

Algebra	versus	SQL
Puro		Impuro
Teórico		Práctico
No implementado		Implementado
No standard		Standardizado ANSI-92 x ejemplo
No devuelve tuplas repetidas		Puede devolver tuplas repetidas
Unión compatible estricta		Unión compatible más flexible
Solo consultas		DML,DDL,SDL,VDL
9 instrucciones		1 instrucción: SELECT

BD: /var/lib/firebird/3.0/data/1314tp2ej3.fdb

SUCURSAL(NOMBRE,DIRECCION)
 PK(NOMBRE)
 EMPLEADO(DNI,APENOM,NOMBRE)
 PK(DNI)
 FK(NOMBRE,SUCURSAL)
 PRODUCTO(CÓDIGO,DESCRIPCIÓN,STOCK)
 PK(CÓDIGO)
 SEVENDE(CÓDIGO,NOMBRE,PRECIO,STOCK)
 PK(CÓDIGO,NOMBRE)
 FK(CÓDIGO,PRODUCTO)
 FK(NOMBRE,SUCURSAL)

Asignación / Renombrar	Alias a nivel de tabla / columna Pseudónimo / alias para las tablas: SELECT P.CÓDIGO,S.NOMBRE FROM PRODUCTO P,SUCURSAL S Pseudónimo / alias para las tablas y columnas: Ej1: SELECT P.CÓDIGO COD,S.NOMBRE NOM FROM PRODUCTO P,SUCURSAL S Ej2: SELECT P.CÓDIGO "MI CÓDIGO",S.NOMBRE "MI NOMBRE" FROM PRODUCTO P,SUCURSAL S No recomendable nombres de atributos con espacios intermedios
Proyección	Ej: Listar los distintos nombres de productos que hay en esta BD (si hay dos o más productos que tienen el mismo nombre, solo aparece una única vez): SELECT DISTINCT DESCRIPCIÓN FROM PRODUCTO Si hay dos o más productos que tienen el mismo nombre, quiero que los muestre a todos: SELECT DESCRIPCIÓN

	FROM PRODUCTO Quiero que la salida se ordene (A-Z) por DESCRIPCIÓN: SELECT DESCRIPCIÓN FROM PRODUCTO ORDER BY DESCRIPCIÓN ASC Quiero que la salida se ordene (Z-A) por DESCRIPCIÓN: SELECT DESCRIPCIÓN FROM PRODUCTO ORDER BY DESCRIPCIÓN DESC Quiero mostrar código y descripción de productos: SELECT CÓDIGO, DESCRIPCIÓN FROM PRODUCTO ORDER BY DESCRIPCIÓN DESC, CÓDIGO ASC (ordena por descripción en forma descendente y, a igualdad de descripción, ordena por código en forma ascendente) ... Zzzz 100 Zzzz 200 ... Lista todas las columnas de producto: SELECT * FROM PRODUCTO Lista todas las columnas de producto y ordena la salida por CODIGO (de menor a mayor): SELECT * FROM PRODUCTO ORDER BY CÓDIGO ASC
Selección	Clausula WHERE <condición> Condición: AND, OR, NOT, IN(), NOT IN(), BETWEEN .. , STARTING WITH .., ALL .. , SOME .. ,ANY.. EXISTS(<SELECT>), NOT EXISTS(<SELECT>), IS NULL, IS NOT NULL, LIKE .., NOT LIKE .., SIMILAR TO, NOT SIMILAR TO Listar los productos con código > 100: SELECT * FROM PRODUCTO WHERE CODIGO > 100 Listar los productos con código entre 100 y 500: SELECT * FROM PRODUCTO WHERE CODIGO >= 100 AND CODIGO <= 500

O bien:

```
SELECT * FROM PRODUCTO  
WHERE CODIGO BETWEEN 100 AND 500
```

Op condicionales: >, <, >=, <=, <>
AND,OR,NOT

Clausula IN(<lista de valores>)

Listar los productos con códigos 100,110,120,130:

```
SELECT * FROM PRODUCTO  
WHERE CODIGO IN(100,110,120,130)
```

