PRÀCTICA 2

Bases de Dades

Marc Llort Maulion – marc.llort.2016
Alex Almansa Casanoves – alex.almansa.2016

Índex

Resum de l'enunciat

Fase 1

A causa de que molts alumnes estan enganxats a netflix, ens cal connectar-nos al servidor Puigpedrós de la Salle i implementar un buscador de pel·lícules amb java. Aquest servei de buscar pel·lícules ha d'estar disponible buscant pel·lícules a través del servidor de la salle, o be des de local, per tant caldrà que descarreguem la base de dades localment.

Caldrà connectar-nos al servidor per ssh, i des de el servidor fer us de les comandes de mysql per connectar-nos a la base de dades i interactuar amb ella per descobrir el esquema de la base de dades i posteriorment la importi al local.

Apart, caldrà descriure els conceptes teòrics següents: OLTP, OLAP i Data Warehouse.

Finalment s'haurà d'entregar la memòria explicant cada punt del procés dut a terme per realitzar la pràctica.

Fase 2

A la fase dos, haurem de, amb la interfície proporcionada executar diferents comandes, com poden ser cerques avançades a dins de la bbdd que importarem des de el servidor Puigpedrós.

Per poder arribar a realitzar les cerques, ens caldrà estar registrats a la plataforma LSMovies. En cas d'estar-ho, es descarregarà la base de dades del servidor i es guardarà en local per poder realitzar les diferents cerques.

En cas de no estar registrat podràs registrar-te, sempre tenint en compte que el usuari no estigui repetit amb algun ja existent.

Explicació Conceptes Teòrics

OLTP (On-Line Transaction Processing)

És un procediment que de forma relativament fàcil, ens permet realitzar una gran quantitat de transaccions curtes com pot ser un INSERT, UPDATE o un DELETE. Es tracta de processos que acostumen a ser consultes ràpides, i d'accés a registres individuals. Quan es fa us de OLTP, es per tractar dades detallades i actuals. L'eficiència es mesura per la quantitat d'ordres per segon.

De forma resumida, es tracta de una base de dades que farem servir quan vulguem tractar amb dades simples i on haguem de fer us d'accions curtes i fàcils, com per exemple canviar el preu de un producte, o borrar-lo.

OLAP (On-Line Analytical Processing)

En el cas de les OLAP, es fa servir per buscar dades històriques que no es fan amb molta freqüència i que son complexes de fer. S'utilitzen molt alhora de minar dades i fer anàlisis, i la seva eficiència es mesura en el temps que triguen a retornar els resultats. La estructura en que estan guardades les dades, està optimitzada per tractar amb grans quantitats de dades que amb una base de dades normal, trigaríem més.

Data warehouse

Es tracta de grans quantitats de informació emmagatzemades en un sistema especialitzat, pensat per poder realitzar certes tasques de forma ràpida i òptima. La majoria de sistemes OLAP es basen en data warehouse's dissenyats específicament per les tasques requerides.

En aquests magatzems de dades emmagatzemem dades en les quals ja no hi treballem, son dades simplement per consultar. Al tenir tanta informació, dins de el data warehouse, trobem que hi ha diferents subsistemes per diferents unitats de informació.

