# 6CI007 Database Server Management Assessment

**Introduction**

## University of Wolverhampton

## School of Technology

## 6CI007 Database Server Management

### Assessment 1

### Hand in April 18th 2013

In this assignment you will build a small database to support the needs of a training company called Trainset. To do this you will need to create tables to support the following scenario.

Trainset run professional short courses. These courses have a title, a duration of 1,2,3,4,or 5 days, and a cost that varies between £300 and £2000. The courses are repeatedly run at various locations. The start date of each offering, its location and its instructor are recorded. Students attend the courses and Trainset need to record the student name, gender, telephone number, and company name. Each time a student attends a course they will give an evaluation of the course (the evaluation is a number from 0 to 4), and attempt an examination. Trainset need to record the examination result of each student in each examination. It is not a requirement that all offerings of courses have attendees registered, and it is also not required that every student has to have attended a course.

**Contents:**

Title page/Introduction - Page 1

Contents - Page 2

Entity Model - Page 3

Table creation statements - Page 4 - 7

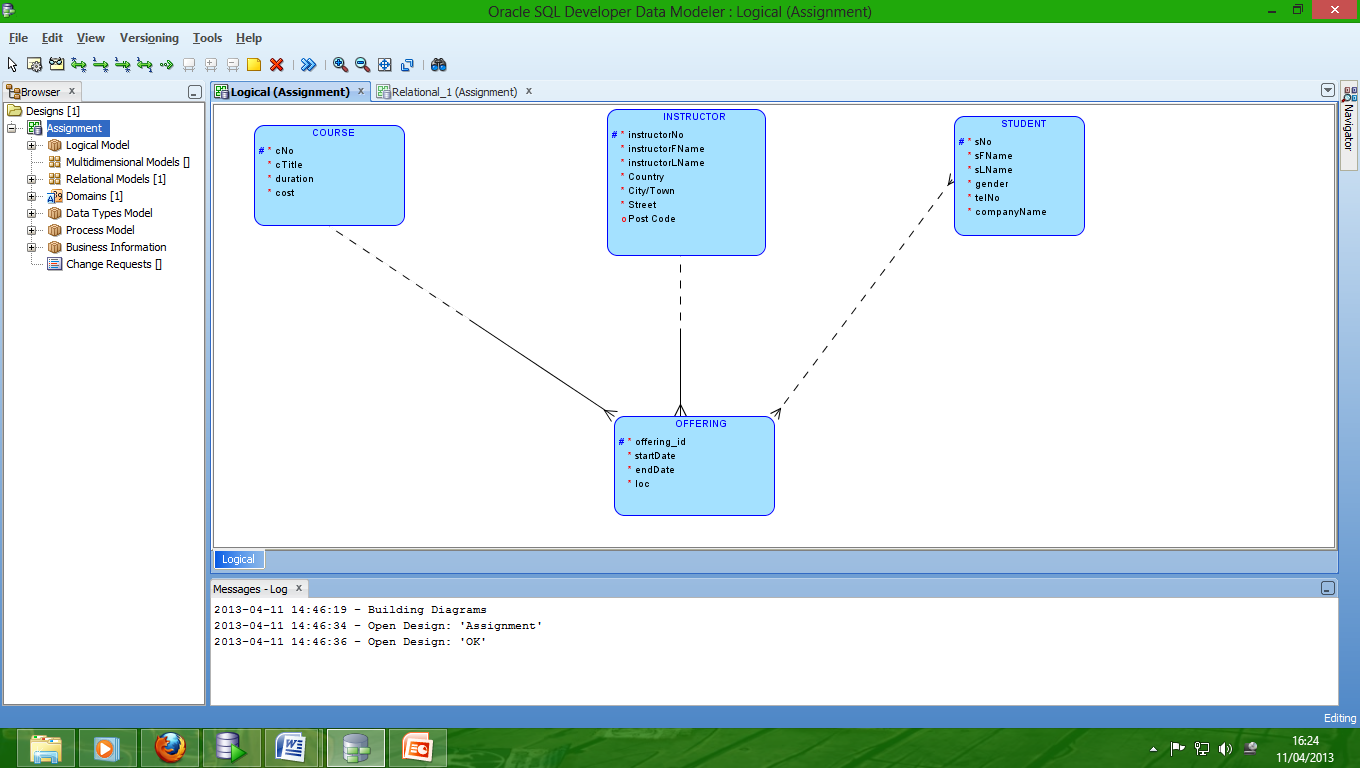
Test data - Page 8 -

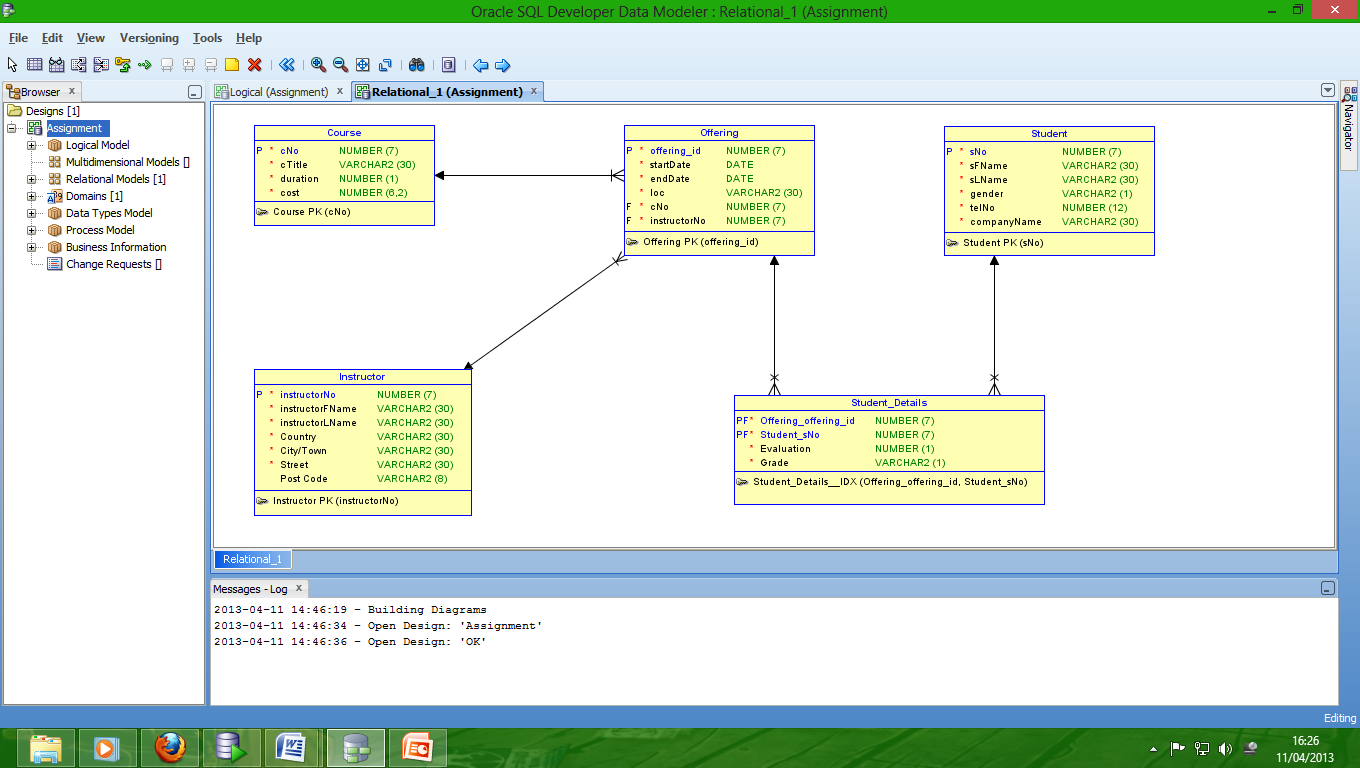
Constraints -

View definition -

Trigger definitions -

Query Output-

