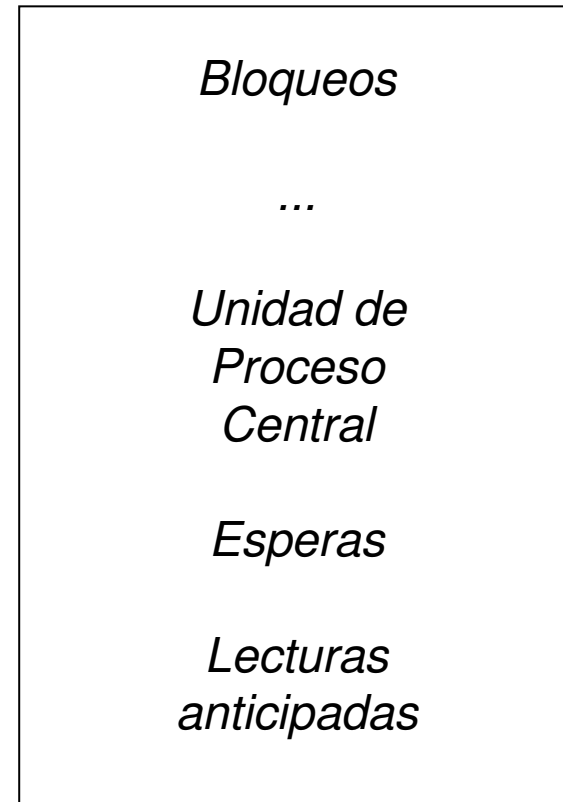
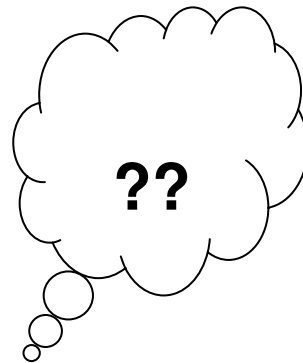
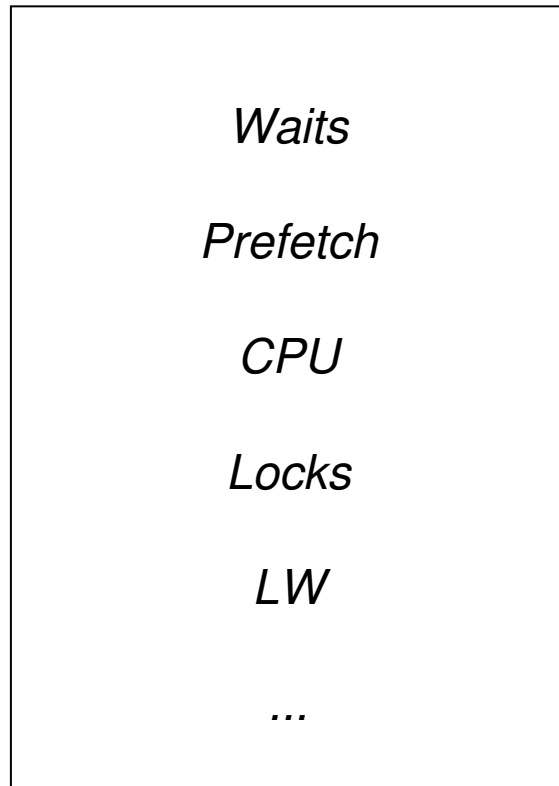

Lenguaje SQL

M&Y

Antes de empezar...



Tools

Manuales

Accounting

Capítulo 1 – Introducción



Objetivos

- ✓ **Conocer los diferentes lenguajes dentro del SQL**
 - ✓ **Definir los objetos básicos que se manejan con SQL**
 - ✓ **Aprender la nomenclatura dentro de las bases de datos**
-

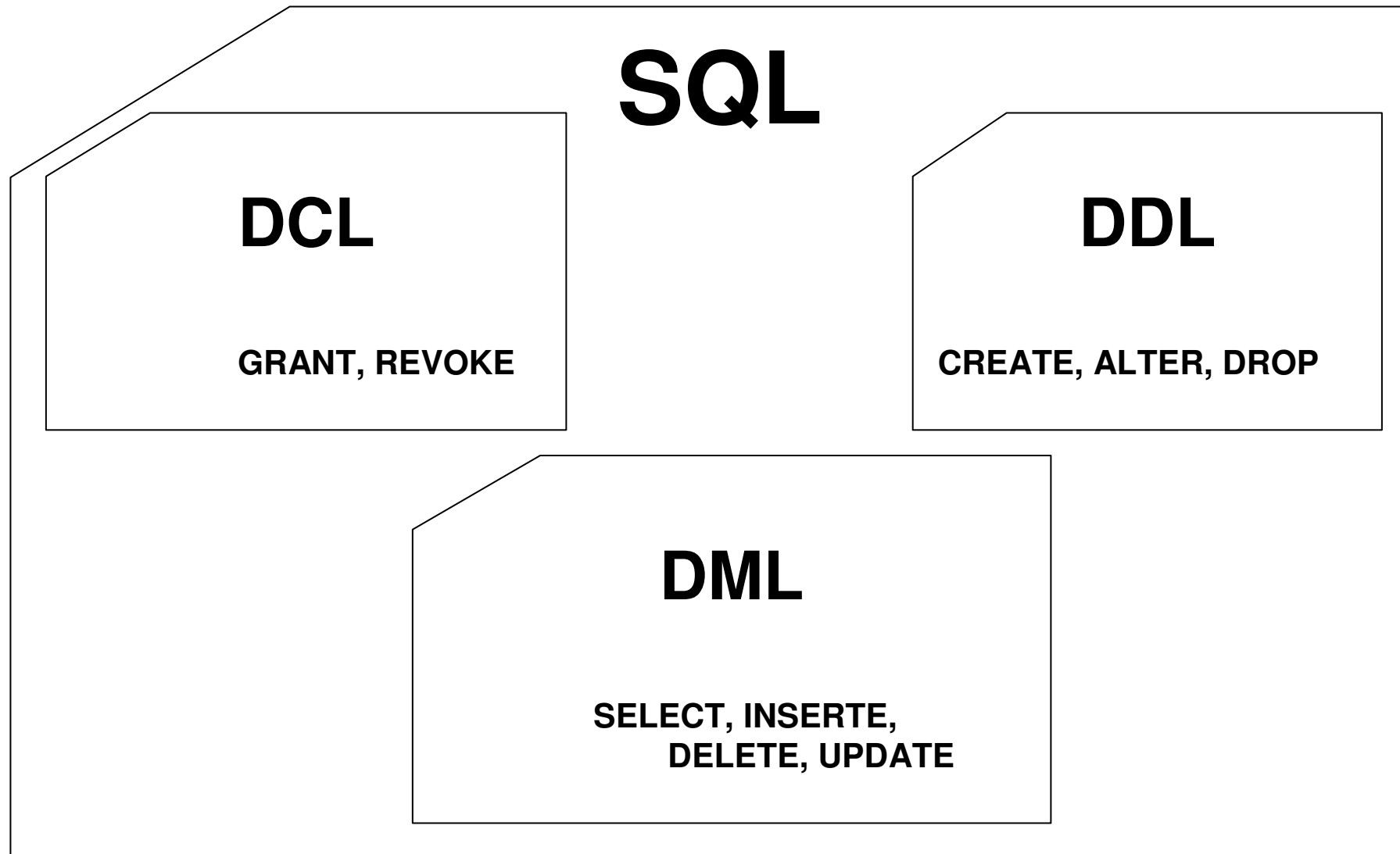
Su Curriculum Vitae

Los 70'	IBM lo desarrolla, denominándolo SEQUEL, SEQUEL-2 y por último SQL
Los 80'	La ANSI (American National Standard Institute) lo convierte en estándar para la definición y manipulación de datos en RDBMS.
Los 90'	Mejoras: SQL embebido (89), varias revisiones del estándar (SQL92, SQL9x). Nuevas versiones propias de ciertos SGDB
Hoy	Usado en todos los SGDB relacionales

Características básicas del lenguaje SQL

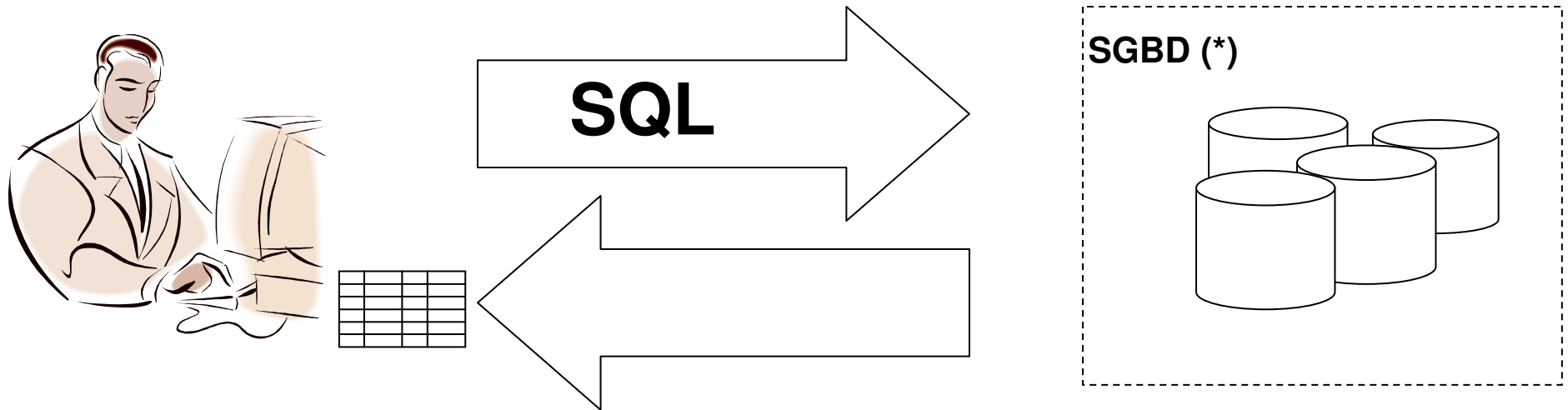
- **Está formado por tres tipos de sentencias:**
 - **DDL: Data Definition Language**
 - **DML: Data Manipulation Language**
 - **DCL: Data Control Language**
 - **No navegacional**
 - **Estándar**
-

Tipos de sentencias SQL...



Uso fundamental del lenguaje SQL

- Al ser estándar puede usarse contra diferentes bases de datos con muy pocas modificaciones.
- Interacción con las bases de datos



(*) DB2, ORACLE, SQL Server, Informix, etc.

Datos en forma de Tabla

Tabla PEDIDOS

NU_PEDIDO	NU_CAJA	FX_PEDIDO	NU_CLIENTE	CA_IMPORTE	DESCUENTO	COD_POSTAL
125	011	2005-11-07	005432	245	1	49029
126	013	2005-11-07	001238	236	5	49034
127	007	2005-11-07	000123	14	2	49347
128	008	2005-11-07	000354	19	-	49112
129	014	2005-11-07	002302	125	2	49201
130	002	2005-11-08	000259	21	-	49001
....

Fila

*Un tipo de dato por
columna*

Columna

Nulo

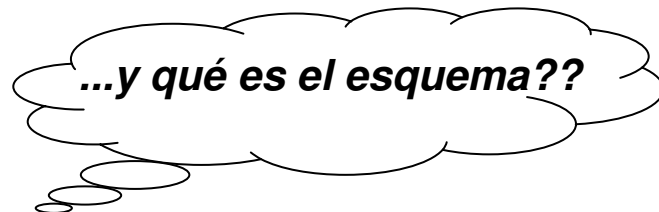
Cómo se nombran los objetos ?

- **Nombre compuesto de columna:**
nombre de la tabla + nombre de la columna.

PEDIDOS.NU_PEDIDO,

PEDIDOS.NU_CAJA, PEDIDOS.FX_PEDIDO

- **El nombre compuesto de la tabla:**
esquema + nombre simple de la tabla



El esquema

Sirve para clasificar las tablas y demás objetos en grupos

Esquema PROD

PROD.CLIENTES
PROD.PEDIDOS
PROD.PRODUCTOS

Esquema TEST

TEST.CLIENTES
TEST.PEDIDOS
TEST.PRODUCTOS

Esquema DESA

DESA.CLIENTES
DESA.PEDIDOS
DESA.PRODUCTOS

La definición de la tabla

```
CREATE TABLE CURSO.PEDIDOS
(NU_PEDIDO      INTEGER      NOT NULL,
 NU_CAJA        INTEGER      NOT NULL,
 FX_PEDIDO      DATE         NOT NULL,
 NU_CLIENTE     CHAR(8)      NOT NULL,
 CA_IMPORTE     DECIMAL(9,2) NOT NULL WITH DEFAULT,
 DESCUENTO      DECIMAL(3,2) ,
 COD_POSTAL     CHAR(5) )
```



**Nombre de
columna**



Tipo de dato



**Atributo
NULL**

Las columnas

Cada columna contiene un único tipo de datos.

De entre los siguientes:

- *Numéricos*
 - *Alfanuméricos*
 - *Fecha*
 - *Hora*
 - *Timestamp*
 - *LOBs*
 - *ROWID, UDT,...*
-

Los tipos de datos Numéricos

	Descripción	DB2	ORACLE
SMALLINT	<u>Small integer</u> . Entero binario con 15 bits de precisión	-32768 a +32767	
INTEGER	<u>Large integer</u> . Entero binario con 31 bits de precisión	-2147483648 a +2147483647	
REAL / DOUBLE	<u>Single precision floating-point</u> . Número de punto flotante corto/largo (32/64 bits)	-7.2E+75 a 7.2E+75	
DECIMAL(a,b)	<u>Packed decimal</u> . Limitado a 31 dígitos.	1 - 10 ³¹ a 10 ³¹ - 1. a: n° total dígitos b: n° posiciones decim.	
NUMBER(p,s)	Numérico con p dígitos y escala s.	N/A	p: de 1 a 38 s: de -84 a 127

Los tipos de datos Alfanuméricos

	Descripción	DB2	ORACLE
CHAR	Cadena de caracteres de longitud fija	1 a 255	1 a 2000
VARCHAR	Cadena de caracteres de longitud variable	1 a 32 Kb.	(VARCHAR2) 1 a 4000
		(+ 2 bytes para longitud)	
CLOB	Character LOB. Cadena de caracteres de longitud variable	Hasta 2 Gb.	Hasta 4 Gb.
NCLOB	Character LOB contiendo UNICODE	N/A	Hasta 4 Gb.
NCHAR / NVARCHAR2	Como CHAR / VARCHAR2 pero conteniendo UNICODE.	N/A	Idem CHAR / VARCHAR2

Los tipos de datos gráficos

	Descripción	DB2
GRAPHIC	Cadena de gráficos de longitud fija	1 a 127
VARGRAPHIC	Cadena de gráficos de longitud variable	1 a 16352 (Además 2 bytes para la longitud)
DBCLOB	Double-byte character LOB. Cadena de caracteres de longitud variable	Hasta 1 Gb.

Otros tipos de datos.

	Tipo de datos	DB2	ORACLE
BLOB	Cadena binaria de longitud variable.	Hasta 2 Gb.	Hasta 4 Gb.
ROWID	Identificador de fila	Internamente ocupa 17 bytes	
BFILE	Como BLOB. Es un puntero a un archivo binario externo.	N/A	Hasta 4 Gb.
RAW(s)	Datos binarios	N/A	1 a 2000
...

Fecha, hora y Timestamp: formato de presentación

	Tamaño interno	Formato externo según standard			
		USA	EUR	ISO	JIS
DATE	4 bytes	MM/DD/YYYY	DD.MM.YYYY	YYYY-MM-DD	YYYY-MM-DD
TIME	2 bytes	HH:MM AM/PM	HH.MM.SS	HH.MM.SS	HH:MM:SS
TIMESTAMP	10 bytes	YYYY-MM-DD-HH.MM.SS.MMMMMM			

Ejercicio 1



Capítulo 2 – Consultas SQL

Objetivos

- ✓ **Comprender el funcionamiento de la sentencia SELECT**
 - ✓ **Ver las diferentes cláusulas de una SELECT**
 - ✓ **Escribir condiciones para recuperar algunas filas**
 - ✓ **Manejar los operadores de rango especiales**
 - ✓ **Ordenar filas de salida**
-

Las cláusulas del SELECT y su contenido

SELECT	<i>Columnas Funciones (escalares, columna) Expresiones aritméticas Literales Subconsultas</i>
FROM	<i>Tablas, vistas, sinónimos</i>
WHERE	<i>Condiciones (predicados) de selección Subconsultas</i>
GROUP BY	<i>Columnas</i>
HAVING	<i>Condiciones sobre los grupos Subconsultas</i>
ORDER BY	<i>Columnas o posición de la columna</i>

Tablas para los ejemplos - CLIENTES

TABLA DE CLIENTES

CLIENTE	NOMBRE	APELLIDO	TIPO	TELEFONO	FX_ALTA	VENDEDOR	SEXO
110	AMPARO	ROCA	A1	649108410	01/01/1995	V110	F
220	MIGUEL	SOLA	A2	642911220	10/10/2003	V111	M
330	ANA	NOTARIO	A3	658490610	05/04/2005	V212	F
550	JUAN	RENATE	A1	683810205	17/08/2002	V342	M
660	LUIS	TEMPORA	A2	979291935	14/09/2003	V342	M
770	EVA	RAMIREZ	A4	696673695	30/09/2000	V111	F
990	ELENA	REZAGO	B1		15/08/2000	V145	F
1100	TOMAS	GOTA	A2	911999340	19/06/2000	V145	M
1210	JAIME	TAPADO	A4	943084050	16/05/2003	V231	M

Tablas para los ejemplos - VENDEDORES

TABLA DE VENDEDORES

VENDEDOR	NOMBRE	APELLIDO	FX_CONTRAT	SALARIO	RESPONSABLE
V110	JUAN ANTONIO	DONADO	2002-01-02	12500.00	V212
V111	CRISTINA	ALVAREZ	2002-01-13	11500.00	V212
V123	LUCIA	GOMEZ	2001-12-17	13500.00	V212
V134	ALVARO	GARCIA	2001-10-04	14500.00	V231
V145	LUIZ	PITINHO	2002-03-17	12500.00	V231
V212	CARMEN	MARQUEZ	2002-03-10	18500.00	V231
V213	PEDRO	MARIAS	2002-02-02	17500.00	V423
V222	ANGEL	RENILLA	2002-06-08	18000.00	V423

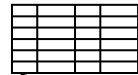
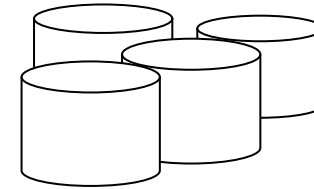
La SELECT más sencilla

*Vamos a obtener los datos
de todos los clientes*



SELECT *
FROM CLIENTES

DB2



CLIENTE	NOMBRE	APELLIDO	TIPO	TELEFONO	FX_ALTA	VENDEDOR	SEXO
110	AMPARO	ROCA	A1	649108410	01/01/1995	V110	F
220	MIGUEL	SOLA	A2	642911220	10/10/2003	V111	M
330	ANA	NOTARIO	A3	658490610	05/04/2005	V212	F
550	JUAN	RENATE	A1	683810205	17/08/2002	V342	M
660	LUIS	TEMPORA	A2	979291935	14/09/2003	V342	M
770	EVA	RAMIREZ	A4	696673695	30/09/2000	V111	F
990	ELENA	REZAGO	B1		15/08/2000	V145	F
1100	TOMAS	GOTA	A2	911999340	19/06/2000	V145	M
1210	JAIME	TAPADO	A4	943084050	16/05/2003	V231	M

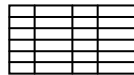
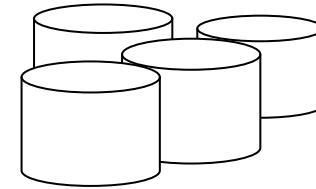
Seleccionar varias columnas

*Quiero recuperar el nombre,
apellido y fecha de alta de
los clientes*



```
SELECT NOMBRE,  
APELLIDO, FX_ALTA  
FROM CLIENTES
```

DB2



NOMBRE	APELLIDO	FX_ALTA
AMPARO	ROCA	01/01/1995
MIGUEL	SOLA	10/10/2003
ANA	NOTARIO	05/04/2005
JUAN	RENATE	17/08/2002
LUIS	TEMPORA	14/09/2003
EVA	RAMIREZ	30/09/2000
ELENA	REZAGO	15/08/2000
TOMAS	GOTA	19/06/2000
JAIME	TAPADO	16/05/2003

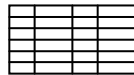
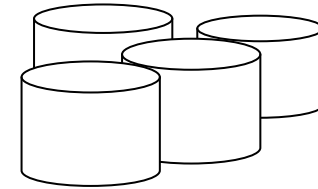
Ordenar las filas resultado