Diagrames

Diagrama Conceptual

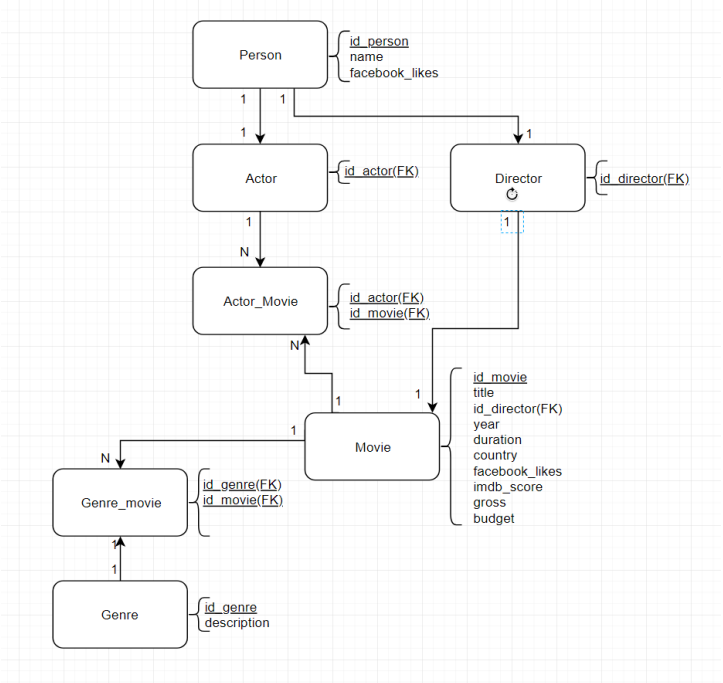
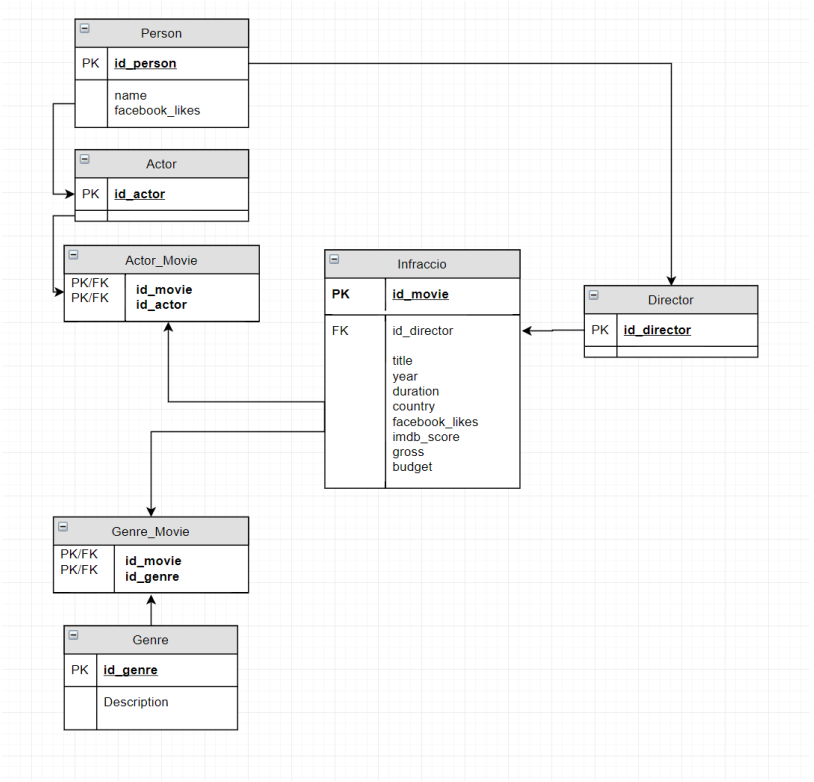


Diagrama Relacional



Mètode obtenció BBDD

Explicació

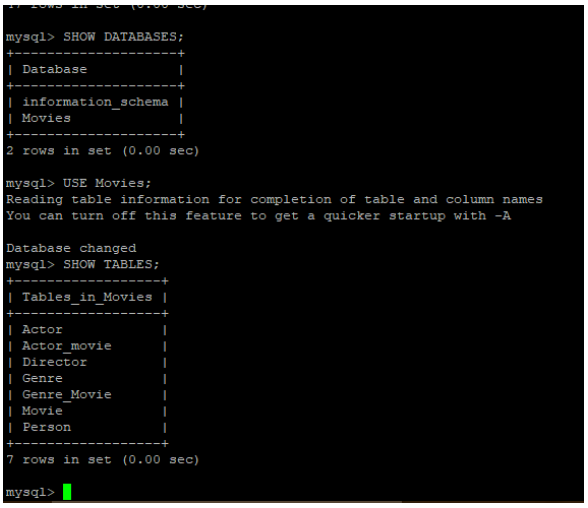
Per aconseguir saber de quina base de dades es tractava, i veure quines taules contenia la mateixa, vàrem fer us del terminal amb la comanda: *SHOW DATABASE* i posteriorment *SHOW TABLES*.

