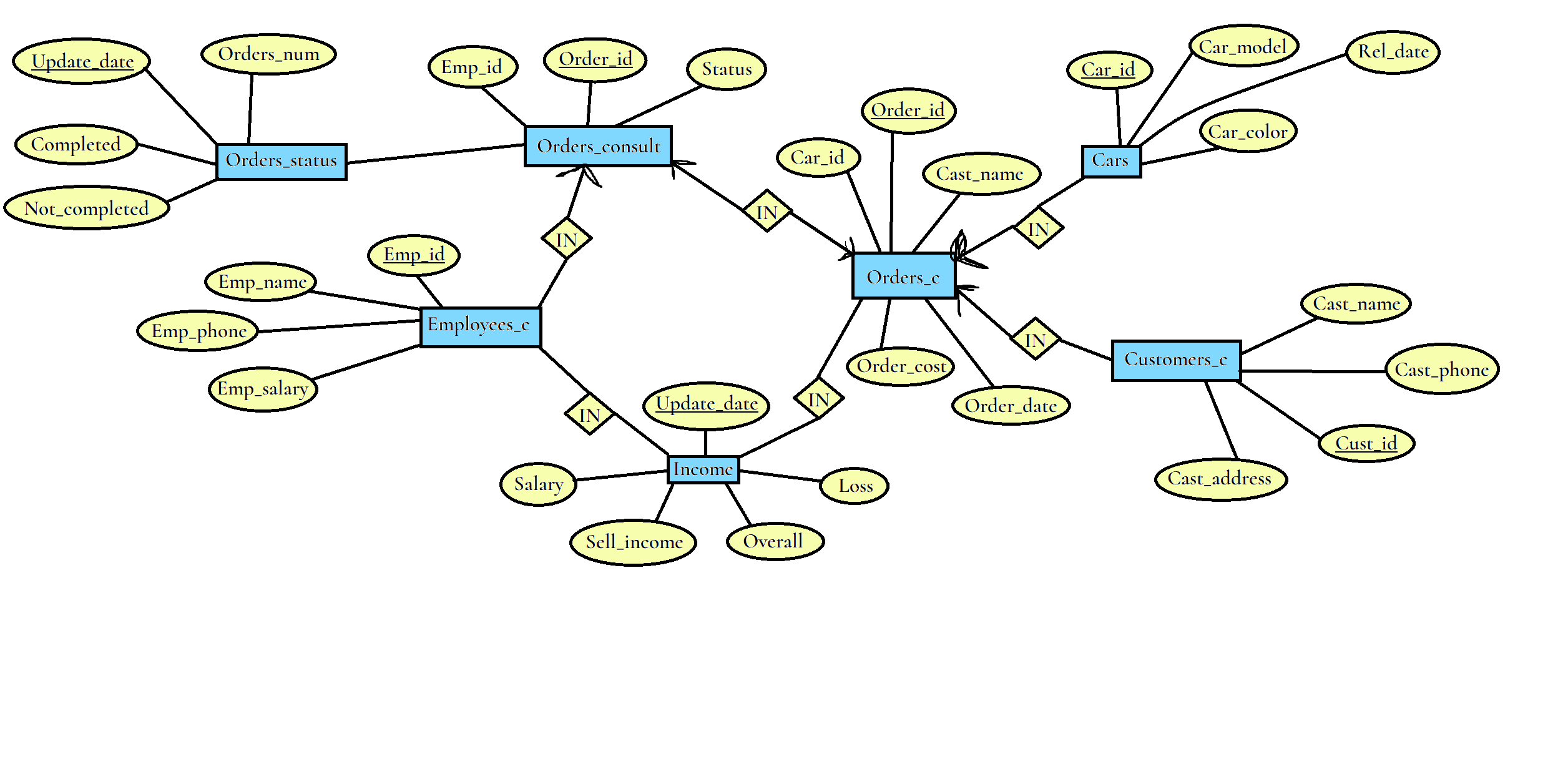
**Report**

My database is created for a Car Shop. It is aimed at the staff and management of the store. It has 7 tables. They are: **Cars**, **Customers\_c, Employees\_c, Orders\_c, Orders\_consult, Orders\_status** and **Income tables.**



**First table Cars:**

There are four columns in Cars table: Car\_id, Car\_model, Car\_color, Rel\_date. This table about cars. Car\_id – unique identifier number of cars. All cars in this table is unique and there are no storages for cars, so there is no quantity of cars. Because after order cars will be imported from other countries. Car\_id – primary key of this table.

