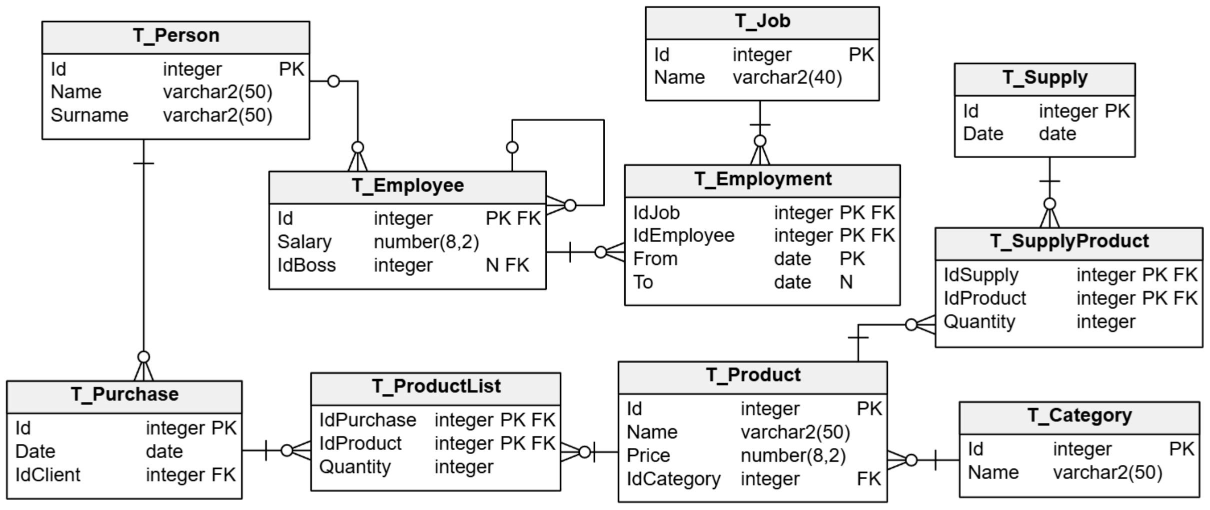
SBD Lab08
Basics of PL/SQL

Link to generate the database: link.
Link to drop the database: link.



Task 1
Write a simple program in PL/SQL. Declare a variable, assign the number of records from the T_Person table to this variable and print the result using DBMS_OUTPUT. The output should look like this: *"There are 99 records in the T_Person table"*.

Task 2
Count the number of records in the T_Person table. If there are less than 11 records then insert a new person and print a message *“Added Name Surname with id= idNumber”*. If there are already 11 records or more then simply print a message stating that no data was inserted. Let the new person's Id take the value of the largest existing id + 1 from T_Person table.

Task 3
In the anonymous block, handle errors (in EXCEPTION) such as no_data_found, too_many_rows, dup_val_on_index including all other errors. Depending on the type of error list "Record not found", "More than one record found", "PK constraints violated" and "Other error" respectively. Declare a variable v_name of type Varchar2 in order to test errors on it.

Then try the following operations one at a time:
-Assign the product name with id= 15 to the v_name variable (no_data_found error)
-Assign product names with id IN (1,2,3) to the v_name variable (too_many_rows error)
-Add a new product with id=1 (dup_val_on_index error)
-Assign 10/0 to the v_name variable (other error)

Task 4

Create a procedure called "UpdatePrice" that will increase the price of all products by the value given in the parameter. If the value is not provided then it should be increased by 0.01 (using the DEFAULT parameter). Additionally we want to print a message with information by how much price was increased and how many records were modified (using the SQL%ROWCOUNT system variable).

Task 5

Write the "EmployeeData" procedure that will accept the employee's ID and return his First Name and Last Name using the OUT parameter. If the employee with the given ID does not exist, we should return information: "The employee with the given ID does not exist."

Task 6

Write a procedure called "NewPurchase" that will create a new purchase for a given customer with today's date. The procedure should take the customer's ID as a parameter and return the ID of the created purchase in the OUT parameter. The primary key "id" in the T_Purchase table has the IDENTITY property, that means the id is generated automatically thus we do not provide this value. Inside the procedure we want to print the information "New purchase has been registered with id: [id]". We also want to be able to capture the Id of a newly created purchase and print it outside of the procedure (using OUT parameter).
TIP: for a date of the new purchase you can use: *to_date(current_date, 'YYYY-MM-DD')*

Task 7

Create a procedure called "AddProductToPurchase" which will add the product to a given purchase. As parameters the procedure should take IdProduct, Quantity and IdPurchase, which we receive from the "NewPurchase" procedure (from Task 6) via the OUT parameter. You should check whether the product and purchase with the given ID exist and whether the quantity is greater than 0, if not, print an appropriate message and cancel the procedure (you can use RETURN statement). If a given product is already assigned to a given purchase, we simply increase its quantity in UPDATE and print "Product [id] increased in purchase [id] by: [quantity]". Otherwise, we add the product to the purchase and print: "Product [id] added to purchase [id], quantity: [quantity]"

Using these procedures, create a new purchase for a client with id=1, capture purchase id and add to it a product with id=1 in quantity of 1. Do it twice for the same purchase, so that you get 1 insert and 1 update.

Task 8

Write a procedure that will update the job of a given employee. The procedure is to assign the employee to a new job with today's date (you can use: *to_char(sysdate, 'YYYY-MM-DD')*) and at the same time remove him from the old job also with today's date ("To" column in the T_Employment table). The procedure should accept v_EmployeeId and v_JobId as arguments. If the employee is currently assigned to the given job, we do not add him again but instead print the message "The employee is already assigned to this job". If the employee or job with the given id does not exist, we print the appropriate message “Employee/Job does not exist” and cancel the procedure (you can use the RETURN statement). The job for a given employee can be changed only once a day, if today's date already exists in the "To" column in the T_Employment table for a given employee, then we do not update his job and print the information: "Changes have not been saved, the job can only be updated once a day for a given employee."