

Pràctica 2

Rogelio Sansaloni Sanjuan

rogelio.sansaloni

Marta Medina Lara

marta.medina

Índex

1. Resum de l'enunciat	2
2. Explicació teòrica	2
3. Model OLTP	3
4. Model relacional OLAP	5
5. Explicació Java	6
6. Creació d'usuaris	11
7. Stored procedures	12
8. Event	13
9. OLTP vs OLAP	14
Query 1:	14
OLTP	14
OLAP	14
Comentaris query 1:	15
Query 2:	15
OLTP	15
OLAP	16
Comentaris query 2:	17
Query 3:	17
OLTP	17
OLAP	17
Comentaris query 3:	18
Query 4:	18
OLTP	18
OLAP	19
Comentaris query 4:	20
Query 5:	20
OLTP	20
OLAP	20
Comentaris query 5:	21
10. Dedicació	22
11. Conclusions	22
12. Webgrafia	23

1. Resum de l'enunciat

Es demana extraure del servidor de Puigpedrós les bases de dades que utilitzarem. Ens connectarem a les bases de dades remotes mitjançant Java i copiarem les dades remotes a la base de dades local. Més tard, s'hauran de crear les taules de l'OLAP, el qual crearem desnormalitzant les taules de l'OLTP, i un esdeveniment o trigger haurà de fer que les dades es passin de l'OLTP local a l'OLAP cada dues setmanes.

S'ha de crear tres perfils d'usuari, en els quals n'hi haurà un que tindrà permès tots els SELECT, VIEW i podrà visualitzar tota la base de dades, un altre que haurà de tenir les bases de dades al dia i el tercer que tindrà permisos per a crear nous usuaris.

S'haurà de fer una comparació del funcionament del nostre OLTP amb el de l'OLAP fent varis tipus de queries i mirant el temps que triga cada una a realitzar la mateixa acció.

2. Explicació teòrica

- **OLTP:** És un tipus de processament que facilita les transaccions en línia d'aplicacions per a l'ingrés de dades. Utilitza un processament de tipus client/servidor. Es caracteritza per un gran volum de transaccions curtes (sobretot INSERT, UPDATE i DELETE).
- **OLAP:** Processament que agilitza les consultes de forma selectiva de grans quantitats de dades. S'utilitzen vàries estructures de dades (multidimensionals o Cubs OLAP). És la forma més ràpida d'executar sentències de tipus SELECT (poques transaccions llargues).
- **Data warehouse:** col·lecció de dades que estan orientades a un àmbit determinat organitzades de tal forma que permeten fàcilment depurar informació per a després poder-la processar per al seu anàlisi. S'orienta a temes, és a dir, cada element que pertany al mateix objecte o event és relacionat entre si. És variant en el temps, de forma que les dades poden canviar però no són volàtils, és a dir, que no es poden eliminar, Aquestes dades modificades quedaran guardades de forma que es puguin consultar.

3. Model OLTP

Per a aconseguir descobrir el model relacional del *Puigpedros* cal fer servir el Putty. Ens connectem amb el nostre *login* i *contrasenya*, utilitzem la comanda *mysql -u lsair_user -p* per connectar-nos a la base de dades, seleccionem la base *flight_db_00*, descobrim quines taules hi han amb *SHOW TABLES*; i el seu contingut amb *SELECT * FROM nom_taula*;

puigpedros.salle.url.edu - PuTTY

```
rogelio.sansaloni@puigpedros:~>mysql -u lsair_user -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 14299
Server version: 5.7.26-0ubuntu0.18.04.1 (Ubuntu)

Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

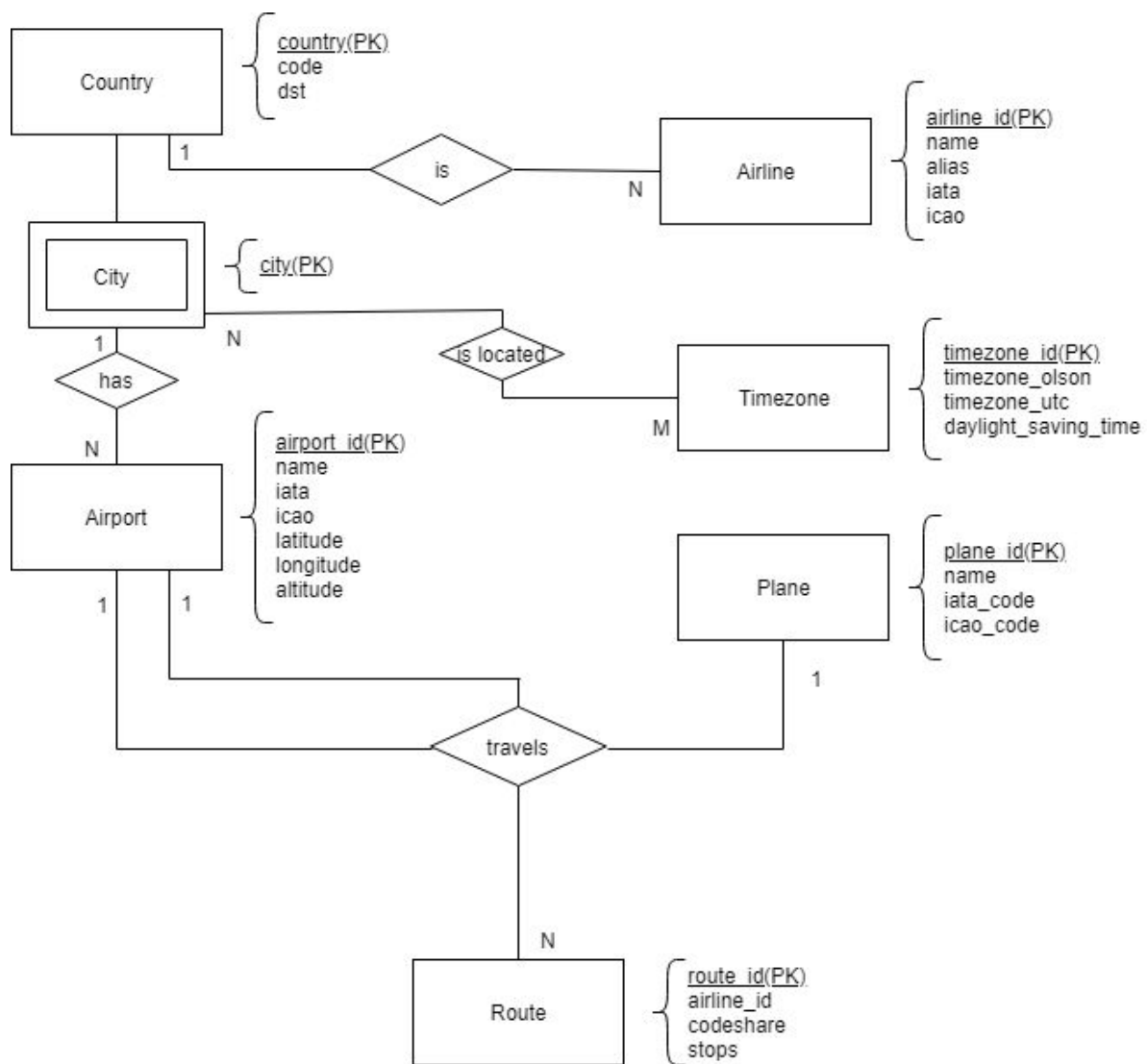
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use flight_db_00
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

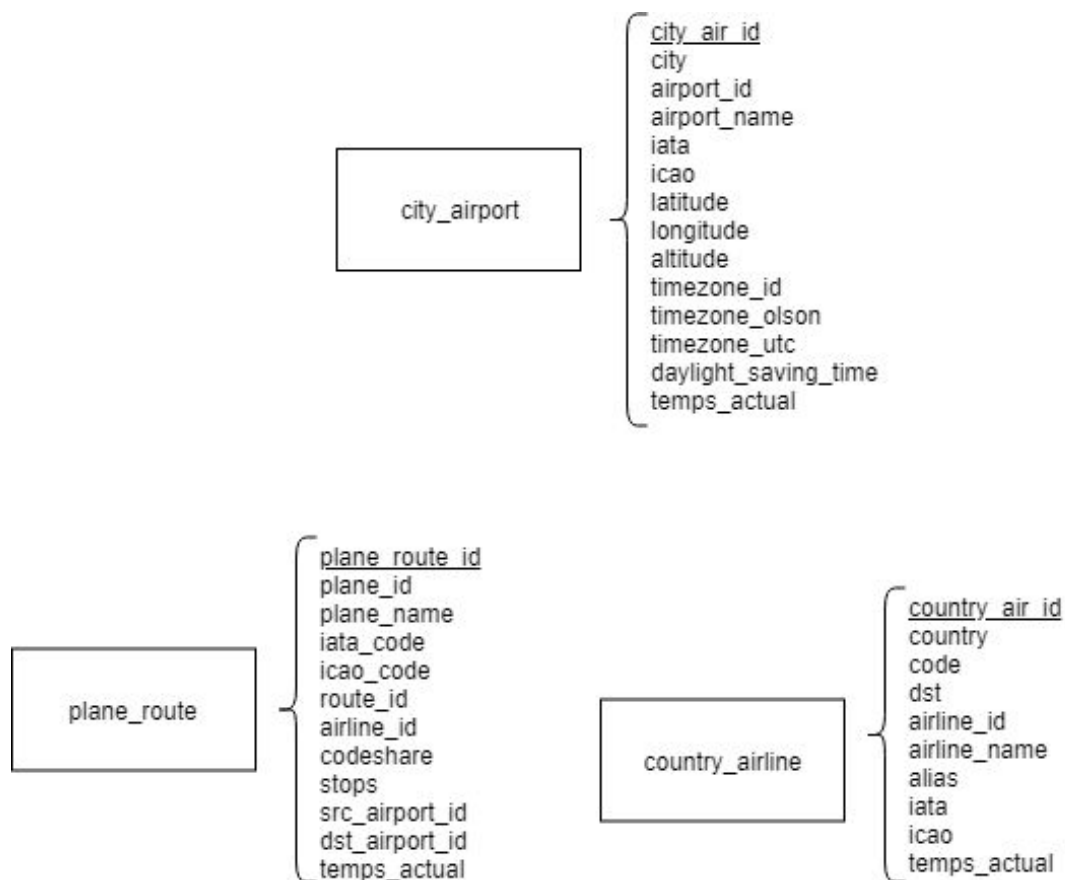
Database changed
mysql> SHOW TABLES
-> SHOW TABLES;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that
corresponds to your MySQL server version for the right syntax to use near 'SHOW
TABLES' at line 2
mysql> SHOW TABLES;
+-----+
| Tables_in_flight_db_00 |
+-----+
| airline                 |
| airport                 |
| city                    |
| country                  |
| plane                   |
| route                   |
| timezone                |
| timezone_city           |
+-----+
8 rows in set (0.00 sec)

mysql> mysql> SELECT * FROM country LIMIT 1;
ERROR 1146 (42S02): Table 'flight_db_00.airlinecountry' doesn't exist
mysql> SELECT * FROM country LIMIT 1;
+-----+-----+-----+
| country | code | dst |
+-----+-----+-----+
| Afghanistan | AF | U |
+-----+-----+-----+
1 row in set (0.00 sec)
```