**Entity Model**

****

**Table Creation Statements**

-- Generated by Oracle SQL Developer Data Modeler 3.1.3.709

-- at: 2013-04-11 16:30:37 BST

-- site: Oracle Database 11g

-- type: Oracle Database 11g

CREATE TABLE Course

(cNo NUMBER (7) NOT NULL ,

cTitle VARCHAR2 (30) NOT NULL ,

duration NUMBER (1) NOT NULL ,

cost NUMBER (6,2) NOT NULL);

ALTER TABLE Course

ADD CONSTRAINT "Course PK" PRIMARY KEY ( cNo );

CREATE TABLE Instructor

(instructorNo NUMBER (7) NOT NULL ,

instructorFName VARCHAR2 (30) NOT NULL ,

instructorLName VARCHAR2 (30) NOT NULL ,

Country VARCHAR2 (30) NOT NULL ,

"City/Town" VARCHAR2 (30) NOT NULL ,

Street VARCHAR2 (30) NOT NULL ,

"Post Code" VARCHAR2 (8));

ALTER TABLE Instructor

ADD CONSTRAINT "Instructor PK" PRIMARY KEY ( instructorNo );

CREATE TABLE Offering

(offering\_id NUMBER (7) NOT NULL ,

startDate DATE NOT NULL ,

endDate DATE NOT NULL ,

loc VARCHAR2 (30) NOT NULL ,

cNo NUMBER (7) NOT NULL ,

instructorNo NUMBER (7) NOT NULL);

ALTER TABLE Offering

ADD CONSTRAINT "Offering PK" PRIMARY KEY ( offering\_id );

CREATE TABLE Student

(sNo NUMBER (7) NOT NULL ,

sFName VARCHAR2 (30) NOT NULL ,

sLName VARCHAR2 (30) NOT NULL ,

gender VARCHAR2 (1) NOT NULL ,

telNo NUMBER (12) NOT NULL ,

companyName VARCHAR2 (30) NOT NULL);

ALTER TABLE Student

ADD CONSTRAINT "Student PK" PRIMARY KEY ( sNo );

CREATE TABLE Student\_Details

(Offering\_offering\_id NUMBER (7) NOT NULL ,

Student\_sNo NUMBER (7) NOT NULL ,

Evaluation NUMBER (1) NOT NULL ,

Grade VARCHAR2 (1) NOT NULL);

