Bases de Datos - SQL Server

1.-Diseño de base de datos.

1.1.-Dado el detalle explicado en la parte inicial de este documento debería generar una base de datos similar al siguiente.

- 1.2.-Los registros de cada tabla deberían quedar de la siguiente forma .

tabla campeonato

id_campeonato	nombre_campeonato	sede
camp-111	Campeonato Unifranz	El Alto
camp-222	Campeonato Unifranz	Cochabamba

tabla equipo

id_equipo	nombre_equipo	categoria	id_campeonato
equ-111	Google	VARONES	camp-111
equ-222	404 Not found	VARONES	camp-111
equ-333	girls unifranz	MUJERES	camp-111

tabla jugador

id_jugador	nombres	apellidos	ci	edad	id_equipo
jug-111	Carlos	Villa	8997811LP	19	equ-222
jug-222	Pedro	Salas	8997822LP	20	equ-222
jug-333	Saul	Araj	8997833LP	21	equ-222
jug-444	Sandra	Solis	8997844LP	20	equ-333
jug-555	Ana	Mica	8997855LP	23	equ-333

- 1.2.-Manejo de Conceptos.
- Que es **DDL** y **DML**, adicionalmente muestra un ejemplo en la base de datos **UNIFRANZITOS**.

```sql
--DDL crea las tablas
create table campeonato (
id_campeonato  varchar (100) primary key,
nombre_capeonato varchar (100),
sede varchar (100)
);
```

```sql
--DML el que inserta y evalua .
insert into campeonato(id_campeonato, nombre_capeonato,sede)
values ('camp-111','Campeonato Unifranz','El Alto');
 insert into campeonato(id_campeonato, nombre_capeonato,sede)
values ('camp-222','Campeonato Unifranz','Cochabamba');
```

- 1.2.-Manejo de Conceptos.
- Que significa **PRIMARY KEY y FOREIGN KEY.**

```
-- el PRIMARY KEY es uno o mas campos que identifican de una manera unica en cada una de las filas de la tabla .

create table Equipo(
id_equipo varchar(100) primary key,
nombre_equipo varchar(100),
categoria varchar(100),
id_campeonato varchar (100),
foreign key (id_campeonato)
references campeonato  (id_campeonato)
);
```

```
--la restriccion FOREIGN KEY genara un vinculo entre dos tablas de las cuales se denominan tabla padre y tablña hijo

create table jugador(
id_jugador varchar (100)primary key,
nombres varchar(100),
apellidos varchar (101),
CI varchar (100),
edad integer ,
id_equipo varchar (100),
foreign key (id_equipo)
references equipo (id_equipo)
);
```

- 1.2.-Manejo de Conceptos.
- Defina que es una **TABLA y el uso de IDENTITY**.

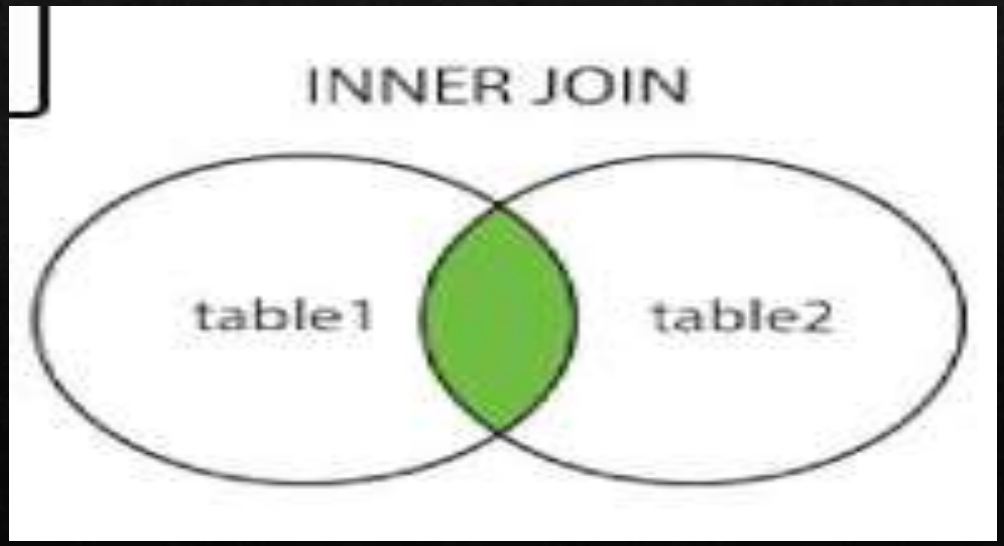
```
--Una identidad o identity en SQL Server es una columna que se asigna al crear o alterar (alter) una
--tabla desde el diseñador o por T-SQL. Una columna como identidad es auto incrementable, especificando el increment para cada nuevo registro.
  insert into equipo (id_equipo, nombre_equipo, categoria ,id_campeonato)
values('equ-111','google','VARONES','camp-111');
  insert into equipo (id_equipo, nombre_equipo, categoria ,id_campeonato)
values('equ-222','404 Not found','VARONES','camp-111');
  insert into equipo (id_equipo, nombre_equipo, categoria ,id_campeonato)
values('equ-333','girls unifranz','MUJERES','camp-111');
```

- 1.2.-Manejo de Conceptos.
- Para que se utiliza la cláusula **WHERE**.