Si extraïem tota la informació del servidor, obtenim el model relacional següent:



4. Model relacional OLAP



Per a aconseguir aquest model el que hem fet ha sigut simplificar (desnormalitzant) les taules que ja ens venien donades a l'OLTP de forma que hi hagués les mínimes taules, però tenint en compte que les taules que l'OLTP que ajuntéssim per a formar la de l'OLAP havien de tenir sentit juntes.

És a dir, en el cas de `city_airport` hem ajuntat les dades de les taules `city`, `airport` i `timezone` perquè un aeroport concret està a una ciutat i a la vegada a una zona horària concreta, aquesta taula desaria les dades que relacionen la part de la localització. Per exemple, no tindria sentit posar la taula `route` en aquesta, ja que no ens indica el lloc concret, sinó el recorregut que farà un avió. Aquesta taula l'hem posat, juntament amb `plane`, a la taula nova anomenada `plane_route`. Una altra nova taula de l'OLAP és `country_airline`, en la qual hi hem posat els camps de les taules de l'OLTP `country` i `airline`.

Com es pot observar, també s'han eliminat les relacions entre taules, com els FK i la taula-relació anomenada `timezone_city`.

5. Explicació Java

Classe Main

La classe **Main**, encarregada de crear les classes que representen els connectors del servidor *Puigpedros* i el *OLTP*, i de la classe encarregada de copiar la informació del servidor *Puigpedros* al *OLTP*.

```
import java.util.Scanner;
import db.*;

public class Main {
    public static void main(String[] args) {

        String password = "rsrsrsrs";
        //System.out.println("Password: ");
        //Scanner sc = new Scanner(System.in);
        //String password = sc.nextLine();

        ConectorDB connectorLocal = new ConectorDB("localhost","root", password, "OLTP", 3306);
        connectorLocal.connect();

        ConectorDB connectorServer = new ConectorDB("puigpedros.salle.url.edu", "lsair_user", "lsair_bbdd", "flight_db_00", 3306);
        connectorServer.connect();

        Copy copy = new Copy();
        copy.copiar(connectorServer, connectorLocal);
    }
}
```

Classe ConectorDB

Aquesta classe s'encarrega d'establir la connexió amb la base de dades corresponent. També conté procediments i funcions per executar queries a la base de dades des de Java. Nosaltres només farem servir la funció *connect*, que és la que estableix connexió amb la base de dades; i la funció *selectQuery*, que permet fer un select d'una base de dades, i ens servirà per extreure la informació del servidor *Puigpedros*. El procediment *cleanDatabase* no l'utilitzem però pot ser útil per netejar la base de dades OLTP abans de inserir dades.

```

package db;

import ...

public class ConectorDB {
    private String userName;
    private String password;
    private String db;
    private int port;
    private String url = "jdbc:mysql://";
    private Connection conn = null;
    private Statement s;

    public ConectorDB(String url, String usr, String pass, String db, int port) {
        this.userName = usr;
        this.password = pass;
        this.db = db;
        this.port = port; //siempre es 3306
        this.url += url + ":" + port + "/" + db + "?verifyServerCertificate=false&useSSL=true";
    }

    public boolean connect() {
        try {
            Class.forName("com.mysql.jdbc.Connection");
            conn = (Connection) DriverManager.getConnection(url, userName, password);
            if (conn != null) {
                System.out.println("La conexió a la base de dades "+url+" ha estat satisfactòria!");
            }
            return true;
        } catch (SQLException ex) {
            System.out.println("Hi ha hagut un problema al connecta-nos a la base de dades -> "+url);
        }
        catch (ClassNotFoundException ex) {
            System.out.println(ex);
        }
        return false;
    }
}

```

```

    public void insertQuery(String query){
        try {
            s =(Statement) conn.createStatement();
            s.executeUpdate(query);
        } catch (SQLException ex) {
            System.out.println("Problema al Inserir -> " + ex.getSQLState());
        }
    }

    public void updateQuery(String query){
        try {
            s =(Statement) conn.createStatement();
            s.executeUpdate(query);
        } catch (SQLException ex) {
            System.out.println("Problema al Modificar -> " + ex.getSQLState());
        }
    }

    public void deleteQuery(String query){
        try {
            s =(Statement) conn.createStatement();
            s.executeUpdate(query);
        } catch (SQLException ex) {
            System.out.println("Problema al Eliminar -> " + ex.getSQLState());
        }
    }

    public ResultSet selectQuery(String query){
        ResultSet rs = null;
        try {
            s =(Statement) conn.createStatement();
            rs = s.executeQuery (query);
        }
    }

