F24-95736
Advanced Database Management
Professor: Randy Trzeciak

In this lab, we shall learn how to do the following:

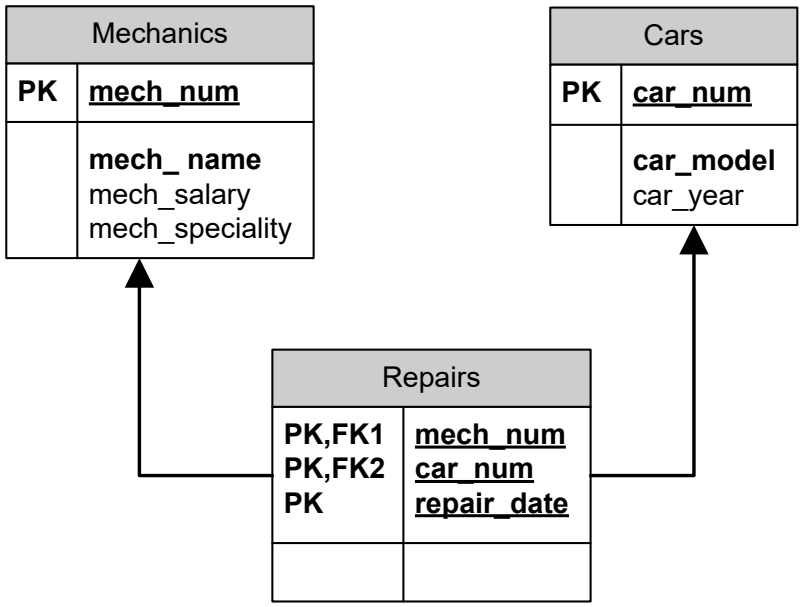
1. Learn how to implement **static business rules** from user requirements that ensure the integrity of business data.
2. Learn how to implement **dynamic business rules** from user requirements that can **automate** certain business processes.
3. Learn how to create and use triggers on the **Oracle server** that are applicable to **all** client applications (whether they are SQL, Oracle Application Express, or Oracle Forms clients)
4. Learn how to **unit test** the code we write for the triggers

Please take a few minutes to read the mini-case given below. This is the case we have been working with in our previous labs, and is shown here to refresh our memory.

Mini-Case Description (repeated)

An automobile garage has mechanics, where each mechanic has a **number, name, salary** and **specialty**. The mechanics work on cars. Each car has a **number, model** and **year**. The garage wants to keep track of which mechanic worked on which car on what date.

If we were to draw an ER diagram, it would look like this:



An ER Diagram for the problem described above.

We have created a relational database in Oracle, for the problem stated above. The creation of the database was done using the *lab5_create.txt* script in the earlier labs. We also inserted some test data into the database, ran some sample SQL queries and wrote some PL/SQL procedures, functions and a package. All of this was done in the earlier labs.

Today, we'll try to **enhance** this application, by looking at some additional static and dynamic business rules. Now let's see the business rules.

Based on your interviews with the customers, you have additionally identified the following business rules that need to be obeyed:

1. The salary of all mechanics has to fall within a range of $5000. At no point can this be violated. Thus, the difference between the highest and lowest salaries has to be <=$5000.

2. Because of mechanic union rules, at no point can a mechanic be allowed to work on more than 10 repair jobs in a day.

You also identify the following business activity that is being performed manually, but that can be automated:

1. If a mechanic works on more than 500 jobs in a calendar year, then that mechanic gets rewarded as follows: His / her salary is immediately raised to the top of the range allowed (*i.e.,* they make the maximum permissible salary). Note that the maximum possible salary is the minimum salary + $5000 (based on an earlier business rule). Subsequently, whenever they do another job in that year, their salary is checked, and it is made sure they are making the maximum allowed.

Thus, we have 2 static business rules that preserve the integrity of our business data, and one business activity that is potentially automatable. Let us use triggers to support these.

For this lab, please create a **folder** of your own on the PC. Call it by a convenient name like *serv_triglab1*. **All the files we create today should be saved in this folder**.

