

Modelo Relacional para una Base de Datos de Fútbol: Evaluación de Rendimiento de Equipos y Jugadores

Michael David Gualteros García¹, Raúl Andrés Gamba Hastamorir²,
Juan Sebastián Heredia Heredia³

¹⁻²⁻³Maestría en Analítica de Datos

Universidad Central

Curso de Bases de Datos

Bogotá, Colombia

{¹*mgualterosg*,²*rgambah*,³*jherediah1*}@ucentral.edu.co

November 29, 2024

Contents

1	Introducción	4
2	Características del proyecto de investigación que hace uso de Bases de Datos	4
2.1	Titulo del proyecto de investigación	4
2.2	Objetivo general	4
2.3	Objetivos específicos	4
2.4	Alcance	5
2.5	Pregunta de investigación	5
2.6	Hipotesis	5
3	Reflexiones sobre el origen de datos e información.	6
3.1	¿Cual es el origen de los datos e información ?	6
3.2	¿Cuales son las consideraciones legales o eticas del uso de la información?	6
3.3	¿Cuales son los retos de la información y los datos que utilizara en la base de datos en terminos de la calidad y la consolidación?	6
3.4	¿Que espera de la utilización de un sistema de Bases de Datos para su proyecto?	7
4	Diseño del Modelo de Datos del SMBD (Sistema Manejador de Bases de Datos)	8

4.1	Características del SMBD (Sistema Manejador de Bases de Datos)	
	para el Proyecto	8
4.2	Diagrama modelo de datos	9
4.3	Imágenes de la Base de Datos	10
4.4	Código SQL - lenguaje de definición de datos (DDL)	12
4.4.1	Tabla Estadios	12
4.4.2	Tabla Clubes	12
4.4.3	Tabla Jugadores	13
4.4.4	Tabla Posiciones	13
4.4.5	Tabla Técnicos	14
4.4.6	Otras sentencias	14
4.5	Código SQL - Manipulación de datos (DML)	14
4.5.1	Tabla posiciones	14
4.5.2	Tabla Técnicos	15
4.5.3	Tabla Estadios	15
4.5.4	Tabla Clubes	15
4.5.5	Tabla Jugadores	16
4.6	Código SQL + Resultados: Vistas	16
4.6.1	Vista de Clubes y Estadios	16
4.6.2	Resultados de la Vista de Clubes y Estadios	17
4.6.3	Vista de Jugadores por País	17
4.6.4	Resultados de la Vista de Jugadores por País	17
4.6.5	Vista de Técnicos por País	18
4.6.6	Resultados de la Vista de Técnicos por País	18
4.6.7	Vista de Jugadores por Posición	19
4.6.8	Resultados de la Vista de Jugadores por Posición	19
4.6.9	Vista de Estadísticas de Clubes	20
4.6.10	Resultados de la Vista de Estadísticas de Clubes	20
4.6.11	Vista de los 15 Jugadores Más Caros	20
4.6.12	Resultados de la Vista de los 15 Jugadores Más Caros	21
4.7	Código SQL + Resultados: Triggers	21
4.7.1	Trigger: Validar estadísticas jugadores	21
4.7.2	Resultados del Trigger: Validar estadísticas jugadores	22
4.8	Código SQL + Resultados: Funciones	23
4.8.1	Función para Calcular Goles por Edad	23
4.8.2	Resultados de la Función para Calcular Goles por Edad	23
4.8.3	Función para Calcular Minutos Jugados por Edad	23
4.8.4	Resultados de la Función para Calcular Minutos Jugados por Edad	24
4.9	Código SQL + Resultados: procedimientos almacenados	24
4.9.1	Procedimiento actualizar-valor-mercado-jugadores	24
4.9.2	Contexto del Procedimiento	25
4.9.3	Parámetros del Procedimiento	25
4.9.4	Importancia de los Parámetros	25
4.9.5	Ejecución y resultados procedimiento	26
4.9.6	Procedimiento actualizar-estadísticas-clubes	26

4.9.7	Funcionamiento	27
4.9.8	Ejecución y resultados procedimiento	27
5	Bases de Datos No-SQL	28
5.1	Diagrama Bases de Datos No-SQ	28
5.2	SMBD utilizado para la Base de Datos No-SQL	32
5.2.1	¿Por qué MongoDB para este proyecto?	32
5.2.2	Explicación del código:	33
6	Aplicación de ETL (Extract, Transform, Load) y Bodega de Datos	35
6.1	Ejemplo de aplicación de ETL y Bodega de Datos	35
6.2	Automatización de Datos	35
6.3	Integración de Datos	36
7	Proximos pasos	38
8	Lecciones aprendidas	39
9	Bibliografía	42

1 Introducción

El fútbol ha evolucionado de ser un deporte a una industria multimillonaria, donde la valoración de los jugadores es clave. En grandes ligas como la Premier League, los clubes invierten grandes sumas en fichajes, buscando optimizar tanto el rendimiento en el campo como los ingresos comerciales. El valor de un jugador no solo se basa en su rendimiento deportivo, sino también en su edad, potencial de marketing y demanda en el mercado de traspasos.

Actualmente, el uso de datos es esencial en la toma de decisiones dentro de los clubes. Sin embargo, la valoración de los futbolistas sigue siendo en gran parte subjetiva, lo que genera incertidumbre en un mercado de fichajes volátil. La Premier League, con su alto nivel competitivo y gran audiencia global, es un entorno ideal para desarrollar un modelo que permita organizar y evaluar de manera más estructurada los datos relacionados con los jugadores y los equipos.

Este proyecto se centra en la creación de un modelo relacional de base de datos para la Premier League, que permitirá analizar y gestionar de manera más eficiente información clave como el rendimiento de los jugadores, las características tácticas de los equipos y otros factores determinantes. Con una estructura de datos bien diseñada, los clubes podrán acceder a información más precisa y objetiva, mejorando sus decisiones en cuanto a fichajes y gestión de plantillas. En un mercado tan competitivo, contar con una base de datos relacional sólida será crucial para optimizar los recursos y mantenerse a la vanguardia.

(Realizar referenciación y citación del texto para generar la solidez del texto)
(Mencionar la importancia del modelo)

2 Características del proyecto de investigación que hace uso de Bases de Datos

2.1 Título del proyecto de investigación

Modelo Relacional para una Base de Datos de Fútbol: Evaluación de Rendimiento de Equipos y Jugadores

2.2 Objetivo general

Desarrollar un modelo relacional de base de datos para la Premier League que permita gestionar y analizar de manera eficiente la información sobre jugadores, equipos y estadísticas, con el fin de optimizar la toma de decisiones en la gestión deportiva y financiera de los clubes.

(validar el alcance del objetivo general)

2.3 Objetivos específicos

- Crear un esquema relacional que contemple las principales entidades y relaciones del fútbol, incluyendo jugadores, equipos, partidos y estadísticas,

asegurando la integridad y consistencia de los datos.

- Desarrollar herramientas que permitan a los usuarios realizar consultas complejas y generar reportes sobre el rendimiento de los jugadores, análisis táctico y tendencias de mercado, facilitando el acceso a información relevante para la toma de decisiones.
- Establecer mecanismos para importar y actualizar datos de fuentes externas, como análisis de rendimiento y estadísticas de medios deportivos, con el objetivo de enriquecer la base de datos y mejorar la calidad del análisis

2.4 Alcance

El alcance del proyecto consiste en desarrollar un modelo relacional de base de datos para gestionar de manera integral la información relacionada con jugadores, equipos, partidos y estadísticas de la Premier League. Este modelo permitirá almacenar y organizar datos de múltiples temporadas, asegurando la integridad y consistencia de la información a través de un esquema relacional robusto que incluye las principales entidades y sus interrelaciones. La base de datos facilitará la realización de consultas complejas y la generación de reportes detallados, cubriendo desde el rendimiento individual de jugadores hasta análisis tácticos de equipos y tendencias de mercado.

Además, el proyecto incluirá la implementación de mecanismos que permitan la importación y actualización automática de datos provenientes de fuentes externas, como análisis de rendimiento y estadísticas deportivas, asegurando que la base de datos se mantenga actualizada y relevante. Esto permitirá a los usuarios acceder a información precisa para la toma de decisiones en la gestión deportiva y financiera de los clubes, con un enfoque en mejorar la planificación de fichajes, la optimización de tácticas y el análisis de mercado en tiempo real.

2.5 Pregunta de investigación

¿Cuáles son los factores más determinantes para la variación en el precio de mercado de un futbolista de la Premier League?

2.6 Hipotesis

El precio de mercado de un futbolista está significativamente influenciado por su rendimiento en el campo, donde las variables como goles y asistencias tienen un mayor impacto en el valor de delanteros, mientras que los minutos jugados y las apariciones son más determinantes para defensas, mediocampistas y porteros. Además, las tarjetas recibidas tienen un efecto negativo en el valor de mercado de todos los jugadores, aunque en menor medida.

3 Reflexiones sobre el origen de datos e información.

3.1 ¿Cual es el origen de los datos e información ?

En plataformas como Transfermarkt y FootyStats, los datos provienen de diversas fuentes oficiales y públicas, como ligas deportivas, clubes, asociaciones de fútbol, medios de comunicación y estadísticas recopiladas en tiempo real durante los partidos. También integran contribuciones de usuarios y expertos que verifican, actualizan y complementan la información. Estos datos incluyen estadísticas de rendimiento de jugadores, detalles de transferencias, resultados de partidos y otras métricas relevantes. Al ser plataformas de dominio público, la información está accesible para usuarios, pero debe respetarse su uso ético y legal, como el cumplimiento de normativas de derechos de autor y la protección de datos personales, evitando la explotación comercial sin autorización.

3.2 ¿Cuales son las consideraciones legales o eticas del uso de la información?

Para este proyecto se usaron las plataformas, Transfermarkt y FootyStats para consultar información futbolística, en estas es esencial considerar los aspectos legales y éticos relacionados con la protección de datos y la propiedad intelectual. Ambas páginas manejan información sobre jugadores, equipos y competiciones que puede estar protegida por derechos de autor. Es importante atribuir adecuadamente las fuentes y no reproducir contenido sin autorización, respetando las normativas de derechos de propiedad intelectual aplicables.

En cuanto al uso ético, la información debe ser empleada para fines legítimos, evitando el uso indebido con fines comerciales no autorizados o manipulación de datos para actividades fraudulentas, como apuestas ilegales. También es clave respetar la privacidad de los datos personales de los jugadores, cumpliendo con regulaciones como el GDPR (Reglamento General de Protección de Datos) en Europa, que protege la información personal y establece el derecho a la privacidad.