WHERE	Utilizada para especificar las condiciones que deben reunir los registros que se van a seleccionar.

- 1.2.-Manejo de Conceptos.
- Para que se utiliza la instrucción **INNER JOIN.**

Combina los registros de dos tablas si hay valores coincidentes en un campo común.



- 1.2.-Manejo de Conceptos.
- Ejemplo de **INNER JOIN. Adjuntar una imagen de conjuntos y la consulta SQL que refleje el INNER JOIN**

```
SELECT A.*, B.* FROM personal A
    INNER JOIN direcciones B
        ON A.id = B.id
```

Results | Messages

id	nombre	id	domicilio
3	Andrés	3	Buenavista # 5B
4	Berenice	4	Av. La Paz # 1
5	Zenó	5	Unión # 38-A

```
SELECT p.Name AS ProductName,
NonDiscountSales = (OrderQty * UnitPrice),
Discounts = ((OrderQty * UnitPrice) * UnitPriceDiscount)
FROM Production.Product AS p
JOIN Sales.SalesOrderDetail AS sod
ON p.ProductID = sod.ProductID
ORDER BY ProductName DESC;
GO
```

100 %

Results | Messages

	ProductName	NonDiscountSales	Discounts
1	Women's Tights, S	179.976	0.00
2	Women's Tights, S	359.952	0.00
3	Women's Tights, S	224.97	0.00
4	Women's Tights, S	89.988	0.00
5	Women's Tights, S	89.988	0.00
6	Women's Tights, S	179.976	0.00
7	Women's Tights, S	224.97	0.00
8	Women's Tights, S	89.988	0.00
9	Women's Tights, S	618.6675	30.9334
10	Women's Tights, S	89.988	0.00

- 1.2.-Manejo de Conceptos.
- Ejemplo de **LEFT JOIN.** Adjuntar una **imagen de conjuntos** y la consulta SQL que refleje el **LEFT JOIN**

```
SELECT A.*, B.*
    FROM personal A
    LEFT JOIN direcciones B
        ON A.id = B.id
```

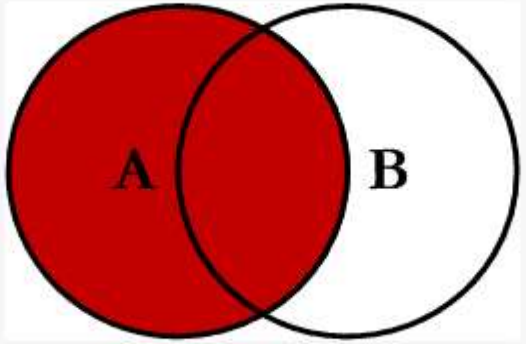
Results | Messages

id	nombre	id	domicilio
1	Laura	NULL	NULL
2	Moisés	NULL	NULL
3	Andrés	3	Buenavista # 5B
4	Berenice	4	Av. La Paz # 1
5	Zenó	5	Unión # 38-A

Cláusula LEFT JOIN

A diferencia de un `INNER JOIN`, donde se busca una intersección respetada por ambas tablas, con `LEFT JOIN` damos prioridad a la tabla de la izquierda, y buscamos en la tabla derecha.

Si no existe ninguna coincidencia para alguna de las filas de la tabla de la izquierda, de igual forma **todos los resultados de la primera tabla se muestran**.



He aquí una consulta de ejemplo:

```
SELECT
  E.Nombre as 'Empleado',
  D.Nombre as 'Departamento'
FROM Empleados E
LEFT JOIN Departamentos D
ON E.DepartamentoId = D.Id
```

- 1.2.-Manejo de Conceptos.
- Ejemplo de **RIGHT JOIN** .Adjuntar una **imagen de conjuntos** y la consulta SQL que refleje el **RIGHT JOIN.**

```
SELECT A.*, B.*
    FROM personal A
    RIGHT JOIN direcciones B
        ON A.id = B.id
```