Car\_model – is model of car. Car\_color – is color of car, Rel\_date – is release date of this model of car. It is very simple.

**Normalization:**

1-norm: All fields in this table contain maximum one data or element;

2-norm: There is only one key column. So partial dependency is impossible.

3-norm: All non-key columns of this table depend on Car\_id key column.

**So, it is time to explain my triggers in this table. There is only one trigger. Same triggers will be used in other tables.**

AUTOID trigger just put the new inserting row with the right Car\_id. First, declares number type Max\_n value. And when insert row doesn’t have Car\_id column (What is right, because it is too long to check last Car\_id manually.) it is automatically look for last Car\_id, add 1 to this number and give it to Max\_n. Then just give max\_n value to new inserted row’s Car\_id field.

**Table Customers\_c:**

There are also four columns in this table. Cust\_id (Cust is from customer), Cast\_name (I miss clicked, so it’s should be cust\_name…), Cast\_phone and Cast\_address. This table contain info about clients of Car Shop. The address of customers will not be used in midterm. It is oriented to future.

Cust\_id - primary key of this table.

**Normalization:**

1-norm: All fields in this table contain maximum one data or element;

2-norm: There is only one key column. So partial dependency is impossible.

3-norm: All non-key columns of this table depend on Cust\_id key column.

**TIME OF TRIGGERS!**

**There is only one trigger and it is as same as in the Cars table’s AUTOID trigger. So pass it.**

**Table Employees\_c:**

There are four columns in this table. Emp\_id (Emp is from employee), Emp\_name, Cast\_phone and Emp\_salary. This table contain info about staff of Car Shop. I think that name of this columns help to understand aim of columns very easy.

Emp\_id - primary key of this table.

**Normalization:**

1-norm: All fields in this table contain maximum one data or element;

2-norm: There is only one key column. So partial dependency is impossible.

3-norm: All non-key columns of this table depend on Emp\_id key column.

**TIME OF TRIGGERS!**

**Oh, there are two triggers now! First trigger as same as AUTOID in the Cars table. So, pass and go to second one.**

**It is – Salary\_Report trigger! We declare three more values for our comfort. First – “sal” number type value. Sal takes the SUM of all employees’s salary. Second – “inc” number type value. It takes the Income of all orders, that have been done, and just SUM it. Simple, right? Third value is “ove”. It just take 40% of all Income value add minus from it “sal” and some value of wastes.**

**Table Orders\_c:**

In this table I have 5 columns: Order\_id, Cust\_id, Car\_id, Order\_cost, Order\_date. This is main table of my DB and its aim is to collect info about all customer’s orders. Order\_id – is primary key. Cust\_id is Foreign key referencing itself from Customers\_c table. Car\_id is Foreign key referencing itself from Cars table. Order\_cost is cash, that Customer paid or should pay for the car. Yes, there is no fixed price of cars. Order\_date is just date of inserting the order into table.

This table more complicated than other.

**Normalization:**

1-norm: All fields in this table contain maximum one data or element;

2-norm: There is only one key column. So partial dependency is impossible.

3-norm: All non-key columns of this table depend on Order\_id key column.