ALTER TABLE Student\_Details

ADD CONSTRAINT Student\_Details\_\_IDX PRIMARY KEY

( Offering\_offering\_id, Student\_sNo );

ALTER TABLE Offering

ADD CONSTRAINT Course\_Offering FOREIGN KEY

( cNo ) REFERENCES Course ( cNo );

ALTER TABLE Student\_Details

ADD CONSTRAINT FK\_Student\_Details FOREIGN KEY

( Student\_sNo ) REFERENCES Student ( sNo )

ON DELETE CASCADE;

ALTER TABLE Student\_Details

ADD CONSTRAINT FK\_Student\_Offering FOREIGN KEY

( Offering\_offering\_id ) REFERENCES Offering

( offering\_id ) ON DELETE CASCADE;

ALTER TABLE Offering

ADD CONSTRAINT Instructor\_Offering FOREIGN KEY

( instructorNo ) REFERENCES Instructor ( instructorNo );

-- Oracle SQL Developer Data Modeler Summary Report:

-- CREATE TABLE 5

-- CREATE INDEX 0

-- ALTER TABLE 9

-- CREATE VIEW 0

-- CREATE PACKAGE 0

-- CREATE PACKAGE BODY 0

-- CREATE PROCEDURE 0

-- CREATE FUNCTION 0

-- CREATE TRIGGER 0

-- ALTER TRIGGER 0

-- CREATE STRUCTURED TYPE 0

-- CREATE COLLECTION TYPE 0

-- CREATE CLUSTER 0

-- CREATE CONTEXT 0

-- CREATE DATABASE 0

-- CREATE DIMENSION 0

-- CREATE DIRECTORY 0

-- CREATE DISK GROUP 0

-- CREATE ROLE 0

-- CREATE ROLLBACK SEGMENT 0

-- CREATE SEQUENCE 0

-- CREATE MATERIALIZED VIEW 0

-- CREATE SYNONYM 0

-- CREATE TABLESPACE 0

-- CREATE USER 0

-- DROP TABLESPACE 0

-- DROP DATABASE 0

-- ERRORS 0

-- WARNINGS 0

**Test Data**

**Course**

SQL> SELECT \* FROM COURSE;

CNO CTITLE DURATION COST

---------- ------------------------------ ---------- ----------

1000001 Computer Science 3 730

1000002 Computing 2 455

1000003 English Literature 4 1260

1000004 Mathematics 5 1530

1000005 Biology 3 890

1000006 Physics 1 510

6 rows selected.

SQL> SPOOL OFF

**Offering**

SQL> SELECT \* FROM OFFERING;

OFFERING\_ID STARTDATE ENDDATE LOC CNO INSTRUCTORNO

------------ ----------- ---------- ----------- ------------ --------------

3000001 09-OCT-07 11-OCT-07 London 1000001 2000001

3000012 24-APR-08 27-APR-08 London 1000001 2000012

3000002 22-JUN-12 26-JUN-12 Rome 1000004 2000002

3000003 01-MAY-13 01-MAY-13 Berlin 1000006 2000003

3000004 14-NOV-10 16-NOV-10 Paris 1000005 2000004

3000005 18-OCT-12 18-OCT-12 Berlin 1000006 2000005

3000006 06-JAN-07 08-JAN-07 Paris 1000005 2000006

3000008 19-DEC-12 23-DEC-12 Rome 1000004 2000008

3000009 12-NOV-11 14-MAR-11 London 1000001 2000009

3000011 17-MAY-09 20-MAY-09 Naples 1000003 2000011

3000013 12-AUG-08 15-AUG-08 Naples 1000003 2000011

11 rows selected.

SQL> SPOOL OFF

**Instructor**

SQL> SELECT \* FROM INSTRUCTOR;

INSTRUCTORNO INSTRUCTORFNAME INSTRUCTORLNAME COUNTRY CITY STREET POST\_CODE

------------ --------------- --------------- -------------- ----------- ------------------- -----------------

2000001 Joe Sullivan United Kingdom London 14 MiddleCross Road DY11 SW3

2000002 Angelo Rossi Italy Rome 25 Via Maggio 00194 Roma

2000003 Mark Lannister United Kingdom Birmingham 29 Forest Grove DY13 EA2

2000004 Camille Roux France Paris 27 Avenue Foch 97439 Sainte-Rose

2000005 Derick Faber Germany Berlin 23 Jermyn Street 53424 Remagen

2000006 Joey Bruno Italy Milan 26 Via Sacra 20090 Buccinasco

2000008 Lance Bernard France Lille 19 Avenue Kleber 97100 Basse-Terre

2000009 Liam Williams United Kingdom Cambridge 1 Dunhampton Drive DY19 OU7

2000011 Sergio Falippez Italy Naples 6 Via Giulia 00745 Campania

2000012 Frederick Krause Germany Hamburg 9 Frankfort avenue 23843 Bad Oldesloe

10 rows selected.

