

# CA IDMS™ 19.0 SQL Enhancements for Modernization

Carla Pereira  
Jonathan Thomson  
CA Technologies

IUA/CA IDMS™ Technical Conference May 7-11, 2018



## 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.



Copyright © 2018 CA. All rights reserved.



2

## Agenda

- 1 SQL VIRTUAL FOREIGN KEYS
- 2 ISO STANDARD SQL CONSTRAINT DDL

# SQL Virtual Foreign Keys

## 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

## Comparison of Mapping Techniques

	Standard SQL	New programs	Application changes	Restructure	Exposes 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 Foreign Keys

### Expose Sets as Referential Constraints

- Queries
- Updates

### Based on ROWID

- Virtual primary key = ROWID
- Virtual foreign key = FKEY\_<set\_name>

### Define on SQL Schema definition

### Visible through JDBC and ODBC metadata

### Enable access for common tools and application frameworks

### Use SQL to access and maintain network databases from Java

## Virtual Foreign Key Benefits

- Use standard SQL to access and update CA IDMS
  - Removes INSERT and UPDATE limitations
  - Capabilities like network DML
  - Write applications using Java, .NET, and popular frameworks
  - No special CA IDMS SQL syntax
- Easy to implement
  - No database changes
  - No changes to network definitions
  - No need for table procedures



Copyright © 2018 CA. All rights reserved.



9

## Virtual Primary Key

- Pseudo column ROWID
  - Uniquely identifies a row
  - Available for every table
  - Not nullable
- Datatype ROWID
  - Length 8 bytes
  - Contains DBKEY and Page Info
  - Value not persistent and thus not a true primary key
  - Relational operator support extended (<,>,<=,>=)



Copyright © 2018 CA. All rights reserved.



10

## Virtual Foreign Key

- Pseudo column FKEY\_setname
  - “setname” is the SQL transformed name of set
  - Value is ROWID of owner of set-name or NULL if no membership
  - Available for every set in which record is member
  - Always appears nullable, but cannot set to NULL for MA sets
- Datatype ROWID
  - Must be 8 bytes
  - contains DBKEY and Page Info of set owner



Copyright © 2018 CA. All rights reserved.



11

## Virtual Foreign Keys

- 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 defined columns
    - ROWID
    - Virtual foreign keys in alphabetical order



Copyright © 2018 CA. All rights reserved.



12

## Enabling Virtual Foreign Keys

- “Create” virtual keys with SQL schema definition
 

```
create schema EMPSCHEM
  for nonsql schema EMPDICT.EMPSCHEM version 100
  with virtual keys
```
- ROWID and virtual foreign keys become visible
  - SELECT \*
  - INSERT column list when list is omitted
  - UPDATE column list when list is omitted
- “Remove” virtual keys
 

```
alter schema EMPSCHEM without virtual keys
```

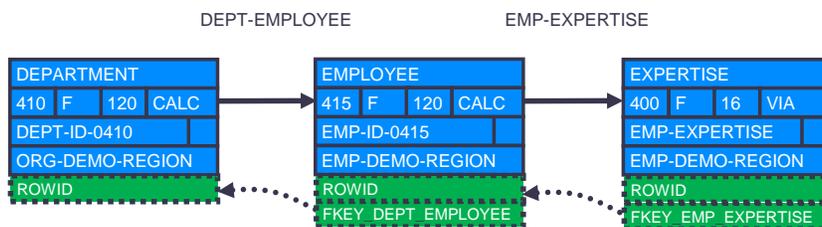


Copyright © 2018 CA. All rights reserved.



13

## EMPSCHEM With Virtual Foreign Keys



Copyright © 2018 CA. All rights reserved.



14

## SELECT 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
```



Copyright © 2018 CA. All rights reserved.



15

## More SELECT 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
```



Copyright © 2018 CA. All rights reserved.



16

## INSERT using Virtual Keys

- Use to establish set membership
  - Set virtual foreign key column to owner ROWID or NULL
  - Set membership options define integrity constraints
  - MA set disallows NULL valued virtual foreign key
- Use ROWID to suggest DBKEY for a DIRECT record
  - Defaults to -1 on STORE (first available DBKEY in page range used)
  - Ignored if record location mode is not DIRECT



Copyright © 2018 CA. All rights reserved.



