



**UNIVERSIDAD CATÓLICA DEL URUGUAY**  
**Dámaso Antonio Larrañaga**

**Ingeniería Informática**

**Proyecto obligatorio**

**Asignatura: Bases de Datos 1**

**Docente: Bernardo Rychtenberg**

**Juan Tabarez, Sebastián Quintana, Amanda Seara**

**03/07/2023**

## Tabla de Contenido

Resumen ejecutivo .....	3
Introducción .....	3
Desarrollo.....	3
Referencias.....	15

## Resumen ejecutivo

Este informe refleja el desarrollo de un proyecto de gestión de una base de datos de la Fórmula 1. Se crearon las tablas en una base de datos MySQL, se insertaron los registros proporcionados y se realizaron consultas para obtener información sobre la Fórmula 1.

## Introducción

En este informe se presentará el desarrollo de un proyecto de gestión de una base de datos de la Fórmula 1. En este proyecto, se llevó a cabo el estudio de las tablas proporcionadas por Kaggle, comprendiendo como son las relaciones existentes entre ellas y determinando las claves primarias y claves foráneas que permiten la consistencia de la base de datos. Realizado este paso se realizó la creación de las tablas en una base de datos MySQL y se insertaron todos los registros dados en la base de datos Kaggle. Una vez se han insertado todos los datos en la base de datos se procedió a obtener cierta información realizando determinadas consultas específicas con el fin de obtener información sobre la Formula 1.

«La Fórmula 1 es la categoría máxima del automovilismo deportivo a nivel mundial, donde pilotos y equipos compiten en un apasionante campeonato que abarca múltiples circuitos alrededor del mundo. La gestión eficiente de la información relacionada con este deporte es fundamental para el desarrollo de estrategias, análisis de desempeño y toma de decisiones por parte de los equipos y aficionados.» (OpenAI, 2023).

A lo largo de este informe, se detallarán los pasos seguidos para el estudio de las tablas proporcionadas, la creación de la base de datos, la inserción de los registros y la realización de las consultas.

