CA IDMS™ 19.0
SQL Enhancements for Modernization

Jean-Paul DeFeyter
Jake Hart
CA Technologies

IUA/CA IDMS™ Technical Conference
May 16-20, 2016

Abstract

- Customers need to leverage their investment in CA IDMS to create new applications based on services in the application economy, and they need to do this with current technologies and developer skill sets. This session shows how SQL enhancements in CA IDMS 19.0 improve standards compliance, developer productivity, and compatibility with third party tools and applications. The SQL virtual foreign key feature enables developers to use standard SQL to access and update network databases without the need to use network DML or table procedures. Enhancements to SQL DDL enable users to define databases using standard DDL compatible with other databases.

Agenda

- Virtual Key Feature
- SQL DDL Enhancements

Virtual Key Feature

CA IDMS SQL Relational to Network Mapping

SQL		Network	
Table	• Identically named	Record definition	• Via SCHEMA
Row	• Equivalent	Record occurrence	• Equivalent
Column	• Automatically renamed	Element	• Except ODO, redefines, …
Referential constraint	• Partial mapping	Set	• Lack of foreign keys

Relational To Network Mapping Techniques

- **Syntax extensions**
- **Views**
- **Procedures**
  - Table procedures
  - Called procedures
- **Foreign keys**
  - Referential sets

Relational to Network Mapping Comparison of Techniques

	Standard SQL	New programs	Application changes	Restructure	Sets
SQL Extensions	No	No	No	No	Limited
Views	Yes	No	No	No	Limited
Table Procedures	Yes	Yes	No	No	Encapsulate
Referential Sets	Yes	No	Yes	Maybe	Referential constraints

Virtual Key Feature Overview

- Enhances the CA IDMS SQL capability to access network-defined databases
- Enables SQL programmers and development tools to access CA IDMS with standard SQL
- Virtual keys allow you to use referential constraint to access and manipulate network data through SQL
- Does not require special CA IDMS SQL syntax
- Removes restrictions on INSERT and UPDATE for network DB

Virtual Key Feature Overview

Provides the following benefits

- Does not require database changes
- Does not require changes to network definitions
- Does not require table procedures
- Allows flexible navigation, similar to network DML
- Provides easy, non-disruptive adoption

Enabling Access to Data Using Virtual Keys

- Creating a referencing schema with virtual keys
  CREATE SCHEMA EMPSCHM FOR NONSQL SCHEMA EMPDICT.EMPSCHM VERSION 100 WITH VIRTUAL KEYS
- Change a referencing schema to remove virtual keys
  ALTER SCHEMA EMPSCHM WITHOUT VIRTUAL KEYS
- When WITH VIRTUAL KEYS specified, ROWID and virtual foreign keys become visible
  - On SELECT *
  - In INSERT column list when list is omitted
  - In UPDATE column list when list is omitted

Virtual Primary Key - ROWID

- Available since CA IDMS Release 16.0
- Pseudo column of datatype ROWID
  - Uniquely identifies a row
  - Length 8 bytes
  - Contains DBKey and page info
- Value not persistent and thus **not a true primary key**
- Available for every table
- Not nullable for base tables
- Relational operator support extended (<,>,<=,>=)

Virtual Foreign Key - FKEY_setname

- Pseudo column of name **FKEY_setname** where *setname* is the SQL transformed name of set
- Datatype ROWID
  - Must be 8 bytes
  - contains DBKey and page info of set owner
- Available for every set in which record is member
- Always nullable
- Value is rowid of owner of set-name or null if no membership
- Participates as a foreign key in the referential relationship with ROWID as primary key

Virtual Foreign Key - FKEY_setname

- Virtual foreign keys can be referenced in SQL statements like a user-defined column
- Included in list of “all” columns (SELECT *)
- Included in INSERT and UPDATE default column list when column list is excluded
- Column order:
  - User columns
  - ROWID
  - Virtual foreign keys in alphabetical order

SELECT using Virtual Keys Examples

- Obtain DEPARTMENT data of an EMPLOYEE

```
select *
 from DEPARTMENT where ROWID =
  (select FKEY_DEPT_EMPLOYEE
    from EMPLOYEE
   where EMP_ID = 1001)
```