17

## INSERT using Virtual Keys

- Use of scalar-query-expression in VALUES clause
  - New R19.0 feature for INSERT statements for all SQL schemas
  - With virtual keys, allows INSERT to contain queries that retrieve the virtual foreign key values to be set in the inserted row
- Support for virtual keys in query-specification
  - Alternative to VALUES clause on INSERT
  - Query result must have same number of columns as named in INSERT statement
  - If no columns named, number of columns must match the total number of columns, including ROWID and all VFKs



Copyright © 2018 CA. All rights reserved.



18

## INSERT Examples

### VALUES clause with column list

```
insert into EMPLOYEE
  (EMP_ID,
   EMP_FNAME,
   EMP_LNAME,
   FKEY_DEPT_EMPLOYEE,
   FKEY_OFFICE_EMPLOYEE)
values
  (12,
   'TOM',
   'BRADY',
   (select ROWID from DEPARTMENT where DEPT_ID = 9003),
   (select ROWID from OFFICE where OFFICE_CODE = '001'))
```



Copyright © 2018 CA. All rights reserved.



19

## INSERT Examples

### VALUES clause without column list

```
insert into EMPLOYEE
values
  (12,
   'TOM',
   'BRADY',
   NULL,
   (select ROWID from DEPARTMENT where DEPT_ID = 9003),
   (select ROWID from OFFICE where OFFICE_CODE = '001'))
```

- EMPLOYEE columns are  
EMP\_ID, EMP\_FNAME, EMP\_LNAME, **ROWID**,  
FKEY\_DEPT\_EMPLOYEE, FKEY\_OFFICE\_EMPLOYEE



Copyright © 2018 CA. All rights reserved.



20

## INSERT Examples

### 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
```



Copyright © 2018 CA. All rights reserved.



21

## 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



Copyright © 2018 CA. All rights reserved.



22

## 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



Copyright © 2018 CA. All rights reserved.



23

## DELETE

- Use virtual key columns the same as user-defined columns

```
delete from EMPLOYEE  
where FKEY_DEPT_EMPL =  
(select ROWID from DEPARTMENT  
where DEPT_NAME = 'Sales')
```



Copyright © 2018 CA. All rights reserved.



24

## Discovering Virtual Keys

- JDBC and ODBC metadata APIs expose virtual keys
- java.sql.DatabaseMetaData interface methods
  - getPrimaryKeys
  - getExportedKeys
  - getImportedKeys
  - getCrossReference
- ODBC API functions
  - SQLPrimaryKeys
  - SQLForeignKeys

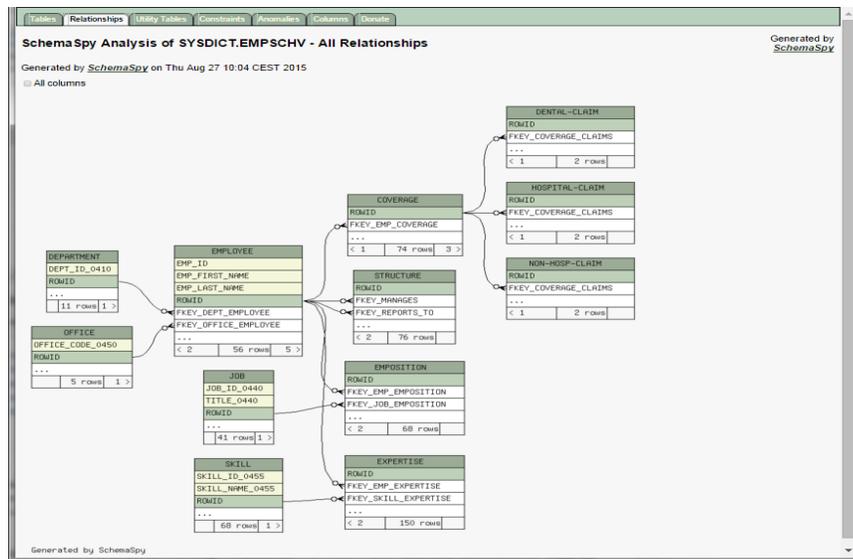


Copyright © 2018 CA. All rights reserved.



25

## Discovering Network Relationships



Copyright © 2018 CA. All rights reserved.



26

## Virtual Keys Compared to other Mapping Techniques

	Standard SQL	New programs	Application changes	Restructure	Exposes 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

## ISO Standard SQL Constraint DDL

## SQL DDL Enhancements

- ISO Standard Constraint Definition DDL
  - CREATE TABLE
  - ALTER TABLE
  - Unique constraints
  - Primary Key constraints
  - Referential constraints
- Improves integration with frameworks and tools
- DISPLAY TABLE

## Defining UNIQUE Constraints

- 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
  - Name is generated if not named
- More than one unique constraint may be defined

## UNIQUE Constraint Example

