Bases de dades (2019-2020)

PRÀCTICA 2

Alumnes		
Nom	**Cognoms**	**Login**
Curial	Iglesias Clua	curialjosep.iglesias
Pau	Guirao Castells	pau.guirao

laSalle
UNIVERSITAT RAMON LLULL

ÍNDEX

1. RESUM DE L'ENUNCIAT

Aquesta practica consisteix en adaptar una base de dades OLTP que conté informació de la Formula 1 a una OLAP i realitzar una sèrie de comprovacions i querys per comprovar que funciona correctament.

Els passos que hem de seguir son:
Primer accedir a la base de dades OLTP de puigpedros amb un usuari i contrasenya donats a l'enunciat.
Seguidament importar aquesta base de dades a una OLTP local mitjançant un programa JDBC de Java.
El seguent pas es crear una base de dades OLAP i importar l'informació de la OLTP local utilitzant stored procedures.
Continuem realitzant una comprovació de que s'hagi importat correctament la informació a la OLAP utilitzant un Event.
Finalment realitzem 5 querys en la base de dades OLAP

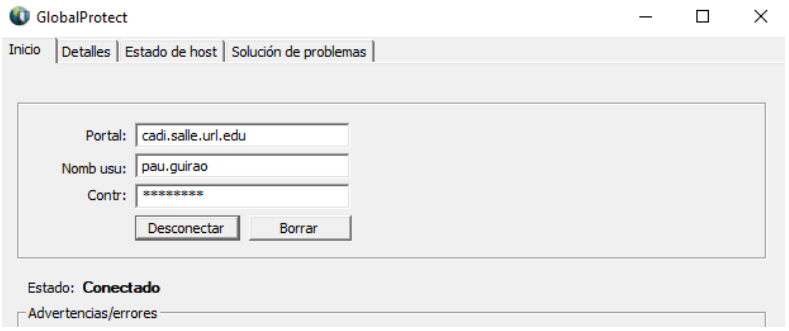
2. EXPLICACIÓ CONCEPTES OLAP I OLTP

Els sistemes OLTP són bases de dades orientades a el processament de transaccions. Una transacció genera un procés atòmic (que ha de ser validat amb un commit, o invalidat amb un rollback), i que pot involucrar operacions d'inserció, modificació i esborrat de dades

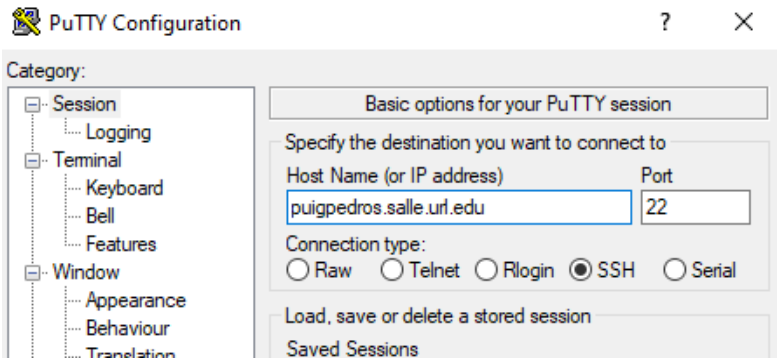
Els sistemes OLAP són bases de dades orientades a el processament analític. Aquesta anàlisi sol implicar, generalment, la lectura de grans quantitats de dades per arribar a extreure algun tipus d'informació útil: tendències de vendes, patrons de comportament dels consumidors, elaboració d'informes complexos ... etc. Aquest sistema és típic dels datamarts.

3. EXPLICACIÓ ACCEDIR OLTP REMOTA

Per trobar la OLTP remota primer de tot ens hem d'activar la VPN per poder connectar-nos als servidors de la salle.



Despres amb el putty ens connectem al servidor **puigpedros.salle.url.edu**



Un cop ens connectem ens apareixerà una finestra de comnades a on hem d'introduir el nostre login i contrasenya del estudy. Un cop les hem introduït correctament escriurem **mysql -u lsmotor_user -p** i ens demaran la contrasenya. Introduirem **lsmotor_bbdd.**

```
pau.guirao@puigpedros:~>mysql -u lsmotor_user -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8426
Server version: 5.7.30-0ubuntu0.18.04.1 (Ubuntu)

Copyright (c) 2000, 2020, 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>
```

Aqui ja podem introduir comandes mysql, i per mostrar la base de dades farem **SHOW DATABASES;**

```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| F1                 |
+--------------------+
2 rows in set (0.00 sec)

mysql>
```

Per mostrar les taules de la base de dades introduirem USE F1; per utilitzar la base de dades F1 i despres SHOW TABLES;

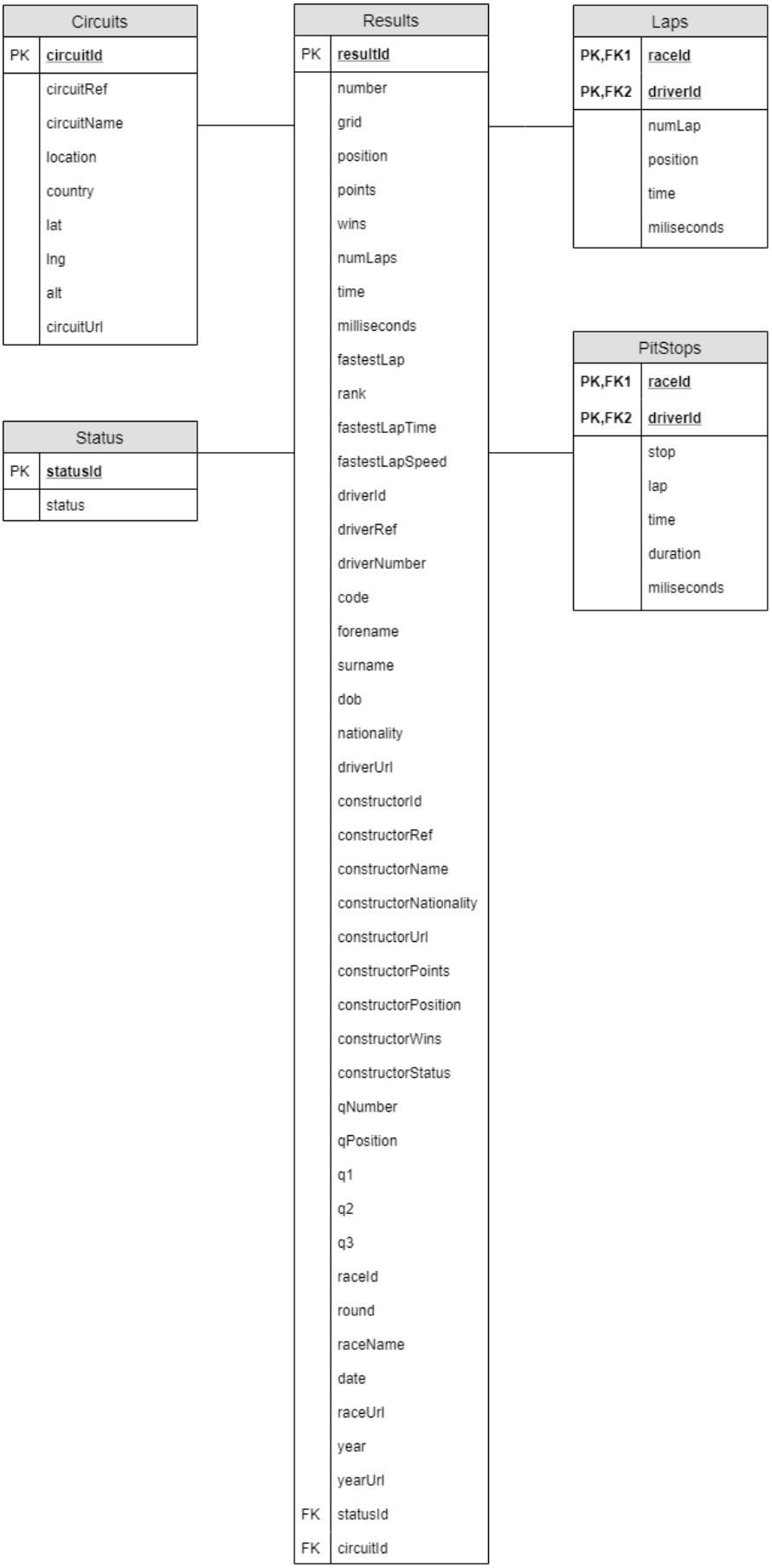
```
mysql> USE F1;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> SHOW TABLES;
+----------------------+
| Tables_in_F1         |
+----------------------+
| circuits             |
| constructorResults   |
| constructorStandings |
| constructors         |
| driverStandings      |
| drivers              |
| lapTimes             |
| pitStops             |
| qualifying           |
| races                |
| results              |
| seasons              |
| status               |
+----------------------+
13 rows in set (0.00 sec)
```