SQL> SPOOL OFF

**Student**

SQL> SELECT \* FROM STUDENT;

SNO SFNAME SLNAME GENDER TELNO COMPANYNAME

---------- ------------------ ---------------- ------------ -------------- ------------------

4000001 James Green M 44789644569 Computing Inc

4000002 Milly Stark F 84738395487 Networking Ltd

4000003 Marie Tully F 84755758576 Physics Ltd

4000004 John Crow M 54453534535 Intravenous Inc

4000005 Brandon Dondarrion M 86767443321 Calcualtions Ltd

4000006 Catelyn Fletcher F 97843645384 Prime Ltd

4000007 Ben Tyrell M 43978653478 Blockmakers Inc

4000008 Elizabeth Newey F 74657684334 Organics Ltd

4000009 Tom Smith M 87438957344 Connections Inc

4000010 Eddard Norris M 74389569456 Electric Ltd

10 rows selected.

SQL> SPOOL OFF

**Student Details**

SQL> SELECT \* FROM STUDENT\_DETAILS;

OFFERING\_OFFERING\_ID STUDENT\_SNO EVALUATION GRADE

-------------------- ----------- ---------- -----

3000005 4000004 2 E

3000011 4000007 3 C

3000011 4000002 2 E

3000011 4000005 3 A

3000001 4000001 1 D

3000002 4000006 3 B

3000004 4000003 2 C

3000005 4000010 4 A

3000006 4000008 0 E

3000008 4000005 3 C

3000009 4000002 2 D

3000011 4000004 3 A

12 rows selected.

SQL> SPOOL OFF;

**Constraints**

ALTER TABLE Course

ADD CONSTRAINT "Course PK" PRIMARY KEY ( cNo );

ALTER TABLE Instructor

ADD CONSTRAINT "Instructor PK" PRIMARY KEY ( instructorNo );

ALTER TABLE Offering

ADD CONSTRAINT "Offering PK" PRIMARY KEY ( offering\_id );

ALTER TABLE Student

ADD CONSTRAINT "Student PK" PRIMARY KEY ( sNo );

ALTER TABLE Student\_Details

ADD CONSTRAINT Student\_Details\_\_IDX PRIMARY KEY

( Offering\_offering\_id, Student\_sNo );

ALTER TABLE Offering

ADD CONSTRAINT Course\_Offering FOREIGN KEY

( cNo ) REFERENCES Course ( cNo );

ALTER TABLE Student\_Details

ADD CONSTRAINT FK\_Student\_Details FOREIGN KEY

( Student\_sNo ) REFERENCES Student ( sNo )

ON DELETE CASCADE;

ALTER TABLE Student\_Details

ADD CONSTRAINT FK\_Student\_Offering FOREIGN KEY

( Offering\_offering\_id ) REFERENCES Offering

( offering\_id ) ON DELETE CASCADE;

ALTER TABLE Offering

ADD CONSTRAINT Instructor\_Offering FOREIGN KEY

( instructorNo ) REFERENCES Instructor ( instructorNo );

ALTER TABLE Student

ADD CONSTRAINT Gender\_CK CHECK

(Gender IN ('M', 'F'));

ALTER TABLE Course

ADD CONSTRAINT Cost\_CK CHECK

(COST BETWEEN 300 AND 2000);

ALTER TABLE Course

ADD CONSTRAINT Duration\_CK CHECK

(DURATION IN (1, 2, 3, 4, 5));

ALTER TABLE Offering

ADD CONSTRAINT Loc\_CK CHECK

(Loc IN ('London', 'Rome', 'Birmingham', 'Paris', 'Berlin', 'Milan', 'Lille', 'Cambridge', 'Naples', 'Hamburg'));

ALTER TABLE Student\_Details

ADD CONSTRAINT Evaluation\_CK CHECK

(Evaluation IN (0, 1, 2, 3, 4));

ALTER TABLE Student\_Details

ADD CONSTRAINT Grade\_CK CHECK

(Grade IN ('A', 'B', 'C', 'D', 'E', 'F'));

**View Definition**

SQL> create or replace view courses as

2 select course.cNo, cTitle, duration, cost, offering\_id, startdate, enddate, loc, instructorno

3 from course, offering

4 where course.cNo = offering.cNo;

View created.

SQL> select \* from courses;

CNO CTITLE DURATION COST OFFERING\_ID STARTDATE ENDDATE LOC INSTRUCTORNO

------- ------------------- ---------- ---------- ----------- --------- --------- --------- --------------

1000001 Computer Science 3 730 3000012 24-APR-08 27-APR-08 London 2000012

1000001 Computer Science 3 730 3000001 09-OCT-07 11-OCT-07 London 2000001

1000001 Computer Science 3 730 3000009 12-NOV-11 14-MAR-11 London 2000009

1000003 English Literature 4 1260 3000013 12-AUG-08 15-AUG-08 Naples 2000011

1000003 English Literature 4 1260 3000011 17-MAY-09 20-MAY-09 Naples 2000011

1000004 Mathematics 5 1530 3000008 19-DEC-12 23-DEC-12 Rome 2000008

1000004 Mathematics 5 1530 3000002 22-JUN-12 26-JUN-12 Rome 2000002

1000005 Biology 3 890 3000006 06-JAN-07 08-JAN-07 Paris 2000006

1000005 Biology 3 890 3000004 14-NOV-10 16-NOV-10 Paris 2000004

1000006 Physics 1 510 3000003 01-MAY-13 01-MAY-13 Berlin 2000003

1000006 Physics 1 510 3000005 18-OCT-12 18-OCT-12 Berlin 2000005

11 rows selected.