```
create table ABC.TABLE1(  
  COL1 CHAR(8) unique,  
  COL2 CHAR(8),  
  constraint CON1 unique (COL2, COL1))
```

## Defining PRIMARY KEY Constraints

- 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
  - Name is generated if not named
- ONE Primary Key constraint per 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))
```

## Generated Enforcing Index

- Enforces unique and primary key constraints
- Created by CREATE TABLE and ALTER TABLE
  - Suppresses creation of default index
  - Uses default values for index attributes
  - Cannot be altered to be NOT UNIQUE
  - Cannot be dropped with DROP INDEX unless replacement defined
- Generated Index names
  - IDX#####
  - ### is a 15 digit number unique in the schema

## Replacing a Generated Enforcing Index

- Use CREATE UNIQUE INDEX
- Columns must match enforcing index
- Inherits enforcing index attribute
- When primary key index dropped
  - A replacement inherits the primary key attribute
- Generated indexes dropped automatically
- Non-generated indexes must be manually dropped



Copyright © 2018 CA. All rights reserved.



35

## Defining 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



Copyright © 2018 CA. All rights reserved.



36

## 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))
```

## Display Table Example

```
DISPLAY TABLE ABC.TABLEP;  
*+ CREATE TABLE ABC.TABLEP  
*+ ( COL1 CHARACTER(8) NOT NULL,  
*+ COL2 CHARACTER(8) NOT NULL,  
*+ COL3 CHARACTER(10),  
*+ COL5 CHARACTER(8) NOT NULL,  
*+ PRIMARY KEY  
*+ ( COL1, COL2 ),  
*+ CONSTRAINT JHT1 UNIQUE  
*+ ( COL5 )  
*+ )  
*+ IN SQLDEMO.EMPLAREA  
*+ NO DEFAULT INDEX  
*+ TABLE ID 1036  
*+ IDX733846748625851 UNIQUE INDEX ON ( COL1, COL2 )  
*+ JHT1 UNIQUE INDEX ON ( COL5 )  
*+ REFERENCED BY ABC.TABLEX IN CONSTRAINT CON733846762869316  
*+ REFERENCED BY ABC.TABLEX IN CONSTRAINT CON1  
*+ ;
```

## Display Table Example with FULL option

```
*+ CREATE UNIQUE INDEX JHT1
*+   ON ABC.TABLEP
*+   DATE CREATED 2018-04-30-14.12.00.842335
*+   THIS IS AN ENFORCING INDEX
*+   ( COL5 ASC )
*+   UNCOMPRESSED
*+   IN SQLDEMO.EMPLAREA
*+   INDEX BLOCK CONTAINS 10 KEYS DISPLACEMENT IS 2 PAGES
*+   INDEX ID 13
*+   ;
```



## Discovering Constraints

- java.sql.DatabaseMetaData interface methods
  - getIndexInfo
  - getPrimaryKeys
  - getExportedKeys
  - getImportedKeys
  - getCrossReference
- ODBC API functions
  - SQLStatistics
  - SQLPrimaryKeys
  - SQLForeignKeys



## 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



Copyright © 2018 CA. All rights reserved.



41

## FOR INFORMATION PURPOSES ONLY Terms of this Presentation

This presentation was based on current information and resource allocations as of May 2018 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.



Copyright © 2018 CA. All rights reserved.



42

# Questions & 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

The evaluation form includes the following sections:

- Session Number: \_\_\_\_\_
- Name (Optional): \_\_\_\_\_
- Session Title: \_\_\_\_\_
- Rate the overall session: 

Not	Good	Excellent
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
- Strongly disagree, disagree, neutral, agree, strongly agree (with checkboxes)
- Comments: \_\_\_\_\_
- The speaker was prepared and knowledgeable of the subject matter (with checkboxes)
- Comments: \_\_\_\_\_
- The session met my expectations (with checkboxes)
- Comments: \_\_\_\_\_
- The material is related to my current job (with checkboxes)
- Comments: \_\_\_\_\_
- I would recommend this session to a colleague (with checkboxes)
- Comments: \_\_\_\_\_
- The session length was appropriate for the content (with checkboxes)
- Comments: \_\_\_\_\_
- This session should be offered as a webinar (with checkboxes)
- Comments: \_\_\_\_\_
- General Comments: \_\_\_\_\_