3.3 ¿Cuales son los retos de la información y los datos que utilizara en la base de datos en terminos de la calidad y la consolidación?

Uno de los principales desafíos es asegurar la veracidad y actualización de los datos, ya que la información sobre estadísticas y transferencias puede cambiar rápidamente. Es fundamental integrar datos de diversas fuentes, como FootyStats y Transfermarkt, de manera coherente, evitando duplicaciones y garantizando la estandarización. Además, la consolidación de datos debe abordar la inconsistencia de formatos para asegurar que la base de datos sea accesible y útil para análisis precisos. Esto implica un riguroso proceso de validación y verificación, así como la implementación de herramientas adecuadas para monitorear cambios

en tiempo real, asegurando que la información sea fiable y relevante para la toma de decisiones.

3.4 ¿Que espera de la utilización de un sistema de Bases de Datos para su proyecto?

Se espera que la utilización de un sistema de bases de datos en el proyecto facilite la organización eficiente de datos sobre jugadores, equipos y estadísticas, mejorando así la toma de decisiones en los clubes. Este sistema permitirá generar reportes claros sobre el rendimiento, proporcionando herramientas útiles para análisis estratégico. Además, se busca que sea flexible y escalable, permitiendo la incorporación de más datos en el futuro. Finalmente, se implementarán mecanismos de seguridad para garantizar que solo personal autorizado acceda a información sensible, promoviendo una gestión más efectiva en el ámbito futbolístico.

4 Diseño del Modelo de Datos del SMBD (Sistema Manejador de Bases de Datos)

4.1 Características del SMBD (Sistema Manejador de Bases de Datos) para el Proyecto

En el presente proyecto, se utilizará **Oracle PL/SQL** como el sistema manejador de bases de datos. A continuación, se presentan las características más relevantes de Oracle PL/SQL que lo hacen adecuado para el desarrollo y la gestión de nuestra base de datos:

1. **Soporte para Procedimientos Almacenados:** Oracle PL/SQL permite la creación de procedimientos y funciones almacenadas, lo que facilita la encapsulación de la lógica de negocio y mejora la reutilización del código. Esto es especialmente útil para realizar cálculos complejos, como la estimación del valor de mercado de los jugadores, y para mantener la integridad de los datos.
2. **Manejo Eficiente de Transacciones:** PL/SQL proporciona un manejo robusto de transacciones, asegurando que las operaciones sobre la base de datos sean atómicas. Esto significa que las transacciones se pueden deshacer en caso de error, garantizando la consistencia y la integridad de los datos.
3. **Integración de SQL y PL/SQL:** Oracle PL/SQL combina la potencia del lenguaje de consulta SQL con la programación estructurada, permitiendo a los desarrolladores realizar operaciones complejas de manera más efectiva. Esto es esencial para interactuar con nuestras tablas, ejecutar consultas y manipular datos de forma dinámica.
4. **Gestión de Excepciones:** PL/SQL incluye un manejo de excepciones sofisticado, lo que permite a los desarrolladores gestionar errores y condiciones excepcionales de manera controlada. Esto es crucial para mantener la estabilidad de la aplicación y proporcionar una mejor experiencia al usuario.
5. **Rendimiento Optimizado:** Oracle PL/SQL está diseñado para optimizar el rendimiento de las aplicaciones al permitir que las operaciones se realicen en el servidor, reduciendo así la cantidad de datos que deben transferirse entre el servidor y el cliente. Esto resulta en una ejecución más rápida y eficiente de las consultas y procedimientos.
6. **Soporte para Programación Orientada a Objetos:** PL/SQL ofrece características de programación orientada a objetos, lo que permite a los desarrolladores crear tipos de datos definidos por el usuario y utilizar principios de encapsulamiento, herencia y polimorfismo. Esto puede ser útil para modelar estructuras más complejas dentro de nuestra base de datos.

7. **Escalabilidad y Flexibilidad:** Oracle PL/SQL es altamente escalable, lo que lo hace adecuado para aplicaciones de cualquier tamaño, desde pequeños proyectos hasta grandes sistemas empresariales. Su flexibilidad permite adaptar la base de datos a las necesidades cambiantes del proyecto a medida que evoluciona.
8. **Seguridad y Control de Acceso:** Oracle ofrece mecanismos de seguridad avanzados que permiten establecer controles de acceso a nivel de usuario y a nivel de objeto. Esto es esencial para proteger la información sensible y garantizar que solo los usuarios autorizados tengan acceso a los datos.
9. **Facilidad de Integración:** PL/SQL facilita la integración con otras tecnologías y herramientas de Oracle, así como con sistemas de terceros, lo que permite una mejor interoperabilidad en entornos heterogéneos.
10. **Herramientas de Desarrollo:** Oracle proporciona diversas herramientas de desarrollo, como Oracle SQL Developer, que permiten una gestión más eficiente de las bases de datos, facilitando la creación de objetos, la ejecución de consultas y la depuración de procedimientos.

Oracle PL/SQL es una elección sólida para este proyecto debido a su robustez, flexibilidad y características avanzadas que facilitan el desarrollo y la gestión de bases de datos. Estas características ayudarán a asegurar que el sistema sea eficiente, seguro y capaz de adaptarse a las necesidades del análisis de datos relacionados con el rendimiento de los futbolistas y sus valores de mercado.

4.2 Diagrama modelo de datos

El diagrama de base de datos presenta un diseño estructurado que abarca un total de cinco tablas, cada una diseñada para almacenar información específica relacionada con clubes de fútbol, sus jugadores y el cuerpo técnico. Este diseño permite la organización eficiente de datos y establece relaciones entre diferentes entidades, facilitando la consulta y el manejo de la información.

La primera tabla, estadios, almacena detalles sobre los estadios, incluyendo el identificador único de cada estadio, su nombre, ciudad y capacidad. Esta tabla es fundamental para relacionar los clubes con sus respectivos estadios, permitiendo obtener información sobre el lugar donde se llevan a cabo los partidos.

La segunda tabla, clubes, es el núcleo del sistema, donde se registra información sobre cada club, su nombre, y la relación con los estadios y directores técnicos (DT). Esta tabla incluye un campo para el identificador del estadio, lo que permite vincular a cada club con el estadio donde juega. Además, cuenta con una clave foránea que hace referencia a la tabla de técnicos.

La tabla de jugadores contiene información detallada sobre los futbolistas, incluyendo su nombre, apellido, posición, estadísticas de juego y el club al que pertenecen. Esta tabla no solo permite almacenar datos relevantes sobre el

rendimiento de los jugadores, sino que también establece relaciones con las tablas de posiciones y clubes.

La tabla de posiciones es crucial para clasificar a los jugadores en función de su rol dentro del equipo, como delantero, mediocampista o defensor. Este enfoque permite realizar análisis más profundos sobre la composición de los equipos y su rendimiento en el campo.

Por último, la tabla de tecnicos almacena información sobre los directores técnicos de los clubes, incluyendo su nombre, apellido, país y edad. Esta tabla permite gestionar datos sobre el cuerpo técnico, un aspecto esencial para el rendimiento y la estrategia de cada club.

En conjunto, este diagrama de base de datos no solo facilita la organización de información relevante en el contexto del fútbol, sino que también permite la implementación de consultas complejas para obtener estadísticas y reportes útiles para la gestión deportiva. La interrelación entre las tablas garantiza la integridad de los datos y mejora la eficiencia en la recuperación de información.

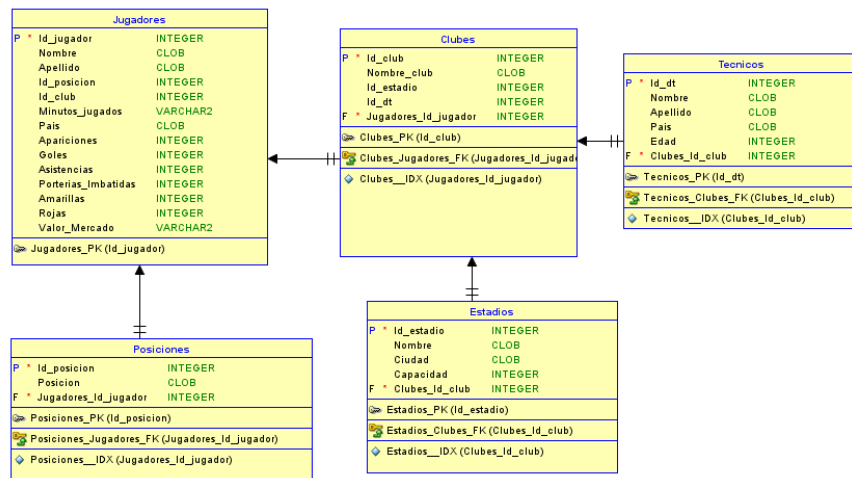


Figure 1: Diagrama relacional de la base de datos

4.3 Imágenes de la Base de Datos

A continuación se presentan imágenes de la información en la base de datos:

	ID_POSICION	POSICION
1	1	Defender
2	2	Midfielder
3	3	Goalkeeper
4	4	Forward

Figure 2: Tabla Posiciones en la base de datos

ID_DT	NOMBRE	APELLIDO	PAIS	EDAD
1	Eddie	Howe	Reino Unido	41
2	Unai	Emery	España	47
3	Chris	Bloom	Reino Unido	48
4	Sean	Dyche	Reino Unido	48
5	Neil	Warnock	Reino Unido	70
6	Maurizio	Sarri	Italia	60
7	Roy	Hodgson	Reino Unido	71
8	Marco	Silva	Portugal	41
9	Slaviša	Jokanović	Serbia	50
10	David	Wagner	Alemania	47
11	Brendan	Rodgers	Reino Unido	46

Figure 3: Tabla Técnicos en la base de datos

ID_ESTADIO	NOMBRE	CIUDAD	CAPACIDAD
5	Cardiff City Stadium	Cardiff	33332
6	Stamford Bridge	Londres	40834
7	Selhurst Park	Londres	26200
8	Goodison Park	Liverpool	39414
9	Craven Cottage	Londres	25200
10	John Smiths Stadium	Huddersfield	24500
11	King Power Stadium	Leicester	32262
12	Anfield	Liverpool	54074
13	Etihad Stadium	Manchester	55300
14	Old Trafford	Manchester	74879
15	St James Park	Newcastle	52305