SQL> SPOOL OFF

**VIEW TESTS**

**1) UPDATES**

**UPDATE ON CNO**

SQL> update courses set cNo = 101 where cNo = 1000001;

update courses set cNo = 101 where cNo = 1000001

\*

ERROR at line 1:

ORA-01779: cannot modify a column which maps to a non key-preserved table

The error occurs as the column cNo is the primary key of one of the tables therefore it is not updateable.

**UPDATE ON CTITLE**

**SQL> update courses set cTitle = 'Computers' where cTitle = 'Computer Science';**

**update courses set cTitle = 'Computers' where cTitle = 'Computer Science'**

**\***

**ERROR at line 1:**

**ORA-01779: cannot modify a column which maps to a non key-preserved table**

The error occurs because the course table this column is mapped to is not key-preserved, this is because in the joined offering table there is no primary key on a column in the table that is referenced in the course table.

**UPDATE ON DURATION**

**SQL> update courses set duration = 1 where duration = 3;**

**update courses set duration = 1 where duration = 3**

**\***

**ERROR at line 1:**

**ORA-01779: cannot modify a column which maps to a non key-preserved table**

The error occurs because the course table this column is mapped to is not key-preserved, this is because in the joined offering table there is no primary key on a column in the table that is referenced in the course table.

**UPDATE ON COST**

**SQL> update courses set cost = 300 where cost = 730;**

**update courses set cost = 300 where cost = 730**

**\***

**ERROR at line 1:**

**ORA-01779: cannot modify a column which maps to a non key-preserved table**

The error occurs because the course table this column is mapped to is not key-preserved, this is because in the joined offering table there is no primary key on a column in the table that is referenced in the course table.

**UPDATE ON OFFERING\_ID**

**SQL> update courses set offering\_id = 312 where offering\_id = 3000012;**

**1 row updated.**

The update was successful because the column being updated is mapped to a key-preserved table this is due to the primary key cNo being referenced in the offering table.

**UPDATE ON STARTDATE**

**SQL> update courses set startdate = '23-APR-08' where startdate = '24-APR-08';**

**1 row updated.**

The update was successful because the column being updated is mapped to a key-preserved table this is due to the primary key cNo being referenced in the offering table.

**UPDATE ON ENDDATE**

**SQL> update courses set enddate = '26-APR-08' where enddate = '27-APR-08';**

**1 row updated.**

The update was successful because the column being updated is mapped to a key-preserved table this is due to the primary key cNo being referenced in the offering table.

**UPDATE ON LOC**

**SQL> update courses set loc = 'Naples' where loc = 'London';**

**3 rows updated.**

The update was successful because the column being updated is mapped to a key-preserved table this is due to the primary key cNo being referenced in the offering table.

**UPDATE ON INSTRUCTORNO**

**SQL> update courses set instructorno = 2000011 where instructorno = 2000012;**

**1 row updated.**

The update was successful because the column being updated is mapped to a key-preserved table this is due to the primary key cNo being referenced in the offering table.

**COURSES TABLE AFTER UPDATES**

**SQL> select \* from courses;**

**CNO CTITLE DURATION COST OFFERING\_ID STARTDATE ENDDATE LOC INSTRUCTORNO**

**---------- ------------------ ---------- ---------- ----------- --------- --------- ---------- ---------------**

**1000001 Computer Science 3 730 312 23-APR-08 26-APR-08 Naples 2000011**

**1000001 Computer Science 3 730 3000001 09-OCT-07 11-OCT-07 Naples 2000001**

**1000001 Computer Science 3 730 3000009 12-NOV-11 14-MAR-11 Naples 2000009**

**1000003 English Literature 4 1260 3000013 12-AUG-08 15-AUG-08 Naples 2000011**

**1000003 English Literature 4 1260 3000011 17-MAY-09 20-MAY-09 Naples 2000011**

**1000004 Mathematics 5 1530 3000008 19-DEC-12 23-DEC-12 Rome 2000008**

**1000004 Mathematics 5 1530 3000002 22-JUN-12 26-JUN-12 Rome 2000002**

**1000005 Biology 3 890 3000006 06-JAN-07 08-JAN-07 Paris 2000006**

**1000005 Biology 3 890 3000004 14-NOV-10 16-NOV-10 Paris 2000004**

**1000006 Physics 1 510 3000003 01-MAY-13 01-MAY-13 Berlin 2000003**

**1000006 Physics 1 510 3000005 18-OCT-12 18-OCT-12 Berlin 2000005**

**11 rows selected.**

**SQL> SPOOL OFF**

**2) INSERTS**

SQL> INSERT INTO COURSES VALUES

2 (1000006, 'Physics', 2, 300, 300045, '12-JAN-13', '13-JAN-13', 'London', 2000012);

INSERT INTO COURSES VALUES

\*

ERROR at line 1:

ORA-01779: cannot modify a column which maps to a non key-preserved table

The error occurs because the courses table is not key-preserved due to the lack of a primary key from the offering table being referenced in the courses table, therefore the values cannot be inserted.