**TIME OF TRIGGERS!**

**There are two triggers also. One of the triggers is like AUTOID from the first table of my report. Second trigger is AUTO\_CONSULTING. When some customer ordering, there should be employee that can help to Customer with his different questions.**

**Firstly, we declare 6 (!) additional values. All of them is number type. There are: n\_id, no\_id, emdn\_id, sal, inc and ove. We know about last three numbers and their aim from Employees\_c table’s trigger. N\_id value that contain the maximum number of Order\_id from Orders\_c table. No\_id contain last number of Order\_id from Orders\_consult table. Emp\_id contain random Emp\_id from Employees\_c table. Then we just insert into Orders\_consult these three values if n\_id doesn’t equal no\_id. That’s mean, that if there in Orders\_c just some Update of order, it does the report for Income table but doesn’t make new insert into Orders\_consult.**

**Table Orders\_consult:**

It is very simple table with tree columns: Order\_id, Emp\_id and status. When some order inserted in the table Orders\_c, there also will be inserted new row with Order\_id of report and random Emp\_id. Order\_id – primary and Foreign key of the table. Emp\_id – Foreign key that reference itself from Employees\_c table. Status – is status of order. If it is 1 – order have been completed, if it is 0 – it is not completed.

**Normalization:**

1-norm: All fields in this table contain maximum one data or element;

2-norm: There is only one key column. So partial dependency is impossible.

3-norm: All non-key columns of this table depend on Order\_id key column.

**TIME OF TRIGGERS!**

There is only one trigger Orders\_report. It does the report for Orders\_status table. Firstly, as usual, we declare three values. All is number: all\_o, comp\_o, n\_comp\_o. All\_o contains counted number of all orders from Orders\_consult. Comp\_o contains counted number of all orders with status 1. And n\_comp\_o contains counted number of all orders with status 0. Then it takes all these values and just insert into Orders\_status with actual time.

**Table Orders\_status:**

Simple table with four columns: Update\_date, Orders\_num, Completed, Not\_completed. Update date id primary key and it is time of updating the status of all orders. Orders\_num – number of orders in Orders\_c. Completed – number of completed orders of it and not\_completed – number of not completed orders. Very easy.

**Normalization:**

1-norm: All fields in this table contain maximum one data or element;

2-norm: There is only one key column. So partial dependency is impossible.

3-norm: All non-key columns of this table depend on Update\_date key column.

**TIME OF TRIGGERS!**

It has no triggers.

**Table Income:**

My last table have 5 columns: Update\_date, Salary, Sell\_income, Loss and Overall. Update\_date is Primary key and like in previous table it shows the time of income status updating. Salary displays loss of money for salary of employees. Sell\_income – 40% of Order\_cost price, Car Shop resell the cars so it takes only 40% of all price. Loss – money for service computers, servers, office and so on. Overall – clear income of Car Shop!

**Normalization:**

1-norm: All fields in this table contain maximum one data or element;

2-norm: There is only one key column. So partial dependency is impossible.

3-norm: All non-key columns of this table depend on Update\_date key column.

**TIME OF TRIGGERS!**

It has no triggers.

**SO, that’s all about my tables and triggers! Let’s do some functions, procedures and other operations!**

1. **GROUP BY PROCEDURE**

**CREATE OR REPLACE PROCEDURE Order\_cars AS**

**BEGIN**

**DECLARE**

**CURSOR order\_cu IS**

**SELECT Orders\_c.car\_id, Cars.car\_model, COUNT(Orders\_c.car\_id) as Numb\_of**

**FROM Cars, Orders\_c**

**WHERE Orders\_c.car\_id = Cars.car\_id**

**GROUP BY Orders\_c.car\_id, Cars.car\_model**

**ORDER BY Orders\_c.car\_id;**

**order\_rec order\_cu%ROWTYPE;**

**BEGIN**

**OPEN order\_cu;**

**LOOP**

**FETCH order\_cu INTO order\_rec;**

**EXIT WHEN order\_cu%NOTFOUND;**

**DBMS\_OUTPUT.PUT\_LINE('Car id: ' || order\_rec.car\_id || ' Car model: ' || order\_rec.car\_model || ' Number of orders: ' || order\_rec.Numb\_of);**

**END LOOP;**

**CLOSE order\_cu;**

**END;**

**END Order\_cars;**

**This procedure outputs the cars id, model and number of how many times it was ordered. First, we declare the cursor Order\_cu. This cursor take car\_id from order\_c, it’s model from Cars and Count all Car\_id from orders\_c. And then order it by car\_id and group it by it also. More I will explain at midterm.**