Figure 4: Tabla Estadios en la base de datos

ID_CLUB	NOMBRE	ID_ESTADIO	ID_DT	TOTAL_G...	TOTAL_A...	TOTAL_A...	TOTAL_R...	TOTAL_PORTERIAS_IMBATIDAS
1	1 AFC Bou...	1	1	0	0	0	0	0
2	2 Arsenal	2	2	0	0	0	0	0
3	3 Brighto...	3	3	0	0	0	0	0
4	4 Burnley	4	4	0	0	0	0	0
5	5 Cardiff...	5	5	0	0	0	0	0
6	6 Chelsea	6	6	0	0	0	0	0
7	7 Crystal...	7	7	0	0	0	0	0
8	8 Everton	8	8	0	0	0	0	0
9	9 Fulham	9	9	0	0	0	0	0
10	10 Hudders...	10	10	0	0	0	0	0

Figure 5: Tabla Clubes en la base de datos

ID_JUGADOR	NOMBRE	APELLIDO	EDAD	ID_POSICION	ID_CLUB	MINUTOS_JUGADOS	PAIS	APARICIONES	GOLES	AS
561	319 Alex	Oxlade-Chamberlain	29	2	12	19	Inglaterra	2	0	
562	320 Alisson	Becker	30	3	12	3420	Brasil	38	0	
563	321 Andrew	Robertson	28	1	12	3219	Escocia	36	0	
564	322 Curtis	Jones	21	2	12	0	Inglaterra	0	0	
565	323 Daniel	Sturridge	33	4	12	496	Inglaterra	18	2	
566	324 Dejan	Lovren	33	1	12	985	Croacia	13	1	
567	325 Divock	Origi	27	4	12	366	Bélgica	12	3	
568	326 Fabinho	(null)	29	2	12	2013	Brasil	28	1	
569	327 Georginio	Wijnaldum	31	2	12	2735	Países Bajos	35	3	
570	328 James	Milner	36	2	12	1785	Inglaterra	31	5	

Figure 6: Tabla Jugadores en la base de datos

4.4 Código SQL - lenguaje de definición de datos (DDL)

El Lenguaje de Definición de Datos (DDL) es un componente esencial del SQL que se utiliza para definir y modificar la estructura de una base de datos. A través de comandos como CREATE, ALTER y DROP, el DDL permite crear tablas, establecer relaciones y gestionar la organización de los datos. A continuación, se presentan los códigos DDL utilizados en el presente proyecto para estructurar la base de datos de clubes de fútbol y sus jugadores.

4.4.1 Tabla Estadios

```
CREATE TABLE estadios (
    id_estadio INT NOT NULL,
    nombre VARCHAR(255) NOT NULL,
    ciudad VARCHAR(255) NOT NULL,
    capacidad INT,
    PRIMARY KEY (id_estadio)
);
```

4.4.2 Tabla Clubes

```
CREATE TABLE clubes (
    id_club INT NOT NULL,
```

```

    nombre_club          VARCHAR(255) NOT NULL,
    id_estadio           INT,
    id_dt                INT,
    PRIMARY KEY (id_club),
    FOREIGN KEY (id_estadio) REFERENCES estadios(id_estadio),
    FOREIGN KEY (id_dt) REFERENCES tecnicos(id_dt)
);
ALTER TABLE Clubes
ADD Total_goles INTEGER DEFAULT 0;
ALTER TABLE Clubes
ADD Total_asistencias INTEGER DEFAULT 0;
ALTER TABLE Clubes
ADD Total_amarillas INTEGER DEFAULT 0;
ALTER TABLE Clubes
ADD Total_rojas INTEGER DEFAULT 0;
ALTER TABLE Clubes
ADD Total_porterias_imbatidas INTEGER DEFAULT 0;

```

4.4.3 Tabla Jugadores

```

CREATE TABLE jugadores (
    id_jugador           INT NOT NULL,
    nombre               VARCHAR(255) NOT NULL,
    apellido             VARCHAR(255) NOT NULL,
    id_posicion          INT,
    id_club              INT,
    minutos_jugados      INT,
    pais                 VARCHAR(255),
    apariciones          INT,
    goles               INT,
    asistencias          INT,
    porterias_imbatidas INT,
    amarillas            INT,
    rojas                INT,
    valor_mercado        NUMBER,
    PRIMARY KEY (id_jugador),
    FOREIGN KEY (id_posicion) REFERENCES posiciones(id_posicion),
    FOREIGN KEY (id_club) REFERENCES clubes(id_club)
);

```

4.4.4 Tabla Posiciones

```

CREATE TABLE posiciones (
    id_posicion          INT NOT NULL,
    posicion             VARCHAR(255) NOT NULL,

```

```

        PRIMARY KEY (id_posicion)
    );

```

4.4.5 Tabla Técnicos

```

CREATE TABLE tecnicos (
    id_dt          INT NOT NULL,
    nombre         VARCHAR(255) NOT NULL,
    apellido       VARCHAR(255),
    pais           VARCHAR(255),
    edad           INT,
    PRIMARY KEY (id_dt),
);

```

En las tablas definidas en la base de datos, es posible realizar diversas operaciones como modificaciones, inserciones y eliminaciones de campos que no son necesarios. Ejemplos de tales operaciones son las siguientes sentencias:

4.4.6 Otras sentencias

```

ALTER TABLE Jugadores
ADD COLUMN numero_camisa INT;

ALTER TABLE Estadios
MODIFY capacidad INT NOT NULL;

DROP TABLE IF EXISTS Jugadores;

DROP TABLE IF EXISTS Clubes;

```

Aunque estas instrucciones son útiles para ajustar la estructura de la base de datos, en el presente proyecto no se utilizaron, ya que las tablas fueron diseñadas para satisfacer los requisitos específicos del mismo.

4.5 Código SQL - Manipulación de datos (DML)

Para este proyecto se recopiló la información y se hicieron los inserts para cada tabla.

4.5.1 Tabla posiciones

A continuación se presentan algunas de las sentencias ejecutadas

```

INSERT INTO posiciones (Id_Posicion, Posicion) VALUES (1, 'Defender');
INSERT INTO posiciones (Id_Posicion, Posicion) VALUES (2, 'Midfielder');
INSERT INTO posiciones (Id_Posicion, Posicion) VALUES (3, 'Goalkeeper');
INSERT INTO posiciones (Id_Posicion, Posicion) VALUES (4, 'Forward');

```

4.5.2 Tabla Técnicos

A continuación se presentan algunas de las sentencias ejecutadas

```
INSERT INTO tecnicos (Id_dt, Nombre, Apellido, Pais, Edad)
VALUES (1, 'Eddie', 'Howe', 'Reino Unido', 41);
INSERT INTO tecnicos (Id_dt, Nombre, Apellido, Pais, Edad)
VALUES (2, 'Unai', 'Emery', 'España', 47);
INSERT INTO tecnicos (Id_dt, Nombre, Apellido, Pais, Edad)
VALUES (3, 'Chris', 'Bloom', 'Reino Unido', 48);
INSERT INTO tecnicos (Id_dt, Nombre, Apellido, Pais, Edad)
VALUES (4, 'Sean', 'Dyche', 'Reino Unido', 48);
INSERT INTO tecnicos (Id_dt, Nombre, Apellido, Pais, Edad)
VALUES (5, 'Neil', 'Warnock', 'Reino Unido', 70);
INSERT INTO tecnicos (Id_dt, Nombre, Apellido, Pais, Edad)
VALUES (6, 'Maurizio', 'Sarri', 'Italia', 60);
```

4.5.3 Tabla Estadios

A continuación se presentan algunas de las sentencias ejecutadas

```
INSERT INTO estadios (Id_Estadio, Nombre, Ciudad, Capacidad)
VALUES (1, 'Vitality Stadium', 'Bournemouth', '11700');
INSERT INTO estadios (Id_Estadio, Nombre, Ciudad, Capacidad)
VALUES (2, 'Emirates Stadium', 'Londres', '60432');
INSERT INTO estadios (Id_Estadio, Nombre, Ciudad, Capacidad)
VALUES (3, 'Amex Stadium', 'Brighton', '30750');
INSERT INTO estadios (Id_Estadio, Nombre, Ciudad, Capacidad)
VALUES (4, 'Turf Moor', 'Burnley', '21401');
INSERT INTO estadios (Id_Estadio, Nombre, Ciudad, Capacidad)
VALUES (5, 'Cardiff City Stadium', 'Cardiff', '33332');
```

4.5.4 Tabla Clubes

A continuación se presentan algunas de las sentencias ejecutadas

```
INSERT INTO clubes (Id_Club, Nombre_club, Id_Estadio, Id_dt, Total_goles,
Total_asistencias, Total_amarillas, Total_rojas, Total_porterias_imbatidas)
VALUES (1, 'AFC Bournemouth', 1, 1, 0, 0, 0, 0, 0);
INSERT INTO clubes (Id_Club, Nombre_club, Id_Estadio, Id_dt, Total_goles,
Total_asistencias, Total_amarillas, Total_rojas, Total_porterias_imbatidas)
VALUES (2, 'Arsenal', 2, 2, 0, 0, 0, 0, 0);
INSERT INTO clubes (Id_Club, Nombre_club, Id_Estadio, Id_dt, Total_goles,
Total_asistencias, Total_amarillas, Total_rojas, Total_porterias_imbatidas)
VALUES (3, 'Brighton Hove Albion', 3, 3, 0, 0, 0, 0, 0);
INSERT INTO clubes (Id_Club, Nombre_club, Id_Estadio, Id_dt, Total_goles,
Total_asistencias, Total_amarillas, Total_rojas, Total_porterias_imbatidas)
VALUES (4, 'Burnley', 4, 4, 0, 0, 0, 0, 0);
```

```
INSERT INTO clubes (Id_Club, Nombre_club, Id_Estadio, Id_dt, Total_goles,
Total_asistencias, Total_amarillas, Total_rojas, Total_porterias_imbatidas)
VALUES (5, 'Cardiff City', 5, 5, 0, 0, 0, 0, 0);
```

