



Licenciatura en Sistemas de Información

Base de datos I

Año 2022

Profesor: Sambrana, Ivan.

Grupo N °: 7.

Tema: Proyecto Mercado de pases "Transfer Market".

Integrantes:

- Acevedo, Ariel Alejandro.
- De Bortoli, Carlos Horacio.
- Escobar, Justo Daniel.
- Nuñez, Cesar Ivan.

CAPÍTULO I

Objetivos generales

Se establece como objetivo realizar el diseño y modelado de una base de datos relacional para obtener un informe de transferencias de jugadores de fútbol que ocurren entre diferentes clubes de las ligas más importantes del país en una temporada, para permitir un mejor desempeño en las consultas o búsquedas de jugadores libres o con contrato.

Alcance del Proyecto

La base de datos es para una aplicación web llamada "Transfer Market". Este sistema va dirigido a los profesionales del ámbito futbolístico como responsables de clubes, jugadores, etc. Y para aquellos que busquen información acerca de las transferencias realizadas en diferentes temporadas del fútbol argentino.

Descripción detallada del Proyecto

Se desea guardar los datos personales de los jugadores (nombre, apellido, club, nacionalidad, fecha de nacimiento, posición, altura, pie hábil, valor de mercado). Un jugador solo puede tener una posición (arquero, defensor, mediocampista, delantero).

Con respecto a los clubes se quiere saber el nombre, año de fundación, responsable del mismo y la liga a la que pertenece (primera, segunda, tercera, etc), cada club tiene un número único de identificación. Un club sólo puede ser parte de una liga.

Como información de las ligas se desea almacenar el nombre, cantidad de equipos, cantidad de fechas. Cada jugador puede tener uno o varios representantes y éstos pueden representar a más de un jugador. Del representante se desea obtener dni, nombre, apellido, teléfono y correo.

Se permite registrar una o varias transferencias con la información de la fecha, jugador, clubes que intervienen, representante, valor del mercado y estado (libre, préstamo, contrato).

Se desea emitir un informe por temporada con las diferentes transferencias de jugadores detalladas a continuación:

INFORME TRANSFERENCIAS

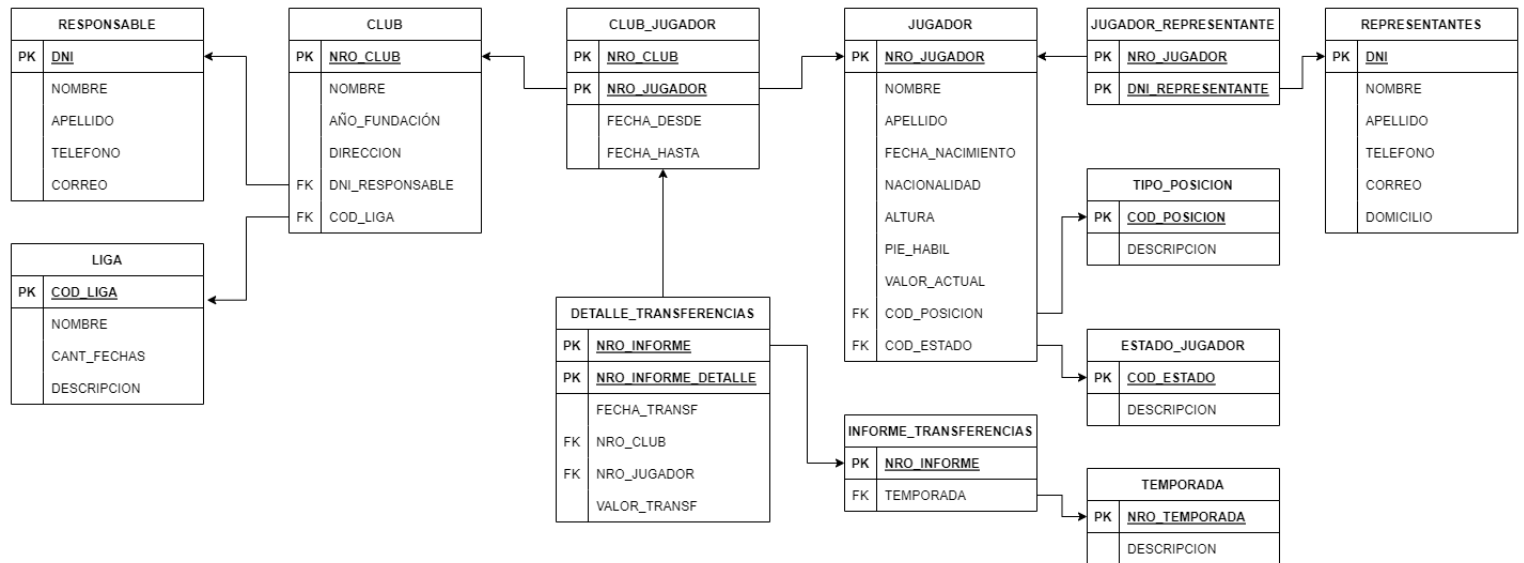
TEMPORADA 20-21

NRO INFORME: 540

FECHA	NOMBRE JUGADOR	APELLIDO JUGADOR	CLUB ORIGEN	CLUB DESTINO	REPRESENTANTE	PRECIO	ESTADO
2021-03-01	Ivan	Sambrana	Mandiyu	Lanús	Mónica González	€2M	Libre
2021-09-24	Dario	Villegas	Boca Unidos	Patronato	María Fernández	€1.5M	Préstamo
2020-11-30	Pepe	Sand	Lanús	Club Sportivo	Rubén López	\$400K	Contrato

CAPÍTULO II

Diagrama del Modelo de datos



Entidades

Descripción de tablas:

A_ TIPO POSICIÓN

COD TIPO POSICIÓN [PK]	DESCRIPCIÓN
1	2

B_ ESTADO _ JUGADOR

COD ESTADO [PK]	DESCRIPCIÓN
1	2

C_ REPRESENTANTE

DNI [PK]	NOMBRE	APELLIDO	DOMICILIO	TELÉFONO	CORREO
1	2	3	4	5	6

D_ JUGADOR

NRO_JUGADOR [PK]	NOMBRE	APELLIDO	FECHA_NAC	NACIONALIDAD	ALTURA	PIE HÁBIL
1	2	3	4	5	6	7

VALOR ACTUAL	COD ESTADO [FK]	COD TIPO POSICIÓN [FK]
8	9	10

E_ JUGADOR_REPRESENTANTE

NRO_JUGADOR [PK]	DNI_REPRESENTANTE [PK]
1	2

F_ RESPONSABLE

DNI [PK]	NOMBRE	APELLIDO	TELÉFONO	CORREO
1	2	3	4	5

G_ LIGA

COD LIGA [PK]	NOMBRE	CANT FECHAS
1	2	3

H_ CLUB

NRO CLUB [PK]	NOMBRE	AÑO FUND	DIRECCIÓN	COD LIGA [FK]	DNI RESP [FK]
1	2	3	4	5	6

I_ CLUB _ JUGADOR

NRO CLUB [PK]	NRO_JUGADOR [PK]	FECHA DESDE	FECHA HASTA
1	2	3	4

J_ TEMPORADA

NRO TEMPORADA [PK]	DESCRIPCIÓN
1	2

K_ INF TRANSF CABECERA

NRO INFORME [PK]	NRO TEMPORADA [FK]
1	2

L_ INF TRANSF DETALLE

NRO INFORME [PK]	NRO INFORME DETALLE [PK]	FECHA TRANSF	NRO CLUB [FK]	NRO JUGADOR [FK]	VALOR TRANSF
1	2	3	4	5	6

RELACIONES:

1) L (1) → K (1)	5) H (5) → G (1)	9) D (10) → A (1)
2) K (2) → J (1)	6) H (6) → F (1)	10) E (1) → D (1)
3) L (4 + 5) → I (1 + 2)	7) I (2) → D (1)	11) E (2) → C (1)
4) I (1) → H (1)	8) D (9) → B (1)	

Diccionario de datos

TABLA “TIPO POSICIÓN”

CLAVE/LLAVE	NOMBRE	TIPO DE DATO	TAMAÑO	DESCRIPCIÓN
Clave Primaria [PK]	cod_tipo_posicion	Int (Identity)	11	Clave única de tipo de posición.
	descripción	Varchar	20	Descripción del tipo de posición de un jugador.