La segona forma es fent un SELECT de on estan guardades les taules (els data base) i les taules que conté. Per fer-ho, ens caldrà fer us de la següent queri:

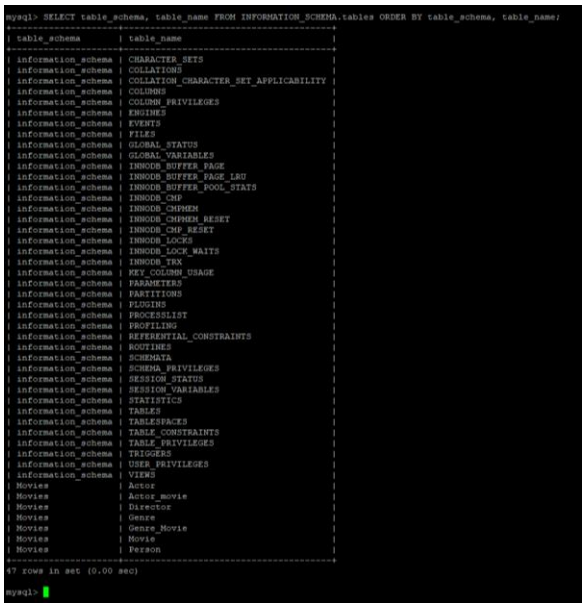
SELECT table_schema, table_name FROM INFORMATION_SCHEMA.tables ORDER BY table_schema, table_name;

Captures de codi i resultats

Manera 1



Manera 2



Exportació i Importació

Explicació

Alhora de realitzar la exportació de la base de dades des de al servidor al local, nosaltres hem decidit agafar el model de la base de dades que trobem al servidor, i creem una base de dades idèntica al local, per tant ens funcionaria amb diferents bases de dades que no sempre siguin la de Movies (en aquest primer pas).

Per exportar primer mirem les taules i columnes. Un cop estem a cada columna guardem el nom i el tipus i els afegim a un array. Un cop acabat, correm per el array i fem un “drop table” per si existís i la tornem a crear amb els noms i tipus de les columnes.

En un primer moment volíem fer que funciones amb qualsevol tipus de bbdd, i el primer pas de crear la nova bbdd ens funciona correctament, però en el 2n pas, en el de un cop exportades les dades realitzar la importació a la base de dades local vam trobar-nos amb molts problemes per realitzar-ho, i vam fer directament que estigues pensada per importar la taula movies. El principal problema amb el que ens vam trobar va ser a l’hora de rebre la informació de la querie ja que per exemple si es un int haurem de fer un de fer un getInt i si es un string un getString. Per això principalment no hem pogut fer que funcioni per qualsevol base de dades genèrica. Vam pensar també a fer-ho tot en strings a la base de dades local, però després vam pensar que potser això ens generaria problemes a l’hora de fer les quèries.

Per importar fem un “switch” de cada taula diferent, fem un “select *” de la taula que volíem copiar del servidor, fem un “while rs.next()” per iterar per totes les files que ens ha retornat la queri, i dins del “while” anem fent un “insert into” de la nostra base de dades local.

Captures

Copia de l’estructura de la base de dades que hi ha al servidor (serveix per qualsevol bbdd)

```
13  public class Main {
14      public static void main(String[] args) throws SQLException {
15          ArrayList<String> tables= new ArrayList<>();
16          //Conexio amb bdd local
17          ConectorDB local = new ConectorDB("root","alex", "localhost", 3306, "Movies");
18          local.connect();
19          //conexio amb bdd server
20          ConectorDB bbdd = new ConectorDB("grup4","marcalex", "puigpedros.salleurl.edu", 3306, "Movies");
21          bbdd.connect();
22  //Intent d'script que fa copia general de qualsevol bdd, crea la copia de qualsevol bdd però no insereix la info
23          local.insertQuery("DROP DATABASE Movies");
24          local.insertQuery("CREATE DATABASE Movies");
25          local.insertQuery("Use Movies");
26  //Guardem els noms de totes les taules que hi ha a la bdd
27          ResultSet rs = bbdd.selectQuery("SHOW TABLES");
28          while(rs.next()){
29              tables.add(rs.getString("Tables_in_movies"));
30          }
31          //iterem per totes les taules fent un select de totes les seves columnes i els corresponents tipus
32          for (String t: tables){
33              System.out.println("TAULA : "+t);
34              ResultSet rsss = bbdd.selectQuery("SHOW COLUMNS FROM "+ t);
35              ArrayList<String> info= new ArrayList<>();
36              ArrayList<String> tipu= new ArrayList<>();
37
38              while (rsss.next()) {
39
40                  info.add(rsss.getString("Field"));
41                  tipu.add(rsss.getString("Type"));
42              }
```

```
            //creem la taula amb les columnes del mateix tipus que la original
            String querie;
            String querie1;
            int i = 0;
            querie1 ="DROP TABLE IF EXISTS "+ t+";";
            querie = " CREATE TABLE "+t+ "(";
            while ( i < info.size()-1 ){
                querie+= info.get(i)+ " "+ tipu.get(i) + " , ";
                i++;
            }
            querie+= info.get(i)+ " "+ tipu.get(i)+ ");";
            System.out.println(querie1);
            local.insertQuery(querie1);
            System.out.println(querie);
            local.insertQuery(querie);
```