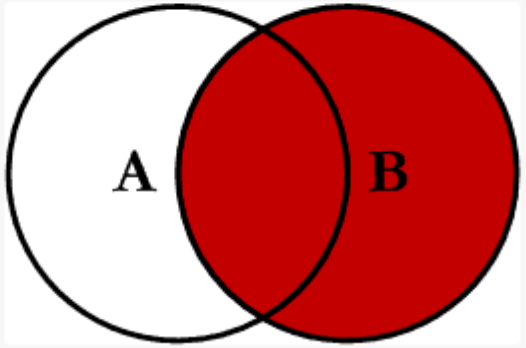
%

Results | Messages

id	nombre	id	domicilio
3	Andrés	3	Buenavista # 5B
4	Berenice	4	Av. La Paz # 1
5	Zenó	5	Unión # 38-A
NULL	NULL	6	Cerezas # 908
NULL	NULL	7	Av. Central # 67

Cláusula RIGHT JOIN

En el caso de `RIGHT JOIN` la situación es muy similar, pero aquí se da prioridad a la tabla de la derecha.



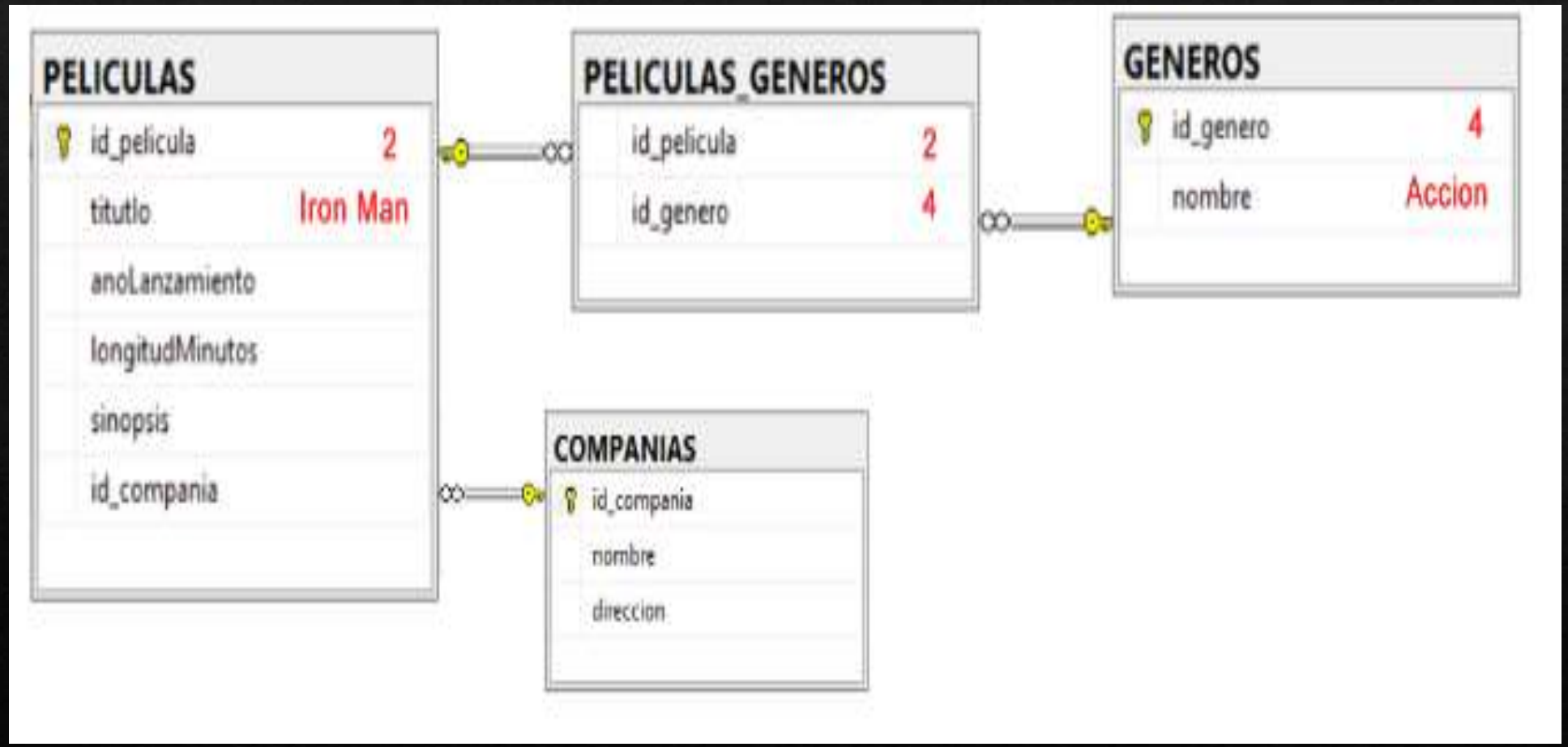
De tal modo que si usamos la siguiente consulta, estaremos **mostrando todas las filas de la tabla de la derecha**:

```
SELECT
  E.Nombre as 'Empleado',
  D.Nombre as 'Departamento'
FROM Empleados E
RIGHT JOIN Departamentos D
ON E.DepartamentoId = D.Id
```

La tabla de la izquierda es `Empleados` , mientras que `Departamentos` es la tabla de la derecha.

