider creating multiple CASERVER Tasks
- Use Task limits for CASERVER Tasks
- Create SYSCA-ACCESSIBLE Tables views to limit access to schemas by function, group, user

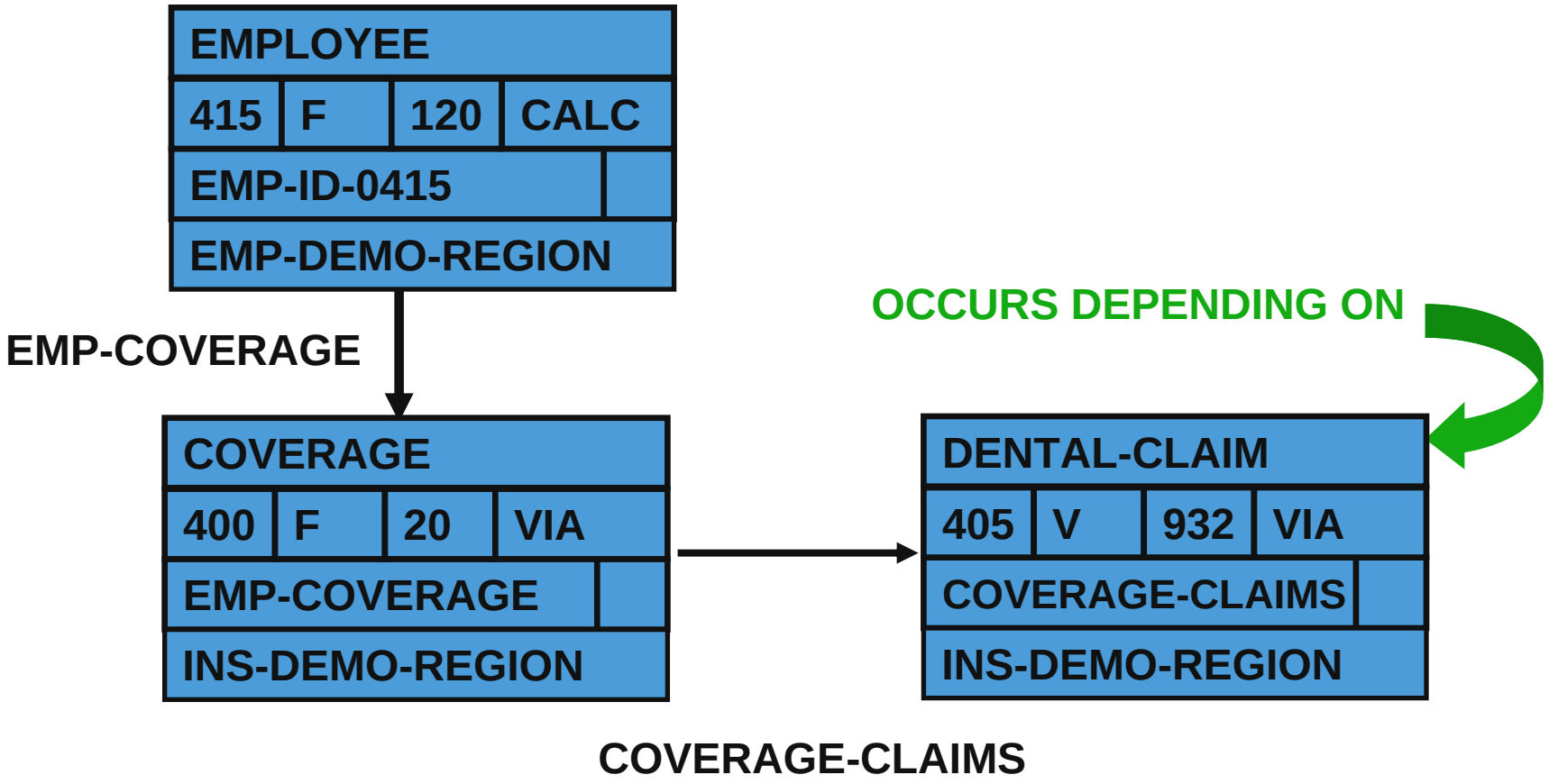
➢ Trade off Considerations

- ➢ May be too restrictive for power users

Techniques

Syntax Extensions
Table Procedures
Foreign Keys

EMPSCHM Database



Syntax Extensions

Set Specification

```
SELECT EMP_ID_0415,
       INS_PLAN_CODE_0400,
       DENTIST_LICENSE_NUMBER_0405
  FROM EMPSQL.EMPLOYEE,
       EMPSQL.COVERAGE,
       EMPSQL."DENTAL-CLAIM" C
  WHERE "EMP-COVERAGE"
       AND "COVERAGE-CLAIMS".C
```