TABLA “ESTADO JUGADOR”

CLAVE/LLAVE	NOMBRE	TIPO DE DATO	TAMAÑO	DESCRIPCIÓN
Clave Primaria [PK]	cod_estado	Int (Identity)	11	Clave única de estado.
	descripción	Varchar	20	Descripción del estado de un jugador.

TABLA “REPRESENTANTE”

CLAVE/LLAVE	NOMBRE	TIPO DE DATO	TAMAÑO	DESCRIPCIÓN
Clave Primaria [PK]	dni	Int	8	Clave única de representante.
	nombre	Varchar	30	Nombre del representante.
	apellido	Varchar	30	Apellido del representante.
	domicilio	Varchar	200	Domicilio del representante.
	teléfono	Varchar	11	Teléfono del representante.
	correo	Varchar	50	Correo del representante.

TABLA “LIGA”

CLAVE/LLAVE	NOMBRE	TIPO DE DATO	TAMAÑO	DESCRIPCIÓN
Clave Primaria [PK]	cod_liga	Int (Identity)	11	Clave única de la liga.
	nombre	Varchar	30	Nombre de la liga.
	cant_fechas	Int	2	Cantidad de fechas de la liga.

TABLA “TEMPORADA”

CLAVE/LLAVE	NOMBRE	TIPO DE DATO	TAMAÑO	DESCRIPCIÓN
Clave Primaria [PK]	nro_temporada	int(Identity)	11	Clave única de temporada.
	descripción	Varchar	20	Descripción de la temporada.

TABLA “JUGADOR”

CLAVE/LLAVE	NOMBRE	TIPO DE DATO	TAMAÑO	DESCRIPCIÓN
Clave Primaria [PK]	nro_jugador	Int (Identity)	11	Clave única del jugador.
	nombre	Varchar	30	Nombre del Jugador.
	apellido	Varchar	30	Apellido del jugador.
	fecha_nac	Date	-	Fecha de nacimiento del jugador.
	nacionalidad	Varchar	30	Nacionalidad del jugador.
	altura	Decimal	1,2	Altura del jugador.
	pie_hábil	Varchar	20	Pie hábil del jugador.
	valor_actual	Float	9,2	Valor actual del jugador.
Clave Foránea [FK]	cod_estado	Int (Identity)	11	Clave foránea de estado.
Clave Foránea [FK]	cod_tipo_posicion	Int (Identity)	11	Clave foránea del tipo de posición.

TABLA “JUGADOR_REPRESENTANTE”

CLAVE/LLAVE	NOMBRE	TIPO DE DATO	TAMAÑO	DESCRIPCIÓN
Clave Primaria [PK]	nro_jugador	Int (Identity)	11	Clave única de número de jugador.
Clave Primaria [PK]	dni_representante	Int	8	Clave única del dni del representante.

TABLA “RESPONSABLE”

CLAVE/LLAVE	NOMBRE	TIPO DE DATO	TAMAÑO	DESCRIPCIÓN
Clave Primaria [PK]	dni_responsable	int	8	Clave única del responsable del club.
	nombre	Varchar	30	Nombre del responsable del club.
	apellido	Varchar	30	Apellido del responsable del club.
	teléfono	int	11	Teléfono del responsable del club.
	correo	Varchar	50	Correo del responsable del club.

TABLA “CLUB”

CLAVE/LLAVE	NOMBRE	TIPO DE DATO	TAMAÑO	DESCRIPCIÓN
Clave Primaria [PK]	nro_club	Int (Identity)	11	Clave única de número del club.
	nombre	varchar	30	Nombre del club.
	año_fund	int	4	Año de fundación del club.
	dirección	varchar	200	Dirección del club.
Clave Foránea [FK]	cod_liga	Int (Identity)	11	Clave Foránea de la liga.
Clave Foránea [FK]	dni_resposable	Int (Identity)	11	Clave Foránea del responsable del club.

