Use the COURSE and PARTICIPANT tables created in earlier tutorial.

The concat() function and concatenation operator ||

Assume that a query based on above tables should generate a output -

participant_name **takes course in** course_name **of** credit **credits.**

- **Using concat operator, ||**

```
SELECT PNAME || ' takes course in ' || CNAME || ' of ' || CREDIT || ' credits.'
    FROM COURSE C JOIN PARTICIPANT P
    ON C.CID = P.CID;
```

- **Using concat() function**

```
SELECT CONCAT(PNAME, CONCAT(' takes course in ',
    CONCAT( CNAME, CONCAT(' of ', CONCAT(CREDIT, ' credits.')))))
    FROM COURSE C JOIN PARTICIPANT P
    ON C.CID = P.CID;
```

- **Generated column with user-defined column-name**

```
SELECT PNAME || ' takes course in ' || CNAME || ' of ' || CREDIT || ' credits.'
        AS "Participant --> Course"
    FROM COURSE C JOIN PARTICIPANT P
    ON C.CID = P.CID;
```

Record Deletion in Parent-Child Relationship [Foreign Key]

Establishing a parent-child relationship between two relations [or tables] necessitates inclusion of a foreign key in child [or dependent] relation referencing either a primary key or unique key in parent table.

Deletion of a child table record is pretty straight forward and is always valid. But when a record from a parent table is deleted (which has assocoaited child records), it necessitates defining alternatives to dealing with anticipated situations. Following situations are defined -

1. Delete assocated child records [CASCADE].
2. Set the foreign key attribute value as null [SET NULL].
3. Set the foreign key attribute value to a default value [SET DEFAULT].
4. Do nothing, do not allow such deletion [NO ACTION]

In Oracle, when a foreign key is defined without ON DELETE clause, the default action for deleting a parent record is set to NO ACTION.

```
DELETE FROM PARTICIPANT
    WHERE CID = 101;
```

```
DELETE FROM COURSE
    WHERE CID = 101;
```

```
ERROR at line 1:
ORA-02292: integrity constraint (ABC.PARTICIPANT_FK_COURSE_CID)
violated - child record found
```

To check the ON DELETE action for a foreign key, the USER_CONSTRAINTS view may be queried as below -

```
DESC USER_CONSTRAINTS;
```

Let us modify for some columns, the column heading and column width.

```
COLUMN TABLE_NAME HEADING "Table" FORMAT A15
COLUMN CONSTRAINT_NAME HEADING "Constraint" FORMAT A25
COLUMN CONSTRAINT_TYPE HEADING "Typ" FORMAT A3
COLUMN DELETE_RULE HEADING "On Delete" FORMAT A10

SELECT TABLE_NAME, CONSTRAINT_NAME,
        CONSTRAINT_TYPE, DELETE_RULE
    FROM USER_CONSTRAINTS
    WHERE TABLE_NAME IN ('COURSE','PARTICIPANT')
        AND UPPER(CONSTRAINT_NAME) LIKE '%_FK_%';
```

Let us change the FK action for PARTICIPANT table to allow for associated dependent [or child] records when a parent record is removed.

```
ALTER TABLE PARTICIPANT
    DROP CONSTRAINT PARTICIPANT_FK_COURSE_CID;

ALTER TABLE PARTICIPANT
    ADD CONSTRAINT PARTICIPANT_FK_COURSE_CID
        FOREIGN KEY (CID) REFERENCES COURSE(CID)
        ON DELETE CASCADE;

SELECT TABLE_NAME, CONSTRAINT_NAME,
        CONSTRAINT_TYPE, DELETE_RULE
    FROM USER_CONSTRAINTS
    WHERE CONSTRAINT_NAME LIKE 'PARTICIPANT_FK%';
```

Now, remove a record from COURSE and observe the effect in PARTICIPANT. Before executing the DELETE statement, list the contents of COURSE and PARTICIPANT. On observing, the results, undo the changes [use ROLLBACK].