Syntax Extensions CONTINUED

ROWID Pseudo-Column

```
UPDATE EMPSQL.COVERAGE C
  SET SELECTION_YEAR_0400 = 20
  WHERE C.ROWID IN (
    SELECT CI.ROWID
      FROM EMPSQL.EMPLOYEE E,
        EMPSQL.COVERAGE CI
      WHERE "EMP-COVERAGE"
        AND EMP_ID_0415 = 23)
```


Table Procedures

Best Practice Recommendations

> Do not create a Table Procedure for each record type

> Create Retrieval and Update Table Procedures

> Use SQL mapping for long fields to simplify the column

> Do not use a generic schema for Procedure definitions

> Return more data than expected to support a wide variety of queries

Table Procedures

Best Practice Recommendations

> Create Work Records for Update Procedures

> Naming Standard for Table and Program

Table Procedures

Encapsulate Native DML

> External program called at runtime to return rows of data to the SQL engine

> Accessed like table in SQL DML

> Returns a result set like a table

> Generated by CA IDMS™ SQL QuickBridge

> Encapsulate relationships

Call Interface

Like a Remote Procedure Call

> Extract logic from application

- Subroutines
- Map-less dialogs
- Extracted code

> Invoke using

- SQL procedures
- Socket programs
- JCA adapters
- Web services

Call Interface

```
CREATE PROCEDURE CCTSPROC.TCHRSRCE
  ( TCHR_NAME       CHARACTER(30),
    CRSE_ID          CHARACTER(6),
    CRSE_TITLE       CHARACTER(20),
    CRSE_DBKEY      INTEGER,
    END_OF_CRSE    CHARACTER(1)
  )
    EXTERNAL NAME TCHRSRCE PROTOCOL ADS
    DEFAULT DATABASE NULL
    SYSTEM MODE
    LOCAL WORK AREA 0
    GLOBAL WORK AREA 2048;
```


Call Interface Case Study

> Extract logic from application
  - Subroutines
  - Map-less dialogs
  - Extracted code

> .NET Front-End for Users

> Addressing ADSA Control and Security

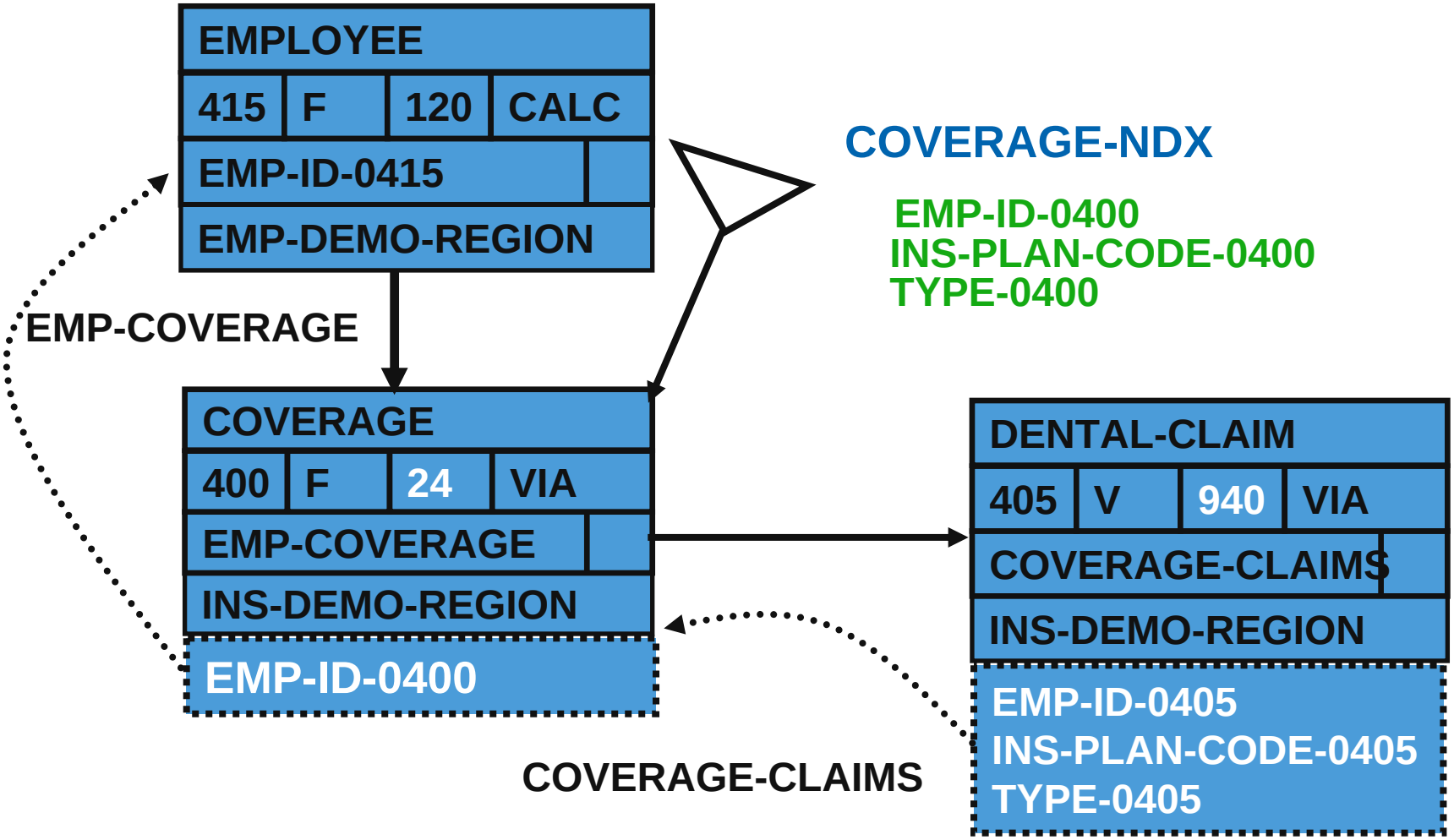
> Access from .NET with Non-SQL and SQL data