```



```

    } catch (SQLException ex) {
        System.out.println("Problema al Recuperar les dades -> " + ex.getSQLState());
    }
    return rs;
}

public void disconnect(){
    try {
        conn.close();
        System.out.println("Desconnectat!");
    } catch (SQLException e) {
        System.out.println("Problema al tancar la connexió -> " + e.getSQLState());
    }
}

public void cleanDatabase (){
    try {
        PreparedStatement ps = conn.prepareStatement("DROP DATABASE IF EXISTS OLTP;");
        ps.executeUpdate();
        ps = conn.prepareStatement("CREATE DATABASE OLTP;");
        ps.executeUpdate();
        ps = conn.prepareStatement("Use OLTP;");
        ps.executeUpdate();

        ps = conn.prepareStatement("CREATE TABLE country(\n" +
            "    country varchar(255),\n" +
            "    code varchar(2),\n" +
            "    dst char,\n" +
            "    PRIMARY KEY (country)\n" +
            "    );");
        ps.executeUpdate();
        ps = conn.prepareStatement("CREATE TABLE city(\n" +
            "    country varchar(255),\n" +
            "    city varchar(255),\n" +
            "    foreign key (country) references country(country),\n" +
            "    PRIMARY KEY (country, city)\n" +
            "    );");

```

```

        ps.executeUpdate();
        ps = conn.prepareStatement("CREATE TABLE airline(\n" +
            "    airline_id int,\n" +
            "    name varchar(2),\n" +
            "    alias varchar(3),\n" +
            "    iata text,\n" +
            "    icao text,\n" +
            "    country varchar(255),\n" +
            "    foreign key (country) references country(country),\n" +
            "    PRIMARY KEY (airline_id)\n" +
            "    );");
        ps.executeUpdate();
        ps = conn.prepareStatement("CREATE TABLE airport(\n" +
            "    airport_id int,\n" +
            "    name text,\n" +
            "    city varchar(255),\n" +
            "    country varchar(255),\n" +
            "    iata text,\n" +
            "    icao text,\n" +
            "    latitude text,\n" +
            "    longitude double,\n" +
            "    altitude double,\n" +
            "    foreign key (country, city) references city(country, city),\n" +
            "    PRIMARY KEY (airport_id)\n" +
            "    );");
        ps.executeUpdate();
        ps = conn.prepareStatement("CREATE TABLE timezone(\n" +
            "    timezone_id int,\n" +
            "    timezone_olson varchar(255),\n" +
            "    timezone_utc double,\n" +
            "    daylight_saving_time char,\n" +
            "    PRIMARY KEY (timezone_id)\n" +
            "    );");
        ps.executeUpdate();
        ps = conn.prepareStatement("CREATE TABLE timezone_city(\n" +
            "    country varchar(255),\n" +
            "    city varchar(255),\n" +
            "    timezone_id int,\n" +

```

```

        "    foreign key (country, city) references city(country, city),\n" +
        "    PRIMARY KEY (country, city, timezone_id)\n" +
        "    );";
ps.executeUpdate();
ps = conn.prepareStatement("CREATE TABLE plane(\n" +
    "    plane_id int,\n" +
    "    name varchar(255),\n" +
    "    iata_code text,\n" +
    "    icao_code text,\n" +
    "    PRIMARY KEY (plane_id)\n" +
    "    );");
ps.executeUpdate();
ps = conn.prepareStatement("CREATE TABLE route(\n" +
    "    route_id int,\n" +
    "    airline_id int,\n" +
    "    src_airport_id int,\n" +
    "    dst_airport_id int,\n" +
    "    codeshare text,\n" +
    "    stops text,\n" +
    "    plane int,\n" +
    "    foreign key (src_airport_id) references airport(airport_id),\n" +
    "    foreign key (dst_airport_id) references airport(airport_id),\n" +
    "    foreign key (plane) references plane(plane_id),\n" +
    "    PRIMARY KEY (route_id)\n" +
    "    );");
ps.executeUpdate();

/*
CallableStatement stmt = conn.prepareCall("{call wo(?, ?)}");
stmt.setString(1, "Spain");
stmt.registerOutParameter(2, Types.INTEGER);
stmt.execute();
int num_country = stmt.getInt(2);
System.out.println("Num country = " + num_country);
*/
} catch (SQLException ex) {
    System.out.println("Problema al netejar la base de dades -> " + ex.getSQLState());
}
}
}

```

Classe Copy

La classe *Copy* és la que copia la informació des de el servidor *Puigpedros* cap al *OLTP*. Per a fer-ho primer creem i inicialitzem dos *ConnectorDB* i establim la connexió. Després cridem al procediment *copiar*, que farà el mateix per a cada taula que calgui copiar al OLTP: farà un *SELECT * FROM nom_taula*, i mentre hi hagi informació al select, l'anirà inserint al OLTP. Per a això utilitza un *PreparedStatement*, i recorre la informació amb un *while*.

```

public class Copy {

    public Copy() {
    }

    public void copiar(ConectorDB server, ConectorDB local) {

        boolean error = false;
        //String query_clean = "CALL cleanDatabase();";
        //local.selectQuery(query_clean);
        //local.cleanDatabase();

        String query_country = "SELECT * FROM country;";
        ResultSet select_country = server.selectQuery(query_country); //seleccionem la informació del country del servidor Puigpedros
        PreparedStatement ps_country;

        try {
            while (select_country.next()) {
                ps_country = local.conn.prepareStatement("INSERT INTO country(country, code, dst) VALUES (?, ?, ?);");
                ps_country.setString(1, select_country.getString("country").replace("'", " "));
                ps_country.setString(2, select_country.getString("code"));
                ps_country.setString(3, select_country.getString("dst"));
                ps_country.execute();
            }
            System.out.println("S'han copiat les dades de Country des de el servidor Puigpedros a la base de dades local OLTP.");
        } catch (SQLException e) {
            e.printStackTrace();
            error = true;
        }
    }
}

```

```

if(!error){
    String query_city = "SELECT * FROM city;";
    ResultSet select_city = server.selectQuery(query_city);
    PreparedStatement ps_city;
    try {
        while (select_city.next()) {
            ps_city = local.conn.prepareStatement("INSERT INTO city(country, city) VALUES (?, ?);");
            ps_city.setString(1, select_city.getString("country").replace("'", " "));
            ps_city.setString(2, select_city.getString("city").replace("'", " "));
            ps_city.execute();
        }
        System.out.println("S'han copiat les dades de City des de el servidor Puigpedros a la base de dades local OLTP.");
    } catch (SQLException e) {
        e.printStackTrace();
        error = true;
    }
}

if(!error){
    String query_airline = "SELECT * FROM airline;";
    ResultSet select_airline = server.selectQuery(query_airline);
    PreparedStatement ps_airline;

    try {
        while (select_airline.next()) {
            ps_airline = local.conn.prepareStatement("INSERT INTO airline(airline_id, name, alias, iata, icao, country) VALUES (?, ?, ?, ?, ?, ?);");
            ps_airline.setString(1, select_airline.getString("airline_id"));
            ps_airline.setString(2, select_airline.getString("name"));
            ps_airline.setString(3, select_airline.getString("alias"));
            ps_airline.setString(4, select_airline.getString("iata"));
            ps_airline.setString(5, select_airline.getString("icao"));
            ps_airline.setString(6, select_airline.getString("country").replace("'", " "));
            ps_airline.execute();
        }
        System.out.println("S'han copiat les dades de Airline des de el servidor Puigpedros a la base de dades local OLTP.");
    } catch (SQLException e) {
    }
}