SQL> INSERT INTO COURSES VALUES

2 (1000006, 'Physics', 3, 500, 3000089, '02-MAR-12', '04-MAR-12', 'Berlin', 2000012);

INSERT INTO COURSES VALUES

\*

ERROR at line 1:

ORA-01779: cannot modify a column which maps to a non key-preserved table

SQL> SPOOL OFF

The error occurs because the courses table is not key-preserved due to the lack of a primary key from the offering table being referenced in the courses table, therefore the values cannot be inserted.

**3) DELETES**

SQL> delete from courses where cNo = 1000006;

2 rows deleted.

SQL> select \* from courses;

CNO CTITLE DURATION COST OFFERING\_ID STARTDATE ENDDATE LOC INSTRUCTORNO

-------- ------------------------- ---------- ---------- ----------- --------- --------- -------- ------------

1000001 Computer Science 3 730 3000012 24-APR-08 27-APR-08 London 2000012

1000001 Computer Science 3 730 3000001 09-OCT-07 11-OCT-07 London 2000001

1000001 Computer Science 3 730 3000009 12-NOV-11 14-MAR-11 London 2000009

1000003 English Literature 4 1260 3000011 17-MAY-09 20-MAY-09 Naples 2000011

1000003 English Literature 4 1260 3000013 12-AUG-08 15-AUG-08 Naples 2000011

1000004 Mathematics 5 1530 3000002 22-JUN-12 26-JUN-12 Rome 2000002

1000004 Mathematics 5 1530 3000008 19-DEC-12 23-DEC-12 Rome 2000008

1000005 Biology 3 890 3000004 14-NOV-10 16-NOV-10 Paris 2000004

1000005 Biology 3 890 3000006 06-JAN-07 08-JAN-07 Paris 2000006

9 rows selected.

SQL> SPOOL OFF

The delete was successful because the view contains a primary key on the non key preserved table as well as a key preserved table due to the primary key cNo being referenced in the offering table.

**Materialised View Definition**

**a)**

SQL> create materialized view offerings\_mv

2 build immediate

3 refresh complete on demand

4 enable query rewrite as

5 select cTitle, loc, C.cNo, count(\*) counts

6 from course C, offering O

7 where C.cNo = O.cNo

8 group by cTitle, loc, C.cNo

9 having count(\*) = (select max(count(\*))

10 from offering O1

11 where loc = O.loc

12 group by cNo);

Materialized view created.

**b)**

SQL> select \* from offerings\_mv;

CTITLE LOC CNO COUNTS

------------------------------ ------------------------------ ---------- ----------

Computer Science London 1000001 3

English Literature Naples 1000003 2

Physics Berlin 1000006 2

Mathematics Rome 1000004 2

Biology Paris 1000005 2

SQL> set autotrace on

SQL> select cTitle, loc, C.cNo, count(\*) counts

2 from course C, offering O

3 where C.cNo = O.cNo

4 group by cTitle, loc, C.cNo

5 having count(\*) = (select max(count(\*))

6 from offering O1

7 where loc = O.loc

8 group by cNo);

CTITLE LOC CNO COUNTS

------------------------------ ------------------------------ ---------- ----------

Computer Science London 1000001 3

English Literature Naples 1000003 2

Physics Berlin 1000006 2

Mathematics Rome 1000004 2

Biology Paris 1000005 2

Execution Plan

----------------------------------------------------------

Plan hash value: 2400066672

---------------------------------------------------------------------------------------------

| Id | Operation | Name | Rows | Bytes | Cost (%CPU)| Time |

---------------------------------------------------------------------------------------------

| 0 | SELECT STATEMENT | | 5 | 300 | 3 (0)| 00:00:01 |

| 1 | MAT\_VIEW REWRITE ACCESS FULL| OFFERINGS\_MV | 5 | 300 | 3 (0)| 00:00:01 |

---------------------------------------------------------------------------------------------

Note

-----

- dynamic sampling used for this statement (level=2)

Statistics

----------------------------------------------------------

19 recursive calls

0 db block gets

12 consistent gets

1 physical reads

0 redo size

690 bytes sent via SQL\*Net to client

476 bytes received via SQL\*Net from client

2 SQL\*Net roundtrips to/from client

0 sorts (memory)

0 sorts (disk)

5 rows processed

**c)**

SQL> update course set cTitle = 'Computers' where cTitle = 'Computer Science';

1 row updated.

SQL> set autotrace on

SQL> select cTitle, loc, C.cNo, count(\*) counts

2 from course C, offering O

3 where C.cNo = O.cNo

4 group by cTitle, loc, C.cNo

5 having count(\*) = (select max(count(\*))

6 from offering O1

7 where loc = O.loc

8 group by cNo);

CTITLE LOC CNO COUNTS

------------------------------ ------------------------------ ---------- ----------

