Introducción al SQL para usuarios y programadores

SOLUCIONARIO

Enrique Rivero
Luis Martínez
Luis Reina
Juan Benavides
Juan Mª Olaizola

APÉNDICE F. SOLUCIONES A LOS EJERCICIOS PROPUESTOS

A continuación se incluyen las soluciones a los ejercicios propuestos al final de los capítulos de la primera parte. En la mayoría de ellos se incluye también el resultado cuando éste contiene un número pequeño de filas. Para resolver los ejercicios marcados con asterisco es necesario haber leido el capítulo dedicado al manejo de fechas.

Solucionario Capítulo 3

"TIPOS DE DATOS"

3.1). Decir el tipo de dato de las constantes siguientes.

+0000.00
-0000.00
1000.00
1000.
1000
01000
1.E2
01.E02
'01000'
'A"B"C'

Solución:

+0000.00	Decimal (6, 2)
-0000.00	Decimal (6, 2)
1000.00	Decimal (6, 2)
1000.	Decimal (4)
1000	Entero grande
01000	Entero grande
1.E2	Coma flotante
01.E02	Coma flotante
'01000'	Hilera de caracteres
'A"B"C'	Hilera de caracteres (valor = A"B"C)

Solucionario Capítulo 4

"CONSULTAS SENCILLAS"

4.1). Hallar por orden alfabético los nombres de los departamentos cuyo director lo es en funciones y no en propiedad.

Solución:

```
SELECT   NOMDE
FROM     TDEPTO
WHERE    TIDIR = 'F'
ORDER BY NOMDE
```

Resultado:

```
NOMDE
------------------------------------
ORGANIZACION
PERSONAL CONTRATADO
SECTOR INDUSTRIAL
```

4.2). Obtener un listín telefónico de los empleados del departamento 121 incluyendo nombre de empleado, número de empleado y extensión telefónica. Por orden alfabético.

Solución:

```
SELECT   NOMEM, NUMEM, EXTEL
FROM     TEMPLE
WHERE    NUMDE = 121
ORDER BY NOMEM
```

Resultado:

NOMEM	NUMEM	EXTEL
PEREZ, JULIO	150	340
PONS, CESAR	110	350
RUIZ, FABIOLA	370	360
VEIGA, JULIANA	190	350

4.3). Obtener por orden creciente una relación de todos los números de extensiones telefónicas de los empleados.

Solución:

```
SELECT   DISTINCT EXTEL
FROM     TEMPLE
ORDER BY EXTEL
```

Resultado:

```
EXTEL
---------
    200
    220
    250
    340
    350
    360
    410
    450
    480
    500
    508
```

550
610
620
650
660
740
750
760
780
800
810
840
850
880
910

4.4). Hallar la comisión, nombre y salario de los empleados con más de tres hijos, clasificados por comisión, y dentro de comisión por orden alfabético.

Solución:

```
SELECT   COMIS, NOMEM, SALAR
FROM     TEMPLE
WHERE    NUMHI > 3
ORDER BY COMIS, NOMEM
```

Resultado:

COMIS	NOMEM	SALAR
200	FLOR, DOROTEA	2900
-	LOPEZ, ANTONIO	7200
-	VEIGA, JULIANA	3000

4.5). Obtener salario y nombre de los empleados sin hijos por orden decreciente de salario y por orden alfabético dentro de salario.

Solución:

```
SELECT   SALAR, NOMEM
FROM     TEMPLE
WHERE    NUMHI = 0
ORDER BY SALAR DESC, NOMEM
```

Resultado:

SALAR	NOMEM
4500	ALBA, ADRIANA
4400	PEREZ, JULIO
4200	GARCIA, AUGUSTO
4000	FIERRO, CLAUDIA
3800	POLO, OTILIA
2800	DIEZ, AMELIA
2100	DURAN, LIVIA
2100	PEREZ, SABINA
2000	VAZQUEZ, HONORIA
1850	LARA, LUCRECIA
1800	MARTIN, MICAELA
1800	TORRES, HORACIO
1750	MUÑOZ, AZUCENA
1000	SANTOS, SANCHO

Solucionario Capítulo 5

"EXPRESIONES"

5.1). Decir los resultados de las sentencias siguientes, suponiendo que las ejecuta el usuario UABAD y que éste es el creador de las tablas TDEPTO Y TEMPLE.

1) SELECT USER FROM TDEPTO
2) SELECT USER FROM TEMPLE WHERE NUMHI > 4
3) SELECT DISTINCT USER FROM TEMPLE
4) SELECT DISTINCT USER FROM TDEPTO

Resultados:

1)
```
COL-1
-----------
UABAD
UABAD
UABAD
UABAD
UABAD
UABAD
UABAD
UABAD
UABAD
```

2)
```
COL-1
-----------
UABAD
UABAD
```

3)
```
COL-1
-----------
UABAD
```

4)
```
COL-1
-----------
UABAD
```

5.2). Obtener una relación por orden alfabético de los departamentos cuyo presupuesto es inferior a 50.000 euros. El nombre de los departamentos vendrá precedido de las palabras 'departamento de'.

Solución:

```
SELECT   'DEPARTAMENTO DE', NOMDE
FROM     TDEPTO
WHERE    PRESU < 50
ORDER BY NOMDE
```

Resultado:

```
COL-1                        NOMDE
---------------------------  ----------------------
DEPARTAMENTO DE   FINANZAS
DEPARTAMENTO DE   ORGANIZACION
DEPARTAMENTO DE   PERSONAL
```

Si se desea que las palabras 'Departamento de' aparezcan en la misma columna que el nombre del departamento, habría que usar una operación de concatenación:

```
SELECT   'DEPARTAMENTO DE ' || NOMDE
FROM     TDEPTO
WHERE    PRESU < 50
ORDER BY 1
```

El resultado sería:

```
                                  COL-1
-----------------------------------------------------------
DEPARTAMENTO DE FINANZAS
DEPARTAMENTO DE ORGANIZACION
DEPARTAMENTO DE PERSONAL
```

5.3). Llamemos presupuesto medio mensual de un departamento al resultado de dividir su presupuesto anual por 12. Supongamos que se decide aumentar los presupuestos medios mensuales de todos los departamentos en un 10 % a partir del mes de octubre inclusive. Para los departamentos cuyo presupuesto mensual medio anterior a octubre es de más de 5000 euros, hallar por orden alfabético el nombre de departamento y su presupuesto anual total después del incremento.

Solución:

```
SELECT   NOMDE, DEC((PRESU + 3 * (PRESU / 12) * 0.1),7,3)
FROM     TDEPTO
WHERE    PRESU / 12 > 5
ORDER BY NOMDE
```

Resultado

```
          NOMDE                    COL-2
---------------------------------  ---------------
DIRECCION COMERCIAL                  153,750
DIRECCION GENERAL                    123,000
PERSONAL CONTRATADO                  102,499
SECTOR INDUSTRIAL                    112,749
SECTOR SERVICIOS                      92,250
```

5.4). Suponiendo que en los próximos tres años el coste de vida va a aumentar un 6 % anual y que se suben los salarios en la misma proporción, hallar para los empleados con más de 4 hijos su nombre y su sueldo anual, actual y para cada uno de los próximos tres años, clasificados por orden alfabético.

Solución:

```
SELECT   NOMEM, SALAR * 12, SALAR * 12 * 1.06, SALAR * 12 * 1.06 * 1.06,
         SALAR * 12 * 1.06 * 1.06 * 1.06
FROM     TEMPLE
WHERE    NUMHI > 4
ORDER BY NOMEM
```

Resultado:

```
     NOMEM              COL-2          COL-3          COL-4          COL-5
-------------------------  -------------  ---------------  -------------  ---------------
FLOR, DOROTEA                34800          36888,00       39101,28       41447,35
LOPEZ, ANTONIO               86400          91584,00       97079,04      102903,78
```

5.5). Hallar por orden alfabético los nombres de los empleados tales que si se les da una gratificación de 1000 euros por hijo, el total de esta gratificacón no supera a la décima parte del salario.

Solución:

```
SELECT   NOMEM
FROM     TEMPLE
WHERE    NUMHI * 1000 <= SALAR/10
ORDER BY NOMEM
```

Resultado:

```
          NOMEM
----------------------------
ALBA, ADRIANA
DIEZ, AMELIA
DURAN, LIVIA
FIERRO, CLAUDIA
GARCIA, AUGUSTO
LARA, LUCRECIA
MARTIN, MICAELA
MUÑOZ, AZUCENA
PEREZ, JULIO
PEREZ, SABINA
POLO, OTILIA
SANTOS, SANCHO
TORRES, HORACIO
VAZQUEZ, HONORIA
```

5.6). Para los empleados del departamento 112 hallar el nombre y el salario total de cada uno (salario más comisión), por orden de salario total decreciente, y por orden alfabético dentro de salario total.

Solución:

```
SELECT   NOMEM, SALAR + COMIS
FROM     TEMPLE
WHERE    NUMDE = 112
ORDER BY 2 DESC, NOMEM
```

Resultado:

```
          NOMEM              COL-2
---------------------------  ----------
ALBA, ADRIANA                     -
GARCIA, OCTAVIO                4600
LASA, MARIO                    4600
TEROL, LUCIANO                 4000
DIEZ, AMELIA                   3700
PEREZ, SABINA                  3100
TORRES, HORACIO                2800
```

5.7). Hallar por orden de número de empleado el nombre y salario total (salario más comisión) de los empleados cuyo salario total supera a 3000 euros mensuales.

Solución:

```
SELECT   NUMEM, NOMEM, SALAR + COMIS
FROM     TEMPLE
WHERE    SALAR + COMIS > 3000
ORDER BY NUMEM
```

Resultado:

```
NUMEM          NOMEM              COL-3
-----------  ---------------------------  ----------
     120     LASA, MARIO              4600
     130     TEROL, LUCIANO           4000
     160     AGUIRRE, AUREO           4200
     180     PEREZ, MARCOS            5300
     240     SANZ, LAVINIA            3800
     270     GARCIA, OCTAVIO          4600
     280     FLOR, DOROTEA            3100
     330     DIEZ, AMELIA             3700
     360     LARA, DORINDA            3500
     440     DURAN, LIVIA             3100
```

450	PEREZ, SABINA	3100
480	PINO, DIANA	3100

5.8). Obtener los números de los departamentos en los que haya algún empleado cuya comisión supere al 20 % de su salario.

Solución:

```
SELECT   DISTINCT NUMDE
FROM     TEMPLE
WHERE    COMIS > 0.2 * SALAR
ORDER BY NUMDE
```

Resultado:

```
NUMDE
-----------
      111
      112
```

Solucionario Capítulo 6

"PREDICADOS"

6.1). Decir el valor de los predicados siguientes (Verdadero, Falso o Desconocido):

```
-1000 < 0
-1000 < -999.99
1000.00 = 1000
'1000.00' = '1000'
1000.00E00 = 1000
1E-1 = 0.1
```

Solución:

-1000 < 0	Verd.
-1000 < -999.99	Verd.
1000.00 = 1000	Verd.
'1000.00' = '1000'	Falso
1000.00E00 = 1000	Verd.
1E-1 = 0.1	Verd.

6.2). Hallar por orden de número de empleado el nombre y salario total (salario más comisión) de los empleados cuyo salario total supera al salario mínimo en 3000 euros mensuales.

Solución:

```
SELECT   NUMEM, NOMEM, SALAR + COMIS

FROM     TEMPLE
WHERE    SALAR + COMIS > SOME (SELECT SALAR + 3000
                               FROM   TEMPLE)
ORDER BY NUMEM
```

Resultado:

NUMEM	NOMEM	COL-3
120	LASA, MARIO	4600
160	AGUIRRE, AUREO	4200
180	PEREZ, MARCOS	5300
270	GARCIA, OCTAVIO	4600

6.3). Para los empleados que no tienen comisión obtener por orden alfabético el nombre y el cociente entre su salario y el número de hijos.

Solución:

```
SELECT   NOMEM, SALAR / NUMHI

FROM     TEMPLE

WHERE    COMIS IS NULL AND

                NUMHI <> 0

ORDER BY NOMEM
```

Resultado:

NOMEM	COL-2
CAMPOS, ROMULO	2000,00
CAMPS, AURELIO	4500,00
FLOR, DOROTEA	580,00
GALVEZ, PILAR	1900,00
GIL, GLORIA	900,00
LOPEZ, ANTONIO	1200,00
MORA, VALERIANA	2100,00
MORAN, CARMEN	2150,00
PONS, CESAR	1033,33
RUIZ, FABIOLA	1900,00
SANZ, CORNELIO	2025,00
VEIGA, JULIANA	750,00

6.4). Se desea hacer un regalo de un 1% del salario a los empleados en el día de su onomástica. Hallar por orden alfabético los nombres y cuantía de los regalos en euros para los que celebren su santo el día de San Honorio.

Solución:

```
SELECT NOMEM, SALAR * 0.01
FROM TEMPLE
WHERE NOMEM LIKE '%,%HONORIO%' OR
      NOMEM LIKE '%,%HONORIA%'
ORDER BY  NOMEM
```

Resultado:

NOMEM	COL-2
VAZQUEZ, HONORIA	20,00

6.5). Obtener por orden alfabético los nombres y salarios de los empleados del departamento 111 que tienen comisión si hay alguno de ellos cuya comisión supere al 15 % de su salario.

Solución:

```
SELECT   NOMEM, SALAR
FROM     TEMPLE
WHERE    NUMDE = 111 AND
                 COMIS IS NOT NULL AND
                 EXISTS (SELECT *
                                  FROM   TEMPLE
                                  WHERE  NUMDE = 111 AND
                                                COMIS IS NOT NULL AND
                                                COMIS > 0.15 * SALAR)
ORDER BY NOMEM
```

Resultado:

NOMEM	SALAR
AGUIRRE, AUREO	3100
DURAN, LIVIA	2100
LARA, DORINDA	2500
PINO, DIANA	2100
SANTOS, SANCHO	1000
SANZ, LAVINIA	2800
VAZQUEZ, HONORIA	2000

6.6). En la fiesta de Reyes se desea organizar un espectáculo para los hijos de los empleados, que se representará en dos días diferentes. El primer día asistirán los empleados cuyo apellido empiece por las letras desde A hasta L, ambas inclusive. El segundo día se cursarán invitaciones para el resto. A cada empleado se le asignarán tantas invitaciones gratuitas como hijos tenga y dos más. Además en la fiesta se entregará a cada empleado un obsequio por hijo. Obtener una lista por orden alfabético de los nombres a quienes hay que invitar el primer día de la representación, incluyendo también cuántas invitaciones corresponden a cada nombre y cuántos regalos hay que preparar para él.

(Obsérvese que si dos empleados están casados, esta consulta calculará dos veces el número de invitaciones familiar si los hijos figuran en la tabla tanto en la fila del marido como de la esposa).

Solución:

```
SELECT   NOMEM, NUMHI + 2, NUMHI
FROM     TEMPLE
WHERE    NOMEM BETWEEN 'A' AND 'LZ'
ORDER BY NOMEM
```

Resultado:

NOMEM	COL-2	COL-3
AGUIRRE, AUREO	4	2
ALBA, ADRIANA	2	0
CAMPOS, ROMULO	3	1
CAMPS AURELIO	3	1
DIEZ, AMELIA	2	0
DURAN, LIVIA	2	0
FIERRO, CLAUDIA	2	0
FLOR, DOROTEA	7	5
GALVEZ, PILAR	4	2
GARCIA, AUGUSTO	2	0
GARCIA, OCTAVIO	5	3
GIL, GLORIA	5	3
LARA, DORINDA	4	2
LARA, LUCRECIA	2	0
LASA, MARIO	3	1
LOPEZ, ANTONIO	8	6

6.7). Hallar por orden alfabético los nombres y salarios de empleados de los departamentos 110 y 111 que o bien no tengan hijos o bien su salario por hijo supere a 1000 euros, si hay alguno sin comisión en los departamentos 111 ó 112.

Solución:

```
SELECT   NOMEM, SALAR
FROM     TEMPLE
WHERE    (NUMDE = 110 OR NUMDE = 111) AND
         (NUMHI = 0   OR SALAR > 1000 * NUMHI) AND
```

```
                    EXISTS (SELECT *
                                FROM   TEMPLE
                                WHERE  (NUMDE = 111 OR NUMDE = 112)
                                       AND COMIS IS NULL)
            ORDER BY NOMEM
```

Resultado:

NOMEM	SALAR
AGUIRRE, AUREO	3100
CAMPOS, ROMULO	2000
DURAN, LIVIA	2100
LARA, DORINDA	2500
LARA, LUCRECIA	1850
MORAN, CARMEN	2150
PEREZ, MARCOS	4800
PINO, DIANA	2100
SANTOS, SANCHO	1000
VAZQUEZ, HONORIA	2000

6.8). Hallar por orden alfabético los nombres de departamentos que o bien tienen directores en funciones o bien en propiedad y su presupuesto anual excede a 50 000 euros o bien no dependen de ningún otro.

Solución:

```
SELECT   NOMDE
FROM     TDEPTO
WHERE    TIDIR = 'F' OR
         (TIDIR = 'P' AND PRESU > 50) OR
          DEPDE IS NULL
ORDER BY NOMDE
```

Resultado:

```
NOMDE
-----------------------------------
DIRECCION COMERCIAL
DIRECCION GENERAL
ORGANIZACION
PERSONAL CONTRATADO
PROCESO DE DATOS
SECTOR INDUSTRIAL
SECTOR SERVICIOS
```

Solucionario Capítulo 7

"FUNCIONES ESCALARES"

7.1). Hallar los nombres de los empleados que no tienen comisión, clasificados de manera que aparezcan primero aquellos cuyos nombres son más cortos.

Solución:

```
SELECT   LENGTH (NOMEM), NOMEM
FROM     TEMPLE
WHERE    COMIS IS NULL
ORDER BY 1, 2
```

Resultado:

COL-1	NOMEM
11	GIL, GLORIA
11	PONS, CESAR
12	PEREZ, JULIO
12	POLO, OTILIA
13	ALBA, ADRIANA
13	FLOR, DOROTEA
13	GALVEZ, PILAR
13	MORAN, CARMEN
13	RUIZ, FABIOLA
14	CAMPOS, ROMULO
14	CAMPS, AURELIO
14	LARA, LUCRECIA
14	LOPEZ, ANTONIO
14	MUÑOZ, AZUCENA
14	SANZ, CORNELIO
14	VEIGA, JULIANA
15	FIERRO, CLAUDIA
15	GARCIA, AUGUSTO
15	MARTIN, MICAELA
15	MORA, VALERIANA

7.2). Hallar por orden alfabético los nombres de empleados suprimiendo las tres últimas letras de los nombres de pila, para los empleados cuyos nombres de pila tengan más de 6 letras.

Solución:

```
SELECT   SUBSTR (NOMEM, 1, LENGTH (NOMEM) - 3)
FROM     TEMPLE
WHERE    NOMEM LIKE '%, _______%'
ORDER BY 1
```

Resultado:

```
   COL-1
------------------
ALBA, ADRI
CAMPS, AURE
FIERRO, CLAU
FLOR, DORO
GARCIA, AUGU
GARCIA, OCTA
LARA, DORI
```

LARA, LUCRE
LOPEZ, ANTO
MARTIN, MICA
MORA, VALERI
MUÑOZ, AZUC
RUIZ, FABI
SANZ, CORNE
SANZ, LAVI
TEROL, LUCI
TORRES, HORA
VAZQUEZ, HONO
VEIGA, JULI

7.3). Obtener el nombre del empleado y el valor del código ASCII del segundo carácter de aquél cuyo número de empleado es el 120.

Solución:

```
SELECT NOMEM, ASCII ( SUBSTR (NOMEM, 2) )
FROM    TEMPLE
WHERE NUMEM = 120
```

Resultado:

NOMEM	COL-2
LASA, MARIO	65

7.4). ¿En qué día del año (número de orden) y en qué día de semana entró en la empresa el empleado que se llama Aureo?.

Solución:

```
SELECT DAYOFYEAR (FECIN), DAYNAME (FECIN)
FROM    TEMPLE
WHERE NOMEM LIKE '%AUREO'
```

Resultado:

COL-1	COL-2
316	lunes

7.5). Obtener la primera posición de la letra A dentro de los nombres de los empleados del departamento 100.

Solución:

```
SELECT NOMEM, LOCATE ('A', NOMEM)
FROM TEMPLE
WHERE NUMDE = 100
```

Resultado:

NOMEM	COL-2
GALVEZ, PINAR	2
ALBA, ADRIANA	1
LOPEZ, ANTONIO	8

7.6). Codificar una sentencia SQL que devuelva aproximadamente la décima parte de las filas de la tabla de Empleados (nombre y departamento).

Solución:

```
SELECT NUMEM,NUMDE
FROM TEMPLE
WHERE RAND() < 0.1
ORDER BY NUMEM
```

7.7). Obtener la lista de los empleados (número de empleado y comisiones que cobran) con salario mayor de 4.000 €; especificar con valor 0 si en alguna fila la comisión está a nulos.

Solución:

```
SELECT NUMEM, COALESCE (COMIS, 0)
FROM    TEMPLE
WHERE SALAR > 4000
```

Resultado:

NUMEM	COL-2
150	0
180	500
250	0
260	0
310	0
320	0
350	0

Solucionario Capítulo 8

"UTILIZACIÓN DE FECHAS Y HORAS"

***8.1).** Obtener los nombres y sueldos de los empleados que hayan empezado a trabajar en la empresa el año 88 o después, por orden alfabético.