Despres per accedir a la info per taules de la base de dades faríem un describe. Per exemple per obtenir la informació dels circuits farem DESCRIBE circuits;

```
mysql> DESCRIBE circuits;
+------------+--------------+------+-----+---------+----------------+
| Field      | Type         | Null | Key | Default | Extra          |
+------------+--------------+------+-----+---------+----------------+
| circuitId  | int(11)      | NO   | PRI | NULL    | auto_increment |
| circuitRef | varchar(255) | NO   |     |         |                |
| name       | varchar(255) | NO   |     |         |                |
| location   | varchar(255) | YES  |     | NULL    |                |
| country    | varchar(255) | YES  |     | NULL    |                |
| lat        | float        | YES  |     | NULL    |                |
| lng        | float        | YES  |     | NULL    |                |
| alt        | int(11)      | YES  |     | NULL    |                |
| url        | varchar(255) | NO   | UNI |         |                |
+------------+--------------+------+-----+---------+----------------+
9 rows in set (0.00 sec)
```

4. MODEL RELACIONAL OLAP



En el nostre model OLAP hem decidit deixar les taules **Circuit**, **Laps**, **PitStops** i **Status** iguals a com estan al model OLTP i en canvi hem decidit juntar tota la resta de taules en una sola taula **Results** (Constructor, ConstructorResults, ConstructorStandings, Drivers, DriverStandings, Qualify, Races, Seasons, Results).

Hem pres aquesta decisió ja que d'aquesta forma la taula **Results** no aumenta el seu nombre de files original. Si també li haguesim afegit les taules de **Circuits**, **Laps** o **PitStops** s'hauria multiplicat diversos cops el nombre de files i creiem que no seria tan optim a l'hora de realitzar consultes.

La taula **Status** no seria necessari deixarla apart però la necessitem per realitzar la query 1.

5. EXPLICACIÓ JAVA

La nostre aplicació java segueix el model M-V-C (Model – Vista - Controlador). El controllador s'encarregara de controlar totes les funcions necessàries perquè el programa funcioni. El primer que fem es establir la connexió remota i local a partir de les funcions **startRemoteConnection** i **startLocalConnection** del Controlador.

```
public boolean startRemoteConnection(){
    try{
        Class.forName("com.mysql.cj.jdbc.Driver");
        remoteConnection = DriverManager.getConnection( url: "jdbc:mysql://puigpedros.salleurl.edu/?user=" + Settings.REMOTEUSER + "&password=" + Settings.REMOTEPASSWORD + "&serverTimezone=UTC");
        return true;
    }catch(Exception e){
        e.printStackTrace();
        return false;
    }
}

public boolean startLocalConnection(){
    try{
        Class.forName("com.mysql.cj.jdbc.Driver");
        localConnection = DriverManager.getConnection( url: "jdbc:mysql://localhost:3306/F1_oltp?allowPublicKeyRetrieval=true&useSSL=false", user: "root", password: "0205");
        return true;
    }catch(Exception e){
        e.printStackTrace();
        return false;
    }
}
```

Si veiem que la connexió tant remota com local s'han pogut duur a terme comencem a cargar la informació de totes les taules de la base de dades remota a la local amb la funció **loadDataBaseInfo.** A la funció li passem el nom de la taula que volem extreure les dades.

```
System.out.println("Connecting to Database...");
if (!mysqlController.startRemoteConnection()) System.exit( status: 1);
if (!mysqlController.startLocalConnection()) System.exit( status: 1);
```

```
public void loadDataBaseInfo(String tableName,Model model) throws SQLException
```

Dins la pròpia funció creem Statements (Querys) mitjançant la connexió remota per fer el USE de la base de dades i el SELECT de la taula que em passat per parametre. Finalment els executem.

```
stmt = remoteConnection.createStatement();
stmt.executeQuery( sql: "USE F1");
```

```
PreparedStatement remoteSt = remoteConnection.prepareStatement( sql: "select * from " + tableName + ";");
rs = remoteSt.executeQuery();
```

Extreiem el numero de columnes que te la taula per poder iterar

```
//extraiem la metadata i d'alla el numero de columnes
ResultSetMetaData rsmd = rs.getMetaData();
int col = rsmd.getColumnCount();
```

Mentres seguim tenim dades al resultat obtingut d'executar el select generem la sentencia UPDATE adequada a la taula i finalment l'executem.

```
while (rs.next()) {
    String insert = fabricarString(tableName, col);

    System.out.println("UPDATE: " + localSt);

    localSt = localConnection.prepareStatement(insert);

    for(int i=1;i<=col;i++){
        //inserim el valor de cada columna respectivament dins del update
        localSt.setString(i,rs.getString(i));
    }
    //executem el update
    localSt.executeUpdate();
}
```

Aquesta funció la cridem 13 cops al Main, una per cada taula de la base de dades remota

```
try {
    mysqlController.loadDataBaseInfo( tableName: "circuits",model);
    mysqlController.loadDataBaseInfo( tableName: "pitStops",model);
    mysqlController.loadDataBaseInfo( tableName: "lapTimes",model);
    mysqlController.loadDataBaseInfo( tableName: "results",model);
    mysqlController.loadDataBaseInfo( tableName: "races",model);
    mysqlController.loadDataBaseInfo( tableName: "drivers",model);
    mysqlController.loadDataBaseInfo( tableName: "driverStandings",model);
    mysqlController.loadDataBaseInfo( tableName: "constructors",model);
    mysqlController.loadDataBaseInfo( tableName: "constructorStandings",model);
    mysqlController.loadDataBaseInfo( tableName: "constructorResults",model);
    mysqlController.loadDataBaseInfo( tableName: "qualifying",model);
    mysqlController.loadDataBaseInfo( tableName: "seasons",model);
    mysqlController.loadDataBaseInfo( tableName: "status",model);

} catch (SQLException e) {
    e.printStackTrace();
}
```

6. EXPLICACIÓ USERS

El analytic_user s'encarrega de fer selects, creació i visualització de vistes a la base de dades OLAP.

```
DROP USER 'analytic_user';
CREATE USER 'analytic_user';
GRANT SELECT,CREATE VIEW,SHOW VIEW ON F1_OLAP.* TO 'analytic_user';
```

El manager_user serveix per mantenir les dos base de dades actualitzades Hem cregut pertinent donar grants de UPDATE,INSERT,DELETE,DROP I TRIGGER per que tots son necessaris per fer la gestió de les dos.

```
DROP USER 'manager_user';
CREATE USER 'manager_user';
GRANT UPDATE,INSERT,DELETE,DROP,EVENT,TRIGGER ON F1_OLAP.* TO 'manager_user';
GRANT UPDATE,INSERT,DELETE,DROP,EVENT,TRIGGER ON F1_OLTP.* TO 'manager_user';
```

El rrhh_user serveix per crear usuaris a les dos bases de dades. Li hem donat el GRANT de CREATE USER a les dues bases de dades

```
DROP USER 'rrhh_user';
CREATE USER 'rrhh_user';
GRANT CREATE USER ON *.* TO 'rrhh_user';
```

