Orcl5\_17

Name: DatabaseOracleJan20

1. Common questions about what is DDL. PL/SQL (same concepts in Microsoft SQL and how they are applied on Oracle)
2. For Lab1A, create table process as found in printed documents
3. Slide 13 (important)
4. Oracle naming constraints
5. There are things you can’t use in names
6. All oracle names begin with alpha character
7. It cannot be an sql reserve word
8. Check slide 15, about char and varchar
9. Bottom of slide 14, gender char.
10. Turning to numbers on Oracle (slide 16) to do with precision and scale)
11. So go through numbers & string (char & varchar) and how they are treated in Oracle
12. How we treat integers too
13. Slide 20 about date in oracle
14. Oracle date format
15. Constraints on oracle (integrity and value)
16. Integrity are just PK & FK
17. Value?
18. How null values are treated
19. Slide 25, you don’t delete a table, you drop it. Goes into a recycle bin
20. ‘purge’
21. Sequence slide 29 (not the same as @@identity in MS SQL
22. Slide 30, syntax, if you start with a +ve number, you increment by a +ve number
23. No cache
24. Slide 31
25. Slide 32 scripts
26. When creating a table, make sure you name your constraint (check on the slide on naming convection for constraint name)
27. Look at your ‘in’ operator in the slides
28. CREATE TABLE MM\_STUDENT
29. ( SID NUMBER (8.,0) CONSTRAINT PK\_MMSTUDENT\_SID PRIMARY KEY
30. CONSTRAINT NN\_MMSTUDENT\_SID NOT NULL,
31. SNAME VARCHAR2(50) DEFAULT ‘UNKNOWN’
32. CONSTRAINTS N\_MMSTUDENT\_SNAME NULL,
33. GENDER CHAR(1) CONSTRAINT CK\_MMSTUDENT\_GENDER\_MFN
34. CHECK(GENDER IN (‘M’, ‘F’, ‘N’))
35. CONSTRAINT NN\_MMSTUDENT\_GENDER NOT NULL,
36. EDATE DATE DEFAULT SYSDATE
37. CONSTRAINT NN\_MMSTUDENT\_EDATE NOT NULL
38. );

20200110 Class

Regular expressions (Check the file regular expressions on moodle)

For example, Microsoft regular expression might be Postal\_Code like ‘[a-z][0-9]….’ While Oracle is REGEXP\_LIKE(POSTAL\_CODE,’[A-Z][0-9]…’)

Hint; characters for start ^ and the $ for end are important

Read this about creating a table with both foreign key and primary key

create table MM\_student

( sid number(8,0) Constraint PK\_mmstudent\_sid PRIMARY KEY

Constraint NN\_mmstudent\_sid NOT NULL,

sname varchar2(50) default 'UNKNOWN'

Constraint N\_mmstudent\_sname NULL,

gender char(1) Constraint CK\_mmstudent\_gender\_MFN

Check(gender in ('M', 'F', 'N'))

Constraint NN\_mmstudent\_gender NOT NULL,

edate date default sysdate

Constraint NN\_mmstudent\_edate NOT NULL

);

create table MM\_course

( cid char(8) Constraint PK\_mmcourse\_cid PRIMARY KEY

Constraint NN\_mmcourse\_cid NOT NULL,

cname varchar2(50)

Constraint N\_mmcourse\_cname NULL,

location varchar2(20)

Constraint N\_mmcourse\_location NULL,

ccost number(6,2) default '575.00'

Constraint N\_mmcourse\_ccost NULL

);

create table MM\_grade

( sid number(8,0) Constraint FK\_MMGRADE\_SID FOREIGN KEY REFERENCES MMSTUDENT (SID)

CONSTRAINT NN\_MMGRADE\_SID NOT NULL,

CID CHAR(8) CONSTRAINT FK\_MMCOURSE\_CID FOREIGN KEY REFERENCES MMCOURSE (CID)

CONSTRAINT NN\_MMGRADE\_SID NOT NULL,

CONSTRAINT MMGrade\_SID\_CID\_PK PRIMARY KEY (SID, CID)