```

```

        e.printStackTrace();
        error = true;
    }
}

if(!error){
    String query_airport = "SELECT * FROM airport;";
    ResultSet select_airport = server.selectQuery(query_airport);
    PreparedStatement ps_airport;

    try {
        while (select_airport.next()) {
            ps_airport = local.conn.prepareStatement("INSERT INTO airport(airport_id, name, city, country, iata, icao, latitude, longitude, altitude) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?);");
            ps_airport.setString(1, select_airport.getString("airport_id"));
            ps_airport.setString(2, select_airport.getString("name"));
            ps_airport.setString(3, select_airport.getString("city").replace("'", " "));
            ps_airport.setString(4, select_airport.getString("country").replace("'", " "));
            ps_airport.setString(5, select_airport.getString("iata"));
            ps_airport.setString(6, select_airport.getString("icao"));
            ps_airport.setString(7, select_airport.getString("latitude"));
            ps_airport.setString(8, select_airport.getString("longitude"));
            ps_airport.setString(9, select_airport.getString("altitude"));
            ps_airport.execute();
        }
        System.out.println("S'han copiat les dades de Airport des de el servidor Puigpedros a la base de dades local OLTP.");
    } catch (SQLException e) {
        e.printStackTrace();
        error = true;
    }
}

if(!error){
    String query_timezone = "SELECT * FROM timezone;";
    ResultSet select_timezone = server.selectQuery(query_timezone);
    PreparedStatement ps_timezone;

```

```

    try {
        while (select_timezone.next()) {
            ps_timezone = local.conn.prepareStatement("INSERT INTO timezone(timezone_id, timezone_olson, timezone_utc, daylight_saving) VALUES (?, ?, ?, ?);");
            ps_timezone.setString(1, select_timezone.getString("timezone_id"));
            ps_timezone.setString(2, select_timezone.getString("timezone_olson"));
            ps_timezone.setString(3, select_timezone.getString("timezone_utc"));
            ps_timezone.setString(4, select_timezone.getString("daylight_saving_time"));
            ps_timezone.execute();
        }
        System.out.println("S'han copiat les dades de Timezone des de el servidor Puigpedros a la base de dades local OLTP.");
    } catch (SQLException e) {
        e.printStackTrace();
        error = true;
    }
}

if(!error){
    String query_timezone_city = "SELECT * FROM timezone_city;";
    ResultSet select_timezone_city = server.selectQuery(query_timezone_city);
    PreparedStatement ps_timezone_city;

    try {
        while (select_timezone_city.next()) {
            ps_timezone_city = local.conn.prepareStatement("INSERT INTO timezone_city(country, city, timezone_id) VALUES (?, ?, ?);");
            ps_timezone_city.setString(1, select_timezone_city.getString("country").replace("'", " "));
            ps_timezone_city.setString(2, select_timezone_city.getString("city").replace("'", " "));
            ps_timezone_city.setString(3, select_timezone_city.getString("timezone_id"));
            ps_timezone_city.execute();
        }
        System.out.println("S'han copiat les dades de Timezone_city des de el servidor Puigpedros a la base de dades local OLTP.");
    } catch (SQLException e) {
        e.printStackTrace();
        error = true;
    }
}

```



```

if(!error){
    String query_plane = "SELECT * FROM plane;";
    ResultSet select_plane = server.selectQuery(query_plane);
    PreparedStatement ps_plane;

    try {
        while (select_plane.next()) {
            ps_plane = local.conn.prepareStatement("INSERT INTO plane(plane_id, name, iata_code, icao_code) VALUES (?, ?, ?, ?);");
            ps_plane.setString(1, select_plane.getString("plane_id"));
            ps_plane.setString(2, select_plane.getString("name"));
            ps_plane.setString(3, select_plane.getString("iata_code"));
            ps_plane.setString(4, select_plane.getString("icao_code"));
            ps_plane.execute();
        }
        System.out.println("S'han copiat les dades de Plane des de el servidor Puigpedros a la base de dades local OLTP.");
    } catch (SQLException e) {
        e.printStackTrace();
        error = true;
    }
}

if(!error){
    String query_route = "SELECT * FROM route;";
    ResultSet select_route = server.selectQuery(query_route);
    PreparedStatement ps_route;

    try {
        while (select_route.next()) {
            ps_route = local.conn.prepareStatement("INSERT INTO route(route_id, airline_id, src_airport_id, dst_airport_id, codeshare, stops, plane) VALUES (?, ?, ?, ?, ?, ?, ?);");
            ps_route.setString(1, select_route.getString("route_id"));
            ps_route.setString(2, select_route.getString("airline_id"));
            ps_route.setString(3, select_route.getString("src_airport_id"));
            ps_route.setString(4, select_route.getString("dst_airport_id"));
            ps_route.setString(5, select_route.getString("codeshare"));
            ps_route.setString(6, select_route.getString("stops"));
            ps_route.setString(7, select_route.getString("plane"));
            ps_route.execute();
        }
        System.out.println("S'han copiat les dades de Route des de el servidor Puigpedros a la base de dades local OLTP.");
        System.out.println("S'ha copiat tota la informació a la base de dades local OLTP!");
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
}
}

```

6. Creació d'usuaris

```

DROP USER IF EXISTS analytic_user;
DROP USER IF EXISTS manager_user;
DROP USER IF EXISTS rrhh_user;

CREATE USER analytic_user;
CREATE USER manager_user;
CREATE USER rrhh_user;

GRANT SELECT, CREATE VIEW, SHOW VIEW ON OLAP.* TO analytic_user;

GRANT ALL ON OLAP.* TO manager_user;
GRANT ALL ON OLTP.* TO manager_user;

GRANT CREATE USER ON *.* TO rrhh_user;

```

Primer hem creat els tres usuaris. Per a poder donar permisos utilitzem l'eina GRANT. A l'usuari *analytic_user* li donem permisos per crear i veure vistes així com per fer els selects que vulgui a la base de dades OLAP. Com el *manager_user* ha de mantenir les dos bases

de dades al dia, li donem tots els permisos a totes dues bases de dades. Per últim, al *rrhh_user* li donem permisos de creació d'usuaris.

7. Stored procedures

```
USE OLAP;

DELIMITER $$
DROP PROCEDURE IF EXISTS copiarInfo$$
CREATE PROCEDURE copiarInfo()
BEGIN

    DECLARE temps_actual TIMESTAMP;
    SET temps_actual = CURRENT_TIMESTAMP();

    /*Inserim a la taula de OLAP COUNTRY_AIRLINE la info de les taules de OLTP COUNTRY i AIRLINE */
    INSERT INTO OLAP.country_airline (country, code, dst, airline_id, airline_name, alias, iata, icao, temps_actual)
    SELECT DISTINCT country.country, country.code, country.dst, airline.airline_id, airline.name, airline.alias,
        airline.iata, airline.icao, temps_actual FROM OLTP.country AS country JOIN OLTP.airline AS airline ON country.country = airline.country;

    /*Inserim a la taula de OLAP CITY_AIRPORT la info de les taules de OLTP CITY, TIMEZONE_CITY, TIMEZONE i AIRPORT*/
    INSERT INTO OLAP.city_airport (city, country, airport_id, airport_name, iata, icao, latitude, longitude,
        altitude, timezone_id, timezone_olson, timezone_utc, daylight_saving_time, temps_actual)
    SELECT DISTINCT city.city, city.country, airport.airport_id, airport.name, airport.iata, airport.icao, airport.latitude, airport.longitude, airport.altitude,
        timezone.timezone_id, timezone.timezone_olson, timezone.timezone_utc, timezone.daylight_saving_time, temps_actual
    FROM OLTP.city AS city JOIN OLTP.timezone_city AS timezone_city ON city.city = timezone_city.city
    JOIN OLTP.timezone AS timezone ON timezone.timezone_id = timezone_city.timezone_id
    JOIN OLTP.airport AS airport ON airport.city = city.city AND airport.country = city.country;

    /*Inserim a la taula de OLAP AIRPORT_PLANE la info de les taules de OLTP AIRPORT, PLANE i ROUTE */
    INSERT INTO OLAP.airport_plane (plane_id, plane_name, iata_code, icao_code, route_id, airline_id, codeshare, stops, src_airport_id, dst_airport_id, temps_actual)
    SELECT DISTINCT plane.plane_id, plane.name, plane.iata_code, plane.icao_code, route.route_id, route.airline_id, route.codeshare, route.stops,
        route.src_airport_id, route.dst_airport_id, temps_actual
    FROM OLTP.plane as plane JOIN OLTP.route AS route ON plane.plane_id = route.plane;

END $$
DELIMITER ;
```

En la pràctica utilitzem un Stored Procedure que és l'encarregat de inserir la informació des de l'OLTP al OLAP. A més, insereix el temps actual en el moment de realitzar el procediment amb l'ús d'una variable anomenada *temps_actual*, que mitjançant un SET li assignem el valor de CURRENT_TIMESTAMP().