7. EXPLICACIÓ STORED PROCEDURES

Els stored procedures utilitzats en aquesta pràctica són els que utilitzem dins dels trigger per actualitzar la informació. Tots el trigger que s'activen quan hi ha un insert nou, criden al seu procedure corresponen. Dins d'aquest procedure hi ha el INSERT en cas de que sigui el primer cop que introduïm dades a la taula i pot haver-hi també un UPDATE en cas que volguem omplenar els nulls generats al fer un INSERT a una taula de la OLAP.

```
CREATE PROCEDURE insert_circuits(
INSERT INTO f1_olap.Circuits (circuitId, circuitRef, circuitName, location, country, lat, lng, alt, circuitUrl)
VALUES(cirId,cirRef,cirName,loc,Coun,lati,lon,alti,cUrl);
```

```
CREATE PROCEDURE insert_pitstops(
INSERT INTO f1_olap.pitstops (raceId, driverId, stop, lap, time, duration, milliseconds)
VALUES(rId,dId,stopIn,lapIn,timeIn,durationIn,milis);
```

```
CREATE PROCEDURE insert_laps
INSERT INTO f1_olap.Laps (raceId, driverId, lap, position, time, milliseconds)
VALUES(rId,dId,lapIn,posIn,timeIn,milis);
```

```
CREATE PROCEDURE insert_status
```

```
INSERT INTO f1_olap.status (statusId, status)
VALUES(sID,statusIn);
```

8. EXPLICACIÓ EVENTS I TRIGGERS

- Trigger de Circuits

```
CREATE TRIGGER trigger_circuits AFTER INSERT ON f1_oltp.circuits
```

```
INSERT INTO f1_olap.Circuits (circuitId, circuitRef, circuitName, location, country, lat, lng, alt, circuitUrl)
VALUES(NEW.circuitId,NEW.circuitRef,NEW.name,NEW.location,NEW.country,NEW.lat,NEW.lng,NEW.alt,NEW.url);
```

El trigger **trigger_circuits** insereix la tota informació de la taula circuits de la OLTP a la OLAP. Al ser les mateixes taules inserim tota la informació amb un sol trigger.

- Trigger de PitStop

```
CREATE TRIGGER trigger_pitStops AFTER INSERT ON f1_oltp.pitstops
```

```
INSERT INTO f1_olap.pitstops (raceId, driverId, stop, lap, time, duration, milliseconds)
VALUES(NEW.raceId,NEW.driverId,NEW.stop,NEW.lap, NEW.time, NEW.duration, NEW.milliseconds);
```

El trigger **trigger_pitStops** s'activa quan hi ha un INSERT a la taula pitStop de la OLTP i insereix la tota informació a la taula pitStops de la OLAP. Al ser les mateixes taules inserim tota la informació amb un sol trigger.

- Trigger de Laps

```
CREATE TRIGGER trigger_laps AFTER INSERT ON f1_oltp.laptimes
```

```
INSERT INTO f1_olap.Laps (raceId, driverId, lap, position, time, milliseconds)
VALUES(NEW.raceId,NEW.driverId,NEW.lap,NEW.position,NEW.time,NEW.milliseconds);
```

El trigger **trigger_laps** s'activa quan hi ha un INSERT a la taula laptimes de la OLTP i insereix la tota informació a la taula laps de la OLAP. Al ser les mateixes taules inserim tota la informació amb un sol trigger.

- Trigger de Status

```
CREATE TRIGGER trigger_status AFTER INSERT ON f1_oltp.status
```

```
INSERT INTO f1_olap.status (statusId, status)
VALUES(NEW.statusId,NEW.status);
```

El trigger **trigger_status** s'activa quan hi ha un INSERT a la taula status de la OLTP i insereix la tota informació a la taula status de la OLAP. Al ser les mateixes taules inserim tota la informació amb un sol trigger.

- Trigger de Results

La taula de results de la OLAP es una combinació de les taules results,races,divers,constructors,contructorresults,constructorstandings,seasons i qualifying. La taula results de la OLAP té totes les columnes d'aquestes taules amb els triggers omplenarem les files.

```
CREATE TRIGGER trigger_results_1 AFTER INSERT ON f1_oltp.results
```

```
INSERT INTO f1_olap.results(resultId, number, grid, position, points, numLaps, time, milliseconds, fastestLap, `rank`, fastestLapTime, fastestLapSpeed,driverId,statusId,constructorId,raceId)
VALUES(NEW.resultId,NEW.number,NEW.grid,NEW.position,NEW.points,NEW.laps,NEW.time,NEW.milliseconds,
       NEW.fastestLap,NEW.`rank`,NEW.fastestLapTime,NEW.fastestLapSpeed,NEW.driverId,NEW.statusId,NEW.constructorId,NEW.raceId);
```

El trigger **trigger_results** s'activa quan hi ha un INSERT a la taula results de la OLTP i insereix la tota informació a la taula results de la OLAP. Introduim totes les files de la taula results a la seva columna corresponent.

```
CREATE TRIGGER trigger_results_2 AFTER INSERT ON f1_oltp.races
```

```
UPDATE f1_olap.results SET year = NEW.year,round = NEW.round,circuitId = NEW.circuitId,raceName = NEW.name,date = NEW.date,time = NEW.time,raceUrl = NEW.url
WHERE NEW.raceId = f1_olap.results.raceId;
```

El trigger **trigger_results_2** s'activa quan hi ha un INSERT a la taula races de la OLTP i insereix la tota informació a la taula results de la OLAP. Omplenem tots els nuls generats al inserir la taula results mitjançant UPDATE

```
CREATE TRIGGER trigger_results_3 AFTER INSERT ON f1_oltp.drivers
```

```
UPDATE f1_olap.results
SET driverRef = NEW.driverRef,driverNumber = NEW.number,code = NEW.code,
forename = NEW.forename,surename = NEW.surname,dob = NEW.dob,nationality = NEW.nationality,driverUrl = NEW.url
WHERE NEW.driverId = f1_olap.results.driverId;
```

El trigger **trigger_results_3** s'activa quan hi ha un INSERT a la taula drivers de la OLTP i insereix la tota informació a la taula results de la OLAP. Omplenem tots els nuls generats al inserir la taula results mitjançant UPDATE

```
CREATE TRIGGER trigger_results_4 AFTER INSERT ON f1_oltp.constructors
```

```
UPDATE f1_olap.results SET constructorRef = NEW.constructorRef,constructorName = NEW.name,constructorNationality = NEW.nationality,constructorUrl = NEW.url
WHERE NEW.constructorId = f1_olap.results.constructorID;
```

El trigger **trigger_results_4** s'activa quan hi ha un INSERT a la taula constructors de la OLTP i insereix la tota informació a la taula results de la OLAP. Omplenem tots els nuls generats al inserir la taula results mitjançant UPDATE

```
CREATE TRIGGER trigger_results_5 AFTER INSERT ON f1_oltp.constructorstandings
```

```
UPDATE f1_olap.results SET constructorPoints = NEW.points,constructorPosition = NEW.position,constructorWins = NEW.wins
WHERE NEW.raceId = f1_olap.results.raceId AND NEW.constructorId = f1_olap.results.constructorID;
```

El trigger **trigger_results_5** s'activa quan hi ha un INSERT a la taula constructorsstandings de la OLTP i insereix la tota informació a la taula results de la OLAP. Omplenem tots els nuls generats al inserir la taula results mitjançant UPDATE

```
CREATE TRIGGER trigger_results_6 AFTER INSERT ON f1_oltp.constructorresults
```

```
UPDATE f1_olap.results SET constructorStatus = NEW.status
WHERE NEW.raceId = f1_olap.results.raceId AND NEW.constructorId = f1_olap.results.constructorID;
```

El trigger **trigger_results_6** s'activa quan hi ha un INSERT a la taula constructorresults de la OLTP i insereix la tota informació a la taula results de la OLAP. Omplenem tots els nuls generats al inserir la taula results mitjançant UPDATE

```
CREATE TRIGGER trigger_results_7 AFTER INSERT ON f1_oltp.qualifying
```

```
UPDATE f1_olap.results SET qNumber = NEW.number,qPosition = NEW.position,q1 = NEW.q1,q2 = NEW.q2,q3 = NEW.q3
WHERE NEW.raceId = f1_olap.results.raceId AND
      NEW.driverId = f1_olap.results.driverId AND
      NEW.constructorId = f1_olap.results.constructorID;
```