MARK NUMBER(5,2) Constraint CK\_mmGRADE\_MARK\_

CHECK ((MARK > 0) AND (MARK <100))

CONSTRAINT NN\_MMGRADE\_MARK NOT NULL

);

Class 20200113

1. Create stored procedure NB: This procedure has no error checks so no error message for this example. P stands for parameter

CREATE OR REPLACE PROCEDURE PR\_UPDATE\_MARK1

(P\_SID NUMBER, P\_CID CHAR, P\_MARK NUMBER)

AS

BEGIN

UPDATE MM\_GRADE

SET MARK = P\_MARK

WHERE SID = P\_SID AND

CID = P\_CID;

END PR\_UPDATE\_MARK1;

/

SHOW ERRORS;

1. Slide 7 (B-2) there is a command called SHOW ERROR; which shows some errors
2. ANOTHER procedure
3. NOTE BELOW HOW INPUT PARAMETERS ARE TREATED VS HOW LOCAL VARIABLES ARE TREATED
4. CREATE OR REPLACE PROCEDURE PR\_UPDATE\_MARK2

(P\_SNAME VARCHAR2, P\_CNAME VARCHAR2, P\_MARK NUMBER)

AS

V\_SID NUMBER(8,0);

V\_CID CHAR(8);

BEGIN

SELECT SID

INTO V\_SID

FROM MM\_STUDENT

WHERE SNAME = P\_SNAME;

SELECT CID

INTO V\_CID

FROM MM\_COURSE

WHERE CNAME = P\_CNAME;

UPDATE MM\_GRADE

SET MARK = P\_MARK

WHERE SID = V\_SID AND

CID = V\_CID;

END PR\_UPDATE\_MARK2;

/

SHOW ERRORS;

1. How do you run this procedure? Slide 18 has ‘EXEC PR\_ADD\_... (example of an execute command)
2. Can the procedure above be done without the local variables? Yes, by using subqueries
3. A subquery is a select! Does not have the where clause
4. But function might cause errors. Not advised

20200117

1. SQL PLUS; Used for formatting your data from SQL to a text editor (like how you will do in lab 1b) each line below on the code is line by line to show how you can format your data
2. Try this
3. Set Pagesize 99

Set Linesize 80

COLUMN Portfolio\_Number Format A9 Heading “Portfolio/Number”

COLUMN transaction)date format a20 heading “Transaction/Date”

COLUMN price\_per\_share format a10 heading “Price/Per/Share”

COLUMN exchange\_code format a8 heading “exchange/code”

COLUMN quantity format a8 heading “Quantity”

Select Portfolio\_Number,

To\_char(transaction\_date, ‘yyyy-mon-dd hh24:mi:ss’) Transaction\_Date,

Stock\_code, exchange code,

Broker\_number, Buy\_sell, quantity, to\_char(Quantity, ‘9,999,999’) Quantity,

To\_char(price\_per\_share, “$9,999.99’) Price\_per\_share

From transaction

Where Portfolio\_Number>535

1. Play with 80 & 99 to manage how your data will be displayed
2. HINT: The where clause in Lab 1b is crucial
3. 20200122
4. DMIT2019
5. CREATE OR REPLACE PACKAGE PKG\_STUDENT
6. AS
7. FUNCTION PKG\_FN\_GET\_EDATE
8. (P\_SID NUMBER)
9. RETURN DATE;
10. FUNCTION PKG\_FN\_GET\_EDATE
11. (P\_SNAME VARCHAR2)
12. RETURN DATE;
13. END PKG\_STUDENT;
14. /
15. SHOW ERRORS;
16. CREATE OR REPLACE PACKAGE BODY PKG\_STUDENT
17. IS
18. FUNCTION PKG\_FN\_GET\_EDATE
19. (P\_SID NUMBER)
20. RETURN DATE
21. IS
22. V\_EDATE DATE;
23. BEGIN
24. SELECT EDATE -----every select statement in Oracle must have an into clause
25. INTO V\_EDATE
26. FROM MM\_STUDENT
27. WHERE SID=P\_SID;
28. RETURN V\_EDATE;
29. END PKG\_FN\_GET\_EDATE;
30. FUNCTION PKG\_FN\_GET\_EDATE
31. (P\_SNAME VARCHAR2)
32. RETURN DATE
33. IS V\_EDATE DATE;
34. BEGIN
35. SELECT EDATE
36. INTO V\_EDATE
37. FROM MM\_STUDENT
38. WHERE SNAME = P\_SNAME;
39. RETURN V\_EDATE;
40. END PKG\_FN\_GET\_EDATE;
41. END PKG\_STUDENT;
42. /
43. SHOW ERRORS;
44. CURSORS
45. January 24th, 2020
46. Given 3 tables (Same format as the mickey mouse ERD)