Solución:

```
SELECT   NOMEM, SALAR
FROM     TEMPLE
WHERE    FECIN > '31.12.1987'
ORDER BY 1
```

Resultado:

NOMEM	SALAR
FIERRO, CLAUDIA	4000
MARTIN, MICAELA	1800
MORA, VALERIANA	2100
MUÑOZ, AZUCENA	1750
SANTOS, SANCHO	1000
TORRES, HORACIO	1800

***8.2).** Obtener por orden alfabético los nombres de los empleados que empezaron a trabajar en la empresa en el año 1966.

Solución:

```
SELECT   NOMEM
FROM     TEMPLE
WHERE    FECIN BETWEEN '1.1.1966' AND '31.12.1966'
ORDER BY NOMEM
```

Resultado:

NOMEM
GARCIA, OCTAVIO
SANZ, LAVINIA

***8.3).** Obtener por orden alfabético los nombres de los empleados que han ingresado el 1.1.88 o en el día de hoy.

Solución:

```
SELECT   NOMEM
FROM     TEMPLE
WHERE    FECIN IN ('1.1.1988', CURRENT DATE)
ORDER BY NOMEM
```

Si esta sentencia se hubiera ejecutado el 21.1.88, su resultado habría sido:

NOMEM
MARTIN, MICAELA
SANTOS, SANCHO
TORRES, HORACIO

***8.4).** Obtener por orden alfabético los nombres y salarios de los empleados que o bien ingresaron después del 1.1.88 o bien antes y además tienen un salario inferior al salario más bajo de los que ingresaron con posterioridad al 1.1.88 incrementado en un 100 %.

Solución:

```
SELECT   NOMEM, SALAR
FROM     TEMPLE
WHERE    FECIN > '1.1.1988' OR
         (FECIN <= '1.1.1988' AND SALAR < ALL (SELECT SALAR * 2
                                               FROM   TEMPLE
                                               WHERE  FECIN > '1.1.1988'))
ORDER BY NOMEM
```

Resultado:

NOMEM	SALAR
FIERRO, CLAUDIA	4000
LARA, LUCRECIA	1850
MARTIN, MICAELA	1800
MORA, VALERIANA	2100
MUÑOZ, AZUCENA	1750
RUIZ, FABIOLA	1900
SANTOS, SANCHO	1000
TORRES, HORACIO	1800

***8.5).** Supongamos que según el convenio laboral de la empresa, para los empleados con más de un año de servicio el número de días de vacaciones anuales expresado en días laborables es de 20 incrementados en uno más por cada tres años de servicio cumplidos en el año anterior. Para los empleados que este año cumplen 45 o más años de edad y tienen más de un año de servicio, hallar por orden alfabético el nombre y el número de días laborables de vacaciones anuales que corresponde a cada uno.

Solución:

```
SELECT   NOMEM,  20 + (YEAR (CURRENT DATE) - YEAR (FECIN) - 1) / 3
FROM     TEMPLE
WHERE    YEAR (CURRENT DATE) - YEAR (FECNA) >= 45 AND
         YEAR (CURRENT DATE) - YEAR (FECIN) >  1
ORDER    BY NOMEM
```

Resultado (suponiendo que CURRENT DATE es el 5.7.90):

NOMEM	COL-2
AGUIRRE, AUREO	27
GALVEZ, PILAR	30
GARCIA, OCTAVIO	27
LASA, MARIO	27
LOPEZ, ANTONIO	27
PEREZ, JULIO	33
PEREZ, MARCOS	31
PONS, CESAR	33
SANZ, LAVINIA	27
TEROL, LUCIANO	26
VEIGA, JULIANA	29

***8.6).** Se desea analizar un plan de jubilación anticipada para los empleados con 60 años cumplidos en el que se ofrece una paga adicional extra de jubilación equivalente al salario actual de un mes por cada año de servicio cumplido. Hallar una lista por orden alfabético de los empleados que este año cumplen 60 ó más años indicando para cada uno la cuantía de esta paga extra.

Solución:

```
SELECT   NOMEM, SALAR * (YEAR (CURRENT DATE) - YEAR (FECIN) )
FROM     TEMPLE
WHERE    YEAR (CURRENT DATE) - YEAR (FECNA) >= 60
ORDER    BY NOMEM
```

Resultado (suponiendo que CURRENT DATE es el 5.7.90):

NOMEM	COL-2
PEREZ, JULIO	184800
PONS, CESAR	124000

***8.7).** Para los empleados de los departamentos 111 y 112 hallar por orden alfabético: nombre, edad en años cumplidos en la fecha del día de hoy y edad que tenían cuando ingresaron en la empresa.

Solución:

```
SELECT   NOMEM, YEAR (CURRENT DATE - FECNA),
         YEAR (FECIN - FECNA)
FROM     TEMPLE
WHERE    NUMDE = 111 OR NUMDE = 112
ORDER    BY NOMEM
```

Resultado (suponiendo que CURRENT DATE es el 5.7.90):

NOMEM	COL-2	COL-3
AGUIRRE, AUREO	50	29
DIEZ, AMELIA	41	23
DURAN, LIVIA	23	19
GARCIA, OCTAVIO	45	21
LARA, DORINDA	31	9
LARA, LUCRECIA	20	18
LASA, MARIO	55	33
MARTIN, MICAELA	22	19
PEREZ, SABINA	23	19
PINO, DIANA	25	20
SANTOS, SANCHO	20	18
SANZ, LAVINIA	48	23
TEROL, LUCIANO	44	23
TORRES, HORACIO	26	23
VAZQUEZ, HONORIA	24	21

***8.8).** Para los empleados de los departamentos 110 y 111 hallar por orden alfabético: nombre y tiempo que llevan en la empresa en el día de hoy expresado en años, meses y días.

Solución:

```
SELECT   NOMEM, YEAR  (CURRENT DATE - FECIN),
         MONTH (CURRENT DATE - FECIN),
         DAY   (CURRENT DATE - FECIN)
FROM     TEMPLE
WHERE    NUMDE = 110 OR NUMDE = 111
ORDER BY NOMEM
```

Resultado (suponiendo que CURRENT DATE es el 5.7.90):

NOMEM	COL-2	COL-3	COL-4
AGUIRRE, AUREO	21	7	24
CAMPOS, ROMULO	3	8	4
DURAN, LIVIA	4	4	5
LARA, DORINDA	21	8	26
LARA, LUCRECIA	2	8	4
MORAN, CARMEN	3	8	28
PEREZ, MARCOS	34	3	18
PINO, DIANA	4	4	5
SANTOS, SANCHO	2	5	15
SANZ, LAVINIA	24	4	9
VAZQUEZ, HONORIA	3	6	4

***8.9).** Hallar para los empleados de los departamentos 110 y 112 su nombre y su mes y día de cumpleaños, por orden creciente de éstos.

Solución:

```
SELECT   NOMEM, MONTH (FECNA), DAY (FECNA)
FROM     TEMPLE
WHERE    NUMDE = 110 OR NUMDE = 112
ORDER BY 2, 3
```

Resultado:

NOMEM	COL-2	COL-3
AGUIRRE, AUREO	2	19
MARTIN, MICAELA	3	30
CAMPOS, ROMULO	5	4
GARCIA, OCTAVIO	5	21
TORRES, HORACIO	6	6
LASA, MARIO	6	9
DIEZ, AMELIA	8	19
TEROL, LUCIANO	9	9
PEREZ, MARCOS	10	18
PEREZ, SABINA	10	21

***8.10).** Azucena Muñoz recibió un préstamo para vivienda el día en que ingresó en la empresa con vencimientos anuales a 180 días del día y mes de su ingreso. Hallar la fecha en que vence la anualidad del préstamo correspondiente al año actual.

Solución:

```
SELECT   NUMEM,
         FECIN +
         (YEAR (CURRENT DATE) - YEAR (FECIN) ) YEARS
         +  180 DAYS, CURRENT DATE
FROM     TEMPLE
WHERE    NOMEM = 'MUÑOZ, AZUCENA'
ORDER BY NUMEM
```

Resultado (suponiendo que CURRENT DATE es el 5.7.90):

NUMEM	COL-2	COL-3
410	1991-04-11	1990-07-05

***8.11).** Todos los empleados tienen un período de 6 meses después de su ingreso antes de firmar su contrato de empleo definitivo. Hallar para los empleados que este año cumplen menos de 40 años de edad, por orden alfabético: nombre y fecha de firma de su contrato definitivo.

Solución:

```
SELECT   NOMEM, FECIN + 6 MONTHS
FROM     TEMPLE
WHERE    YEAR (CURRENT DATE) - YEAR (FECNA) < 40
ORDER BY NOMEM
```

Resultado (suponiendo que CURRENT DATE = 5.7.1990):

NOMEM	COL-2
CAMPOS, ROMULO	1987-05-01
DURAN, LIVIA	1986-08-28
FIERRO, CLAUDIA	1989-05-19
LARA, DORINDA	1969-04-10
LARA, LUCRECIA	1988-05-01
MARTIN, MICAELA	1988-07-01
MORA, VALERIANA	1989-05-19
MORAN, CARMEN	1987-04-08
MUÑOZ, AZUCENA	1989-04-13
PEREZ, SABINA	1986-08-28
PINO, DIANA	1986-08-28
RUIZ, FABIOLA	1987-07-20
SANTOS, SANCHO	1988-07-21
SANZ, CORNELIO	1978-08-05
TORRES, HORACIO	1988-07-01
VAZQUEZ, HONORIA	1987-07-01

***8.12).** Claudita Fierro y Horaciete Torres tras un volcánico y fugaz noviazgo han decidido unirse eternamente en matrimonio mientras no se divorcien. La boda se celebrará dentro de 2 días, y tomarán 20 días de vacaciones para el viaje de novios. La empresa le entregará a cada uno como regalo de boda un 1 % de su salario actual por cada año de servicio. Hallar: la fecha de la boda, la fecha en que se incorporarán al trabajo después del viaje de novios y el regalo de boda correspondiente a cada uno de ellos.

Solución:

```
SELECT   NOMEM,
             CURRENT DATE + 2 DAYS AS BODA,
             CURRENT DATE + 23 DAYS AS FIN,
             SALAR * 0.01 * (YEAR ((CURRENT DATE + 2 DAYS) –FECIN)) AS
                                                          REGALO
FROM     TEMPLE
WHERE    NUMEM = 420 OR NUMEM = 490
ORDER BY NOMEM
```

Resultado (suponiendo que CURRENT DATE es el 31.12.89):

NOMEM	BODA	FIN	REGALO
FIERRO, CLAUDIA	1990-01-02	1990-01-23	40,00
TORRES, HORACIO	1990-01-02	1990-01-23	36,00

Solucionario Capítulo 9

"FUNCIONES DE COLUMNAS"

9.1). Hallar cuántos departamentos hay y el presupuesto anual medio de ellos.

Solución:

```
SELECT COUNT (*) AS DEPS, AVG (DEC(PRESU,7,2)) * 1000 AS PRESMED
FROM   TDEPTO
```

Resultado:

```
   DEPS          PRESMED
-----------      --------------
         9          77777,77
```

9.2). Como la pregunta anterior, pero para los departamentos que no tienen director en propiedad.

Solución:

```
SELECT COUNT (*), AVG (DEC(PRESU,7,2)) * 1000
FROM   TDEPTO
WHERE  TIDIR <> 'P'
```

Resultado:

```
  COL-1          COL-2
-------------    -------------
         3          70000
```

***9.3).** Para los departamentos 111 y 112 hallar la media de los años de servicio de sus empleados en el día de hoy.

Solución:

```
SELECT AVG ( YEAR (CURRENT DATE - FECIN) ),
            CURRENT DATE
FROM   TEMPLE
WHERE  NUMDE = 111 OR NUMDE = 112
```

Resultado (suponiendo que CURRENT DATE es el 5.7.90):

```
  COL-1            COL-2
---------------    ----------------
        11         1990-07-05
```

***9.4).** Para los departamentos 111 y 112 hallar la media de los años de servicio de sus empleados el día 31.12.86.

Solución:

```
SELECT  AVG ( YEAR ('31.12.1986' - FECIN) )
FROM    TEMPLE
WHERE   (NUMDE = 111 OR NUMDE = 112) AND
        FECIN < '31.12.1986'
```

Resultado:

```
COL-1
----------
        12
```

9.5). (Como el ejercicio 6.2). Hallar por orden de número de empleado el nombre y salario total (salario más comisión) de los empleados cuyo salario total supera al salario mínimo en 3000 euros mensuales.

Solución:

```
SELECT   NUMEM, NOMEM, SALAR + COMIS
FROM     TEMPLE
```

```
WHERE    SALAR + COMIS > (SELECT MIN(SALAR) + 3000 FROM TEMPLE)
ORDER BY NUMEM
```

Resultado:

NUMEM	NOMEM	COL-3
120	LASA, MARIO	4600
160	AGUIRRE, AUREO	4200
180	PEREZ, MARCOS	5300
270	GARCIA, OCTAVIO	4600

***9.6).** Para los empleados que han ingresado en la empresa en los últimos 5 años, hallar la edad media en años cumplidos de la edad a la que han ingresado.

Solución:

```
SELECT  AVG ( YEAR (FECIN - FECNA) ),
            CURRENT DATE
FROM  TEMPLE
WHERE FECIN BETWEEN CURRENT DATE - 5 YEARS AND
            CURRENT DATE
```

Resultado (suponiendo que CURRENT DATE es el 5.7.90)*:*

COL-1	COL-2
19	1990-07-05

9.7). Hallar la masa salarial anual (salario más comisión) de la empresa (se suponen 14 pagas anuales).

Solución:

```
SELECT (SUM (SALAR) + SUM (COMIS)) * 14
FROM TEMPLE
```

Resultado:

COL-1
1633800

Obsérvese que no se obtiene el resultado correcto si se especifica:

```
SELECT (SUM (SALAR + COMIS)) * 14
FROM  TEMPLE
```

En efecto, los valores *Nulos* de COMIS hacen nula su suma con SALAR, y estos valores no entran entonces en el cálculo de la función SUM. Análogamente si todos los valores de COMIS fueran *Nulos*, tampoco la primera solución daría el valor correcto, pues SUM (COMIS) sería *Nulo* y por tanto también lo sería SUM (SALAR) + SUM (COMIS). Para darse cuenta de si este es el caso, convendría obtener los sumandos separadamente:

```
SELECT SUM (SALAR), SUM (COMIS),
            (SUM (SALAR) + SUM (COMIS)) * 14
FROM TEMPLE
```

Lo mejor es usar la función escalar COALESCE, ya conocida, ó la VALUE, que es equivalente.

```
SELECT  VALUE ( SUM (SALAR)+SUM (COMIS),
                    SUM (SALAR) ) * 14
FROM  TEMPLE
```

En esta sentencia el resultado será la suma de salarios si todas las comisiones son *Nulas*. También se podría especificar así:

```
SELECT  (SUM (SALAR + VALUE (COMIS, 0) ) ) * 14
FROM  TEMPLE
```

***9.8).** Hallar cuántos empleados han ingresado en el año actual.

Solución:

```
SELECT  COUNT (*)
FROM  TEMPLE
WHERE  YEAR (FECIN) = YEAR (CURRENT DATE)
```

Resultado (suponiendo que la fecha actual es 8.7.88):

```
COL-1
--------
       6
```

9.9). Hallar el salario medio de los empleados cuyo salario no supera en más de 20 % al salario mínimo de los empleados que tienen algún hijo y su salario medio por hijo es mayor que 1000 euros.

Solución:

```
SELECT  AVG (SALAR)
FROM  TEMPLE
WHERE  SALAR <= (SELECT 1.20 * MIN(SALAR)
                        FROM   TEMPLE
                        WHERE  NUMHI <> 0 AND
                               SALAR > NUMHI * 1000)
```

Resultado:

```
 COL-1
--------------
1896,1538
```

9.10). Hallar la diferencia entre el salario más alto y el más bajo.

Solución:

```
SELECT  MAX (SALAR) - MIN (SALAR)
FROM  TEMPLE
```

Resultado:

```
COL-1
---------
   6200
```

***9.11).** Hallar la edad media en años cumplidos en el día de hoy de los empleados que tienen más de 2 hijos.

Solución:

```
SELECT AVG (YEAR (CURRENT DATE - FECNA))
FROM  TEMPLE
WHERE NUMHI > 2
```

Resultado (suponiendo que la fecha actual es 8.7.90):

```
COL-1
--------
      48
```

9.12). Hallar el presupuesto medio de los departamentos cuyo presupuesto supera al presupuesto medio de los departamentos cuyo presupuesto supera al presupuesto medio de los departamentos.

Solución:

```
SELECT AVG (PRESU)
FROM TDEPTO
WHERE PRESU > (SELECT AVG(PRESU)
                  FROM   TDEPTO
                  WHERE  PRESU > (SELECT AVG(PRESU)
                                     FROM   TDEPTO) )
```

Resultado:

```
 COL-1
-----------
   135,00
```

9.13). Hallar el número medio de hijos por empleado para todos los empleados que no tienen más de dos hijos.

Solución:

```
SELECT  AVG (NUMHI)
FROM  TEMPLE
WHERE NUMHI <= 2
```

Resultado:

```
 COL-1
------------
          0
```

Obsérvese que en el resultado no aparecen cifras decimales. Esto se debe a que NUMHI es entero y por tanto el resultado de la función también lo es. Para obtener decimales habría que hacer que el argumento de la función fuera decimal. Puede forzarse esto haciendo intervenir a un número decimal en la operación. Por ejemplo:

```
SELECT  AVG ( 1.000 * NUMHI)
FROM  TEMPLE
WHERE NUMHI <= 2
```

El resultado entonces es:

```
  COL-1
------------
0,7037037
```

En un capítulo anterior dedicado a funciones escalares se vió que hay unas que permiten moldear valores. En concreto, la invocación DECIMAL (NUMHI, 5, 3) moldea el valor entero de NUMHI como un número decimal de 5 cifras (precisión) de las que 3 son decimales (escala). Por consiguiente la sentencia anterior para obtener la media con decimales se podría especificar así:

```
SELECT AVG (DECIMAL (NUMHI, 5, 3) )
FROM  TEMPLE
WHERE NUMHI <= 2
```

***9.14).** (Como el ejercicio 8.4). Obtener por orden alfabético los nombres y salarios de los empleados que o bien ingresaron después del 1.1.88 o bien antes y además tienen un salario inferior al salario más bajo de los que ingresaron con posterioridad al 1.1.88 incrementado en un 100 %.

Solución:

```
SELECT   NOMEM, SALAR
FROM    TEMPLE
WHERE    FECIN > '1.1.1988' OR
         (FECIN <= '1.1.1988' AND
         SALAR < (SELECT MIN(SALAR) * 2
                     FROM   TEMPLE
                     WHERE  FECIN > '1.1.1988'))
ORDER BY NOMEM
```

Resultado:

NOMEM	SALAR
FIERRO, CLAUDIA	4000
LARA, LUCRECIA	1850
MARTIN, MICAELA	1800
MORA, VALERIANA	2100
MUÑOZ, AZUCENA	1750
RUIZ, FABIOLA	1900
SANTOS, SANCHO	1000
TORRES, HORACIO	1800

Solucionario Capítulo 10

"CONSULTAS CON AGRUPAMIENTO DE FILAS"

***10.1).** Hallar el salario medio y la edad media en años para cada grupo de empleados con igual comisión y para los que no la tengan.

Solución:

```
SELECT   COMIS, AVG (SALAR),
              AVG (YEAR (CURRENT DATE - FECNA) )
FROM     TEMPLE
GROUP BY COMIS
ORDER BY COMIS
```

Resultado (suponiendo CURRENT DATE = 5.7.90):

COMIS	COL-2	COL-3
500	4800,00	55
800	3800,00	45
900	2800,00	41
1000	2200,00	28
1100	3166,66	49
1200	1000,00	20
-	3285,00	36

10.2). Para los departamentos en los que hay algún empleado cuyo salario sea mayor que 4000 euros al mes hallar el número de empleados y la suma de sus salarios, comisiones y número de hijos.

Solución:

```
SELECT   NUMDE, COUNT (*), SUM (SALAR),
              SUM (COMIS), SUM (NUMHI)
FROM     TEMPLE
WHERE    NUMDE IN (SELECT DISTINCT NUMDE
                          FROM   TEMPLE
                          WHERE  SALAR > 4000 )
GROUP BY NUMDE
ORDER BY NUMDE
```

Resultado:

NUMDE	COL-2	COL-3	COL-4	COL-5
100	3	15500	-	8
110	3	8950	500	4
121	4	12400	-	8
122	5	16200	-	4
130	3	11100	-	5

También se podría especificar así:

```
SELECT   NUMDE, COUNT (*), SUM (SALAR),
              SUM (COMIS), SUM (NUMHI)
FROM     TEMPLE
GROUP BY NUMDE
HAVING   NUMDE IN (SELECT DISTINCT NUMDE
                          FROM   TEMPLE
                          WHERE  SALAR > 4000)
ORDER BY NUMDE
```


***10.3).** Para los departamentos en los que la antigüedad media de sus empleados supera a la de la empresa, hallar el salario mínimo, el medio y el máximo.