El trigger **trigger_results_7** s'activa quan hi ha un INSERT a la taula qualifying de la OLTP i insereix la tota informació a la taula results de la OLAP. Omplenem tots els nuls generats al inserir la taula results mitjançant UPDATE

```
CREATE TRIGGER trigger_results_8 AFTER INSERT ON f1_oltp.seasons
```

```
UPDATE f1_olap.results SET yearUrl = NEW.url
WHERE NEW.year = f1_olap.results.year;
```

El trigger **trigger_results_8** s'activa quan hi ha un INSERT a la taula seasons de la OLTP i insereix la tota informació a la taula results de la OLAP. Omplenem tots els nuls generats al inserir la taula results mitjançant UPDATE

- Event

El event compara_files ens servirà per comprovar si la importació de dades entra la OLTP i la OLAP s'ha fet correctament.

```
CREATE EVENT IF NOT EXISTS compara_files
ON SCHEDULE EVERY 5 MINUTE STARTS '2020-07-04 12:50:00'
```

El primer que fem al event es crear una taula que té com a columnes:

nom: el nom de taula que hem comprovat

oltp_num_files: el numero de files de la taula a la OLTP

olap_num_files: el numero de files de la taula a la OLAP

```
DROP TABLE IF EXISTS Comprovacio CASCADE;
CREATE TABLE Comprovacio (
    nom VARCHAR(255) PRIMARY KEY,
    oltp_num_files INT,
    olap_num_files INT
);
```

Segon, creem les variables numOLTP I numOLAP que ens serviran per contar el numero de files de les taules.

```
DECLARE numOLTP INT;
DECLARE numOLAP INT;
```

Tercer, fem els SET de les dues variables mitjançant un SELECT COUNT de totes les files de les taules de la OLTP i OLAP. Aquí podem veure un exemple de com fem el Count de la taula circuits de la OLTP i la OLAP. Un cop hem fet els dos SET farem un INSERT a la taula que hem creat per després poder comprovar si la importació s'ha fet bé.

```
SET numOLTP = (SELECT COUNT(circuitId) FROM f1_oltp.circuits);

SET numOLAP = (SELECT COUNT(circuitId) FROM f1_olap.circuits);

INSERT INTO Comprovacio(nom, oltp_num_files, olap_num_files)
VALUES ('Circuits', numOLTP, numOLAP);
```

Insert de la taula Status

```
SET numOLTP = (SELECT COUNT(statusId) FROM f1_oltp.status);

SET numOLAP = (SELECT COUNT(statusId) FROM f1_olap.status);

INSERT INTO Comprovacio(nom, oltp_num_files, olap_num_files)
VALUES ('Status', numOLTP, numOLAP);
```

Insert de la taula Laps

```
SET numOLTP = (SELECT COUNT(raceId) FROM f1_oltp.lapTimes);

SET numOLAP = (SELECT COUNT(raceId) FROM f1_olap.laps);

INSERT INTO Comprovacio(nom, oltp_num_files, olap_num_files)
VALUES ('Laps', numOLTP, numOLAP);
```

Insert de la tala PitStops

```
SET numOLTP = (SELECT COUNT(raceId) FROM f1_oltp.pitStops);

SET numOLAP = (SELECT COUNT(raceId) FROM f1_olap.pitStops);

INSERT INTO Comprovacio(nom, oltp_num_files, olap_num_files)
VALUES ('PitStops', numOLTP, numOLAP);
```

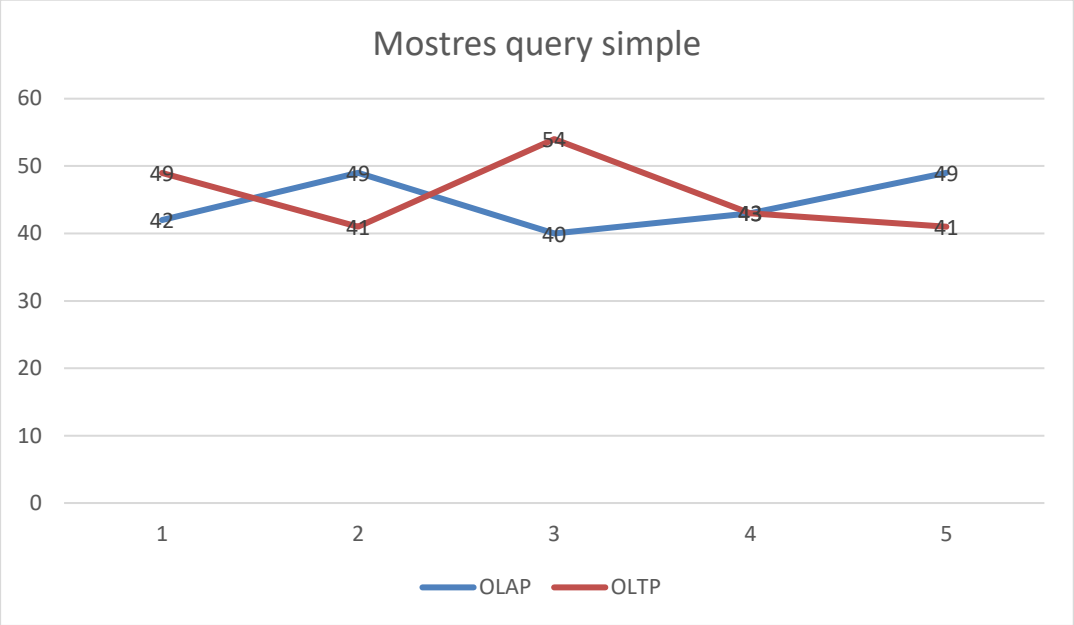
Finalment fem un select all de la taula Comprovacio i podem veure que la importació s'ha fet correctament ja que el numero de files de la OLTP correspon al numero de files de la OLAP.

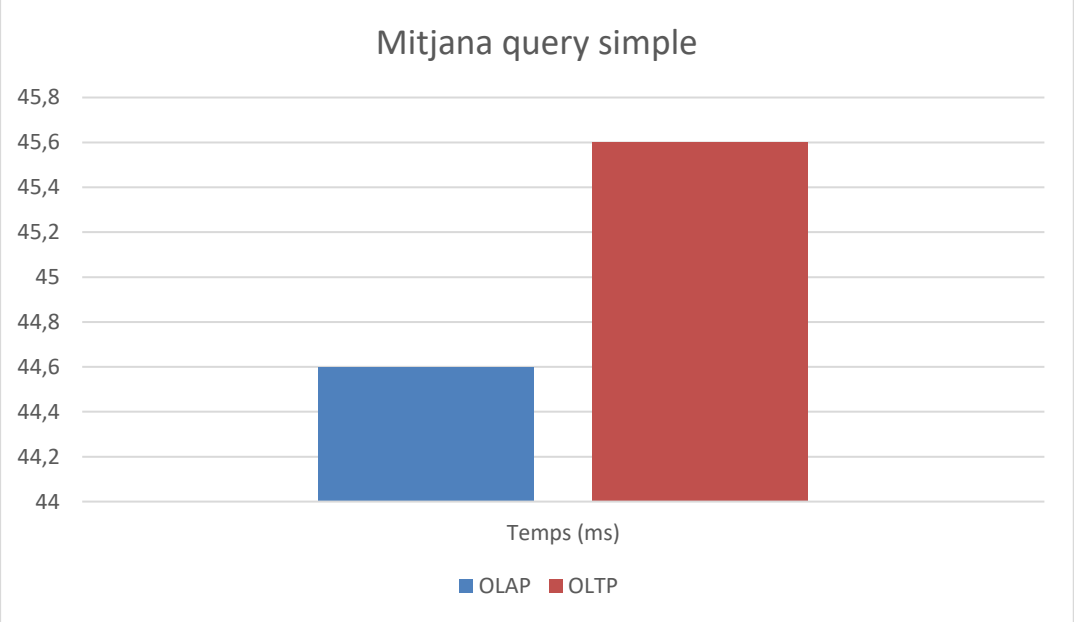
```
SELECT * FROM Comprovacio;
```

	nom	oltp_num_files	olap_num_files
1	Circuits	74	74
2	Laps	472504	472504
3	PitStops	7436	7436
4	Results	24620	24620
5	Status	135	135