Per a la primera taula de l'OLAP (country_airline) hi inserim les dades que hi ha a les taules country i airline de l'OLTP.

Per a la segona taula de l'OLAP (city_airport) inserim les dades procedents de les taules city, timezone_city, timezone i airport de l'OLTP.

Per a la última taula de l'OLAP (airport_plane) inserim les dades de les taules airport, plane i route de l'OLTP.

8. Event

```
SET GLOBAL event_scheduler = ON;

DROP EVENT IF EXISTS actualitzaOLAP;
CREATE EVENT actualitzaOLAP
ON SCHEDULE EVERY 2 WEEK
DO
    CALL copiarInfo;
```

Primer per a què l'esdeveniment es produeixi en el moment indicat cal activar el `event_scheduler`. Després simplement el creem, indiquem que s'ha de produir cada 2 setmanes i cridem al procediment *`copiarInfo`* que hem creat anteriorment.

9. OLTP vs OLAP

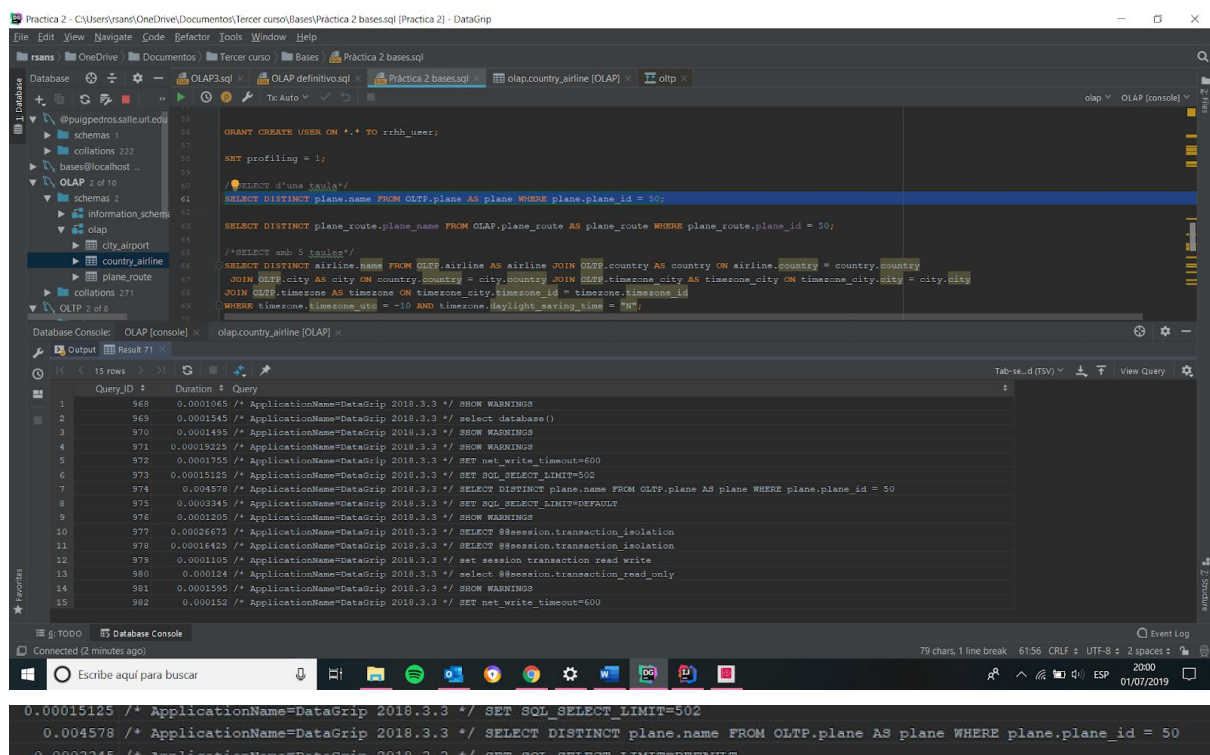
Query 1:

Seleccionem els noms dels avions en els que el seu identificador sigui '50'.

OLTP

0'004578 segons

```
SELECT DISTINCT plane.name FROM OLTP.plane AS plane WHERE plane.plane_id = 50;
```



Ampliació del resultat.

OLAP

0'02137325 segons


```

-- SQL Script
INSERT INTO OLAP.city_airport (city, country, airport_id, airport_name, iata, icao, latitude, longitude, altitude, timezone_id, timezone_name, timezone_utc, daylight_saving)
VALUES (null, null, null, null, null, null, null, null, null, null, "Europe/Barcelona", 1, "N");

--UPDATE a una taula*/
/*SELECT*/
UPDATE OLAP.airline SET name = "000" WHERE OLAP.airline.airline_id = 2;
/*DELETE*/
UPDATE OLAP.country_airline SET airline_name = "000" WHERE OLAP.country_airline.airline_id = 2;
/*DELETE a una taula*/
SHOW PROFILES;

-- Execution Results
Query_ID | Duration | Query
1 | 0.00014975 | /* ApplicationName=DataGrip 2018.3.3 */ SHOW WARNINGS
2 | 0.00015575 | /* ApplicationName=DataGrip 2018.3.3 */ select database()
3 | 0.00018675 | /* ApplicationName=DataGrip 2018.3.3 */ SHOW WARNINGS
4 | 0.00011475 | /* ApplicationName=DataGrip 2018.3.3 */ SHOW WARNINGS
5 | 0.0002245 | /* ApplicationName=DataGrip 2018.3.3 */ SET net_write_timeout=600
6 | 0.00015125 | /* ApplicationName=DataGrip 2018.3.3 */ SET SQL_SELECT_LIMIT=502
7 | 0.02137325 | /* ApplicationName=DataGrip 2018.3.3 */ SELECT DISTINCT plane_route.plane_name FROM OLAP.plane_route AS plane_route WHERE plane_route.plane_id = 50
8 | 0.00018825 | /* ApplicationName=DataGrip 2018.3.3 */ SET SQL_SELECT_LIMIT=DEFAULT
9 | 0.0001855 | /* ApplicationName=DataGrip 2018.3.3 */ SHOW WARNINGS
10 | 0.000227 | /* ApplicationName=DataGrip 2018.3.3 */ SELECT @@session.transaction_isolation
11 | 0.0002965 | /* ApplicationName=DataGrip 2018.3.3 */ SELECT @@session.transaction_isolation
12 | 0.00024125 | /* ApplicationName=DataGrip 2018.3.3 */ set session transaction read write
13 | 0.00018025 | /* ApplicationName=DataGrip 2018.3.3 */ select @@session.transaction_read_only
14 | 0.00011425 | /* ApplicationName=DataGrip 2018.3.3 */ SHOW WARNINGS
15 | 0.00018625 | /* ApplicationName=DataGrip 2018.3.3 */ SET net_write_timeout=600

```

Comentaris query 1:

La diferència de temps és de 0'01697525 segons. En aquest cas és l'OLAP el que triga més a realitzar aquest tipus de query. Això es deu a que la query només requereix una taula, per la qual cosa no s'aprofita l'avantatge d'utilitzar l'OLAP, que és molt més útil amb selects que relacionen varies taules.

Query 2:

En aquesta query seleccionarem el nom de les aerolínies en les que el seu aeroport tingui la zona horària a -10 UTC i que l'estalvi de llum sigui 'N'.

OLTP

0'27691675 segons


```
0.00012275 /* ApplicationName=DataGrip 2018.3.3 */ SET SQL_SELECT_LIMIT=502
0.14000875 /* ApplicationName=DataGrip 2018.3.3 */ SELECT DISTINCT country_airline.airline_name FROM OLAP.country_airline AS
0.00016325 /* ApplicationName=DataGrip 2018.3.3 */ SET SQL_SELECT_LIMIT=DEFAULT
```

Comentaris query 2:

En aquest cas la query triga a l'OLTP 0'136908 segons més que a l'OLAP. A diferència de la query anterior, en aquest cas sí que s'utilitzen varies taules, per la qual cosa l'OLAP és molt més útil ja que no hi ha que establir relacions entre FKs per poder fer el SELECT.