A los clientes de antes, quiero verlos clasificados según la fecha de alta de cada uno



```
SELECT FX_ALTA,  
NOMBRE, APELLIDO  
FROM CLIENTES  
ORDER BY FX_ALTA
```

DB2



FX_ALTA	NOMBRE	APELLIDO
01/01/1995	AMPARO	ROCA
19/06/2000	TOMAS	GOTA
15/08/2000	ELENA	REZAGO
30/09/2000	EVA	RAMIREZ
17/08/2002	JUAN	RENATE
16/05/2003	JAIME	TAPADO
14/09/2003	LUIS	TEMPORA
10/10/2003	MIGUEL	SOLA
05/04/2005	ANA	NOTARIO

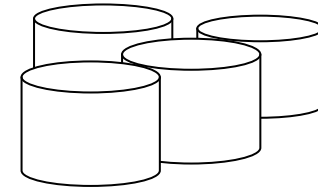
Ordenar las filas por varios campos

*Quiero ver los clientes anteriores
ordenados por el tipo de cliente
en orden descendente
y la fx_alta ascendente*



```
SELECT NOMBRE, APELLIDO,  
TIPO, FX_ALTA  
FROM CLIENTES  
ORDER BY TIPO DESC,  
FX_ALTA
```

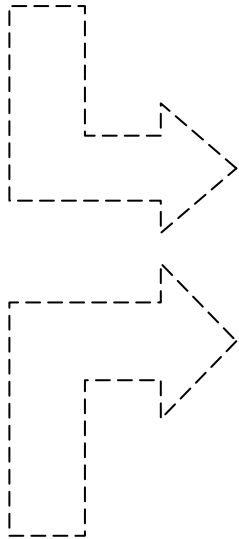
DB2



NOMBRE	APELLIDO	TIPO	FX_ALTA
ELENA	REZAGO	B1	2000-08-15
EVA	RAMIREZ	A4	2000-09-30
JAIME	TAPADO	A4	2003-05-16
ANA	NOTARIO	A3	2005-04-05
TOMAS	GOTA	A2	2000-06-19
LUIS	TEMPORA	A2	2003-09-14
MIGUEL	SOLA	A2	2003-10-10
AMPARO	ROCA	A1	1995-01-01
JUAN	RENATE	A1	2002-08-17

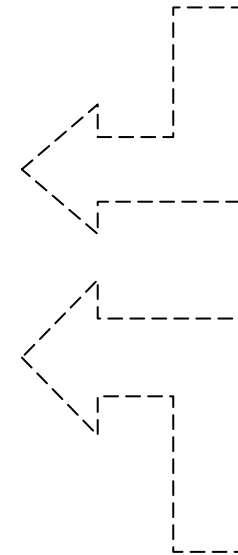
Cuatro formas diferentes de pedir el mismo orden

```
SELECT NOMBRE, APELLIDO,  
TIPO, FX_ALTA  
FROM CLIENTES  
ORDER BY TIPO DESC, FX_ALTA
```



NOMBRE	APELLIDO	TIPO	FX_ALTA
ELENA	REZAGO	B1	2000-08-15
EVA	RAMIREZ	A4	2000-09-30
JAIME	TAPADO	A4	2003-05-16
ANA	NOTARIO	A3	2005-04-05
TOMAS	GOTA	A2	2000-06-19
LUIS	TEMPORA	A2	2003-09-14
MIGUEL	SOLA	A2	2003-10-10
AMPARO	ROCA	A1	1995-01-01
JUAN	RENATE	A1	2002-08-17

```
SELECT NOMBRE, APELLIDO,  
TIPO, FX_ALTA  
FROM CLIENTES  
ORDER BY 3 DESC, FX_ALTA
```



```
SELECT NOMBRE, APELLIDO,  
TIPO, FX_ALTA  
FROM CLIENTES  
ORDER BY 3 DESC, 4
```

```
SELECT NOMBRE, APELLIDO,  
TIPO, FX_ALTA  
FROM CLIENTES  
ORDER BY TIPO DESC, 4
```

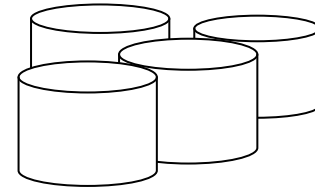
Seleccionar filas no repetidas

*Veamos qué vendedores han
atendido a los clientes*



**SELECT DISTINCT VENDEDOR
FROM CLIENTES**

DB2



VENDEDOR
V110
V111
V145
V212
V231
V342

Recuperación de un subconjunto de filas

TABLA DE VENDEDORES

VENDEDOR	NOMBRE	APELLIDO	FX_CONTRAT	SALARIO	RESPONSABLE
V110	JUAN ANTONIO	DONADO	2002-01-02	12500.00	V212
V111	CRISTINA	ALVAREZ	2002-01-13	11500.00	V212
V123	LUCIA	GOMEZ	2001-12-17	13500.00	V212
V134	ALVARO	GARCIA	2001-10-04	14500.00	V231
V145	LUIZ	PITINHO	2002-03-17	12500.00	V231
V212	CARMEN	MARQUEZ	2002-03-10	18500.00	V231
V213	PEDRO	MARIAS	2002-02-02	17500.00	V423
V222	ANGEL	RENILLA	2002-06-08	18000.00	V423

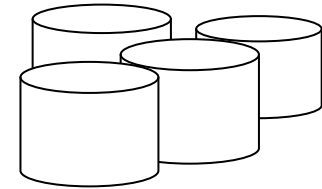
La cláusula WHERE... (evaluación de columnas numéricas)

***Quiénes son los vendedores
que cobran menos de 15000***



```
SELECT VENDEDOR, NOMBRE,  
SALARIO  
FROM VENDEDOR  
WHERE SALARIO < 13000
```

DB2



VENDEDOR	NOMBRE	SALARIO
V110	JUAN ANTONIO	12500.00
V111	CRISTINA	11500.00
V145	LUIZ	12500.00

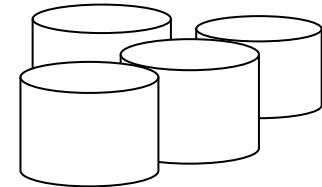
La cláusula WHERE... (uso de columnas alfanuméricas)

***Quiero sólo los clientes
que son del tipo A1***



```
SELECT CLIENTE, NOMBRE,  
APELLIDO, TIPO  
FROM CLIENTES  
WHERE TIPO = 'A1'
```

DB2



CLIENTE	NOMBRE	APELLIDO	TIPO
110	AMPARO	ROCA	A1
550	JUAN	RENATE	A1

**"Al evaluar columnas alfanuméricas
los valores van entre comillas"**

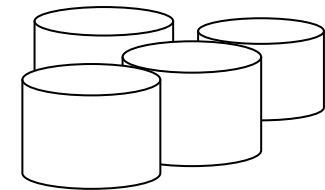
La cláusula WHERE... (Uso de columnas fecha, hora...)



```
SELECT *  
FROM PEDIDOS  
WHERE HORA_COMPRA > '20:00'
```

```
SELECT *  
FROM VENDEDOR  
WHERE FX_CONTRATO < '01-01-2000'
```

DB2



Operadores de comparación en el WHERE

=	Igual
<	Menor
>	Mayor
<=	Menor o igual
>=	Mayor o igual
NOT= , ≠ , <>	Distinto (El símbolo puede variar según el SGBD)

Evaluación de múltiples predicados

- ✓ Para evaluar más de un predicado se usan los operadores lógicos AND y OR.

AND: son ciertos cuando los predicados a ambos lados del AND son ciertos

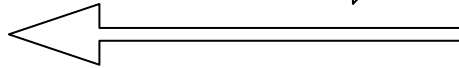
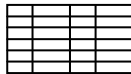
OR: son ciertos si alguno de los dos predicados es cierto

Varios predicados en la cláusula WHERE... 1/4

"¿Qué clientes son del tipo A2 y se dieron de alta desde el 2001? "

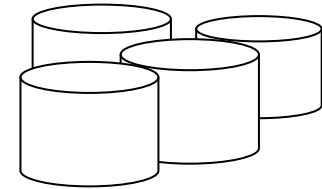


```
SELECT CLIENTE, NOMBRE,  
APELLIDO, TIPO, FX_ALTA  
FROM CLIENTES  
WHERE TIPO = 'A2' AND  
FX_ALTA >= '01-01-2001'
```



CLIENTE	NOMBRE	APELLIDO	TIPO	FX_ALTA
220	MIGUEL	SOLA	A2	10/10/2003
660	LUIS	TEMPORA	A2	14/09/2003

DB2



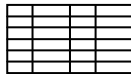
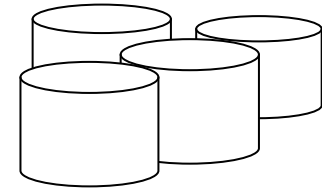
Varios predicados en la cláusula WHERE... 2/4

"¿Qué clientes son del tipo A2 ó se dieron de alta desde el 2001? "



```
SELECT CLIENTE, NOMBRE,  
APELLIDO, TIPO, FX_ALTA  
FROM CLIENTES  
WHERE TIPO = 'A2' OR  
FX_ALTA > '01-01-2001'
```

DB2



CLIENTE	NOMBRE	APELLIDO	TIPO	FX_ALTA
220	MIGUEL	SOLA	A2	10/10/2003
660	LUIS	TEMPORA	A2	14/09/2003
1100	TOMAS	GOTA	A2	19/06/2000
330	ANA	NOTARIO	A3	05/04/2005
550	JUAN	RENATE	A1	17/08/2002
1210	JAIME	TAPADO	A4	16/05/2003

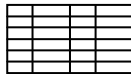
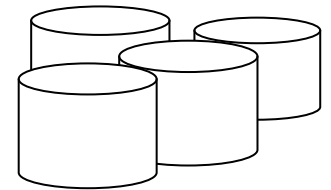
Varios predicados en la cláusula WHERE... 3/4

"Lista de clientes de tipo A2 y dados de alta desde el 2001, y de los clientes con alta anterior a 1998 "



```
SELECT CLIENTE, NOMBRE,  
APELLIDO, TIPO  
FROM CLIENTES  
WHERE TIPO = 'A2' AND  
FX_ALTA > '01-01-2001'  
OR FX_ALTA < '01-01-1998'
```

DB2



CLIENTE	NOMBRE	APELLIDO	TIPO	FX_ALTA
220	MIGUEL	SOLA	A2	10/10/2003
660	LUIS	TEMPORA	A2	14/09/2003
110	AMPARO	ROCA	A1	01/01/1995

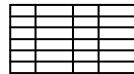
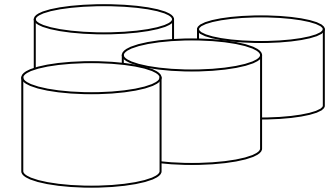
Varios predicados en la cláusula WHERE... 4/4

"Lista de clientes de tipo A2 y con alta mayor que el 1-10-2003 ó con anterioridad a 2001 "



```
SELECT CLIENTE, NOMBRE,  
APELLIDO, TIPO  
FROM CLIENTES  
WHERE TIPO = 'A2' AND  
(FX_ALTA > '01-10-2003' OR  
FX_ALTA < '01-01-2001')
```

DB2



CLIENTE	NOMBRE	APELLIDO	TIPO	FX_ALTA
220	MIGUEL	SOLA	A2	10/10/2003
1100	TOMAS	GOTA	A2	19/06/2000

"Cuando existen varios predicados es recomendable utilizar paréntesis para evitar confusiones"

Otros operadores de comparación en el WHERE

LIKE	Se usa con caracteres comodín: % → uno, ninguno o varios caracteres _ → un único carácter posicional
BETWEEN	Para recuperar un rango de filas incluyendo el límite inferior y superior
IN	Para evaluar una lista de valores

IS NULL	Para buscar filas con valores nulos en un determinado campo
----------------	---

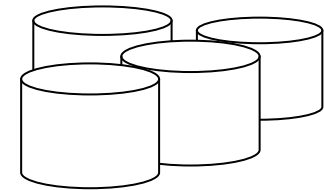
Uso del operador *LIKE* ... 1/3

"Lista de los clientes cuyo teléfono comienza por 9 "



```
SELECT CLIENTE, NOMBRE,  
APELLIDO, TELEFONO  
FROM CLIENTES  
WHERE TELEFONO LIKE '9%'
```

DB2



CLIENTE	NOMBRE	APELLIDO	TELEFONO
660	LUIS	TEMPORA	939291935
1100	TOMAS	GOTA	911999340
1210	JAIME	TAPADO	943084050

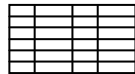
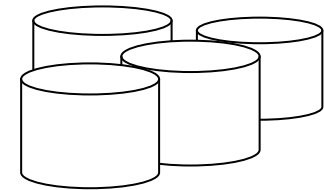
Uso del operador *LIKE* ... 2/3

"Lista de los vendedores en los que el segundo carácter del código de vendedor sea un '2'"



```
SELECT VENDEDOR, NOMBRE,  
APELLIDO  
FROM VENDEDOR  
WHERE VENDEDOR LIKE '_2%'
```

DB2



VENDEDOR	NOMBRE	APELLIDO
V212	CARMEN	MARQUEZ
V213	PEDRO	MARIAS
V222	ANGEL	RENILLA

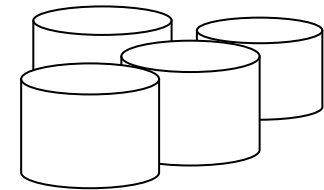
Uso del operador *LIKE* ... 3/3

"Lista de los clientes cuyo NOMBRE contenga la letra L en cualquier posición"



```
SELECT CLIENTE, NOMBRE,  
APELLIDO  
FROM CLIENTES  
WHERE NOMBRE LIKE '%L%'
```

DB2



CLIENTE	NOMBRE	APELLIDO
220	MIGUEL	SOLA
660	LUIS	TEMPORA
990	ELENA	REZAGO

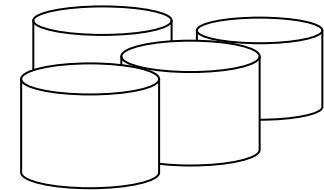
Uso del operador *BETWEEN*

"Lista de los VENDEDORES contratados desde en el último trimestre de 2001"



```
SELECT VENDEDOR, NOMBRE,  
APELLIDO  
FROM VENDEDOR  
WHERE FX_CONTRAT BETWEEN  
'01-09-2001' AND '31-12-2001'
```

DB2



VENDEDOR	NOMBRE	APELLIDO	FX_CONTRAT
V123	LUCIA	GOMEZ	2001-12-17
V134	ALVARO	GARCIA	2001-10-04

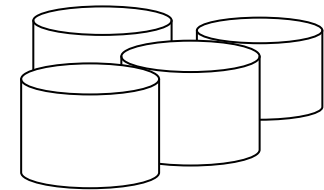
Uso del operador IN

*"Lista de los clientes de TIPO
A3, A4 ó B1 "*



```
SELECT CLIENTE, NOMBRE,  
APELLIDO, TIPO  
FROM CLIENTES  
WHERE TIPO IN ('A3', 'A4', 'B1')
```

DB2



CLIENTE	NOMBRE	APELLIDO	TIPO
330	ANA	NOTARIO	A3
770	EVA	RAMIREZ	A4
990	ELENA	REZAGO	B1
1210	JAIME	TAPADO	A4

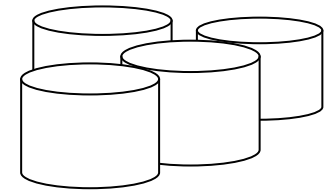
Uso de la negación: NOT

*"Lista de los clientes
cuyo TIPO no
empiece por A"*



```
SELECT CLIENTE, NOMBRE,  
APELLIDO, TIPO  
FROM CLIENTES  
WHERE TIPO NOT LIKE 'A%'
```

DB2



CLIENTE	NOMBRE	APELLIDO	TELEFONO
990	ELENA	REZAGO	-

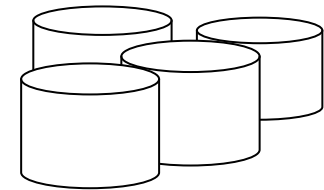
Uso del operador IS NULL

*"Lista de los clientes
sin teléfono
(el teléfono es nulo) "*



```
SELECT CLIENTE, NOMBRE,  
APELLIDO, TELEFONO  
FROM CLIENTES  
WHERE TELEFONO IS NULL
```

DB2



CLIENTE	NOMBRE	APELLIDO	TELEFONO
990	ELENA	REZAGO	-

Ejercicio 2

Ejercicio 2 – Tablas para todos los ejercicios

CLIENTES		
CLIENTE	CHAR(6)	NOT NULL
NOMBRE	CHAR(15)	NOT NULL
APELLIDO1	CHAR(20)	NOT NULL
APELLIDO2	CHAR(20)	NOT NULL
TIPO	CHAR(3)	
TELEFONO	CHAR(9)	
FX_ALTA	DATE	
VENDEDOR	CHAR(6)	
CODPOSTAL	CHAR(5)	
SEXO	CHAR(1)	
FX_NACIMIENTO	DATE	
COMPRAS_ANUAL	DECIMAL(8,2)	
EMPRESA	INTEGER	

TIPOCLIE		
TIPO	CHAR(3)	NOT NULL
DS_TIPO	VARCHAR(25)	
LIMITE_COMPRAS	DECIMAL(8,2)	
RESPONSABLE	CHAR(6)	

TARJETAS		
CLIENTE	CHAR(6)	NOT NULL
NUM_TARJETA	CHAR(15)	NOT NULL
NOM_AUTORIZADO	CHAR(15)	NOT NULL
APE_AUTORIZADO	CHAR(20)	NOT NULL
FX_CADUCIDAD	DATE	
LIM_MES	SMALLINT	

VENDEDOR		
VENDEDOR	CHAR(6)	NOT NULL
NOMBRE	CHAR(15)	NOT NULL
APELLIDO	CHAR(20)	NOT NULL
FX_CONTRAT	DATE	
SALARIO	DECIMAL(8,2)	
COM_VENTAS	SMALLINT	
RESPONSABLE	CHAR(6)	

VALEDESC		
CLIENTE	CHAR(6)	NOT NULL
FX_VALE	DATE	NOT NULL
IMPORTE	DECIMAL(6,2)	

Ejercicio 2 – Tablas para todos los ejercicios

CLIENTES		
CLIENTE	CHAR(6)	NOT NULL
NOMBRE	CHAR(15)	NOT NULL
APELLIDO1	CHAR(20)	NOT NULL
APELLIDO2	CHAR(20)	NOT NULL
TIPO	CHAR(3)	
TELEFONO	CHAR(9)	
FX_ALTA	DATE	
VENDEDOR	CHAR(6)	
CODPOSTAL	CHAR(5)	
SEXO	CHAR(1)	
FX_NACIMIENTO	DATE	
COMPRAS_ANUAL	DECIMAL(8,2)	
EMPRESA	INTEGER	

TIPOCLIE		
TIPO	CHAR(3)	NOT NULL
DS_TIPO	VARCHAR(25)	
LIMITE_COMPRAS	DECIMAL(8,2)	
RESPONSABLE	CHAR(6)	

TARJETAS		
CLIENTE	CHAR(6)	NOT NULL
NUM_TARJETA	CHAR(15)	NOT NULL
NOM_AUTORIZADO	CHAR(15)	NOT NULL
APE_AUTORIZADO	CHAR(20)	NOT NULL
FX_CADUCIDAD	DATE	
LIM_MES	SMALLINT	

VENDEDOR		
VENDEDOR	CHAR(6)	NOT NULL
NOMBRE	CHAR(15)	NOT NULL
APELLIDO	CHAR(20)	NOT NULL
FX_CONTRAT	DATE	
SALARIO	DECIMAL(8,2)	
COM_VENTAS	SMALLINT	
RESPONSABLE	CHAR(6)	

VALEDESC		
CLIENTE	CHAR(6)	NOT NULL
FX_VALE	DATE	NOT NULL
IMPORTE	DECIMAL(6,2)	

Capítulo 3 – Las funciones escalares y las operaciones

Objetivos

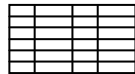
- ✓ **Aprender a utilizar correctamente las operaciones aritméticas en el SQL**
 - ✓ **Aritmética y funciones escalares de fechas y horas**
 - ✓ **Aprender el uso de algunas funciones escalares y saber cómo usar cualquier función escalar**
 - ✓ **Evitar los problemas de rendimiento que pueden desencadenar las funciones y las operaciones**
-

Uso de operaciones aritméticas en el SQL 1/3

"Total de compras de los dos últimos años de los clientes que en el año en curso hayan hecho compras por más de 1000 euros "

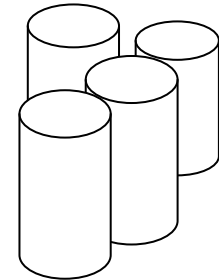


```
SELECT CLIENTE, COMP_AÑO_ACT,  
COMP_AÑO_ANT,  
COMP_AÑO_ACT + COMP_AÑO_ANT AS TOTAL  
FROM CLIENTES  
WHERE COMPRAS_AÑO_ACTUAL > 1000
```



CLIENTE	COMP_AÑO_ACT	COMP_AÑO_ANT	TOTAL
990	1250,45	978,00	2228.45
1540	2121,00	4010,00	6131,00

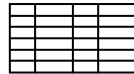
DB2



Uso de operaciones aritméticas en el SQL 2/3

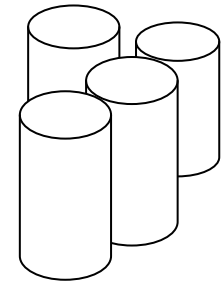
"Importe neto de las compras de los dos últimos años de los clientes que en el año en curso hayan hecho compras por más de 1000 euros "

```
SELECT CLIENTE,  
COMP_AÑO_ACT + COMP_AÑO_ANT AS TOTAL,  
((COMP_AÑO_ACT + COMP_AÑO_ANT) / 116) * 100  
AS NETO  
FROM CLIENTES  
WHERE COMP_AÑO_ACT > 1000
```



CLIENTE	TOTAL	NETO
990	2228.45	1921,0775
1540	6131,00	5285,3448

DB2



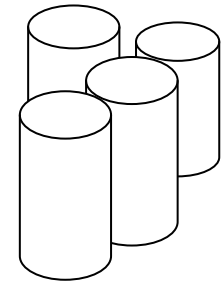
Uso de operaciones aritméticas en el SQL 3/3

"Clientes que hayan comprado un 20% más este año "



```
SELECT CLIENTE,
COMP_AÑO_ACT/COMP_AÑO_ANT AS PORC
FROM CLIENTES
WHERE COMP_AÑO_ACT/COMP_AÑO_ANT > 1,20
```

DB2



990	1,2785
-----	--------

Operandos tipo DATETIME y duraciones etiquetadas..(1)

Lista de duraciones etiquetadas (SQL Reference)

(1)	
>> function-invocation	YEAR <<
(expression)	YEARS
constant	MONTH
column-name	MONTHS
host-variable	DAY
	DAYS
	HOUR
	HOURS
	MINUTE
	MINUTES
	SECOND
	SECONDS
	MICROSECOND
	MICROSECONDS

Note:

(1) Includes all functions except table functions.

