Laboratory work 7

Student: Almat Begaidarov
Instructor: Aibek Kuralbaev
Assistant: Bermagambet Duisek

1. How can we store large-object types?

Large objects (photos, videos, CAD files, etc.) are stored as a large object:

blob: binary large object -- object is a large collection of uninterpreted binary data (whose interpretation is left to an application outside of the database system). Stores any kind of data in binary format such as images, audio, and video.

clob: character large object -- object is a large collection of character data. Stores string data in the database having character set format. Used for large set of characters/strings or documents that use the database character.

When a query returns a large object, a pointer is returned rather than the large object itself.

2. What is the difference between privilege, role and user?

- create *accountant*, *administrator*, *support* roles and grant appropriate privileges
- create some users and assign them roles,
- give to some of them permission to grant roles to other users
- revoke some privilege from particular user

Privilege - the method, when you may authorize the user all, none, or a combination of these types of privileges on specified parts of a database, such as a relation or a view.

Role - a way to distinguish among various users -- as far as what these users can access/update in the database.

User - need to be assigned to the role.

```sql
-- 2.1
CREATE ROLE accountant;
CREATE ROLE administrator;
CREATE ROLE support;

GRANT SELECT, INSERT, UPDATE on accounts, transactions to accountant;
GRANT ALL PRIVILEGES on accounts, customers, transactions to administrator;
GRANT INSERT, UPDATE, DELETE on accounts, customers to support;

-- 2.2
CREATE USER head;
CREATE USER user1;
CREATE USER user2;
CREATE USER user3;

GRANT administrator to user1;
GRANT accountant to user2;
GRANT support to user3;

-- 2.3
CREATE ROLE director CREATEROLE;
GRANT director to head;

-- 2.4
REVOKE INSERT on accounts, customers FROM support;
```

3. Add appropriate constraints
    - check if transaction has same currency for source and destination accounts (use assertion)
    - add not null constraints

```sql
-- 3.2
ALTER TABLE customers
    ALTER COLUMN name SET NOT NULL;
ALTER TABLE customers
    ALTER COLUMN birth_date SET NOT NULL;
```

4. Change currency column type to user-defined in accounts table

5. Create indexes:
   - index so that each customer can only have one account of one currency
   - index for searching transactions by currency and balance

```
-- 5.1
CREATE UNIQUE INDEX account_currency on accounts(customer_id, currency);
-- 5.2
CREATE INDEX currency_balance on accounts(currency, balance);
```

6. Write a SQL transaction that illustrates money transaction from one account to another:
   - create transaction with “init” status
   - increase balance for destination account and decrease for source account
   - if in source account balance becomes below limit, then make rollback
   - update transaction with appropriate status(commit or rollback)

```
DO
$$
    DECLARE
        bal INT;
        lim INT;
    BEGIN
--      SAVEPOINT save1;
        UPDATE accounts
        SET balance = balance - 1000
        WHERE account_id = 'RS88012';
        UPDATE accounts
        SET balance = balance + 1000
        WHERE account_id = 'NT10204';
        SELECT balance INTO bal FROM accounts
        WHERE account_id = 'RS88012';
        SELECT accounts.limit INTO lim FROM accounts
        WHERE account_id = 'RS88012';
        IF bal < lim THEN
            UPDATE transactions SET status = 'rollback' WHERE id = 1;
            ROLLBACK;
--          ROLLBACK TO save1;
```

```
        ELSE
            UPDATE transactions SET status = 'commited' WHERE id = 1;
            COMMIT;
        END IF;
    END;
$$
```