9. COMPARACIÓ OLTP VS OLAP

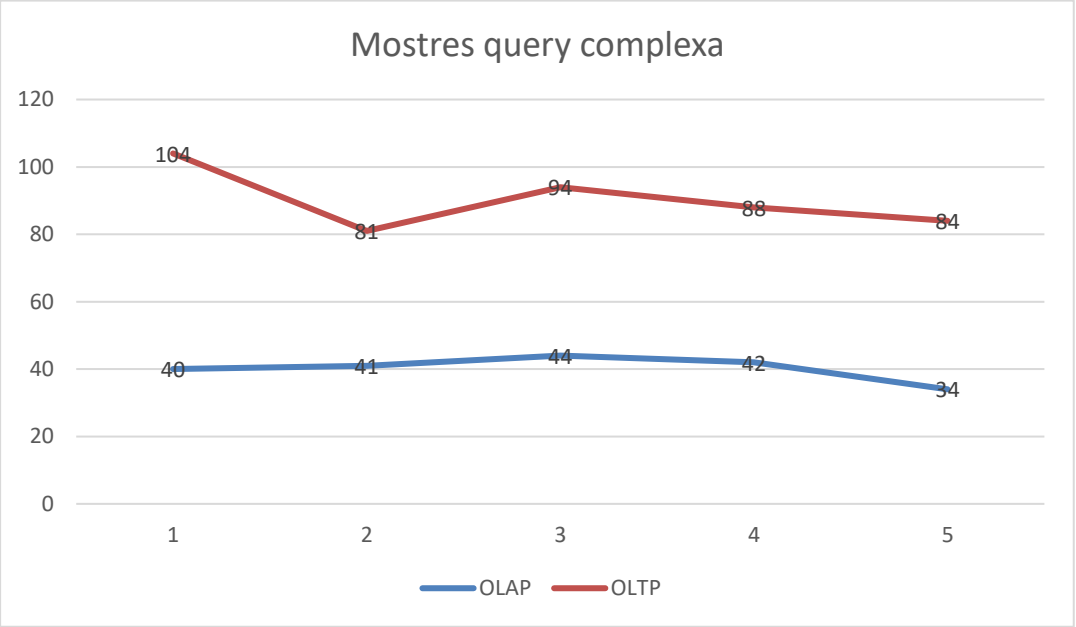
- Query simple que involucri només una taula

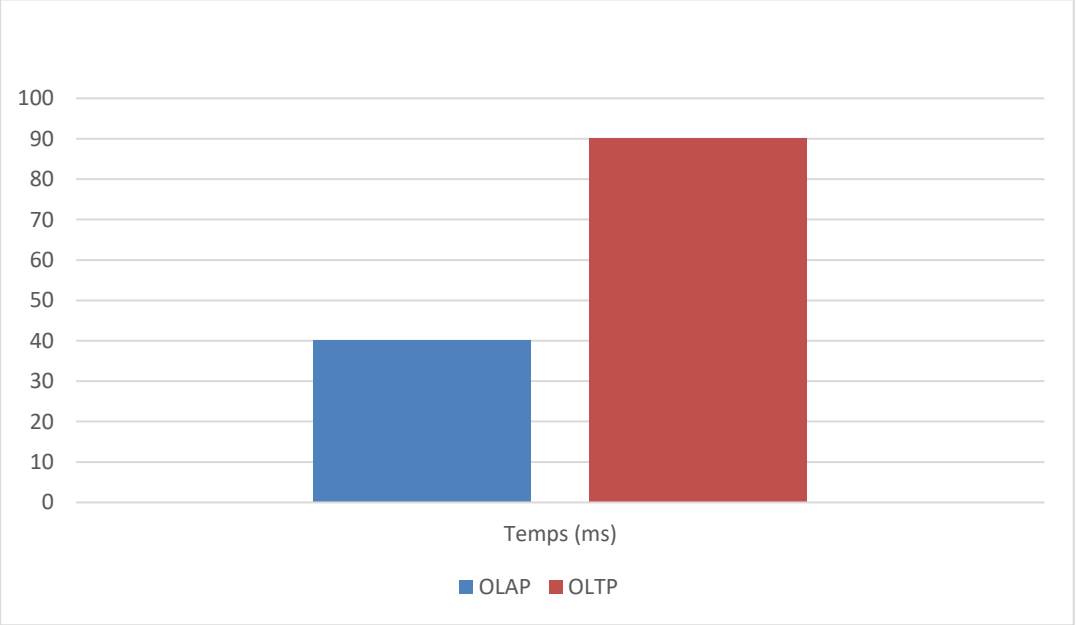




A la query simple podem veure que tant a la OLAP i la OLTP son pràcticament iguals, tot i que la OLAP té una mitjana lleugerament mes baixa no creiem que sigui significatiu a l'hora de fer una query simple.