Operandos tipo DATETIME y duraciones etiquetadas .. (2)

La resta de datos tipo datetime se expresa en diferentes formatos:

hora – hora	→ time duration
fecha – fecha	→ date duration
timestamp – timestamp	→ timestamp duration

DATE DURATION

Una "date duration" representa un número de años, meses y días expresados como un número en DECIMAL(8,0). El número tiene el formato: yyymmdd

TIME DURATION

Una "time duration" representa un número de horas, minutos y segundos expresados como un número en DECIMAL(6,0). El número tiene el formato: hhmmss

TIMESTAMP DURATION

DECIMAL(20,6). El número tiene el formato: yyyyxxddhhmmsszzzzzz

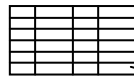
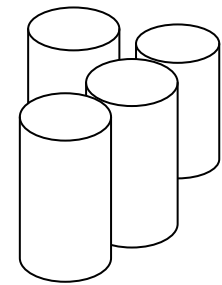
Operaciones con fechas

"Clientes que se hayan dado de alta con menos de 18 años"



```
SELECT CLIENTE, NOMBRE, APELLIDO1,  
FX_ALTA, FX_NACIMIENTO  
FROM CLIENTES  
WHERE FX_ALTA - FX_NACIMIENTO < 180000
```

DB2



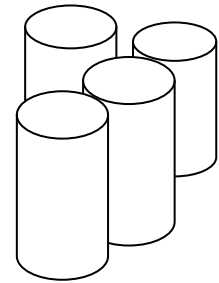
CLIENTE	NOMBRE	APELLIDO	FX_ALTA	FX_NACIMIENTO
770	EVA	RAMIREZ	2000-09-30	1983-05-26
990	ELENA	REZAGO	2000-08-15	1991-05-15
1100	TOMAS	GOTA	2000-06-19	1986-12-18

Operaciones con fechas ... 2



```
SELECT CLIENTE, NOMBRE,  
FX_ALTA, FX_ALTA + 1 YEAR + 10 MONTHS  
FROM CLIENTES
```

DB2



CLIENTE	NOMBRE	FX_ALTA	
770	EVA	2000-09-30	2002-08-30
990	ELENA	2000-08-15	2002-07-15
1100	TOMAS	2000-06-19	2002-05-19

Funciones escalares: POSSTR

>>—POSSTR (source-string, search-string)————><



```
SELECT ORGANISMO, POSSTR(ORGANISMO, 'DISTRITO') AS INI,  
       POSSTR(ORGANISMO, '-') AS FIN  
FROM PAPELEOS
```


ORGANISMO	INI	FIN
JUNTA MUNICIPAL DISTRITO TETUAN - 019	17	33
J.M. DEL DISTRITO PROSPERIDAD – 027	10	31
J.M. DISTRITO DOS HERMANAS – 034	6	28

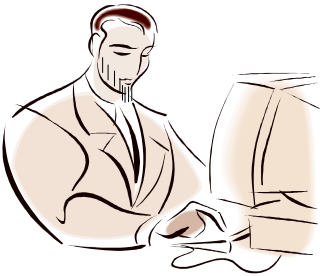
Funciones escalares: SUBSTR

```
>>—SUBSTR(string-expression, start—┐——><
                                     └┐, length┘
```



Funciones escalares: *SUBSTRING*

```
>>-SUBSTRING-(-string-expression-, -start-  
                                     |  
                                     |, -length-  
                                     |  
                                     | CODEUNITS32  
                                     | CODEUNITS16  
                                     | OCTETS  
                                     |  
-)><
```



- Desde la versión 7 los datos se pueden almacenar en Unicode.
- En la versión 8 los datos siempre se almacenan en unicode.
- La función substring es una mejora de substr
- Nos permite especificar que "start" y "length" cuentan:

CODEUNITS32	Unidades de 32-bit (UTF-32)
CODEUNITS16	Unidades de 16-bit (UTF-16)
OCTETS	Bytes.

Funciones escalares: *VALUE* / *COALESCE*

>>—**VALUE**—(expression—, expression—)——><



```
SELECT TELEFONO, VALUE (TELEFONO, 'NO TIENE')  
FROM CLIENTES  
WHERE CLIENTE > '2134'
```


TELEFONO	TLF
—	NO TIENE
979291935	979291935
696673695	696673695
—	NO TIENE
911999340	911999340

Funciones escalares: *VALUE* / *COALESCE* ...2



```
SELECT IMPORTE, PORC_DESC, (IMPORTE * PORC_DESC)/100 AS DESC,  
      (IMPORTE - (IMPORTE * PORC_DESC)/100) AS TOTAL  
FROM PEDIDOS  
WHERE FX_PEDIDO = '03-03-2006'
```


IMPORTE	PORC_DESC	DESC	TOTAL
128,10	10	12,81	115,29
210	5	10,5	199,5
121	—	—	—
90	8	7,2	82,8
115	—	—	—

Funciones escalares: *VALUE / COALESCE ...3*



```
SELECT IMPORTE, PORC_DESC,  
VALUE(((IMPORTE * PORC_DESC)/100),0) AS DESC,  
(IMPORTE - VALUE(((IMPORTE * PORC_DESC)/100),0) AS TOTAL  
FROM PEDIDOS  
WHERE FX_PEDIDO = '03-03-2006'
```


IMPORTE	PORC_DESC	DESC	TOTAL
128,10	10	12,81	115,29
210	5	10,5	199,5
121	—	0	121
90	8	7,2	82,8
115	—	0	115

Funciones escalares: RTRIM / LTRIM

>>—RTRIM(string-expression)—————><



```
SELECT TELEFONO, RTRIM(TELEFONO) AS TLF
FROM CLIENTES
WHERE TELEFONO IS NOT NULL
```


TELEFONO	TLF
84910	84910
84291	84291
658490610	658490610
88381	88381
979291935	979291935

Funciones escalares: *STRIP*

>>—STRIP (string-expression) —><

- , BOTH
- , B
- , LEADING
- , L
- , TRAILING
- , T

, strip-character



```
SELECT TELEFONO, STRIP (TELEFONO, TRAILING) AS TLF
FROM CLIENTES
WHERE TELEFONO IS NOT NULL
```

TELEFONO	TLF
84910	84910
84291	84291
658490610	658490610

Funciones escalares: VARIAS

LENGTH (expression)	Devuelve la longitud de la expresión
DIGITS (expression)	Convierte numéricos a caracteres
CONCAT ó 	Concatena dos campos
DECIMAL (expression, -----) -- integer-, --integer-	Para convertir una expresión a formato decimal(m,n) donde m es la escala y n la precisión.
CHAR (.....) <i>Ejemplo:</i> <i>CHAR (FX_PEDIDO, ISO)</i>	Permite convertir a cadena de caracteres: - una fecha (formato ISO, USA, EUR, etc.) - un entero - un decimal - etc.

"... el resto, en la SQL Reference "

Funciones escalares: VARIAS

```
SELECT CHAR(DECIMAL(:VISITAS_DIA,7,2))  
FROM SYSIBM.SYSDUMMY1;
```

VISITAS	INTEGER	10000
---------	---------	-------

'10000.00'

```
SELECT CHAR(SALARIO, ',', '  
FROM VENDEDOR WHERE SALARIO = 52750.00;
```

SALARIO	DECIMAL(9,2)
---------	--------------

'0052750,00'

```
SELECT LENGTH(SALARIO),  
LENGTH(DIGITS(SALARIO))  
FROM VENDEDOR WHERE SALARIO = 52750.00;
```

SALARIO	DECIMAL(9,2)
---------	--------------

5	9
---	---

```
SELECT NOM, APELL, APELL || ', ' || NOM  
FROM VENDEDOR WHERE VENDEDOR = 'V123';
```

LUCIA	GOMEZ	GOMEZ, LUCIA
-------	-------	--------------

Funciones escalares de fechas

```
>>-DAY (expression)—————><
>>-DAYOFWEEK (expression)————><
>>-DAYOFWEEK_ISO (expression)——><
>>-DAYOFMONTH (expression)———><
>>-DAYOFYEAR (expression)————><
>>-MONTH (expression)—————><
>>-YEAR (expression)—————><
>>-HOUR (expression)—————><
>>————...etc...————><
>>-DAYS (expression)—————><
```

Extrae el día de una fecha
Día de la semana: 1 a 7 (1 – Dom)
ISO: 1 es lunes

...

...

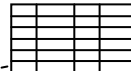
Número de días desde el 1-1-0001
hasta la fecha dada

"Dependiendo de las funciones, "expression" puede ser un DATE, o una representación de una fecha en CHAR, o una time duration, date duration ó timestamp duration "

Funciones escalares de fechas: Ejemplo



```
SELECT CLIENTE, NOMBRE, FX_ALTA, DAY(FX_ALTA) AS DIA,  
MONTH(FX_ALTA) AS MES, DAYOFWEEK(FX_ALTA) AS DIA_s,  
DAYOFWEEK_ISO(FX_ALTA) AS DIA_S_ISO,  
DAYOFYEAR(FX_ALTA) AS DIA_AÑO, DAYS(FX_ALTA) AS DIAS,  
DAYS(FX_ALTA) - DAYS(FX_NACIMIENTO) AS DIAS_EDAD  
FROM CLIENTES  
WHERE CLIENTE = '990'
```



CLIENTE	NOMBRE	FX_ALTA	DIA	MES	DIA_S	DIA_S_ISO	DIA_AÑO	DIAS	DIAS_EDAD
990	ELENA	2000-08-15	15	8	3	2	228	730347	6380

CAST : especificaciones y funciones

```
>>—CAST—(—expression—AS—data-type—)————><
                |
                |—NULL—
                |
                |—parameter-marker—
```

```
SELECT VENDEDOR, CAST(SALARIO AS INTEGER)
FROM VENDEDOR
```



```
CREATE DISTINCT TYPE EDAD AS DECIMAL(2,0);
```

Se crean dos **funciones CAST** automáticamente: una convierte de decimal a EDAD y la otra de EDAD a decimal.

Sintaxis 1: `CAST(:WS-AÑOS AS EDAD);`

Sintaxis 2: `AGE(:WS-AÑOS);`

[illegible]

Expresiones CASE - Ejemplos

```
SELECT VENDEDOR, NOMBRE,  
       CASE SUBSTR(VENDEDOR,1,2)  
       WHEN 'V0' THEN 'Prácticas'  
       WHEN 'V1' THEN 'Junior'  
       WHEN 'V2' THEN 'Senior'  
       ELSE 'Tipo erróneo'  
       END  
FROM VENDEDOR;
```

```
SELECT COMPRAS_AÑO, COMPRAS_AÑO_ANT,  
       CASE  
       WHEN COMPRAS_AÑO_ANT = 0 THEN 0  
       WHEN COMPRAS_AÑO_ANT > 0 THEN  
       (COMPRAS_AÑO / COMPRAS_AÑO_ANT)  
       END AS VARIACION  
FROM COMPRAS_CLIENTE
```

Uso de funciones escalares

- ✓ Pueden ahorrar codificación adicional en los programas
- ✓ Bien usadas pueden suponer una mejora en el rendimiento de los procesos
- ✓ Mal usadas empeoran drásticamente el rendimiento
- ✓ Pueden existir diferentes funciones en diferentes entornos de ejecución –Unix, z/OS, AS/400, etc.-
- ✓ Hay funciones de tratamiento de cadenas, funciones matemáticas, funciones de tratamiento de fechas, funciones de integración con MQSeries, funciones de integración / conversión a XML, etc.

"Hemos visto sólo ejemplos de unas pocas funciones ... "

Ejercicio 3

- ✓ Al modelo de datos del ejercicio anterior se añade la tabla **COMPCLIE**, que contiene las compras de cada cliente. Se presentan las compras totales del año en curso y de los cuatro anteriores.
-

Ejercicio 3 – Tabla agregada

CLIENTES		
CLIENTE	CHAR(6)	NOT NULL
NOMBRE	CHAR(15)	NOT NULL
APELLIDO1	CHAR(20)	NOT NULL
APELLIDO2	CHAR(20)	NOT NULL
TIPO	CHAR(3)	
TELEFONO	CHAR(9)	
FX_ALTA	DATE	
VENDEDOR	CHAR(6)	
CODPOSTAL	CHAR(5)	
SEXO	CHAR(1)	
COMPRAS_ANUAL	DECIMAL(8,2)	
FX_NACIMIENTO	DATE	

TARJETAS		
CLIENTE	CHAR(6)	NOT NULL
NUM_TARJETA	CHAR(15)	NOT NULL
NOM_AUTORIZADO	CHAR(15)	NOT NULL
APE_AUTORIZADO	CHAR(20)	NOT NULL
FX_CADUCIDAD	DATE	
LIM_MES	SMALLINT	

TIPOCLIE		
TIPO	CHAR(3)	NOT NULL
DS_TIPO	VARCHAR(25)	
LIMITE_COMPRAS	DECIMAL(8,2)	
RESPONSABLE	CHAR(6)	

COMPCLIE		
CLIENTE	CHAR(6)	NOT NULL,
AÑO_ACTUAL	DECIMAL(8,2)	
AÑO_ANT	DECIMAL(8,2)	
AÑO_DOS_ANT	DECIMAL(8,2)	
AÑO_TRE_ANT	DECIMAL(8,2)	
AÑO_CUA_ANT	DECIMAL(8,2)	

VENDEDOR		
VENDEDOR	CHAR(6)	NOT NULL
NOMBRE	CHAR(15)	NOT NULL
APELLIDO	CHAR(20)	NOT NULL
FX_CONTRAT	DATE	
SALARIO	DECIMAL(8,2)	
COM_VENTAS	SMALLINT	
RESPONSABLE	CHAR(6)	

VALEDESC		
CLIENTE	CHAR(6)	NOT NULL
FX_VALE	DATE	NOT NULL
IMPORTE	DECIMAL(6,2)	

Ejercicio 3 – Tabla agregada

CLIENTES		
CLIENTE	CHAR(6)	NOT NULL
NOMBRE	CHAR(15)	NOT NULL
APELLIDO1	CHAR(20)	NOT NULL
APELLIDO2	CHAR(20)	NOT NULL
TIPO	CHAR(3)	
TELEFONO	CHAR(9)	
FX_ALTA	DATE	
VENDEDOR	CHAR(6)	
CODPOSTAL	CHAR(5)	
SEXO	CHAR(1)	
COMPRAS_ANUAL	DECIMAL(8,2)	
FX_NACIMIENTO	DATE	

TARJETAS		
CLIENTE	CHAR(6)	NOT NULL
NUM_TARJETA	CHAR(15)	NOT NULL
NOM_AUTORIZADO	CHAR(15)	NOT NULL
APE_AUTORIZADO	CHAR(20)	NOT NULL
FX_CADUCIDAD	DATE	
LIM_MES	SMALLINT	

TIPOCLIE		
TIPO	CHAR(3)	NOT NULL
DS_TIPO	VARCHAR(25)	
LIMITE_COMPRAS	DECIMAL(8,2)	
RESPONSABLE	CHAR(6)	

COMPCLIE		
CLIENTE	CHAR(6)	NOT NULL,
AÑO_ACTUAL	DECIMAL(8,2)	
AÑO_ANT	DECIMAL(8,2)	
AÑO_DOS_ANT	DECIMAL(8,2)	
AÑO_TRE_ANT	DECIMAL(8,2)	
AÑO_CUA_ANT	DECIMAL(8,2)	

VENDEDOR		
VENDEDOR	CHAR(6)	NOT NULL
NOMBRE	CHAR(15)	NOT NULL
APELLIDO	CHAR(20)	NOT NULL
FX_CONTRAT	DATE	
SALARIO	DECIMAL(8,2)	
COM_VENTAS	SMALLINT	
RESPONSABLE	CHAR(6)	

VALEDESC		
CLIENTE	CHAR(6)	NOT NULL
FX_VALE	DATE	NOT NULL
IMPORTE	DECIMAL(6,2)	

Capítulo 4 – Funciones de agrupación

Objetivos

- ✓ **Uso del GROUP BY**
 - ✓ **Manejo de la cláusula HAVING**
 - ✓ **Nuevas combinaciones con las cláusulas de grupo**
 - ✓ **Distinguir las diferentes funciones de agregación**
-

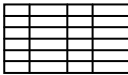
Lista de funciones de agrupación

AVG	Obtiene la media
COUNT, COUNT_BIG	Cuenta filas
MAX	Valor máximo
MIN	Valor mínimo
STDDEV	Desviación típica
SUM	Suma de valores
VARIANCE	Varianza

Funciones de agregación: ejemplo



```
SELECT SUM(IMPORTE_PEDIDO) , AVG (IMPORTE_PEDIDO) ,  
STDDEV (IMPORTE_PEDIDO) , VARIANCE (IMPORTE_PEDIDO)  
FROM PEDIDOS
```

			
COL1	COL2	COL3	COL4
3.274.554	152,30	36,45	1,12

Funciones de agregación: ejemplo 2



```
SELECT MAX(IMPORTE_PEDIDO) , MIN(IMPORTE_PEDIDO) ,  
COUNT(*) , COUNT(DISTINCT IMPORTE_PEDIDO)  
FROM PEDIDOS
```

A diagram showing a small table with 5 rows and 4 columns positioned above a larger table with 2 rows and 4 columns. Dashed lines connect the top-left and top-right corners of the small table to the top-left and top-right corners of the larger table, suggesting a relationship or transformation.

COL1	COL2	COL3	COL4
1534,00	2,5	21.500	21.124

Uso de agrupaciones por filas



```
SELECT SUM(IMPORTE_PEDIDO) AS TOTAL_AÑO,  
       AVG(IMPORTE_PEDIDO) AS IMPORTE_MEDIO  
FROM PEDIDOS  
WHERE DEPART = 'HOGAR'  
AND FX_PEDIDO BETWEEN '01/01/2005' AND '31/12/2005'
```


TOTAL_AÑO	IMPORTE_MEDIO
390.245	180,20

La cláusula GROUP BY



```
SELECT DS_DEPART, IMPORTE_PEDIDO
FROM PEDIDOS
WHERE YEAR(FX_PEDIDO) = 2005
```

DS_DEPART	IMPORTE_PEDIDO
HOGAR	129,50
INFORMATICA	38,50
CALZADO	68,95
HOGAR	2129,00
HOGAR	916,00
DEPORTES	45,00
INFORMATICA	245,00
HOGAR	69,90
HOGAR	225,00
INFORMATICA	890,00
CALZADO	18,95
...	...