repositorio: <https://github.com/Seba-Quintana/F1>

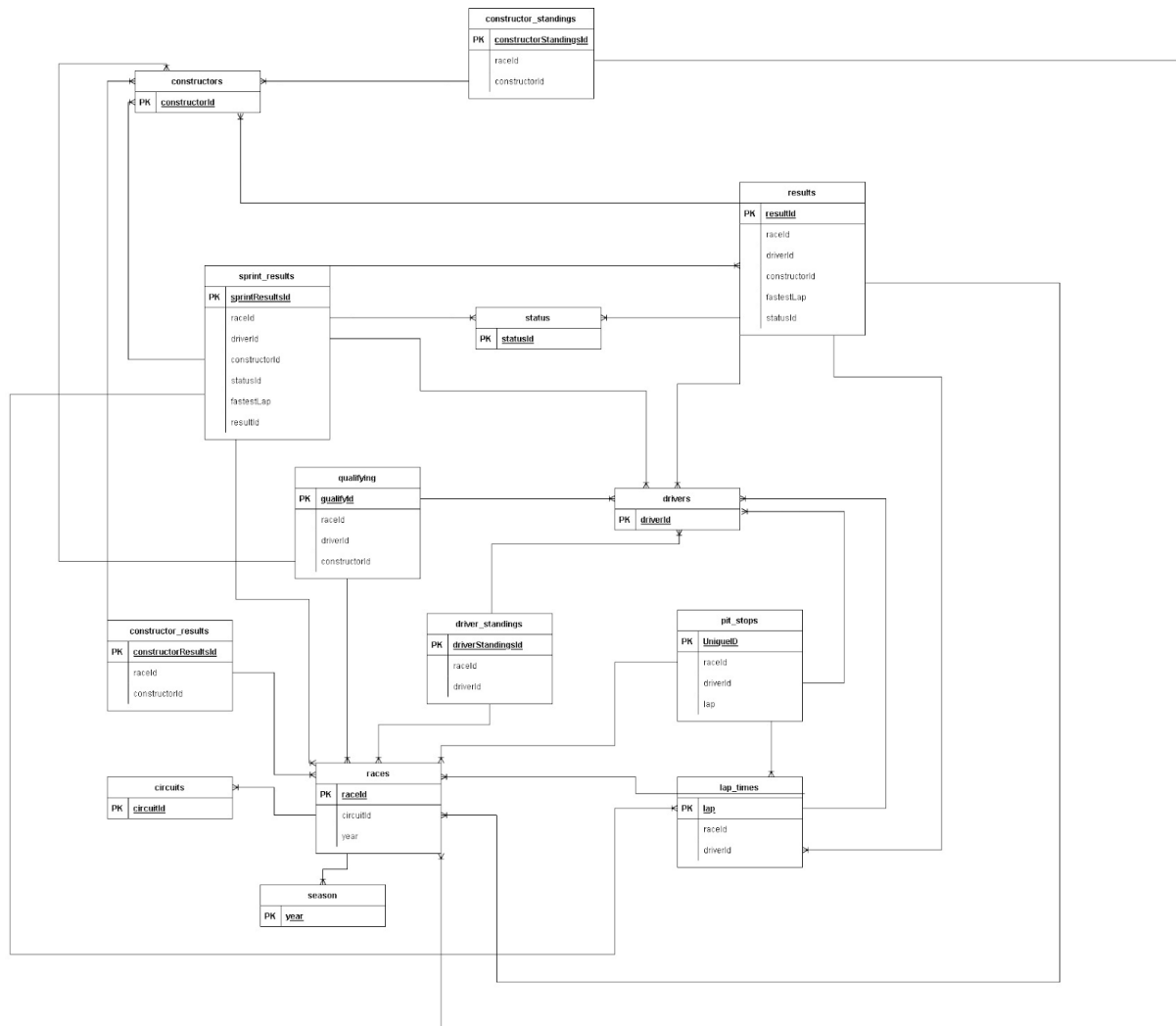
## Desarrollo

### Modelo Entidad relación (MER)

Se comenzó realizando un Modelo Entidad Relación de los datos proporcionados, definiendo las claves primarias, foráneas y las relaciones entre las tablas.

Esto facilitó el problema a la hora de la creación de las tablas en la base de datos, debido a que se pudo visualizar más fácilmente que tablas se debieron crear primero y como estaban definidas.

Cuadro N°1: MER



Fuente: Elaboración propia.

## Creación de tablas e inserción

Para insertar los datos lo primero que se debe hacer es instalar el driver para realizar la conexión con la base de datos (en nuestro caso jdbc), y conectarlo al código. El método DBConn de la clase DBConnection es el que se encarga de realizar esta conexión entre la base de datos y el programa por medio del driver. Luego se invoca al otro método de la clase llamado setup, el cual crea el Schema F1.

Una vez hecha la conexión, se guardan los archivos csv en una carpeta “archivos” junto con el código, desde el cual se lee la carpeta y se extraen todos los archivos que se encuentran dentro de ella.

La creación de las tablas las realizamos a mano, dado que un programa que cree las tablas a partir de los datos que contienen es costoso en términos de tiempo, ya que para inferir el tipo de datos de una tabla, se debe recorrer las filas de la tabla hasta encontrar un dato con el cual poder inferir el tipo de la columna, y repetir el procedimiento para todas las columnas, y lo mismo con todas las tablas. Por esta razón decidimos crear las tablas con el método createTables, el cual se encarga de crearlas por código. Esta creación se realiza en un orden específico para que no haya problema con las claves foráneas.

Por último, se insertan los valores en la base de datos, con el método insertValues. Este método sanitiza los nombres de las tablas, de las columnas, e incluso de los datos, para que los datos que se encuentran en la carpeta archivos no den problema. Es un método genérico, por lo que se podrían agregar más archivos, aunque como la sanitización está orientada a los datos proporcionados, hay que tener en cuenta que algún dato en particular puede no estar contemplado en la misma, lo que puede conllevar a errores.

