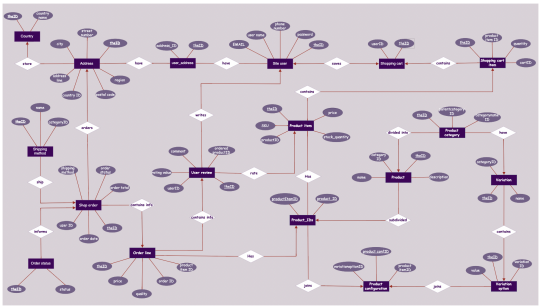
DBMS-2 project REPORT

- **INTRODUCTION**

The primary purpose of our project is to create an online store where all actions are performed on the website. This is undoubtedly convenient for promoting small businesses in any field. Our task was to develop the convenience of saving data.

- **ER diagram**



- Our structure corresponds to all normal forms. **The first normal form** is observed, as a rule, each cell contains one value, and there are no duplicate groups or arrays.

theID	category_ID	name	products_desc
1	1	Crocheted froggy-hat	Crocheted green froggy-hat Crocheted brown froggy-hat

theID	category_ID	name	products_desc
1	1	Crocheted froggy-hat	Crocheted green froggy-hat
1	1	Crocheted froggy-hat	Crocheted brown froggy-hat

```
country Table:
create table country(
    theID number PRIMARY KEY,
    countryName varchar(100)
);
```

Country

2NF

Primary key: theID

Non-key attributes: countryName

The non-key attribute depends on the primary key **theID**.

3NF

The table has no transitive dependencies.

```
Address table:
CREATE TABLE Address(
    theID NUMBER NOT NULL,
    streetNumber NUMBER,
    addressLine1 varchar(100),
    city varchar(100),
    region varchar(100),
    postalCode NUMBER check (postalCode between 0 and 9),
    countryID NUMBER,
    PRIMARY KEY (theID),
    FOREIGN KEY (countryID) references Country(theID)
);
```

Address

2NF

Primary key: theID

Non-key attributes: streetNumber, addressLine1, city, region, postalCode, countryID

The non-key attributes depend on the primary key **theID**.

3NF

The table has no transitive dependencies. The foreign key constraint on **countryID** ensures that there is no possibility of a foreign key dependency cycle.

```
user_address table:
create table user_address(
    theID number;
    addressID number;
    PRIMARY KEY (theID),
    FOREIGN KEY (addressID) references site_user(theID)
);
```

- ***user_address***
  2NF
  Primary key: theID
  Non-key attributes: addressID
  The non-key attributes depend on the primary key **theID**.
  3NF
  The table has no transitive dependencies. The foreign key constraint on **addressID** ensures no possibility of a foreign key dependency cycle.

```
site_user table:
CREATE TABLE site_user(
    theID number,
    email varchar(100),
    username varchar(100),
    phoneNumber number,
    pass varchar(100),
    PRIMARY KEY (theID)
);
```

- ***site_user***
  2NF
  Primary key: theID
  Non-key attributes: email, username, phoneNumber, pass
  The non-key attributes depend on the primary key **theID**.
  3NF
  The table has no transitive dependencies.

- ***shopping_cart***
  2NF
  Primary key: theID
  Non-key attributes: userID
  The non-key attributes depend on the primary key **theID**.
  3NF

The table has no transitive dependencies. The foreign key constraint on **userID** ensures no possibility of a foreign key dependency cycle.

```
cart_item table:
CREATE TABLE cart_item(
    theID number,
    cartID number,
    productItemID number,
    quantity number,
    PRIMARY KEY (theID),
    FOREIGN KEY (cartID) references shopping_cart(theID),
    FOREIGN KEY (productItemID) references productItem(theID),
);
```

cart_item

2NF

Primary key: theID

Non-key attributes: cartID, productItemID, quantity

The non-key attributes depend on the primary key **theID**.

3NF

The table has no transitive dependencies.

```
product_item table:
CREATE TABLE product_item(
    theID number,
    SKU number,
    stock_quantity number,
    price number,
    PRIMARY KEY (theID)
);
```

product_item

2NF

Primary key: theID

Non-key attributes: SKU, stock_quantity, price

The non-key attributes depend on the primary key **theID**.