La tabla asociada al `FROM` será siempre la tabla `LEFT` , y la tabla que viene después del `JOIN` será la tabla `RIGHT` .

- 1.2.-Manejo de Conceptos.
- Crear 3 tablas y crear una consulta SQL que muestra el uso de **INNER JOIN**



3.Manejo de consultas

- Mostrar que **jugadores** que son del equipo **equ-222**

	id_jugador	nombres	apellidos	ci	edad	id_equipo
1	jug-111	Carlos	Villa	8997811LP	19	equ-222
2	jug-222	Pedro	Salas	8997822LP	20	equ-222
3	jug-333	Saul	Araj	8997833LP	21	equ-222

3.Manejo de consultas

- Mostrar que jugadores(**nombres, apellidos**) que juegan en la **sede de El Alto**.

```
select ju.nombres,ju.apellidos,ju.edad,eq.id_equipo,eq.nombre_equipo
from campeonato as camp inner join equipo as eq on camp.id_campeonato = eq.id_campeonato
inner join jugador as ju on ju.id_equipo = eq.id_equipo
where ju.edad>20 and eq.id_equipo = 'equ-333' and camp.sede = 'El Alto'
```

100 %

Results | Messages

	nombres	apellidos	edad	id_equipo	nombre_equipo
1	Ana	Mica	23	equ-333	girls unifmaz

3.Manejo de consultas

- Mostrar aquellos jugadores mayores o igual a **21** años que sean de la categoría **VARONES.**

```
select ju.nombres,ju.apellidos,ju.edad
from jugador as ju inner join equipo as eq on ju.id_equipo = eq.id_equipo
where eq.nombre_equipo='404 Not found' and ju.edad <= 20
```

00 %

Results | Messages

nombres	apellidos	edad

3.Manejo de consultas

- Mostrar a todos los estudiantes en donde su apellido empiece con la letra **S.** Podría utilizar la instrucción **LIKE**

```
select *
from jugador as ju
where ju.nombres like 's%' and ju.apellidos like '%s'

select ju.nombres,ju.apellidos
from premios as pre inner join jugador as ju on pre.id_equipo = ju.id_equipo
where pre.posicion='primero'
```

00 %

Results | Messages

	id_jugador	nombres	apellidos	ci	edad	id_equipo
1	jug-444	Sandra	Solis	8997844LP	20	equ-333

3.Manejo de consultas

- Mostrar que equipos forman parte del campeonato **camp-111** y además sean de la categoría **MUJERES.**

4	jug-444	Sandra	Solis	8997844LP	20	equ-333
5	jug-555	Ana	Mica	8997855LP	23	equ-333

- Mostrar el nombre del equipo del jugador con id_jugador igual a **jug-333**

3	jug-333	Saul	Araj	8997833LP	21	equ-222

Mostrar el nombre del campeonato del jugador con id_jugador igual a **jug-333**

3	jug-333	Saul	Araj	8997833LP	21	equ-222

Crear una consulta SQL que maneje las 3 tablas de la base de datos. ¿Qué estrategia utilizaría para determinar **cuántos equipos inscritos hay**?

Podría utilizar la función de agregación **COUNT**

¿Qué estrategia utilizaría para determinar cuántos jugadores pertenecen a la categoría **VARONES** o Categoría **MUJERES**.

Para esto puede utilizar la función de agregación **COUNT**

```
SELECT EmployeeID, COUNT(*) AS TodasFilas, COUNT(ShipRegion) AS FilasNoNulas,
MIN(ShippedDate) AS FechaMin, MAX(ShippedDate) AS FechaMax,
SUM(Freight) TotalPeso, AVG(Freight) Promedio
FROM Orders
GROUP BY EmployeeID
```

Results | Messages

	EmployeeID	TodasFilas	FilasNoNulas	FechaMin	FechaMax	TotalPeso	Promedio
1	9	43	14	1996-07-15 00:00:00.000	1998-05-04 00:00:00.000	3326,26	77,3548
2	3	127	56	1996-07-15 00:00:00.000	1998-05-06 00:00:00.000	10884,74	85,7066
3	6	67	32	1996-07-10 00:00:00.000	1998-04-24 00:00:00.000	3780,47	56,4249
4	7	72	22	1996-08-28 00:00:00.000	1998-05-05 00:00:00.000	6665,44	92,5755
5	1	123	50	1996-07-23 00:00:00.000	1998-05-06 00:00:00.000	8836,64	71,8426
6	4	156	62	1996-07-11 00:00:00.000	1998-05-01 00:00:00.000	11346,14	72,7316