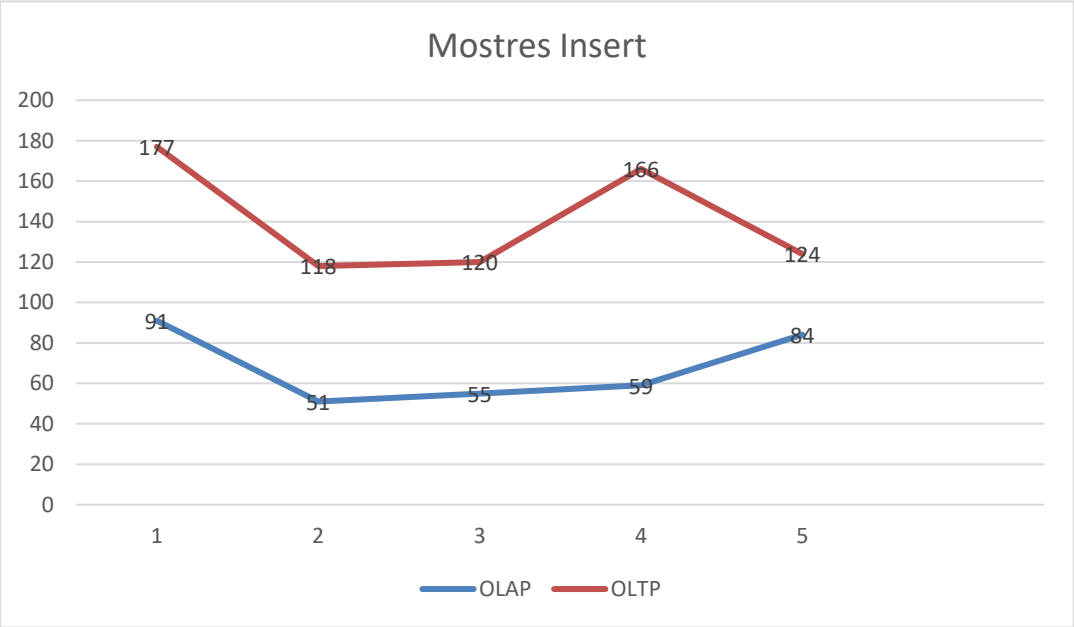
- Query complexa que involucri 5 taules

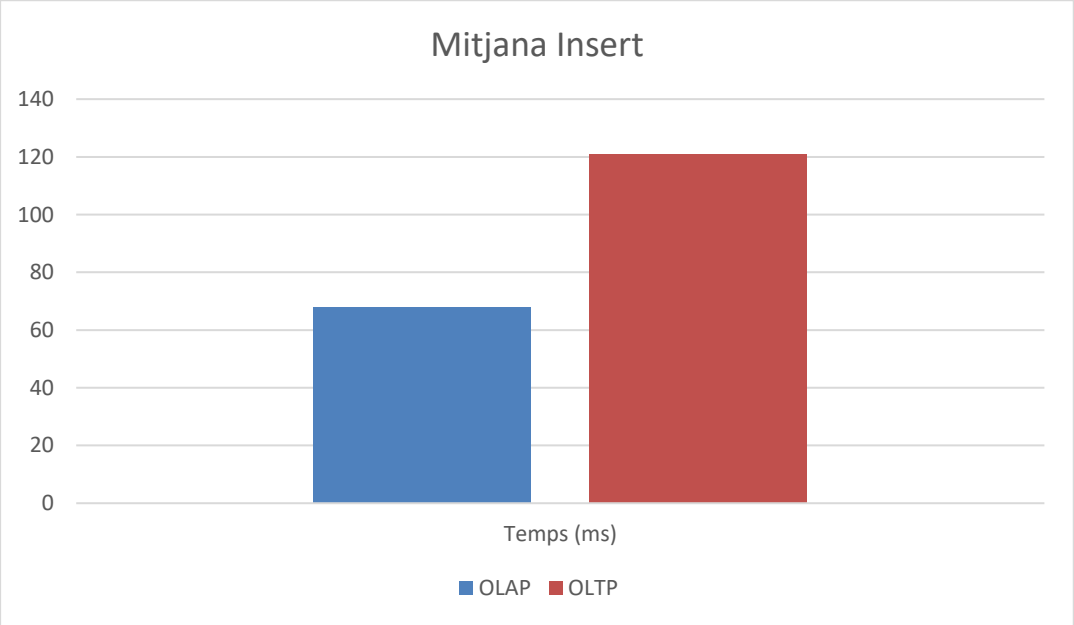




Quan fem una query complexa clarament la mitjana de temps de molt més baixa a la OLAP i en cap experiment la OLTP ha estat més ràpida. Hem arribat a la conclusió de que fer una query complexa a la OLAP es molt més ràpid.

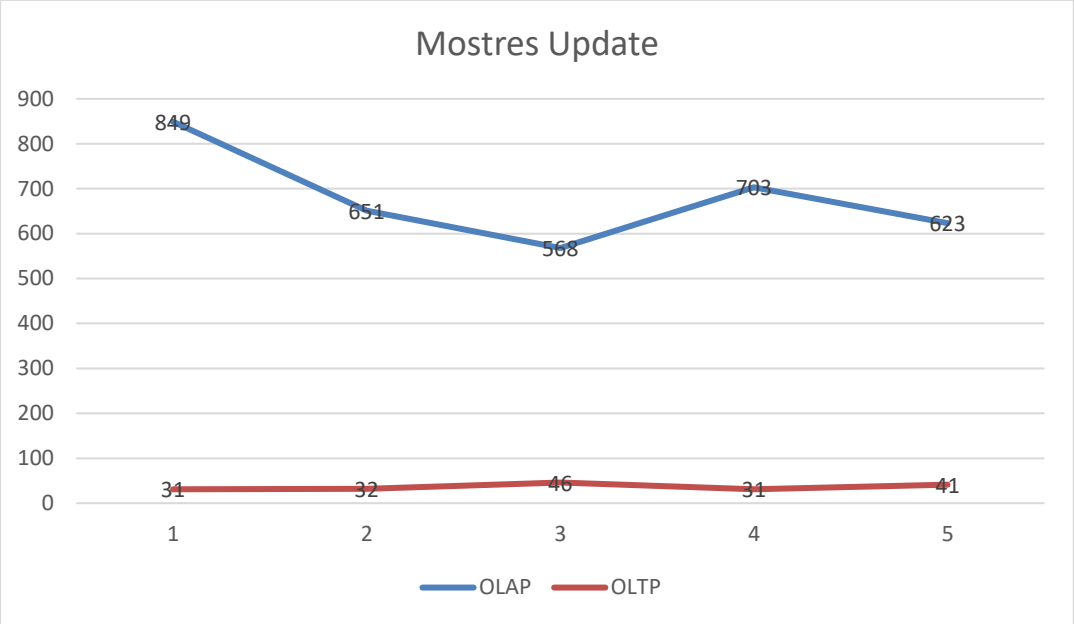
- Insert a una taula

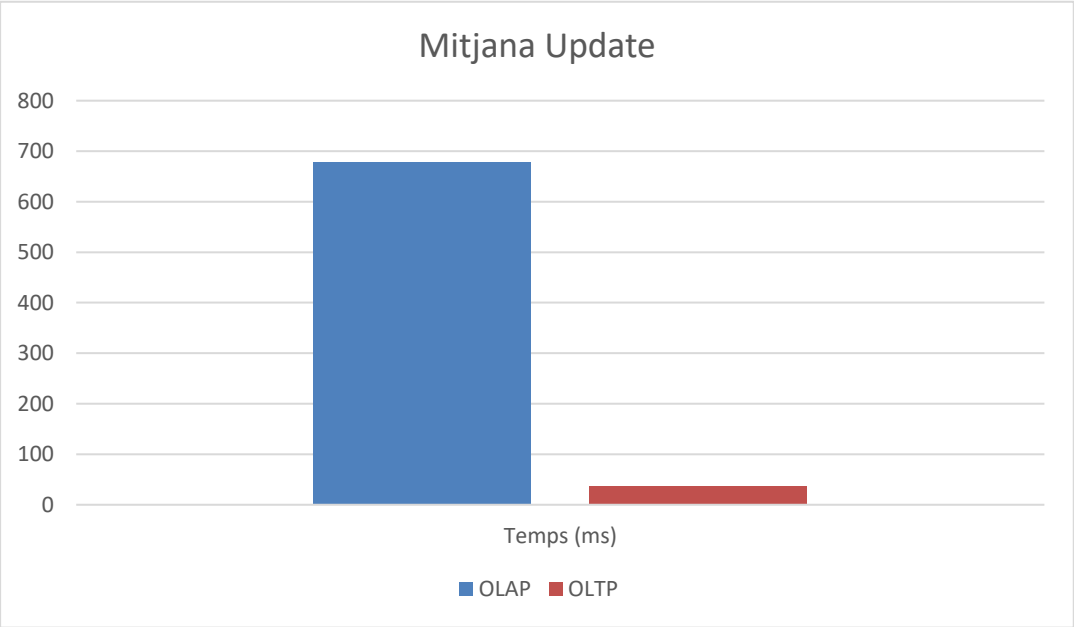




Quan fem un insert clarament la mitjana de temps de molt més baixa a la OLAP(el doble de ràpid) i en cap experiment la OLTP ha estat més ràpida.