Solución:

```
SELECT   NUMDE, MIN (SALAR), AVG (SALAR), MAX (SALAR)
FROM     TEMPLE
GROUP BY NUMDE
HAVING   AVG (DAYS (CURRENT DATE) - DAYS (FECIN) ) >
            (SELECT AVG (DAYS (CURRENT DATE) - DAYS (FECIN) )
                  FROM   TEMPLE)
ORDER BY NUMDE
```

Resultado (suponiendo CURRENT DATE = 5.7.90):

NUMDE	COL-2	COL-3	COL-4
100	3800	5166,66	7200
120	2700	2700,00	2700
121	1900	3100,00	4400

***10.4).** Para los departamentos en los que haya algún empleado con más de 10 años de antigüedad y tales que la media de hijos por cada uno de estos empleados sea superior a 1, hallar el salario medio de estos empleados.

Solución:

Al ser el SQL un lenguaje artificial y especializado, diseñado expresamente para el tratamiento de datos relacionales, se le ha podido dotar de la cualidad de ser preciso, es decir, de que el significado de una sentencia esté bien definido. No ocurre lo mismo con los lenguajes naturales, en los que la evolución no planificada impuesta por el uso posibilita que haya ambigüedades causantes de que un mismo texto pueda interpretarse de diferentes maneras. En el presente ejercicio por ejemplo podrían darse dos interpretaciones, y por tanto dos soluciones SQL correctas, aunque distintas. Este tipo de situación puede presentarse con frecuencia y es misión del analista definir el problema con precisión aclarando cuál de las interpretaciones posibles es la verdaderamente buscada.

Interpretación 1:

```
SELECT   NUMDE, AVG (SALAR)
FROM     TEMPLE
WHERE    YEAR (CURRENT DATE - FECIN) >= 10
GROUP BY NUMDE
HAVING   AVG ( DECIMAL (NUMHI, 5, 3)) > 1
ORDER BY NUMDE
```

Obsérvese que se ha tranformado NUMHI en decimal antes de aplicar la función AVG para que el resultado de ésta tenga decimales.

Resultado 1 (suponiendo que CURRENT DATE es el 5.7.90):

NUMDE	COL-2
100	5166,66
111	2800,00
112	3250,00
120	2700,00
121	3500,00
130	3550,00

Interpretación 2:

```
SELECT   NUMDE, AVG (SALAR)
FROM     TEMPLE
```

```
WHERE    NUMDE IN (SELECT DISTINCT NUMDE
                   FROM   TEMPLE
                   WHERE  YEAR (CURRENT DATE - FECIN) >= 10 )
GROUP BY NUMDE
HAVING   AVG ( DECIMAL (NUMHI, 5, 3)) > 1
ORDER BY NUMDE
```

Resultado 2 (suponiendo que CURRENT DATE es el 5.7.90):

NUMDE	COL-2
100	5166,66
110	2983,33
120	2700,00
121	3100,00
130	3700,00

10.5). Agrupando por número de hijos, hallar la media por hijo del salario total (salario y comisión).

Interpretación 1:

```
SELECT    NUMHI, SUM (SALAR) / SUM (NUMHI),
          SUM (COMIS) / SUM (NUMHI)
FROM      TEMPLE
WHERE     NUMHI <> 0
GROUP BY  NUMHI
ORDER BY  NUMHI
```

Resultado 1:

NUMHI	COL-2	COL-3
1	2607	300
2	1762	308
3	1033	150
4	750	-
5	580	-
6	1200	-

Interpretación 2:

```
SELECT    NUMHI, AVG (SALAR) / NUMHI,
          AVG (COMIS) / NUMHI
FROM      TEMPLE
WHERE     NUMHI <> 0
GROUP BY  NUMHI
ORDER BY  NUMHI
```

Resultado 2:

NUMHI	COL-2	COL-3
1	2607,14	1050,00
2	1762,50	462,50
3	1033,33	300,00
4	750,00	-
5	580,00	-
6	1200,00	-

10.6). Para cada departamento, hallar la media de la comisión con respecto a los empleados que la reciben y con respecto al total de empleados.

Solución:

```
SELECT   NUMDE, AVG (COMIS),
         SUM (COMIS) / COUNT (*)
FROM     TEMPLE
GROUP BY NUMDE
ORDER BY NUMDE
```

Resultado:

NUMDE	COL-2	COL-3
100	-	-
110	500,00	166
111	1042,85	912
112	983,33	842
120	-	-
121	-	-
122	-	-
130	-	-

10.7). Para cada extensión telefónica hallar cuántos empleados la usan y el salario medio de éstos.

Solución:

```
SELECT   EXTEL, COUNT (*), AVG (SALAR)
FROM     TEMPLE
GROUP BY EXTEL
ORDER BY EXTEL
```

Resultado:

EXTEL	COL-1	COL-2
200	1	3800,00
220	1	7200,00
250	1	4500,00
340	1	4400,00
350	2	3050,00
360	1	1900,00
410	1	2900,00
450	1	4000,00
480	1	4200,00
500	1	2150,00
508	1	4800,00
550	1	2000,00
610	1	4500,00
620	2	3925,00
650	1	2100,00
660	1	1750,00
740	1	3100,00
750	2	2250,00
760	3	2333,33
780	2	1425,00
800	1	3800,00
810	1	2900,00
840	1	3500,00
850	1	2800,00
880	3	1900,00
910	1	2700,00

10.8). Para cada extensión telefónica y cada departamento hallar cuántos empleados la usan y el salario medio de éstos.

Solución:

```
SELECT   EXTEL, NUMDE, COUNT (*), AVG (SALAR)
FROM     TEMPLE
GROUP BY NUMDE, EXTEL
ORDER BY EXTEL, NUMDE
```

Resultado:

EXTEL	NUMDE	COL-3	COL-4
200	100	1	3800,00
220	100	1	7200,00
250	100	1	4500,00
340	121	1	4400,00
350	121	2	3050,00
360	121	1	1900,00
410	130	1	2900,00
450	130	1	4000,00
480	130	1	4200,00
500	110	1	2150,00
508	110	1	4800,00
550	110	1	2000,00
610	122	1	4500,00
620	122	2	3925,00
650	122	1	2100,00
660	122	1	1750,00
740	111	1	3100,00
750	111	2	2250,00
760	111	3	2333,33
780	111	2	1425,00
800	112	1	3800,00
810	112	1	2900,00
840	112	1	3500,00
850	112	1	2800,00
880	112	3	1900,00
910	120	1	2700,00

10.9). Hallar los números de extensión telefónica mayores de los diversos departamentos, sin incluir los números de éstos.

Solución:

```
SELECT   DISTINCT MAX (EXTEL)
FROM     TEMPLE
GROUP BY NUMDE
ORDER BY 1
```

Resultado:

COL-1
250
360
480
550
660
780
880
910

10.10). Para cada extensión telefónica hallar el número de departamentos a los que sirve.

Solución:

```
SELECT   EXTEL, COUNT (DISTINCT NUMDE)
FROM     TEMPLE
GROUP BY EXTEL, NUMDE
ORDER BY EXTEL
```

Resultado:

EXTEL	COL-2
200	1
220	1
250	1
340	1
350	1
360	1
410	1
450	1
480	1
500	1
508	1
550	1
610	1
620	1
650	1
660	1
740	1
750	1
760	1
780	1
800	1
810	1
840	1
850	1
880	1
910	1

10.11). Para los departamentos en los que algún empleado tiene comisión, hallar cuántos empleados hay en promedio por cada extensión telefónica.

Solución:

```
SELECT   NUMDE,
         DECIMAL (COUNT (*), 6, 3) / COUNT (DISTINCT EXTEL)
FROM     TEMPLE
GROUP BY NUMDE
HAVING   NUMDE IN (SELECT DISTINCT NUMDE FROM   TEMPLE
                   WHERE  COMIS IS NOT NULL )
ORDER BY NUMDE
```

Resultado:

NUMDE	COL-2
110	1,00
111	2,00
112	1,40

10.12). Para los empleados que tienen comisión, hallar para los departamentos cuántos empleados hay en promedio por cada extensión telefónica.

Solución:

```
SELECT   NUMDE,
DECIMAL ( COUNT (*), 6, 3) / COUNT (DISTINCT EXTEL)
FROM     TEMPLE
WHERE    COMIS IS NOT NULL
GROUP BY NUMDE
ORDER BY NUMDE
```

Resultado:

```
NUMDE       COL-2
----------  ---------
       110      1,00
       111      1,75
       112      1,20
```

10.13). Obtener por orden creciente los números de extensiones telefónicas de los departamentos que tienen más de dos y que son compartidas por menos de 4 empleados, excluyendo las que no son compartidas.

Solución:

```
SELECT   EXTEL
FROM     TEMPLE
WHERE    NUMDE IN (SELECT   NUMDE
                   FROM     TEMPLE
                   GROUP BY NUMDE
                   HAVING   COUNT (DISTINCT EXTEL) > 2)
GROUP BY EXTEL
HAVING   COUNT (*) BETWEEN 2 AND 3
ORDER BY EXTEL
```

Resultado:

```
EXTEL
----------
       350
       620
       750
       760
       780
       880
```

10.14). Para los departamentos cuyo salario medio supera al de la empresa, hallar cuántas extensiones telefónicas tienen.

Solución:

```
SELECT   NUMDE, COUNT (DISTINCT EXTEL)
FROM     TEMPLE
GROUP BY NUMDE
HAVING   AVG (SALAR) > (SELECT AVG (SALAR)
                        FROM   TEMPLE )
ORDER BY NUMDE
```

Resultado:

NUMDE	COL-2
100	3
121	3
122	4
130	3

10.15). Para cada centro hallar los presupuestos medios de los departamentos dirigidos en propiedad y en funciones, excluyendo del resultado el número del centro.

Solución:

```
SELECT   DISTINCT TIDIR, AVG (PRESU)
FROM     TDEPTO
GROUP BY NUMCE, TIDIR
ORDER BY 1, 2
```

Resultado:

TIDIR	COL-2
F	30,00
F	100,00
F	110,00
P	55,00
P	120,00

10.16). Hallar el máximo valor de la suma de los salarios de los departamentos

Solución:

```
SELECT   NUMDE, SUM (SALAR)
FROM     TEMPLE
GROUP BY NUMDE
HAVING   SUM (SALAR) >= ALL (SELECT   SUM (SALAR)
                             FROM     TEMPLE
                             GROUP BY NUMDE)
ORDER BY NUMDE
```

Resultado:

NUMDE	COL-2
112	18700

10.17). Hallar la suma de presupuestos de la empresa, desglosándola por centros y departamentos. Como NUMCE puede ser *Nulo* deseamos distinguir este caso en la salida.

Solución:

```
SELECT NUMCE, NUMDE, SUM ( PRESU) AS TOT_PRESU,
          GROUPING (NUMCE) AS NUMCE_NUL
FROM TDEPTO
GROUP BY ROLLUP ( (NUMCE), (NUMDE))
ORDER BY NUMCE, NUMDE
```

Resultado:

NUMCE	NUMDE	TOT_PRESU	NUMCE_NUL
10	100	120	0
10	120	30	0
10	121	20	0
10	122	60	0
10	130	20	0

10	-	250	0
20	110	150	0
20	111	110	0
20	112	90	0
20	-	250	0
-	123	100	1
-	-	100	1
-	-	600	0

Solucionario Capítulo 11

"CONSULTAS SOBRE VARIAS TABLAS"

11.1). Para los departamentos cuyo director lo sea en funciones hallar el número de empleados y la suma de sus salarios, comisiones y número de hijos.

Solución 1 (con sentencia subordinada no correlacionada):

```
SELECT   NUMDE, COUNT (*), SUM (SALAR),
         SUM (COMIS), SUM (NUMHI)
FROM     TEMPLE
WHERE    NUMDE IN (SELECT NUMDE
                   FROM   TDEPTO
                   WHERE  TIDIR = 'F')
GROUP BY NUMDE
ORDER BY NUMDE
```

Resultado:

NUMDE	COL-2	COL-3	COL-4	COL-5
111	8	17450	7300	8
120	1	2700	-	3

Solución 2 (con sentencia subordinada correlacionada):

```
SELECT   NUMDE, COUNT (*), SUM (SALAR),
         SUM (COMIS), SUM (NUMHI)
FROM     TEMPLE E
WHERE    EXISTS (SELECT *
                 FROM   TDEPTO
                 WHERE  TIDIR = 'F' AND
                        E.NUMDE = NUMDE    )
GROUP BY NUMDE
ORDER BY NUMDE
```

Recuérdese que una sentencia subordinada correlacionada se evalúa una vez por cada fila de la sentencia principal. En nuestro caso, se evaluará una vez por cada fila de TEMPLE. En cambio la no correlacionada (*Solución 1*) se evalúa una sola vez. Esto significa que para una tabla TDEPTO pequeñá la solución 1 será probablemente más eficiente que la solución 2. Si TDEPTO es mediana o grande (unos miles de filas o más), y hay índices definidos sobre su clave (NUMDE), será probablemente más eficiente la correlacionada.

Solución 3 (con yunción):

```
SELECT   E.NUMDE, COUNT (*), SUM (SALAR),
         SUM (COMIS), SUM (NUMHI)
FROM     TEMPLE E, TDEPTO D
WHERE    E.NUMDE = D.NUMDE AND TIDIR = 'F'
GROUP BY E.NUMDE
ORDER BY E.NUMDE
```

***11.2).** Para los departamentos ubicados en la calle de Alcalá en los que haya algún empleado con más de 10 años de antigüedad y tales que la media de hijos por cada uno de estos empleados sea superior a 1, hallar el salario medio de estos empleados.

Interpretación 1:

```
SELECT   NUMDE, AVG (SALAR)
FROM     TEMPLE
```

```
WHERE    NUMDE IN (SELECT NUMDE
                   FROM   TDEPTO
 WHERE  NUMCE IN (SELECT NUMCE
                                    FROM   TCENTR
                                    WHERE  SEÑAS LIKE
                                    '%ALCALA%') )
          AND
          YEAR (CURRENT DATE - FECIN) >= 10
    GROUP BY NUMDE
    HAVING   AVG ( DECIMAL (NUMHI, 6, 3)) > 1
    ORDER BY NUMDE
```

Resultado:

NUMDE	COL-2
100	5166,66
120	2700,00
121	3100,00
130	3700,00

Obsérvese que se ha moldeado NUMHI como decimal antes de aplicar la función AVG para que el resultado de ésta tenga decimales.

También se podría especificar con una sentencia subordinada correlacionada o con yunción. La formulación de esta última es más natural:

```
SELECT   E.NUMDE, AVG (SALAR)
FROM     TEMPLE E, TDEPTO D, TCENTR C
WHERE    E.NUMDE = D.NUMDE AND
         D.NUMCE = C.NUMCE AND
         SEÑAS LIKE '%ALCALA%' AND
         YEAR (CURRENT DATE - FECIN) >= 10
GROUP BY E.NUMDE
HAVING   AVG ( DECIMAL (NUMHI, 6, 3)) > 1
ORDER BY 1
```

Interpretación 2:

```
SELECT   NUMDE, AVG (SALAR)
FROM     TEMPLE
WHERE    NUMDE IN (SELECT NUMDE
                   FROM   TDEPTO
                   WHERE  NUMCE IN (SELECT NUMCE
                                    FROM   TCENTR
                                    WHERE  SEÑAS LIKE '%ALCALA%') )
         AND
         NUMDE IN (SELECT DISTINCT NUMDE
                   FROM   TEMPLE
                   WHERE  YEAR (CURRENT DATE - FECIN) >= 10 )
GROUP BY NUMDE
HAVING   AVG (1.000 * NUMHI) > 1
ORDER BY NUMDE
```

Formulación con yunción y sentencia subordinada correlacionada:

```
SELECT   E.NUMDE, AVG (SALAR)
FROM     TEMPLE E, TDEPTO D, TCENTR C
```

```
WHERE    E.NUMDE = D.NUMDE AND
         D.NUMCE = C.NUMCE AND
         SEÑAS LIKE '%ALCALA%'
         AND
          EXISTS (SELECT *
                        FROM   TEMPLE E2
                        WHERE  E2.NUMDE = E.NUMDE AND
                         YEAR (CURRENT DATE - E2.FECIN) >= 10  )
GROUP BY E.NUMDE
HAVING   AVG (1.000 * NUMHI) > 1
ORDER BY 1
```

11.3). Para los departamentos cuyo presupuesto anual supera a 60 000 euros, hallar cuántos empleados hay en promedio por cada extensión telefónica.

Solución:

```
SELECT   NUMDE, COUNT (*) / COUNT (DISTINCT EXTEL)
FROM     TEMPLE
GROUP BY NUMDE
HAVING   NUMDE IN (SELECT DISTINCT NUMDE
                          FROM   TDEPTO
                          WHERE  PRESU > 60)
ORDER BY NUMDE
```

Solución con yunción:

```
SELECT   E.NUMDE, COUNT (*) / COUNT (DISTINCT EXTEL)
FROM     TEMPLE E, TDEPTO D
WHERE    E.NUMDE = D.NUMDE AND
          PRESU > 60
GROUP BY E.NUMDE
ORDER BY 1
```

Solución con sentencia correlacionada:

```
SELECT   NUMDE, COUNT (*) / COUNT (DISTINCT EXTEL)
FROM     TEMPLE E
GROUP BY NUMDE
HAVING   EXISTS (SELECT *
                        FROM   TDEPTO
                        WHERE  E.NUMDE = NUMDE AND
                         PRESU > 60)
ORDER BY NUMDE
```

Resultado:

NUMDE	COL-2
100	1
110	1
111	2
112	1

11.4). Obtener por orden alfabético los nombres de empleados cuyo apellido empieza por G y trabajan en un departamento ubicado en algún centro de trabajo de la calle Alcalá.

Solución con yunción:

```
SELECT    NOMEM
FROM TEMPLE E, TDEPTO D, TCENTR C
WHERE E.NUMDE = D.NUMDE AND
          D.NUMCE = C.NUMCE AND
          NOMEM LIKE 'G%'   AND
          SEÑAS LIKE '%ALCALA%'
ORDER BY NOMEM
```

Solución con sentencia subordinada no correlacionada:

```
SELECT    NOMEM
FROM TEMPLE
WHERE NOMEM LIKE 'G%' AND
          NUMDE IN (SELECT NUMDE
                              FROM TDEPTO
                              WHERE NUMCE IN (SELECT DISTINCT NUMCE
                                                          FROM TCENTR
                                                          WHERE SEÑAS LIKE '%ALCALA%'))
ORDER BY NOMEM
```

Solución con sentencia subordinada correlacionada:

```
SELECT    NOMEM
FROM TEMPLE E
WHERE NOMEM LIKE 'G%' AND
          EXISTS (SELECT *
                    FROM   TDEPTO D
                    WHERE  NUMDE = E.NUMDE AND
                              EXISTS (SELECT *
                                        FROM   TCENTR
                                        WHERE  NUMCE = D.NUMCE AND
                                                  SEÑAS LIKE '%ALCALA%'))
ORDER BY NOMEM
```

Resultado:

```
          NOMEM
------------------------------
GALVEZ, PILAR
GARCIA, AUGUSTO
GIL, GLORIA
```

11.5). Hallar por orden alfabético los distintos nombres de los empleados que son directores en funciones.

Solución con yunción:

```
SELECT   DISTINCT NOMEM
FROM     TDEPTO D, TEMPLE E
WHERE    D.DIREC = E.NUMEM AND
             TIDIR = 'F'
ORDER BY NOMEM
```

Solución con sentencia subordinada no correlacionada:

```
SELECT   DISTINCT NOMEM
FROM     TEMPLE
WHERE    NUMEM IN (SELECT DIREC
                          FROM   TDEPTO
```

```
                                    WHERE  TIDIR = 'F')
            ORDER BY NOMEM
```

Resultado:

```
     NOMEM
-------------------------
PEREZ, JULIO
PEREZ, MARCOS
```

11.6). Para todos los departamentos que no sean de dirección ni de sectores, hallar número de departamento y sus extensiones telefónicas, por orden creciente de departamento y, dentro de éste, por número de extensión creciente.

Solución 1 :

```
SELECT   DISTINCT D.NUMDE, EXTEL
FROM     TEMPLE E, TDEPTO D
WHERE    D.NUMDE = E.NUMDE AND
             NOMDE NOT LIKE '%DIRECC%' AND
             NOMDE NOT LIKE '%SECTOR%'
ORDER BY D.NUMDE, EXTEL
```

Solución 2:

```
SELECT   DISTINCT NUMDE, EXTEL
FROM     TEMPLE
WHERE    NUMDE IN (SELECT NUMDE
                           FROM   TDEPTO
                           WHERE  NOMDE NOT LIKE '%DIRECC%' AND
                                  NOMDE NOT LIKE '%SECTOR%')
ORDER BY NUMDE, EXTEL
```

Resultado:

NUMDE	EXTEL
120	910
121	340
121	350
121	360
122	610
122	620
122	650
122	660
130	410
130	450
130	480

11.7). A los distintos empleados que son directores en funciones se les asignará una gratificación del 5 % de su salario. Hallar por orden alfabético los nombres de estos empleados y la gratificación correspondiente a cada uno.