Query 3:

En aquesta query inserirem els següents valors inventats a la taula timezone:

timezone_id = 0, timezone_olson = "Europe/Barcelona", timezone_utc = 1
daylight_saving_time = "N".

OLTP

0'00741275 segons

```
INSERT INTO OLTP.timezone (timezone_id, timezone_olson, timezone_utc, daylight_saving_time)
VALUES (0, "Europe/Barcelona", 1, "N");
```

The screenshot shows the DataGrip IDE interface. The top pane displays a SQL query: `INSERT INTO OLTP.timezone (timezone_id, timezone_olson, timezone_utc, daylight_saving_time) VALUES (0, "Europe/Barcelona", 1, "N");`. The middle pane shows the execution plan, indicating the query is executed in the OLTP database. The bottom pane shows the results of the query execution, including the duration of 0.00741275 seconds.

```
0.000169 /* ApplicationName=DataGrip 2018.3.3 */ SET net_write_timeout=600
0.00741275 /* ApplicationName=DataGrip 2018.3.3 */ INSERT INTO OLTP.timezone (timezone_id, timezone_olson,
0.00016 /* ApplicationName=DataGrip 2018.3.3 */ SHOW WARNINGS
```

OLAP

0'003841 segons


```
DELETE FROM OLAP.city_airport WHERE timezone_id = 0;
INSERT INTO OLAP.city_airport (city, country, airport_id, airport_name, iata, icao, latitude, longitude, altitude, timezone_id, timezone_olson, timezone_utc, daylight_saving_time)
VALUES (null, null, null, null, null, null, null, null, null, 0, "Europe/Barcelona", 1, "N");
SHOW PROFILES;
```

Practica 2 - C:\Users\rsans\OneDrive\Documents\Tercer curso\Bases\Practica 2 bases.sql [Practica 2] - DataGrip

Database: OLAP [console] x olap.country_airline [OLAP] x oltp.timezone [OLTP] x olap.country_airline [OLAP] x oltp x

Database Console: OLAP [console] x olap.country_airline [OLAP] x oltp.timezone [OLTP] x

Query_ID	Duration	Query
1	1207	0.00021175 /* ApplicationName=DataGrip 2018.3.3 */ SELECT @@session.transaction_isolation
2	1208	0.00131025 /* ApplicationName=DataGrip 2018.3.3 */ SELECT @@session.transaction_isolation
3	1209	0.0003835 /* ApplicationName=DataGrip 2018.3.3 */ SET session transaction read write
4	1210	0.00022075 /* ApplicationName=DataGrip 2018.3.3 */ select @@session.transaction_read_only
5	1211	0.00017475 /* ApplicationName=DataGrip 2018.3.3 */ SHOW WARNINGS
6	1212	0.000207 /* ApplicationName=DataGrip 2018.3.3 */ select @@session.transaction_read_only
7	1213	0.0001465 /* ApplicationName=DataGrip 2018.3.3 */ SET net write timeout=600
8	1214	0.003841 /* ApplicationName=DataGrip 2018.3.3 */ INSERT INTO OLAP.city_airport (city, country, airport_id, airport_name, iata, icao, latitude, longitude, altitude,
9	1215	0.000169 /* ApplicationName=DataGrip 2018.3.3 */ SHOW WARNINGS
10	1216	0.0002605 /* ApplicationName=DataGrip 2018.3.3 */ SELECT @@session.transaction_isolation
11	1217	0.000026 /* ApplicationName=DataGrip 2018.3.3 */ SELECT @@session.transaction_isolation
12	1218	0.000162 /* ApplicationName=DataGrip 2018.3.3 */ set session transaction read write
13	1219	0.00063975 /* ApplicationName=DataGrip 2018.3.3 */ select @@session.transaction_read_only
14	1220	0.0001155 /* ApplicationName=DataGrip 2018.3.3 */ SHOW WARNINGS
15	1221	0.00011675 /* ApplicationName=DataGrip 2018.3.3 */ SET net write timeout=600

0.0001465 /* ApplicationName=DataGrip 2018.3.3 */ SET net_write_timeout=600

0.003841 /* ApplicationName=DataGrip 2018.3.3 */ INSERT INTO OLAP.city_airport (city, country, ai

0.000169 /* ApplicationName=DataGrip 2018.3.3 */ SHOW WARNINGS

Comentaris query 3:

Aquesta query triga 0'00357175 segons més en realitzar-se a l'OLTP que a l'OLAP. Al realitzar-se un INSERT, l'OLTP hauria de ser més ràpid, no obstant veiem que la diferència de temps entre ambdòs bases de dades és molt petita.

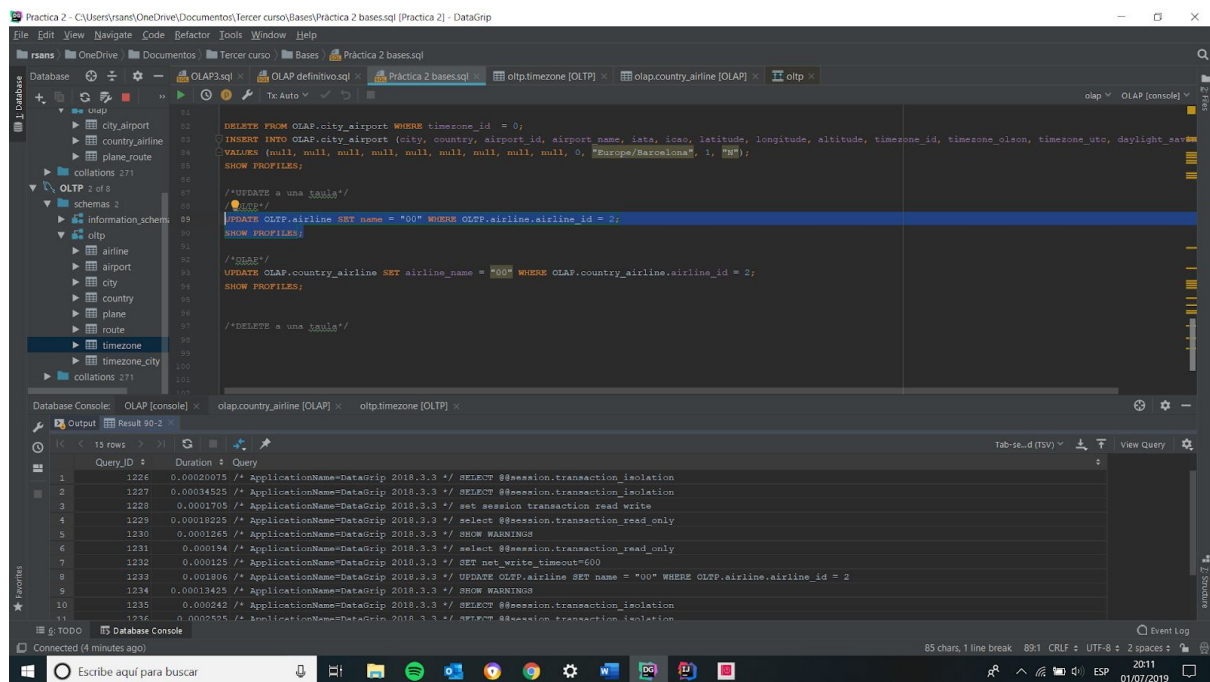
Query 4:

En aquesta query farem un UPDATE i canviarem el nom de l'aerolínia que tingui l'identificador igual a 2.