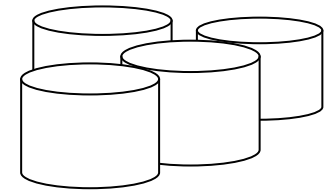
La cláusula GROUP BY

*"Importe total y medio
de los pedidos por
departamento"*



```
SELECT DS_DEPART,  
SUM(IMPORTE_PEDIDO) AS TOTAL_AÑO,  
AVG(IMPORTE_PEDIDO) AS IMPORTE_MEDIO  
FROM PEDIDOS  
WHERE YEAR(FX_PEDIDO) = 2005  
GROUP BY DS_DEPART  
ORDER BY DS_DEPART
```

DB2



DS_DEPART	TOTAL_AÑO	IMPORTE_MEDIO
CALZADO	93.213,20	29,00
DEPORTES	115.190,30	45,50
HOGAR	390.245,55	180,20
INFORMATICA	290.155,45	39,50
...

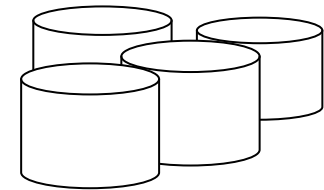
La cláusula **HAVING** (+GROUP BY)

"Importe total y medio de los pedidos por departamento de aquellos que tengan una media mayor a 40 euros"



```
SELECT DS_DEPART,  
SUM(IMPORTE_PEDIDO) AS TOTAL_AÑO,  
AVG(IMPORTE_PEDIDO) AS IMPORTE_MEDIO  
FROM PEDIDOS  
WHERE YEAR(FX_PEDIDO) = 2005  
GROUP BY DS_DEPART  
HAVING AVG(IMPORTE_PEDIDO) > 40
```

DB2



GROUP BY {
HAVING {

DS_DEPART	TOTAL_AÑO	IMPORTE_MEDIO
HOGAR	390.245,55	180,20
DEPORTES	115.190,30	45,50
ROPA MUJER	214.230	65,00
INFORMATICA	290.155,45	39,501
CALZADO	93.213,20	29,00
ROPA NIÑO	85.234,20	32,22
...

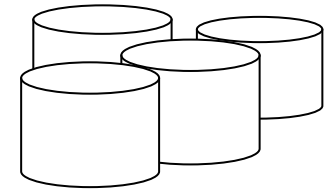
La cláusula **HAVING** (+GROUP BY)

"Lista de los vendedores que hayan vendido más de 4000 euros en un mes en 2005"



```
SELECT VENDEDOR,  
MONTH(FX_PEDIDO) AS MES,  
SUM(IMPORTE_PEDIDO) AS TOTAL_MES  
FROM PEDIDOS  
WHERE YEAR(FX_PEDIDO) = 2005  
GROUP BY VENDEDOR, MONTH(FX_PEDIDO)  
HAVING SUM(IMPORTE_PEDIDO) > 4000
```

DB2

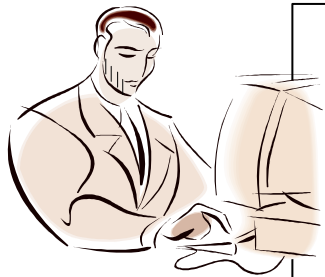


GROUP BY {
HAVING {

VENDEDOR	MES	TOTAL_MES
V213	12	7230,50
V234	12	6500
V211	1	4250
V213	1	3210
V234	1	2950
V211	12	2100
...

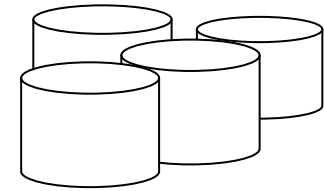
La cláusula **HAVING** (+GROUP BY)

"Lista de las facturas duplicadas"



```
SELECT CLIENTE, NU_FACTURA,  
COUNT(*) AS CANT  
FROM FACTURAS  
GROUP BY CLIENTE, NU_FACTURA  
HAVING COUNT(*) > 1
```

DB2



GROUP BY {
HAVING {

CLIENTE	FACTURA	CANT
890	2734	2
990	2735	2
1100	2736	2
890	1135	1
990	1136	1
1100	1137	1
...

Ejercicio 4

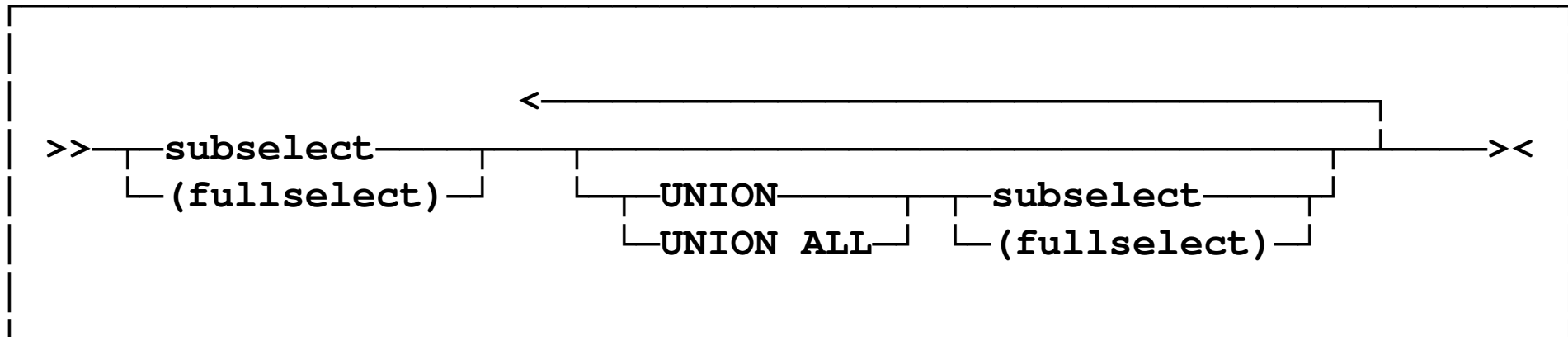
Capítulo 5 –UNION, INTERSECT y MINUS



Objetivos

- ✓ **Definir una fullselect**
 - ✓ **Conocer la cláusula UNION**
 - ✓ **Distinguir UNION y UNION ALL**
 - ✓ **Situaciones en las que usar UNION [ALL]**
 - ✓ **Cláusulas INTERSECT y MINUS**
-

Fullselect



"Una fullselect es un componente de una sentencia-select, CREATE VIEW ó sentencia INSERT. (...)

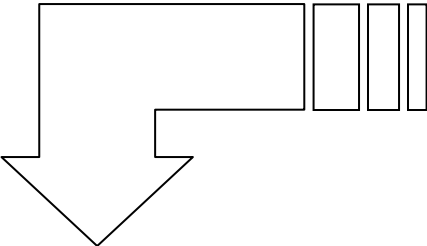
(...) Si no se usa la cláusula UNION, la fullselect es lo mismo que la subselect (...)"

UNION vs. UNION ALL

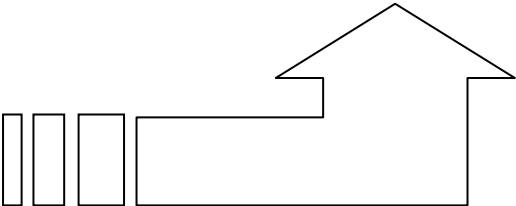
- ✓ **Permiten ejecutar dos select como una misma sentencia**
 - ✓ **Útil para recuperar información de dos tablas idénticas**
 - ✓ **UNION elimina filas duplicadas en las SELECT**
 - ✓ **UNION ALL devuelve todas las filas sin quitar duplicados**
 - ✓ **Las dos selects deben tener el mismo número de columnas, y las columnas deben tener tipos de datos compatibles**
 - ✓ **El ORDER BY siempre provoca una clasificación**
-

UNION

NOMBRE	APELLIDO1	TELEFONO
ANGEL	RENILLA	645535347
MARCOS	GOMEZ	980546233
LUIS	RAPINO	942445656



SELECT NOMBRE, APELLIDO1, TELEFONO
FROM CLIENTES
UNION
SELECT NOMBRE, APELLIDO, TELEFONO
FROM VENDEDOR

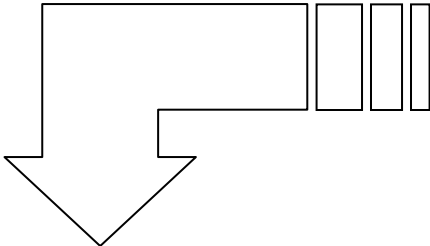


NOMBRE	APELLIDO1	TELEFONO
ANGELA	MARIA	971232323
ANGEL	RENILLA	645535347
ANA	MARQUEZ	-

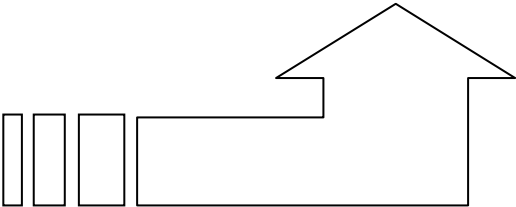
NOMBRE	APELLIDO1	TELEFONO
ANA	MARQUEZ	-
ANGEL	RENILLA	645535347
ANGELA	MARIA	971232323
LUIS	RAPINO	942445656
MARCOS	GOMEZ	980546233

UNION ALL

NOMBRE	APELLIDO1	TELEFONO
ANGEL	RENILLA	645535347
MARCOS	GOMEZ	980546233
LUIS	RAPINO	942445656



SELECT NOMBRE, APELLIDO1, TELEFONO
FROM CLIENTES
UNION ALL
SELECT NOMBRE, APELLIDO, TELEFONO
FROM VENDEDOR



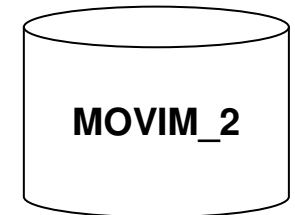
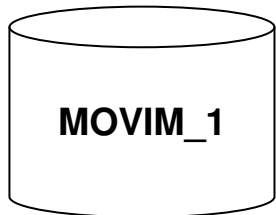
NOMBRE	APELLIDO1	TELEFONO
ANGELA	MARIA	971232323
ANGEL	RENILLA	645535347
ANA	MARQUEZ	-

NOMBRE	APELLIDO1	TELEFONO
ANA	MARQUEZ	-
ANGEL	RENILLA	645535347
ANGEL	RENILLA	645535347
ANGELA	MARIA	971232323
LUIS	RAPINO	942445656
MARCOS	GOMEZ	980546233

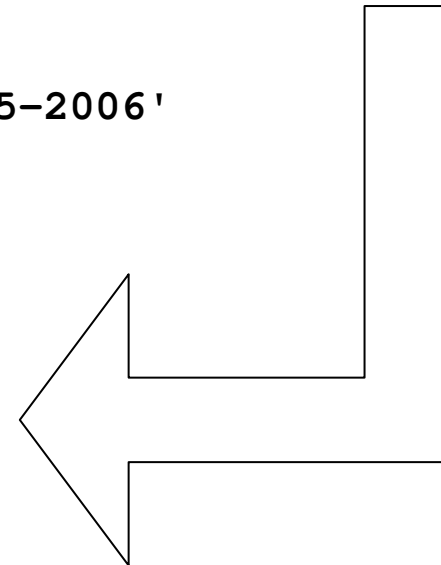
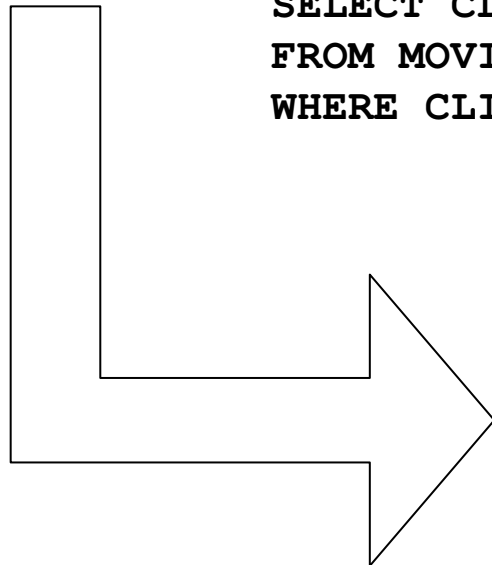
Usos de UNION [ALL]



**"Movimientos del cliente '012312'
el día 03-05-2006 "**



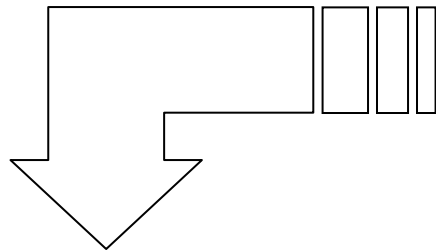
```
SELECT CLIENTE, NU_MOV, IMPORTE
FROM MOVIM_1
WHERE CLIENTE = '012312' AND FX_MOV = '03-05-2006'
UNION
SELECT CLIENTE, NU_MOV, IMPORTE
FROM MOVIM_2
WHERE CLIENTE = '012312' AND FX_MOV = '03-05-2006'
```



CLIENTE	NU_MOV	IMPORTE
012312	012453	-100,00
012312	012787	1200,00
012312	012912	-73,13

"Recuperar información de dos tablas idénticas "

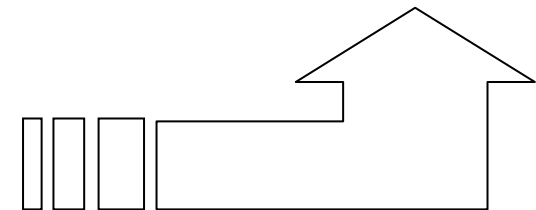
INTERSECT*



```
SELECT NOMBRE, APELLIDO1, TELEFONO
FROM CLIENTES
INTERSECT
SELECT NOMBRE, APELLIDO, TELEFONO
FROM VENDEDOR
```

NOMBRE	APELLIDO1	TELEFONO
ANGEL	RENILLA	645535347
MARCOS	GOMEZ	980546233
LUIS	RAPINO	942445656

NOMBRE	APELLIDO1	TELEFONO
ANGELA	MARIA	971232323
ANGEL	RENILLA	645535347
ANA	MARQUEZ	-

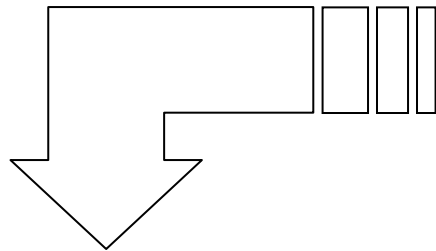


NOMBRE	APELLIDO1	TELEFONO
ANGEL	RENILLA	645535347

"Recuperamos la información que está en ambas SELECT "

* Sólo disponible en Oracle

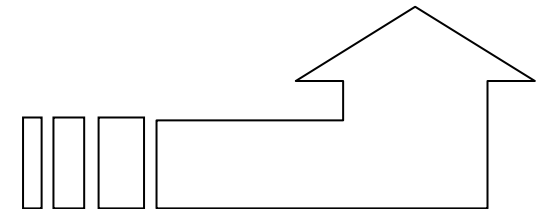
MINUS*



SELECT NOMBRE, APELLIDO1, TELEFONO
FROM CLIENTES
MINUS
SELECT NOMBRE, APELLIDO, TELEFONO
FROM VENDEDOR

NOMBRE	APELLIDO1	TELEFONO
ANGEL	RENILLA	645535347
MARCOS	GOMEZ	980546233
LUIS	RAPINO	942445656

NOMBRE	APELLIDO1	TELEFONO
ANGELA	MARIA	971232323
ANGEL	RENILLA	645535347
ANA	MARQUEZ	-



NOMBRE	APELLIDO1	TELEFONO
ANGELA	MARIA	971232323
ANA	MARQUEZ	-

"Elimina de la primera SELECT las filas que están repetidas en la segunda SELECT "

** Sólo disponible en Oracle*

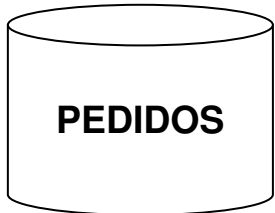
Ejercicio 5

Capítulo 6 – El JOIN: Selección de filas de varias tablas

Objetivos

- ✓ **Mecánica del JOIN**
 - ✓ **Diferenciar las sintaxis dentro del JOIN**
 - ✓ **Escribir sentencias JOIN legibles y estandarizadas**
 - ✓ **Conocer el inner JOIN**
 - ✓ **Diferenciar los diferentes tipos de OUTER JOIN**
 - ✓ **Join de múltiples tablas**
-

Recuperación de filas desde varias tablas



Opciones para recuperar columnas de dos tablas:

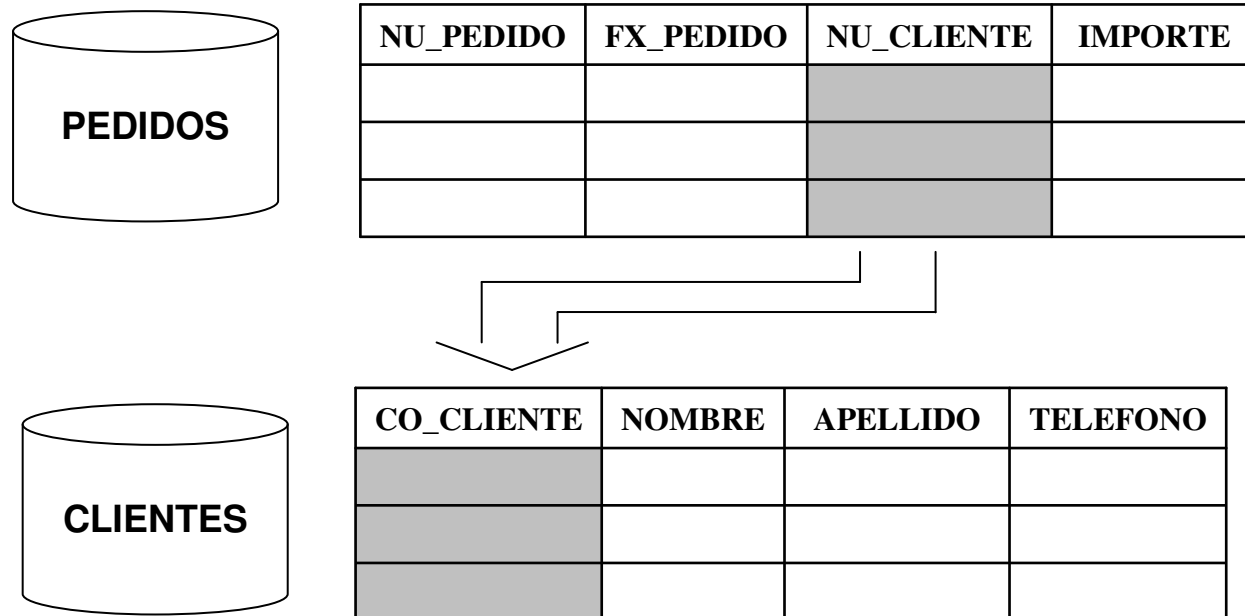
- 1. Dos cursores (de tabla única)**
- 2. Un cursor que ataque a las dos tablas (JOIN)**

Siempre debe existir una columna "común" en ambas tablas

Diagrama de una consulta JOIN entre las tablas PEDIDOS y CLIENTES. Se muestra una consulta en SQL que une las tablas PEDIDOS y CLIENTES por la columna CLIENTE. El resultado es una tabla con 4 columnas: CLIENTE, NOMBRE, NU_PEDIDO y IMPORTE. La consulta se muestra en un cuadro de texto flotante sobre la tabla de resultados.