Listar los productos que NO tengan los códigos 100,110,120,130:

```
SELECT * FROM PRODUCTO  
WHERE CODIGO NOT IN(100,110,120,130)
```

Clausula IN(<SELECT>)

Listar los productos que se venden en la sucursal de 'LUJAN'

-productos que se venden en lujan:

```
SELECT CODIGO FROM SEVENDE WHERE NOMBRE = 'LUJAN'  
123  
151  
231  
...
```

```
SELECT * FROM PRODUCTO  
WHERE CODIGO IN(  
    SELECT CODIGO  
    FROM SEVENDE  
    WHERE NOMBRE = 'LUJAN'  
)
```

Listar los productos que NO se venden en la sucursal de 'LUJAN'

```
SELECT * FROM PRODUCTO  
WHERE CODIGO NOT IN(  
    SELECT CÓDIGO  
    FROM SEVENDE  
    WHERE NOMBRE = 'LUJAN'  
)
```

Listar los productos cuyo nombre comiencen con 'TE' (ej:
TENAZA)

```
SELECT * FROM PRODUCTO  
WHERE DESCRIPCIÓN STARTING WITH 'TE'
```

O bien:

```
SELECT * FROM PRODUCTO
```

	WHERE DESCRIPCIÓN LIKE 'TE%' Like's Wildcards: % [] _ ? (puede variar de un SGBD a otro) 'TE[nN]%' 'TE[nN][aA]%' Los corchetes no funcionan en firebird 3.0, tampoco el ?, si funciona el _ Las expresiones regulares funcionan con cláusula SIMILAR TO SELECT * FROM PRODUCTO WHERE DESCRIPCION SIMILAR TO 'PR[oO][Dd]UCTO%'; Trae todos los productos con descripciones tales como: PRODUCTO 100 PRODUCTO 200 PRODUCTO 300 PRODUCTO 400 PRODUCTO 500 Listar los productos con código que supere en todos los casos a los códigos 100 y 200 de producto: SELECT * FROM PRODUCTO WHERE CODIGO > ALL (SELECT CODIGO FROM PRODUCTO WHERE CODIGO IN(100,200)) Listar los productos con código que supere en algún caso a los códigos 100 y 200 de producto: SELECT * FROM PRODUCTO WHERE CODIGO > SOME (SELECT CODIGO FROM PRODUCTO WHERE CODIGO IN(100,200)) SELECT * FROM PRODUCTO WHERE CODIGO > ANY (SELECT CODIGO FROM PRODUCTO WHERE CODIGO IN(100,200))
Intersección	En algunos SGBD se utiliza la cláusula INTERSECT dentro de FROM (ej. Oracle), FB 3.0 no tiene INTERSECT, usar INNER JOIN
Resta	En algunos SGBD se utiliza la cláusula MINUS dentro de FROM (ej. Oracle), FB 3.0 no tiene MINUS, por lo tanto, hay que hacerlo en forma manual con los recursos que tiene Fb 3.0, por ejemplo, usar LEFT JOIN combinado con WHERE preguntando por IS NULL --RESTA --EJ: LISTAR LOS PRODUCTOS QUE NO SE VENDEN EN NINGUNA SUCURSAL -- TODOS LOS PRODUCTOS - LOS PRODUCTOS QUE SE