In order to get started, please do the following:

1. Log in to your SQL Developer SQL session.

2. Start a text file with the name of Lab5

3. Drop the schema by typing `start <pathname>\lab5_drop.txt` in SQL Developer.

4. Recreate the schema by typing `start <pathname>\lab5_create.txt`. **Do not insert** any data into the tables.

Creating File *serv_trig1.txt*:

Using an editor on your PC type in the following file. Remember to keep saving the file frequently in your folder. Call the file *serv_trig1*. The file extension should be *.txt*.

```
REM This file is part of the server triggers lab. It is a before
REM insert or update trigger on the mechanics table.
REM IT ensures that whenever you add a mechanic to the database, or
REM update the salary of an existing mechanic, that the salary is
REM in the acceptable range. Otherwise it raises an unhandled
REM exception that causes the transaction to be rolled back.


CREATE OR REPLACE TRIGGER check_mech_sal
BEFORE INSERT OR UPDATE ON mechanics
FOR EACH ROW

DECLARE
   min_sal mechanics.mech_salary%type;
   max_sal mechanics.mech_salary%type;
   count_rows NUMBER;
   unfair_sal EXCEPTION;
BEGIN
   SELECT count(*) INTO count_rows FROM mechanics;
   IF count_rows<1 THEN
     RETURN;
   END IF;
   SELECT min(mech_salary) INTO min_sal FROM mechanics;
   SELECT max(mech_salary) INTO max_sal FROM mechanics;
   IF :new.mech_salary>=min_sal AND :new.mech_salary<=max_sal THEN
      RETURN;
   END IF;
   IF :new.mech_salary<min_sal THEN min_sal := :new.mech_salary;
   END IF;
```

```
   IF :new.mech_salary>max_sal THEN max_sal := :new.mech_salary;
   END IF;
   IF (max_sal - min_sal) >5000 THEN RAISE unfair_sal;
   END IF;
EXCEPTION
   WHEN unfair_sal THEN
      RAISE_APPLICATION_ERROR (-20326,'Error in Mechanics Salary');
END;
/
```

This trigger basically implements the business rule that: the salary of all mechanics has to fall within a range of $5000 (described earlier). Execute the file in SQL Developer,

```
start <pathname>\serv_trig1.txt .
```

The file should run, and the trigger should be created.

Creating File *serv_trig2.txt*:

Using an editor type in the following file, saving it as *serv_trig2.txt* in your folder:

```
REM This file is part of the server triggers lab
REM It contains a trigger before insert and update on
REM the repairs table.
REM  It raises an unhandled exception if a mechanic gets >10
REM repair jobs in a calendar day.

CREATE OR REPLACE TRIGGER check_daily_num_jobs_per_mech
BEFORE INSERT OR UPDATE ON repairs
FOR EACH ROW

DECLARE
   count_jobs NUMBER;
   too_many_jobs EXCEPTION;
BEGIN
   SELECT count(*) INTO count_jobs FROM repairs
   WHERE :new.mech_num=repairs.mech_num
   AND repairs.repair_date=:new.repair_date;
   IF count_jobs>=10 THEN
     DBMS_OUTPUT.PUT_LINE('Too many jobs for today!');
     RAISE too_many_jobs;
   END IF;
END;
/
```

Now we are ready to run this file. This trigger implements the business rule that: at no point can a mechanic be allowed to work on more than 10 repair jobs in a day. This rule was also described earlier. Click back to our SQL window (in case you exited SQL Developer, simply log in again for another SQL session). Type in

```
start <pathname>\serv_trig2.txt .
```

This should create the second trigger.

Creating File *serv_trig3.txt*:

Now we are ready to automate the business activity: Using an editor, type in the following file, saving it as *serv_trig3.txt*.

```
REM This file is part of the server triggers lab.
REM The trigger here automates the business activity of
REM rewarding mechanics who have done over
REM 500 repair jobs in one calendar year.
REM Note how the first of the year is calculated.

CREATE OR REPLACE TRIGGER reward_best_mechanic
BEFORE INSERT OR UPDATE ON repairs
FOR EACH ROW

DECLARE
   count_jobs_done_this_year NUMBER;
   first_of_year DATE;
   lowest_sal mechanics.mech_salary%TYPE;
BEGIN
   SELECT trunc(SYSDATE,'year') INTO first_of_year FROM dual;
   SELECT count(*) INTO count_jobs_done_this_year FROM repairs
   WHERE :new.mech_num=repairs.mech_num
   AND repairs.repair_date>first_of_year;
   IF count_jobs_done_this_year>500 THEN
     SELECT min(mechanics.mech_salary) INTO lowest_sal FROM mechanics
     WHERE mechanics.mech_num != :new.mech_num;
     UPDATE mechanics SET mech_salary=lowest_sal+5000 WHERE
       mechanics.mech_num=:new.mech_num;
   END IF;
END;
/
```

Now execute the file, just like we ran the 2 files described earlier. The third trigger is created. Type in

```
start <pathname>\serv_trig3.txt .
```

Next, let us try to test these triggers by actually inserting some test data in the tables. Note that so far**, there is no data** in our tables.