- Join DEPARTMENT and EMPLOYEE

```
select d.*, e.*
 from DEPARTMENT d, EMPLOYEE e
   where d.ROWID = e.FKEY_DEPT_EMPLOYEE
```

SELECT using Virtual Keys Examples

- Select all EXPERTISE records for an EMPLOYEE

```
select * from EXPERTISE
  where FKEY_EMP_EXPERTISE =
     (select ROWID
       from EMPLOYEE
       where EMP_ID = 1001)
```

- Show all DEPARTMENTs including those without EMPLOYEEs

```
select d.*, e.*
  from DEPARTMENT d left join EMPLOYEE e
   on d.ROWID = e.FKEY_DEPT_EMPLOYEE
```


INSERT using Virtual Keys

- Supports specifying values for virtual keys (ROWID and VFKs)
- Referential integrity enforced for all sets in which record participates as a member
- Set membership options used to define integrity constraints (e.g. MA set disallows NULL valued virtual foreign key)
- You can specify ROWID to supply suggested DBKEY for a record with location mode DIRECT
  - If value not supplied or supplied as NULL, defaults to -1 on STORE (first available dbkey in page range used)
  - Supplied value for non-DIRECT record location mode is ignored

INSERT using Virtual Keys

- Use of scalar-query-expression in VALUES clause
  - New R19.0 feature available on INSERT statements for all SQL schemas
  - When used with virtual keys, allows an INSERT to contain queries that retrieve the virtual foreign key values that are then used in the row insertion
- Support for virtual keys in query-specification
  - Query-specification is alternative method to VALUES clause on INSERT
  - The result of the query must have same number of columns as have been named in INSERT statement
  - If no columns named, number of columns must match the total number of columns including ROWID and all VFKs

INSERT using Virtual Keys Examples

- Specifying VALUES clause and column list

```
insert into EMPLOYEE  (EMP_ID, EMP_FNAME, EMP_LNAME,
FKEY_DEPT_EMPLOYEE,  FKEY_OFFICE_EMPLOYEE)
values  (976, 'SEBASTIAN', 'VOLLMER',
        (select ROWID from DEPARTMENT where DEPT_ID = 9003),
        (select ROWID from OFFICE where OFFICE_CODE = '002'));
```

- Specifying VALUES clause without column list

```
insert into EMPLOYEE values  (976, 'SEBASTIAN', 'VOLLMER', NULL,
(select ROWID from DEPARTMENT where DEPT_ID = 9003),
(select ROWID from OFFICE where OFFICE_CODE = '002'));
```

* Note: EMPLOYEE columns are

EMP_ID, EMP_FNAME, EMP_LNAME, ROWID, FKEY_DEPT_EMPLOYEE, FKEY_OFFICE_EMPLOYEE

INSERT using Virtual Keys Examples (cont.)

- Specifying virtual keys using query-specification

```
insert into EMPOSITION
 (START_YEAR, FKEY_EMP_EMPOSITION, FKEY_JOB_EMPOSITION)
 select 2015, E.ROWID, J.ROWID from EMPLOYEE E, JOB J
 where E.EMP_ID = 23 and J.JOB_ID = 2025;
```


UPDATE

- Virtual Foreign Keys can be specified in SET clause
- Equivalent to CONNECT/DISCONNECT DML verbs
- Allowed for sets with membership options
  - OPTIONAL (OA/OM)
  - MANDATORY MANUAL (MM) when record not yet connected to set
- SET FKEY_setname values allowed
  - NULL
  - ROWID of owner occurrence of setname
  - ROWID of member occurrence of setname

UPDATE Examples

- Disconnecting the EMPLOYEE from the DEPT-EMPL set

```
UPDATE EMPLOYEE SET FKEY_DEPT_EMPL = NULL
      WHERE EMP_ID = 23;
```

- EMPLOYEE connected to DEPARTMENT 4000

```
UPDATE EMPLOYEE SET FKEY_DEPT_EMPL =
  (SELECT ROWID FROM DEPARTMENT WHERE DEPT_ID=4000)
WHERE EMP_ID = 23;
```