CLIENTE	NOMBRE	NU_PEDIDO	IMPORTE

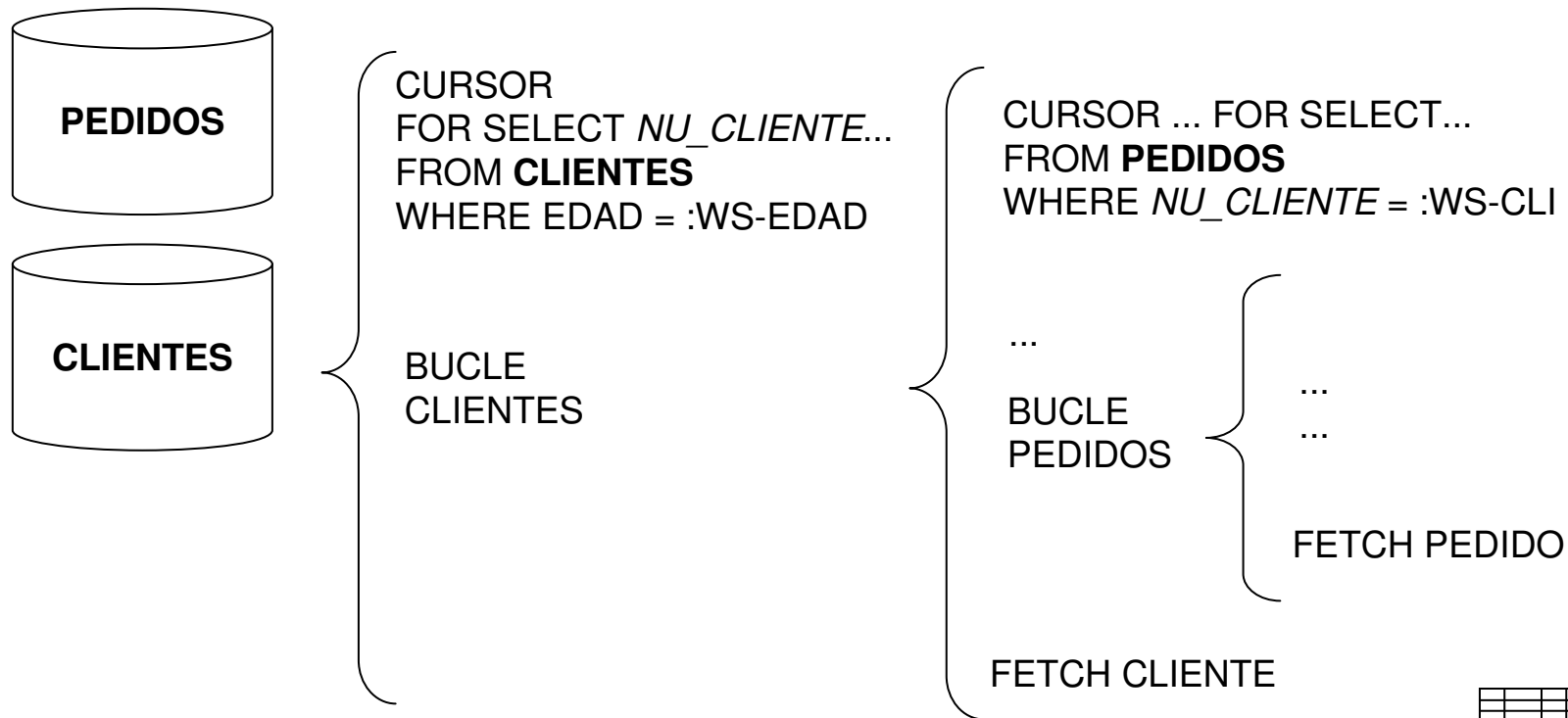
Mecánica del JOIN



"La columna "común" en ambas tablas es sobre la que se escribirá el predicado de join"

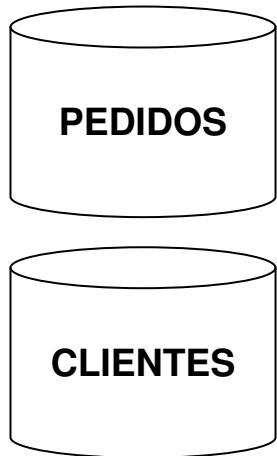
"Pueden tener diferente nombre de columna, pero el mismo contenido"

Recuperación de filas desde varias tablas: dos cursores



CLIENTE	NOMBRE	NU_PEDIDO	IMPORTE
127	ANA	012453	32
213	LUIS	012787	45
423	ANGEL	012912	18

Recuperación de filas desde varias tablas: JOIN



```
CURSOR PEDI-CLI... FOR  
SELECT...  
FROM CLIENTES JOIN PEDIDOS  
ON NU_CLIENTE = CLIENTE_PED  
WHERE EDAD = :WS-EDAD
```

...

```
BUCLE  
PEDI-CLI
```

...

```
FETCH PEDI-CLI
```


CLIENTE	NOMBRE	NU_PEDIDO	IMPORTE
127	ANA	012453	32
213	LUIS	012787	45
423	ANGEL	012912	18

La sintaxis del JOIN

Sintaxis antigua:

```
SELECT A.CLIENTE, NOMBRE, APELLIDO, PEDIDO, FX_PEDIDO
FROM CLIENTES A, PEDIDOS B
WHERE A.NU_CLIENTE = B.NU_CLIENTE
AND FX_PEDIDO = '24-04-2006'
```

EI INNER JOIN

EMPLEADO

NU_EMPL	NOMBRE	DEPT
101	ANA	B1
102	LUIS	B2
103	EVA	-

DEPARTAMENTO

DEPT	NOM_DEPT	...
B1	RR.HH.	
B2	MARKETING	
B3	VENTAS	

```
SELECT NU_EMPL, NOMBRE, DEPT, NOM_DEPT
FROM EMPLEADO E INNER JOIN DEPARTAMENTO D
ON E.DEPT = D.DEPT
[WHERE ...]
```


NU_EMPL	NOMBRE	DEPT	NOM_DEPT
101	ANA	B1	RR.HH.
102	LUIS	B2	MARKETING

EI LEFT OUTER JOIN

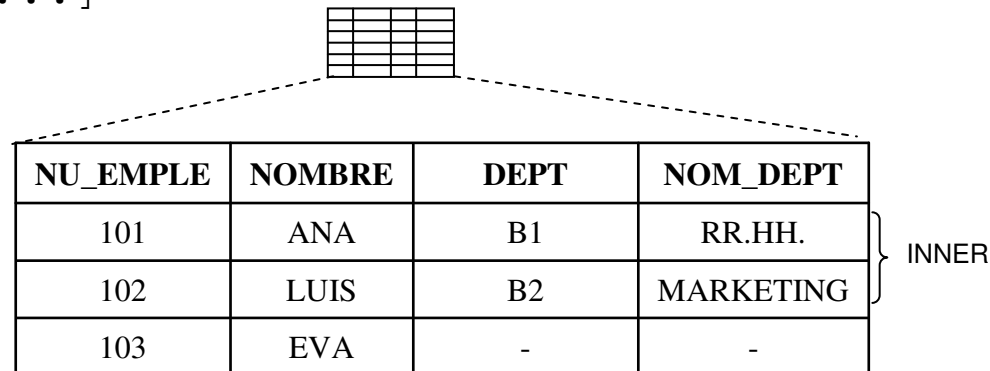
EMPLEADO

NU_EMPLE	NOMBRE	DEPT
101	ANA	B1
102	LUIS	B2
103	EVA	-

DEPARTAMENTO

DEPT	NOM_DEPT	...
B1	RR.HH.	
B2	MARKETING	
B3	VENTAS	

```
SELECT NU_EMPLE, NOMBRE, DEPT, NOM_DEPT
FROM EMPLEADO E LEFT OUTER JOIN DEPARTAMENTO D
ON E.DEPT = D.DEPT
[WHERE ...]
```



NU_EMPLE	NOMBRE	DEPT	NOM_DEPT
101	ANA	B1	RR.HH.
102	LUIS	B2	MARKETING
103	EVA	-	-

EI RIGHT OUTER JOIN

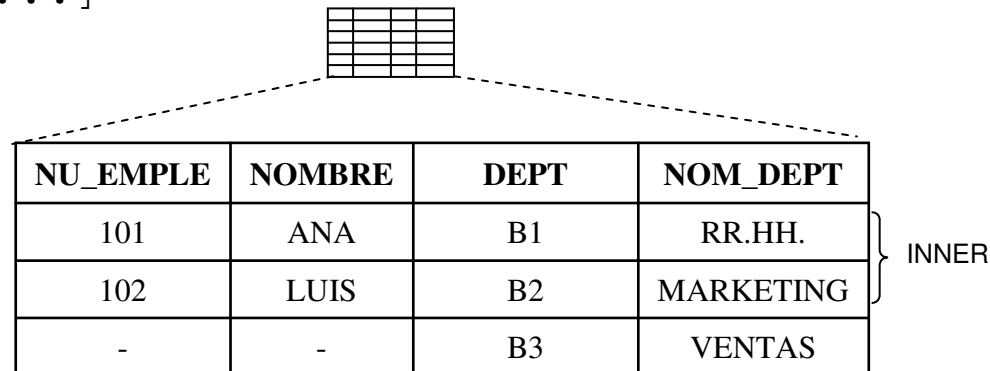
EMPLEADO

NU_EMPLE	NOMBRE	DEPT
101	ANA	B1
102	LUIS	B2
103	EVA	-

DEPARTAMENTO

DEPT	NOM_DEPT	...
B1	RR.HH.	
B2	MARKETING	
B3	VENTAS	

```
SELECT NU_EMPLE, NOMBRE, DEPT, NOM_DEPT
FROM EMPLEADO E RIGTH OUTER JOIN DEPARTAMENTO D
ON E.DEPT = D.DEPT
[WHERE ...]
```



NU_EMPLE	NOMBRE	DEPT	NOM_DEPT
101	ANA	B1	RR.HH.
102	LUIS	B2	MARKETING
-	-	B3	VENTAS

EI FULL OUTER JOIN

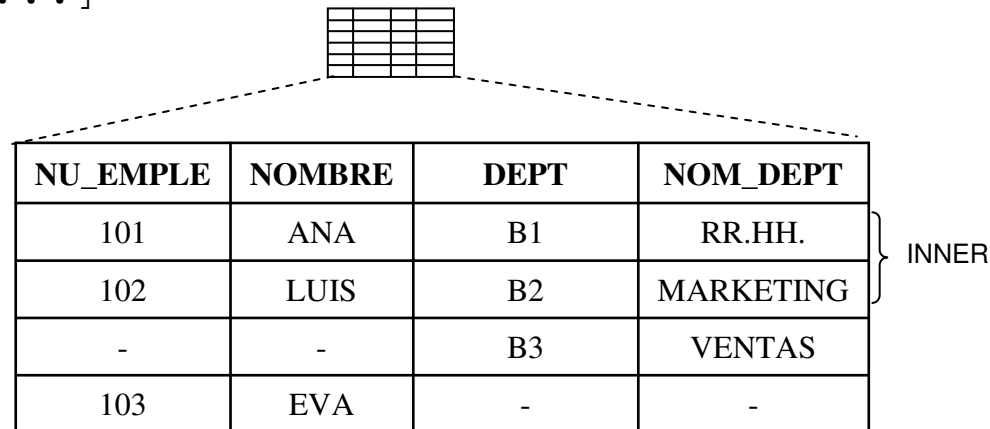
EMPLEADO

NU_EMPLE	NOMBRE	DEPT
101	ANA	B1
102	LUIS	B2
103	EVA	-

DEPARTAMENTO

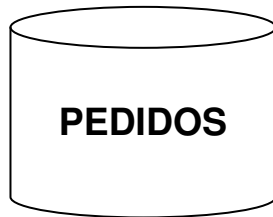
DEPT	NOM_DEPT	...
B1	RR.HH.	
B2	MARKETING	
B3	VENTAS	

```
SELECT NU_EMPLE, NOMBRE, DEPT, NOM_DEPT
FROM EMPLEADO E FULL OUTER JOIN DEPARTAMENTO D
ON E.DEPT = D.DEPT
[WHERE ...]
```



NU_EMPLE	NOMBRE	DEPT	NOM_DEPT
101	ANA	B1	RR.HH.
102	LUIS	B2	MARKETING
-	-	B3	VENTAS
103	EVA	-	-

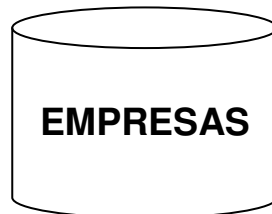
JOIN de más de dos tablas



NU_PEDIDO	FX_PEDIDO	NU_CLIENTE	IMPORTE



CO_CLIENTE	NOMBRE	APELLIDO	TELEFONO	CO_EMPRESA

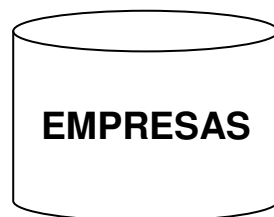


CO_EMPRESA	NOMBRE	CIF	CLIENTE_CONTACTO

JOIN cíclico



CO_CLIENTE	NOMBRE	APELLIDO	TELEFONO	CO_EMPRESA
27				5



CO_EMPRESA	NOMBRE	CIF	INTERLOCUTOR
5			36



CO_CLIENTE	NOMBRE	APELLIDO	TELEFONO	CO_EMPRESA
36				

¿ Es mejor la nueva sintaxis ?

Sintaxis antigua:

```
SELECT A.CLIENTE, NOMBRE, APELLIDO, PEDIDO, FX_PEDIDO
FROM CLIENTES A, PEDIDOS B, FACTURAS F, TARJETAS T
WHERE A.NU_CLIENTE = B.NU_CLIENTE
AND F.NU_PEDIDO_REF = B.NU_PEDIDO
AND T.NU_TARJETA = F.NU_TARJETA
AND FX_PEDIDO = '24-04-2006'
```

"Nueva" sintaxis:

```
SELECT A.CLIENTE, NOMBRE, APELLIDO, PEDIDO, FX_PEDIDO
FROM CLIENTES A INNER JOIN PEDIDOS B
ON A.NU_CLIENTE = B.NU_CLIENTE
INNER JOIN FACTURAS F
ON F.NU_PEDIDO_REF = B.NU_PEDIDO
INNER JOIN TARJETAS T
ON T.NU_TARJETA = F.NU_TARJETA
WHERE FX_PEDIDO = '24-04-2006'
```

PROS y CONTRAS de los joins

VARIOS CURSORES		JOIN	
A FAVOR	EN CONTRA	A FAVOR	EN CONTRA
	<i>Son varias SQLs que se ejecutan en serie</i>	<i>Es una única SQL que puede paralelizarse</i>	
<i>Fáciles de mantener</i>			<i>Un join de más de tres tablas empieza a ser "complicado" de mantener</i>
<i>El cursor puede dar mal rendimiento</i>			<i>El join puede dar muy mal rendimiento (producto cartesiano de accesos)</i>
	<i>No son tan rápidos como un buen join</i>	<i>Bien usados son más rápidos que los cursores</i>	

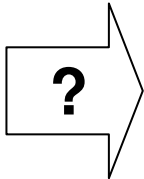
Notas sobre los joins

```
SELECT A.CLIENTE, NOMBRE, APELLIDO, PEDIDO, FX_PEDIDO
FROM CLIENTES A INNER JOIN PEDIDOS B
ON A.NU_CLIENTE = B.NU_CLIENTE
WHERE FX_PEDIDO = '24-04-2006'
```

"DB2 aplica primero* los predicados locales para limitar el número de filas para hacer el join"

"Posteriormente aplica los predicados de join"

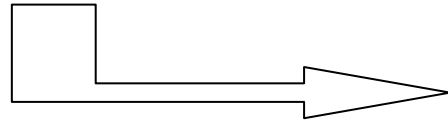
```
SELECT A.CLIENTE, NOMBRE, APELLIDO, PEDIDO, FX_PEDIDO
FROM CLIENTES A LEFT OUTER JOIN PEDIDOS B
ON A.NU_CLIENTE = B.NU_CLIENTE
AND FX_PEDIDO = '24-04-2006'
```



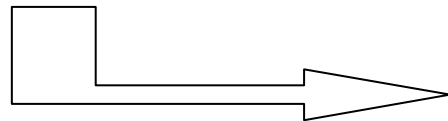
* - Dependiendo del tipo y método de ejecución del join

No es lo mismo...

```
SELECT C.CLIENTE, C.NOMBRE, C.APELLIDO1,  
V.VENDEDOR AS VEND,  
V.FX_CONTRAT AS ANTIG_VEND  
FROM CLIENTES C LEFT OUTER JOIN VENDEDOR V  
ON C.VENDEDOR = V.VENDEDOR  
AND FX_CONTRAT = '2003-11-21'
```



```
SELECT C.CLIENTE, C.NOMBRE, C.APELLIDO1,  
V.VENDEDOR AS VEND,  
V.FX_CONTRAT AS ANTIG_VEND  
FROM CLIENTES C LEFT OUTER JOIN VENDEDOR V  
ON C.VENDEDOR = V.VENDEDOR  
WHERE FX_CONTRAT = '2003-11-21'
```



CLIENTE	NOMBRE	APELLIDO	VEND	ANTIG_VEND
1210	JAIME	TAPADO	V231	
3740	JESUS	AYUSO	V123	
1870	JORGE	GARCILASO	V212	
1650	JOSE	TOPON	V134	
550	JUAN	RENATE	V342	2003-11-21
3190	JUAN	ABELLAN	V453	
2640	JUAN JOSE	RUIZ	V435	
1980	LEYRE	TEZ	V342	2003-11-21
3630	LUIS	SOTO	V212	
660	LUIS	TEMPORA	V342	2003-11-21

CLIENTE	NOMBRE	APELLIDO	VEND	ANTIG_VEND
550	JUAN	RENATE	V342	2003-11-21
1980	LEYRE	TEZ	V342	2003-11-21
660	LUIS	TEMPORA	V342	2003-11-21

Ejercicio 6

✓ Al modelo de datos de los ejercicios anteriores se añade ahora la tabla **EMPRESAS**, que contiene información de los clientes que son una empresa (dirección, CIF, interlocutor, etc.).

Obs: el interlocutor será uno de los clientes que están ya en nuestra tabla de Clientes

Ejercicio 6 – Tabla agregada: EMPRESAS

CLIENTES		
CLIENTE	CHAR(6)	NOT NULL
NOMBRE	CHAR(15)	NOT NULL
APELLIDO1	CHAR(20)	NOT NULL
APELLIDO2	CHAR(20)	NOT NULL
TIPO	CHAR(3)	
TELEFONO	CHAR(9)	
FX_ALTA	DATE	
VENDEDOR	CHAR(6)	
CODPOSTAL	CHAR(5)	
SEXO	CHAR(1)	
FX_NACIMIENTO	DATE	
COMPRAS_ANUAL	DECIMAL(8,2)	
EMPRESA	INTEGER	

COMPCLIE		
CLIENTE	CHAR(6)	NOT NULL
AÑO_ACTUAL	DECIMAL(8,2)	
AÑO_ANT	DECIMAL(8,2)	
AÑO_DOS_ANT	DECIMAL(8,2)	
AÑO_TRE_ANT	DECIMAL(8,2)	
AÑO_CUA_ANT	DECIMAL(8,2)	

TIPOCLIE		
TIPO	CHAR(3)	NOT NULL
DS_TIPO	VARCHAR(25)	
LIMITE_COMPRAS	DECIMAL(8,2)	
RESPONSABLE	CHAR(6)	

EMPRESAS		
CO_EMPRESA	INTEGER	NOT NULL
DS_EMPRESA	CHAR(40)	NOT NULL
CIF	CHAR(12)	NOT NULL
INTERLOCUTOR	CHAR(6)	
DIRECCION	VARCHAR(50)	
PROVINCIA	CHAR(2)	

TARJETAS		
CLIENTE	CHAR(6)	NOT NULL
NUM_TARJETA	CHAR(15)	NOT NULL
NOM_AUTORIZADO	CHAR(15)	NOT NULL
APE_AUTORIZADO	CHAR(20)	NOT NULL
FX_CADUCIDAD	DATE	
LIM_MES	SMALLINT	

VENDEDOR		
VENDEDOR	CHAR(6)	NOT NULL
NOMBRE	CHAR(15)	NOT NULL
APELLIDO	CHAR(20)	NOT NULL
FX_CONTRAT	DATE	
SALARIO	DECIMAL(8,2)	
COM_VENTAS	SMALLINT	
RESPONSABLE	CHAR(6)	

VALEDESC		
CLIENTE	CHAR(6)	NOT NULL
FX_VALE	DATE	NOT NULL
IMPORTE	DECIMAL(6,2)	

Ejercicio 6 – Tabla agregada: EMPRESAS

CLIENTES		
CLIENTE	CHAR(6)	NOT NULL
NOMBRE	CHAR(15)	NOT NULL
APELLIDO1	CHAR(20)	NOT NULL
APELLIDO2	CHAR(20)	NOT NULL
TIPO	CHAR(3)	
TELEFONO	CHAR(9)	
FX_ALTA	DATE	
VENDEDOR	CHAR(6)	
CODPOSTAL	CHAR(5)	
SEXO	CHAR(1)	
FX_NACIMIENTO	DATE	
COMPRAS_ANUAL	DECIMAL(8,2)	
EMPRESA	INTEGER	

COMPCLIE		
CLIENTE	CHAR(6)	NOT NULL
AÑO_ACTUAL	DECIMAL(8,2)	
AÑO_ANT	DECIMAL(8,2)	
AÑO_DOS_ANT	DECIMAL(8,2)	
AÑO_TRE_ANT	DECIMAL(8,2)	
AÑO_CUA_ANT	DECIMAL(8,2)	