TABLA “CLUB_JUGADOR”

CLAVE/LLAVE	NOMBRE	TIPO DE DATO	TAMAÑO	DESCRIPCIÓN
Clave Primaria [PK]	nro_club	Int (Identity)	11	Clave única de número de club.
Clave Primaria [PK]	nro_jugador	Int (Identity)	11	Clave única del número de jugador.
	fecha_desde	Date	-	Fecha ingreso de jugador con el club
	fecha_hasta	Date	-	Fecha fin de jugador con el club

TABLA “INF TRANSF CABECERA”

CLAVE/LLAVE	NOMBRE	TIPO DE DATO	TAMAÑO	DESCRIPCIÓN
Clave Primaria [PK]	nro_informe	int(Identity)	11	Clave única de informe transferencia cabecera.
Clave Foránea [FK]	nro_temporada	Int (Identity)	11	Clave foránea del número de temporada.

TABLA “INF TRANSF DETALLE”

CLAVE/LLAVE	NOMBRE	TIPO DE DATO	TAMAÑO	DESCRIPCIÓN
Clave Primaria [PK]	nro_informe	Int (Identity)	11	Clave única del informe transferencia cabecera
Clave Primaria [PK]	nro_informe_detalle	Int (Identity)	11	Clave única del informe transferencia detalle.
	fecha_transf	Date	-	Fecha en la que se realizó la transferencia de un jugador.
Clave Foránea [FK]	nro_club	Int (Identity)	11	Clave foránea de número de club.
Clave Foránea [FK]	nro_jugador	Int (Identity)	11	Clave foránea del número de jugador
	valor_transf	Float	9,2	Valor de la transferencia del jugador.

CAPÍTULO III

Metodologías/Herramientas

Diagrams.io: Es la herramienta con la que creamos y editamos los diagramas, permitiéndonos trabajar en equipo y en línea, facilitando también la forma de guardar y editar los cambios realizados en Google Drive.

Sql Server Management Studio v2008: Es un sistema de gestión de base de datos relacional, desarrollado por la empresa Microsoft. El lenguaje de desarrollo utilizado es Transact-SQL, una implementación del estándar ANSI del lenguaje SQL, utilizado para manipular y recuperar datos, crear tablas y definir relaciones entre ellas. Decidimos utilizar este motor ya que es con el que trabajamos durante las clases prácticas y así se nos resultaba más fácil de implementar.

GitHub: Es una forja para alojar proyectos utilizando el sistema de control de versiones Git. Lo utilizamos para trabajar juntos en el desarrollo del proyecto.

CAPÍTULO IV

Temas técnicos de trabajo

Funciones y Procedimientos almacenados

Los procedimientos almacenados y las funciones son componentes de código que podemos utilizar el SQL SERVER para crear rutinas.

Existen muchas diferencias entre ambos objetos. La primera y más obvia diferencia, es que las funciones siempre retornan un valor, mientras que un procedimiento almacenado puede que retorne un valor o puede que no lo haga. Es decir que puede comportarse como un método o como una función, haciendo una analogía con el desarrollo de software.

Otra diferencia muy importante para los desarrolladores es que los procedimientos almacenados pueden ser invocados desde el entorno de desarrollo, es decir desde .NET, pero las funciones no. Así que al momento de desarrollar, siempre nos comunicaremos con procedimientos almacenados.

Las funciones de hecho no pueden ser invocadas por sí solas, mientras que los procedimientos almacenados sí. Para invocar una función, esta debe estar dentro de una instrucción.

Sin embargo, aunque pareciera que las funciones tienen ciertas desventajas sobre los procedimientos almacenados, en realidad no es así, simplemente sirven para diferentes cosas.

Uno de las grandes ventajas es que las funciones retornan valores procesables. Es cierto los stored procedures también retornan valores...pero cuando eso sucede solamente podemos procesar los valores de tipo escalar.

Pero si mi stored procedure retorna una tabla o conjunto de tablas, no es posible recuperar esos valores y usarlos para procesarlos dentro de otro código. En cambio, todo valor que retorna de una función puede ser procesado. Por ejemplo, una función puede retornar una tabla completa y puedo operar con ella.

Por definición, las funciones son ideales para retornar valores con los que deseo operar, mientras que los procedimientos almacenados suelen ser la parte de salida final del código. Es decir que el procedimiento almacenado es el que se comunica directamente con el usuario y le presenta el resultado final.

Por eso, una función no puede invocar un procedimiento almacenado, solamente puede invocar otra función. En cambio, el procedimiento almacenado, como es código final, puede invocar funciones y procedimientos almacenados.