4.5.5 Tabla Jugadores

A continuación se presentan algunas de las sentencias ejecutadas

```
INSERT INTO jugadores (id_jugador, nombre, apellido, edad, id_posicion,
id_club, minutos_jugados, pais, apariciones, goles, asistencias, porterias_imbatidas,
amarillas, rojas, valor_mercado)
VALUES ('1', 'Adam', 'Smith', '31', 1, 1, 2073, 'Inglaterra', 25, 1, 1, 9, 6, 1, 7);
INSERT INTO jugadores (id_jugador, nombre, apellido, edad, id_posicion,
id_club, minutos_jugados, pais, apariciones, goles, asistencias, porterias_imbatidas,
amarillas, rojas, valor_mercado)
VALUES ('2', 'Andrew', 'Surman', '36', 2, 1, 1438, 'Inglaterra', 18, 0, 0, 5, 2, 0, 2.5);
INSERT INTO jugadores (id_jugador, nombre, apellido, edad, id_posicion,
id_club, minutos_jugados, pais, apariciones, goles, asistencias, porterias_imbatidas,
amarillas, rojas, valor_mercado)
VALUES ('3', 'Artur', 'Boruc', '42', 3, 1, 1080, 'Polonia', 12, 0, 0, 4, 2, 0, 1);
INSERT INTO jugadores (id_jugador, nombre, apellido, edad, id_posicion,
id_club, minutos_jugados, pais, apariciones, goles, asistencias, porterias_imbatidas,
amarillas, rojas, valor_mercado)
VALUES ('4', 'Asmir', 'Begović', '35', 3, 1, 2160, 'Bosnia y Herzegovina',
24, 0, 0, 5, 0, 0, 7);
```

En el anexo se encuentra la totalidad de las sentencias

4.6 Código SQL + Resultados: Vistas

Para el presente proyecto se plantearon seis vistas, cada una diseñada para facilitar el análisis y la consulta de información relevante en la base de datos. A continuación, se detallan las vistas junto con su código y propósito:

4.6.1 Vista de Clubes y Estadios

Esta vista permite visualizar información sobre los clubes y sus estadios. Proporciona detalles como el nombre del club, el nombre del estadio, la ciudad y la capacidad del estadio.

```
CREATE VIEW vista_clubes_estadios AS
SELECT
    c.id_club,
    c.nombre_club,
    e.nombre AS nombre_estadio,
    e.ciudad,
    e.capacidad
FROM
```



```

clubes c
JOIN
estadios e ON c.id_estadio = e.id_estadio;

```

4.6.2 Resultados de la Vista de Clubes y Estadios

ID_CLUB	NOMBRE_CLUB	NOMBRE_ESTADIO	CIUDAD	CAPACIDAD
1	1 AFC Bournemouth	Vitality Stadium	Bournemouth	11700
2	2 Arsenal	Emirates Stadium	Londres	60432
3	3 Brighton Hove Albion	Amex Stadium	Brighton	30750
4	4 Burnley	Turf Moor	Burnley	21401
5	5 Cardiff City	Cardiff City Stadium	Cardiff	33332
6	6 Chelsea	Stamford Bridge	Londres	40834
7	7 Crystal Palace	Selhurst Park	Londres	26200
8	8 Everton	Goodison Park	Liverpool	39414
9	9 Fulham	Craven Cottage	Londres	25200
10	10 Huddersfield Town	John Smiths Stadium	Huddersfield	24500
...				

Figure 7: Vista de Clubes y Estadios

4.6.3 Vista de Jugadores por País

Esta vista ofrece un resumen de la cantidad de jugadores agrupados por país, lo que facilita el análisis de la diversidad de jugadores en los clubes.

```

CREATE VIEW vista_jugadores_por_pais AS
SELECT
    j.pais,
    COUNT(j.id_jugador) AS total_jugadores
FROM
    jugadores j
GROUP BY
    j.pais;

```

4.6.4 Resultados de la Vista de Jugadores por País

PAIS	TOTAL_JUGADORES
1 Bosnia y Herzegovina	2
2 Gabón	3
3 Israel	1
4 Guinea	1
5 Corea del Sur	2
6 República de Irlanda	20
7 Argentina	16
8 Irán	1
9 Filipinas	1
10 Kenia	1

Figure 8: Vista de Jugadores por País

4.6.5 Vista de Técnicos por País

Esta vista proporciona información sobre los técnicos agrupados por país, permitiendo identificar de dónde provienen los entrenadores de los clubes.

```
CREATE VIEW vista_tecnicos_por_pais AS
SELECT
    t.pais,
    COUNT(t.id_dt) AS total_tecnicos
FROM
    tecnicos t
GROUP BY
    t.pais;
```

4.6.6 Resultados de la Vista de Técnicos por País

	PAIS	TOTAL_TECNICOS
1	Argentina	1
2	Portugal	2
3	Noruega	1
4	Alemania	2
5	Italia	1
6	Austria	1
7	Reino Unido	6
8	Chile	1
9	España	4
10	Serbia	1

Figure 9: Vista de Técnicos por País

4.6.7 Vista de Jugadores por Posición

Esta vista permite conocer la cantidad de jugadores por cada posición en el campo, facilitando la evaluación de la estructura del equipo.

```
CREATE VIEW vista_jugadores_por_posicion AS
SELECT
    p.posicion,
    COUNT(j.id_jugador) AS total_jugadores
FROM
    jugadores j
JOIN
    posiciones p ON j.id_posicion = p.id_posicion
GROUP BY
    p.posicion;
```

4.6.8 Resultados de la Vista de Jugadores por Posición

	POSICION	TOTAL_JUGADORES
1	Midfielder	211
2	Forward	113
3	Goalkeeper	57
4	Defender	189

Figure 10: Vista de Jugadores por Posición

4.6.9 Vista de Estadísticas de Clubes

Esta vista proporciona un resumen de las estadísticas de rendimiento de cada club, incluyendo total de goles, asistencias, tarjetas amarillas y rojas, y porterías imbatidas.

```
CREATE VIEW vista_estadisticas_clubes AS
SELECT
    c.id_club,
    c.nombre_club,
    c.Total_goles,
    c.Total_asistencias,
    c.Total_amarillas,
    c.Total_rojas,
    c.Total_porterias_imbatidas
FROM
    clubes c;
```

4.6.10 Resultados de la Vista de Estadísticas de Clubes

ID...	NOMBRE CLUB	TOTAL_GOLES	TOTAL_ASISTENCIAS	TOTAL_AMARILLAS	TOTAL_ROJAS	TOTAL_PORTERIAS_IMBATIDAS
1	1 AFC Bournemouth	0	0	0	0	0
2	2 Arsenal	0	0	0	0	0
3	3 Brighton Hove Albion	0	0	0	0	0
4	4 Burnley	0	0	0	0	0
5	5 Cardiff City	0	0	0	0	0
6	6 Chelsea	0	0	0	0	0
7	7 Crystal Palace	0	0	0	0	0
8	8 Everton	0	0	0	0	0
9	9 Fulham	0	0	0	0	0

Figure 11: Vista de Estadísticas de Clubes

4.6.11 Vista de los 15 Jugadores Más Caros

Esta vista proporciona una lista de los 15 jugadores más caros, mostrando su nombre, apellido, club y valor de mercado. Esta vista es útil para analizar los jugadores con mayor valor en el mercado.

```
CREATE VIEW vista_jugadores_mas_caros AS
SELECT
    j.id_jugador,
    j.nombre,
    j.apellido,
    c.nombre_club,
    j.valor_mercado
FROM
    jugadores j
JOIN
    clubes c ON j.id_club = c.id_club
```

```
ORDER BY
    j.valor_mercado DESC
FETCH FIRST 15 ROWS ONLY;
```

4.6.12 Resultados de la Vista de los 15 Jugadores Más Caros

ID_JUGADOR	NOMBRE	APELLIDO	NOMBRE_CLUB	VALOR_MERCADO
1	333 Mohamed	Salah	Liverpool	150
2	155 Eden	Hazard	Chelsea	150
3	468 Harry	Kane	Tottenham Hotspur	150
4	362 Raheem	Sterling	Manchester City	140
5	355 Kevin	De Bruyne	Manchester City	130
6	337 Sadio	Mané	Liverpool	120
7	345 Bernardo	Silva	Manchester City	100
8	165 NGolo	Kanté	Chelsea	100
9	459 Christian	Eriksen	Tottenham Hotspur	100
10	389 Paul	Pogba	Manchester United	100

Figure 12: Vista de de los 15 Jugadores Más Caros

4.7 Código SQL + Resultados: Triggers

Para el presente proyecto, se ha planteado un triggers que garantizan la integridad de los datos en la base de datos y la consistencia de las estadísticas de los jugadores y clubes.

4.7.1 Trigger: Validar estadísticas jugadores

Este trigger se activa antes de realizar una actualización en la tabla jugadores. Su función es validar que los nuevos valores de las estadísticas de cada jugador no sean menores que los valores actuales. Esto es crucial para mantener un registro preciso y evitar inconsistencias en los datos.

```
CREATE OR REPLACE TRIGGER validar_estadisticas_jugadores
BEFORE UPDATE ON jugadores
FOR EACH ROW
BEGIN
    IF :NEW.goles < :OLD.goles THEN
        RAISE_APPLICATION_ERROR(-20001,
            'El número de goles no puede ser menor que el actual.');
```

```
    END IF;

    IF :NEW.asistencias < :OLD.asistencias THEN
        RAISE_APPLICATION_ERROR(-20002,
            'El número de asistencias no puede ser menor que el actual.');
```

```
    END IF;
```

```

IF :NEW.amarillas < :OLD.amarillas THEN
    RAISE_APPLICATION_ERROR(-20003,
        'El número de tarjetas amarillas no puede ser menor que el actual.');
```

END IF;

```

IF :NEW.rojas < :OLD.rojas THEN
    RAISE_APPLICATION_ERROR(-20004,
        'El número de tarjetas rojas no puede ser menor que el actual.');
```

END IF;

```

IF :NEW.porterias_imbatidas < :OLD.porterias_imbatidas THEN
    RAISE_APPLICATION_ERROR(-20005,
        'El número de porterías imbatidas no puede ser menor que el actual.');
```

END IF;

```

IF :NEW.minutos_jugados < :OLD.minutos_jugados THEN
    RAISE_APPLICATION_ERROR(-20006,
        'Los minutos jugados no pueden ser menores que los actuales.');
```

END IF;

```

IF :NEW.apariciones < :OLD.apariciones THEN
    RAISE_APPLICATION_ERROR(-20007,
        'El número de apariciones no puede ser menor que el actual.');
```

END IF;

END;