OLTP

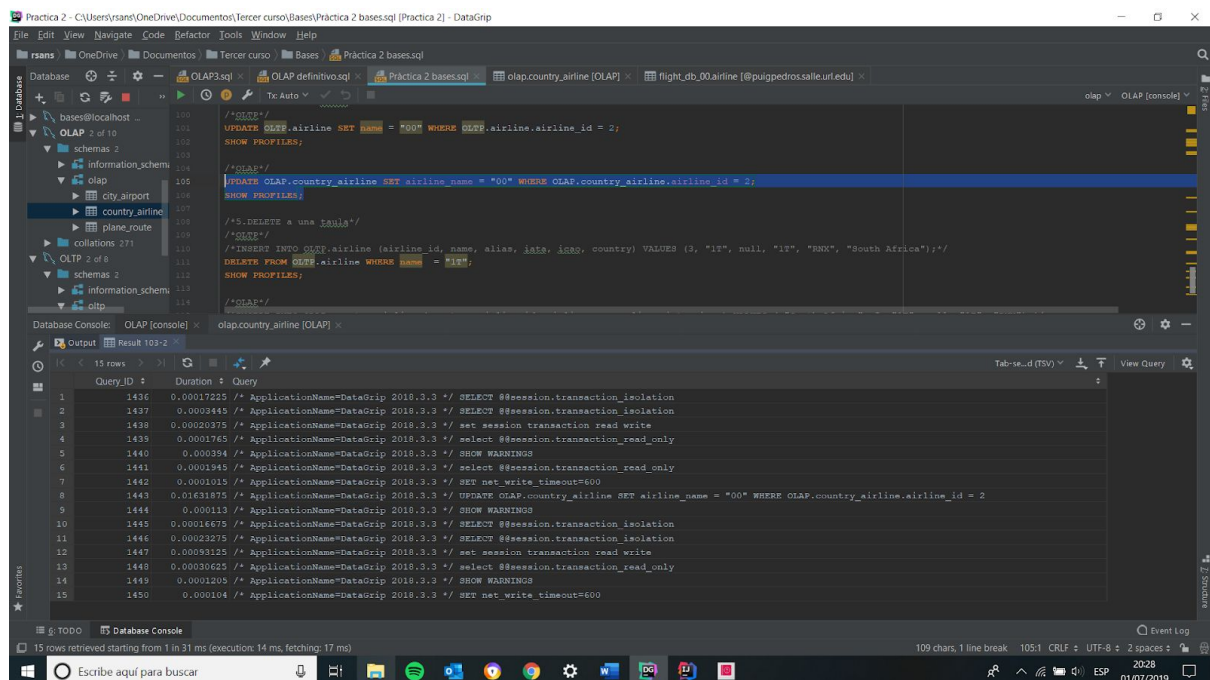
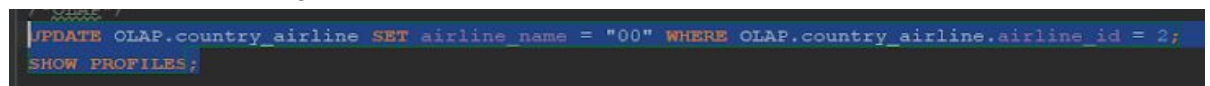
0'001806 segons

```
UPDATE OLTP.airline SET name = "00" WHERE OLTP.airline.airline_id = 2;
SHOW PROFILES;
```



OLAP

0'01631875 segundos



```

0.0001015 /* ApplicationName=DataGrip 2018.3.3 */ SET net_write_timeout=600
0.01631875 /* ApplicationName=DataGrip 2018.3.3 */ UPDATE OLAP.country_airline SET airline_name = "00" WHERE OLAP.
0.000113 /* ApplicationName=DataGrip 2018.3.3 */ SHOW WARNINGS

```

Comentaris query 4:

Aquesta query triga 0'01451275 segons més realitzar-la a l'OLAP que a l'OLTP. El resultat té sentit ja que l'OLAP hauria de ser més lent que l'OLTP al fer un UPDATE.

Query 5:

En aquesta query eliminem la fila de la taula airline en la qual el nom de l'aerolínia és "1T".

OLTP

0'0098335 segons

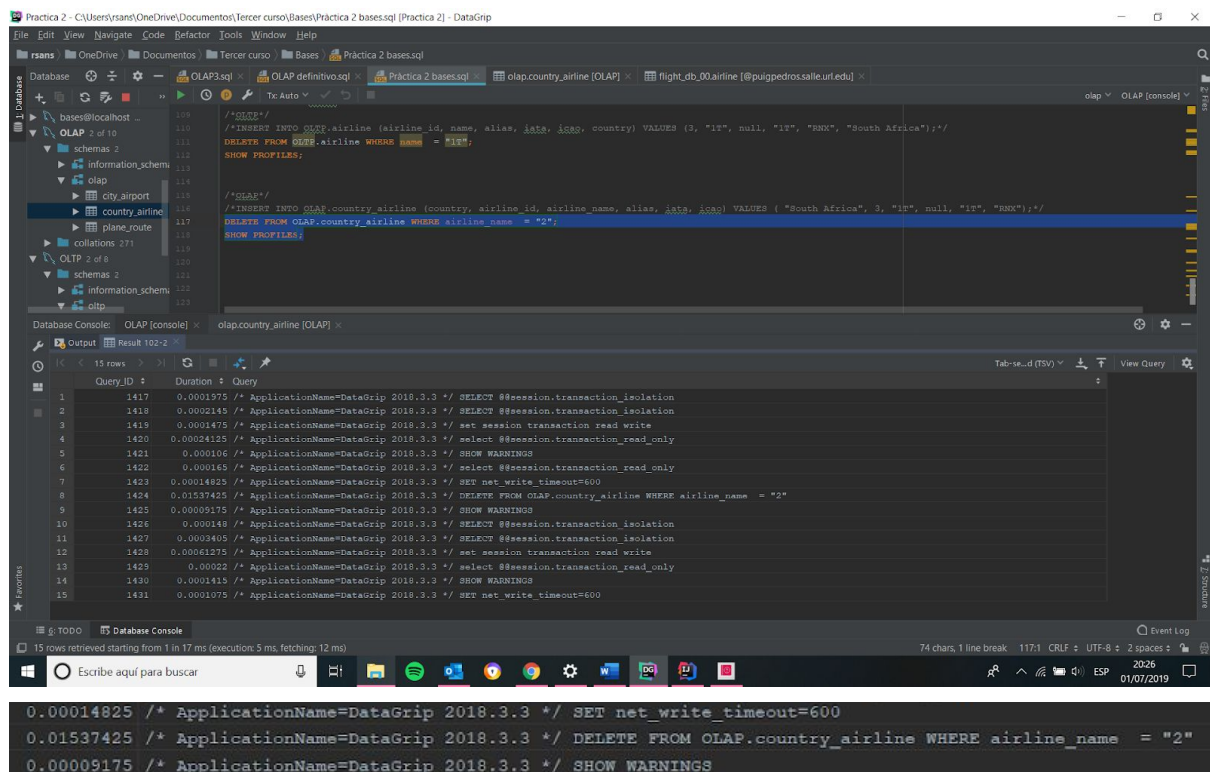
```
/*INSERT INTO OLTP.airline (airline_id, name, al
DELETE FROM OLTP.airline WHERE name = "1T";
SHOW PROFILES;
```

The screenshot shows the DataGrip application window. The top pane displays a SQL query: `/*INSERT INTO OLTP.airline (airline_id, name, al`, `DELETE FROM OLTP.airline WHERE name = "1T";`, and `SHOW PROFILES;`. The middle pane shows the database schema for 'bases@localhost', including tables like 'country_airline' and 'plane_route'. The bottom pane shows the 'Database Console' with a list of queries and their execution times. The first query is selected, showing its execution time as 0.000117 seconds. The second query is also selected, showing its execution time as 0.0098335 seconds. The third query is selected, showing its execution time as 0.000288 seconds.