## Consultas

1) Para averiguar quién es la persona que gana un campeonato mundial, es necesario saber cuál fue el corredor que obtuvo más puntos durante una temporada, y para saber quién gana más campeonatos mundiales se debe hacer lo mismo para todas las temporadas y ver quien tiene más.

Para resolver esta query fue necesario separarla en distintas partes, y resultó tan compleja que se decidió realizar una sección del informe explicando la solución.

Primero se pensó en encontrar el nombre del piloto, pero como la tabla del piloto no tiene datos acerca de los puntos ni de las carreras, no se podía utilizar el dato sin hacer joins, por lo que se decidió cambiar el enfoque; el nombre del piloto sería el paso final, y se empezó por la cantidad de puntos que tuvieron los pilotos por año.

De ahí surgió esta primera consulta:

```
SELECT results.driverId, races.year, SUM(results.points) AS Puntaje
FROM races
```

```
JOIN results ON races.raceId = results.raceId
GROUP BY results.driverId, races.year
```

Una vez se consiguen los puntajes que tuvo cada piloto en cada año, hay que evaluar quien gano en cada uno de esos años. Por eso, hay que hacerle un MAX a cada uno de esos puntajes. Para esto hay que conseguir los puntajes, y luego aplicarle un MAX, lo que fue resuelto con una subquery.

La consulta con el MAX seria:

```
SELECT nombreMax.year, MAX(nombreMax.Puntaje) AS MaxPuntaje
FROM (
    # subquery con el mismo puntaje de antes
    SELECT results.driverId, races.year, SUM(results.points) AS Puntaje
    FROM races
    JOIN results ON races.raceId = results.raceId
    GROUP BY results.driverId, races.year
) AS nombreMax
```

Con el JOIN con Puntajes (nombrePuntaje) queda asi:

```
SELECT results.driverId, races.year, SUM(results.points) AS Puntaje
FROM races
JOIN results ON races.raceId = results.raceId
GROUP BY results.driverId, races.year
```

# la linea siguiente cierra un join anterior, que sera explicado posteriormente

```
) AS nombrePuntaje ON drivers.driverId = nombrePuntaje.driverId
JOIN (
```

# puntaje del piloto que hizo mas puntos por año

```
SELECT nombreMax.year, MAX(nombreMax.Puntaje) AS MaxPuntaje
FROM (
```

# el mismo puntaje de antes, esta en el from porque saca los valores del select

# de la tabla resultante de la subquery

```
SELECT results.driverId, races.year, SUM(results.points) AS Puntaje
FROM races
JOIN results ON races.raceId = results.raceId
GROUP BY results.driverId, races.year
```

```
) AS nombreMax
```

Luego se agrupan por año los puntajes agregando:

```
GROUP BY nombreMax.year
```

Una vez agrupados, devuelve el máximo puntaje que hubo en cada uno de los años. Ahora se debe saber quién fue el que gana esos campeonatos (consultar por el nombre del piloto), y sumar la cantidad de veces que aparece en el resultado, lo que da cuantas veces tuvo el máximo puntaje (es decir, cuantas veces gana un campeonato).

Para hacer esto se agrega al principio:

```
SELECT drivers.forename, drivers.surname, COUNT(*) AS CantCampeonatos  
FROM drivers
```

A esto se le hace un JOIN con las otras consultas anteriores, y se evalúan las condiciones para salir campeón; es decir que hay que ver que el campeonato haya terminado (2023 no tiene ganador), y de los puntajes de todos los años tomar los mayores (para tomar a los campeones).