	VENDEN -- <pre>SELECT P.CODIGO,P.DESCRIPCION FROM PRODUCTO P LEFT JOIN SEVENDE SV ON P.CODIGO = SV.CODIGO WHERE SV.CODIGO IS NULL ;</pre>
Unión	Asumiendo que existen las tablas T1, T2 y que las columnas T1.A = T2.X, T1.B = T2.Y, T1.C = T2.Z podemos hacer la unión de esta forma: <pre>SELECT A,B,C FROM T1 UNION SELECT X,Y,Z FROM T2</pre> Y si queremos conservar las tuplas repetidas: <pre>SELECT A,B,C FROM T1 UNION ALL SELECT X,Y,Z FROM T2</pre> Listar los empleados de Luján y de Chivilcoy: <pre>SELECT DNI,APENOM,NOMBRE WHERE NOMBRE = 'LUJAN' UNION ALL SELECT DNI,APENOM,NOMBRE WHERE NOMBRE = 'CHIVILCOY'</pre>
Producto	Quiero combinar todos los productos con todas las sucursales: <pre>SELECT CODIGO,NOMBRE FROM PRODUCTO,SUCURSAL</pre> Atención: esto funciona porque no hay ambigüedad con los nombres código y nombre, para evitar ambigüedad podemos usar pseudónimos para las tablas y columnas: <pre>SELECT P.CODIGO,S.NOMBRE FROM PRODUCTO P,SUCURSAL S</pre>
Reunión Natural	Ej: listar los productos que se venden en las sucursales: <pre>SELECT S.NOMBRE,SV.CODIGO,SV.PRECIO FROM SUCURSAL S NATURAL JOIN SEVENDE SV</pre> Agregamos la descripción de producto: <pre>SELECT S.NOMBRE,SV.CODIGO,P.DESCRIPCION,SV.PRECIO FROM SUCURSAL S NATURAL JOIN SEVENDE SV NATURAL JOIN PRODUCTO P</pre> Listo!, no funciona!! No da error el query pero no devuelve ninguna tupla! eso se debe a cómo funciona el NATURAL JOIN : el NATURAL JOIN hace join por igualdad de valor en los campos que tienen el mismo nombre y en este caso, si descomponemos la consulta: <pre>SUCURSAL NATURAL JOIN campos en común: NOMBRE SUCURSAL + SEVENDE NATURAL JOIN PRODUCTO campos en</pre>

	común: CODIGO, STOCK Como los valores de SEVENDE.STOCK no coinciden con los de PRODUCTO.STOCK -> no devuelve ninguna tupla Hay que tener cuidado con los campos con igual nombre cuando usamos NATURAL JOIN
Reunión Theta	Idem anterior, agregando theta join con producto (para mostrar la descripción del producto): <pre>SELECT S.NOMBRE,SV.CODIGO,P.DESCRIPCION,SV.PRECIO FROM (SUCURSAL S NATURAL JOIN SEVENDE SV) INNER JOIN PRODUCTO P ON P.CODIGO = SV.CODIGO</pre> Habiendo campos con igual nombre, una forma de simplificar los JOIN's (theta y externos) es usar la cláusula USING: <pre>SELECT S.NOMBRE,SV.CODIGO,P.DESCRIPCION,SV.PRECIO FROM (SUCURSAL S NATURAL JOIN SEVENDE SV) INNER JOIN PRODUCTO P USING(CODIGO)</pre> El mismo query ahora todos los joins por igualdad y utilizando los campos en común (observe que solo necesito indicar JOIN y USING cuando es por igualdad y con campos con igual nombre): <pre>SELECT S.NOMBRE,SV.CODIGO,P.DESCRIPCION,SV.PRECIO FROM SUCURSAL S JOIN SEVENDE SV USING(NOMBRE) JOIN PRODUCTO P USING(CODIGO)</pre>
Reunión externa izquierda	Muestra producto 500 que no se vende en ninguna sucursal <pre>--REUNION EXTERNA IZQUIERDA --EJ: LISTAR TODOS LOS PRODUCTOS Y LAS SUCURSALES EN DONDE SE VENDE, SI NO --SE VENDE EN NINGUNA SUCURSAL, MOSTRAR NULL EN SUCURSAL SELECT P.CODIGO, P.DESCRIPCION, SV.NOMBRE FROM PRODUCTO P LEFT JOIN SEVENDE SV ON P.CODIGO = SV.CODIGO</pre>
Reunión externa derecha	Muestra sucursal Mercedes agregada nueva con NULL porque no vende ningún producto <pre>--REUNION EXTERNA DERECHA --EJ: LISTAR TODAS LAS SUCURSALES EN DONDE SE VENDEN PRODUCTOS --Y EN CASO DE QUE NO VENDAN NINGUN PRODUCTO PONER NULL SELECT SV.NOMBRE,S.NOMBRE,S.DIRECCION FROM SEVENDE SV RIGHT JOIN SUCURSAL S ON SV.NOMBRE = S.NOMBRE</pre>
Reunión externa full	<pre>--REUNION EXTERNA FULL --EJ: LISTAR TODAS LAS SUCURSALES EN DONDE SE VENDEN PRODUCTOS --Y EN CASO DE QUE NO VENDAN NINGUN PRODUCTO PONER</pre>