Importació de la informació de la bdd del servidor a la nostra bdd local

```
            switch (t) {

                case "Actor":
                    local.insertQuery("ALTER TABLE Actor ADD PRIMARY KEY (id_actor)");
                    while (rss.next()) {
                        for (String p : info) {

                            int f = rss.getInt("id_actor");
                            local.insertQuery("INSERT INTO Actor (id_actor) VALUE (" + f + ")");
                        }
                    }break;

                case "Movie":
                    local.insertQuery("ALTER TABLE Movie ADD PRIMARY KEY (id_movie);");

                    while (rss.next()) {

                            int id_movie = rss.getInt("id_movie");
                            String title = rss.getString("title");
                            int id_director = rss.getInt("id_director");
                            int year = rss.getInt("year");
                            int duration = rss.getInt("duration");
                            String country = rss.getString("country");
                            int movie_facebook_likes = rss.getInt("movie_facebook_likes");
                            float imdb_score = rss.getFloat("imdb_score");
                            int gross = rss.getInt("gross");
                            BigDecimal budget = rss.getBigDecimal("budget");
                        System.out.println("INSERT INTO Movie(id_movie,title,id_director,year,duration,country, movie_facebook_lik
                                " VALUES ("+ id_movie+","+'"'+title+'"'+","+id_director+","+year+","+duration+",'"+country+"',"+mo
                            local.insertQuery("INSERT INTO Movie(id_movie,title,id_director,year,duration,country, movie_facebook_
                                    " VALUES ("+ id_movie+","+'"'+title+'"'+","+id_director+","+year+","+duration+",'"+country+"',
```

```
    //Inserim tota la informacio de les taules del servidor a les locals
    ResultSet rss = bbdd.selectQuery("SELECT * FROM " + t);

    switch (t) {

        case "Actor":
            local.insertQuery("ALTER TABLE Actor ADD PRIMARY KEY (id_actor)");
            while (rss.next()) {
                for (String p : info) {

                    int f = rss.getInt("id_actor");
                    local.insertQuery("INSERT INTO Actor (id_actor) VALUE (" + f + ")");
                }
            }break;

        case "Movie":
            local.insertQuery("ALTER TABLE Movie ADD PRIMARY KEY (id_movie);");

            while (rss.next()) {

                    int id_movie = rss.getInt("id_movie");
                    String title = rss.getString("title");
                    int id_director = rss.getInt("id_director");
                    int year = rss.getInt("year");
                    int duration = rss.getInt("duration");
                    String country = rss.getString("country");
                    int movie_facebook_likes = rss.getInt("movie_facebook_likes");
                    float imdb_score = rss.getFloat("imdb_score");
                    int gross = rss.getInt("gross");
                    BigDecimal budget = rss.getBigDecimal("budget");
                System.out.println("INSERT INTO Movie(id_movie,title,id_director,year,duration,country, movie_facebook_likes, imdb_score, gross, budget) " +
                        " VALUES ("+ id_movie+","+'"'+title+'"'+","+id_director+","+year+","+duration+",'"+country+"',"+movie_facebook_likes+","+imdb_score+","+gross+","+budget+")");
                local.insertQuery("INSERT INTO Movie(id_movie,title,id_director,year,duration,country, movie_facebook_likes, imdb_score, gross, budget) " +
                        " VALUES ("+ id_movie+","+'"'+title+'"'+","+id_director+","+year+","+duration+",'"+country+"',"+movie_facebook_likes+","+imdb_score+","+gross+","+budget+")");

            }break;
```

```
            case "Actor_movie":
                local.insertQuery("ALTER TABLE actor_Movie ADD PRIMARY KEY (id_actor,id_movie);");
                while (rss.next()) {
                    int id_actor = rss.getInt("id_actor");
                    int id_movie = rss.getInt("id_movie");
                    local.insertQuery("INSERT INTO Actor_movie (id_actor, id_movie) VALUES (" + id_actor+","+id_movie + ")");

                }break;
            case "Director":
                local.insertQuery("ALTER TABLE Director ADD PRIMARY KEY (id_director);");
                while (rss.next()) {
                    int id_director = rss.getInt("id_director");

                    local.insertQuery("INSERT INTO Director (id_director) VALUES ("+ id_director + ")");

                }break;

            case "Genre":
                local.insertQuery("ALTER TABLE Genre ADD PRIMARY KEY (id_genre);");
                while (rss.next()) {
                    int id_genre = rss.getInt("id_genre");

                    String description = rss.getString("description");

                    local.insertQuery("INSERT INTO Genre (id_genre, description) VALUES (" + id_genre +",'"+ description + "')");

                }

            case "Genre_movie":
                local.insertQuery("ALTER TABLE Genre_movie ADD PRIMARY KEY (id_genre,id_movie );");
                while (rss.next()) {
                    int id_genre = rss.getInt("id_genre");
                    int id_movie = rss.getInt("id_movie");
                    local.insertQuery("INSERT INTO genre_movie (id_genre, id_movie) VALUES (" + id_genre+","+id_movie + ")");

                }
                break;
            case "Person":
                local.insertQuery("ALTER TABLE Person ADD PRIMARY KEY (id_person );");
                while (rss.next()) {
                    int id_person = rss.getInt("id_person");
                    String name = rss.getString("name");
                    int facebook_likes = rss.getInt("facebook_likes");

                    local.insertQuery("INSERT INTO Person (id_person, name, facebook_likes) VALUES (" + id_person+"," +'"'+name +'"' +","+facebook_likes+ ")");

                }
                break;
```

```
//Afegim les fk a les taules locals
        local.insertQuery("ALTER TABLE Movie ADD FOREIGN KEY (id_director) REFERENCES Director (id_director);");
        local.insertQuery("ALTER TABLE Actor_movie ADD FOREIGN KEY (id_actor) REFERENCES Actor (id_actor);");
        local.insertQuery("ALTER TABLE Actor_movie ADD FOREIGN KEY (id_movie) REFERENCES Movie (id_movie);");
        local.insertQuery("ALTER TABLE Actor ADD FOREIGN KEY (id_actor) REFERENCES Person (id_person);");
        local.insertQuery("ALTER TABLE Director ADD FOREIGN KEY (id_director) REFERENCES Person (id_person);");
        local.insertQuery("ALTER TABLE Genre_movie ADD FOREIGN KEY (id_genre) REFERENCES Genre (id_genre);");
        local.insertQuery("ALTER TABLE Genre_movie ADD FOREIGN KEY (id_movie) REFERENCES Movie (id_movie);");
```