Así que a correcta integración de un sistema que use ambos, sería utilizar procedimientos almacenados para comunicarse y enviar datos a la aplicación; mientras que las funciones se usan para procesar datos internamente dentro del stored procedure.

Disparadores

Un TRIGGER, también conocido como disparador, es una especie de script de programación SQL para base de datos. Los triggers son procedimientos que se ejecutarán según nuestras indicaciones cuando se realicen operaciones sobre la información de la base de datos. Estas operaciones pueden ser de actualización (UPDATE), inserción (INSERT) y borrado (DELETE).

Los triggers además pueden modificar la información de la base de datos e incluso detener la ejecución de consultas erróneas.

Los triggers son una de las funcionalidades más útiles de las que disponemos cuando diseñamos y mantenemos bases de datos. Gracias a ellos podremos implementar ciertas características de nuestra base de datos sin necesidad de desarrollar programación en otros lenguajes externos. Por ejemplo, duplicar en una segunda tabla toda la información que se inserte en otra. No existe ninguna forma de indicarle a la base de datos que lo realice de forma automática. Sin embargo, gracias a un trigger que se ejecute tras un INSERT, podemos insertar esa información en la segunda tabla, todo esto sin que el usuario/programador que lanzó el INSERT tenga que hacer nada.

Vistas

La principal y más básica de las practicas que nos ayudan a mantener y garantizar la integridad de la información almacenada en la Base de datos que administremos en nuestra organización, es la aplicación de Vistas con el fin de obtener la información ya sea de una o varias tablas.

Una vista es una tabla virtual cuyo contenido está definido por una consulta. Al igual que una tabla real, una vista consta de un conjunto de columnas y filas de datos con un nombre. Sin embargo, a menos que esté indexada, una vista no existe como conjunto de valores de datos almacenados en una base de datos. Las filas y las columnas de datos proceden de tablas a las que se hace referencia en la consulta que define la vista y se producen de forma dinámica cuando se hace referencia a la vista.

Una vista actúa como filtro de las tablas subyacentes a las que se hace referencia en ella. La consulta que define la vista puede provenir de una o de varias tablas, o bien de otras vistas de la base de datos actual u otras bases de datos. Así mismo, es posible utilizar las consultas distribuidas para definir vistas que utilicen datos de orígenes heterogéneos. Esto puede resultar de utilidad, por ejemplo, si desea combinar datos de estructura similar que proceden de distintos servidores, cada uno de los cuales almacena los datos para una región distinta de la organización.

No existe ninguna restricción a la hora de consultar vistas y muy pocas restricciones a la hora de modificar los datos de éstas.

En otras palabras, una vista es una ventana a través de la cual se puede consultar o cambiar información de la tabla a la que está asociada. Esto, claro está, en relación con los privilegios que posea el usuario de la base de datos. Si el usuario solamente tiene privilegios de lectura en una entidad, en la vista tampoco podrá agregar o modificar información; si el usuario no tiene acceso a determinadas tablas, tampoco podrá crear una vista con información proveniente de las mismas.

Permisos

Otro mecanismo que se aplica en las Bases de datos para garantizar la integridad de la información es la administración de permisos mediante roles para garantizar que solo los usuarios autorizados puedan modificar, eliminar e incluso ingresar la información que se encuentra en la base de datos. Es necesario definir que usuarios tienen permisos para las diferentes actividades, loque implica algún tipo de manipulación de la información almacenada.

Por lo cual es recomendable activar los registros de auditoría para las diferentes sentencias que permiten la modificación de información en la base de datos. Por ejemplo, en una institución financiera en donde la manipulación de la información puede causar grandes pérdidas tanto económicas como institucionales, para la compañía es necesario activar la auditoria de las sentencias tales como Insert, Update y Delete. De esta manera poder llevar un seguimiento y control sobre las operaciones que se realicen y/o modifiquen la integridad de los datos.

Una definición que nos puede ayudar a mantener la integridad de los datos es la utilización del concepto de integridad referencial que es: "Un sistema de reglas que utilizan la mayoría de las bases de datos relacionales para asegurarse que los registros de tablas relacionadas son válidos y que no se borren o cambien datos relacionados de forma accidental produciendo errores de integridad.