|  |
| --- |
| **MM.GRADE** |
| SID |
| CID |
| MARK |

|  |
| --- |
| **MM.HELP** |
| CID |
| MARK |
| SID |

|  |
| --- |
| **MM.HONORS** |
| MARK |
| SID |
| CID |

1. We will create a table in which if the mark is less than 50, it will go into the help table. If it is more than 80, it will go into honours table
2. Start now;
3. CREATE OR REPLACE PROCEDURE PR\_POPULATE\_NEW\_TABLES
4. IS
5. V\_CID CHAR(8);
6. V\_SID NUMBER(8,0);
7. V\_MARK NUMBER(5,2);
8. CURSOR C\_GRADES IS SELECT CID, SID, MARK
9. FROM MM\_GRADE
10. WHERE MARK < 50 OR MARK>= 80;
11. BEGIN
12. OPEN C\_GRADES; --open cursor
13. FETCH C\_GRADES INTO V\_CID, V\_SID, V\_MARK;
14. WHILE C\_GRADES%FOUND LOOP
15. IF V\_MARK < 50 THEN
16. INSERT INTO MM.HELP
17. (CID, SID, MARK)---where there is an insert statement, there must be a column list!
18. VALUES
19. (V\_CID, V\_SID, V\_MARK);
20. ELSE
21. INSERT INTO MM.HONOURS
22. (CID, SID, MARK)
23. VALUES
24. (V\_CID, V\_SID, V\_MARK);
25. END IF;
26. FETCH C\_GRADES INTO V\_CID, V\_SID, V\_MARK;
27. END LOOP;
28. CLOSE C\_GRADES;
29. END PR\_POPULATE\_NEW\_TABLES;
30. /
31. SHOW ERRORS;
32. **Key:**
33. Double check if you have an infinite loop, you will notice it using the ‘fetch’. Make sure you have if statement closed, is the fetch inside your loop or outside?
34. NOTE, 4 Parts it has, declare, open, fetch, close
35. January 27th, 2019
36. **Review Lab 1A**
37. **More cursors in a function.**
38. **What does the function do? Returns a course name and marks for a student**

|  |
| --- |
| **MM.STUDENT** |
| SID |
| SNAME |
| GENDER |
| EDATE |

|  |
| --- |
| **MM.STUDENT** |
| CID |
| CNAME |
| CCOST |
| LOCATION |

|  |
| --- |
| **MM.STUDENT** |
| SID |
| CID |
| MARK |

1. CREATE OR REPLACE FUNCTION FN\_SHOW COURSE\_NAMES\_AND\_MARKS
2. (P\_SID NUMBER)
3. RETURN VARCHAR2
4. AS
5. V\_CNAME VARCHAR2(50);
6. V\_MARK NUMBER (5,2);
7. V\_OUTPUT VARCHAR2 (1500);
8. CURSOR C\_NAMES IS SELECT MMCOURSE.CNAME, MMGRADE.MARK
9. FROM MMCOURSE, MMGRADE
10. WHERE MMGRADE.SID = P\_SID AND MMGRADE.CID = MMCOURSE.CID;
11. BEGIN
12. OPEN C\_CNAMES;
13. FETCH C\_NAMES INTO V\_CNAME. V\_MARK;
14. IF C\_CNAMES%NOTFOUND THEN
15. V\_OUTPUT := ‘THE STUDENT’