Let us remove a course with course ID = 101.

```
SELECT * FROM COURSE;
```

```
SELECT * FROM PARTICIPANT
    WHERE CID = 101;

DELETE FROM COURSE
    WHERE CID = 101;

SELECT * FROM PARTICIPANT
    WHERE CID = 101;

ROLLBACK;
```

Now, check DELETE_RULE to find whether it has been reverted to NO ACTION. You will notice that it isn't [as it was a DDL which brought about this change].

```
SELECT TABLE_NAME, CONSTRAINT_NAME,
        CONSTRAINT_TYPE, DELETE_RULE
    FROM USER_CONSTRAINTS
    WHERE CONSTRAINT_NAME LIKE 'PARTICIPANT_FK%';
```

Meanwhile, organization policy change ocwurs, that allows for parent record deletion in presence of associated child records. While allowing such removals, the child record FK attribute should be set to null.
Maintain your schema to enforce the policy change.

```
ALTER TABLE PARTICIPANT
    DROP CONSTRAINT PARTICIPANT_FK_COURSE_CID;

ALTER TABLE PARTICIPANT
    ADD CONSTRAINT PARTICIPANT_FK_COURSE_CID
        FOREIGN KEY (CID) REFERENCES COURSE(CID)
        ON DELETE SET NULL;

SELECT TABLE_NAME, CONSTRAINT_NAME,
        CONSTRAINT_TYPE, DELETE_RULE
    FROM USER_CONSTRAINTS
    WHERE CONSTRAINT_NAME LIKE 'PARTICIPANT_FK%';
```

Note: Oracle does not support SET DEFAULT in ON DELETE clause.

Let us observe the effect while we remove a course with course ID = 101.

```
DELETE FROM COURSE
    WHERE CID = 101;
SELECT * FROM PARTICIPANT
    WHERE CID = 101;
ROLLBACK;
```

JOINS

Join is presumably the most expensive operation of database. It involves cartesian product over the argument relations and applying a qualifying condition to produce the result-set. Oracle has following alternative to implementing different joins -

- Cross Join or Cartesian Product
- Natural Join, Equi Joins and Theta Joins
- Outer Joins – Left Outer, Right Outer and Full Outer

[Refer class notes for detailed discussion on Joins]

Let us create two tables, PARTICIPANT_1 and PARTICIPANT_2 as below -

```
CREATE TABLE PARTICIPANT_1
      AS SELECT * FROM PARTICIPANT
            WHERE PID IN (1002, 1004, 1005);

CREATE TABLE PARTICIPANT_2
      AS SELECT * FROM PARTICIPANT
            WHERE PID IN (1001, 1003, 1004, 1005);

SELECT * FROM PARTICIPANT_1;
SELECT * FROM PARTICIPANT_2;
```

