**Sentencias Creadas Para El Trabajo**

1. **Kenzo Van Waerebeek Alfaro**

--Cree un procedimiento que muestre el campeón que ha sido elegido más veces mediante el cálculo del nro de veces que cada campeón ha sido elegido.

--Esta información es importante para saber al final de cada temporada u hoy en día cual es el campeón más popular.

create procedure maselegido

as

Select a.N1 as Nombre , b.N2 as NroElecciones

from(

select CHAMP.CHAMP as N1

from (select cu.D\_Nombre\_Champ as CHAMP, count(\*) as Num

from Campeon\_User cu

group by cu.D\_Nombre\_Champ) as CHAMP

where CHAMP.Num=

(Select max(CHAMP.Num)

from (select cu.D\_Nombre\_Champ as CHAMP, count(\*) as Num

from Campeon\_User cu

group by cu.D\_Nombre\_Champ) as CHAMP)

)a

join

(

Select max(Num) as N2

from (select cu.D\_Nombre\_Champ as CHAMP, count(\*) as Num

from Campeon\_User cu

group by cu.D\_Nombre\_Champ) as Tab

) b on 1=1

go

exec maselegido

--Cree una función que determine si un usuario específico puede comprar un skin con sus puntos en cuenta. Esta función es importante en la base de datos pues cuando un usuario quiera comprar un skin se deberá verificar que tenga la cantidad requerida de puntos para comprarla.

--\*Nota la compra se hace mediante puntos de influencia.\*

create function si\_o\_no(@usuario char(18),@server int, @skin char(18)) returns bit

as

Begin

declare @varr bit

if

(select s.M\_Precio\_En\_Puntos

from Skin s

where s.Nombre\_Skin=@skin)

>=

(select u.Q\_PtsInfluencia

from [User] u

where [u.C\_Username=@usuario](mailto:u.C_Username=@usuario) and u.c\_Server=@server)

set @varr='true'

return @varr

End

Go

print si\_o\_no('Kenzo',1,’NombreCampeonAqui’)

**o si no podríamos usar esta función**

create function si\_o\_noo3(@usuario char(18),@server int, @skin char(18)) returns table

as

return (

select u.C\_Username, u.Q\_PtsInfluencia, u.T\_Correo

from [User] u

where u.C\_Username=@usuario and u.C\_Server=@server and u.Q\_PtsInfluencia>=(select s.M\_Precio\_En\_Puntos

from Skin s

where s.Nombre\_Skin=@skin)

)

Go

--Muestre el tipo de partidas y la cantidad de partidas jugadas por tipo de manera ascendente.

--Esto sería importante para 'Riot Games', pues le permitirá saber cuál es el tipo de juego que a la gente mas le gusta. Bajo esta información decidir qué tipos de juego dejar, impulsar o eliminar del juego.

select tip.C\_IdTipo as CodTipo, count(\*) as NroPartidas

from Tipo tip join Partida p on p.C\_IdTipo=tip.C\_IdTipo

group by tip.C\_IdTipo

order by count(\*) asc

--Crear un trigger que notifique en una tabla especial los datos de los jugadores más ricos (es decir, han alcanzado el tope de pts Riot).

CREATE TABLE Ricos (Nombre char(18), Apellido CHAR (18),Correo char (18) )

create TRIGGER ricoss ON [User]

FOR UPDATE

AS

DECLARE @Temp1 VARCHAR(20),@Temp2 VARCHAR(20),@Temp3 VARCHAR(20)

SELECT @Temp1 = D\_Apellido FROM INSERTED

SELECT @Temp2 = D\_Nombre\_Usuario FROM INSERTED

SELECT @Temp3 = T\_Correo FROM INSERTED

IF (SELECT Q\_PtsRiot FROM INSERTED)=10000

BEGIN

INSERT INTO Ricos VALUES (@Temp1, @Temp2, @Temp3)

END

1. **Nora Castillo**

--Calcule el monto total de dinero obtenido por transacciones de usuarios por servidor. Este procedimiento es importante para la empresa o servidor porque significa la ganancia que está teniendo por parte de los usuarios en la compra de ítems en el juego. Esto no solo significaría el crecimiento del juego sino que tendremos más jugadores conforme pase el tiempo.

select s.D\_Nombre\_Server, SUM(t.M\_Consumo) AS MontoTransac

from [User] u join Server s on u.C\_Server=s.C\_Server

join Transaccion t on u.C\_Username=t.C\_Username

group by s.D\_Nombre\_Server

--Cada vez que se realiza un parche del juego o actualiza a una nueva versión, las páginas de maestría pueden ser reiniciadas por los usuarios, ya que el estilo de juego (por los cambios que se dan a campeones o tipos de partida) puede cambiar.