3NF

The table has no transitive dependencies.

```
product_ids table:
CREATE TABLE product_ids(
    product_item_ID number,
    product_ID number,
    FOREIGN KEY (product_item_ID) REFERENCES product_item(theID),
    FOREIGN KEY (product_ID) REFERENCES product(theID),
    PRIMARY KEY (product_item_ID, product_ID)
);
```

product_IDs

2NF

Primary key: product_item_ID, product_ID

3NF

The table has no transitive dependencies.

```
product table:
CREATE TABLE product(
    theID number,
    category_id number,
    name varchar(100),
    product_desc varchar(255),
    PRIMARY KEY (theID),
    FOREIGN KEY (category_id) REFERENCES product_category(theID)
);
```

product

2NF

Primary key: theID

Non-key attributes: category_id, name, product_desc

The non-key attributes depend on the primary key **theID**.

3NF

The table has no transitive dependencies. The foreign key constraint on **categoryID** ensures that there is no possibility of a foreign key dependency cycle.

```
variation table:
CREATE TABLE variation(
    theID number,
    categoryID number,
    theName varchar(255),
    PRIMARY KEY (theID),
    FOREIGN KEY (categoryID) references
product_category(theID)
);
```

variation

2NF

Primary key: theID

Non-key attributes: category_id, theName

The non-key attributes depend on the primary key **theID**.

3NF

The table has no transitive dependencies. The foreign key constraint on **categoryID** ensures that there is no possibility of a foreign key dependency cycle.

```
variation_option table:
CREATE TABLE variation_option(
    theID number,
    variationID number,
    theValue number,
    PRIMARY KEY (theID),
    FOREIGN KEY (variationID) references variation(theID)
);
```

variation_option

2NF

Primary key: theID

Non-key attributes: variationID, theValue

The non-key attributes depend on the primary key **theID**.

3NF

The table has no transitive dependencies. The foreign key constraint on **variationID** ensures that there is no possibility of a foreign key dependency cycle.

```
product_configuration table:
CREATE TABLE product_configuration(
    productconfID number,
    productItemID number,
    variationOptionID number,
    PRIMARY KEY (productconfID),
    FOREIGN KEY (productItemID) references
product_item(theID),
    FOREIGN KEY (variationOptionID) references
variation_option(theID)
);
```

product_configuration

2NF

Primary key: productconfID
Non-key attributes: productItemID, variationOptionID
The non-key attributes depend on the primary key **productconfID**.
3NF
The table has no transitive dependencies.

```
product_category table:
CREATE TABLE product_category(
    theID number,
    parent_category_ID number,
    category_name varchar(255),
    PRIMARY KEY (theID)
);
```

product_category
2NF
Primary key: theID
Non-key attributes: parent_category_ID, category_name
The non-key attributes depend on the primary key **theID**.
3NF
The table has no transitive dependencies.

```
user_review table:
CREATE TABLE user_review(
    theID number,
    userID number,
    orderedProductID number,
    rating number,
    commentary varchar(255),
    PRIMARY KEY (theID),
    FOREIGN KEY (userID) references site_user(theID),
    FOREIGN KEY (orderedProductID) references shop_order(theID)
);
```

user_review
2NF
Primary key: theID
Non-key attributes: userID, orderedProductID, rating, commentary
The non-key attributes userID, orderedProductID, rating, and commentary all depend on the primary key **theID**
3NF
It does not have any transitive dependencies or repeating groups.

```
shop_order table:
CREATE TABLE shop_order(
    theID number,
    userID number,
    order_date date,
    shipping_method number,
    order_total number,
    order_status number,
    PRIMARY KEY (theID),
    FOREIGN KEY (userID) references site_user(theID),
    FOREIGN KEY (shipping_method) references shipping_method(theID),
    FOREIGN KEY (order_status) references order_status(theID)
);
```

shop_order

2NF

Primary key: theID

Non-key attributes: userID, order_date, shipping_method, order_total, order_status

The non-key attributes depend on the primary key **theID**.