El resultado de esto es:

```
SELECT drivers.forename, drivers.surname, COUNT(*) AS CantCampeonatos  
FROM drivers
```

```
JOIN (
```

# puntaje total de un piloto por año

```
SELECT results.driverId, races.year, SUM(results.points) AS Puntaje  
FROM races  
JOIN results ON races.raceId = results.raceId  
GROUP BY results.driverId, races.year
```

```
) AS nombrePuntaje ON drivers.driverId = nombrePuntaje.driverId
```

```
JOIN (
```

# puntaje del piloto que hizo mas puntos por año

```
SELECT nombreMax.year, MAX(nombreMax.Puntaje) AS MaxPuntaje  
FROM (
```

```

# el mismo puntaje de antes

SELECT results.driverId, races.year, SUM(results.points) AS Puntaje
FROM races
JOIN results ON races.raceId = results.raceId
GROUP BY results.driverId, races.year
) AS nombreMax
GROUP BY nombreMax.year
) AS nombreMaxAnio ON nombreMaxAnio.year = nombrePuntaje.year
WHERE nombreMaxAnio.year < 2023
AND nombrePuntaje.Puntaje = nombreMaxAnio.MaxPuntaje
GROUP BY drivers.forename, drivers.surname

```

Notar que también se debe agrupar por piloto. Esta consulta da como resultado cuantos campeonatos ganó cada piloto. Pero lo que se quiere saber es quién fue el que gana más. Por ende, se le debe agregar un COUNT, para que cuente cuantas tiene cada uno, y un MAX para que de esas tome solo la más grande.

Para esto se agrega un HAVING COUNT(\*), y para calcular el MAX se hace una subquery en el having: SELECT MAX(T.CantCampeonatos) FROM, donde en el FROM va el resultado de la consulta anterior, para poder aplicarle MAX (notar que cuando una subquery está en el FROM, en el SELECT se puede utilizar la tabla resultante de la subquery, por lo que se toma la consulta previa y se ubica en la subquery para poder aplicarle MAX).

Es decir, lo último que se debe hacer es copiar y pegar la consulta de arriba en el FROM, y un ORDER BY para dejar a Schumacher primero. Esto deja como resultado final:



```

SELECT drivers.forename, drivers.surname, COUNT(*) AS CantCampeonatos
FROM drivers
JOIN (
    # puntaje total de un piloto por año
    SELECT results.driverId, races.year, SUM(results.points) AS Puntaje
    FROM races
    JOIN results ON races.raceId = results.raceId
    GROUP BY results.driverId, races.year
) AS nombrePuntaje ON drivers.driverId = nombrePuntaje.driverId
JOIN (
    # puntaje del piloto que hizo mas puntos por año
    SELECT nombreMax.year, MAX(nombreMax.Puntaje) AS MaxPuntaje
    FROM (
        # el mismo puntaje de antes
        SELECT results.driverId, races.year, SUM(results.points) AS Puntaje
        FROM races
        JOIN results ON races.raceId = results.raceId
        GROUP BY results.driverId, races.year
    ) AS nombreMax
    GROUP BY nombreMax.year
) AS nombreMaxAnio ON nombreMaxAnio.year = nombrePuntaje.year
WHERE nombreMaxAnio.year < 2023
AND nombrePuntaje.Puntaje = nombreMaxAnio.MaxPuntaje
GROUP BY drivers.forename, drivers.surname
HAVING COUNT(*) = (
    SELECT MAX(T.CantCampeonatos)
    FROM (
        SELECT drivers.forename, drivers.surname, COUNT(*) AS CantCampeonatos
        FROM drivers
        JOIN (
            SELECT results.driverId, races.year, SUM(results.points) AS Puntaje
            FROM races
            JOIN results ON races.raceId = results.raceId
            GROUP BY results.driverId, races.year
        ) AS nombrePuntaje ON drivers.driverId = nombrePuntaje.driverId
    ) AS nombreMax
    GROUP BY nombreMax.year
) AS nombreMaxAnio ON nombreMaxAnio.year = nombrePuntaje.year
WHERE nombreMaxAnio.year < 2023
AND nombrePuntaje.Puntaje = nombreMaxAnio.MaxPuntaje
GROUP BY drivers.forename, drivers.surname
) AS T)
ORDER BY CantCampeonatos DESC;