**Then we declare order\_rec that have type of rows of order\_cu. In begin we open order\_cu and loop. In this loop we put data from order\_cu to order\_rec and Output it with easy to understand info. It will stop when there is no more data in order\_cu that we can put in order\_rec.**

**MORE IN MID.**

1. **COUNT FUNCTION**

**CREATE OR REPLACE FUNCTION total\_orders**

**RETURN number IS**

**total number := 0;**

**BEGIN**

**SELECT count(order\_id) into total**

**FROM Orders\_c;**

**RETURN total;**

**END;**

Very simple function that just count number of orders and return it. There is code that use this function.

DECLARE

c number;

BEGIN

c := total\_orders();

dbms\_output.put\_line('Total no. of Orders: ' || c);

END;

1. **%rowcount procedure**

**CREATE OR REPLACE PROCEDURE orders\_done AS**

**BEGIN**

**DECLARE**

**CURSOR order\_done IS**

**SELECT orders\_consult.order\_id, Cars.car\_model**

**FROM Cars, orders\_c, orders\_consult**

**WHERE orders\_c.car\_id = Cars.car\_id**

**AND orders\_c.order\_id = orders\_consult.order\_id**

**AND orders\_consult.status = 1**

**order by orders\_consult.order\_id;**

**order\_done\_rec order\_done%ROWTYPE;**

**BEGIN**

**OPEN order\_done;**

**LOOP**

**FETCH order\_done INTO order\_done\_rec;**

**EXIT WHEN order\_done%NOTFOUND;**

**DBMS\_OUTPUT.PUT\_LINE('Car id: ' || order\_done\_rec.order\_id || ' Car model: ' || order\_done\_rec.car\_model);**

**END LOOP;**

**DBMS\_OUTPUT.PUT\_LINE('Orders that done: ' || order\_done%ROWCOUNT);**

**CLOSE order\_done;**

**END;**

**END orders\_done;**

***This procedure output all orders that have been done with the model of car that have been ordered. First, we declare cursor that will be used. In this cursor we take two values from different tables. Sort them by status “1” and order them by order\_id.***

***In begin the order\_done cursor is opened, and a loop is started to fetch rows from the cursor into the order\_done\_rec record variable. The DBMS\_OUTPUT.PUT\_LINE statement is used to display the fetched data, which includes the order\_id and car\_model values.***

***The order\_done%ROWCOUNT attribute is used to retrieve the total number of rows fetched by the cursor. This value is displayed using the DBMS\_OUTPUT.PUT\_LINE statement.***

1. **Trigger with user-defined exeption**

**create or replace TRIGGER cars\_insert\_trigger**

**BEFORE INSERT ON Cars**

**FOR EACH ROW**

**DECLARE**

**car\_model\_length EXCEPTION;**

**BEGIN**

**IF LENGTH(:NEW.Car\_model) < 5 THEN**

**RAISE car\_model\_length;**

**END IF;**

**EXCEPTION**

**WHEN car\_model\_length THEN**

**RAISE\_APPLICATION\_ERROR(-20001, 'Car model must be at least 5 characters long');**

**END;**

***Very easy trigger with user-defined exception* car\_model\_length declared. When the length of new value Car\_model is less than 5, it raises this exception;**

1. **Trigger with user-defined exeption**

**create or replace TRIGGER AutoID**

**before insert ON Cars**

**for each row**

**declare**

**max\_n number;**

**begin**

**if :NEW.car\_id is null then**

**select max(car\_id) into max\_n from Cars;**

**dbms\_output.put\_line('The current number of rows: ' || max\_n);**

**max\_n := max\_n + 1;**

**:NEW.car\_id := max\_n;**

**end if;**

**end;**

***To solve this task with trigger I will use already existing trigger AUTOID. I explained it already at the head of these report. There is just one more additional row in AUTOID code.***

**dbms\_output.put\_line('The current number of rows: ' || max\_n);**

***Already existing MAX\_N value is check for last ID of these table. Then I will just output it with some text. Very easy!***

***IT IS END OF MY REPORT!***