4.7.2 Resultados del Trigger: Validar estadísticas jugadores

```

Error starting at line : 3 in command -
UPDATE jugadores
SET goles = 1 --4 Este valor es menor que el valor actual (13)
WHERE id_jugador = 5
Error at Command Line : 5 Column : 20
Error report -
SQL Error: ORA-20001: El número de goles no puede ser menor que el actual.
ORA-06512: at "ADMIN.VALIDAR_ESTADISTICAS_JUGADORES", line 3
ORA-04088: error during execution of trigger 'ADMIN.VALIDAR_ESTADISTICAS_JUGADORES'

Error starting at line : 3 in command -
UPDATE jugadores
SET asistencias = 3 --4 Este valor es menor que el valor actual (13)
WHERE id_jugador = 5
Error at Command Line : 5 Column : 20
Error report -
SQL Error: ORA-20002: El número de asistencias no puede ser menor que el actual.
ORA-06512: at "ADMIN.VALIDAR_ESTADISTICAS_JUGADORES", line 7
ORA-04088: error during execution of trigger 'ADMIN.VALIDAR_ESTADISTICAS_JUGADORES'
```

Figure 13: Trigger: Validar estadísticas jugadores

4.8 Código SQL + Resultados: Funciones

Para el presente proyecto, se han desarrollado dos funciones en SQL que permiten obtener estadísticas sobre los jugadores según su edad. Estas funciones son:

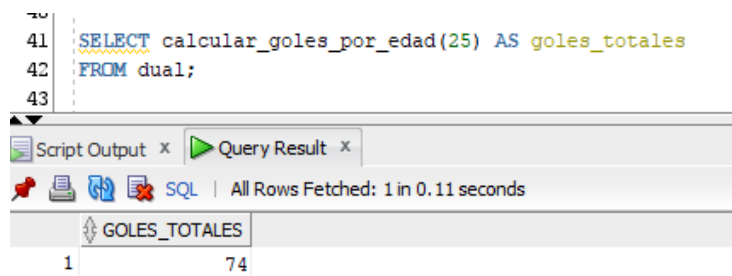
4.8.1 Función para Calcular Goles por Edad

Esta función recibe como parámetro p-edad, que representa la edad de los jugadores. Calcula y devuelve la suma total de los goles anotados por todos los jugadores que tienen esa edad. Si no hay jugadores de esa edad, devolverá 0.

```
CREATE OR REPLACE FUNCTION calcular_goles_por_edad(p_edad INT)
RETURN INT
IS
    total_goles INT;
BEGIN
    SELECT NVL(SUM(goles), 0)
    INTO total_goles
    FROM jugadores
    WHERE edad = p_edad;

    RETURN total_goles;
END;
/
```

4.8.2 Resultados de la Función para Calcular Goles por Edad



The screenshot shows a SQL IDE interface. The top pane contains the following SQL query:

```
41 SELECT calcular_goles_por_edad(25) AS goles_totales
42 FROM dual;
43
```

The bottom pane shows the query result. It has a tab labeled "Query Result" and a status bar indicating "All Rows Fetched: 1 in 0.11 seconds". The result is displayed in a table with one column named "GOLES_TOTALES" and one row with the value 74.

GOLES_TOTALES
74

Figure 14: Resultados de la Función para Calcular Goles por Edad

4.8.3 Función para Calcular Minutos Jugados por Edad

Esta función también recibe como parámetro p-edad. Calcula y devuelve la suma total de los minutos jugados por todos los jugadores que tienen esa edad. Si no hay jugadores de esa edad, devolverá 0.

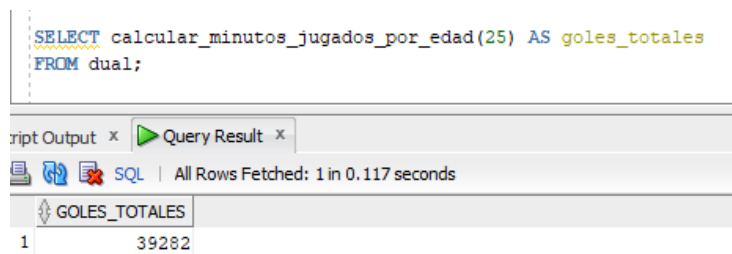
```

CREATE OR REPLACE FUNCTION calcular_minutos_jugados_por_edad(p_edad INT)
RETURN INT
IS
    total_minutos INT;
BEGIN
    SELECT NVL(SUM(minutos_jugados), 0)
    INTO total_minutos
    FROM jugadores
    WHERE edad = p_edad;

    RETURN total_minutos;
END;
/

```

4.8.4 Resultados de la Función para Calcular Minutos Jugados por Edad



The screenshot shows a SQL query editor with the following query:

```
SELECT calcular_minutos_jugados_por_edad(25) AS goles_totales
FROM dual;
```

Below the query, the 'Query Result' tab is active, displaying the following result:

GOLES_TOTALES
39282

Figure 15: Resultados de la Función para Calcular Minutos Jugados por Edad

4.9 Código SQL + Resultados: procedimientos almacenados

4.9.1 Procedimiento actualizar-valor-mercado-jugadores

Para este proyecto se creo el procedimiento actualizar-valor-mercado-jugadores, este procedimiento almacenado ha sido diseñado específicamente para el proyecto que busca establecer un modelo de predicción del precio de mercado de futbolistas. Dado que el valor de mercado de un jugador puede fluctuar en función de su rendimiento en la cancha, es fundamental contar con una herramienta que permita actualizar estos valores de manera efectiva y automática. El procedimiento permite ajustar el valor de mercado de todos los jugadores de un club, utilizando factores que pueden ser modificados según el rendimiento observado en cada temporada.

```

CREATE OR REPLACE PROCEDURE actualizar_valor_mercado_jugadores(
    club_id IN INT,
    factor_goles IN FLOAT,

```



```

        factor_asistencias IN FLOAT,
        factor_apariciones IN FLOAT
    )
IS
BEGIN
    -- Actualizar el valor de mercado de los jugadores en función de su rendimiento
    UPDATE jugadores
    SET valor_mercado = valor_mercado * (1 + (goles * factor_goles)
    + (asistencias * factor_asistencias) + (apariciones * factor_apariciones))
    WHERE id_club = club_id;
END;
/

```

4.9.2 Contexto del Procedimiento

En el contexto de este proyecto, que se centra en la estimación del valor de mercado de los futbolistas, es crucial tener un método que permita actualizar el valor de los jugadores en función de métricas relevantes como los goles anotados, las asistencias y las apariciones en partidos. El modelo de predicción de precios de mercado que se desarrollará requerirá un conjunto de datos actualizado que refleje el rendimiento actual de cada jugador. Este procedimiento permite que esas actualizaciones se realicen de forma dinámica, adaptándose a los cambios en las estadísticas de los jugadores.

4.9.3 Parámetros del Procedimiento

- **club_id:** Este parámetro especifica a qué club pertenecen los jugadores cuyos valores de mercado se desean actualizar. Esto permite una actualización selectiva basada en el club.
- **factor_goles, factor_asistencias, factor_apariciones:** Estos factores son multiplicadores que reflejan el impacto de cada estadística en el valor de mercado. Por ejemplo, un **factor_goles** de 0.02 sugiere que por cada gol anotado, el valor de mercado del jugador se incrementará en un 2%. Es importante mencionar que estos valores son hipotéticos y se utilizan como ejemplos.

4.9.4 Importancia de los Parámetros

Estos factores serán ajustados y calibrados en función del modelo de predicción que se desarrolle en el futuro. Es posible que, tras realizar un análisis estadístico o entrenar un modelo de machine learning, se determinen los valores óptimos para cada uno de estos factores.

Por lo tanto, el procedimiento es flexible y permite que, a medida que se ajusten los modelos predictivos, los parámetros utilizados para calcular el valor de mercado se modifiquen sin necesidad de reescribir el procedimiento.

4.9.5 Ejecución y resultados procedimiento

```

55 DECLARE
56     v_club_id NUMBER;
57     v_factor_goles NUMBER := 0.02; -- Por cada gol, aumenta el 2%
58     v_factor_asistencias NUMBER := 0.01; -- Por cada asistencia, aumenta el 1%
59     v_factor_apariciones NUMBER := 0.005; -- Por cada aparición, aumenta el 0.5%
60 BEGIN
61     SELECT id_club INTO v_club_id FROM jugadores WHERE id_jugador = 333;
62     actualizar_valor_mercado_jugadores(v_club_id, v_factor_goles, v_factor_asistencias, v_factor_apariciones);
63     COMMIT;
64 END;
65
66
67

```

Script Output x Query Result x

Task completed in 0.474 seconds

PL/SQL procedure successfully completed.

Figure 16: Ejecución procedimiento actualizar-valor-mercado-jugadores

Script Output x Query Result x

All Rows Fetched: 1 in 0.1 seconds

APellidos	EDAD	ID_POSICION	ID_CLUB	MINUTOS_JUGADOS	PAS	APARICIONES	GOLES	ASISTENCIAS	PORTERIAS_IMBATIDAS	AMARILLAS	ROJAS	VALOR_MERCADO
alah	30	4	12	3262	Egipto	38	22	6	21	1	0	256.5

Figure 17: Resultados procedimiento actualizar-valor-mercado-jugadores

4.9.6 Procedimiento actualizar-estadisticas-clubes

También se creó el procedimiento almacenado actualizar-estadisticas-clubes, el cual se encarga de actualizar las estadísticas de los clubes en función de los datos de los jugadores. Este procedimiento suma los goles, asistencias, tarjetas amarillas, tarjetas rojas y porterías imbatidas de todos los jugadores de cada club, asegurando así que las estadísticas se mantengan actualizadas y reflejen el rendimiento actual de cada equipo.

```

CREATE OR REPLACE PROCEDURE actualizar_estadisticas_clubes AS
    CURSOR c_jugadores IS
        SELECT
            id_club,
            SUM(goles) AS total_goles,
            SUM(asistencias) AS total_asistencias,
            SUM(amarillas) AS total_amarillas,
            SUM(rojas) AS total_rojas,
            SUM(porterias_imbatidas) AS total_porterias_imbatidas
        FROM
            jugadores
        GROUP BY
            id_club;

    v_total_goles NUMBER;
    v_total_asistencias NUMBER;

```

```

v_total_amarillas NUMBER;
v_total_rojas NUMBER;
v_total_porterias_imbatidas NUMBER;
BEGIN
  FOR r IN c_jugadores LOOP
    v_total_goles := NVL(r.total_goles, 0);
    v_total_asistencias := NVL(r.total_asistencias, 0);
    v_total_amarillas := NVL(r.total_amarillas, 0);
    v_total_rojas := NVL(r.total_rojas, 0);
    v_total_porterias_imbatidas := NVL(r.total_porterias_imbatidas, 0);

    -- Actualizar la tabla clubes
    UPDATE clubes
    SET
      Total_goles = v_total_goles,
      Total_asistencias = v_total_asistencias,
      Total_amarillas = v_total_amarillas,
      Total_rojas = v_total_rojas,
      Total_porterias_imbatidas = v_total_porterias_imbatidas
    WHERE id_club = r.id_club;
  END LOOP;
END;
```

4.9.7 Funcionamiento

- **Cursor para la Selección de Datos:** Se define un cursor c-jugadores que selecciona el id-club y suma las estadísticas relevantes de los jugadores agrupadas por club. Se utiliza la función SUM() para obtener los totales de cada estadística.
- **Actualización de Estadísticas:** Se inicializan variables para almacenar los totales calculados de goles, asistencias, tarjetas amarillas, tarjetas rojas y porterías imbatidas. Se recorre cada fila del cursor y se asignan los totales a las variables correspondientes, usando NVL() para manejar posibles valores nulos. Se ejecuta un UPDATE sobre la tabla clubes para reflejar las nuevas estadísticas calculadas.