Solución 1:

```
SELECT   DISTINCT NOMEM, 0.05 * SALAR
FROM     TDEPTO D, TEMPLE E
WHERE    D.DIREC = E.NUMEM AND
              TIDIR = 'F'
ORDER BY NOMEM
```

Solución 2:

```
SELECT   DISTINCT NOMEM, 0.05 * SALAR
FROM     TEMPLE
WHERE    NUMEM IN (SELECT DIREC
                   FROM   TDEPTO
                   WHERE  TIDIR = 'F')
ORDER BY NOMEM
```

Resultado:

```
     NOMEM              COL-2
-----------------------  ---------
PEREZ, JULIO             220,00
PEREZ, MARCOS            240,00
```

11.8). Hallar si hay algún departamento en la tabla TDEPTO cuyo centro de trabajo no exista en la tabla TCENTR.

Solución correlacionada:

```
SELECT   *
FROM     TDEPTO D
WHERE    NOT EXISTS (SELECT *
                     FROM   TCENTR
                     WHERE  NUMCE = D.NUMCE)
ORDER BY NOMDE
```

Solución no correlacionada:

```
SELECT   *
FROM     TDEPTO
WHERE    NUMCE IS NULL OR NUMCE NOT IN (SELECT DISTINCT NUMCE
                                        FROM TCENTR)
ORDER BY NOMDE
```

Resultado:

NUMDE	NUMCE	DIREC	TIDIR	PRESU	DEPDE	NOMDE
123	-	150	F	100	121	PERSONAL CONTRATADO

No existe ningún centro en la Tabla de Centros con valor nulo.

11.9). Hallar si hay algún departamento de reciente creación que aún no tenga empleados asignados ni director en propiedad.

Solución (correlacionada):

```
SELECT   *
FROM     TDEPTO
WHERE    TIDIR = 'F' AND
         NOT EXISTS (SELECT *
                     FROM   TEMPLE
                     WHERE  NUMDE = TDEPTO.NUMDE)
ORDER BY NOMDE
```

Solución (no correlacionada):

```
SELECT   *
FROM     TDEPTO
WHERE    TIDIR = 'F' AND
             NUMDE NOT IN (SELECT DISTINCT NUMDE
                                  FROM   TEMPLE)
ORDER BY NOMDE
```

Resultado:

NUMDE	NUMCE	DIREC	TIDIR	PRESU	DEPDE	NOMDE
123	-	150	F	100	121	PERSONAL CONTRATADO

11.10). Hallar por orden alfabético los nombres de los empleados que son directores de primer nivel, es decir, que dirigen departamentos de los que no dependen otros departamentos.

Solución con yunción y correlación:

```
SELECT   E.NOMEM
FROM     TDEPTO D, TEMPLE E
WHERE    E.NUMEM = D.DIREC AND
             NOT EXISTS (SELECT *
                                FROM   TDEPTO
                                WHERE  DEPDE = D.NUMDE)
ORDER BY E.NOMEM
```

Solución no correlacionada:

```
SELECT   NOMEM
FROM     TEMPLE
WHERE    NUMEM IN (SELECT DISTINCT DIREC
                                FROM   TDEPTO
                                WHERE  NUMDE NOT IN (SELECT DISTINCT DEPDE
                                                                FROM   TDEPTO
                                                                WHERE  DEPDE IS NOT NULL))
ORDER BY NOMEM
```

Resultado:

```
      NOMEM
-------------------------
CAMPS, AURELIO
GARCIA, AUGUSTO
GARCIA, OCTAVIO
PEREZ, JULIO
PEREZ, MARCOS
```

11.11). Hallar por orden alfabético los nombres de los empleados de los departamentos cuyo director en propiedad lo sea también en funciones de algún otro, excluyendo del resultado a los directores.

Solución:

```
SELECT          NOMEM
FROM            TEMPLE E, TDEPTO D
WHERE           E.NUMDE = D.NUMDE AND
                TIDIR = 'P' AND
                 EXISTS (SELECT *
```

```
                                    FROM   TDEPTO
                                    WHERE  TIDIR = 'F' AND
                                           DIREC = D.DIREC) AND
                            DIREC <> NUMEM
ORDER BY  NOMEM
```

Resultado:

```
      NOMEM
--------------------------
CAMPOS, ROMULO
MORAN, CARMEN
PONS, CESAR
RUIZ, FABIOLA
VEIGA, JULIANA
```

11.12). Comprobar que no hay empleados cuyo departamento no esté en TDEPTO.

Solución:

```
SELECT      *
FROM        TEMPLE E
WHERE       NOT EXISTS (SELECT *
                        FROM   TDEPTO
                        WHERE  NUMDE = E.NUMDE)
ORDER BY  NOMEM
```

11.13). Hallar si hay algún departamento de reciente creación que aún no tenga empleados asignados, excepto el director en propiedad.

Solución:

```
SELECT      *
FROM        TDEPTO D
WHERE       TIDIR = 'P' AND
            NOT EXISTS (SELECT *
                        FROM   TEMPLE
                        WHERE  NUMDE = D.NUMDE AND
                        NUMEM <> D.DIREC)
ORDER BY  NOMDE
```

11.14). Comprobar que todos los empleados que son directores de departamento existen en la tabla de empleados.

Solución:

```
SELECT   *
FROM    TDEPTO
WHERE   NOT EXISTS (SELECT    *
                    FROM     TEMPLE
                    WHERE    TDEPTO.DIREC = NUMEM)
```

11.15). Comprobar que los directores en propiedad son empleados de su departamento.

Solución:

```
SELECT      *
FROM        TDEPTO D
WHERE       TIDIR = 'P' AND
            NOT EXISTS (SELECT *
                        FROM    TEMPLE
                        WHERE   NUMDE = D.NUMDE AND
                        NUMEM = D.DIREC)
```

11.16). Comprobar que ningún director en propiedad es director inmediato en funciones de su director inmediato.

Solución:

```
SELECT      D1.*
FROM        TDEPTO D1, TDEPTO D2
WHERE       D1.DEPDE = D2.NUMDE AND
            D1.TIDIR = 'P' AND
            EXISTS   (SELECT *
                      FROM   TDEPTO
                      WHERE  D2.DEPDE = NUMDE AND
                             DIREC = D1.DIREC AND
                             TIDIR = 'F')
ORDER BY  D1.NOMDE
```

11.17). Obtener por orden alfabético todos los datos de los centros de trabajo en los que hay algún departamento cuyo director lo sea en funciones.

Solución:

```
SELECT      *
FROM        TCENTR
WHERE       EXISTS (SELECT *
                    FROM    TDEPTO
                    WHERE   NUMCE = TCENTR.NUMCE AND
                            TIDIR = 'F')
ORDER BY  NOMCE
```

Solución con yunción:

```
SELECT      DISTINCT C.*
FROM        TDEPTO D, TCENTR C
WHERE       C.NUMCE = D.NUMCE AND
            TIDIR = 'F'
ORDER BY    C.NOMCE
```

Resultado:

NUMCE	NOMCE	SEÑAS
20	RELACION CON CLIENTES	C. ATOCHA, 405, MADRID
10	SEDE CENTRAL	C. ALCALA, 820, MADRID

11.18). Hallar por orden alfabético los nombres de los empleados cuyo director de departamento es Marcos Pérez, bien en propiedad o bien en funciones, indicando cuál es el caso para cada uno de ellos.

Solución:

```
SELECT      E1.NOMEM, TIDIR
FROM        TEMPLE E1, TDEPTO D, TEMPLE E2
WHERE       E1.NUMDE = D.NUMDE AND
            DIREC = E2.NUMEM AND
            E2.NOMEM = 'PEREZ, MARCOS'
ORDER BY  1
```

Resultado:

NOMEM	TIDIR
AGUIRRE, AUREO	F
CAMPOS, ROMULO	P
DURAN, LIVIA	F

LARA, DORINDA	F
LARA, LUCRECIA	F
MORAN, CARMEN	P
PEREZ, MARCOS	P
PINO, DIANA	F
SANTOS, SANCHO	F
SANZ, LAVINIA	F
VAZQUEZ, HONORIA	F

11.19). Para cada empleado que es director, hallar por orden alfabético su nombre y la suma de los salarios de los empleados que están directamente a su cargo (o sea en los departamentos que él dirige), en dos grupos separados según sea en funciones o en propiedad.

Solución:

```
SELECT      E2.NOMEM, TIDIR, SUM (E1.SALAR)
FROM        TEMPLE E1, TDEPTO D, TEMPLE E2
WHERE       E1.NUMDE = D.NUMDE AND
            DIREC = E2.NUMEM
GROUP BY    E2.NOMEM, TIDIR
ORDER BY    1
```

Resultado:

NOMEM	TIDIR	COL-3
CAMPS, AURELIO	P	16200
GARCIA, AUGUSTO	P	11100
GARCIA, OCTAVIO	P	18700
LOPEZ, ANTONIO	P	15500
PEREZ, JULIO	F	2700
PEREZ, JULIO	P	12400
PEREZ, MARCOS	F	17450
PEREZ, MARCOS	P	8950

11.20). Hallar por orden alfabético los nombres de los empleados que dirigen departamentos de los que dependen otros departamentos, indicando cuántos empleados hay en total en éstos.

Solución:

```
SELECT      DIR.NOMEM, COUNT (*)
FROM        TDEPTO D, TDEPTO DEP, TEMPLE E, TEMPLE DIR
WHERE       DEP.DEPDE = D.NUMDE AND
            E.NUMDE = DEP.NUMDE AND
            D.DIREC = DIR.NUMEM
GROUP BY    DIR.NUMEM, DIR.NOMEM
ORDER BY    1
```

Resultado:

NOMEM	COL-2
LOPEZ, ANTONIO	7
PEREZ, JULIO	9
PEREZ, MARCOS	15

11.21). Cada director de departamento, tanto en propiedad como en funciones, es responsable de las promociones y sueldos de los empleados del departamento que dirige, excluido él mismo, y de las de algunos de los directores que dependen de él. Hallar para cada director su nombre y la masa salarial total de los empleados de cuya promoción es responsable, excluyendo a los directores que dependen de él, por orden alfabético.

Solución:

```
SELECT      DIR.NOMEM, SUM (E.SALAR)
FROM        TDEPTO D, TEMPLE DIR, TEMPLE E
WHERE       DIREC = DIR.NUMEM AND
             D.NUMDE = E.NUMDE AND
             DIR.NUMEM <> E.NUMEM
GROUP BY    DIR.NUMEM, DIR.NOMEM
ORDER BY    1
```

Resultado:

```
        NOMEM             COL-2
------------------------- --------
CAMPS, AURELIO            11700
GARCIA, AUGUSTO            6900
GARCIA, OCTAVIO           14900
LOPEZ, ANTONIO             8300
PEREZ, JULIO              10700
PEREZ, MARCOS             21600
```

11.22). Hallar por orden alfabético los nombres de los centros en los que hay algún director que dirige algún departamento en otro centro.

Solución:

```
SELECT      NOMCE
FROM        TCENTR C
WHERE             EXISTS (SELECT *
                    FROM   TDEPTO D
                    WHERE  NUMCE = C.NUMCE AND
                           EXISTS (SELECT *
                                   FROM   TDEPTO
                                   WHERE  DIREC = D.DIREC AND
                                    NUMCE <> C.NUMCE) )
ORDER BY    NOMCE
```

Resultado:

```
NOMCE
-----------
```

11.23). Hallar por orden alfabético los nombres de los empleados cuyo salario supera al salario medio de los departamentos en los que la masa salarial (suma de salarios de sus empleados) supera a la de su propio departamento.

Interpretación 1:

```
SELECT   NOMEM
FROM     TEMPLE E
WHERE    SALAR > ALL
                 (SELECT   AVG (SALAR)
                  FROM     TEMPLE
                  GROUP BY NUMDE
                  HAVING   SUM (SALAR) > (SELECT SUM (SALAR)
                                          FROM   TEMPLE
                                          WHERE  NUMDE = E.NUMDE))
ORDER BY NOMEM
```

Resultado 1:

```
NOMEM
----------------------------
AGUIRRE, AUREO
ALBA, ADRIANA
CAMPS, AURELIO
DIEZ, AMELIA
GALVEZ, PILAR
GARCIA, OCTAVIO
LASA, MARIO
LOPEZ, ANTONIO
MARTIN, MICAELA
PEREZ, SABINA
POLO, OTILIA
SANZ, CORNELIO
SANZ, LAVINIA
TEROL, LUCIANO
TORRES, HORACIO
```

Interpretación 2:

```
SELECT   NOMEM
FROM     TEMPLE E
WHERE    SALAR > (SELECT AVG(SALAR)
                  FROM   TEMPLE
                  WHERE  NUMDE IN (SELECT   DISTINCT NUMDE
                                   FROM     TEMPLE
                                   GROUP BY NUMDE
                                     HAVING   SUM (SALAR) >
                                              (SELECT SUM (SALAR)
                                               FROM   TEMPLE
                                               WHERE  NUMDE = E.NUMDE)))
ORDER BY NOMEM
```

Resultado 2:

```
NOMEM
----------------------------
AGUIRRE, AUREO
ALBA, ADRIANA
CAMPS, AURELIO
FIERRO, CLAUDIA
GALVEZ, PILAR
GARCIA, AUGUSTO
LOPEZ, ANTONIO
PEREZ, JULIO
PEREZ, MARCOS
POLO, OTILIA
PONS, CESAR
SANZ, CORNELIO
SANZ, LAVINIA
VEIGA, JULIANA
```

11.24). Hallar por orden alfabético los nombres de los departamentos cuyo presupuesto medio por empleado supera a la media de sus salarios.

Solución:

```
SELECT      NOMDE
FROM        TDEPTO D, TEMPLE E
WHERE       D.NUMDE = E.NUMDE
```

```
GROUP BY    D.NUMDE, PRESU, NOMDE
HAVING      PRESU / COUNT (*) > AVG (SALAR)
ORDER BY  NOMDE
```

Resultado:

```
NOMDE
----------
```

11.25). Para los empleados que trabajan en la calle de Atocha y comparten su extensión telefónica con otro con menor salario que ellos, hallar la suma de sus salarios por departamento y el nombre de éste, por orden alfabético.

Solución:

```
SELECT      D.NOMDE, SUM (E.SALAR)
FROM        TEMPLE E, TDEPTO D, TCENTR C
WHERE       E.NUMDE = D.NUMDE AND
            D.NUMCE = C.NUMCE AND
            SEÑAS LIKE '%ATOCHA%' AND
            EXISTS (SELECT *
                      FROM   TEMPLE
                      WHERE  EXTEL = E.EXTEL AND
                             SALAR < E.SALAR)
GROUP BY    D.NOMDE
ORDER BY  1
```

Resultado:

```
        NOMDE              COL-2
------------------------------ --------------
SECTOR INDUSTRIAL           7150
SECTOR SERVICIOS            2100
```

11.26). Hallar por orden alfabético los nombres de los empleados que comparten su extensión telefónica con otro de otro centro.

Solución:

```
SELECT      E.NOMEM
FROM        TEMPLE E, TDEPTO D
WHERE       E.NUMDE = D.NUMDE AND
            EXISTS (SELECT *
                      FROM   TEMPLE E2, TDEPTO D2
                       WHERE  E2.NUMDE = D2.NUMDE AND
                              E2.EXTEL = E.EXTEL AND
                              D2.NUMCE <> D.NUMCE )
ORDER BY  1
```

Resultado:

```
NOMEM
------------
```

11.27). Hallar cuántos empleados hay que compartan su extensión telefónica con otro de otro departamento.

Solución:

```
SELECT      COUNT (*)
FROM        TEMPLE E
WHERE       EXISTS (SELECT *
```

```
                    FROM  TEMPLE
                    WHERE  EXTEL = E.EXTEL AND
                    NUMDE <> E.NUMDE )
```

Resultado:

```
 COL-1
-----------
         0
```

11.28). Hallar por orden alfabético los nombres de los empleados que disfrutan de una extensión telefónica no compartida con otros. Hallar también para cada uno de ellos su salario y el salario medio de su departamento, excluyéndoles a ellos.

Solución:

```
SELECT      E1.NOMEM, E1.SALAR, AVG (E2.SALAR)
FROM        TEMPLE E1, TEMPLE E2
WHERE       E1.NUMDE = E2.NUMDE AND
            E1.NUMEM <> E2.NUMEM AND
            NOT EXISTS (SELECT *
                        FROM    TEMPLE
                        WHERE   EXTEL = E1.EXTEL AND
                                NUMEM <> E1.NUMEM )
GROUP BY    E1.NUMEM, E1.NOMEM, E1.SALAR
ORDER BY  1
```

Resultado:

NOMEM	SALAR	COL-3
AGUIRRE, AUREO	3100	2050,00
ALBA, ADRIANA	4500	5500.00
CAMPOS, ROMULO	2000	3475,00
CAMPS, AURELIO	4500	2925,00
DIEZ, AMELIA	2800	2650,00
FIERRO, CLAUDIA	4000	3550,00
FLOR, DOROTEA	2900	4100,00
GALVEZ, PILAR	3800	5850,00
GARCIA, AUGUSTO	4200	3450,00
GARCIA, OCTAVIO	3800	2483,33
LASA, MARIO	3500	2533,33
LOPEZ, ANTONIO	7200	4150,00
MORA, VALERIANA	2100	3525,00
MORAN, CARMEN	2150	3400,00
MUÑOZ, AZUCENA	1750	3612,50
PEREZ, JULIO	4400	2666,66
PEREZ, MARCOS	4800	2075,00
RUIZ, FABIOLA	1900	3500,00
TEROL, LUCIANO	2900	2633,33

***11.29).** Hallar por orden alfabético los nombres y salarios de los empleados cuyo salario supera a la media de salarios de los que ingresaron 2 años antes o después que él, excluyéndole.

Interpretación 1:

```
SELECT   NOMEM, SALAR
FROM     TEMPLE E
WHERE    SALAR > (SELECT AVG (SALAR)
                  FROM   TEMPLE
                  WHERE  YEAR (FECIN) = YEAR (E.FECIN) - 2 OR
                         YEAR (FECIN) = YEAR (E.FECIN) + 2
```

```
                                        AND
                                        NUMEM <> E.NUMEM)
        ORDER BY NOMEM
```

Resultado 1:

NOMEM	SALAR
AGUIRRE, AUREO	4500
CAMPS, AURELIO	4500
FIERRO, CLAUDIA	4000
GARCIA, AUGUSTO	4200
LASA, MARIO	3500
LOPEZ, ANTONIO	7200
MORA, VALERIANA	2100
PEREZ, JULIO	4400
POLO, OTILIA	3800

Interpretación 2:

```
SELECT   NOMEM, SALAR
FROM     TEMPLE E
WHERE    SALAR > (SELECT AVG (SALAR)
                  FROM   TEMPLE
                  WHERE  FECIN BETWEEN E.FECIN - 2 YEARS AND
                                       E.FECIN + 2 YEARS
                  AND
                  NUMEM <> E.NUMEM )
ORDER BY NOMEM
```

Resultado 2 :

NOMEM	SALAR
AGUIRRE, AUREO	4500
CAMPOS, ROMULO	2000
CAMPS, AURELIO	4500
FIERRO, CLAUDIA	4000
GARCIA, AUGUSTO	4200
LOPEZ, ANTONIO	7200
MORA, VALERIANA	2100
MORAN, CARMEN	2150
POLO, OTILIA	3800

***11.30).** Hallar por orden alfabético los nombres de los departamentos en los que hay algún empleado que cumpla este año más de 50 años de edad.

Solución:

```
SELECT   D.NOMDE
FROM     TDEPTO D
WHERE    EXISTS (SELECT *
                 FROM   TEMPLE
                 WHERE  NUMDE = D.NUMDE AND
YEAR (CURRENT DATE) - YEAR (FECNA) >= 50)
ORDER BY NOMDE
```

Solución con yunción:

```
SELECT          DISTINCT NOMDE
FROM            TDEPTO D, TEMPLE E
WHERE           D.NUMDE = E.NUMDE AND
```

```
                    YEAR (CURRENT DATE) - YEAR (FECNA) >= 50
ORDER BY  NOMDE
```

Resultado (con un valor de '11.11.1989' para el CURRENT DATE):

```
            NOMDE
----------------------------------
DIRECCION COMERCIAL
PERSONAL
SECTOR INDUSTRIAL
SECTOR SERVICIOS
```

11.31). Hallar el número de departamento y el salario máximo para los departamentos cuyo salario máximo es menor que el salario medio de los empleados de todos los demás departamentos.

Solución:

```
SELECT     NUMDE, MAX (SALAR)
FROM       TEMPLE E
GROUP BY NUMDE
HAVING     MAX (SALAR) < (SELECT AVG (SALAR)
                          FROM   TEMPLE
                          WHERE  NOT NUMDE = E.NUMDE )
ORDER BY NUMDE
```

Resultado:

```
NUMDE          COL-2
-----------    -----------
       111           3100
       120           2700
```

11.32). Supongamos que hay dos tablas, llamadas TCURSO y TCURRI. La primera tiene una sola columna llamada NOMCU y la segunda tiene dos, llamadas NOMEM y NOMCU. La tabla TCURSO tiene una fila por cada curso de idiomas impartido a los empleados de la empresa, con el nombre del curso correspondiente en la columna NOMCU. La TCURRI tiene una fila por cada empleado y curso, con el nombre de empleado en la columna NOMEM y el de curso en la NOMCU. El significado de una fila de TCURRI (cuyo nombre viene de curriculum) es que el empleado ha asistido al curso correspondiente.

Se desea hallar una relación de los nombres de empleados que han asistido a todos los cursos impartidos.

Este tipo de consultas en las que se buscan los valores de una tabla que están combinados con todos los de otra se llama DIVISION en Teoría de Bases de Datos Relacionales. En SQL se puede resolver con sentencias correlacionadas y el predicado EXISTS.