TIPOCLIE		
TIPO	CHAR(3)	NOT NULL
DS_TIPO	VARCHAR(25)	
LIMITE_COMPRAS	DECIMAL(8,2)	
RESPONSABLE	CHAR(6)	

EMPRESAS		
CO_EMPRESA	INTEGER	NOT NULL
DS_EMPRESA	CHAR(40)	NOT NULL
CIF	CHAR(12)	NOT NULL
INTERLOCUTOR	CHAR(6)	
DIRECCION	VARCHAR(50)	
PROVINCIA	CHAR(2)	

TARJETAS		
CLIENTE	CHAR(6)	NOT NULL
NUM_TARJETA	CHAR(15)	NOT NULL
NOM_AUTORIZADO	CHAR(15)	NOT NULL
APE_AUTORIZADO	CHAR(20)	NOT NULL
FX_CADUCIDAD	DATE	
LIM_MES	SMALLINT	

VENDEDOR		
VENDEDOR	CHAR(6)	NOT NULL
NOMBRE	CHAR(15)	NOT NULL
APELLIDO	CHAR(20)	NOT NULL
FX_CONTRAT	DATE	
SALARIO	DECIMAL(8,2)	
COM_VENTAS	SMALLINT	
RESPONSABLE	CHAR(6)	

VALEDESC		
CLIENTE	CHAR(6)	NOT NULL
FX_VALE	DATE	NOT NULL
IMPORTE	DECIMAL(6,2)	

Capítulo 7 – Subconsultas

Objetivos

- ✓ **Diferenciar los tipos de subconsultas:**
 - Correlacionadas**
 - No correlacionadas**
 - ✓ **Operadores para usar en subconsultas**
 - SOME / ANY**
 - ALL**
 - EXISTS**
 - ✓ **Dónde escribir las subconsultas**
-

Subconsultas: Terminología

OUTER QUERY ó consulta externa



INNER QUERY ó consulta interna

```
SELECT . . . . .  
FROM TABLA_EXTERNA  
WHERE CAMPO1 = *  
      (SELECT CAMPO_B FROM TABLA_INTERNA ** )
```

(*) Es imprescindible que la comparación sea consistente

(**) TABLA_EXTERNA y TABLA_INTERNA pueden ser la misma tabla

Subconsultas: No correlacionadas ... 1/2

"Clientes con un importe de compras superior a la media"



```
SELECT NOMBRE, APELLIDO, IMPORTE_COMPRAS AS IMP
FROM CLIENTES
WHERE IMPORTE_COMPRAS >
(SELECT AVG (IMPORTE_COMPRAS) FROM CLIENTES)
```

"No existe dependencia entre la consulta externa y la interna"

Subconsultas: No correlacionadas ... 2/2

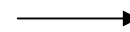
"Clientes con un importe de compras superior a la media"



```
SELECT NOMBRE, APELLIDO, IMPORTE_COMPRAS AS IMP
FROM CLIENTES
WHERE IMPORTE_COMPRAS >
(SELECT AVG (IMPORTE_COMPRAS) FROM CLIENTES)
```

Orden de ejecución:

1°	SELECT AVG (IMPORTE_COMPRAS) FROM CLIENTES
2°	SELECT CLIENTE, NOMBRE, APELLIDO, FX_ALTA FROM CLIENTES WHERE IMPORTE_COMPRAS > 1982,14



1982,14



NOMBRE	APELLIDO	IMP
LUISA	RATO	2121.00
JOSE	TOPON	4073.68
MARIA	PONTE	10512.00

Subconsultas: Correlacionadas ... 1/2

"Clientes con un importe de compras superior a la media dentro de su tipo"



```
SELECT NOMBRE, APELLIDO, IMPORTE_COMPRAS
FROM CLIENTES A
WHERE IMPORTE_COMPRAS >
      (SELECT AVG (IMPORTE_COMPRAS) FROM CLIENTES B
       WHERE B.TIPO = A.TIPO)
```

"La consulta interna depende del resultado obtenido en la consulta externa"

Subconsultas: Correlacionadas ... 2/2

1º

```
SELECT NOMBRE, APELLIDO, TIPO, IMPORTE_COMPRAS AS EUROS
FROM CLIENTES A
WHERE IMPORTE_COMPRAS >
      (SELECT AVG(IMPORTE_COMPRAS)
       FROM CLIENTES B
        WHERE B.TIPO = A.TIPO)
```

Tabla clientes

NOMBRE	APELLIDO	TIPO	EUROS
LUISA	ROCA	A1	94.12
MIGUEL	SOLA	A2	151.00
ANA	NOTARIO	A3	344.00
JUAN	RENATE	A1	32.10
LUIS	TEMPORA	A2	193.40
EVA	RAMIREZ	A4	724.30
ELENA	REZAGO	B1	1250.45
....

2º

Medias por tipo

A1	59.073333333333333333333333333333
A2	173.6900000000000000000000000000
A3	363.0570000000000000000000000000
A4	796.1733333333333333333333333333
B1	1250.4500000000000000000000000000

NOMBRE	APELLIDO	TIPO	EUROS
LUISA	ROCA	A1	94.12
LUIS	TEMPORA	A2	193.40
...

Comparación de valores múltiples. Cláusula IN.

"Clientes que han recibido vales descuento "



```
SELECT CLIENTE, NOMBRE, APELLIDO
FROM CLIENTES
WHERE CLIENTE IN
(SELECT DISTINCT CLIENTE FROM VALEDESC)
```

2º

CLIENTE	NOMBRE	APELLIDO
1100	TOMAS	GOTA
1210	JAIME	TAPADO
1320	SIMON	DOMINGUEZ
1430	DOLORES	GUERRA

1º

CLIENTE
1100
1210
1320
1430

Comparación de valores múltiples. Cláusula ALL.

"Clientes con un importe de compras superior a la media de compras de todos los tipos de cliente"



2º

```
SELECT CLIENTE, NOMBRE, APELLIDO,  
       IMPORTE_COMPRAS AS EUROS  
FROM CLIENTES  
WHERE IMPORTE_COMPRAS > ALL  
      (SELECT AVG (IMPORTE_COMPRAS)  
       FROM CLIENTES  
       GROUP BY TIPO)
```

CLIENTE	NOMBRE	APELLIDO	EUROS
1980	LEYRE	TEZ	11299,00
3410	MATILDE	ASENSI	21000,00

1º

Medias por tipo

A1	59.073
A2	173.69
A3	363.06
A4	796.17
B1	1250.45

Comparación de valores múltiples. Cláusula ANY/SOME.

"Clientes con un importe de compras superior a la media de compras de alguno de los tipos de cliente"



2º

```
SELECT CLIENTE, NOMBRE, APELLIDO,  
IMPORTE_COMPRAS AS EUROS  
FROM CLIENTES  
WHERE IMPORTE_COMPRAS > SOME  
  (SELECT AVG (IMPORTE_COMPRAS)  
   FROM CLIENTES  
   GROUP BY TIPO)
```

CLIENTE	NOMBRE	APELLIDO	EUROS
110	AMPARO	ROCA	94.12
220	MIGUEL	SOLA	151.00
330	ANA	NOTARIO	344.00
660	LUIS	TEMPORA	193.40
....

1º

Medias por tipo

A1	59.073
A2	173.69
A3	363.06
A4	796.17
B1	1250.45

Cláusula EXISTS

"Clientes que se llamen igual que algún vendedor"



```
SELECT CLIENTE, NOMBRE, APELLIDO  
FROM CLIENTES  
WHERE EXISTS  
(SELECT VENDEDOR FROM VENDEDOR  
WHERE NOMBRE = CLIENTES.NOMBRE AND  
APELLIDO = CLIENTES.APELLIDO)
```

La cláusula EXISTS se usa siempre en combinación con una subconsulta

Subconsultas: comparar grupos de valores

"Clientes que se llamen igual que algún vendedor"



```
SELECT CLIENTE, NOMBRE, APELLIDO,  
FROM CLIENTES  
WHERE (NOMBRE, APELLIDO1) IN  
(SELECT NOMBRE, APELLIDO  
FROM VENDEDOR)
```

"Podemos comparar duplas, ternas, etc. "

Subconsultas en la cláusula HAVING

Tipos de cliente con una media de compras superior a la media general :

```
SELECT TIPO, COUNT(*) AS NUM, AVG (COMPRAS_ANUAL)
FROM CLIENTES
GROUP BY TIPO
HAVING AVG (COMPRAS_ANUAL) > (SELECT AVG (COMPRAS_ANUAL) FROM
CLIENTES)
```

Subconsultas en la cláusula SELECT

Lista de clientes, compras, media de compras por tipo y media de compras general :

```
SELECT CLIENTE, COMPRAS_ANUAL,  
  (SELECT AVG (COMPRAS_ANUAL) AS MEDIA_TIPO FROM CLIENTES WHERE  
  TIPO = C.TIPO) ,  
  (SELECT AVG (COMPRAS_ANUAL) AS MEDIA_GENERAL FROM CLIENTES)  
FROM CLIENTES C  
WHERE SEXO = 'F'
```

Ejercicio 7

Capítulo 8 – Consultas sobre expresiones de tabla

Objetivos

- ✓ **Expresiones de tabla anidadas**
 - ✓ **Expresiones de tabla comunes**
 - ✓ **SQL recursivo**
-

Expresiones de tabla anidadas ...



```
SELECT X.CAMPO1, X.CAMPO2, ...  
FROM  
    ( SELECT .... FROM CLIENTES  
      WHERE COD_POSTAL LIKE '49%'  
      UNION  
      SELECT .... FROM VENDEDOR ) AS X  
  
WHERE X.CAMPO1 =  
GROUP BY X.CAMPO2
```

Expresiones de tabla anidadas ...



```
SELECT X.CAMPO1, Y.CAMPO2, ...  
FROM  
    ( SELECT CAMPO1, ... FROM CLIENTES ) AS X  
JOIN  
    ( SELECT CAMPO2, ... FROM CLIENTES ) AS Y  
ON X.CAMPO1 = Y.CAMPO2
```

Expresiones de tabla comunes

"Con la palabra clave WITH, defino una tabla que luego usaré en el FROM dentro de la misma sentencia"



```
WITH  
    X AS  
        ( SELECT CAMPO1, ... FROM CLIENTES ),  
    Y AS  
        ( SELECT CAMPO2, ... FROM CLIENTES )  
SELECT X.CAMPO1, Y.CAMPO2, ...  
    X JOIN Y  
ON X.CAMPO1 = Y.CAMPO2
```

Expresiones de tabla comunes: SQL recursivo



WITH

TABLA_NUEVA (A, B, C) AS

(SELECT ...

FROM TABLA_EXISTENTE

WHERE ...

} "SELECT DE INICIO"

UNION ALL

SELECT

FROM **TABLA_NUEVA** N, TABLA_EXISTENTE V

WHERE N.CAMPO1 = V.CAMPO2)

"SELECT ITERATIVA"

"SELECT PRINCIPAL"

SELECT A, B...

FROM TABLA_NUEVA

GROUP BY ...

"Es útil en aplicaciones de listas de materiales, sistemas de reservas y planificación de redes"

SQL recursivo: ejemplo 1/3

WITH

LISTA_CAJAS (CAJA, SUBCAJA, CANTIDAD) AS

(SELECT M.CAJA, M.SUBCAJA, M.CANTIDAD

FROM MATERIAL M

WHERE M.CAJA = 'A'

UNION ALL

SELECT H.CAJA, H.SUBCAJA, H.CANTIDAD

FROM LISTA_CAJAS P, MATERIAL H

WHERE P.SUBCAJA = H.CAJA)

SELECT CAJA, SUBCAJA, SUM(CANTIDAD)

FROM LISTA_CAJAS

GROUP BY CAJA, SUBCAJA



SQL recursivo: ejemplo 2/3

Tabla MATERIAL

CAJA	SUBCAJA	CANTIDAD
A	B	2
A	C	3
A	D	1
B	G	2
B	H	5
D	H	2
G	J	1
G	K	2
H	M	3
H	N	4

SELECT INICIAL

1ª ITERACIÓN

2ª ITERACIÓN

Tabla LISTA_CAJAS

CAJA	SUBCAJA	CANTIDAD
A	B	2
A	C	3
A	D	1

(1)

B	G	2
B	H	5
D	H	2

(2)

G	J	1
G	K	2
H	M	3
H	N	4
H	M	3
H	N	4

(3)

SQL recursivo: ejemplo 3/3

Tabla LISTA_CAJAS

CAJA	SUBCAJA	CANTIDAD
A	B	2
A	C	3
A	D	1

B	G	2
B	H	5
D	H	2

G	J	1
G	K	2
H	M	3
H	N	4
H	M	3
H	N	4

SELECT PRINCIPAL

Tabla LISTA_CAJAS

CAJA	SUBCAJA	CANTIDAD
A	B	2
A	C	3
A	D	1
B	G	2
B	H	5
D	H	2
G	J	1
G	K	2
H	M	6
H	N	8

Ejercicio 8

Capítulo 9 – Sentencias de actualización de datos

Objetivos

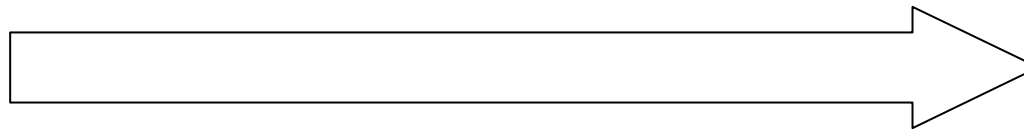
- ✓ **Insertar filas en las tablas**
 - ✓ **Borrar filas**
 - ✓ **Actualizar columnas de las tablas**
 - ✓ **Sentencia INSERT / SELECT**
 - ✓ **Sentencia SELECT / INSERT**
-

Añadir filas a las tablas: INSERT

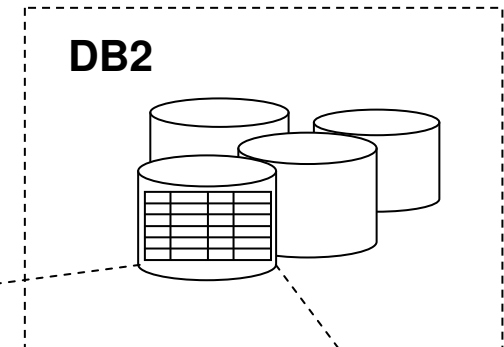
"Insertar un cliente nuevo "



```
INSERT INTO CLIENTES
(CLIENTE, NOMBRE, APELLIDO, TELEFONO, FX_ALTA)
VALUES ('2100', 'LUIS', 'EGIDO', '655444666', '03-05-2006')
```



```
INSERT INTO CLIENTES
VALUES ('2101', 'ANA', 'GARCIA', NULL,
CURRENT DATE, 'V213')
```



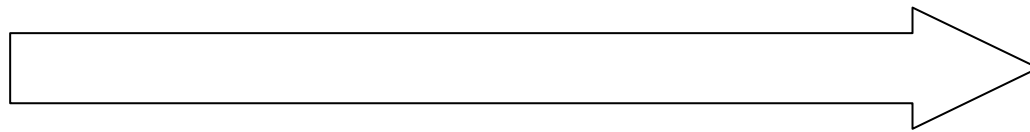
CLIENTE	NOMBRE	APELLIDO	TELEFONO	FX_ALTA	VENDEDOR	TIPO	COD_POSTAL
330	ANA	NOTARIO	959323232	05-01-2001	V112	A3	15123
770	EVA	RAMIREZ	666777888	03-01-2002	V212	A4	28010
990	ELENA	REZAGO	656565656	01-07-2002	V115	B1	08100
2100	LUIS	EGIDO	655444666	03-05-2006	-	-	-
2101	ANA	GARCIA	-	03-05-2006	V213	-	-

Actualizar columnas en las tablas: UPDATE

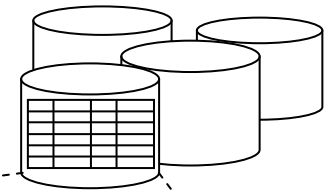
"Al cliente 2100, modificarle el código postal para que aparezca el 49137"



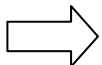
```
UPDATE CLIENTES  
SET COD_POSTAL = '49137'  
WHERE CLIENTE = '2100'
```



DB2



CLIENTE	NOMBRE	APELLIDO	TELEFONO	FX_ALTA	VENDEDOR	TIPO	COD_POSTAL
330	ANA	NOTARIO	959323232	05-01-2001	V112	A3	15123
770	EVA	RAMIREZ	666777888	03-01-2002	V212	A4	28010
990	ELENA	REZAGO	656565656	01-07-2002	V115	B1	08100
2100	LUIS	EGIDO	655444666	03-05-2006	-	-	49137
2101	ANA	GARCIA	-	03-05-2006	V213	-	-

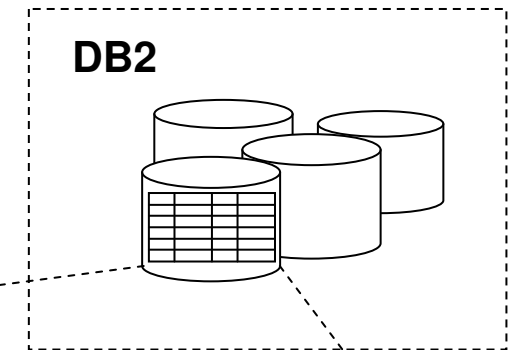
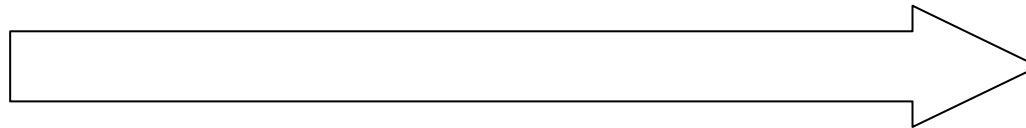


Actualizar columnas en las tablas: UPDATE ..