--Cree un procedimiento que actualice (reiniciando) los valores la página de maestrías de un determinado usuario pero no la borre, pues esto enojaría al usuario.

create procedure reiniciopm

@pagina char(18),

@usu char(18)

as

Begin

update PagMaestrias set Q\_Astucia=0, Q\_Ferocidad=0, Q\_Valor=0 where C\_Username=@usu

End

go

exec reiniciopm @pagina='Nombre\_Pagina', @usu='ID Usuario'

--Cree una función que devuelva la descripción y datos de una estadística en una partida específica.

--Esta función sería importante pues se puede hacer una evaluación a todas las partidas, pero si yo quiero evaluar una en específico podre utilizar esta función.

create function estadisticas(@varr char(18)) returns Table

as

Return(

select e.D\_Estadistica, e.Q\_DatoEstadistico , p.C\_IdPartida

from Partida p join Estadistica e on e.C\_IdPartida=p.C\_IdPartida

where p.C\_IdPartida=@varr

group by p.C\_IdPartida,e.D\_Estadistica, e.Q\_DatoEstadistico

)

go

Select \* from estadisticas('IDPartidaAqui')

--Cree un trigger que permita saber si una inserción de Nombre Usuario es valida... note que por cada servidor el nombre de usuario es único.

--Esto es importante porque si no habrían varios usuarios con el mismo nombre en todos los servidores y sería muy desorganizado tener una base de datos de tal forma. Asimismo, a la hora de hacer consultas, gestionar datos, o información todo el proceso sería muy tedioso.

--\*identificable en las reglas de negocio...no puede haber un usuario con el mismo nombre en un servidor\*.

CREATE TRIGGER insertausuario ON [User]

FOR INSERT

AS

IF (Select count (\*)

From INSERTED, [User] u

Where INSERTED.C\_Username = u.C\_Username and INSERTED.C\_Server=u.C\_Server) =1

BEGIN

Rollback transaction

PRINT 'Ya existe ese nombre de usuario en ese Servidor'

END

ELSE

PRINT 'El usuario fue ingresado correctamente'

1. **Ricardo Bueno**

--Seleccione las runas pertenecientes a cada página de cada jugador de la BD

--Esto es importante para saber el detalle de las páginas de runas de cada jugador y podría ayudar a ver que runas no están siendo utilizadas por los usuarios para quitarlas del juego.

select pr.C\_Username as Usuario, r.D\_NombreRuna as Runa, pr.N\_Pagina as EnPagina

from PagRuna pr join [User] u on u.C\_Username=pr.C\_Username

join Runa r on r.N\_Pagina=pr.N\_Pagina

group by pr.C\_Username, r.D\_NombreRuna, pr.N\_Pagina

--Cree un procedimiento que consulte el nombre de los skins existentes en el juego así como el nombre de campeón al que pertenecen. Este procedimiento tiene importancia, ya que al final de cada temporada se podrán haber agregado más skins...es necesario llevar la cuenta o rastreo de las skins que ya existen para cada personaje.

create procedure skinss

as

select sk.Nombre\_Skin, c.D\_Nombre\_Champ

from Skin sk join Campeon c on c.D\_Nombre\_Champ=sk.D\_Nombre\_Champ

group by c.D\_Nombre\_Champ,sk.Nombre\_Skin

go

exec skinss

--Cree una función que retorne quienes son los jugadores con un estilo de juego agresivo pero a la vez no 'bruto'; es decir, cuyo rubro de ferocidad sea el que tiene más puntos entre los tres tipos y donde sus puntos de astucia sean mayores a sus puntos de valor.

--Este procedimiento podría modificarse en varias partes para conseguir otras comparaciones o estilos de juego...y de esta manera clasificar posteriormente a los jugadores en diferentes estilos de juego. Esta importancia podría ser más para los jugadores (ver el estilo de juego de sus amigos o jugadores en su misma partida) que para la empresa.

create function jugadorcondeterminadoestilo() returns table

as

Return

(select u.C\_Username as Codigo, u.D\_Nombre\_Usuario, pm.C\_Nombre\_P\_Runa

from [User] u join PagMaestrias pm on u.C\_Username=pm.C\_Username

where pm.Q\_Ferocidad>pm.Q\_Astucia and pm.Q\_Astucia>pm.Q\_Valor)

go

print jugadorcondeterminadoestilo()

--Creen un trigger que notifique si un usuario eliminado ha efectuado por lo menos una transacción.

--Esto sería importante pues si un usuario es eliminado deberíamos eliminar también su transacción o al menos tener en cuenta que el usuario ya no es un jugador activo.

CREATE TRIGGER elimina ON [user]

FOR Delete

AS

IF EXISTS (SELECT \* FROM Transaccion t

WHERE t.C\_Username = (SELECT C\_Username FROM DELETED) )

Begin

Rollback transaction

print 'El usuario eliminado ha efectuado por lo menos 1 transaccion'

End

1. **Diego Linares**

De cada partida que haya durado más de 60 minutos mostrar el ID de la partida, Tipo y Servidor en el que se alojó.