Benefits of Foreign Keys

Standard SQL

- Referential constraints enforced
  - INSERT
  - UPDATE
  - DELETE
- Standard metadata
  - ODBC
  - JDBC
- Compatibility with common tools
  - Application generators
  - Data warehousing

EMPSCHM Database Modernized



Implementing Foreign Keys

Exposes Sets as Constraints

> Add foreign keys to record definition

> Modify set definitions to add primary/foreign keys

> Restructure member record and populate keys

> Maintain foreign keys in network applications

> Alternative to migration

- Can do incrementally
- Less cost
- Less effort

Tradeoffs

	Standard SQL	New programs	Application changes	Restructure	Sets
SQL extensions	No	No	No	No	Limited
Table procedures	Yes	Yes	No	No	Encapsulate
Foreign keys	Yes	No	Limited	Targeted	Referential constraints

JDBC/ODBC Case Study

- JDBC tools used to develop Web Pages
- JDBC used to mimic current Mainframe ADS Applications
  - Table Procedures
  - Called Procedures
  - Native SQL access
- ODBC Tools used to report off of Non-SQL
- .NET Functions used to extend Non-SQL Applications

Summary

> Modernization makes sense

> Modern applications and tools can work with CA IDMS

> SQL can access CA IDMS out of the box

- Some legacy databases need help
- A variety of techniques to help standardize SQL access

> Modernization can be incremental

> No need to migrate

Legal

This presentation was based on current information and resource allocations as of October 10, 2007 and is subject to change or withdrawal by CA at any time without notice. Notwithstanding anything in this presentation to the contrary, this presentation shall not serve to (i) affect the rights and/or obligations of CA or its licensees under any existing or future written license agreement or services agreement relating to any CA software product; or (ii) amend any product documentation or specifications for any CA software product. The development, release and timing of any features or functionality described in this presentation remain at CA's sole discretion. Notwithstanding anything in this presentation to the contrary, upon the general availability of any future CA product release referenced in this presentation, CA will make such release available (i) for sale to new licensees of such product; and (ii) to existing licensees of such product on a when and if-available basis as part of CA maintenance and support, and in the form of a regularly scheduled major product release. Such releases may be made available to current licensees of such product who are current subscribers to CA maintenance and support on a when and if-available basis. In the event of a conflict between the terms of this paragraph and any other information contained in this presentation, the terms of this paragraph shall govern.

Legal

Certain information in this presentation may outline CA's general product direction. All information in this presentation is for your informational purposes only and may not be incorporated into any contract. CA assumes no responsibility for the accuracy or completeness of the information. To the extent permitted by applicable law, CA provides this document "as is" without warranty of any kind, including without limitation, any implied warranties or merchantability, fitness for a particular purpose, or non-infringement. In no event will CA be liable for any loss or damage, direct or indirect, from the use of this document, including, without limitation, lost profits, lost investment, business interruption, goodwill, or lost data, even if CA is expressly advised of the possibility of such damages.

Questions & Answers

Transforming IT Management