3NF

The table has no transitive dependencies. The foreign keys ensure that there is no possibility of a foreign key dependency cycle.

```
shipping_method table:
CREATE TABLE shipping_method(
    theID number,
    categoryID number,
    theName varchar(100),
    PRIMARY KEY (theID),
    FOREIGN KEY (categoryID) references product_category(theID)
);
```

shipping_method

2NF

Primary key: theID

Non-key attributes: categoryID,theName

The non-key attributes depend on the primary key **theID**.

3NF

The table has no transitive dependencies. The foreign key constraint on **categoryID** ensures that there is no possibility of a foreign key dependency cycle.

```
order_line table:
CREATE TABLE order_line(
    theID number,
    productItemID number,
    orderID number,
    quantity number,
    price number,
    PRIMARY KEY (theID),
    FOREIGN KEY (productItemID) references productItem(theID),
);
```

order_line

2NF

Primary key: theID

Non-key attributes: productItemID, orderID, quantity, price

The non-key attributes depend on the primary key **theID**.

3NF

The table has no transitive dependencies. The foreign key constraint on **productItemID** ensures that there is no possibility of a foreign key dependency cycle.

```
order_status table:
CREATE TABLE order_status(
    theID number,
    status varchar(100),
    PRIMARY KEY (theID)
);
```

order_status

2NF

Primary key: theID

Non-key attributes: status

The non-key attributes depend on the primary key **theID**.

3NF

The table has no transitive dependencies.

- Procedure which does group by information:

```sql
CREATE OR REPLACE PROCEDURE with_group_by IS
  item_id order_line.productitemid%TYPE;
  quantity order_line.quantity%TYPE;
  totalprice order_line.price%TYPE;

  CURSOR retrieve IS
    SELECT productitemid, SUM(quantity) AS quantity, SUM(price) AS totalprice
    FROM order_line
    GROUP BY productitemid;

BEGIN
  OPEN retrieve;
  LOOP
    FETCH retrieve INTO item_id, quantity, totalprice;
    EXIT WHEN retrieve%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE('ID: ' || item_id || ', Count: ' || quantity || ' - Total: ' || totalprice);

    IF quantity > 15 THEN
        DBMS_OUTPUT.PUT_LINE('The quantity has exceeded 15! Please, order again.');
    END IF;
  END LOOP;

  CLOSE retrieve;
END;
```

Results | Explain | Describe | Saved SQL | History

```
Procedure created.
```

0.01 seconds

- Function which counts the number of records:

```sql
CREATE OR REPLACE FUNCTION count_records_function
RETURN NUMBER IS
  total NUMBER(5) := 0;
BEGIN
  SELECT COUNT(*) INTO total
  FROM product;

  RETURN total;
END;
/
```

Results | Explain | Describe | Saved SQL | History

```
Function created.
```

0.19 seconds

```
DECLARE
  count_num NUMBER(5);
BEGIN
  count_num := count_records_function();
  DBMS_OUTPUT.PUT_LINE('The number of records in table is ' || count_num);
END;
```

Results | Explain | Describe | Saved SQL | History

```
The number of records in table is 102

Statement processed.

0.00 seconds
```

We can change table name in function and replace it to give information about records in table

- Procedure which uses SQL%ROWCOUNT to determine the number of rows affected:

```
DECLARE
  numberOfSales NUMBER;
  CURSOR products_cursor IS SELECT * FROM product where category_id = 1;
BEGIN
  FOR prod IN products_cursor
  LOOP
    DBMS_OUTPUT.PUT_LINE(prod.name);
  END LOOP;

  UPDATE product_item
  SET price = price * 0.3 where product_id in (select theid from product where category_id = 1);

  numberOfSales := TO_NUMBER(SQL%ROWCOUNT);

  DBMS_OUTPUT.PUT_LINE('Now the price is reduced by 30% for ' ||  numberOfSales || ' products');
END;
```

Output:

```
Crocheted froggy-hat
Crocheted teddy-hat
Crocheted beannie hat
Crocheted cat beannie hat
Crocheted Finns hat
Crocheted pie-beret-hat
Now the price is reduced by 30% for 6 products

1 row(s) updated.


0.01 seconds
```