DELETE

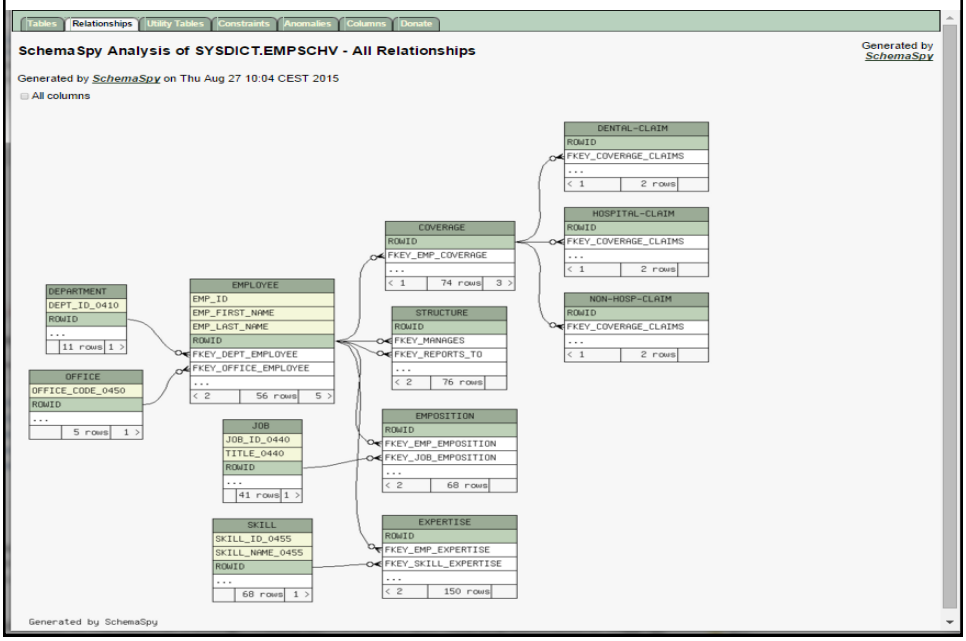
- Virtual key columns may be used in the same way as user-defined columns in a DELETE statement

```
DELETE FROM EMPLOYEE
   WHERE FKEY_DEPT_EMPL =
  (SELECT ROWID FROM DEPARTMENT  WHERE DEPT_NAME ='Sales');
```

Exposing Virtual Keys

- JDBC/ODBC metadata APIs expose virtual keys
- Enables discovery of network database relationships by Java and Windows applications and tools
- See *Using IDMS Server* section on DocOps.ca.com for more information on supported metadata APIs
- *SQL Virtual Foreign Keys Simplify Hibernate Access to Your Existing CA IDMS Databases* session

Discovering Network Relationships



Virtual Key Feature Summary

	Standard SQL	New programs	Application changes	Restructure	Sets
SQL Extensions	No	No	No	No	Limited
Views	Yes	No	No	No	Limited
Table Procedures	Yes	Yes	No	No	Encapsulate
Referential Sets	Yes	No	Yes	Maybe	Referential constraints
Virtual Keys	Yes	No	No	No	Referential constraints

SQL DDL Enhancements

SQL DDL Enhancements

- The SQL CREATE TABLE command is enhanced to support ISO standard DDL
  - Unique constraints
  - Primary key constraints
  - Referential constraints
- Enables better integration of third party frameworks and tools with CA IDMS such as Hibernate and CA Test Data Manager

CREATE TABLE UNIQUE Constraint

- A unique constraint is specified by coding
  - UNIQUE on a column definition
  - UNIQUE (column names) in the column list
  - A constraint may be named
    - CONSTRAINT name
    - Precedes the constraint definition
    - If not named, a name is generated
  - More than one unique constraint may be defined
- Example:

```
CREATE TABLE ABC.TABLE1 (COL1 CHAR(8) UNIQUE,
                         COL2 CHAR(8),
                         CONSTRAINT CON1 UNIQUE (COL2, COL1)
                         );
```

CREATE TABLE
PRIMARY KEY Constraint

- A Primary Key constraint is specified by coding
  - PRIMARY KEY on a column definition
  - PRIMARY KEY (column names) in the column list
  - A constraint may be named
    - CONSTRAINT name
    - Precedes the constraint definition
    - If not named, a name is generated
- ONE Primary Key constraint per table
- ODBC/JDBC metadata functions return Primary Key