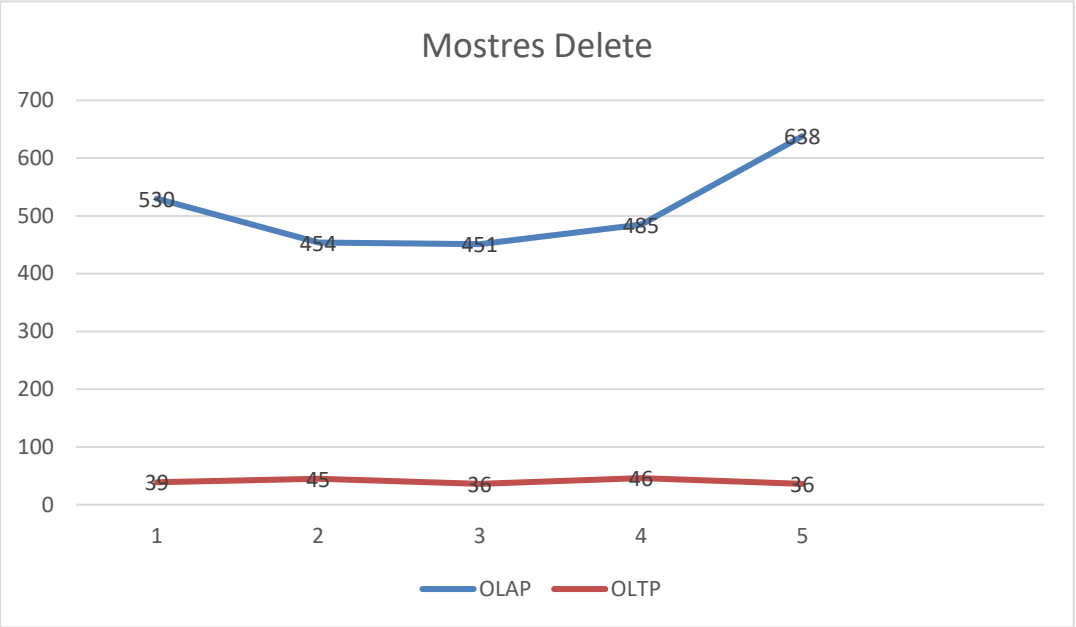
- Update a un camp

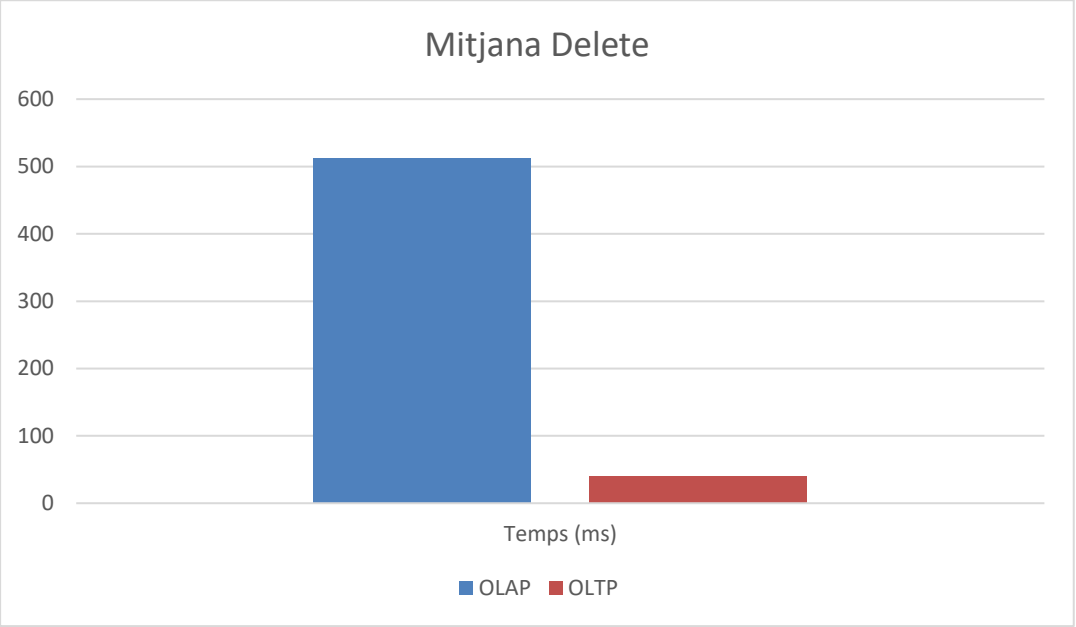




Quan fem un update la mitjana de temps de molt més alta a la OLAP (tarda més) i en cap experiment la OLTP ha estat més lenta. Clarament hem vist que es molt més ràpid fer un update a la OLTP.

- Delete a una taula





Quan fem un delete la mitjana de temps de molt més alta a la OLAP (tarda més) i en cap experiment la OLTP ha estat més lenta. Clarament hem vist que es molt més ràpid fer un delete a la OLTP.

10. QUERYS

- Query 1

Troba els Status que mai s'han utilitzat.

```
SELECT status
FROM status AS s LEFT JOIN results AS r ON s.statusId = r.statusId
WHERE r.statusId IS NULL;
```

	status
1	+49 Laps
2	+38 Laps

- Query 2

Troba la nacionalitat i la mitjana del temps d'aquells corredors que la seva escuderia tingui la mitjana mes baixa de temps en parada en boxes

```
SELECT r.surename, r.forename, r.nationality, AVG(p.milliseconds)
FROM pitStops as p LEFT JOIN results as r ON (p.driverId = r.driverId AND p.raceId = r.raceId)
WHERE r.constructorId IN (
    SELECT r1.constructorId
    FROM pitStops as p1 LEFT JOIN results as r1 ON (p1.driverId = r1.driverId AND p1.raceId = r1.raceId)
    GROUP BY r1.constructorId
    HAVING AVG(p1.milliseconds) <= ALL
        (SELECT AVG(p2.milliseconds) FROM pitStops as p2 LEFT JOIN results as r2 ON (p2.driverId = r2.driverId AND p2.raceId = r2.raceId)
        GROUP BY r2.constructorId
        ORDER BY AVG(p2.milliseconds)))
GROUP BY r.driverId;
```

	surename	forename	nationality	`AVG(p.milliseconds)`
1	d'Ambrosio	Jérôme	Belgian	24908.6829
2	Glock	Timo	German	23470.3056

- Query 3

Busca els corredors que han millorat el temps de la seva qualificació per cada ronda consecutiva i que la seva volta mes ràpida sigui mes ràpida que qualsevol temps de qualificació de altres pilots d'aquella carrera. També comprova que hagi quedat entre els tres primers.

```
SELECT r.forename, r.surename, r.fastestLapTime
FROM results as r
WHERE r.q1 > r.q2 AND r.q2 > r.q3 AND r.position > 0 AND r.position < 4 and  r.fastestLapTime < ALL (
    SELECT r1.q1
    FROM results as r1
    WHERE r.raceId = r1.raceId AND r.driverId <> r1.driverId AND r1.q1 IS NOT NULL)
AND r.fastestLapTime < ALL (
    SELECT r2.q2
    FROM results as r2
    WHERE r.raceId = r2.raceId AND r.driverId <> r2.driverId AND r2.q2 IS NOT NULL)
AND r.fastestLapTime < ALL (
    SELECT r3.q3
    FROM results as r3
    WHERE r.raceId = r3.raceId AND r.driverId <> r3.driverId AND r3.q3 IS NOT NULL);
```

	forename	surename	fastestLapTime
1	Mark	Webber	1:13.733
2	Robert	Kubica	1:14.155
3	Sebastian	Vettel	1:14.283
4	Mark	Webber	1:14.047
5	Lewis	Hamilton	1:41.196
6	Nico	Rosberg	1:40.402
7	Fernando	Alonso	1:42.081
8	Daniel	Ricciardo	1:52.974
9	Nico	Rosberg	1:50.511
10	Valtteri	Bottas	1:52.716

- Query 4

Busca els corredors, la velocitat mes ràpida, la volta mes ràpida i el circuit a on els corredors hagin fet la volta mes ràpida però no hagin marcat la velocitat més alta o al contrari.

```
(SELECT DISTINCT r.forename,r.surename,r.fastestLapSpeed,r.fastestLapTime,c.circuitName
FROM results as r JOIN circuits as c ON (r.circuitId = c.circuitId) JOIN laps as l ON (l.driverId = r.driverId AND l.raceId = r.raceId)
WHERE r.fastestLapSpeed IS NOT NULL AND r.fastestLapTime IS NOT NULL
AND l.milliseconds <= ALL (
    SELECT l1.milliseconds
    FROM laps as l1
    WHERE r.raceId = l1.raceId AND l1.milliseconds IS NOT NULL)
AND r.fastestLapSpeed < ANY (
    SELECT r2.fastestLapSpeed
    FROM results as r2
    WHERE r.raceId = r2.raceId AND r2.fastestLapSpeed IS NOT NULL))
UNION
(SELECT DISTINCT r.forename,r.surename,r.fastestLapSpeed,r.fastestLapTime,c.circuitName
FROM results as r JOIN circuits as c ON (r.circuitId = c.circuitId) JOIN laps as l ON (l.driverId = r.driverId AND l.raceId = r.raceId)
WHERE r.fastestLapSpeed IS NOT NULL AND r.fastestLapTime IS NOT NULL
AND l.milliseconds > ANY (
    SELECT l1.milliseconds
    FROM laps as l1
    WHERE r.raceId = l1.raceId AND l1.milliseconds IS NOT NULL)
AND r.fastestLapSpeed >= ALL (
    SELECT r2.fastestLapSpeed
    FROM results as r2
    WHERE r.raceId = r2.raceId AND r2.fastestLapSpeed IS NOT NULL));
```