English Literature Naples 1000003 2

Physics Berlin 1000006 2

Computers London 1000001 3

Mathematics Rome 1000004 2

Biology Paris 1000005 2

Execution Plan

----------------------------------------------------------

Plan hash value: 433391125

--------------------------------------------------------------------------------------------

| Id | Operation | Name | Rows | Bytes | Cost (%CPU)| Time |

--------------------------------------------------------------------------------------------

| 0 | SELECT STATEMENT | | 1 | 32 | 7 (29)| 00:00:01 |

|\* 1 | FILTER | | | | | |

| 2 | HASH GROUP BY | | 1 | 32 | 7 (29)| 00:00:01 |

| 3 | MERGE JOIN | | 11 | 352 | 6 (17)| 00:00:01 |

| 4 | TABLE ACCESS BY INDEX ROWID| COURSE | 6 | 114 | 2 (0)| 00:00:01 |

| 5 | INDEX FULL SCAN | Course PK | 6 | | 1 (0)| 00:00:01 |

|\* 6 | SORT JOIN | | 11 | 143 | 4 (25)| 00:00:01 |

| 7 | TABLE ACCESS FULL | OFFERING | 11 | 143 | 3 (0)| 00:00:01 |

| 8 | SORT AGGREGATE | | 1 | 13 | 4 (25)| 00:00:01 |

| 9 | SORT GROUP BY | | 1 | 13 | 4 (25)| 00:00:01 |

|\* 10 | TABLE ACCESS FULL | OFFERING | 2 | 26 | 3 (0)| 00:00:01 |

--------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):

---------------------------------------------------

1 - filter(COUNT(\*)= (SELECT MAX(COUNT(\*)) FROM "OFFERING" "O1" WHERE "LOC"=:B1

GROUP BY "CNO"))

6 - access("C"."CNO"="O"."CNO")

filter("C"."CNO"="O"."CNO")

10 - filter("LOC"=:B1)

Statistics

----------------------------------------------------------

122 recursive calls

0 db block gets

129 consistent gets

18 physical reads

0 redo size

672 bytes sent via SQL\*Net to client

476 bytes received via SQL\*Net from client

2 SQL\*Net roundtrips to/from client

7 sorts (memory)

0 sorts (disk)

5 rows processed

SQL> exec dbms\_mview.refresh('offerings\_mv');

PL/SQL procedure successfully completed.

SQL> set autotrace on

SQL> select cTitle, loc, C.cNo, count(\*) counts

2 from course C, offering O

3 where C.cNo = O.cNo

4 group by cTitle, loc, C.cNo

5 having count(\*) = (select max(count(\*))

6 from offering O1

7 where loc = O.loc

8 group by cNo);

CTITLE LOC CNO COUNTS

------------------------------ ------------------------------ ---------- ----------

English Literature Naples 1000003 2

Physics Berlin 1000006 2

Computers London 1000001 3

Mathematics Rome 1000004 2

Biology Paris 1000005 2

Execution Plan

----------------------------------------------------------

Plan hash value: 2400066672

---------------------------------------------------------------------------------------------

| Id | Operation | Name | Rows | Bytes | Cost (%CPU)| Time |

---------------------------------------------------------------------------------------------

| 0 | SELECT STATEMENT | | 5 | 300 | 3 (0)| 00:00:01 |

| 1 | MAT\_VIEW REWRITE ACCESS FULL| OFFERINGS\_MV | 5 | 300 | 3 (0)| 00:00:01 |

---------------------------------------------------------------------------------------------

Note

-----

- dynamic sampling used for this statement (level=2)

Statistics

----------------------------------------------------------

0 recursive calls

0 db block gets

44 consistent gets

7 physical reads

0 redo size

672 bytes sent via SQL\*Net to client

476 bytes received via SQL\*Net from client

2 SQL\*Net roundtrips to/from client

6 sorts (memory)

0 sorts (disk)

5 rows processed

**d)**

There are several ways a materialised view can be refreshed, either refreshing on commit which keeps the materialised view up to date but can affect the performance on base table(s) transactions or refreshing on demand which gives the user freedom to refresh at their discretion but it does require a procedure to be manually invoked each time the user wishes to refresh instead of the DBMS handling this. Whether on commit or demand refreshing can be specified to be fast which is where only the rows in the material view that are affected by changes to the base table(s) which makes this quicker than a complete refresh however it requires the presence or materialized view logs. Refreshing can also be specified to be complete which is where the materialized view is completely rebuilt which is slower than a fast refresh but it does not require materialized view logs. Also refreshing can be specified to be forced which is where the refresh will be fast if there are materialized logs present but if not it forces a complete refresh.

**Trigger Definitions**

**a)**

**b)**

**c)**

SQL> create or replace trigger course\_date

2 before delete

3 on offering

4 for each row

5 begin

6 if :old.startdate <= sysdate then

7 raise\_application\_error(-20100,

8 'Courses already started cannot be removed');

9 end if;

10 end;

11 /

Trigger created.

SQL> delete from offering where startdate = '19-DEC-12';

delete from offering where startdate = '19-DEC-12'

\*

ERROR at line 1:

ORA-20100: Courses already started cannot be removed

ORA-06512: at "OPS$1007022.COURSE\_DATE", line 3

ORA-04088: error during execution of trigger 'OPS$1007022.COURSE\_DATE'

SQL> delete from offering where startdate = '01-MAY-13';

1 row deleted.