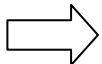
"Al cliente 2101, darle de alta con fecha siete días anterior a la que tiene ahora mismo"



```
UPDATE CLIENTES
SET FX_ALTA = FX_ALTA - 7 DAYS
WHERE CLIENTE = '2101'
```



CLIENTE	NOMBRE	APELLIDO	TELEFONO	FX_ALTA	VENDEDOR	TIPO	COD_POSTAL
330	ANA	NOTARIO	959323232	05-01-2001	V112	A3	15123
770	EVA	RAMIREZ	666777888	03-01-2002	V212	A4	28010
990	ELENA	REZAGO	656565656	01-07-2002	V115	B1	08100
2100	LUIS	EGIDO	655444666	03-05-2006	-	-	49137
2101	ANA	GARCIA	-	10-05-2006	V213	-	-



03-05-2006

Borrar filas de las tablas: DELETE

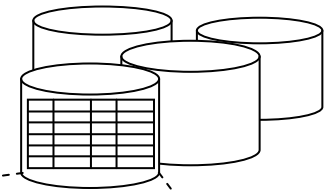
"Borrar los clientes 2100 y 2101"



```
DELETE FROM CLIENTES  
WHERE CLIENTE = '2100'
```

```
DELETE FROM CLIENTES  
WHERE CLIENTE = '2101'
```

DB2

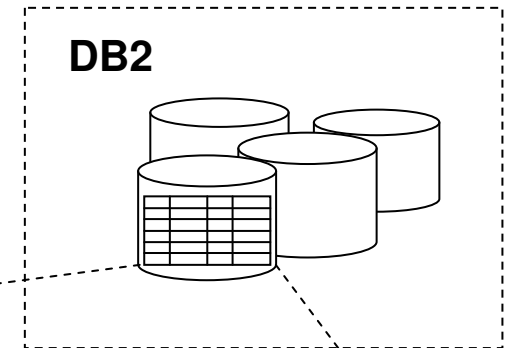
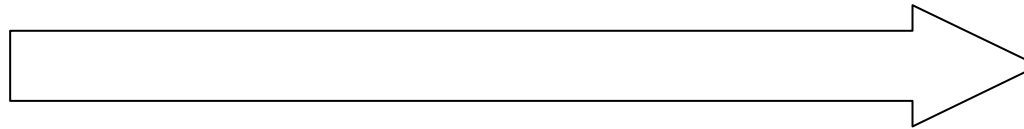


CLIENTE	NOMBRE	APELLIDO	TELEFONO	FX_ALTA	VENDEDOR	TIPO	COD_POSTAL
330	ANA	NOTARIO	959323232	05-01-2001	V112	A3	15123
770	EVA	RAMIREZ	666777888	03-01-2002	V212	A4	28010
990	ELENA	REZAGO	656565656	01-07-2002	V115	B1	08100
2100	LUIS	EGIDO	655444666	03-05-2006	-	-	-
2101	ANA	GARCIA	-	03-05-2006	V213	-	-

INSERT / SELECT

"Poblar la tabla de clientes del entorno de desarrollo con algunos clientes de producción "

```
INSERT INTO DESA.CLIENTES
SELECT CLIENTE, NOMBRE, APELLIDO, TELEFONO, FX_ALTA
FROM PROD.CLIENTES
WHERE CLIENTE BETWEEN 1 AND 500
```

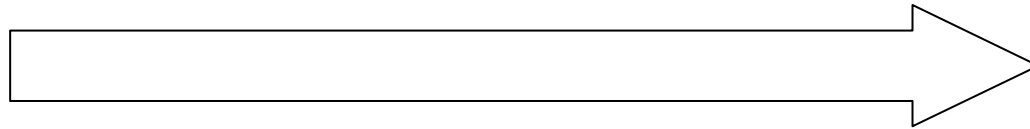


CLIENTE	NOMBRE	APELLIDO	TELEFONO	FX_ALTA	VENDEDOR	TIPO	COD_POSTAL
1
2	-
3	-
...	-

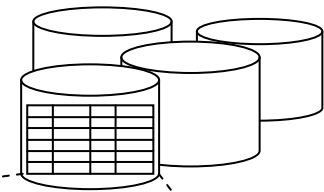
SELECT / INSERT

"Leer el número de cliente que estoy insertando y el tipo de cliente () "*

```
SELECT CLIENTE, TIPO FROM FINAL TABLE  
(INSERT INTO CLIENTES  
VALUES ('LUISA', 'GIL', '66444666', '04-05-2006'))
```



DB2



CLIENTE	NOMBRE	APELLIDO	TELEFONO	FX_ALTA	VENDEDOR	TIPO	COD_POSTAL
...
...	-
...	-
2102	LUISA	GIL	66444666	04-05-2006	-	A2	-



Columna de tipo identity

Columna informada por un before trigger



SELECT / INSERT ... Ejemplo



```
SELECT NU_VENDEDOR  
FROM FINAL TABLE (INSERT INTO VENDEDOR (NU_VENDEDOR)  
SELECT NU_VENDEDOR FROM VENDEDOR_PRU)
```

Ejercicio 9

Capítulo 10 – SQL para creación de objetos

Objetivos

- ✓ **Ver sentencias de creación/modificación de:**
 - ☐ **Tablas**
 - ☐ **Índices**
 - ☐ **Triggers**
 - ☐ **Vistas, alias, etc.**
 - ✓ **Definición de integridad referencial**
 - ✓ **Apuntes sobre los stored procedures**
-

Creación de una tabla (en un tablespace)

```
CREATE TABLESPACE TSCLIEN  
IN BD_PRUEBA;
```

***Base de
datos***



```
CREATE TABLE CLIENTES  
  
    ( CLIENTE      CHAR(6)      NOT NULL,  
      NOMBRE       CHAR(15)     NOT NULL,  
      APELLIDO1    CHAR(20)     NOT NULL,  
      TIPO         CHAR(3)      ,  
      TELEFONO     CHAR(9)      ,  
      VENDEDOR     CHAR(6)      ,  
      FX_NACIMIENTO DATE       )  
  
IN TSCLIEN;
```

Tablespace

Creación de índices en una tabla

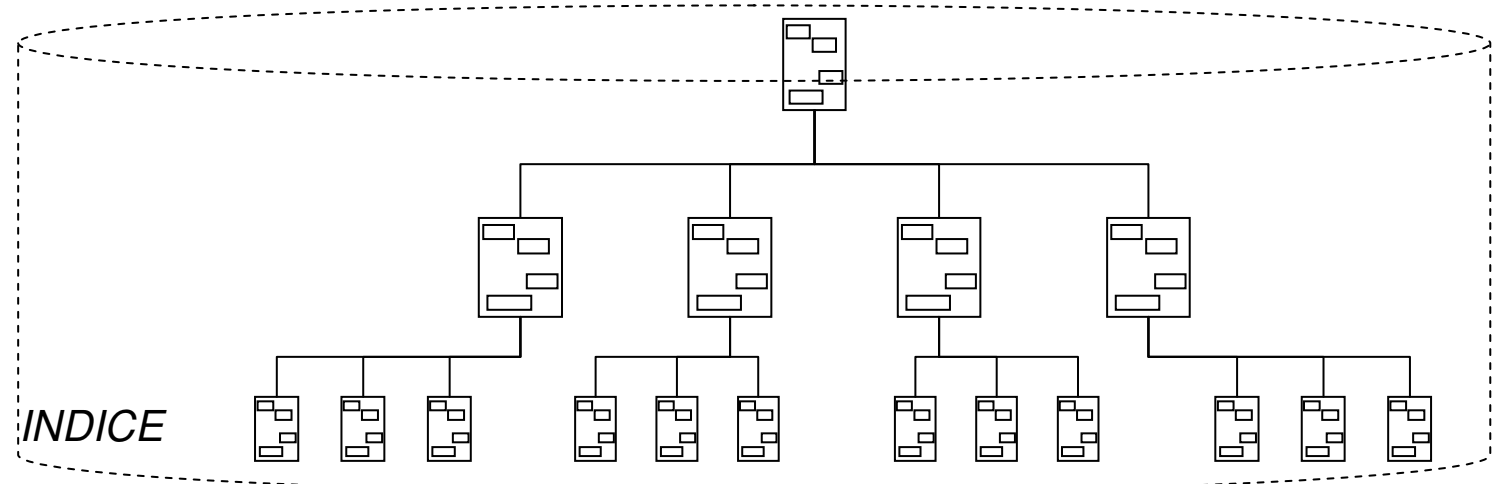
```
CREATE UNIQUE INDEX X1CLIE  
ON CLIENTES  
(CLIENTE ASC);
```

```
CREATE INDEX X2CLIE  
ON CLIENTES  
(NOMBRE, APELLIDO1);
```

Tabla

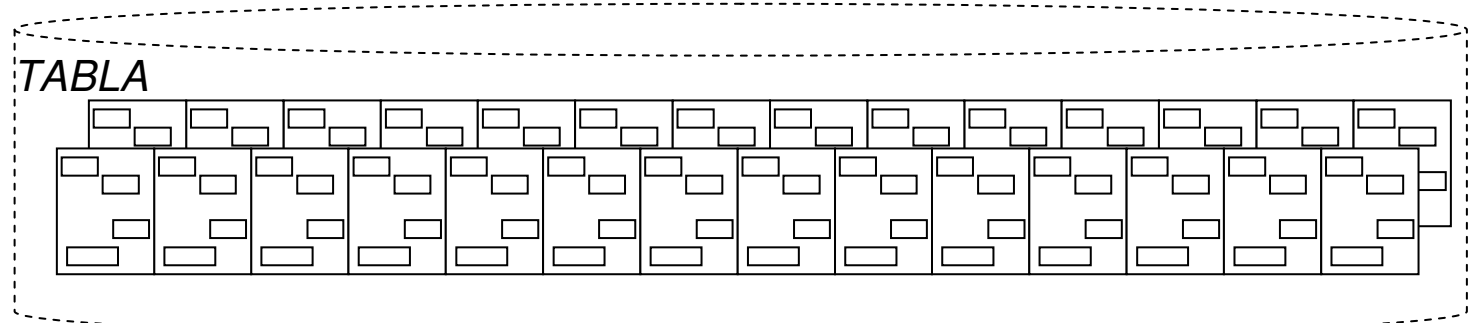
```
graph LR; T[Tabla] --> S1[CREATE UNIQUE INDEX X1CLIE ON CLIENTES (CLIENTE ASC);]; T --> S2[CREATE INDEX X2CLIE ON CLIENTES (NOMBRE, APELLIDO1);];
```


Estructura de un índice



"¿Para qué crear un índice en una tabla?"

"¿Quién los usa? "



Modificación de una tabla

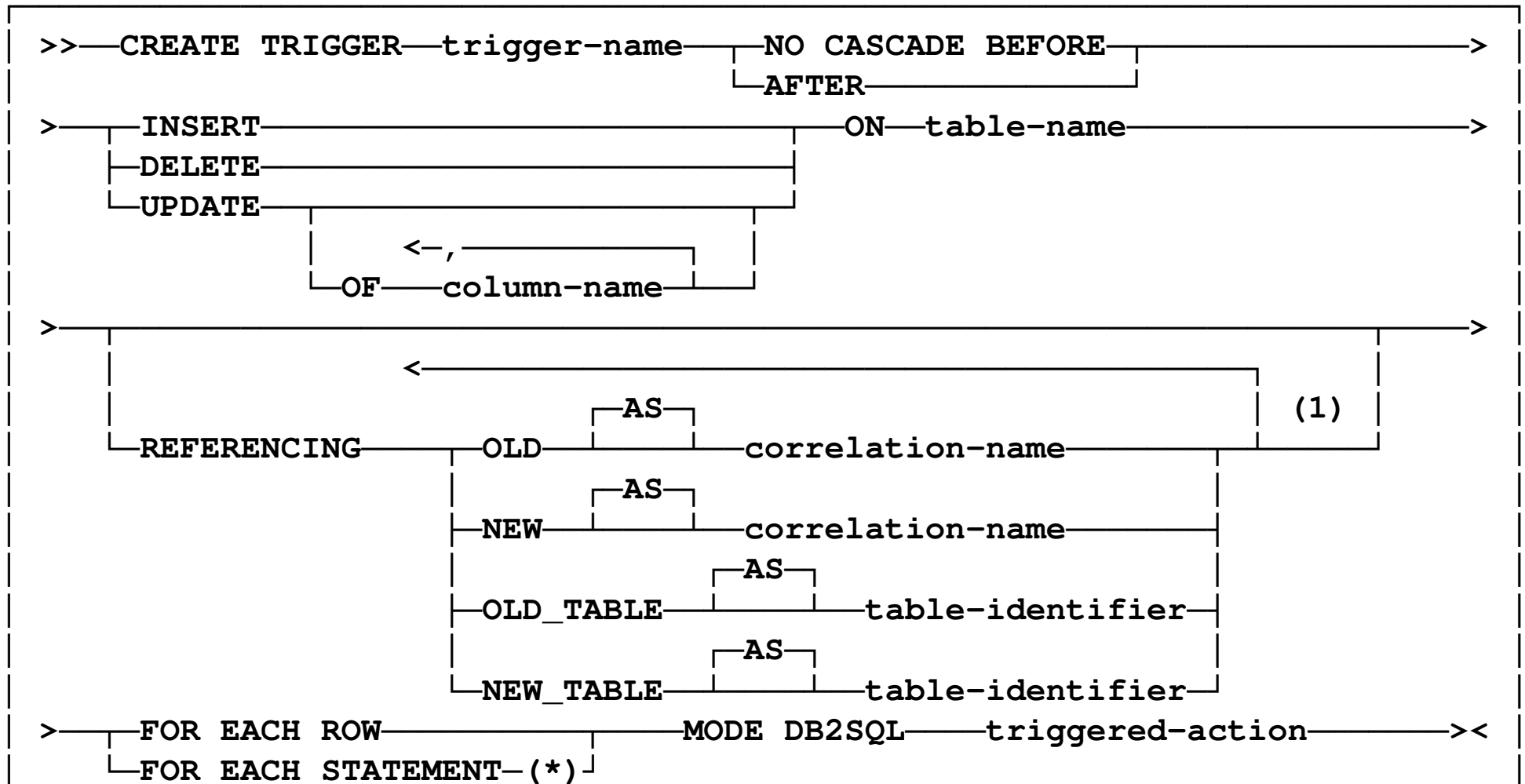
"A la tabla de CLIENTES creada antes, le añadimos la columna FX_ALTA"

```
ALTER TABLE CLIENTES  
      ADD FX_ALTA    DATE;
```

"Con la sentencia ALTER añadimos columnas y definiciones adicionales a la tabla"

```
ALTER TABLE CLIENTES  
      ADD CONSTRAINT control_fecha  
CHECK(FX_ALTA >= '01-05-2006');
```

Los triggers



* - No se usa en los BEFORE

```
CREATE TRIGGER GEST_PEDIDOS
AFTER UPDATE OF STOCK_ACTUAL, MAX_STOCK ON PRODUCTO
REFERENCING NEW AS NUEVO
FOR EACH ROW MODE DB2SQL
WHEN (NUEVO.STOCK_ACTUAL < 0.10 * NUEVO.MAX_STOCK)
BEGIN ATOMIC
    VALUES (LANZAR_PEDIDO (NUEVO.MAX_STOCK - NUEVO.STOCK_ACTUAL,
        NUEVO.CO_PRODUCTO) ) ;
END
```

La integridad referencial

- ✓ **Asegura la integridad de los datos**
 - ✓ **Se puede controlar por programa o la puede controlar el DB2**
 - ✓ **Para que la controle el DB2, se implementa por medio de claves:**
 - ☐ **Claves primarias (PK)**
 - ☐ **Claves ajenas (FK)**
 - ☐ **Claves únicas (UK)**
 - ✓ **Y con la regla de borrado: RESTRICT / CASCADE / SET NULL**
-

La clave primaria (PK)

- ✓ La clave primaria (PK) asegura la integridad de la entidad
- ✓ La PK debe ser:
 - ☐ Única
 - ☐ No nula
 - ☐ Pequeña

```
CREATE TABLE PEDIDOS
  (NU_PEDIDO    INTEGER NOT NULL,
   ... definición columnas ...
   FX_PEDIDO    DATE,
   PRIMARY KEY (NU_PEDIDO))
IN TSPEDIDOS;
```

PK
↓

TABLA DE PEDIDOS

NU_PEDIDO
101	
102	
103	
...	

"La clave primaria es un 'atributo' de la tabla"

La clave única (UK)

- ✓ Una clave única se define sobre una columna o columnas que deben ser referenciadas desde otra tabla
- ✓ Por definición la columna o columnas sobre las que se defina la clave única deben ser únicas y no nulas.

```
CREATE TABLE PEDIDOS
  (NU_PEDIDO    INTEGER NOT NULL,
   ...
   CO_REGALO    CHAR(6) NOT NULL,
   ...
   FX_PEDIDO    DATE,
   PRIMARY KEY (NU_PEDIDO),
   UNIQUE KEY (CO_REGALO))
IN TSPEDIDOS;
```

TABLA DE
PEDIDOS

PK

UK
↓

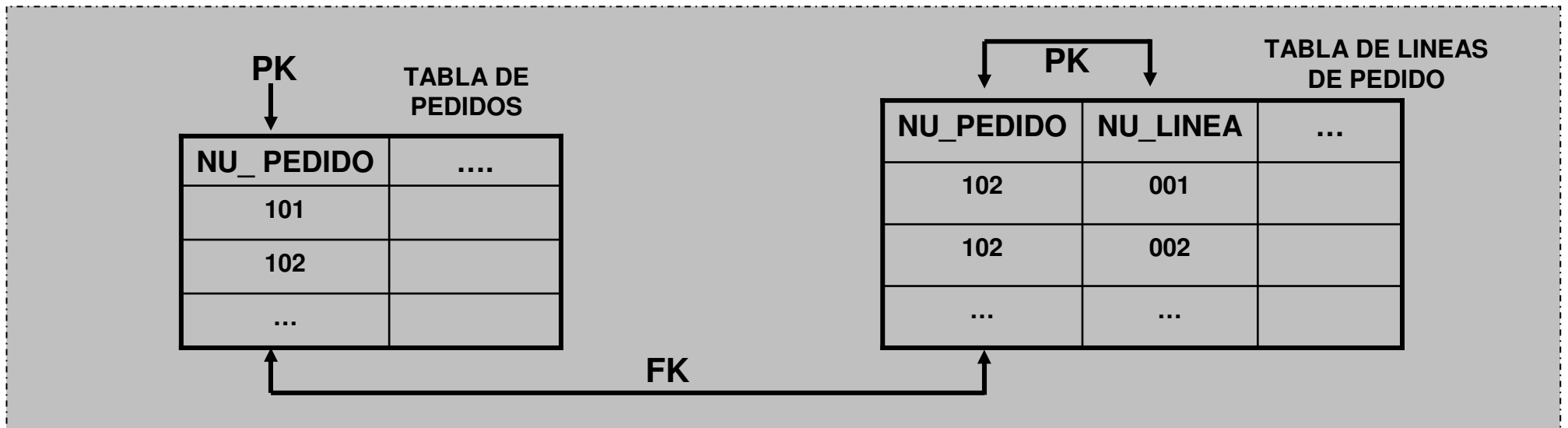
NU_PEDIDO	..	CO_REGALO	...
101		VX12DC	
102		VX12DD	
103		VX12DE	
...		...	

Las claves ajenas (FK) ...

- ✓ **Una clave ajena (FK) asegura la integridad entre dos entidades**
 - ✓ **Una tabla puede tener varias claves ajenas**
 - ✓ **Una clave ajena es un atributo de la tabla**
 - ✓ **Siempre hace referencia a una clave primaria o a una clave única de otra tabla**
 - ✓ **La FK puede ser:**
 - ☐ **Única / No única**
 - ☐ **Nula / No nula**
-

Las claves ajenas (FK)

```
CREATE TABLE LINEAS_PEDIDO
  (NU_PEDIDO INTEGER      NOT NULL,
   NU_LINEA  INTEGER      NOT NULL,
   . . . . .
   FOREIGN KEY (NU_PEDIDO)
     REFERENCES PEDIDOS (NU_PEDIDO) ON DELETE CASCADE)
IN TSLINEAS;
```



La integridad referencial

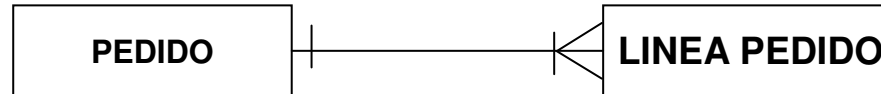


TABLA DE PEDIDOS

ENTIDAD PADRE

Nº PEDIDO
101	
102	
103	
...	

PK

ENTIDAD DEPENDIENTE

TABLA DE LINEAS DE PEDIDO

Nº PEDIDO	Nº LINEA	...
102	001	
102	002	
102	003	
...	...	

FK

Integridad referencial: control automático de las relaciones

¿Quién controla la integridad referencial?

IR DE LA B.D.