310 rows

	forename	surename	fastestLapSpeed	fastestLapTime	circuitName
1	Sebastian	Vettel	159.910	1:15.192	Circuit de Monaco
2	Kamui	Kobayashi	222.120	1:35.478	Silverstone Circuit
3	Valtteri	Bottas	217.484	1:37.513	Silverstone Circuit
4	Felipe	Massa	199.592	1:32.853	Nürburgring
5	Sebastian	Vettel	200.860	1:38.809	Circuit of the Americas
6	Heikki	Kovalainen	218.385	1:27.418	Albert Park Grand Prix Circuit
7	Kimi	Räikkönen	223.978	1:25.235	Albert Park Grand Prix Circuit
8	Kimi	Räikkönen	221.869	1:26.045	Albert Park Grand Prix Circuit
9	Fernando	Alonso	222.807	1:25.683	Albert Park Grand Prix Circuit
10	Michael	Schumacher	226.933	1:24.125	Albert Park Grand Prix Circuit
11	Nico	Rosberg	217.668	1:27.706	Albert Park Grand Prix Circuit
12	Mark	Webber	216.061	1:28.358	Albert Park Grand Prix Circuit
13	Felipe	Massa	214.631	1:28.947	Albert Park Grand Prix Circuit
14	Jenson	Button	214.053	1:29.187	Albert Park Grand Prix Circuit
15	Kimi	Räikkönen	213.845	1:29.274	Albert Park Grand Prix Circuit
16	Nico	Rosberg	206.436	1:32.478	Albert Park Grand Prix Circuit
17	Lewis	Hamilton	209.915	1:30.945	Albert Park Grand Prix Circuit
18	Daniel	Ricciardo	214.510	1:28.997	Albert Park Grand Prix Circuit
19	Kimi	Räikkönen	220.605	1:26.538	Albert Park Grand Prix Circuit
20	Daniel	Ricciardo	222.128	1:25.945	Albert Park Grand Prix Circuit
21	Valtteri	Bottas	223.075	1:25.580	Albert Park Grand Prix Circuit
22	Nick	Heidfeld	209.244	1:35.366	Sepang International Circuit
23	Lewis	Hamilton	206.355	1:36.701	Sepang International Circuit
24	Fernando	Alonso	210.487	1:34.803	Sepang International Circuit
25	Kimi	Räikkönen	208.987	1:35.483	Sepang International Circuit

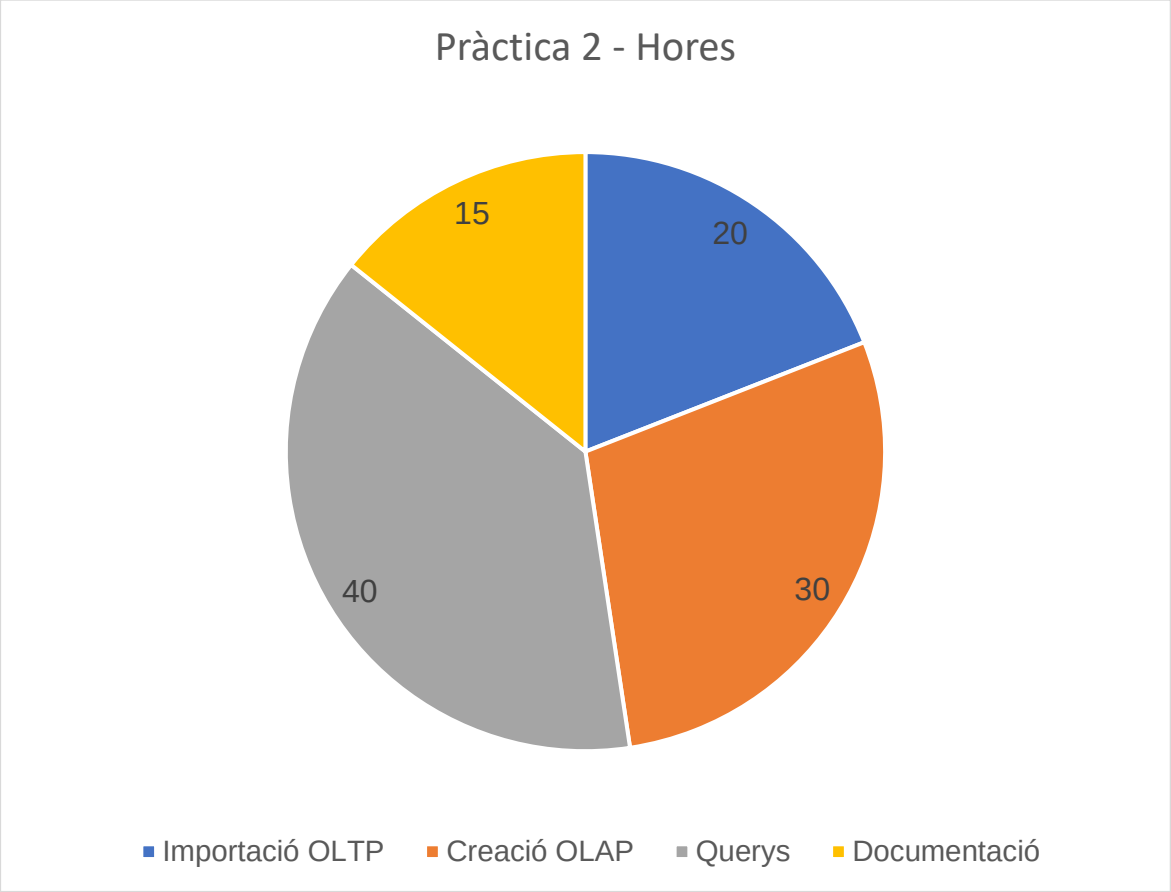
- Query 5

Busca l'adelantament mes gran de la historia de la F1.

```
(SELECT r.forename, r.surename, c.circuitName, r.year, l.position - r.position AS Overtaking
FROM Results AS r, Circuits AS c, Laps AS l
WHERE r.raceId = l.raceId AND r.driverId = l.driverId AND c.circuitId = r.circuitId
  AND l.lap = 2 AND r.position IS NOT NULL AND l.position IS NOT NULL
ORDER BY Overtaking DESC
LIMIT 1)
UNION
(SELECT r.forename, r.surename, c.circuitName, r.year, l1.position - l2.position AS Overtaking
FROM Results AS r, Circuits AS c, Laps AS l1, Laps l2
WHERE r.raceId = l1.raceId AND r.driverId = l1.driverId AND r.raceId = l2.raceId AND r.driverId = l2.driverId AND c.circuitId = r.circuitId
  AND l2.lap = l1.lap + 1 AND r.position IS NOT NULL AND l1.position IS NOT NULL
ORDER BY Overtaking DESC
LIMIT 1);
```

	forename	surename	circuitName	year	Overtaking
1	Sergio	Pérez	Sepang International Circuit	2012	21
2	David	Coulthard	Silverstone Circuit	2003	13

11. TEMPS DEDICAT



12. CONCLUSIONS

Aquesta pràctica posa a prova la capacitat des alumnes per aplicar els conceptes i la teoria explicades tant a classe com a les sessions de laboratori en un cas pràctic com aquest relacionat amb el tractament de les dades de la Formula 1.

En l'apartat d'importació de la OLTP remota no hem tingut cap problema tret de dedicar una estona a trobar-la en el puigpedros.

En l'apartat de creació de la OLAP si que hem tingut problemes ja que al principi no hem entès els conceptes de OLTP i OLAP així que hem perdut molt temps normalitzant el model ja donat de la OLTP quan el que havíem de fer era "simplificar" el model.

I finalment l'apartat de les querys també ens hem entretingut molt però dedicant temps ha acabat sortint,

Estem contents amb el resultat d'aquesta pràctica i creiem que ens ha servir per reforçar els coneixements de classe.

13. BIBLIOGRAFIA

https://stackoverflow.com/

https://www.tutorialspoint.com/

https://www.w3schools.com/

https://dev.mysql.com/