OLAP

0'01537425 segons

```
/*INSERT INTO OLAP.country_airline (country, airline_id, air
DELETE FROM OLAP.country_airline WHERE airline_name = "2";
SHOW PROFILES;
```



```
/*DELETE*/
/*INSERT INTO OLAP.airline (airline_id, name, alias, iata, icao, country) VALUES (3, "1T", null, "1T", "BNK", "South Africa");*/
DELETE FROM OLAP.airline WHERE airline_name = "1T";
SHOW PROFILES;

/*DELETE*/
/*INSERT INTO OLAP.country_airline (country, airline_id, airline_name, alias, iata, icao) VALUES ('South Africa', 3, "1T", null, "1T", "BNK");*/
DELETE FROM OLAP.country_airline WHERE airline_name = "2";
SHOW PROFILES;
```

Query_ID	Query	Duration
1	1417	0.0001575 /* ApplicationName=DataGrip 2018.3.3 */ SELECT @@session.transaction_isolation
2	1418	0.0002145 /* ApplicationName=DataGrip 2018.3.3 */ SELECT @@session.transaction_isolation
3	1419	0.0001475 /* ApplicationName=DataGrip 2018.3.3 */ set session transaction read write
4	1420	0.00024125 /* ApplicationName=DataGrip 2018.3.3 */ select @@session.transaction_read_only
5	1421	0.000106 /* ApplicationName=DataGrip 2018.3.3 */ SHOW WARNINGS
6	1422	0.000165 /* ApplicationName=DataGrip 2018.3.3 */ select @@session.transaction_read_only
7	1423	0.00014825 /* ApplicationName=DataGrip 2018.3.3 */ SET net_write_timeout=600
8	1424	0.01537425 /* ApplicationName=DataGrip 2018.3.3 */ DELETE FROM OLAP.country_airline WHERE airline_name = "2"
9	1425	0.00009175 /* ApplicationName=DataGrip 2018.3.3 */ SHOW WARNINGS
10	1426	0.000148 /* ApplicationName=DataGrip 2018.3.3 */ SELECT @@session.transaction_isolation
11	1427	0.0003405 /* ApplicationName=DataGrip 2018.3.3 */ SELECT @@session.transaction_isolation
12	1428	0.00061275 /* ApplicationName=DataGrip 2018.3.3 */ set session transaction read write
13	1429	0.00002 /* ApplicationName=DataGrip 2018.3.3 */ select @@session.transaction_read_only
14	1430	0.0001815 /* ApplicationName=DataGrip 2018.3.3 */ SHOW WARNINGS
15	1431	0.0001075 /* ApplicationName=DataGrip 2018.3.3 */ SET net_write_timeout=600

15 rows retrieved starting from 1 in 17 ms (execution: 5 ms, fetching: 12 ms)

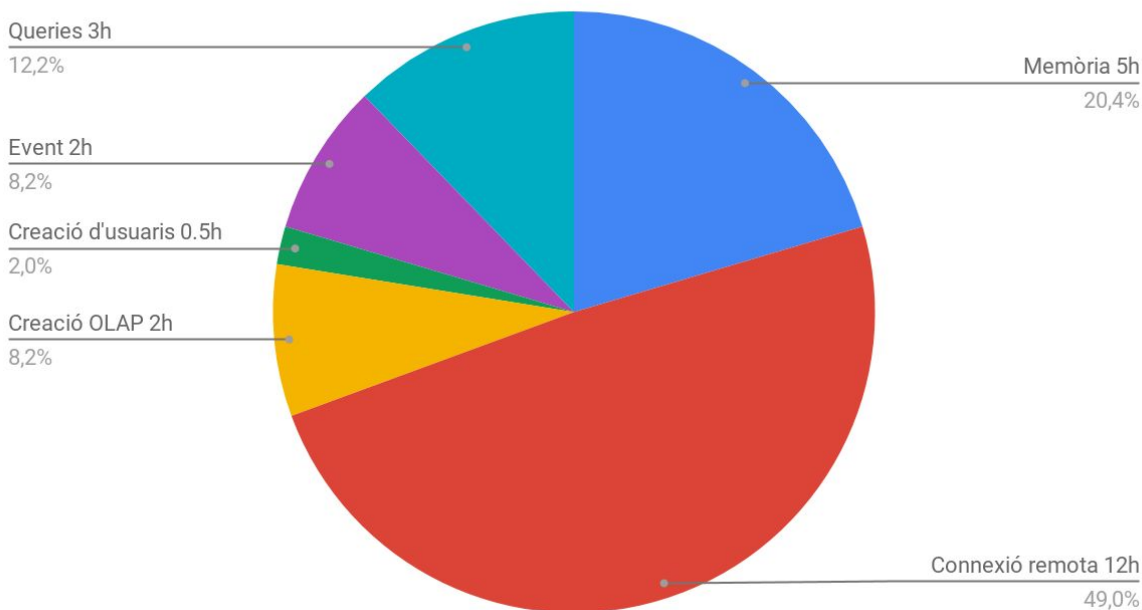
```
0.00014825 /* ApplicationName=DataGrip 2018.3.3 */ SET net_write_timeout=600
0.01537425 /* ApplicationName=DataGrip 2018.3.3 */ DELETE FROM OLAP.country_airline WHERE airline_name = "2"
0.00009175 /* ApplicationName=DataGrip 2018.3.3 */ SHOW WARNINGS
```

Comentaris query 5:

Aquesta query triga 0'00554075 segons més realitzar-la a l'OLAP que a l'OLTP. El resultat té sentit ja que el OLAP tarda més en fer un DELETE al tenir més informació que esborrar.

10. Dedicació

Temps emprat pràctica 2



11. Conclusions

Com a conclusió, aquesta pràctica ens ha estat útil per entendre millor el funcionament d'una base de dades dintre d'un entorn empresarial o de treball real. En les empreses es manegen grans quantitats de dades, per la qual cosa és necessari conèixer l'existència de diferents tipus de bases de dades com l'OLTP o l'OLAP, així com entendre el seu funcionament per saber quin utilitzar segons el cas.

D'altra banda, el fet d'utilitzar Java ha estat un handicap per un dels membres del grup ja que no ha cursat mai DPO, no obstant ha estat molt útil veure com ens podem connectar a bases de dades situades en un servidor, extreure la seva informació i insertar-la en una altra base de dades local, perquè creiem que és el procediment que se segueix habitualment.

En quant a la càrrega de treball, tot i haver-li dedicat moltes hores (sobretot al procés d'extreure dades de Puigpedros i introduir-les al OLTP), pensem que en general aquesta pràctica ha estat més senzilla que la primera. Certamen la segona pràctica requeria entendre conceptes nous i aprendre a establir connexions entre bases de dades a servidors, locals, etc., però la quantitat de feina ha estat menor que a la pràctica anterior.

12. Webgrafia

OLTP vs. OLAP [En línia] 2010 [data de consulta 26/06/2019] Disponible a:
<http://datawarehouse4u.info/OLTP-vs-OLAP.html>

OLTP [En línia] Editat el 24 de maig de 2019 [data de consulta 26/06/2019] Disponible a:
<https://es.wikipedia.org/wiki/OLTP>

OLTP (Procesamiento de Transacciones En Línea) Actualitzat el novembre de 2012 [data de consulta 26/06/2019] Disponible a:
<https://searchdatacenter.techtarget.com/es/definicion/OLTP-Procesamiento-de-Transaccion-es-En-Linea>

OLAP. Actualitzat el 6 d'abril de 2019 [data de consulta 26/06/2019] Disponible a:
<https://es.wikipedia.org/wiki/OLAP>

¿Qué es OLAP? 29 de desembre de 2011 [data de consulta 26/06/2019] Disponible a:
<https://www.businessintelligence.info/definiciones/que-es-olap.html>

¿Qué es definición de OLAP? Gener de 2015 [data de consulta 26/06/2019] Disponible a:
<https://searchdatacenter.techtarget.com/es/definicion/Definicion-de-OLAP-procesamiento-analitico-en-linea>

Datawarehouse. Sinnexus. 2016 [data de consulta 26/06/2019] Disponible a:
https://www.sinnexus.com/business_intelligence/datawarehouse.aspx

Almacén de datos. 19 de febrer de 2019 [data de consulta 26/06/2019] Disponible a:
https://es.wikipedia.org/wiki/Almacén_de_datos