Solución:

```
SELECT DISTINCT X.NOMEM
FROM   TCURRI X
WHERE NOT EXISTS (SELECT *
                  FROM    TCURSO Y
                  WHERE  NOT EXISTS (SELECT *
                                     FROM    TCURRI Z
                                     WHERE  Z.NOMEM = X.NOMEM AND
                                            Z.NOMCU = Y.NOMCU))
ORDER BY 1
```

Como ya hemos dicho el resultado de un SELECT se puede construir ejecutando sus cláusulas en una secuencia, FROM, WHERE, etc. Esta interpretación está en el estilo de las expresiones estudiadas en el Algebra Relacional, de la teoría de Bases de datos Relacionales. Pero también se puede interpretar un

SELECT de SQL como una expresión del estilo de las de Cálculo Relacional. En nuestro ejercicio, esto nos llevaría a describir el SELECT anterior como: hallar todas las filas *x* de TCURRI tales que no hay ninguna fila *y* de TCURSO que combinada con la *x* no exista en TCURRI, o dicho de otra forma, hallar todas las filas *x* de TCURRI tales que al combinarlas con todas las filas *y* de TCURSO se obtengan filas existentes en TCURRI.

Supongamos los siguientes contenidos de las tablas (cursos de inglés, I, y francés, F):

TCURSO)

NOMCU
I
F

TCURRI)

NOMEM	NOMCU
PEPE	I
PACO	I
PACO	F

Veamos el funcionamiento del SELECT en el estilo algebraico, como una secuencia de pasos:

a) FROM.
Tomar las filas de TCURRI

b) WHERE.
Para cada fila ver si se cumple el predicado.
Consideremos la fila *x* = (PEPE, I). Si tomamos una fila de TCURSO tal que la *y* = (I), la combinación de *x* e *y* dan una fila *z* = (PEPE, I), que existe en TCURRI. Sin embargo si tomamos *y* = (F), entonces dan al combinarlas *z* = (PEPE, F), que no existe en TCURRI. Por tanto el predicado es Falso.
Consideremos ahora *x* = (PACO, I). Para *y* = (I), la fila *z* = (PACO, I) existe en TCURRI. Lo mismo pasa para *y* = (F), pues *z* =(PACO, F). Por tanto el predicado es Verdadero.

Resultado:

NOMEM
PACO

11.33). Listar los empleados de los departamentos 100 y 110, mostrando el número de empleado, el nombre, su salario, el porcentaje de su salario con respecto a la suma de todos los empleados de la lista, y el porcentaje acumulado, ordenado por salario.

Solución:

```
SELECT  NUMEM, NOMEM, SALAR,
        DECIMAL((( E1.SALAR / (SELECT   SUM (E2.SALAR)
                FROM TEMPLE   E2
                        WHERE E2.NUMDE IN (100, 110))
                )*100.00),6,1) AS PORC,
        DECIMAL((((SELECT SUM(EY.SALAR)*1.000 FROM TEMPLE EY WHERE
        EY.NUMDE IN (100,110) AND EY.NUMEM <=E1.NUMEM) / (SELECT
        SUM(SALAR) FROM TEMPLE WHERE NUMDE IN (100,110)))*100.00),6,1) AS
        PORCACU
FROM TEMPLE E1
WHERE NUMDE IN (100, 110)
GROUP BY NUMEM, NOMEM, SALAR
ORDER BY  PORCACU
```

Resultado:

NUMEM	NOMEM	SALAR	PORC	PORCACU
180	PEREZ, MARCOS	4800	19.6	19.6
210	GALVEZ, PILAR	3800	15.5	35.1
250	ALBA, ADRIANA	4500	18.4	53.5
260	LOPEZ, ANTONIO	7200	29.4	83.0
390	MORAN, CARMEN	2150	8.7	91.8
510	CAMPOS, ROMULO	2000	8.1	100.0

Solucionario Capítulo 12

"COMPOSICIÓN DE CONSULTAS"

***12.1).** Hallar el salario medio y la edad media en años de los empleados que tienen comisión y los que no.

Solución:

```
SELECT   'CON COMISION', CURRENT DATE, AVG (SALAR),
         AVG (YEAR (CURRENT DATE - FECNA) )
FROM     TEMPLE
WHERE    COMIS IS NOT NULL

UNION

SELECT   'SIN COMISION', CURRENT DATE, AVG (SALAR),
         AVG (YEAR (CURRENT DATE - FECNA) )
FROM     TEMPLE
WHERE    COMIS IS NULL
```

Resultado (suponiendo que el CURRENT DATE vale 5.7.1990)*:*

COL-1	COL-2	COL-3	COL-4
CON COMISION	1990-07-05	2664,28	36
SIN COMISION	1990-07-05	3285,00	36

12.2). Para los empleados que no tienen comisión obtener por orden alfabético el nombre y el cociente entre su salario y el número de hijos (como el ejercicio 6.3), pero si un empleado no tiene hijos, obtener el salario sin más, indicando este caso con un literal.

Solución:

```
SELECT        NOMEM, SALAR / NUMHI, '(SALARIO / HIJOS)'
FROM          TEMPLE
WHERE                 COMIS IS NULL AND NUMHI > 0

UNION

SELECT        NOMEM, SALAR, '(SALARIO)'
FROM          TEMPLE
WHERE                 COMIS IS NULL AND NUMHI = 0
ORDER BY      1
```

Resultado:

NOMEM	COL-2	COL-3
AGUIRRE, AUREO	4500,00	(SALARIO)
CAMPOS, ROMULO	2000,00	(SALARIO / HIJOS)
CAMPS, AURELIO	4500,00	(SALARIO / HIJOS)
FIERRO, CLAUDIA	4000,00	(SALARIO)
FLOR, DOROTEA	580,00	(SALARIO / HIJOS)
GALVEZ, PILAR	1900,00	(SALARIO / HIJOS)
GARCIA, AUGUSTO	4200,00	(SALARIO)
GIL, GLORIA	900,00	(SALARIO / HIJOS)
LARA, LUCRECIA	1850,00	(SALARIO)
LOPEZ, ANTONIO	1200,00	(SALARIO / HIJOS)
MARTIN, MICAELA	1800,00	(SALARIO)

MORA, VALERIANA	2100,00	(SALARIO / HIJOS)
MORAN, CARMEN	2150,00	(SALARIO / HIJOS)
MUÑOZ, AZUCENA	1750,00	(SALARIO)
PEREZ, JULIO	4400,00	(SALARIO)
POLO, OTILIA	3800,00	(SALARIO)
PONS, CESAR	1033,33	(SALARIO / HIJOS)
RUIZ, FABIOLA	1900,00	(SALARIO / HIJOS)
SANZ, CORNELIO	2025,00	(SALARIO / HIJOS)
VEIGA, JULIANA	750,00	(SALARIO / HIJOS)

12.3). Para los empleados que trabajan en la calle de Atocha y su salario supera al salario medio de su departamento, obtener por orden alfabético su nombre y su salario total (salario, o salario más comisión en los que la tengan).

Solución:

```
SELECT      NOMEM, SALAR + COMIS
FROM        TEMPLE E, TDEPTO D, TCENTR C
WHERE       E.NUMDE = D.NUMDE AND D.NUMCE = C.NUMCE AND
            SEÑAS LIKE '%ATOCHA%' AND
            SALAR > (SELECT AVG (SALAR)
                        FROM   TEMPLE
                        WHERE  NUMDE = E.NUMDE)
            AND COMIS IS NOT NULL

UNION

SELECT      NOMEM, SALAR
FROM        TEMPLE E, TDEPTO D, TCENTR C
WHERE       E.NUMDE = D.NUMDE AND D.NUMCE = C.NUMCE AND
            SEÑAS LIKE '%ATOCHA%' AND
            SALAR > (SELECT AVG (SALAR)
                        FROM   TEMPLE
                        WHERE  NUMDE = E.NUMDE)
            AND COMIS IS NULL

ORDER BY    1
```

Otra formulación usando la función escalar COALESCE:

```
SELECT      NOMEM, COALESCE (SALAR + COMIS, SALAR)
FROM        TEMPLE E, TDEPTO D, TCENTR C
WHERE       E.NUMDE = D.NUMDE AND D.NUMCE = C.NUMCE AND
            SEÑAS LIKE '%ATOCHA%' AND
            SALAR > (SELECT AVG (SALAR)
                        FROM   TEMPLE
                        WHERE  NUMDE = E.NUMDE)
ORDER BY    NOMEM
```

Resultado:

NOMEM	COL-2
AGUIRRE, AUREO	4200
DIEZ, AMELIA	3700
GARCIA, OCTAVIO	4600
LARA, DORINDA	3500
LASA, MARIO	4600
PEREZ, MARCOS	5300
SANZ, LAVINIA	3800
TEROL, LUCIANO	4000

12.4). Hallar por departamento la masa salarial total (suma de todos los salarios y comisiones del departamento) y el nombre, por orden alfabético.

En principio se podría pensar en formularla así:

```
SELECT      NOMDE, SUM (SALAR) + SUM (COMIS)
FROM        TEMPLE E, TDEPTO D
WHERE       D.NUMDE = E.NUMDE
GROUP BY    D.NUMDE, D.NOMDE
ORDER BY    NOMDE
```

Sin embargo esta solución no es correcta para los departamentos en que ningún empleado trabaje a comisión. Para corregir este defecto se podría reformular así:

```
SELECT      NOMDE, SUM (SALAR) + SUM (COMIS)
FROM        TEMPLE E, TDEPTO D
WHERE       D.NUMDE = E.NUMDE
GROUP BY    D.NUMDE, D.NOMDE
HAVING      COUNT (DISTINCT COMIS) > 0

UNION

SELECT      NOMDE, SUM (SALAR)
FROM        TEMPLE E, TDEPTO D
WHERE       D.NUMDE = E.NUMDE
GROUP BY    D.NUMDE, D.NOMDE
HAVING      COUNT (DISTINCT COMIS) = 0

ORDER BY    1
```

Se podría también reformular con la ayuda de la función COALESCE:

```
SELECT      NOMDE,
             COALESCE (SUM (SALAR) + SUM (COMIS), SUM (SALAR))
FROM        TEMPLE E, TDEPTO D
WHERE       D.NUMDE = E.NUMDE
GROUP BY    NOMDE
ORDER BY    NOMDE
```

O también:

```
SELECT      NOMDE,
            SUM (COALESCE (SALAR + COMIS, SALAR))
FROM        TEMPLE E, TDEPTO D
WHERE       D.NUMDE = E.NUMDE
GROUP BY    NOMDE
ORDER BY    NOMDE
```

Resultado:

NOMDE	COL-2
DIRECCION COMERCIAL	9450
DIRECCION GENERAL	15500
FINANZAS	11100
ORGANIZACIÓN	2700
PERSONAL	12400
PROCESO DE DATOS	16200
SECTOR INDUSTRIAL	24750
SECTOR SERVICIOS	24600

12.5). Efectuar una explosión de la organización de departamentos. Es decir, para cada departamento obtener su nombre, el de los que dependen de él y el nivel al que dependen. Si un departamento depende directamente de otro este nivel será 1, si depende de uno que depende directamente de éste será 2, y así sucesivamente. Se considerará que un departamento depende de sí mismo a nivel 0. La primera columna del resultado será el nombre de un departamento, la segunda el de un departamento que depende de él, y la tercera el nivel al que depende. Considerar un máximo de 4 niveles de dependencia. Presentar el resultado por orden alfabético. Si de un departamento no depende ningún otro, aparecerá al menos dependiendo de sí mismo a nivel 0.

Solución:

```
SELECT      NOMDE, NOMDE, 0
FROM        TDEPTO

UNION

SELECT      D0.NOMDE, D1.NOMDE, 1
FROM        TDEPTO D0, TDEPTO D1
            WHERE D0.NUMDE = D1.DEPDE

UNION

SELECT      D0.NOMDE, D2.NOMDE, 2
FROM        TDEPTO D0, TDEPTO D1, TDEPTO D2
            WHERE D0.NUMDE = D1.DEPDE AND D1.NUMDE = D2.DEPDE

UNION

SELECT      D0.NOMDE, D3.NOMDE, 3
FROM        TDEPTO D0, TDEPTO D1, TDEPTO D2,
            TDEPTO D3
WHERE       D0.NUMDE = D1.DEPDE AND D1.NUMDE = D2.DEPDE AND
            D2.NUMDE = D3.DEPDE

UNION

SELECT      D0.NOMDE, D4.NOMDE, 4
FROM        TDEPTO D0, TDEPTO D1, TDEPTO D2,
            TDEPTO D3, TDEPTO D4
WHERE       D0.NUMDE = D1.DEPDE AND D1.NUMDE = D2.DEPDE AND
            D2.NUMDE = D3.DEPDE AND D3.NUMDE = D4.DEPDE

ORDER BY    1, 3, 2
```

Resultado:

NOMDE	NOMDE1	COL-3
DIRECCION COMERCIAL	DIRECCION COMERCIAL	0
DIRECCION COMERCIAL	SECTOR INDUSTRIAL	1
DIRECCION COMERCIAL	SECTOR SERVICIOS	1
DIRECCION GENERAL	DIRECCION GENERAL	0
DIRECCION GENERAL	DIRECCION COMERCIAL	1
DIRECCION GENERAL	FINANZAS	1
DIRECCION GENERAL	ORGANIZACION	1
DIRECCION GENERAL	PERSONAL	2
DIRECCION GENERAL	PROCESO DE DATOS	2
DIRECCION GENERAL	SECTOR INDUSTRIAL	2
DIRECCION GENERAL	SECTOR SERVICIOS	2
DIRECCION GENERAL	PERSONAL CONTRATADO	3
FINANZAS	FINANZAS	0

ORGANIZACION	ORGANIZACION	0
ORGANIZACION	PERSONAL	1
ORGANIZACION	PROCESO DE DATOS	1
ORGANIZACION	PERSONAL CONTRATADO	2
PERSONAL	PERSONAL	0
PERSONAL	PERSONAL CONTRATADO	1
PERSONAL CONTRATADO	PERSONAL CONTRATADO	0
PROCESO DE DATOS	PROCESO DE DATOS	0
SECTOR INDUSTRIAL	SECTOR INDUSTRIAL	0
SECTOR SERVICIOS	SECTOR SERVICIOS	0

12.6). Como el ejercicio anterior, pero añadiendo al resultado una columna con el nombre del director del departamento de la segunda columna y otra con la masa salarial total de ese departamento (salarios más comisiones).

Se puede formular usando la función escalar COALESCE o no, análogamente a como se ha hecho en ejercicios anteriores. Mostramos aquí la solución con COALESCE por ser de formulación más breve:

```
SELECT      D0.NOMDE, D0.NOMDE, 0, ED.NOMEM,
            SUM (COALESCE (EM.SALAR + EM.COMIS, EM.SALAR))
FROM        TDEPTO D0, TEMPLE ED, TEMPLE EM
WHERE       D0.DIREC = ED.NUMEM AND D0.NUMDE = EM.NUMDE
GROUP BY    D0.NUMDE, D0.NOMDE, ED.NOMEM

UNION

SELECT      D0.NOMDE, D1.NOMDE, 1, ED.NOMEM,
            SUM (COALESCE (EM.SALAR + EM.COMIS, EM.SALAR))
FROM        TDEPTO D0, TDEPTO D1,
            TEMPLE ED, TEMPLE EM
WHERE       D0.NUMDE = D1.DEPDE AND
            D1.DIREC = ED.NUMEM AND D1.NUMDE = EM.NUMDE
GROUP BY    D0.NUMDE, D0.NOMDE, D1.NUMDE, D1.NOMDE, ED.NOMEM

UNION

SELECT      D0.NOMDE, D2.NOMDE, 2, ED.NOMEM,
            SUM (COALESCE (EM.SALAR + EM.COMIS, EM.SALAR))
FROM        TDEPTO D0, TDEPTO D1, TDEPTO D2,
            TEMPLE ED, TEMPLE EM
WHERE       D0.NUMDE = D1.DEPDE AND D1.NUMDE = D2.DEPDE AND
            D2.DIREC = ED.NUMEM AND D2.NUMDE = EM.NUMDE
GROUP BY    D0.NUMDE, D0.NOMDE, D1.NUMDE, D2.NUMDE,
            D2.NOMDE, ED.NOMEM

UNION

SELECT      D0.NOMDE, D3.NOMDE, 3, ED.NOMEM,
            SUM (COALESCE (EM.SALAR + EM.COMIS, EM.SALAR))
FROM        TDEPTO D0, TDEPTO D1, TDEPTO D2, TDEPTO D3,
             TEMPLE ED, TEMPLE EM
WHERE       D0.NUMDE = D1.DEPDE AND D1.NUMDE = D2.DEPDE AND
            D2.NUMDE = D3.DEPDE AND D3.DIREC = ED.NUMEM AND
             D3.NUMDE = EM.NUMDE
```

```
GROUP BY        D0.NUMDE, D0.NOMDE, D1.NUMDE, D2.NUMDE, D3.NUMDE,
                D3.NOMDE, ED.NOMEM

                UNION

SELECT      D0.NOMDE, D4.NOMDE, 4, ED.NOMEM,
            SUM (COALESCE (EM.SALAR + EM.COMIS, EM.SALAR))
FROM        TDEPTO D0, TDEPTO D1, TDEPTO D2, TDEPTO D3, TDEPTO D4,
            TEMPLE ED, TEMPLE EM
WHERE       D0.NUMDE = D1.DEPDE AND D1.NUMDE = D2.DEPDE AND
            D2.NUMDE = D3.DEPDE AND D3.NUMDE = D4.DEPDE AND
            D4.DIREC = ED.NUMEM AND D4.NUMDE = EM.NUMDE
GROUP BY    D0.NUMDE, D0.NOMDE, D1.NUMDE, D2.NUMDE, D3.NUMDE,
            D4.NUMDE, D4.NOMDE, ED.NOMEM
ORDER BY    1, 3, 2
```

Resultado:

NOMDE	NOMDE1	COL-3	NOMEM	COL-5
DIRECCION COMERCIAL	DIRECCION COMERCIAL	0	PEREZ, MARCOS	9450
DIRECCION COMERCIAL	SECTOR INDUSTRIAL	1	PEREZ, MARCOS	24750
DIRECCION COMERCIAL	SECTOR SERVICIOS	1	GARCIA, OCTAVIO	24600
DIRECCION GENERAL	DIRECCION GENERAL	0	LOPEZ, ANTONIO	15500
DIRECCION GENERAL	DIRECCION COMERCIAL	1	PEREZ, MARCOS	9450
DIRECCION GENERAL	FINANZAS	1	GARCIA, AUGUSTO	11100
DIRECCION GENERAL	ORGANIZACIÓN	1	PEREZ, JULIO	2700
DIRECCION GENERAL	PERSONAL	2	PEREZ, JULIO	12400
DIRECCION GENERAL	PROCESO DE DATOS	2	CAMPS, AURELIO	16200
DIRECCION GENERAL	SECTOR INDUSTRIAL	2	PEREZ, MARCOS	24750
DIRECCION GENERAL	SECTOR SERVICIOS	2	GARCIA, OCTAVIO	24600
FINANZAS	FINANZAS	0	GARCIA, AUGUSTO	11100
ORGANIZACION	ORGANIZACIÓN	0	PEREZ, JULIO	2700
ORGANIZACIÓN	PERSONAL	1	PEREZ, JULIO	12400
ORGANIZACIÓN	PROCESO DE DATOS	1	CAMPS, AURELIO	16200
PERSONAL	PERSONAL	0	PEREZ, JULIO	12400
PROCESO DE DATOS	PROCESO DE DATOS	0	CAMPS, AURELIO	16200
SECTOR INDUSTRIAL	SECTOR INDUSTRIAL	0	PEREZ, MARCOS	24750
SECTOR SERVICIOS	SECTOR SERVICIOS	0	GARCIA, OCTAVIO	24600

Al aparecer la tabla de Empleados en la yunción, desaparecen 4 filas respecto al ejercicio anterior; corresponden a aquéllas en las que figura el departamento de PERSONAL CONTRATADO, y que no tiene empleados.

12.7). Supongamos que algunos departamentos se van a trasladar a otro local. Disponemos de una tabla llamada TTRASL con una sola columna llamada NUMDE donde hay una fila por cada departamento que se traslada al local nuevo. Se desea producir una lista por orden alfabético de todos los departamentos, indicando cuáles se trasladan y cuáles no. Este tipo de operaciones en que se combinan filas de una tabla con las de otra, incluyendo en el resultado también las que no están emparejadas, se conoce en teoría de Bases de Datos relacionales como yunción externa ("outer join"), que se tratará en un capítulo posterior, pero aquí pedimos formularla utilizando el predicado EXISTS y la cláusula UNION. En la tabla TTRASL hemos dado de alta los departamentos 100 y 110.