4.9.8 Ejecución y resultados procedimiento

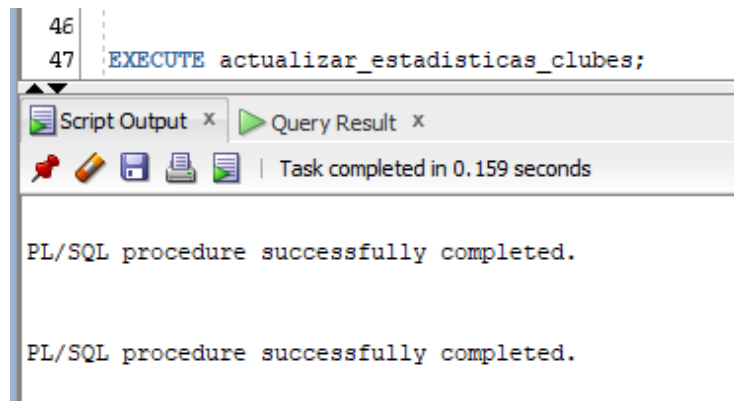


Figure 18: Ejecución procedimiento actualizar-estadisticas-clubes

ID...	NOMBRE_CLUB	TOTAL_GOLES	TOTAL_ASISTENCIAS	TOTAL_AMARILLAS	TOTAL_ROJAS	TOTAL_PORTERIAS_IMBATIDAS
1	1 AFC Bournemouth	55	43	61	1	140
2	2 Arsenal	69	52	75	2	112
3	3 Brighton Hove Albion	35	24	65	4	92
4	4 Burnley	43	32	76	1	100
5	5 Cardiff City	33	20	67	1	138
6	6 Chelsea	61	52	49	0	223
7	7 Crystal Palace	48	33	61	2	156
8	8 Everton	53	34	60	4	196
9	9 Fulham	33	24	72	2	69

Figure 19: Resultados procedimiento actualizar-estadisticas-clubes

5 Bases de Datos No-SQL

5.1 Diagrama Bases de Datos No-SQ

A continuación se presentan digramas de los atributos:

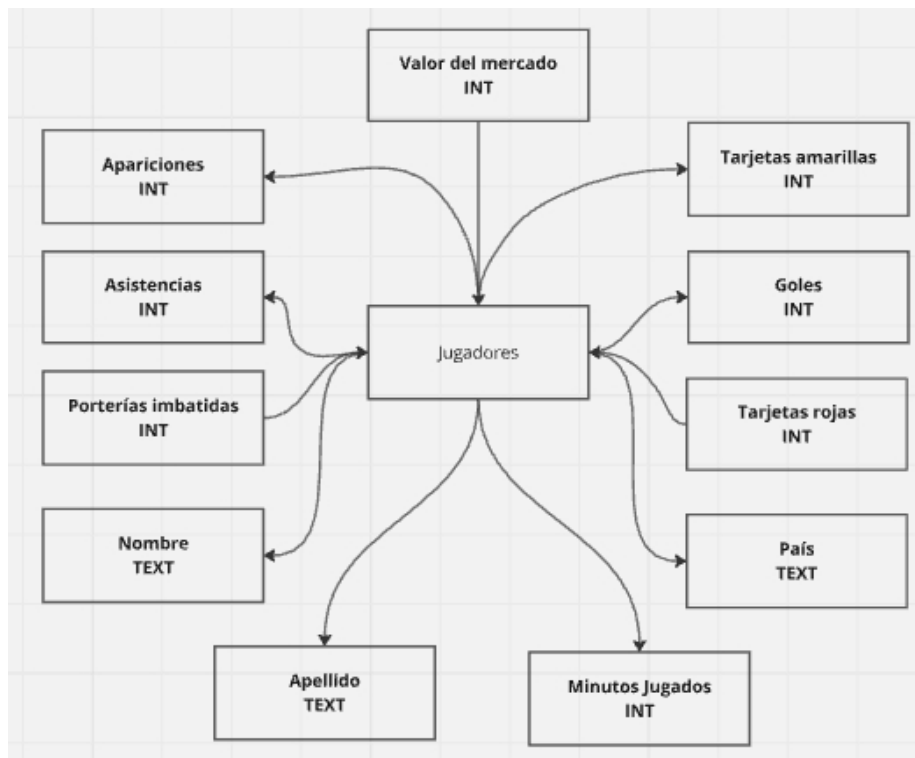


Figure 20: Diagrama atributo jugadores

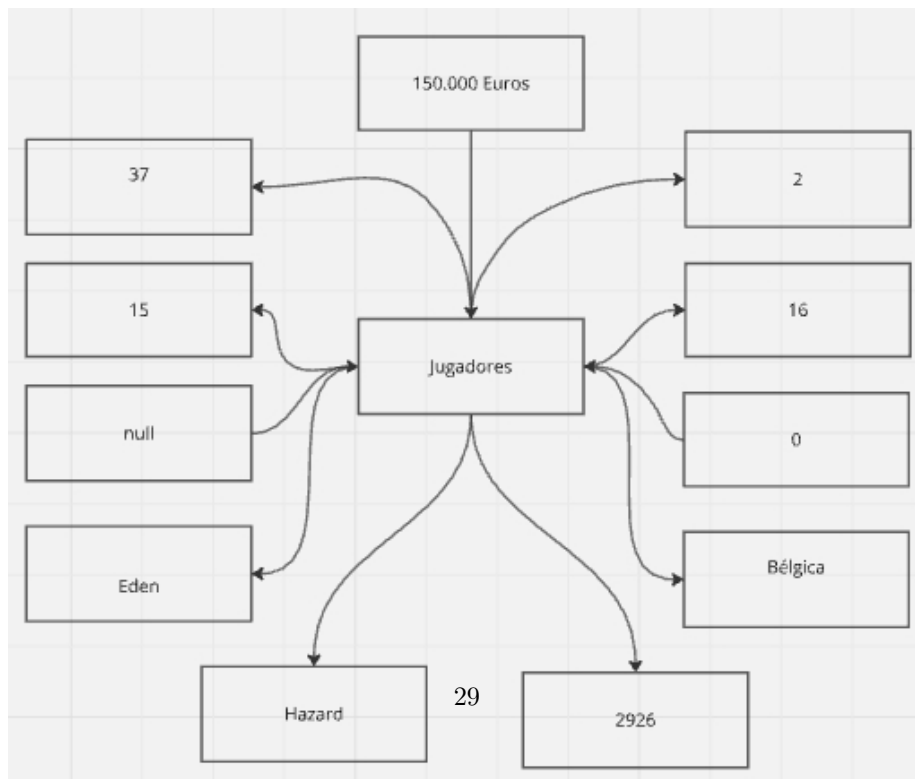


Figure 21: Diagrama atributo jugadores

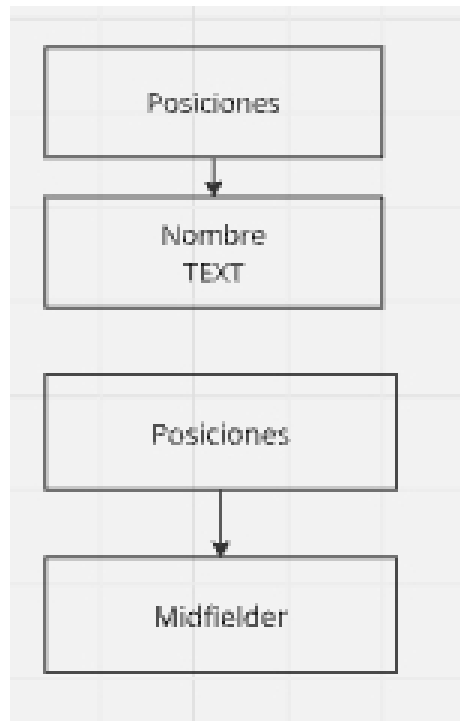


Figure 22: Diagrama atributo posiciones

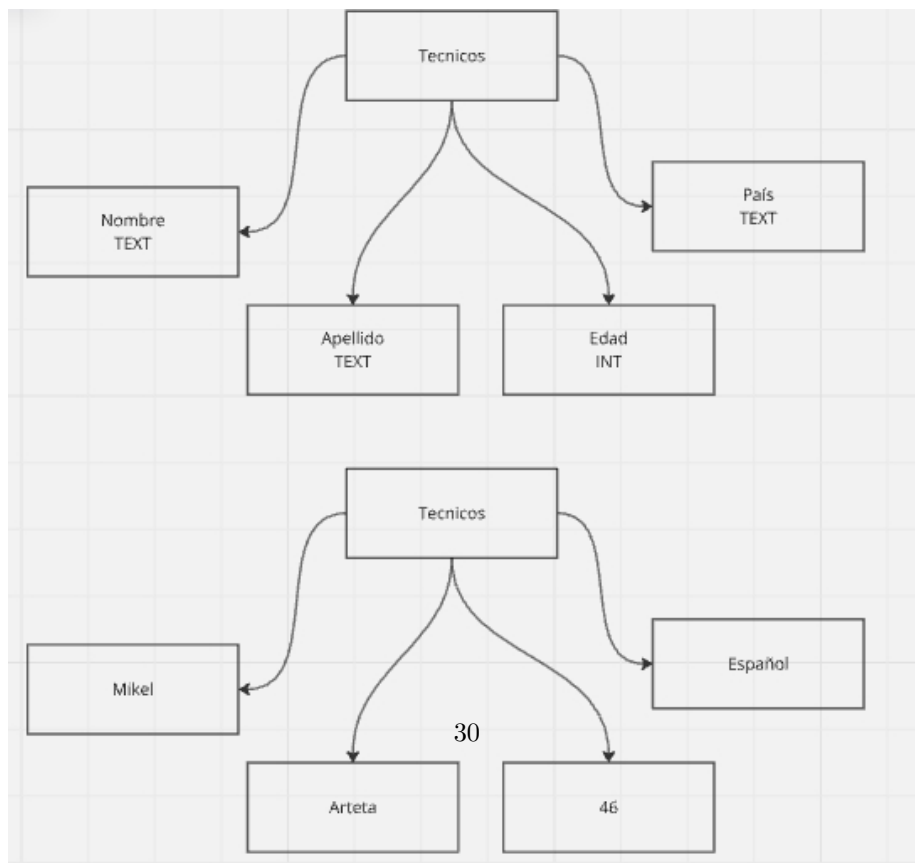


Figure 23: Diagrama atributo técnicos

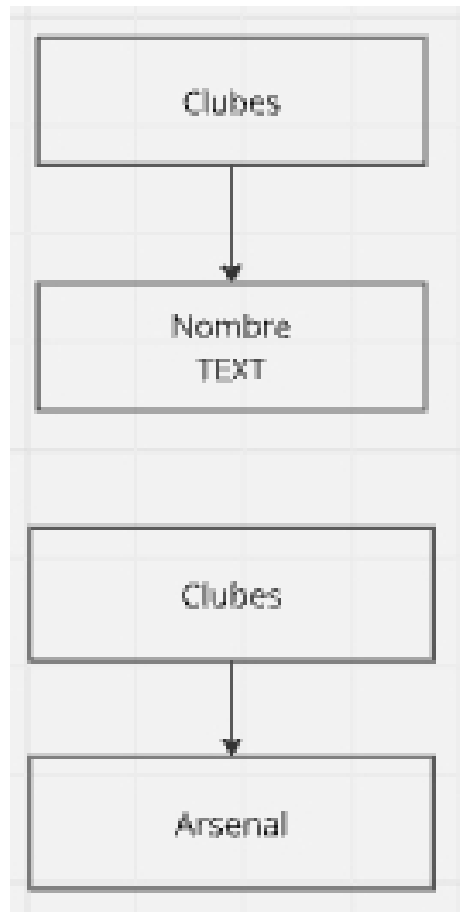


Figure 24: Diagrama atributo clubes

5.2 SMBD utilizado para la Base de Datos No-SQL

En este proyecto, se ha utilizado MongoDB como Sistema de Gestión de Base de Datos No Relacional (No-SQL), una de las soluciones más populares en el ecosistema de bases de datos No-SQL debido a su flexibilidad y alto rendimiento. A diferencia de las bases de datos relacionales tradicionales (SQL), MongoDB se basa en un modelo de datos orientado a documentos, lo que permite almacenar la información en formato JSON-like (BSON), una estructura mucho más flexible que las tablas relacionales.

5.2.1 ¿Por qué MongoDB para este proyecto?

- **Escalabilidad Horizontal:** MongoDB permite escalar de manera eficiente, lo cual es esencial para proyectos con grandes volúmenes de datos o en los que se espera un crecimiento rápido. Gracias a su arquitectura distribuida, MongoDB facilita el manejo de datos a través de múltiples servidores, lo que asegura alta disponibilidad y un balance de carga adecuado.
- **Flexibilidad en el Modelo de Datos:** MongoDB no requiere un esquema rígido y permite almacenar datos semi-estructurados. Esto significa que se pueden modificar los datos sin necesidad de realizar cambios estructurales en toda la base de datos, lo cual es ventajoso para proyectos con requisitos dinámicos o en evolución.
- **Alto Rendimiento:** MongoDB está diseñado para ofrecer un rendimiento rápido en operaciones de lectura y escritura, lo cual es fundamental para aplicaciones que manejan grandes volúmenes de datos o requieren procesamiento en tiempo real, como aplicaciones web, analítica de datos o IoT.
- **Consultas y Agregaciones Avanzadas:** MongoDB proporciona herramientas avanzadas de consulta y agregación, lo que permite realizar búsquedas complejas de manera eficiente. Gracias a su arquitectura orientada a documentos, es posible realizar operaciones de agregación sin necesidad de un esquema fijo, lo que optimiza el manejo de los datos.
- **Compatibilidad con la Nube:** MongoDB, a través de su servicio MongoDB Atlas, permite una integración sencilla con plataformas en la nube. Esto facilita la administración de la base de datos, la escalabilidad automática y la alta disponibilidad, todo ello sin la necesidad de gestionar servidores físicos o virtuales.

Con todo esto, MongoDB se presenta como una opción óptima para este proyecto, permitiendo un manejo eficiente de grandes volúmenes de datos, un alto rendimiento en las operaciones y una flexibilidad que facilita la evolución y adaptación de la estructura de la base de datos según los requerimientos del proyecto.

En este proyecto, para interactuar con la base de datos MongoDB desde Python, se utilizó la librería `pymongo`, la cual es la interfaz oficial de MongoDB

para trabajar con bases de datos en este lenguaje. A continuación, se describen los pasos realizados a través del siguiente código.

5.2.2 Explicación del código:

- **Instalación de las librerías necesarias**

El primer paso consiste en instalar las librerías necesarias para interactuar con MongoDB. El código utiliza los siguientes comandos para instalar las dependencias de `pymongo` y habilitar las conexiones a MongoDB Atlas a través de la opción `srv`:

```
!pip install pymongo
!pip install "pymongo[srv]"
```

- **Conexión a MongoDB:**

Se crea una instancia del cliente MongoDB utilizando el URI de conexión proporcionado por MongoDB Atlas. Este URI incluye la información necesaria para acceder a la base de datos, como el nombre de usuario, la contraseña y el clúster de MongoDB:

```
uri = "mongodb+srv://rgam:yjas1999@rgam.uuuvk.mongodb.net/?retryWrites=true&w=majori
client = MongoClient(uri, server_api=ServerApi('1'))
```

- **Verificación de la conexión:**

Para asegurarse de que la conexión a MongoDB se ha establecido correctamente, se envía un comando `ping` al servidor. Si la conexión es exitosa, se imprime un mensaje de confirmación. Si hay algún error, se captura la excepción y se muestra el error:

```
try:
    client.admin.command('ping')
    print("Pinged your deployment. You successfully connected to MongoDB!")
except Exception as e:
    print(e)
```

- **Creación y selección de la base de datos:**

Una vez establecida la conexión, se accede a la base de datos específica dentro del clúster de MongoDB. En este caso, se selecciona la base de datos llamada `observatorio`:

```
db = client["observatorio"]
```

- **Trabajo con archivos de Excel:**

El siguiente paso es trabajar con un archivo de datos. En este caso, se lee un archivo Excel que contiene estadísticas de jugadores de la Premier League de Inglaterra durante la temporada 2018-2019. Para ello, se utiliza la librería `pandas`, que facilita la manipulación y análisis de datos en Python.

```
!pip install gspread pandas oauth2client  
df = pd.read_excel("england-premier-league-players-2018-to-2019-stats.xlsx")
```

- **Cambio de directorio:**

Finalmente, se cambia el directorio de trabajo a una ubicación específica en el sistema de archivos para garantizar que el archivo de Excel pueda ser leído correctamente:

```
%cd "/content/drive/MyDrive/"
```

Este código establece una conexión con MongoDB a través de MongoDB Atlas utilizando `pymongo`. Luego, crea o accede a una base de datos específica (`observatorio`), y finalmente interactúa con datos almacenados en un archivo de Excel. Esta integración de MongoDB con Python y `pandas` permite realizar un análisis de datos flexible y eficiente, aprovechando tanto la base de datos No-SQL como las poderosas herramientas de procesamiento de datos en Python.

6 Aplicación de ETL (Extract, Transform, Load) y Bodega de Datos

6.1 Ejemplo de aplicación de ETL y Bodega de Datos

ETL

- **Zona de Extracción:**

- **File Reader** para Estadísticas.
- **File Reader** para Precios.

Conecta ambos nodos a la zona de transformación.

- **Zona de Transformación:**

- **String Manipulation** para limpiar nombres de jugadores.
- **Rule Engine** para categorizar jugadores por rango de precio.
- **Joiner** para unir las dos bases de datos (por nombre y equipo).

- **Zona de Carga:**

- **Database Writer** o **Excel Writer** para guardar los datos finales.

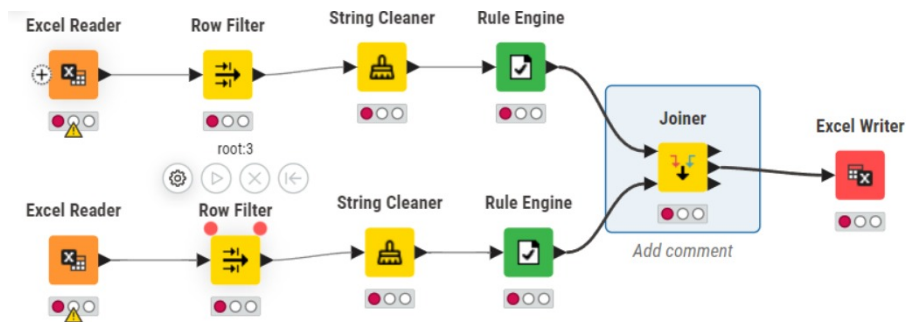


Figure 25: Tabla Jugadores en la base de datos

6.2 Automatización de Datos

Automatización de la Extracción: Si los archivos Excel se encuentran en una ubicación común o en un sistema de almacenamiento compartido, la automatización puede consistir en crear un script que cargue estos

archivos automáticamente a intervalos regulares. Esto podría implicar el uso de bibliotecas de Python como **pandas** para leer archivos Excel (`pd.read_excel()`).

Automatización de la Transformación: Este paso puede ser automatizado con un script que realice las operaciones de limpieza, transformación y combinación de datos. En Python, se pueden usar herramientas como **pandas** para manipular y transformar datos de manera eficiente.