Testing *serv_trig1.txt*:

Using your notepad editor, create file *test_serv_trig1.txt* as follows.

```
REM This file is part of the server triggers lab.
REM This file is used to insert data into the mechanics table
REM It tests trigger check_mech_sal in file serv_trig1.txt


INSERT INTO mechanics VALUES(mech_sequence.NEXTVAL, ' Thomas Smith',
                             55000.00, 'Dodge Specialist');
INSERT INTO mechanics VALUES(mech_sequence.NEXTVAL, 'Sydney Li',
                             50000.00, 'BMW Specialist');
INSERT INTO mechanics VALUES(mech_sequence.NEXTVAL, 'Randy Jones',
                             45000.00, 'Toyota Specialist');
```

Before executing this file, make sure we type in:

`SET SERVEROUTPUT ON` . This command activates the usage of the DBMS_output package, which is used in this block to write to the screen.

Now execute the file. `start <pathname>\test_serv_trig1.txt`

You should see an error displayed, when you insert the third mechanic. If you see the data in the mechanics table, you'll see only 2 rows (use a SELECT statement to see the mechanics table).

Describe why this error occurred. Include your response in your text file submission.

Testing *serv_trig2.txt*:

Next drop the schema, then recreate it, and recreate the trigger in *serv_trig2.txt* also. Please refer back to previous pages in this lab in case you are not sure how to do this.

Next, create file *test_serv_trig2.txt* as follows:

```
REM This file is part of the server triggers lab.
REM This file is used to insert data into the repairs table
REM It tests trigger check_daily_num_jobs_per_mech
REM in file serv_trig2.txt

CREATE OR REPLACE PROCEDURE temp_test1
AS
i   NUMBER;
first_car_value NUMBER;
```

```
first_mech_value NUMBER;

BEGIN
 INSERT INTO mechanics VALUES(mech_sequence.NEXTVAL, 'Thomas Smith',
                              55000.00, 'Dodge Specialist');
 INSERT INTO mechanics VALUES(mech_sequence.NEXTVAL, 'Sydney Li',
                              50000.00, 'BMW Specialist');
 COMMIT;
 FOR i IN 1..11 LOOP
  INSERT INTO cars VALUES(car_sequence.NEXTVAL,'Dummy car',
'01-JAN-2018');
 END LOOP;
 COMMIT;
 SELECT min(car_num) INTO first_car_value FROM cars;
 SELECT min(mech_num) INTO first_mech_value FROM mechanics;
 FOR i IN first_car_value..first_car_value+10 LOOP
  INSERT INTO repairs values(first_mech_value,i,SYSDATE);
  COMMIT;
 END LOOP;
END;
/
```

Now compile this test procedure by typing : `start <pathname>test_serv_trig2.txt .`

Run the procedure by typing: `execute temp_test1;`

Testing *serv_trig3.txt*:

First drop the schema, then recreate it, and recreate ONLY the trigger in *serv_trig3.txt* (Do not recreate serv_trig1 or serv_trig2). Please refer back to previous pages in this lab in case you are not sure how to do this. Create the following file and save it as *test_serv_trig3.txt*.

```
REM This file is part of the server triggers lab.
REM This file is used to insert data into the repairs table
REM It tests trigger reward_best_mechanic
REM in file serv_trig3.txt. We insert 2 mechanics with
REM equal salaries, and increase the number of repair
REM jobs of Thomas Smith to >10. This should cause his salary
REM to go up to the maximum allowed, which is 60,000
REM in this case.

CREATE OR REPLACE PROCEDURE temp_test2
AS
i   NUMBER;
j   NUMBER;
first_car_value NUMBER;
first_mech_value NUMBER;
temp_date DATE;

BEGIN
```

```
 INSERT INTO mechanics VALUES(mech_sequence.NEXTVAL, 'Thomas Smith',
                              55000.00, 'Dodge Specialist');
 INSERT INTO mechanics VALUES(mech_sequence.NEXTVAL, 'Sydney Li',
                              55000.00, 'BMW Specialist');
 COMMIT;
 FOR i IN 1..10 LOOP
  INSERT INTO cars VALUES(car_sequence.NEXTVAL,'Dummy car',
'01-JAN-2018');
 END LOOP;
 COMMIT;
 SELECT min(car_num) INTO first_car_value FROM cars;
 SELECT min(mech_num) INTO first_mech_value FROM mechanics;
 SELECT trunc(SYSDATE,'year') INTO temp_date FROM dual;
 FOR j IN 1..52 LOOP
   FOR i IN first_car_value..first_car_value+9 LOOP
     INSERT INTO repairs values(first_mech_value,i,temp_date);
     COMMIT;
   END LOOP;
   temp_date:=temp_date+1;
 END LOOP;
END;
/
```

Now run this test procedure *temp_test2*. At the end of this test procedure, the salary of *Thomas Smith* should be $60000. *Show that the salary was changed.*

Close your text file.

As part of submission for this lab, please submit the text file Lab5.

For a list of capabilities you acquired in this lab, please refer back to page 1.