Gestió d'usuaris

Creació

Per poder gestionar quins usuaris tenim registrats a la plataforma LSMovies, crearem els nous usuaris a la nostra base de dades local, i li proporcionarem el privilegi de poder fer selects per així, posteriorment poder realitzar les busquedes necessàries.

Per crear un usuari farem:

CREATE USER 'user1'@'localhost' IDENTIFIED BY 'password' i li donem privilegis amb:

*GRANT ALL PRIVILEGES ON * . * TO user1@'localhost';*

Control d'accés

Per poder entrar, ens caldrà escriure el usuari i contrasenya, i llavors executem un procedure que ens retorna el usuari i contrasenya, i en cas de que coincideixin amb les escrites per el usuari permetem que entrin.

En cas de que un usuari es registri, quan escriu el seu nou usuari i contrasenya, comprovem si el usuari ja existeix. En cas afirmatiu mostrem un missatge d'error, i sinó mostrem un missatge de que s'ha registrat amb èxit i el retornem a la pantalla inicial per a que faci el login.

Cercador

Alhora de realitzar la programació del cercador, ens hem adonat de que ens cal fer un *SELECT* simple per començar, on fem:

"*SELECT m.title, p.name, g.description, m.country, m.imdb_score FROM Movie AS m NATURAL JOIN Director AS d JOIN Person AS p ON p.id_person = d.id_director NATURAL JOIN Genre_movie NATURAL JOIN Genre as g WHERE*"

i posteriorment, a mesura que anem mirant si els camps d'informació anirem afegint condicions al *WHERE*.