Automatización de la Carga: Los datos transformados se pueden cargar automáticamente en una ubicación centralizada, ya sea una base de datos o una hoja de Excel para la posterior consulta y análisis. También se pueden automatizar las actualizaciones regulares de estos archivos si se incorporan datos nuevos de la temporada o si los precios cambian.

Automatización: Dependiendo de la complejidad, se pueden usar herramientas como Apache Airflow para programar y gestionar pipelines ETL complejos, o herramientas más sencillas como cron jobs en Linux para ejecutar scripts Python a intervalos regulares.

En resumen, la automatización del proceso ETL en este caso implica extraer los datos de los dos conjuntos de Excel, transformarlos para que estén listos para el análisis conjunto y cargarlos en un formato adecuado para su uso posterior. Este flujo puede ser completamente automatizado mediante scripts y herramientas adecuadas, lo que mejora la eficiencia, la precisión y la capacidad de manejar grandes volúmenes de datos en proyectos más complejos.

6.3 Integración de Datos

En este proyecto, se integraron dos conjuntos de datos relacionados con los jugadores de la Premier League 2018-2019. El primer conjunto contiene información sobre los precios de los jugadores, mientras que el segundo se enfoca en sus estadísticas de rendimiento. La integración de estos datasets tiene como objetivo proporcionar una visión más completa del valor de los jugadores en función de su rendimiento, lo que facilita un análisis más profundo sobre su relación precio-rendimiento.

Durante el proceso de integración, se presentaron algunos desafíos clave relacionados con la calidad y la consistencia de los datos:

- **Discrepancias en los Nombres de los Jugadores:** Los nombres de los jugadores no siempre coincidían exactamente entre ambos datasets. Esto se debió a variaciones en el formato (por ejemplo, "Mohamed Salah" vs. "Salah, Mohamed"). Para abordar este desafío, se implementaron transformaciones para estandarizar los nombres, asegurando que los datos de ambos conjuntos coincidieran correctamente y se pudieran unir de forma precisa.

- **Formatos de Datos Diferentes:** Los precios de los jugadores estaban representados en diferentes monedas (libras esterlinas, euros, etc.), mientras que las estadísticas de rendimiento eran presentadas de manera uniforme en el segundo conjunto de datos. Para resolver esto, se procedió a la normalización de los precios, unificando las unidades y facilitando la comparación con las estadísticas.

TECNICAS DE INTEGRACION

- **Unificación por Clave Común:** La clave común para integrar ambos conjuntos de datos fue el **nombre del jugador**. Utilizando esta clave, se llevó a cabo una combinación precisa de los dos datasets, uniendo la información sobre precios y estadísticas de rendimiento en un solo conjunto de datos.
- **Combinación de Datos:** Después de estandarizar los nombres de los jugadores y normalizar los precios, se procedió a la combinación de los dos conjuntos de datos mediante una operación de **join** en SQL. Esto permitió crear un dataset completo que incluye tanto el valor de mercado de cada jugador como sus estadísticas de rendimiento, como goles, asistencias y minutos jugados.
- **Normalización:** Se realizó la normalización de los precios para asegurar que todos los valores estuvieran en la misma moneda y, de ser necesario, también se normalizaron otras métricas (por ejemplo, estadísticas por partido) para hacer los datos comparables entre los jugadores.

La integración de datos de precios y estadísticas de los jugadores de la Premier League 2018-2019 ha sido exitosa, a pesar de los desafíos iniciales relacionados con discrepancias en los nombres y formatos de los datos. Gracias a las técnicas utilizadas, como la unificación por clave común, la combinación de datos y la normalización, hemos logrado crear un conjunto de datos cohesivo que ofrece una visión integral del rendimiento y valor de los jugadores. Este conjunto integrado de datos ahora facilita análisis más completos y precisos, que pueden ser utilizados para evaluar la relación entre el precio de los jugadores y su rendimiento durante la temporada. La integración realizada a través de SQL garantiza la consistencia y fiabilidad de los datos para futuros análisis y decisiones informadas.

7 Proximos pasos

Con la base de datos relacional diseñada y los diagramas que estructuran las relaciones entre equipos, jugadores y su rendimiento, el siguiente paso es aprovechar esta infraestructura para desarrollar modelos de machine learning que permitan estimar el valor de mercado de los futbolistas. Este proceso implicará seleccionar y limpiar los datos, identificar las variables clave que más influyen en el valor de un jugador (como edad, rendimiento en partidos recientes, posición en el campo, entre otros), y construir algoritmos predictivos que integren estos factores. La aplicación de estos modelos no solo aportará insights valiosos para los clubes al momento de fichar o vender jugadores, sino que también contribuirá a reducir la incertidumbre en las decisiones económicas relacionadas con el mercado futbolístico.

Otro paso fundamental será escalar la base de datos y los modelos desarrollados más allá de la Premier League, ampliándolos a otras ligas de fútbol a nivel global. Este escalamiento requerirá adaptar el modelo relacional para incluir características específicas de cada competición, como reglas, estilo de juego y diferencias de mercado. Asimismo, será necesario incorporar fuentes de datos adicionales que permitan homogeneizar y comparar los parámetros entre ligas. Este enfoque global permitirá una visión más completa y universal del valor de los jugadores, brindando herramientas predictivas que puedan aplicarse en contextos diversos y enriqueciendo la utilidad de la base de datos para la industria del fútbol en su conjunto.

8 Lecciones aprendidas

Para empezar queremos dar un agradecimiento al docente Wilmer Mesias Lopez Lopez. Por su dedicación en la materia y por cada una de las lecciones dadas la cual nos ayuda a conocer y explorar las diferentes herramientas que nos permitan manejar un conjunto de bases de datos y la importancia que tiene para emplearlas en el mercado, mostrándonos los beneficios de cada una de ellas. Posteriormente queremos dar un breve repaso de lo que hemos aprendido en el transcurso de la elaboración de este proyecto y cómo se relaciona con cada uno de los temas propuestos en cada una de las clases.

1. Introducción a las Bases de Datos

Fundamentos Sólidos: Adquirir una comprensión clara de las bases de datos y sus modelos de datos es crucial. Estos conceptos forman la base para el aprendizaje y la implementación de sistemas más complejos.

2. Modelo Relacional y Principios de Normalización

Normalización de Datos: Aprender los principios de normalización es esencial para diseñar bases de datos eficientes. La normalización ayuda a minimizar la redundancia de datos y mejora la integridad de la base de datos.

3. Modelo Entidad-Relación (ER)

Modelado ER: Comprender y aplicar el modelado entidad-relación facilita la creación de bases de datos robustas. Este enfoque ayuda a definir claramente las relaciones y atributos de las entidades en un sistema de datos.

4. Trabajo Práctico de Modelado ER

Aplicación Práctica: Realizar ejercicios prácticos de creación de modelos ER refuerza los conceptos teóricos y mejora las habilidades de modelado.

5. Álgebra y Cálculo Relacional

Operaciones Básicas: Conocer las operaciones básicas del álgebra y el cálculo relacional es fundamental para manipular y consultar datos en un sistema de base de datos relacional.

6. Sistemas Gestores de Bases de Datos (SGBD)

Estructura de los SGBD: Aprender sobre las funciones, componentes y ejemplos de SGBD proporciona una visión integral de cómo funcionan estos sistemas y cómo se gestionan los datos.

7. Lenguaje Estructurado de Consulta (SQL)

Dominio de SQL: Conocer y utilizar SQL es vital para definir, manipular y consultar bases de datos. SQL es la herramienta principal para interactuar con bases de datos relacionales.

8. Del Modelo Conceptual al Modelo Físico

Implementación de Modelos: Transformar modelos conceptuales en

modelos físicos es un paso crucial en la implementación de bases de datos en SGBD. Este proceso garantiza que los diseños abstractos se conviertan en estructuras funcionales.

9. Trabajo Práctico con SQL

Experiencia Práctica: La práctica de crear bases de datos utilizando SQL permite aplicar los conocimientos adquiridos y desarrollar competencia técnica en la gestión de datos.

10. Introducción a No-SQL

Modelos No-SQL: Comprender los diferentes modelos de bases de datos No-SQL y sus aplicaciones amplía las opciones disponibles para manejar grandes volúmenes de datos y necesidades específicas de almacenamiento.

11. Tipos de Bases No-SQL

Diversidad de Modelos: Apropiar conocimientos sobre Key-/Value-Stores, Document Databases y Column-Oriented Databases enriquece la capacidad de seleccionar la mejor solución de almacenamiento para distintos escenarios.

12. Bases de Datos Documentales

Profundización en No-SQL: Explorar las bases de datos documentales implementadas en SGBD permite entender cómo estos sistemas manejan datos no estructurados y semi-estructurados.

13. Trabajo Práctico con No-SQL

Implementación No-SQL: Conocer la implementación de bases de datos No-SQL a través de talleres prácticos solidifica el aprendizaje y la habilidad para gestionar sistemas No-SQL.

14. Integración de Datos

Integración de Sistemas: Apropiar conceptos de integración de bases de datos, incluyendo OLTP y OLAP, mejora la capacidad de diseñar sistemas de información coherentes y eficaces.

15. Bodegas de Datos y Procesos ETL

Conocimientos de DataWarehousing: Entender los principios de DataWarehousing y los procesos ETL es fundamental para manejar grandes volúmenes de datos y preparar datos para análisis complejos.

16. Implementación de ETL

Herramientas ETL: Usar herramientas para la implementación de procesos ETL en talleres prácticos permite poner en práctica los conocimientos adquiridos y mejorar la competencia en la preparación y carga de datos.

Conclusión

En conjunto, estas sesiones han proporcionado una comprensión exhaustiva de los sistemas de bases de datos, desde los modelos relacionales hasta los sistemas No-SQL, y los procesos de integración y manipulación de datos.

La combinación de teoría y práctica ha sido crucial para consolidar el conocimiento y las habilidades necesarias en el diseño y gestión de bases de datos, preparándonos mejor para enfrentar desafíos en el manejo de datos en diferentes contextos y aplicaciones.

9 Bibliografía

- Transfermarkt. (n.d.). Transfermarkt: The football transfer market. Retrieved [”Octubre 17, 2024”], from <https://www.transfermarkt.com/premier-league/startseite/wettbewerb/GB1>
- FootyStats. (n.d.). FootyStats: Football statistics and analysis. Retrieved [”Octubre 17, 2024”], from <https://footystats.org/es/england/premier-league>