**EXCEPTIONS IN THE FUNCTION**

**Class January 29th**

Get Edate from same tables as above but using sname

1. CREATE OR REPLACE FUNCTION FN\_GET\_EDATE1
2. (P\_SNAME VARCHAR2)
3. RETURN DATE
4. AS
5. V\_EDATE DATE;
6. BEGIN
7. SELECT EDATE
8. INTO V\_EDATE
9. FROM MM\_STUDENT
10. WHERE SNAME = P\_SNAME;
11. RETURN V\_EDATE;
12. END FN\_GET\_EDATE1;
13. /
14. SHOW ERRORS;

QUESTION: What if there is no student record we are looking for

1. CREATE OR REPLACE FUNCTION FN\_GET\_EDATE2
2. (P\_SNAME VARCHAR2)
3. RETURN DATE
4. AS
5. V\_EDATE DATE;
6. v\_count number(1);
7. BEGIN
8. SELECT NVL(count(sid),0)
9. INTO V\_count
10. FROM MM\_STUDENT
11. WHERE SNAME = P\_SNAME;
12. If v\_count= 1 then
13. Select edate
14. Into v\_edate
15. From mm\_student
16. Where lower(sname) = lower(p\_sname);
17. Else if v\_count = 0 then
18. V\_edate := TO\_DATE(’31-12-9999’, ‘DD-MM-YYYY’);
19. Else
20. V\_edate := TO\_DATE(’01-01-0001’,’DD-MM-YYYY’);
21. End if;

RETURN V\_EDATE;

1. END FN\_GET\_EDATE2;
2. /
3. SHOW ERRORS;
4. Above is about Exceptions in your slide. Look for types of exceptions we have

Another example (you need 3 different dates when dealing with dates)

1. CREATE OR REPLACE FUNCTION FN\_GET\_EDATE3
2. (P\_SNAME VARCHAR2)
3. RETURN DATE
4. AS
5. V\_EDATE DATE;
6. BEGIN
7. SELECT EDATE
8. INTO V\_EDATE
9. FROM MM\_STUDENT
10. WHERE lower(SNAME) = lower(P\_SNAME);
11. RETURN V\_EDATE;
12. EXCEPTION
13. WHEN NO\_DATE\_FOUND THEN
14. RETURN TO\_DATE(’31-12-9999’, ‘DD-MM-YYYY’);
15. WHEN TOO\_MANY\_ROWS THEN
16. RETURN TO\_DATE(’01-01-00001’, ‘DD-MM-YYYY’);
17. WHEN OTHERS THEN
18. RETURN TO\_DATE(’04-03-0403’, ‘DD-MM-YYYY’);
19. END FN\_GET\_EDATE3;
20. /
21. SHOW ERRORS;
22. NB: year 9999 actually means no data found and 0101 means too many data as above

TRIGGERS

FEBRUARY 3RD, 2020

|  |
| --- |
| **MM\_STUDENT** |
| SID |
| SNAME |
| GENDER |
| EDATE |
|  |
| **MM\_COURSE** |
| CID |
| CNAME |
| CCOST |
| LOCATION |
|  |
| **MM\_GRADE** |
| SID |
| CID |
| MARK |

NB: do not put transaction commit (rollback etc.) on your triggers

We have this

INSERT INTO MM\_GRADE

(SID, CID, MARK)

VALUES

(12345, ‘BC5436’, 99);

(STORY IS NAIT used to have an old course that starts with ‘BCS’ (upper case) so we put this trigger that everytime there is an insert or update message will be raised. NOTE THAT we did not put trigger on table MM.Course because this trigger is about new records. A current student should not have old records

CREATE OR REPLACE TRIGGER TR\_BIUR\_MMGRADE\_NCS\_COURSES\_BAD

BEFORE INSERT

ON MM\_GRADE

FOR EACH ROW

BEGIN

IF UPPER (SUBTR(:NEW.CID, 1,3)) = ‘BCS’ THEN

RAISE\_APPLICATION\_ERROR (-20099, ‘BCS COURSES ARE NOT CURRENT’);

END IF;

END TR\_BIUR\_MM\_GRADE\_BCS\_COURSES\_BAD;

/

SHOW ERRORS;