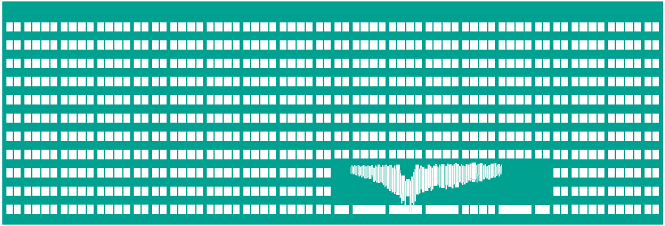
VŠB TECHNICKÁ
UNIVERZITA
OSTRAVA

VSB TECHNICAL
UNIVERSITY
OF OSTRAVA



www.vsb.cz

Database and Information Systems

db.cs@vsb.cz

Department of Computer Science
Faculty of Electrical Engineering and Computer Science
VSB - Technical University of Ostrava

2019/2020

Task 1: Triggers

1. Create trigger `OperationCount` which will record into the table `Statistics` a number of insert, update a delete operations. The table `Statistics` will have two attributes. The first attribute `operation` will be a type of operation and the second attribute `operationCount` will be a number of current operations (Use DML operation detection in trigger).
2. Add attribute `capacity` into the table `Course`, which represents maximum capacity of current course. Create trigger `controlCapacity`, which will generate exception in the case of the capacity overload.
3. In trigger `controlCapacity` declare exception `capacity_exceeded`, which will be called in the case of the capacity overload.

Task 2: Table backup

1. Create stored procedure `CopyTableStructure` with one input parameter `p_table_name`, which will create a copy (only attributes) of table with the name `p_table_name`. New table will be empty and have suffix '_old' and it will have the same attributes with the same names (and types) as original table.

   Tip: Names and types of the attributes can be selected from system catalog. Build the complete command `CREATE TABLE` (which will create a new table) into a string variable. Use command `Execute Immediate` to run command in the string variable.

2. Create stored procedure `CopyTable` with one input parameter `p_table_name`, which will create a copy of the table and copy the data from the original table into the new table.

Task 3A: Bind variables

1. Create PL/SQL anonymous procedure, which will print values of OBJECT_NAME from the table ALL_OBJECTS for each object with OBJECT_ID from 0 to 10 000 using cursor and loop[1].
2. Create same procedure with and without usage of bind variables.
3. Measure the operation time with DBMS_UTILITY.GET_TIME.

[1] Use cursor

Task 3B: Bind variables

Note: unless noted otherwise all tasks are for table `Usertab`.

```
CREATE TABLE Usertab(
  ID NUMBER PRIMARY KEY,
  fname VARCHAR(50) NOT NULL,
  lname VARCHAR(50) NOT NULL)
```

1. Create stored procedure `InsertInUsertab`, which will insert into table `Usertab` in loop 100 000 records, the first one will be (1, 'fname1', 'lname1'), second one (2, 'fname2','lname2') etc.
2. Create same procedure with and without usage of bind variables.
3. Measure the operation time with DBMS_UTILITY.GET_TIME.

Task 3C: COMMIT

1. For the procedure `InsertInUsertab` from the previous task measure the time[2] for two cases: with commit after each inserted record and with commit only after inserting all records.

[2] using DBMS_UTILITY.GET_TIME

Task 4: Exceptions 1/2

Note: unless noted otherwise all tasks are for table `Student`.

1. Change function `LoginExist` to return `false` only if student with current login does not exist in table. In the case of any other exception show it to user.
2. Create function `InsertStudent`, which will create student record and return `true`, if record was successfully created. In the case such login already exists return `false`. Use exception `DUP_VAL_ON_INDEX`. Use this function in procedure `AddStudent3`, which will have the same functionality as `AddStudent2`.

Task 4: Exceptions 2/2

1. Create stored procedure `StudentBecomeTeacher` to be able remove records from table `Student_Course`, which are linked to removing student. (Declare exception associated with number and catch it[3]).

[3] `https://docs.oracle.com/en/database/oracle/oracle-database/18/lnpls/EXCEPTION_INIT-pragma.html`