Solución:

```
SELECT D.*, 'SE TRASLADA'
FROM   TDEPTO D, TTRASL T
 WHERE  D.NUMDE = T.NUMDE
```

```
UNION

SELECT D.*, 'NO SE TRASLADA'
FROM   TDEPTO D
WHERE  NOT EXISTS (SELECT *
                   FROM   TTRASL T
                   WHERE  D.NUMDE = T.NUMDE)

ORDER BY 7
```

Resultado:

NUMDE	NUMCE	DIREC	TIDIR	PRESU	DEPDE	NOMDE	COL-8
110	20	180	P	150	100	DIRECCION COMERCIAL	SE TRASLADA
100	10	260	P	120	-	DIRECCION GENERAL	SE TRASLADA
130	10	310	P	20	100	FINANZAS	NO SE TRASLADA
120	10	150	F	30	100	ORGANIZACION	NO SE TRASLADA
121	10	150	P	20	120	PERSONAL	NO SE TRASLADA
123	-	150	F	100	121	PERSONAL CONTRATADO	NO SE TRASLADA
122	10	350	P	60	120	PROCESO DE DATOS	NO SE TRASLADA
111	20	180	F	110	110	SECTOR INDUSTRIAL	NO SE TRASLADA
112	20	270	P	90	110	SECTOR SERVICIOS	NO SE TRASLADA

Solucionario Capítulo 13

"EXPRESIONES DE TABLA ANIDADA"

13.1). Sin hacer uso de la cláusula de agrupamiento de filas se desea obtener una lista de los empleados del departamento 121, su salario y el mayor salario existente dentro del conjunto completo de los empleados.

Solución:

```
SELECT NUMEM, SALAR,
            (SELECT MAX(SALAR) FROM TEMPLE) AS MAXSALAR
FROM TEMPLE
WHERE NUMDE = 121
```

ó

```
SELECT NUMEM, SALAR, MAXSALAR
FROM TEMPLE,
          (SELECT MAX(SALAR) AS MAXSALAR FROM TEMPLE) AS XXX
WHERE NUMDE = 121
```

Resultado:

NUMEM	SALAR	MAXSALAR
110	3100	7200
150	4400	7200
190	3000	7200
370	1900	7200

13.2). Lo mismo que el anterior pero especificando el mayor salario del departamento.

Solución:

```
SELECT NUMEM, SALAR,
            (SELECT MAX(B.SALAR) FROM TEMPLE B
              WHERE B.NUMDE=A.NUMDE) AS MAXSALAR
FROM TEMPLE A
WHERE NUMDE = 121;
```

ó

```
SELECT NUMEM, SALAR, MAXSALAR
FROM TEMPLE A, TABLE (SELECT MAX(B.SALAR) AS MAXSALAR
                                FROM TEMPLE B
                                WHERE B.NUMDE=A.NUMDE) AS XXX
WHERE NUMDE = 121;
```

Resultado:

NUMEM	SALAR	MAXSALAR
110	3100	4400
150	4400	4400
190	3000	4400
370	1900	4400

13.3). Hallar los departamentos cuyo presupuesto supera al presupuesto medio de los departamentos que dependen del mismo del que depende él, incluido él mismo.

```
SELECT D1.NUMDE, D1.PRESU, PREMED
FROM TDEPTO D1,
        TABLE (SELECT AVG (PRESU) AS PREMED
                FROM TDEPTO D2
                WHERE D2.DEPDE = D1.DEPDE
                ) AS D3
WHERE D1.PRESU > PREMED
ORDER BY D1.NUMDE
```

Resultado:

NUMDE	PRESU	PREMED
110	150	66,66
111	110	100,00
122	60	40,00

Formulación equivalente:

```
SELECT D1.NUMDE, D1.PRESU, AVG (D2.PRESU) AS PREMED
FROM TDEPTO D1, TDEPTO D2
WHERE D1.DEPDE = D2.DEPDE
GROUP BY D1.NUMDE, D1.PRESU
HAVING D1.PRESU > AVG (D2.PRESU)
ORDER BY D1.NUMDE
```

Solucionario Capítulo 14

"EXPRESIONES CONDICIONALES ("CASE")"

14.1). Obtener con el uso de expresiones CASE el número de empleados nacidos antes del 67 y los que ingresaron en la compañía después del 84.

Solución:

```
SELECT  SUM (CASE WHEN FECNA < '01.01.1967' THEN 1 ELSE 0 END),
        SUM (CASE WHEN FECIN > '31.12.1984' THEN 1 ELSE 0 END)
FROM TEMPLE
```

Resultado:

```
COL-1   COL-2
--------  ---------
    28         14
```

14.2). Obtener cuántos empleados con fecha de nacimiento superior al 1 de enero de 1980; incluir en la consulta la media de los salarios. Buscar dos columnas adicionales (usando funciones o expresiones conocidas) a incluir en la consulta que permitan transformar los resultados nulos de la media de salarios en valor cero.

Solución:

```
SELECT COUNT (*) AS C1, AVG (SALAR) AS A1,
          COALESCE ( AVG (SALAR), 0 ) AS A2,
           CASE WHEN AVG (SALAR) IS NULL THEN 0 ELSE AVG (SALAR)
         END AS A3
FROM TEMPLE
WHERE FECNA > '1.1.1980'
```

Resultado:

```
C1       A1       A2       A3
----     ----     ----     ----
  0       -         0        0
```

14.3). Obtener por grupos de empleados, los de los números de empleado del 100 al 199, los del 200 al 299, y el resto, los máximos salarios, los mínimos y las medias.

Solución:

```
SELECT CASE_NUMEM, MAX (SALAR) AS MAXS, MIN (SALAR) AS MINS,
          AVG (SALAR) AS AVGS
FROM   (SELECT SALAR,
           CASE WHEN NUMEM BETWEEN 100 AND 199 THEN 'LOS 100S'
                WHEN NUMEM BETWEEN 200 AND 299 THEN 'LOS 200S'
           END  AS CASE_NUMEM
        FROM TEMPLE) AS X
GROUP BY CASE_NUMEM
```

Resultado:

```
CASE_NUMEM MAXS  MINS    AVGS
------------------ --------- -------  ----------
LOS 100S           4800  2900   3542,85
LOS 200S           7200  2700   3937,50
  -                4500  1000   2457,89
```

14.4). Obtener los números de departamento de aquellos con presupuestos superiores a 90.000 € mostrando el presupuesto inicial y el nuevo teniendo en cuenta que se incrementa en un 10% para los que tienen dependencia de DIRECCIÓN GENERAL, y el 5% al resto. Además se sube un 3% adicional si tiene director EN PROPIEDAD.

Solución:

```
SELECT NUMDE, PRESU,
       PRESU + PRESU*((CASE WHEN DEPDE = 100 THEN 0.10
                            ELSE 0.05 END) +
                      (CASE WHEN TIDIR = 'P' THEN 0.03
                            ELSE 0.00 END)
                     )
                     AS NUEVOPRESU
FROM TDEPTO
WHERE PRESU > 90
```

Resultado:

NUMDE	PRESU	NUEVOPRESU
100	120	129,60
110	150	169,50
111	110	115,50
123	100	105,00

14.5). Transformar la sentencia SQL anterior en otra que haga uso de la cláusula UNION.

Solución:

```
    SELECT NUMDE, PRESU, PRESU*1.13 AS NUEVOPRESU
    FROM TDEPTO
    WHERE DEPDE = 100 AND TIDIR = 'P' AND PRESU > 90
UNION ALL
    SELECT NUMDE, PRESU, PRESU*1.10 AS NUEVOPRESU
    FROM TDEPTO
    WHERE DEPDE = 100 AND TIDIR = 'F' AND PRESU > 90
UNION ALL
    SELECT NUMDE, PRESU, PRESU*1.08 AS NUEVOPRESU
    FROM TDEPTO
    WHERE (DEPDE <> 100 OR DEPDE IS NULL) AND TIDIR = 'P' AND PRESU > 90
UNION ALL
    SELECT NUMDE, PRESU, PRESU*1.05 AS NUEVOPRESU
    FROM TDEPTO
    WHERE (DEPDE <> 100 OR DEPDE IS NULL) AND TIDIR = 'F' AND PRESU > 90
```

14.6). Obtener los números de empleado, sus departamentos y la suma de salarios y comisiones, para aquellos cuyo salario sea mayor de 4 veces lo que ingresa por comisiones y que no estén pagados sólo en salario.

Solución:

```
SELECT NUMEM, NUMDE, (SALAR + COMIS) AS "SALARIO+COMISIONES"
FROM TEMPLE
WHERE (CASE WHEN COMIS IS NULL THEN NULL
            WHEN COMIS = 0 THEN NULL
            ELSE SALAR / COMIS
       END) > 4
ORDER BY NUMEM
```

Resultado:

NUMEM	NUMDE	SALARIO+COMISIONES
180	110	5300
270	112	4600

14.7). Formar tres grupos con los departamentos. En uno se incluirán los departamentos 110, 120 y 130, y lo llamaremos "GRUPO A". En otro los departamentos 111, 112, 121 y 122, y será el "GRUPO B". El resto se incluye en un grupo llamado "OTROS". Para cada grupo hallar el presupuesto máximo, el mínimo y el medio.

Solución:

```
SELECT GRUPOS,
        MAX (PRESU) AS MAX, MIN (PRESU) AS MIN, AVG (PRESU) AS MEDIA
FROM (SELECT PRESU, CASE
                        WHEN NUMDE IN (110,120,130) THEN 'GRUPO A'
                       WHEN NUMDE IN (111,112,121,122)  THEN 'GRUPO B'
                       ELSE 'OTROS'
                    END  AS GRUPOS
      FROM TDEPTO) AS X
GROUP BY GRUPOS
ORDER BY GRUPOS
```

Resultado:

GRUPOS	MAX	MIN	MEDIA
GRUPO A	150	20	66,66
GRUPO B	110	20	70,00
OTROS	120	120	110,00

Solucionario Capítulo 15

"OTRA MANERA DE ESPECIFICAR LAS YUNCIONES ("JOINS")"

15.1). Obtener la lista de los departamentos que no tienen empleados asignados. Mostrar el número y nombre.

Solución:

```
    SELECT D.NUMDE AS DEPTO, NOMDE
    FROM  TDEPTO D LEFT OUTER JOIN TEMPLE E
                ON D.NUMDE = E.NUMDE
EXCEPT
    SELECT D.NUMDE AS DEPTO, NOMDE
    FROM  TDEPTO D INNER JOIN TEMPLE E
                ON D.NUMDE = E.NUMDE
ORDER  BY DEPTO
```

Resultado:

```
DEPTO        NOMDE
-----------  ------------------------------------
      123    PERSONAL CONTRATADO
```

O también:

```
SELECT NUMDE AS DEPTO, NOMDE
FROM TDEPTO
WHERE NOT EXISTS ( SELECT
                        FROM TEMPLE
                        WHERE TEMPLE.NUMDE = TDEPTO.NUMDE)
ORDER BY DEPTO
```

15.2). Hallar cuántos empleados hay que no tengan departamento asignado.

Solución:

```
SELECT COUNT (*) AS EMPLES
FROM (SELECT NUMEM
        FROM  TEMPLE E LEFT OUTER JOIN TDEPTO D
                ON E.NUMDE = D.NUMDE
      EXCEPT
        SELECT NUMEM
        FROM TEMPLE E INNER JOIN TDEPTO D
                ON E.NUMDE = D.NUMDE
        )  AS  TD
```

Resultado:

```
        EMPLES
        -----------
                0
```

O también:

```
SELECT COUNT (*) AS EMPLES
FROM (SELECT NUMEM
        FROM TEMPLE E
        WHERE NOT EXISTS ( SELECT NUMDE
                                FROM  TDEPTO D
                                WHERE D.NUMDE = E.NUMDE)
        ) AS T
```


15.3). Para cada centro, obtener la suma de los salarios de los empleados que trabajan en él. Los centros que no tengan empleados se mostrarán con una suma cero.

Solución:

```
SELECT  C.NUMCE AS CENTRO, COALESCE (SUM (SALAR), 0)  AS SUMA
FROM  (TCENTR C LEFT OUTER JOIN  TDEPTO D
                        ON C.NUMCE = D.NUMCE )
        LEFT OUTER JOIN TEMPLE E
                        ON D.NUMDE = E.NUMDE
GROUP BY C.NUMCE
ORDER BY CENTRO
```

Resultado:

CENTRO	SUMA
10	57900
20	45100
50	0

15.4). Hallar los departamentos que tienen un presupuesto mayor que el del departamento del cual dependen. Incluir también los departamentos que no dependan de otro. Mostrar número de departamento, número de departamento del que dependen y presupuestos de ambos.

Solución:

```
SELECT D1.NUMDE AS DEPTO, D1.DEPDE AS DEPEND, D1.PRESU AS PRESU1,
D2.PRESU AS PRESU2
FROM  TDEPTO D1 LEFT OUTER JOIN TDEPTO D2
ON D1.DEPDE = D2.NUMDE
WHERE (D1.PRESU>D2.PRESU OR D2.PRESU IS NULL)
ORDER BY DEPTO
```

Resultado:

DEPTO	DEPEND	PRESU1	PRESU2
100	-	120	-
110	100	150	120
122	120	60	30
123	121	100	120

Con la yunción externa se obtienen todos los emparejamientos de dependencias incluida la del departamento 100 que no depende de nadie. Mediante la cláusula WHERE se filtran, por un lado aquellas dependencias donde el presupuesto del departamento primero es mayor del del departamento del que dependen, y por otro mantenemos el departamento 100 porque el presupuesto del departamento del que depende (ninguno) es nulo.

15.5). Para los empleados de los departamentos 100 y 110, obtener un listado de ellos que incluya: nombre y salario medio de los que cobran más que él, sea cual sea su departamento. Mostrar también los que no tengan a nadie con mayor salario que el suyo.

Solución:

```
SELECT T1.NOMEM AS NOMBRE, AVG (E2.SALAR) AS SAL_MEDIO
FROM  (SELECT * FROM TEMPLE WHERE NUMDE IN (100,110)) AS T1
            LEFT OUTER JOIN TEMPLE E2
            ON T1.SALAR < E2.SALAR
GROUP BY T1.NOMEM
ORDER BY NOMBRE
```

Resultado :

NOMEM	SAL_MEDIO
ALBA, ADRIANA	6000,00
CAMPOS, ROMULO	3419,23
GALVEZ, PILAR	4706,25
LOPEZ, ANTONIO	-
MORAN, CARMEN	3730,95
PEREZ, MARCOS	7200,00

15.6). Considerando solo los empleados de los departamentos 100 y 110, obtener un listado de ellos que incluya: nombre y salario medio de los que cobran más que él. Mostrar también los que no tengan a nadie con mayor salario que el suyo.

Solución:

```
SELECT T1.NOMEM AS NOMBRE, AVG (T2.SALAR) AS SAL_MEDIO
FROM  (SELECT * FROM TEMPLE WHERE NUMDE IN (100,110)) AS T1
 LEFT OUTER JOIN
          (SELECT * FROM TEMPLE WHERE NUMDE IN (100,110)) AS T2
 ON T1.SALAR < T2.SALAR
GROUP BY T1.NOMEM
ORDER BY NOMBRE
```

Resultado:

NOMEM	SAL_MEDIO
ALBA, ADRIANA	6000,00
CAMPOS, ROMULO	4490,00
GALVEZ, PILAR	5500,00
LOPEZ, ANTONIO	-
MORAN, CARMEN	5075,00
PEREZ, MARCOS	7200,00

15.7). Considerando solo los empleados de los departamentos 100 y 110, obtener un listado de ellos que incluya: nombre, comisión, y cuántos empleados de estos departamentos tienen una comisión mayor que la suya. Este valor se mostrará como un *Nulo* en los que no tengan comisión.

Solución:

```
SELECT T1.NOMEM AS NOMBRE, T1.COMIS AS COMISION,
          CASE
             WHEN T1.COMIS IS NOT NULL AND MAX (T2.COMIS) IS NULL
                       THEN 0
             WHEN T1.COMIS IS NOT NULL THEN COUNT (*)
                                ELSE  T1.COMIS
          END AS EMPLES
FROM  (SELECT * FROM TEMPLE WHERE NUMDE IN (100,110)) AS T1
LEFT OUTER JOIN
         (SELECT * FROM TEMPLE WHERE NUMDE IN (100,110)) AS T2
ON T1.COMIS < T2.COMIS
GROUP BY T1.NOMEM, T1.COMIS
ORDER BY NOMBRE
```

Resultado:

NOMBRE	COMISION	EMPLES
ALBA, ADRIANA	-	-
CAMPOS, ROMULO	-	-
GALVEZ, PILAR	-	-
LOPEZ, ANTONIO	-	-
MORAN, CARMEN	-	-
PEREZ, MARCOS	500	0

15.8). Contestar al ejercicio 12.7 empleando la yunción externa. Recordemos su enunciado:
Supongamos que algunos departamentos se van a trasladar a otro local. Disponemos de una tabla llamada TTRASL con una sola columna llamada NUMDE donde hay una fila por cada departamento que se traslada al local nuevo. Se desea producir una lista por orden alfabético de todos los departamentos, indicando cuáles se trasladan y cuáles no. En la tabla TTRASL hemos dado de alta los departamentos 100 y 110.

Solución:

```
SELECT  D.NOMDE,
        CASE
            WHEN T.NUMDE IS NULL THEN 'NO SE TRASLADA'
             ELSE 'SE TRASLADA'
        END AS TRASLADO
FROM  TDEPTO D LEFT OUTER JOIN TTRASL T
        ON D.NUMDE = T.NUMDE
ORDER BY NOMDE
```

Resultado:

NOMDE	TRASLADO
DIRECCION COMERCIAL	SE TRASLADA
DIRECCION GENERAL	SE TRASLADA
FINANZAS	NO SE TRASLADA
ORGANIZACION	NO SE TRASLADA
PERSONAL	NO SE TRASLADA
PERSONAL CONTRATADO	NO SE TRASLADA
PROCESO DE DATOS	NO SE TRASLADA
SECTOR INDUSTRIAL	NO SE TRASLADA
SECTOR SERVICIOS	NO SE TRASLADA

15.9). Obtener el conjunto de empleados con su número, departamento al que pertenecen y director del departamento aunque no exista el número del departamento. Finalmente se seleccionarán sólo las filas donde el departamento del empleado sea el 110, y donde los empleados tengan un salario inferior a los 3.000 euros.

Solución:

```
SELECT NUMEM, A.NUMDE, DIREC
FROM TEMPLE A LEFT OUTER JOIN TDEPTO B
        ON A.NUMDE=B.NUMDE
WHERE A.NUMDE = 110 AND SALAR < 3000
```

Resultado:

NUMEM	NUMDE	DIREC
390	110	180
510	110	180

Solucionario Capítulo 16

"EXPRESIONES DE TABLA COMÚN Y RECURSIVIDAD"

16.1). Obtener el valor mínimo de las medias de salario de los empleados por departamento.

Solución:

```
WITH TEMP1 AS
            (SELECT NUMDE,
             AVG (SALAR) AS AVGSALAR
             FROM TEMPLE
            GROUP BY NUMDE),
      TEMP2 AS
            (SELECT MIN (AVGSALAR) AS MINSALAR
            FROM TEMP1)

SELECT * FROM TEMP2
```

Resultado:

```
MINSALAR
----------
   2181.25
```

16.2). Obtener el conjunto de departamentos cuyos empleados tengan una media de número de hijos inferior a la media de todos los empleados.

Solución:

```
WITH TEMP1 AS
                (SELECT AVG (NUMHI) AS MEDIAHID, NUMDE AS #NUMDE
                 FROM TEMPLE
                 GROUP BY NUMDE),
      TEMP2 (MEDIAHI) AS
                (SELECT AVG (NUMHI)
                 FROM TEMPLE)

SELECT NUMDE, NOMDE FROM TDEPTO, TEMP1, TEMP2
WHERE NUMDE = #NUMDE AND MEDIAHID < MEDIAHI
```

Resultado:

```
NUMDE       NOMDE
----------  ------------------------------
   112      SECTOR SERVICIOS
   122      PROCESO DE DATOS
```

16.3). Reformular la sentencia SQL anterior haciendo uso de la subselección en cláusula FROM.

Solución:

```
SELECT NUMDE, NOMDE
FROM TDEPTO, (SELECT AVG (NUMHI) AS MEDIAHID, NUMDE AS #NUMDE
                    FROM TEMPLE
                    GROUP BY NUMDE) AS T1,
                    (SELECT AVG (NUMHI) AS MEDIAHI FROM TEMPLE) AS T2
WHERE NUMDE = #NUMDE AND MEDIAHID < MEDIAHI
```

16.4). Obtener para un número de hijos entre 0 y 9, cuántos empleados hay.

Solución:

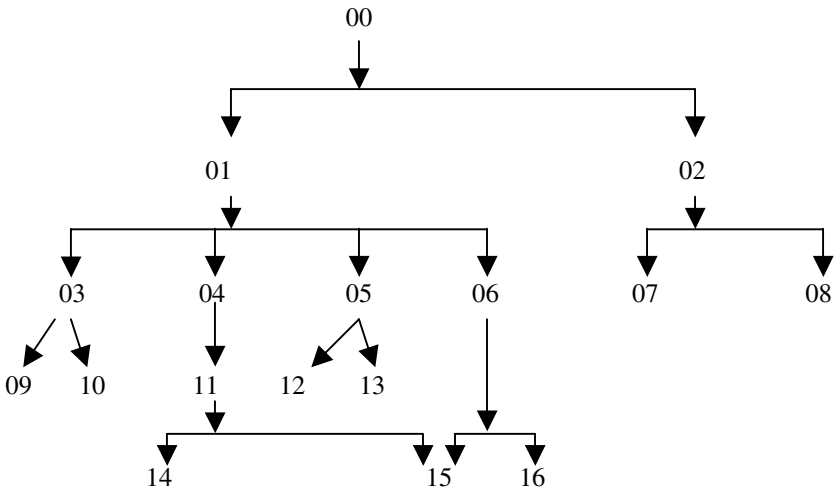
```
WITH NUM_HIJOS (NHIJOS) AS
        (VALUES (0),(1),(2),(3),(4),(5), (6), (7), (8), (9))

SELECT NHIJOS, COALESCE (NUMEMPS,0) AS NUMEMPS
FROM NUM_HIJOS LEFT OUTER JOIN (SELECT NUMHI, COUNT(*) AS NUMEMPS
                                        FROM TEMPLE
                                        WHERE NUMHI < 10
                                        GROUP BY NUMHI) AS XXX
                ON NHIJOS = NUMHI
ORDER BY 1
```

Resultado:

NHIJOS	NUMEMPS
0	14
1	7
2	6
3	4
4	1
5	1
6	1
7	0
8	0
9	0

16.5). Mostrar una tabla con el conjunto de filas que reflejen los elementos de la jerarquía del siguiente gráfico. Se trata de un diagrama de piezas y subpiezas que las forman.