Transacciones

Las Transacciones en SQL son unidades o secuencias de trabajo realizadas de forma ordenada y separada en una base de datos. Normalmente representan cualquier cambio en la base de datos, y tiene dos objetivos principales:

Proporcionar secuencias de trabajo fiables que permitan poder recuperarse fácilmente ante errores y mantener una base de datos consistente incluso frente a fallas del sistema
Proporcionar aislamiento entre programas accediendo a la vez a la base de datos

Una Transacción es una propagación de uno o más cambios en la base de datos, ya sea cuando se crea, se modifica o se elimina un registro. En la práctica suele consistir en la agrupación de consultas SQL y su ejecución como parte de una transacción

Las transacciones siguen cuatro propiedades básicas, bajo el acrónimo ACID (Atomicity, Consistency, Isolation, Durability):

- Atomicidad: Aseguran que todas las operaciones dentro de la secuencia de trabajo se completen satisfactoriamente. si no es así, la transacción se abandona en el punto del error y las operaciones previas retroceden a su estado inicial.
- Consistencia: Aseguran que la base de datos cambie estados en una transacción exitosa.
- Aislamiento: Permiten que las operaciones sean aisladas y transparentes unas de otras.
- Durabilidad: Aseguran que el resultado o efecto de una transacción completada permanezca en caso de error del sistema

Existen Tres controles básicos de control en las transacciones SQL:

- COMMIT: Para guardar los cambios.
- ROLLBACK: Para abandonar la transacción y deshacer los cambios que se hubieran hecho en la transacción.
- SAVEPOINT: Crea Check points, puntos concretos en la transacción donde poder deshacer la transacción hasta esos puntos.

Los comandos de control de transacción se usan sólo con INSERT, DELETE y UPDATE. No pueden utilizarse creando tablas o vaciándolas porque las operaciones se guardan automáticamente en la base de datos.

CAPÍTULO V

Conclusiones / Dificultades / Perspectivas





Como dijo el profe. "Al investigar sobre los objetos denominados "procedimientos y funciones almacenados" en un motor de bases de datos descubrimos que son una herramienta muy útil para conseguir un código más eficiente al reutilizar código almacenado en el mismo motor de base de datos y que según la situación y las rutinas que deseamos implementar podemos optar por funciones o procedimientos almacenados. Este último posee la ventaja de poder ser llamado desde un entorno de desarrollo y ser ejecutado desde allí"

Con respecto a los triggers son herramientas muy potentes y útiles. Nos permiten llevar un control sobre los cambios que se producen en las tablas sin necesidad de apoyarnos en software externos, pero tiene la desventaja que si se le da un uso incorrecto puede derivar en respuestas más lentas por parte del servidor. Podemos decir con seguridad que en futuros trabajos los utilizaremos debido a su simpleza y posibilidades que nos ofrecen.

Las transacciones al igual que los triggers, nos permite un control de las tablas, sin necesidad de un software externo. En un sistema de base de datos, las transacciones permiten asegurar la integridad de los datos. permite realizar acciones sobre las bases de datos deseadas, logrando operaciones de ingreso, borrado, actualización sin poner en riesgo la integridad del mismo.

Para concluir lo visto en relación a los temas de vistas y permisos nos brindaron herramientas para garantizar la integridad de la base de datos. Por el lado de las vistas aprendimos a como acceder a los datos de las distintas tablas, como relacionarlas entre ellas y las maneras que se pueden optimizar las consultas a través de vistas indexadas. Para el caso de los permisos vimos cómo proteger el acceso a los datos por medio de la creación de usuarios y contraseñas, asignándole roles y limitando el acceso solo a la información que necesita cada usuario.

BIBLIOGRAFÍA

-  Catedra de Base de datos I de la UNNE, material de clases prácticas y teóricas, videos, documentos del aula virtual.
-  Marco Vinicio Burbano Sánchez, Pontificia Universidad Católica del Ecuador, 2010 "Guía de mejores prácticas para garantizar, seguridad, integridad y disponibilidad en bases de datos":
-  <http://repositorio.puce.edu.ec/bitstream/handle/22000/3758/T-PUCE-3805.pdf?sequence=1&isAllowed=y>
-  <https://www.srcodigofuente.es/aprender-sql/triggers-sql>

ANEXO

Script SQL

Documentación complementaria