IR POR APLICATIVO

I.R. controlada por el aplicativo

➤ **Implementación potencial:**

Comprueba restricciones:

- ✓ Cada noche
- ✓ En cada COMMIT
- ✓ Después de cada sentencia SQL
- ✓ Inflight

➤ **Beneficios en el rendimiento:**

- ✓ Evita comprobaciones redundantes
- ✓ Evita comprobaciones repetitivas

➤ **Pero:**

- ✓ IR del gestor es casi tan rápido o más que IR aplicativo
 - ✓ Posibles errores en la codificación
-

I.R. controlada por el gestor de base de datos

➤ **Productividad en el desarrollo de aplicaciones:**

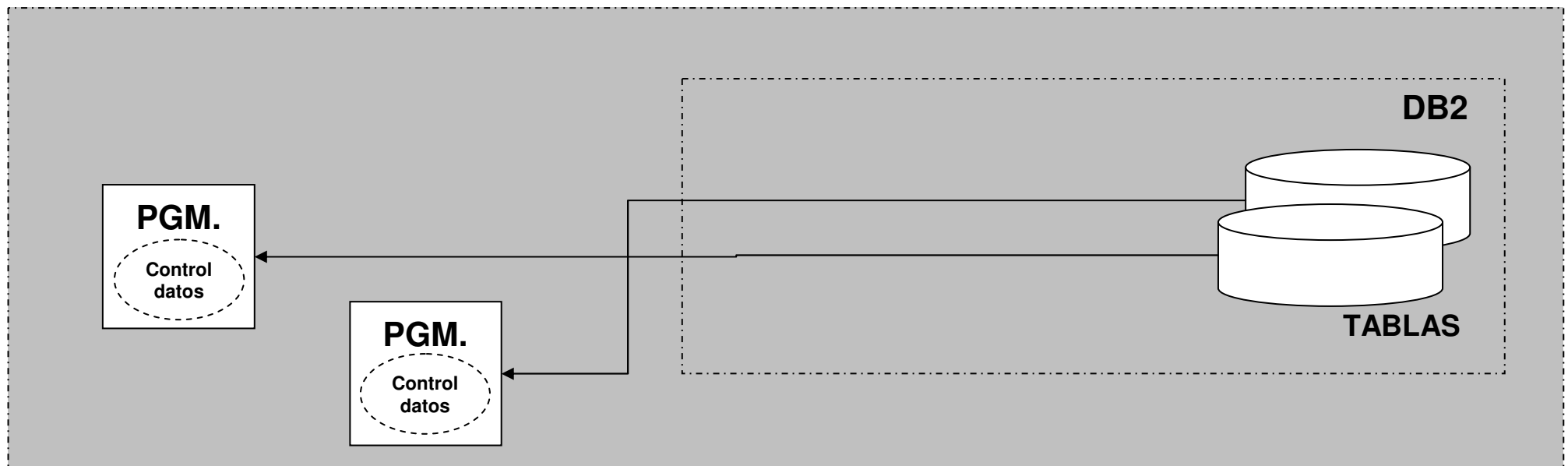
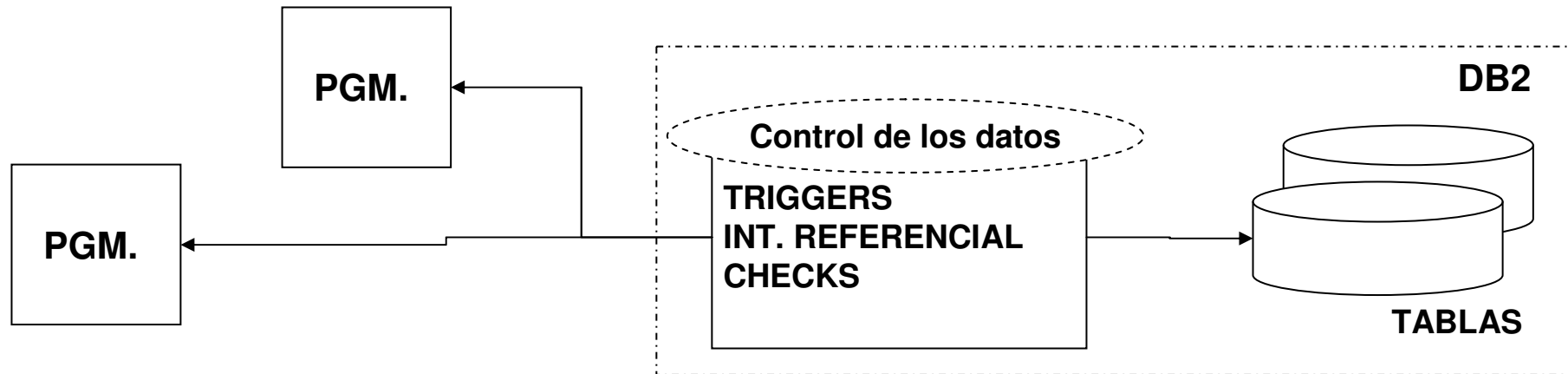
- ✓ **Menos SQL**
- ✓ **Menos coste de pruebas**

➤ **Integridad y consistencia:**

- ✓ **Reglas IR cumplidas en todos los entornos**
- ✓ **Se evitan posibles errores de codificación**

➤ **Soporte en procesos de Recuperación**

Los triggers, las check y la integridad referencial



Las vistas

```
CREATE VIEW V_CLI_A1  
AS SELECT CLIENTE, NOMBRE, APELLIDO, TELEFONO, VENDEDOR  
FROM CLIENTES  
WHERE TIPO = 'A1';
```

TABLA DE CLIENTES

CLIENTE	NOMBRE	APELLIDO	TIPO	TELEFONO	FX_ALTA	VENDEDOR	SEXO
110	AMPARO	ROCA	A1	649108410	01/01/1995	V110	F
220	MIGUEL	SOLA	A2	642911220	10/10/2003	V111	M
330	ANA	NOTARIO	A3	658490610	05/04/2005	V212	F
550	JUAN	RENATE	A1	683810205	17/08/2002	V342	M
660	LUIS	TEMPORA	A2	979291935	14/09/2003	V342	M
770	EVA	RAMIREZ	A4	696673695	30/09/2000	V111	F
990	ELENA	REZAGO	B1		15/08/2000	V145	F
1100	TOMAS	GOTA	A2	911999340	19/06/2000	V145	M
1210	JAIME	TAPADO	A4	943084050	16/05/2003	V231	M



```
SELECT * FROM V_CLI_A1;
```



CLIENTE	NOMBRE	APELLIDO	TELEFONO	VENDEDOR
110	AMPARO	ROCA	649108410	V110
550	JUAN	RENATE	683810205	V342

El uso de las vistas

TABLA DE CLIENTES

CLIENTE	NOMBRE	APELLIDO	TIPO	TELEFONO	FX_ALTA	VENDEDOR	SEXO
110	AMPARO	ROCA	A1	649108410	01/01/1995	V110	F
220	MIGUEL	SOLA	A2	642911220	10/10/2003	V111	M
330	ANA	NOTARIO	A3	658490610	05/04/2005	V212	F
550	JUAN	RENATE	A1	683810205	17/08/2002	V342	M
660	LUIS	TEMPORA	A2	979291935	14/09/2003	V342	M
770	EVA	RAMIREZ	A4	696673695	30/09/2000	V111	F
990	ELENA	REZAGO	B1		15/08/2000	V145	F
1100	TOMAS	GOTA	A2	911999340	19/06/2000	V145	M
1210	JAIME	TAPADO	A4	943084050	16/05/2003	V231	M

```
SELECT CLIENTE, NOMBRE, APELLIDO,  
TELEFONO, VENDEDOR  
FROM CLIENTES  
WHERE TIPO = 'A1'  
AND CLIENTE = '550';
```

```
SELECT * FROM V_CLI_A1  
WHERE CLIENTE = '550';
```



CLIENTE	NOMBRE	APELLIDO	TELEFONO	VENDEDOR
550	JUAN	RENATE	683810205	V342

Dos formas de recuperar la misma información

Ejemplos de vistas

```
CREATE VIEW VPROYECTO
(CODIGO, NOMBRE, DEPART, GERENTE, NOMBRE, APELLIDO, TLF)
AS SELECT P.CO_PROY, P.NOM_PROY, P.NU_DEPT,
        E.NU_EMPLE, E.NOMBRE, E.APELLIDO, E.TELEFONO
FROM PROYECTOS P RIGHT OUTER JOIN EMPLEADOS E
ON P.RESP_PROY = E.NU_EMPLE;
```

...es la misma vista que ...

```
CREATE VIEW VPROYECTO
AS SELECT P.CO_PROY AS CODIGO, P.NOM_PROY AS NOMBRE,
P.NU_DEPT AS DEPART, E.NU_EMPLE AS GERENTE, E.NOMBRE,
E.APELLIDO, E.TELEFONO AS TLF
FROM PROYECTOS P RIGHT OUTER JOIN EMPLEADOS E
ON P.GERENTE = E.NU_EMPLE;
```

Ejemplos de vistas ...

```
CREATE VIEW PRIMER_QTR (NUMERO, IMPORTE, FECHA) AS
  SELECT CO_VENTA, CARGO, FX_VENTA
  FROM MES1
  WHERE FX_VENTA BETWEEN '01/01/2006' and '01/31/2006'
  UNION All
  SELECT CO_VENTA, CARGO, FX_VENTA
  FROM MES2
  WHERE FX_VENTA BETWEEN '02/01/2006' and '02/28/2006'
  UNION All
  SELECT CO_VENTA, CARGO, FX_VENTA
  FROM MES3
  WHERE FX_VENTA BETWEEN '03/01/2006' and '03/31/2006';
```

Las autorizaciones

Los permisos se dan con la sentencia GRANT

```
GRANT UPDATE ON CLIENTES TO USER07;  
GRANT SELECT ON CLIENTES TO PUBLIC;
```

Y se quitan con la sentencia REVOKE

```
REVOKE UPDATE ON CLIENTES FROM USER07;  
REVOKE SELECT ON CLIENTES FROM PUBLIC;
```

Puede dar o quitar permisos:

- **una autoridad administrativa con suficientes privilegios**
 - **el propietario del objeto**
 - **alguien que tenga el permiso concedido WITH GRANT OPTION**
-

Alias

```
CREATE ALIAS STOCKB FOR DB2_BCN_1.PROD.STOCKS;
```

```
SELECT * FROM DB2_BCN_1.PROD.STOCKS  
WHERE CO_PROD = '250256';
```

```
SELECT * FROM STOCKB  
WHERE CO_PROD = '250256';
```

"Se crean sobre tablas o vistas sobre una base de datos que puede estar en la misma máquina o en otra máquina remota"

Sinónimos

```
CREATE SYNONYM PRODUCTOS FOR BDM005A10.XMLPRODXMAT
```

```
SELECT * FROM BDM005A10.XMLPRODXMAT  
WHERE CO_PROD = '250256';
```

```
SELECT * FROM PRODUCTOS  
WHERE CO_PROD = '250256';
```

"Se crean sobre tablas o vistas que estén en la misma máquina"

Ejercicio 10

Capítulo 11 – Manejo de cursores




Objetivos

- ✓ **Uso de los cursores**
 - ✓ **Tipos de cursores**
 - ✓ **Cursores en stored procedures**
-

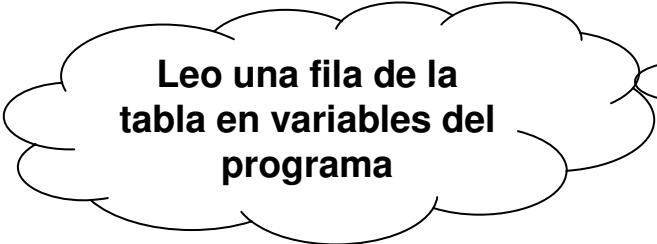
SELECT ... INTO

"En programas, cuando queremos recuperar una fila de una tabla ..."

```
SELECT VENDEDOR, SALARIO, FX_CONTRAT  
INTO :WS-VENDEDOR, :WS-SALARIO, :WS-FX-CONTRAT  
FROM VENDEDOR  
WHERE NU_VENDEDOR = 'V112'
```



VENDEDOR	SALARIO	FX_CONTRAT
V112	11500.00	2002-01-13



Leo una fila de la
tabla en variables del
programa

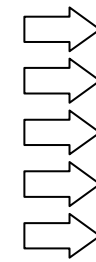


"... usamos una SELECT ... INTO"

Uso de los cursores ... 1/2

"... cuando queremos ejecutar una **SELECT** que recupera más de una fila ... "

```
SELECT VENDEDOR, SALARIO, FX_CONTRAT  
FROM VENDEDOR  
WHERE FX_CONTRAT > '01-01-2000'
```



VENDEDOR	SALARIO	FX_CONTRAT
V110	12500.00	2002-01-02
V111	11500.00	2002-01-13
V123	13500.00	2001-12-17
V134	14500.00	2001-10-04
....

**Se recuperan todas las
filas que cumplen las
condiciones WHERE
pero quiero tratarlas de
una en una**



"... debemos usar un cursor que nos permite recuperar las filas de una en una"

Uso de los cursores ... 2/2

"Un cursor es como una **SELECT** 'descompuesta' en cuatro partes "

1º

```
DECLARE CURSOR CUR_VEND FOR
SELECT VENDEDOR, SALARIO, FX_CONTRAT
FROM VENDEDOR
WHERE FX_CONTRAT > '01-01-2000'
```

2º

```
OPEN CURSOR CUR_VEND
```

3º

```
FETCH CURSOR CUR_VEND
INTO :WS-VENDEDOR, :WS-SALARIO,
:WS-FX-CONTRAT
```

4º

```
CLOSE CURSOR CUR_VEND
```



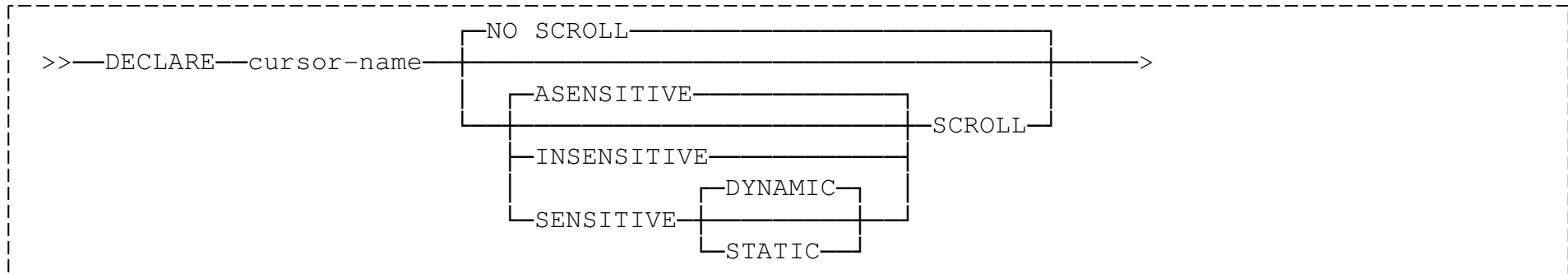
VENDEDOR	SALARIO	FX_CONTRAT
V110	12500.00	2002-01-02
V111	11500.00	2002-01-13
V123	13500.00	2001-12-17
V134	14500.00	2001-10-04
....

Tipos de cursores

- ✓ **FOR UPDATE OF**
 - ✓ **FOR READ ONLY**
 - ✓ **OPTIMIZE FOR n ROWS**
 - ✓ **FETCH FIRST n ROWS ONLY**
 - ✓ **NO SCROLL / SCROLL**
 - ✓ **WITH HOLD**
 - ✓ **WITH RETURN**
-

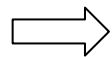
Cursores 'scrollables' ... el DECLARE

SINTAXIS:



EJEMPLO:

```
DECLARE CURSOR_CLI SENSITIVE STATIC SCROLL CURSOR FOR  
SELECT NU_CLIENTE, ....  
FROM CLIENTES  
WHERE COD_POSTAL = .. ;
```



OPEN CURSOR CUR_CLI

(en tiempo de OPEN se crea una declared temporary table con todas las filas cualificadas ordenadas por NU_CLIENTE)

FETCH NEXT/BEFORE/ABSOLUTE +n/ ...etc.

(el fetch se realiza sobre la tabla temporal)

Ejercicio:

Busca información en la web sobre la declaración y manejo de este tipo de cursores

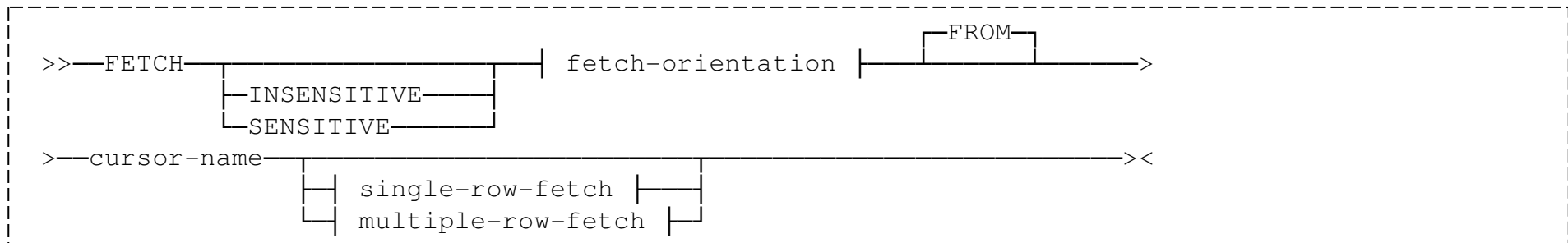
También se producen toques cuando hacemos fetch sensitive –aunque tengamos creada una tabla resultado–

Cursores 'scrollables'...

- ✓ **Cláusula FOR UPDATE OF incompatible con INSENSITIVE**
 - ✓ **Con SENSITIVE STATIC ya podemos poner ORDER BY y FOR UPDATE en un mismo cursor. Sin embargo, la ordenación no está completamente garantizada cuando ha habido modificaciones.**
 - ✓ **Opciones SENSITIVE DYNAMIC y los ASENSITIVE (V8)**
 - ✓ **STATIC se refiere al número de filas del result set. Este número permanece invariable a los inserts y deletes sobre la tabla**
 - ✓ **Con SENSITIVE, cada vez que accede a la tabla temporal comprueba en la tabla que el registro del cursor sigue siendo válido. De no ser válido:**
 - **SQLCODE: -222 AN UPDATE OR DELETE OPERATION WAS ATTEMPTED AGAINST A HOLE USING cursor-name**
 - **SQLCODE: +222 HOLE DETECTED USING cursor-name (al hacer un FETCH)**
-

Cursores 'scrollables' ... el tipo de FETCH

SINTAXIS:



El FETCH puede ser sensitive o insensitive dependiendo de la declaración que se haya hecho en el cursor.

Ejemplo:

Si hacemos un DECLARE INSENSITIVE y posteriormente pedimos un FETCH SENSITIVE se producirá un error

Cursores 'scrollables' ... el FETCH de una fila

***El FETCH para leer una fila.
Orientación del FETCH:***

SINTAXIS:

FETCH

BEFORE

AFTER

NEXT

PRIOR

FIRST

LAST

CURRENT

ABSOLUTE

host-variable

integer-constant

RELATIVE

host-variable

integer-constant

.....

<-

INTO

host-variable

INTO

DESCRIPTOR descriptor-name

Result sets

- ✓ **Desde un store procedure se pueden devolver cursores enteros al programa llamador**
 - ✓ **Cursores rowset positioning: podemos usar el cursor con posicionamiento por filas o por conjuntos de varias filas (rowset positioning)**
 - ✓ **Óptimo para recuperación de filas desde programas ejecutados desde servidores remotos**
-



Ejercicio 11



Capítulo 12 – Cuatro apuntes

Objetivos

- ✓ **Evitar escribir malas sentencias**
 - ✓ **Hacer un buen uso de los índices**
 - ✓ **Ahorrar esfuerzo de programación**
-

Para escribir una buena sentencia - I

- ✓ Dentro de la SELECT podemos poner cualquier expresión por complicada que sea

```
SELECT DAYOFWEEK(CAST('10/11/2005' AS FECHA)),  
       DAYOFWEEK(TIMESTAMP('10/12/2006', '01.02')),  
       DAYOFWEEK(CAST(CAST('10/11/2006' AS FECHA) AS CHAR(20))),  
       DAYOFWEEK(CAST(TIMESTAMP('10/12/1998', '01.02') AS CHAR(20)))  
FROM SYSIBM.SYSDUMMY1;
```

- ✓ ... pero dentro del WHERE los predicados deben ser sencillos

```
SELECT ...  
      INTO :WS-...  
      FROM VENDEDOR  
      WHERE DAYOFWEEK(FX_CONTRATO) = 2;
```

("Mejorado en V8")

Para escribir una buena sentencia - II

✓ ... porque los predicados complejos en el WHERE pueden provocar caminos de acceso con peor rendimiento:

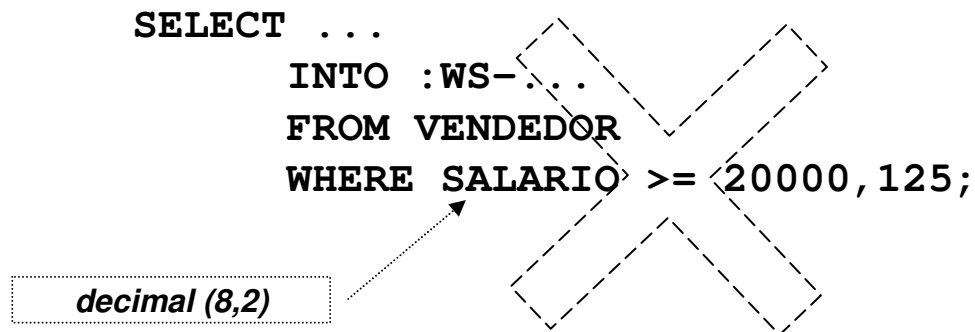
- ❑ Operaciones aritméticas

- ❑ Funciones escalares

- ❑ ...

✓ Es conveniente que los datos comparados en los predicados sean iguales y de la misma longitud:

```
SELECT ...  
  INTO :WS-...  
FROM VENDEDOR  
WHERE SALARIO >= 20000,125;
```



decimal (8,2)

Para escribir una buena sentencia - III

- ✓ **En el WHERE utilizar siempre las primeras columnas de los índices**
 - ✓ **Recuperar sólo las filas absolutamente necesarias.**
 - ✓ **Recuperar únicamente las columnas que se vayan a utilizar.**
 - ✓ **Al utilizar las cláusulas ORDER BY o GROUP BY comprobar que existan índices que respalden y eviten el SORT**
 - ✓ **Las negaciones y los operadores 'distinto' consumen más que los operadores de igualdad**
-

Para escribir una buena sentencia - IV

- ✓ **Los joins pueden ser más rápidos que varias SQLs.**
 - ✓ **Mejor una join que una subconsulta correlacionada.**
 - ✓ **Mejor una subconsulta no correlacionada que un join.**
 - ✓ **Las subconsultas correlacionadas son peligrosas incluso como meras consultas puntuales**
-

FIN