	NULL --TMB LISTAR TODOS LOS PRODUCTOS, INCLUSO LOS QUE NO SE VENDEN --EN NINGUN LADO SELECT P.CODIGO,P.DESCRIPCION,SV.NOMBRE,S.NOMBRE,S.DIRECCION FROM SEVENDE SV FULL JOIN SUCURSAL S ON SV.NOMBRE = S.NOMBRE FULL JOIN PRODUCTO P ON P.CODIGO = SV.CODIGO ;
División	/demos/sql/division.accdb BD MS-Access 2010 SUCPROD(SUC,PROD) PROD(PROD) SUCPROD / PROD Sucursal que vende TODOS los productos Vista PROD_X_SUCPROD: SELECT DISTINCT SP.SUC, P.PROD FROM PROD AS P, SUCPROD AS SP; División: SELECT DISTINCT SUC FROM SUCPROD WHERE SUC NOT IN(SELECT DISTINCT PXS.SUC FROM PROD_X_SUCPRO AS PXS LEFT JOIN SUCPROD AS SP ON PXS.PROD = SP.PROD AND PXS.SUC = SP.SUC WHERE SP.SUC IS NULL AND SP.PROD IS NULL);

HAY DOS TIPOS DE SUB-CONSULTAS EN SQL:

Ejemplo: listar todos los productos que NO se venden en la sucursal 'LUJAN':

Versión CORRELACIONADA (existe uno o más alias del query de nivel superior que se utilizan en el sub-query interior, en este caso, P. Son consultas que NO pueden resolverse por separado, sino que, por cada tupla de PRODUCTO se debe ejecutar el sub-query sobre la tabla SEVENDE), Ejemplo:

```
SELECT * FROM PRODUCTO P
WHERE NOT EXISTS(
  SELECT *
  FROM SEVENDE SV
  WHERE SV.NOMBRE = 'LUJAN' AND SV.CODIGO = P.CODIGO
)
```

NO CORRELACIONADAS (NO existe uno o más alias del query de nivel superior que se utilicen en el sub-query interior. Son consultas que pueden resolverse por separado, se puede ejecutar el sub-query de la tabla SEVENDE, guardar el resultado en memoria y luego usar ese resultado para resolver el query de nivel superior), Ejemplo:

```
SELECT * FROM PRODUCTO
WHERE CODIGO NOT IN(
  SELECT CODIGO
  FROM SEVENDE
  WHERE NOMBRE = 'LUJAN'
)
```

Otras Consultas para practicar SQL:

SUCURSAL(NOMBRE,DIRECCION)

EMPLEADO(DNI,APENOM,NOMBRE)

PRODUCTO(CODIGO,DESCRIPCION,STOCK)

SEVENDE(CODIGO,NOMBRE,PRECIO,STOCK)

1. ¿Cuáles son los productos que puede venderme el empleado 'EMPLEADO 1'?
2. ¿Cuáles son los productos que NO puede venderme el empleado 'EMPLEADO 1'?
3. ¿Cuáles son los precios posibles del producto 'PRODUCTO 200'?
4. ¿Cuáles son los productos que se venden (en cualquier sucursal) a un precio superior (mostrar el precio) al que se vende el producto 'PRODUCTO 200' en la sucursal de 'LUJAN'?
5. ¿Cuáles son los productos que tienen más stock que el producto 'PRODUCTO 200'?

FALTA VER EL TEMA DE AGRUPAMIENTO EN SQL (GROUP BY, FUNCIONES AGREGADAS SQL)

REPASAR INSERT, UPDATE, DELETE, INSERT INTO SELECT

Atte. Guillermo Cherencio