CREATE TABLE
Primary Key Constraint Examples

```
CREATE TABLE ABC.TABLEX (
            COL1 CHAR(8) NOT NULL PRIMARY KEY,
            COL2 CHAR(8) NOT NULL,
            COL3 CHAR(8) NOT NULL
            );

CREATE TABLE ABC.TABLEY (
            COL1 CHAR(8) NOT NULL,
            COL2 CHAR(8) NOT NULL,
            COL3 CHAR(8) NOT NULL,
            PRIMARY KEY (COL2, COL3)
            );
```

CREATE TABLE UNIQUE/PRIMARY KEY

- Creates an enforcing index
- Suppresses creation of default index
- Generated Index names
  - IDX###############
  - ### is a 15 digit number unique on the schema
- Uses default values for index attributes

CREATE TABLE Enforcing Indexes

- Enforces unique / primary key constraints
- Cannot be altered to be NOT UNIQUE
- Cannot be dropped with DROP INDEX
  - Unless there is a replacement index
- When primary key index dropped
  - A replacement inherits the primary key attribute

CREATE TABLE
Replacement Indexes

- Use CREATE UNIQUE INDEX
- Columns must match enforcing index
- Inherits enforcing index attribute
- Generated indexes dropped automatically
- Non-generated indexes must be manually dropped

CREATE TABLE
Referential Constraints

- A referential constraint is specified by coding
  - REFERENCES clause on a column definition
  - FOREIGN KEY clause in the column list
  - A constraint may be named
    - CONSTRAINT name
    - Precedes the constraint definition
    - If not named, a name is generated
      - GEN################
    - More than one referential constraint may be defined

CREATE TABLE
Referential Constraints Example

```
CREATE TABLE ABC.TABLEX (
    COL1 CHAR(8),
    COL2 CHAR(8),
    FOREIGN KEY (COL1,COL2) REFERENCES TABLEP,
    COL3 CHAR(8) NOT NULL
        CONSTRAINT CON1  REFERENCES TABLEP (COL5)
    );
```


Summary

- CA IDMS 19.0 enables Java and Windows developers to use industry standard technology to create, access, and maintain CA IDMS databases
- SQL Virtual Key feature enables developers to use standard SQL to access and update network databases
- SQL DDL enhancements enable CA IDMS to support database definitions provided by popular tools and frameworks
- Additional sessions
  - *SQL Virtual Foreign Keys Simplify Hibernate Access to Your Existing CA IDMS Databases*
  - *SQL Standard DDL Enhances Application Development with CA IDMS and Popular Tools*

FOR INFORMATION PURPOSES ONLY

Terms of this Presentation

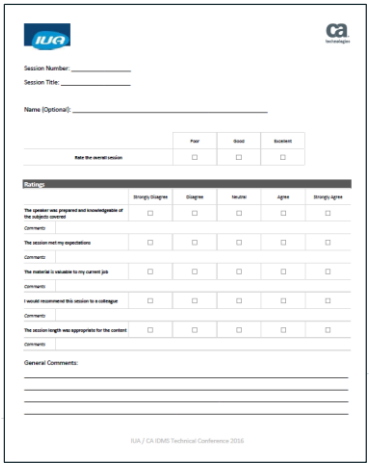
This presentation was based on current information and resource allocations as of May 2016 and is subject to change or withdrawal by CA at any time without notice. Notwithstanding anything in this presentation to the contrary, this presentation shall not serve to (i) affect the rights and/or obligations 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.

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 presentation "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 in advance of the possibility of such damages. CA confidential and proprietary. No unauthorized copying or distribution permitted.

Questions and Answers

Please Complete a Session Evaluation Form

- The number for this session is **A02**
- After completing your session evaluation form, place it in the envelope at the front of the room



Session Number: _____

Session Title: _____

Name (Optional): _____

	Poor	Good	Excellent
Rate the overall session	☐	☐	☐

Rating:

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
The speaker was prepared and knowledgeable of the subjects covered	☐	☐	☐	☐	☐
Comments:					
The session met my expectations	☐	☐	☐	☐	☐
Comments:					
The material is valuable to my current job	☐	☐	☐	☐	☐
Comments:					
I would recommend this session to a colleague	☐	☐	☐	☐	☐
Comments:					
The session length was appropriate for the content	☐	☐	☐	☐	☐
Comments:					

General Comments:

_____

_____

_____

_____

IUA / CA IDMS Technical Conference 2016