Solución:

PIEZA	SUBPIEZA
00	01
00	02
01	03
01	04
01	05

01	06
02	07
02	08
03	09
03	10
04	11
05	12
05	13
11	14
11	15
06	15
06	16

16.6). Obtener el listado de las piezas dependientes de la pieza 04.

Solución:

```
WITH TABLA1 (PIEZA, SUBPIEZA) AS
        (SELECT PIEZA, SUBPIEZA
        FROM PIEZAS
        WHERE PIEZA = '04'

        UNION ALL

        SELECT C.PIEZA, C.SUBPIEZA
        FROM PIEZAS C,  TABLA1 P
        WHERE P.SUBPIEZA = C.PIEZA
        )

SELECT PIEZA, SUBPIEZA
FROM TABLA1
```

Resultado:

PIEZA	SUBPIEZA
04	11
11	14
11	15

16.7). ¿Cuántas piezas distintas hay dependientes de la 00?.

Solución:

```
WITH TABLA1 (PIEZA, SUBPIEZA) AS
        (SELECT PIEZA, SUBPIEZA
        FROM PIEZAS
        WHERE PIEZA = '00'

        UNION ALL

        SELECT C.PIEZA, C.SUBPIEZA
        FROM PIEZAS C, TABLA1 P
        WHERE P.SUBPIEZA = C.PIEZA
        )

SELECT COUNT (DISTINCT SUBPIEZA)
FROM PIEZAS
```

Resultado:

```
COL-1
---------
      16
```

16.8). ¿Qué subpieza forma parte de más de una pieza?.

Solución:

```
WITH TABLA2 (SUBPIEZA) AS
        (SELECT SUBPIEZA
        FROM PIEZAS
        WHERE PIEZA = '00'

        UNION ALL

         SELECT C.SUBPIEZA
        FROM PIEZAS C, TABLA2 P
        WHERE P.SUBPIEZA = C.PIEZA
         )

SELECT SUBPIEZA
FROM TABLA2
GROUP BY SUBPIEZA
HAVING COUNT(*) > 1
```

Resultado:

```
SUBPIEZA
---------------
        15
```

16.9). ¿Cuáles son las subpiezas de último nivel?.

Solución:

```
WITH TABLA3  (SUBPIEZA, NIVEL)  AS
        (SELECT SUBPIEZA, 1
         FROM PIEZAS
        WHERE PIEZA = '00'

        UNION ALL

        SELECT C.SUBPIEZA, P.NIVEL+1
        FROM PIEZAS  C, TABLA3  P
        WHERE P.SUBPIEZA = C.PIEZA
        )

SELECT SUBPIEZA, NIVEL
FROM TABLA3
WHERE NIVEL = (SELECT MAX (NIVEL) FROM TABLA3)
```

Resultado:

```
SUBPIEZA        NIVEL
----------------    ---------
        14          4
        15          4
```

16.10). Mostrar las piezas de nivel 2 con las piezas de las que dependen.

Solución:

```
WITH TABLA4 (PIEZA, SUBPIEZA, NIVEL) AS
        (SELECT PIEZA, SUBPIEZA, 1
         FROM PIEZAS
         WHERE PIEZA = '00'

         UNION ALL

         SELECT C.PIEZA, C.SUBPIEZA, P.NIVEL+1
         FROM PIEZAS C, TABLA4 P
         WHERE P.SUBPIEZA = C.PIEZA
                 AND P.NIVEL+1 < 3
         )

SELECT DISTINCT SUBPIEZA, PIEZA
FROM TABLA4
WHERE NIVEL = 2
```

Resultado:

SUBPIEZA	PIEZA
03	01
04	01
05	01
06	01
07	02
08	02

16.11). Supongamos que hay dos tablas, llamadas TCURSO y TCURRI. La primera tiene una sola columna llamada NOMCU y la segunda tiene dos, llamadas NOMEM y NOMCU. La tabla TCURSO tiene una fila por cada curso de idiomas impartido a los empleados de la empresa, con el nombre del curso correspondiente en la columna NOMCU. La TCURRI tiene una fila por cada empleado y curso, con el nombre del empleado en la columna NOMEM y el de curso en la NOMCU. El significado de una fila de TCURRI (cuyo nombre viene de curriculum) es que el empleado ha asistido al curso correspondiente.

Se desea hallar una relación de los nombres de los empleados que han asistido a todos los cursos impartidos. En el ej. 11.32 se resolvió esta consulta mediante sentencias correlacionadas. Resolverla ahora mediante expresiones de tabla común

Este tipo de consultas en las que se buscan los valores de una tabla que están combinados con todos los de otra se llama DIVISION en Teoría de Bases de Datos Relacionales.

Solución:

```
WITH T1 (NOMEM) AS
                (SELECT DISTINCT NOMEM
                FROM TCURRI),
        T2 (NOMEM, NOMCU) AS
                 (SELECT * FROM T1, TCURSO
                  EXCEPT
                  SELECT * FROM TCURRI)

SELECT * FROM T1
EXCEPT
SELECT DISTINCT NOMEM FROM T2
```

Supongamos los siguientes contenidos de las tablas (cursos de inglés, I, y francés, F):

TCURSO)

NOMCU
I
F

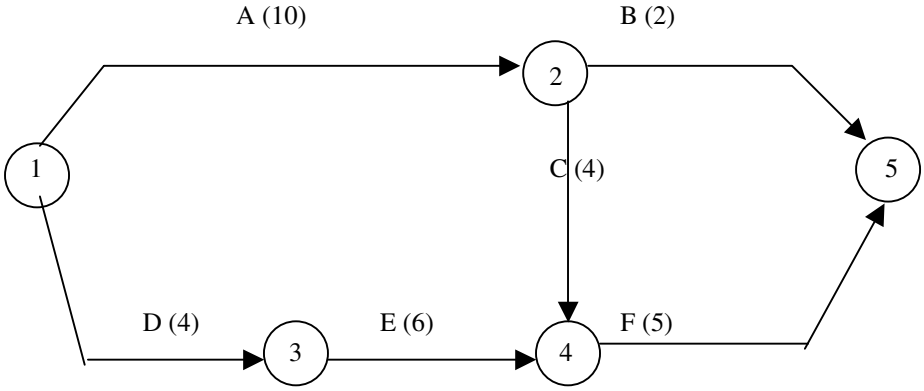
TCURRI)

NOMEM	NOMCU
PEPE	I
PACO	I
PACO	F

Resultado:

NOMEM
PACO

16.12). El siguiente diagrama PERT (Project Evaluation and Review Technique) representa las actividades de un proyecto.



Cada arco del grafo representa una actividad, con su nombre y duración entre paréntesis. Los nodos representan precedencia entre las actividades. Así por ejemplo la actividad F, que sale del nodo 4, no puede empezar hasta que hayan terminado completamente las que llegan al nodo, es decir, la C y la E.

Este grafo se almacena en una tabla TPROY (ORG, DES, NOM, DUR). Cada fila es un arco, con ORG = nodo origen, DEL = nodo destino, NOM = nombre, DUR = duración. Contenido:

ORG	DES	NOM	DUR
1	2	A	10
1	3	D	4
2	5	B	2
2	4	C	4
3	4	E	6
4	5	F	5

Se desea hallar la duración del proyecto.

Solución:

La duración del proyecto es la del camino más largo del grafo yendo desde el nodo inicial al final.

```
WITH CAMINOS (DES, TRAMOS, DUR) AS
                (SELECT DES, 1, DUR
                 FROM TPROY
                 WHERE ORG = 1
                       UNION ALL
                SELECT P.DES, TRAMOS + 1, C.DUR + P.DUR
                FROM CAMINOS C INNER JOIN TPROY P
```

```
                                ON C.DES = P.ORG AND  AND TRAMOS < 10)
    SELECT MAX (DUR) AS DURACION
    FROM CAMINOS
    WHERE DES = 5
```

Aunque un diagrama de este tipo no puede tener ciclos, hemos incluido un control de 10 tramos como máximo para evitar este caso si se presenta por error.

Resultado:

```
DURACION
----------
        19
```

Solucionario Capítulo 18

"SENTENCIAS PARA MODIFICAR DATOS"

En los ejercicios siguientes se supone que disponemos de una tabla llamada TEMPLE2 cuyas columnas tienen igual nombre y características que las de TEMPLE, y que inicialmente está vacía.
(Para resolver los ejercicios marcados con un asterisco es necesario haber leído el capítulo dedicado al manejo de fechas).

18.1). Insertar en TEMPLE2 una fila por cada empleado cuyo salario total (salario más comisión) supere al salario total medio de su departamento.

Interpretación 1:

Empleamos tres sentencias INSERT, una para los empleados con comisión, y las otras para los restantes, distinguiendo en éstos aquellos en cuyo departamento hay alguien con comisión y los que no (se muestran a continuación, separadas por punto y coma):

```
INSERT INTO   TEMPLE2
        SELECT        *
        FROM          TEMPLE E
        WHERE         COMIS IS NOT NULL AND
                      SALAR + COMIS > (SELECT AVG (SALAR) + AVG (COMIS)
                                       FROM   TEMPLE
                                       WHERE  NUMDE = E.NUMDE) ;

INSERT INTO   TEMPLE2
        SELECT        *
        FROM          TEMPLE E
        WHERE         COMIS IS NULL AND
                      SALAR > (SELECT AVG (SALAR) + AVG (COMIS)
                               FROM    TEMPLE
                               WHERE  NUMDE = E.NUMDE) AND
                      0 < (SELECT COUNT (DISTINCT COMIS)
                           FROM   TEMPLE
                           WHERE  NUMDE = E.NUMDE) ;

INSERT INTO   TEMPLE2
        SELECT        *
        FROM          TEMPLE E
        WHERE         COMIS IS NULL AND
                      SALAR > (SELECT AVG (SALAR)
                               FROM    TEMPLE
                               WHERE  NUMDE = E.NUMDE) AND
                      0 = (SELECT COUNT (DISTINCT COMIS)
                           FROM    TEMPLE
                           WHERE  NUMDE = E.NUMDE) ;
```

Empleando la función COALESCE se podría formular con una sola sentencia:

```
INSERT INTO          TEMPLE2
        SELECT       *
        FROM         TEMPLE E
        WHERE        COALESCE (SALAR + COMIS, SALAR) >
                        (SELECT COALESCE ( AVG (SALAR) + AVG (COMIS),
                                           AVG (SALAR))
                         FROM   TEMPLE
                         WHERE  NUMDE = E.NUMDE)
```

Interpretación 2:

Si la media de la comisión se interpreta con relación al total de empleados de un departamento, en vez de solamente a aquellos que tienen comisión, la formulación podría ser:

```
INSERT INTO     TEMPLE2
       SELECT   *
       FROM     TEMPLE E
       WHERE    COMIS IS NOT NULL AND
                SALAR + COMIS > (SELECT AVG (SALAR) + SUM (COMIS) / COUNT (*)
                                 FROM   TEMPLE
                                 WHERE  NUMDE = E.NUMDE) ;

INSERT INTO     TEMPLE2
       SELECT   *
       FROM     TEMPLE E
       WHERE    COMIS IS NULL AND
                SALAR > (SELECT AVG (SALAR) + SUM (COMIS) / COUNT (*)
                         FROM    TEMPLE
                         WHERE  NUMDE = E.NUMDE) AND
                EXISTS (SELECT *
                        FROM   TEMPLE
                        WHERE  NUMDE = E.NUMDE AND
                               COMIS IS NOT NULL) ;

INSERT INTO     TEMPLE2
       SELECT   *
       FROM     TEMPLE E
       WHERE    COMIS IS NULL AND
                SALAR > (SELECT AVG (SALAR)
                         FROM     TEMPLE
                         WHERE    NUMDE = E.NUMDE) AND
                NOT EXISTS (SELECT *
                            FROM     TEMPLE
                            WHERE    NUMDE = E.NUMDE AND
                                     COMIS IS NOT NULL) ;
```

En esta solución hemos utilizado el predicado EXISTS para hallar si todas las comisiones de un departamento son *Nulas*, en vez de contarlas como en la solución anterior. La evaluación de este predicado será más eficiente que el de la función COUNT, pues ésta obliga a leer todas las filas del departamento. Como antes, la utilización de COALESCE permite expresar la petición en una sola sentencia:

```
INSERT INTO     TEMPLE2
       SELECT   *
       FROM     TEMPLE E
       WHERE    COALESCE (SALAR + COMIS, SALAR) >
                (SELECT AVG (COALESCE (SALAR + COMIS, SALAR))
                 FROM   TEMPLE
                 WHERE  NUMDE = E.NUMDE)
```

18.2). Borrar en TEMPLE2 a los empleados cuyo salario (sin incluir comisión) supere al salario medio de los empleados de su departamento.

Solución:

```
DELETE FROM TEMPLE2 E
WHERE  SALAR > (SELECT AVG (SALAR)
                FROM   TEMPLE
                WHERE  NUMDE = E.NUMDE)
```

18.3). Borrar en TEMPLE2 a los empleados cuyo salario (sin incluir comisión) supere al salario medio de los empleados de su departamento, excluyéndole a él mismo.

Solución:

```
DELETE FROM TEMPLE2 E
WHERE  SALAR > (SELECT AVG (SALAR)
                FROM   TEMPLE
                WHERE  NUMDE = E.NUMDE AND
                       NUMEM <> E.NUMEM)
```

18.4). Sumar en TEMPLE2 la comisión en los empleados que la tengan al salario y actualizar éste con este nuevo valor, poniendo además *Nulo* en la comisión.

Solución:

```
UPDATE TEMPLE2
SET    SALAR = SALAR + COMIS, COMIS = NULL
WHERE  COMIS IS NOT NULL
```

18.5). Disminuir en TEMPLE2 en un 5 % el salario de los empleados que superan el 50 % del salario máximo de su departamento.

Solución:

```
UPDATE TEMPLE2 E
SET    SALAR = 0.95 * SALAR
WHERE  SALAR > (SELECT 0.5 * MAX (SALAR)
                FROM   TEMPLE
                WHERE  NUMDE = E.NUMDE)
```

18.6). Borrar todas las filas de TEMPLE2.

Solución:

```
DELETE FROM TEMPLE2
```

18.7). Supongamos que la tabla TEMPLE2 está de nuevo vacía y que disponemos de otra tabla, llamada TBORRA, que tiene una sola columna llamada NUMEM. En esta tabla hay una fila con el número de empleado por cada empleado que causa baja en la empresa en este mes. Hay que extraer de TEMPLE todas las filas de estos empleados y almacenarlas en TEMPLE2 para posteriores procesos, y además borrarlas de TEMPLE.

Solución:

Primero insertamos en TEMPLE2 las filas de los empleados que hay que dar de baja y luego las borramos de TEMPLE.

```
INSERT INTO   TEMPLE2
        SELECT E.*
        FROM   TEMPLE E, TBORRA B
        WHERE  E.NUMEM = B.NUMEM ;

DELETE FROM   TEMPLE E
WHERE  EXISTS (SELECT *
               FROM   TBORRA B
               WHERE  E.NUMEM = B.NUMEM) ;
```

18.8). En TEMPLE2 aumentar el salario de los empleados del departamento 111 en un 10% si no tienen ingresos por comisiones y en un 5% si sí los tienen.

Solución:

```
UPDATE TEMPLE2
```

```
SET SALAR = CASE
                WHEN COMIS IS NULL THEN SALAR*1.1
                ELSE SALAR*1.05
            END
WHERE NUMDE = 111
```

Solucionario Capítulo 19

"DEFINICIÓN DE TABLAS"

19.1). Suponiendo que las palabras siguientes sean o bien nombres de tablas o bien constantes, decir qué es cada una.

ABC
'ABC'
"ABC"
'Abc'
"Abc"
A.BC
A1.E2
1.E2
‘1.E2'
"A1"."E2"

Solución:

ABC	Tabla
'ABC'	Constante de hilera de caracteres
"ABC"	Tabla
'Abc'	Constante de hilera de caracteres
"Abc"	Tabla
A.BC	Tabla
A1.E2	Tabla
1.E2	Constante coma flotante
‘1.E2'	Constante de hilera de caracteres
"A1"."E2"	Tabla

19.2). Crear la tabla TEMPLE2 a la que se refieren los ejercicios del capítulo 18.

Solución:

```
CREATE TABLE TEMPLE2
    ( NUMEM INTEGER NOT NULL, NUMDE INTEGER NOT NULL, EXTEL
    SMALLINT NOT NULL, FECNA DATE NOT NULL, FECIN DATE NOT NULL,
    SALAR DECIMAL (4, 0)  NOT NULL, COMIS DECIMAL (4, 0), NUMHI
    SMALLINT NOT NULL, NOMEM VARCHAR (20) NOT NULL)
```

Resulta más cómodo crear la tabla con la cláusula LIKE:

```
CREATE TABLE TEMPLE2
LIKE TEMPLE INCLUDING DEFAULTS
```

19.3). (Variante del ejercicio 18.3). Hacer que el contenido de la tabla TEMPLE2 incluya a los empleados cuyo salario supere al salario medio de su departamento. Borrar entonces de TEMPLE2 a los empleados cuyo salario supere al salario medio de los empleados de su departamento que hay en TEMPLE2. Obtener un listado de todos los empleados que permanezcan, con sus departamentos y salarios. Emplear las sentencias de SQL que sean necesarias.

Solución 1 (creando una tabla de trabajo):

Para borrar los que tengan un salario superior a la media de los que están en su mismo departamento podemos usar una tabla de trabajo, que llamaremos TRA1. Ejecutaremos por tanto las sentencias siguientes.

1. Asegurarse de que TEMPLE2 está vacía.

```
DELETE FROM TEMPLE2;
```

2. Incluir los empleados cuyo salario supera a la media de su departamento.

```
INSERT INTO TEMPLE2
        SELECT *
        FROM   TEMPLE E
        WHERE  SALAR > (SELECT AVG (SALAR)
                        FROM   TEMPLE
                        WHERE  NUMDE = E.NUMDE)
```

3. Definir la tabla de trabajo.

```
CREATE TABLE TRA1
( NUMDE       INTEGER          NOT NULL
, SALME       DECIMAL (5, 0)   NOT NULL
)
```

4. Hallar el salario medio de los departamentos en TEMPLE2.

```
INSERT INTO TRA1
        SELECT   NUMDE, AVG (SALAR)
        FROM     TEMPLE2
        GROUP BY NUMDE
```

5. Borrar de TEMPLE2 los empleados cuyo salario supera al de su departamento.

```
DELETE FROM TEMPLE2 E
WHERE  SALAR > (SELECT SALME
                FROM   TRA1
                WHERE  NUMDE = E.NUMDE)
 GROUP BY NUMDE
```

6. Obtener listado.

```
SELECT NUMEM, NUMDE, SALAR
FROM TEMPLE2
```

7. Al terminar el proceso conviene destruir la tabla intermedia. Aunque la sentencia correspondiente no se ha explicado aún en el presente capítulo, la mostramos aquí para completar el ejercicio (lo mismo haremos en algunos de los siguientes):

```
DROP TABLE TRA1
```

Solución 2 (con tabla anidada):

```
DELETE FROM TEMPLE2;

INSERT INTO TEMPLE2
        SELECT *
        FROM   TEMPLE E
        WHERE  SALAR > (SELECT AVG (SALAR)
                        FROM   TEMPLE
                        WHERE  NUMDE = E.NUMDE) ;

DELETE FROM TEMPLE2 E
WHERE  SALAR > (SELECT SALME
```

```
                    FROM   (SELECT   NUMDE, AVG (SALAR) AS SALME
                                FROM     TEMPLE2
                                GROUP BY NUMDE) AS TRA1
                    WHERE  NUMDE = E.NUMDE) ;

SELECT NUMEM, NUMDE, SALAR
FROM TEMPLE2 ;
```

Solución 3 (con tablas temporales locales):

```
DELETE FROM TEMPLE2;

INSERT INTO TEMPLE2
WITH TEMPLE3 AS
        (  SELECT *
           FROM   TEMPLE E
           WHERE  SALAR > (SELECT AVG (SALAR)
                           FROM   TEMPLE
                           WHERE  NUMDE = E.NUMDE)),
     TRA1 AS
        (SELECT   NUMDE, AVG (SALAR) AS SALME
        FROM     TEMPLE3
        GROUP BY NUMDE)
SELECT E.NUMEM, E.NUMDE, E.SALAR
FROM TEMPLE3 E, TRA1 T
WHERE E.NUMDE = T.NUMDE AND
        SALAR <= SALME;

SELECT NUMEM, NUMDE, SALAR
FROM TEMPLE2 ;
```

Resultado:

NUMEM	NUMDE	SALAR
130	112	2900
150	121	4400
180	110	4800
240	111	2800
260	100	7200
285	122	3800
320	122	4050
330	112	2800
360	111	2500
420	130	4000

19.4). (Variante del ejercicio 18.5). Disminuir en TEMPLE2 (resultante del ejercicio anterior) en un 5 % el salario de los empleados que superan el 50 % del salario máximo de los empleados de TEMPLE2 que están en su mismo departamento. Obtener un listado con los empleados y el salario resultante.