```

Resultado obtenido:

	forename	surname	CantCampeonatos
1	Michael	Schumacher	7
2	Lewis	Hamilton	7

2) Para saber qué escudería ha ganado el campeonato de constructores más veces en la historia se realiza la siguiente consulta:

```
# nombres de los pilotos y la cant de campeonatos
SELECT constructors.name, COUNT(*) AS CantCampeonatos
FROM constructors
JOIN (
    # puntaje total de un piloto por año
    SELECT constructor_results.constructorId, races.year, SUM(constructor_results.points) AS Puntaje
    FROM races
    JOIN constructor_results ON races.raceId = constructor_results.raceId
    GROUP BY constructor_results.constructorId, races.year
) AS nombrePuntaje ON constructors.constructorId = nombrePuntaje.constructorId
JOIN (
    # puntaje del piloto que hizo mas puntos por año
    SELECT nombreMax.year, MAX(nombreMax.Puntaje) AS MaxPuntaje
    FROM (
        # el mismo puntaje de antes
        SELECT constructor_results.constructorId, races.year, SUM(constructor_results.points) AS Puntaje
        FROM races
        JOIN constructor_results ON races.raceId = constructor_results.raceId
        GROUP BY constructor_results.constructorId, races.year
    ) AS nombreMax
    GROUP BY nombreMax.year
) AS nombreMaxAnio ON nombreMaxAnio.year = nombrePuntaje.year
# 2023 no tiene campeon todavia
WHERE nombreMaxAnio.year < 2023
# se toma el puntaje solamente del que gano la temporada
AND nombrePuntaje.Puntaje = nombreMaxAnio.MaxPuntaje
GROUP BY constructors.name
# cuento cuantos campeonatos ganaron los que ganaron campeonatos
HAVING COUNT(*) = (
    # (de los que ganaron campeonatos devolver el max)
    SELECT MAX(T.CantCampeonatos)
    FROM (
        SELECT constructors.name, COUNT(*) AS CantCampeonatos
        FROM constructors
        JOIN (
            SELECT constructor_results.constructorId, races.year, SUM(constructor_results.points) AS Puntaje
            FROM races
            JOIN constructor_results ON races.raceId = constructor_results.raceId
            GROUP BY constructor_results.constructorId, races.year
        ) AS nombrePuntaje ON constructors.constructorId = nombrePuntaje.constructorId
    ) AS nombreMax
    GROUP BY nombreMax.year
) AS nombreMaxAnio ON nombreMaxAnio.year = nombrePuntaje.year
WHERE nombreMaxAnio.year < 2023
AND nombrePuntaje.Puntaje = nombreMaxAnio.MaxPuntaje
GROUP BY constructors.name
) AS T)
ORDER BY CantCampeonatos DESC;
```

Se realiza igual que la pregunta 1 pero con constructors.

Resultado obtenido:

	name	CantCampeonatos
1	Ferrari	15

3) Para saber qué piloto ganó más veces un Gran Premio se realiza la siguiente consulta:

```
SELECT drivers.forename, drivers.surname, COUNT(results.`position`)
AS carreras_ganadas
FROM drivers
JOIN results ON drivers.driverId = results.driverId
WHERE results.`position` = 1
GROUP BY drivers.forename, drivers.surname
ORDER BY carreras_ganadas DESC;
```

Resultado obtenido:

	forename	surname	carreras_ganadas
1	Lewis	Hamilton	103
2	Michael	Schumacher	91
3	Sebastian	Vettel	53
4	Alain	Prost	51
5	Ayrton	Senna	41
6	Max	Verstappen	35
7	Fernando	Alonso	32
8	Nigel	Mansell	31
9	Jackie	Stewart	27

4) Para saber que grandes premios tuvieron los campeonatos desde el 1996 – 1999 se realiza la siguiente consulta:

```
SELECT DISTINCT races.name
FROM races
WHERE races.`year`
BETWEEN 1996 AND 1999;
```