Per detectar si hem afegit, per exemple, un *LIKE 'Batman'* i per tant hauríem de posar un AND per fer la següent condició, fem us de uns booleans que activem en cas de que el camp de pel·lícula, gènere, director... no estiguin buits, es a dir que el usuari ens hagi introduït un criteri de busqueda.

En el cas de que cap booleà estigues activat, és a dir que no haguessin introduït cap dada a la busqueda i tinguem que mostrar totes les pel·lícules, fem el mateix select que abans però sense el where.

Modificacions a la interfície

De modificacions a la interfície, simplement hem afegit un botó a la pantalla de realitzar el registre per així, si ens hem equivocat i hem entrat a registrar en comptes de fer el login, poder retornar a la pantalla inicial.

També ens ha semblat important modificar la taula i posar-la dins de un jscrollpane per així poder veure tots els resultats i no només els que entren a la taula, tot hi que al intentar-ho varis cops no ens ha sortit be hi no hem acabat implementa-ho.

No hem considerat realitzar altres modificacions ja que creiem que no aportarien gaire, ja que la proporcionada ja és molt funcional.

Dedicació de Temps

Fase 1

Al llarg de les 2 setmanes que hem estat treballant en, primer realitzar la part més teòrica de la pràctica, i posteriorment realitzar el codi i solucionar les diferents dificultats amb les que ens hem anat trobant, hem dedicat aproximadament unes 27 hores entre els dos.

Fase 2

Alhora de realitzar la fase 2, al ja tenir bastants coneixements adquirits durant la fase 1, i la pràctica de dpo, ens ha sigut relativament fàcil, i en unes 12 hores entre els dos hem aconseguit acabar-la.

Conclusions

Per començar, com ja hem comentat anteriorment vàrem començar investigant sobre els diferents conceptes teòrics, i vam veure com diferents sistemes, alguns desconeguts per nosaltres fins ara com el OLAP, servien per diferents utilitats, tot per aconseguir realitzar les tasques mes òptima i ràpidament.

Ens ha servit molt el aprendre a “investigar” quines taules te una base de dades de diferents maneres, i com importar de forma “manual”. Ens sembla molt interessant fer servir altres llenguatges de desenvolupament com java i fer que interactuïn amb mysql però poder realitzar comandes i diferents accions d’una forma mes user-friendly.

En un principi vam intentar fer que el nostre programa realitzes una copia de una base de dades, sense tenir en compte la “forma”/taules que tingues. Com ja hem explicat anteriorment, vam aconseguir fer que es crees una taula amb columnes i tipus correctes, però alhora de realitzar el procés de importació vam veure que era massa complicat i vam adaptar-nos al que deia el enunciat.

En quant a la implementació de la interfície gràfica, creiem que ens ajuda a veure una implementació més real del codi implementat a la fase 1, i veiem el potencial que tenen les bases de dades per poder realitzar cerques molt específiques.

El que més interesant ens ha semblat de tota la pràctica és veure la integració de sql que portem treballant tot el curs, amb un llenguatge de programació més convencional i dedicat a aplicacions com és Java, per així poder crear programes més orientats a usuaris que no cal que tinguin gaires coneixements tècnics.

Bibliografia

Que Es Un Data Warehouse? Retrieved April 11, 2018, from http://www.sinnexus.com/business_intelligence/datawarehouse.aspx

Almacén de datos. Retrieved April 11, 2018, from https://es.wikipedia.org/wiki/Almacén_de_datos

What is a data warehouse? Retrieved April 11, 2018, from https://stackoverflow.com/questions/3097917/what-is-a-data-warehouse

What are OLTP and OLAP. What is the difference between them? Retrieved April 11, 2018, from https://stackoverflow.com/questions/21900185/what-are-oltp-and-olap-what-is-the-difference-between-them

¿Qué es OLTP (Procesamiento de Transacciones En Línea)? - Definición en WhatIs.com. Retrieved April 11, 2018, from https://searchdatacenter.techtarget.com/es/definicion/OLTP-Procesamiento-de-Transacciones-En-Linea

OLTP vs. OLAP. Retrieved April 11, 2018, from http://datawarehouse4u.info/OLTP-vs-OLAP.html

Per els problemes que hem tingut durant la programació, hem fet us majoritàriament de stackoverflow.com