Esta selección es importante para ver cuáles fueron las partidas más interesantes en un servidor (pues la duración es en promedio de 40 minutos). Asimismo, podría ser útil para saber cuál es el tipo de partida que genera más tiempo de juego en alguna otra funcionalidad de la BD.

select p.C\_IdPartida as Codigo, p.C\_IdTipo as Tipo, s.D\_Nombre\_Server as En\_Server

from Partida p join Server s on s.C\_Server=p.C\_Server

where p.T\_Duracion>60

order by p.T\_Duracion desc

--Crear un procedimiento que liste todos los jugadores que han alcanzado el nivel máximo en el juego en sus cuentas. Este procedimiento es importante pues mostrara todos los jugadores dedicados que han logrado alcanzar un nivel alto y pasado una buena cantidad de horas de juego; esto también podría ser importante para llevar a cabo una recompensa cuando alguien llegue al nivel máximo (30).

create procedure jugadoresmayoresanivel29

as

select u.C\_Username, s.D\_Nombre\_Server

from [User] u , Server s

where u.N\_Nivel\_Cta=30

Go

exec jugadoresmayoresanivel29

--Cree una función que retorne los jugadores, sus respectivos campeones usados y el equipo al que pertenecieron de una partida en especial. Esta función es importante pues nos mostrara los participantes de una partida y saber cuál campeón uso. Asimismo, se podría unir con otra función o selección que determine los objetos que compro el usuario en una partida para su campeón.

create function enpartida(@partida char(18)) returns table

as

Return(

select cu.C\_Username as Usuario, cu.D\_Nombre\_Champ as Campeon, eq.B\_Equipo as Equipo

from Equipo eq join Partida p on p.C\_IdPartida=eq.C\_IdPartida

join Campeon\_User cu on eq.B\_Equipo=cu.B\_Equipo

where p.C\_IdPartida=@partida

)

go

Select \* from enpartida('IDPartidaAqui')

Cree un trigger básico que muestre un mensaje de corroboración ante cambios, inserción o borrado en la tabla servidor.

CREATE TRIGGER servv ON [server]

FOR INSERT, UPDATE, DELETE

AS

PRINT 'Se realizo de manera correcta lo deseado!'

1. **Mateo Hervas**

--Determine la cantidad total de puntos puestos en cada rubro de las páginas de maestría a lo largo de toda la BD.

select SUM(pm.Q\_Ferocidad) as TotalFerocidad, sum(pm.Q\_Valor) as TotalValor, sum(pm.Q\_Astucia) as TotalAstucia

from PagMaestrias pm

--Cree un procedimiento que imprima cuales son los jugadores que tienen la página de maestrías con el número mayor de puntos en el rubro 'valor' y el nombre de su página.

--Esto es importante para poder saber quiénes son los jugadores con este estilo de juego más establecido.

--Esta importancia podría ser más para los jugadores que para la empresa, pues si estás buscando un jugador con este estilo de juego para jugar en equipo, haria la búsqueda mas fácil.

create procedure pmconmasvalor

as

select pm.C\_Username as DeUsuario, pm.C\_Nombre\_P\_Runa as NombrePag

from PagMaestrias pm

where pm.Q\_Valor=(select max(pm.Q\_Valor))

go

exec pmconmasvalor

--Cree una función que devuelva el id de una partida y el id del servidor que la alojo donde se usó un campeón especifico.

--Esta función es importante porque sabremos en que partidas se ha jugado al campeón que se desee encontrar y podríamos analizar las estadísticas, objetos que se le compro en una partida, duración, entre otros datos una vez que tengamos lo que nos devolverá esta función.

create function se\_uso\_al\_champ(@champ char(18)) returns table

as

Return(

select p.C\_IdPartida as Partida, p.C\_Server as EnServer

from Equipo eq join Partida p on p.C\_IdPartida=eq.C\_IdPartida

join Campeon\_User cu on eq.B\_Equipo=cu.B\_Equipo

where cu.D\_Nombre\_Champ=@champ

)

go

Select \* from se\_uso\_al\_champ('NombreCampeonAqui')

--Realize un trigger que notifique un error en la duracion de una partida. Este siempre debe ser un numero real positivo.

--Si tuvieramos partidas con duracion 0 quiere decir que hubo un error en la modificacion pues una partida con esta duracion ni siquiera debio haber sido ingresada.

--\*Identificable en las reglas de negocio; hay tiempo ilimitado, sin embargo no tiempo nulo.\*

CREATE TRIGGER notifica\_dur ON Partida

FOR UPDATE

AS

IF (SELECT T\_Duracion FROM Inserted) <=0

BEGIN

PRINT 'La duracion de una partida siempre debe ser mayor a CERO!'

ROLLBACK TRANSACTION

END