Resultado obtenido:

	asc name
1	Australian Grand Prix
2	Brazilian Grand Prix
3	Argentine Grand Prix
4	European Grand Prix
5	San Marino Grand Prix
6	Monaco Grand Prix
7	Spanish Grand Prix
8	Canadian Grand Prix
9	French Grand Prix
10	British Grand Prix
11	German Grand Prix
12	Hungarian Grand Prix
13	Belgian Grand Prix
14	Italian Grand Prix
15	Portuguese Grand Prix
16	Japanese Grand Prix
17	Austrian Grand Prix
18	Luxembourg Grand Prix
19	Malaysian Grand Prix

5) Para saber que quién ganó el Gran Premio de Suzuka, de 1997 se realiza la siguiente consulta:

```
SELECT drivers.forename, drivers.surname
FROM drivers
JOIN results ON drivers.driverId = results.driverId
JOIN races ON results.raceId = races.raceId
WHERE races.`year` = 1997
AND name = 'Japanese Grand Prix'
AND results.`position` = 1;
```

Resultado obtenido:

	asc forename	asc surname
1	Michael	Schumacher

6) Para saber cuántas carreras ganó Jacques Villeneuve se realiza la siguiente consulta:

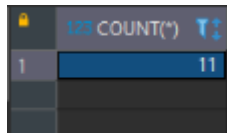
```
SELECT COUNT(*)
```

```

FROM races
JOIN results ON results.raceId = races.raceId
JOIN drivers ON results.driverId = drivers.driverId
AND drivers.forename = 'Jacques'
WHERE drivers.surname = 'Villeneuve'
AND results.`position` = 1;

```

Resultado obtenido:



	123 COUNT(*)
1	11

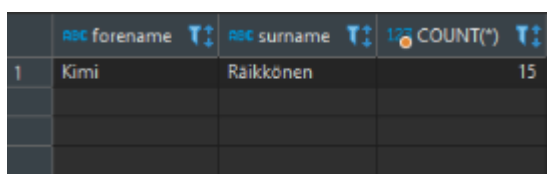
7) Para saber quién ganó más carreras saliendo desde una posición que no era la “Pole Position” se realiza la siguiente consulta:

```

SELECT drivers.forename, drivers.surname, COUNT(*)
FROM drivers
JOIN results ON drivers.driverId = results.driverId
WHERE results.grid <> 1
AND results.`position` = 1
GROUP BY drivers.forename, drivers.surname LIMIT 1;

```

Resultado obtenido:



	asc forename	asc surname	123 COUNT(*)
1	Kimi	Räikkönen	15

8) Para saber quién ganó más carreras saliendo desde la posición “Pole Position” se realiza la siguiente consulta:

```

SELECT drivers.forename, drivers.surname, COUNT(*)
FROM drivers
JOIN results ON drivers.driverId = results.driverId
WHERE results.grid = 1

```

```
AND results.`position` = 1  
GROUP BY drivers.forename, drivers.surname LIMIT 1;
```

Resultado obtenido:

	asc forename	asc surname	123 COUNT(*)
1	Lewis	Hamilton	61

## Conclusiones

Este proyecto fue muy interesante y educativo, ya que le aportó al equipo más conocimiento acerca de cómo utilizar las bases de datos de una forma más práctica. Principalmente con el código se aprendió a conectar una base de datos MySQL con Java, y realizar operaciones sobre la misma, lo que resulta muy útil porque son tecnologías muy usadas en el ámbito laboral.

Las consultas, por otra parte, tuvieron un nivel de dificultad variado, yendo de preguntas muy complejas hasta algunas muy simples, las que incentivaron al equipo a plantearse nuevas formas de resolver problemas. Esto es cierto más que nada con la primera consulta, que hizo que el equipo pensara y abordara el problema de una forma diferente, lo que llevó a aprender nuevas herramientas para el manejo de las bases de datos.

## Referencias

IA abierta. (2023). *ChatGPT* (versión del 03 de julio)

<https://chat.openai.com/chat>