SQL> SPOOL OFF

**http://msdn.microsoft.com/en-us/library/aa258254%28v=sql.80%29.aspx**

**Query Output**

a)

SQL> select C.cNo, C.cTitle, C.Cost

2 from Course C, Offering O

3 where C.cNO = O.cNo and O.loc = 'London';

CNO CTITLE COST

---------- ------------------------------ ----------

1000001 Computer Science 730

SQL> spool off

b)

SQL> select loc, cNo, count(\*) counts

2 from offering O

3 group by loc, cNo

4 having count(\*) = (select max(count(\*))

5 from offering O1

6 where loc = O.loc

7 group by cNo);

LOC CNO COUNTS

------------------------------ ---------- ----------

Rome 1000004 2

Paris 1000005 2

London 1000001 3

Naples 1000003 2

Berlin 1000006 2

SQL> SPOOL OFF

c)

SQL> select O.cNo, O.offering\_id, S.student\_sno

2 from offering O, student\_details S

3 where O.offering\_id = S.offering\_offering\_id

4 order by cNo;

CNO OFFERING\_ID STUDENT\_SNO

---------- ----------- -----------

1000001 3000001 4000001

1000001 3000009 4000002

1000003 3000011 4000002

1000003 3000011 4000005

1000003 3000011 4000004

1000003 3000011 4000007

1000004 3000002 4000006

1000004 3000008 4000005

1000005 3000004 4000003

1000005 3000006 4000008

1000006 3000005 4000004

1000006 3000005 4000010

12 rows selected.

d)

SQL> select offering\_offering\_id, student\_sno, case evaluation

2 when 0 THEN 'TERRIBLE'

3 WHEN 1 THEN 'BAD'

4 WHEN 2 THEN 'MEDIOCRE'

5 WHEN 3 THEN 'FAIR'

6 WHEN 4 THEN 'GOOD'

7 ELSE 'OTHER'

8 END

9 FROM STUDENT\_DETAILS;

OFFERING\_OFFERING\_ID STUDENT\_SNO CASEEVAL

-------------------- ----------- --------

3000005 4000004 MEDIOCRE

3000011 4000007 FAIR

3000011 4000002 MEDIOCRE

3000011 4000005 FAIR

3000001 4000001 BAD

3000002 4000006 FAIR

3000004 4000003 MEDIOCRE

3000005 4000010 GOOD

3000006 4000008 TERRIBLE

3000008 4000005 FAIR

3000009 4000002 MEDIOCRE

3000011 4000004 FAIR

12 rows selected.

SQL> SPOOL OFF

e)

SQL> select S.sno, S.sfname, S.slname, C.cTitle

2 from student S, course C, offering O, student\_details S1

3 where S1.offering\_offering\_id = O.offering\_id and C.cNo = O.cNo and

S.sno = S1.student\_sno;

SNO SFNAME SLNAME CTITLE

---------- ----------- ------------------ ------------------------------

4000001 James Green Computer Science

4000002 Milly Stark English Literature

4000002 Milly Stark Computer Science

4000003 Marie Tully Biology

4000004 John Crow English Literature

4000004 John Crow Physics

4000005 Brandon Dondarrion English Literature

4000005 Brandon Dondarrion Mathematics

4000006 Catelyn Fletcher Mathematics

4000007 Ben Tyrell English Literature

4000008 Elizabeth Newey Biology

4000010 Eddard Norris Physics

12 rows selected.

SQL> SPOOL OFF

f)

SQL> select cTitle, cost, (select avg(cost) from course) as difference

2 from course

3 group by cTitle, cost;

CTITLE COST DIFFERENCE

------------------------------ ---------- ----------

Computer Science 730 895.833333

Mathematics 1530 895.833333

Computing 455 895.833333

English Literature 1260 895.833333

Biology 890 895.833333

Physics 510 895.833333

6 rows selected.

SQL> SPOOL OFF

g)

SQL> select S.sno, S.sfname, S.slname, C.cTitle

2 from student S, course C, offering O, student\_details S1

3 where S1.offering\_offering\_id = O.offering\_id and C.cNo = O.cNo and

S.sno = S1.student\_sno and O.loc != 'Rome';

SNO SFNAME SLNAME CTITLE

---------- ----------------------- --------------- ------------------------

4000001 James Green Computer Science

4000002 Milly Stark English Literature

4000002 Milly Stark Computer Science

4000003 Marie Tully Biology

4000004 John Crow English Literature

4000004 John Crow Physics

4000005 Brandon Dondarrion English Literature

4000007 Ben Tyrell English Literature

4000008 Elizabeth Newey Biology

4000010 Eddard Norris Physics

10 rows selected.

SQL> SPOOL OFF

h)

i)

SQL> select c.cTitle, c.cNO, O.startdate, O.enddate, O.loc

2 from course C, offering O

3 where C.cNo = O.cNo and startdate between '01-SEP-12' and '18-APR-13';

CTITLE CNO STARTDATE ENDDATE LOC

------------------------------ ---------- --------- --------- -------------

Physics 1000006 18-OCT-12 18-OCT-12 Berlin

Mathematics 1000004 19-DEC-12 23-DEC-12 Rome

SQL> SPOOL OFF

j)

SQL> select S.sfname, S.slname

2 from student S, offering O, student\_details S1

3 where S.gender = 'F' and S.sNo = S1.student\_sNo and O.offering\_id =

S1.offering\_offering\_id and O.startdate between '01-JAN-07' and '31-

DEC-07';

SFNAME SLNAME

------------------------------ ------------------------------

Elizabeth Newey

SQL> SPOOL OFF

k)