- Add user-defined exception which disallows to enter title of item (e.g. book) to be less than 5 characters:

```
1   create or replace trigger before_ins_user
2   before insert on site_user
3   for each row
4   declare
5   password_is_right exception;
6   begin
7   if length(:new.pass) < 5 then raise password_is_right;
8   end if;
9   exception
10  when password_is_right then
11  RAISE_APPLICATION_ERROR(-20001, 'The password should be more than 5 symbols');
12  end;
```

Results | Explain | Describe | Saved SQL | History

```
Trigger created.
```

```
INSERT INTO site_user VALUES (101,'rurururu@mail.ru','mac',87777777777,'123', 2);
```

tory

```
ORA-20001: The password should be more than 5 symbols
ORA-06512: at "WKSP_MERAKI.BEFORE_INS_USER", line 8
ORA-04088: error during execution of trigger 'WKSP_MERAKI.BEFORE_INS_USER'
```

- Create a trigger before insert on any entity which will show the current number of rows in the table:

Language PL/SQL Rows 10 Clear Command Find Tables

```
1  CREATE OR REPLACE TRIGGER rows_trigger
2  BEFORE INSERT ON product
3  FOR EACH ROW
4  DECLARE
5      r_count NUMBER;
6  BEGIN
7      SELECT COUNT(*) INTO r_count FROM product;
8      IF r_count >= 105 THEN
9          RAISE_APPLICATION_ERROR(-20001, 'The table has maximum rows');
10         ELSE
11         DBMS_OUTPUT.PUT_LINE('Current number of rows in table: ' || r_count);
12     END IF;
13 END;
```

Results Explain Describe Saved SQL History

```
Trigger created.

0.04 seconds
```

Language PL/SQL Rows 10 Clear Command Find Tables

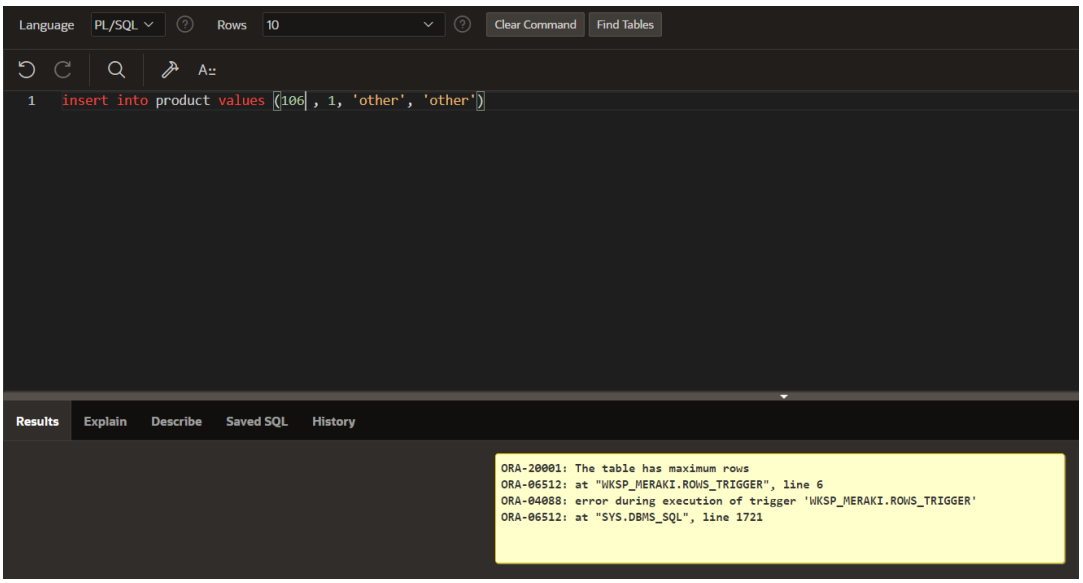
```
1  insert into product values (104 , 1, 'other', 'other')
```

Results Explain Describe Saved SQL History

```
Current number of rows in table: 104

1 row(s) inserted.

0.01 seconds
```



Trigger shows us the current number of rows before insert.
the meaning of the trigger before insert is to prevent insert in condition the number of rows should not exceed 105
Because if the number of products sold has increased to 105, this means that you need to make changes to the database and in many tables.