The Cartesian Product or Cross Join

- The Oracle 11g Way

```
SELECT *
   FROM PARTICIPANT_1 CROSS JOIN PARTICIPANT_2;
```

- The Legacy Way

```
SELECT *
   FROM PARTICIPANT_1, PARTICIPANT_2;
```

The Equi Join

- The Oracle 11g Way

```
SELECT *
   FROM PARTICIPANT_1 P1 JOIN PARTICIPANT_2 P2
   ON P1.PID = P2.PID;
```

- The Legacy Way

```
SELECT *
   FROM PARTICIPANT_1 P1, PARTICIPANT_2 P2
   WHERE P1.PID = P2.PID;
```

The Natural Join

```
SELECT *
   FROM PARTICIPANT_1 NATURAL JOIN PARTICIPANT_2;
```

The Left Outer Join

- The Oracle 11g Way

```
SELECT *
   FROM PARTICIPANT_1 P1 LEFT OUTER JOIN PARTICIPANT_2 P2
   ON P1.PID = P2.PID;
```

- The Legacy Way

```
SELECT *
   FROM PARTICIPANT_1 P1, PARTICIPANT_2 P2
   WHERE P1.PID = P2.PID (+);
```

The Right Outer Join

- The Oracle 11g Way

```
SELECT *
   FROM PARTICIPANT_1 P1 RIGHT OUTER JOIN PARTICIPANT_2 P2
   ON P1.PID = P2.PID;
```

- The Legacy Way

```
SELECT *
   FROM PARTICIPANT_1 P1, PARTICIPANT_2 P2
   WHERE P1.PID (+) = P2.PID;
```

The Full Outer Join

- The Oracle 11g Way

```
SELECT *
   FROM PARTICIPANT_1 P1 FULL OUTER JOIN PARTICIPANT_2 P2
   ON P1.PID = P2.PID;
```

- The Legacy Way [not avaialble]

```
SELECT *
   FROM PARTICIPANT_1 P1, PARTICIPANT_2 P2
   WHERE P1.PID (+) = P2.PID (+);
```

```
ERROR at line 3:
ORA-01468: a predicate may reference only one outer-joined table
```

SET OPERATIONS – UNION, DIFFERENCE, INTERSECTION

These realtional (set based) operations are implmented in Oracle as -

- Set Union [UNION]
- Set Difference or Minus or Except [MINUS]
- Set Intersection [INTERSECT]

The argument relations MUST be union-compatible or type-compatible.

The Set Union

```
SELECT * FROM PARTICIPANT_1
UNION
SELECT * FROM PARTICIPANT_2;
```

The Set Difference

```
SELECT * FROM PARTICIPANT_1
MINUS
SELECT * FROM PARTICIPANT_2;
```

The Set Intersection

```
SELECT * FROM PARTICIPANT_1
INTERSECT
SELECT * FROM PARTICIPANT_2;
```

The Set Intersection using Difference

```
SELECT * FROM PARTICIPANT_1
MINUS (
      SELECT * FROM PARTICIPANT_1
      MINUS
      SELECT * FROM PARTICIPANT_2
);
```

The Full Outer Join using Union

```
SELECT *
      FROM PARTICIPANT_1 P1, PARTICIPANT_2 P2
      WHERE P1.PID (+) = P2.PID
UNION
SELECT *
      FROM PARTICIPANT_1 P1, PARTICIPANT_2 P2
      WHERE P1.PID = P2.PID (+);
```

TRY: Reexecute the Set Union query replacing UNION by UNION ALL to observe effect.

Run appropriate DMLs to ensure the COURSE and PARTICIPANT to contain tuples as shown below -

SELECT * FROM COURSE;

CID	CNAME	CREDIT
103	Computer Networks Management	3
104	Algorithms and Programming	4
101	Database Management Systems	5
102	Object-Oriented Systems	4

SELECT * FROM PARTICIPANT;

PID	PNAME	G	CID
1007	Rafael Nadal	M	103
1008	Roger Federer	M	104
1006	Serena Williams	F	101
1009	Gabriela Sabatini	F	102
1011	Maria Sharapova	F	104
1010	Sindhu P V	F	103
1012	Gopichand Pullela	M	104
1001	Albert DCosta	M	101
1002	Hitman Rohit	M	102
1003	Maria Anderson	F	102
1004	Pamela Smith	F	101
1005	Indiana Jones	M	

Computed Columns and User Defined Column Headings

- For each participant, its name, the course registered and hours per week. Hours per week will be referred as "Lectures" and computed as Credits – 1.

```
SELECT PNAME, CNAME, CREDIT-1 "Lectures"
    FROM COURSE C JOIN PARTICIPANT P
    ON C.CID = P.CID;
```

- List "Term Fee" for each participant to be labelled as "Name of Attendee". Term Fee is computed as credits * 200 – 100.

```
SELECT PNAME "Name of Attendee",CREDIT*200-100 "Term Fee"
    FROM COURSE C JOIN PARTICIPANT P
    ON C.CID = P.CID;
```