Solución 1 (con una tabla de trabajo):

Para hallar el salario máximo de un departamento podemos usar una tabla intermedia de trabajo, que llamaremos TRA1. Ejecutaremos por tanto las sentencias siguientes.

1) Definir la tabla de trabajo.

```
CREATE TABLE TRA1
        ( NUMDE         INTEGER                 NOT NULL
```

```
, SAMAX          DECIMAL (5, 0)          NOT NULL
)
```

2) Hallar el salario máximo de los departamentos en TEMPLE2.

```
INSERT INTO TRA1
        SELECT   NUMDE, MAX (SALAR)
        FROM     TEMPLE2
        GROUP BY NUMDE
```

3) Actualizar el salario de los empleados que superan el 50 % del máximo.

```
UPDATE TEMPLE2 E
SET       SALAR = 0.95 * SALAR
WHERE  SALAR > (SELECT 0.5 * SAMAX
                      FROM   TRA1
                      WHERE  NUMDE = E.NUMDE)
```

4) Obtener listado.

```
SELECT NUMEM, SALARIO
FROM TEMPLE2
```

5) Destruir la tabla intermedia.

```
DROP TABLE TRA1
```

Resultado:

NUMEM	SALAR
130	2755
150	4180
180	4560
240	2660
260	6840
285	3610
320	3847
330	2660
360	2375
420	3800

Solución 2 (con tabla anidada):

```
UPDATE TEMPLE2 E
SET       SALAR = 0.95 * SALAR
WHERE  SALAR > (SELECT 0.5 * SAMAX
                     FROM   (SELECT   NUMDE, MAX (SALAR) AS SAMAX
                             FROM     TEMPLE2
                             GROUP BY NUMDE) AS TRA1
                    WHERE  NUMDE = E.NUMDE) ;

SELECT NUMEM, SALAR
FROM TEMPLE2 ;
```

19.5). Crear una tabla llamada TRABA1 con columnas llamadas NUMDE1, NUMDE2, NUMNIV, NOMDIR y MASALA. Las dos primeras columnas contienen números de departamento, la tercera un número entero,

la cuarta el nombre de un empleado y la quinta una suma de salarios. Las dos últimas pueden ser *Nulas*, así como la segunda.

Solución:

```
CREATE TABLE TRABA1
      ( NUMDE1      INTEGER          NOT NULL
      , NUMDE2      INTEGER
      , NUMNIV      INTEGER          NOT NULL
      , NOMDIR      VARCHAR (20)
      , MASALA      DECIMAL (7, 0)
      )
```

19.6). Hallar para cada departamento su masa salarial total (sin incluir comisión) incluyendo todos sus empleados y los de los departamentos que dependen de él a cualquier nivel (hasta cuatro niveles como máximo). Puede usarse la tabla TRABA1 del ejercicio anterior y más de una sentencia SQL si es necesario.

Solución 1(sin SQL recursivo):

1) Asegurarse de que TRABA1 está vacía.

```
DELETE FROM TRABA1
```

2) Masa salarial de los empleados de cada departamento (nivel 0).

```
INSERT INTO TRABA1 (NUMDE1, NUMNIV, MASALA)
 SELECT   D0.NUMDE, 0, SUM (SALAR)
 FROM     TDEPTO D0, TEMPLE EM
WHERE    D0.NUMDE = EM.NUMDE
GROUP BY D0.NUMDE
```

3) Masa salarial de los empleados de los departamentos que dependen directamente de cada departamento (nivel 1).

```
INSERT INTO TRABA1 (NUMDE1, NUMNIV, MASALA)
 SELECT   D0.NUMDE, 1, SUM (SALAR)
 FROM     TDEPTO D0, TDEPTO D1, TEMPLE EM
 WHERE    D0.NUMDE = D1.DEPDE AND
          D1.NUMDE = EM.NUMDE
 GROUP BY D0.NUMDE
```

4) Masa salarial de los empleados de los departamentos que dependen de cada departamento a nivel 2.

```
INSERT INTO TRABA1 (NUMDE1, NUMNIV, MASALA)
 SELECT   D0.NUMDE, 2, SUM (SALAR)
 FROM     TDEPTO D0, TDEPTO D1, TDEPTO D2, TEMPLE EM
 WHERE    D0.NUMDE = D1.DEPDE AND
          D1.NUMDE = D2.DEPDE AND
          D2.NUMDE = EM.NUMDE
 GROUP BY D0.NUMDE
```

5) Masa salarial de los empleados de los departamentos que dependen de cada departamento a nivel 3.

```
INSERT INTO TRABA1 (NUMDE1, NUMNIV, MASALA)
      SELECT   D0.NUMDE, 3, SUM (SALAR)
      FROM     TDEPTO D0, TDEPTO D1, TDEPTO D2,
               TDEPTO D3, TEMPLE EM
```

```
WHERE    D0.NUMDE = D1.DEPDE AND
         D1.NUMDE = D2.DEPDE AND
         D2.NUMDE = D3.DEPDE AND
         D3.NUMDE = EM.NUMDE
GROUP BY D0.NUMDE
```

6) Masa salarial de los empleados de los departamentos que dependen de cada departamento a nivel 4.

```
INSERT INTO TRABA1 (NUMDE1, NUMNIV, MASALA)
        SELECT   D0.NUMDE, 4, SUM (SALAR)
        FROM     TDEPTO D0, TDEPTO D1, TDEPTO D2,
                 TDEPTO D3, TDEPTO D4, TEMPLE EM
        WHERE    D0.NUMDE = D1.DEPDE AND
                 D1.NUMDE = D2.DEPDE AND
                 D2.NUMDE = D3.DEPDE AND
                 D3.NUMDE = D4.DEPDE AND
                 D4.NUMDE = EM.NUMDE
        GROUP BY D0.NUMDE
```

7) Hallar la masa salarial total, a cualquier nivel, de cada departamento.

Solución:

```
SELECT      NUMDE1, SUM (MASALA)
            FROM        TRABA1
            GROUP BY    NUMDE1
            ORDER BY    NUMDE1
```

Resultado:

```
NUMDE1      COL-2
----------  ---------
       100  103000
       110   45100
       111   17450
       112   18700
       120   31300
       121   12400
       122   16200
       130   11100
```

Solución 2(con SQL recursivo):

```
WITH TRA1  AS
        (SELECT D.NUMDE, DEPDE, SUM (SALAR) AS MASALA
        FROM TDEPTO D, TEMPLE E
        WHERE D.NUMDE = E.NUMDE
        GROUP BY D.NUMDE, DEPDE),
     TRA2 (DEPP, DEPH, NIV, MASALA) AS
        (SELECT NUMDE, NUMDE, 0, MASALA
        FROM TRA1

        UNION ALL

        SELECT P.DEPP, H.NUMDE, NIV + 1, H.MASALA
        FROM TRA2 P, TRA1 H
        WHERE P.DEPH = H.DEPDE AND NIV< 4
)
SELECT DEPP AS DEPTO, SUM (MASALA) AS MASA_SAL
```

```
FROM TRA2
GROUP BY DEPP
ORDER BY DEPP
```

19.7). Escribir las sentencias necesarias para hallar cuántos empleados hay cuyos salarios estén en los intervalos siguientes: (0 a 999), (1000 a 1999), (2000 a 2999), (3000 a 3999), (4000 a 4999), (5000 o más). Hallar también el salario medio dentro de cada intervalo.

Solución1 (con una tabla de trabajo):

Utilizaremos una tabla intermedia de trabajo para almacenar en ella la definición de los intervalos. Ejecutaremos las sentencias siguientes:

1. Definir la tabla de trabajo.

```
CREATE TABLE TRA1
      ( NUMIN       INTEGER          NOT NULL
      , LIMIN       DECIMAL (5, 0)   NOT NULL
      , LIMSU       DECIMAL (5, 0)   NOT NULL
      )
```

2. Incluir los límites de los intervalos (las distintas sentencias se muestran separadas por punto y coma).

```
INSERT INTO TRA1 VALUES (1, 000000, 000999) ;
INSERT INTO TRA1 VALUES (2, 001000, 001999) ;
INSERT INTO TRA1 VALUES (3, 002000, 002999) ;
INSERT INTO TRA1 VALUES (4, 003000, 003999) ;
INSERT INTO TRA1 VALUES (5, 004000, 004999) ;
INSERT INTO TRA1 VALUES (6, 005000, 099999) ;
```

3. Agrupar por intervalos.

```
SELECT    NUMIN, COUNT (*) AS EMPLES, AVG (SALAR) AS SALMED
FROM      TEMPLE, TRA1
WHERE     SALAR BETWEEN LIMIN AND LIMSU
GROUP BY  NUMIN

UNION

SELECT    NUMIN, 0 AS EMPLES, 0 AS SALMED
FROM      TRA1 T
WHERE     NOT EXISTS (SELECT *
                      FROM    TEMPLE
                      WHERE   SALAR BETWEEN T.LIMIN AND
                                            T.LIMSU)
ORDER BY 1
```

Resultado:

NUMIN	EMPLES	SALMED
1	0	0,00
2	6	1683,33
3	13	2396,15
4	7	3442,85
5	7	4350,00
6	1	7200,00

4. Destruir la tabla intermedia.

```
DROP TABLE TRA1
```

Solución 2 (con una tabla temporal local):

```
WITH TRA1 (NUMIN, LIMIN, LIMSU) AS
        ( VALUES  (1, 000000, 000999),
                  (2, 001000, 001999),
                  (3, 002000, 002999),
                  (4, 003000, 003999),
                  (5, 004000, 004999),
                  (6, 005000, 099999)
        )
SELECT NUMIN, COUNT (NUMEN) AS EMPLES, AVG (SALAR) AS SALMED
FROM TRA1 LEFT OUTER JOIN TEMPLE
        ON SALAR BETWEEN LIMIN AND LIMSU
GROUP BY NUMIN
ORDER BY NUMIN
```

***19.8).** Escribir las sentencias de SQL necesarias para hallar cuántos años hay en los que ha habido ingresos de nuevos empleados.

Solución 1 (con tabla de trabajo):

Ejecutaremos las sentencias siguientes:

1. Definir una tabla de trabajo.

```
CREATE TABLE TRA1
        ( NUAÑO         INTEGER                 NOT NULL
        )
```

2. Hallar los años en que ha habido ingresos.

```
INSERT INTO TRA1
        SELECT   DISTINCT YEAR (FECIN)
        FROM     TEMPLE
```

3. Contarlos.

```
SELECT          COUNT (*)
FROM            TRA1
```

4. Destruir la tabla intermedia.

```
DROP TABLE TRA1
```

Solución 2 (con tabla anidada):

```
SELECT          COUNT (*)
FROM            (SELECT   DISTINCT YEAR (FECIN)
                 FROM     TEMPLE) AS TRA1
```

Resultado:

```
COL-1
--------
      16
```

***19.9).** Hallar el salario medio actual de los empleados que han ingresado cada año.

Solución 1 (con tabla de trabajo).

Ejecutaremos las sentencias siguientes:

1. Definir una tabla de trabajo.

```
CREATE TABLE TRA1
      ( NUAÑO       INTEGER            NOT NULL
      , NUMEM       INTEGER            NOT NULL
      , SALAR       DECIMAL (5, 0)     NOT NULL
      )
```

2. Hallar los empleados que han ingresado cada año.

```
                INSERT INTO TRA1
SELECT  YEAR (FECIN),  NUMEM, SALAR FROM    TEMPLE
```

3. Hallar su salario medio.

```
SELECT      NUAÑO, AVG (SALAR)
FROM        TRA1
GROUP BY    NUAÑO
ORDER BY    NUAÑO
```

4. Destruir la tabla intermedia.

```
DROP TABLE TRA1
```

Solución 2 (agrupando con una expresión):

```
SELECT    YEAR (FECIN) AS NUAÑO, AVG (SALAR) AS MEDAÑO
          FROM         TEMPLE
          GROUP BY     YEAR (FECIN)
          ORDER BY      NUAÑO
```

Resultado:

NUAÑO	MEDAÑO
1948	4400.00
1950	3100.00
1956	4800.00
1959	3800.00
1962	3000.00
1966	3300.00
1967	4500.00
1968	3800.00
1969	2900.00
1971	3550.00
1972	2800.00
1978	4050.00
1984	4500.00
1986	2090.00
1987	1916.66
1988	2075.00

***19.10).** Hallar para cada mes de los años 1987 y 1988 el número de empleados que ingresaron y los valores medio, mínimo y máximo de sus salarios actuales.

Solución 1 (con tabla de trabajo):

1. Definir una tabla de trabajo.

```
CREATE TABLE TRA1
       ( NUAÑO      INTEGER          NOT NULL
       , NUMES      INTEGER          NOT NULL
       )
```

2. Hallar los años y meses en que hubo ingresos de nuevos empleados.

```
INSERT INTO TRA1
        SELECT   DISTINCT YEAR (FECIN), MONTH (FECIN)
         FROM    TEMPLE
        WHERE    YEAR (FECIN) IN (1987, 1988)
```

3. Contar los ingresos y hallar los salarios medio, mínimo y máximo.

```
SELECT      NUAÑO, NUMES, COUNT(*) AS EMPLES, DECIMAL(AVG
            (SALAR),6,2) AS    MEDSA, MIN (SALAR) AS MINSA, MAX (SALAR)
            AS MAXSA
FROM        TRA1, TEMPLE
WHERE       NUAÑO = YEAR (FECIN) AND NUMES = MONTH (FECIN)
GROUP BY    NUAÑO, NUMES
ORDER BY    NUAÑO, NUMES
```

4. Destruir la tabla intermedia.

```
DROP TABLE TRA1
```

Resultado:

NUAÑO	NUMES	EMPLES	MEDSA	MINSA	MAXSA
1987	1	2	1950.00	1900	2000
1987	11	1	1850.00	1850	1850
1988	1	3	1533.33	1000	1800
1988	10	1	1750.00	1750	1750
1988	11	2	3050.00	2100	4000

Solución 2 (con tabla temporal local):

```
WITH TRA1 (NUAÑO, NUMES) AS
        (SELECT   DISTINCT YEAR (FECIN), MONTH (FECIN)
                  FROM    TEMPLE
                 WHERE    YEAR (FECIN) IN (1987, 1988)
        )
SELECT      NUAÑO, NUMES, COUNT(*) AS EMPLES, DECIMAL(AVG (SALAR),6,2) AS MEDSA,
MIN (SALAR) AS MINSA, MAX (SALAR) AS MAXSA
        FROM        TRA1, TEMPLE
        WHERE                NUAÑO = YEAR (FECIN) AND NUMES = MONTH (FECIN)
        GROUP BY    NUAÑO, NUMES
        ORDER BY    NUAÑO, NUMES
```

Solución 3 (agrupando con expresiones):

```
SELECT    YEAR (FECIN) AS NUAÑO, MONTH (FECIN) AS NUMES, COUNT(*) AS EMPLES,
          DECIMAL(AVG (SALAR),6,2) AS MEDSA, MIN (SALAR) AS MINSA, MAX (SALAR)
          AS MAXSA
FROM      TEMPLE
WHERE     YEAR (FECIN) IN (1987, 1988)
GROUP BY YEAR (FECIN), MONTH (FECIN)
ORDER BY  NUAÑO, NUMES
```

***19.11).** Igual que el ejercicio anterior, pero mostrando en el resultado todos los meses, incluso aquellos en los que no haya habido ingresos de nuevos empleados.

Solución 1 (con tabla de trabajo):

1. Definir una tabla de trabajo.

```
CREATE TABLE TRA1
      ( NUAÑO      INTEGER          NOT NULL
      , NUMES      INTEGER          NOT NULL
      )
```

2. Preparar valores de años y meses.

```
INSERT INTO TRA1 VALUES (1987, 1)  ;
INSERT INTO TRA1 VALUES (1987, 2)  ;
INSERT INTO TRA1 VALUES (1987, 3)  ;
INSERT INTO TRA1 VALUES (1987, 4)  ;
INSERT INTO TRA1 VALUES (1987, 5)  ;
INSERT INTO TRA1 VALUES (1987, 6)  ;
INSERT INTO TRA1 VALUES (1987, 7)  ;
INSERT INTO TRA1 VALUES (1987, 8)  ;
INSERT INTO TRA1 VALUES (1987, 9)  ;
INSERT INTO TRA1 VALUES (1987, 10) ;
INSERT INTO TRA1 VALUES (1987, 11) ;
INSERT INTO TRA1 VALUES (1987, 12) ;
INSERT INTO TRA1 VALUES (1988, 1)  ;
INSERT INTO TRA1 VALUES (1988, 2)  ;
INSERT INTO TRA1 VALUES (1988, 3)  ;
INSERT INTO TRA1 VALUES (1988, 4)  ;
INSERT INTO TRA1 VALUES (1988, 5)  ;
INSERT INTO TRA1 VALUES (1988, 6)  ;
INSERT INTO TRA1 VALUES (1988, 7)  ;
INSERT INTO TRA1 VALUES (1988, 8)  ;
INSERT INTO TRA1 VALUES (1988, 9)  ;
INSERT INTO TRA1 VALUES (1988, 10) ;
INSERT INTO TRA1 VALUES (1988, 11) ;
INSERT INTO TRA1 VALUES (1988, 12) ;
```

3. Contar los ingresos y hallar los salarios medio, mínimo y máximo.

```
SELECT      NUAÑO, NUMES, COUNT(*),
            AVG (SALAR), MIN (SALAR), MAX (SALAR)
FROM        TRA1, TEMPLE
WHERE       NUAÑO = YEAR (FECIN) AND
            NUMES = MONTH (FECIN)
GROUP BY    NUAÑO, NUMES
UNION
 SELECT     NUAÑO, NUMES, 0, 0, 0, 0
FROM        TRA1
```

```
WHERE                   NOT EXISTS (SELECT *
                                    FROM   TEMPLE
                                    WHERE  YEAR (FECIN) = TRA1.NUAÑO AND
                                           MONTH (FECIN) = TRA1.NUMES)
ORDER BY       1, 2
```

4. Destruir la tabla intermedia.

```
DROP TABLE TRA1
```

Solución 2 (con una tabla temporal local):

```
WITH TRA1 (NUAÑO, NUMES) AS
        (VALUES ( (1987, 1), (1987, 2), (1987, 3), (1987, 4), (1987, 5), (1987, 6),
                  (1987, 7), (1987, 8), (1987, 9), (1987, 10), (1987, 11), (1987, 12),
                  (1988, 1), (1988, 2), (1988, 3), (1988, 4), (1988, 5), (1988, 6),
                  (1988, 7), (1988, 8) ,(1988, 9), (1988, 10), (1988, 11), (1988, 12)
                  )
 )

SELECT  NUAÑO, NUMES, COUNT (NUMEM) AS EMPLES, AVG (SALAR) AS MEDSA,
        MIN (SALAR) AS MINSA, MAX (SALAR) AS MAXSA
FROM TRA1 LEFT OUTER JOIN TEMPLE
      ON NUAÑO = YEAR (FECIN) AND NUMES = MONTH (FECIN)
GROUP BY NUAÑO, NUMES
ORDER BY NUAÑO, NUMES
```

19.12). (Variante del ejercicio 11.21). Cada director de departamento, tanto en propiedad como en funciones, es responsable de las promociones y sueldos de los empleados del departamento que dirige, excluido él mismo. Además la promoción y sueldo de un director en propiedad es responsabilidad del primero de sus directores en propiedad que se halle ascendiendo en la jerarquía de la organización. Escribir las sentencias SQL necesarias para hallar para cada director su nombre y la masa salarial total de los empleados y directores de cuya promoción es responsable, por orden alfabético.

Solución 1 (sin recursividad):

Ejecutaremos los pasos siguientes.

1. Definir una tabla de trabajo.

```
CREATE TABLE TRA1
        ( NUDEP         INTEGER             NOT NULL
        , NUDIR         INTEGER             NOT NULL
        , NUNIV         INTEGER             NOT NULL
        , MASAL         DECIMAL (7, 0)      NOT NULL
        )
```

2. Masa salarial de los empleados de cada departamento.

```
INSERT INTO TRA1
        SELECT   D0.NUMDE, D0.DIREC, 0, SUM (SALAR)
        FROM     TDEPTO D0, TEMPLE EM
        WHERE    D0.NUMDE = EM.NUMDE AND
                 D0.DIREC <> EM.NUMEM
      GROUP BY D0.NUMDE, D0.DIREC
```

3. Masa salarial de los directores en propiedad de los departamentos que dependen directamente de cada director en propiedad (nivel 1).

```
INSERT INTO TRA1
SELECT   D0.NUMDE, D0.DIREC, 1, SUM (SALAR)
 FROM    TDEPTO D0, TDEPTO D1, TEMPLE EM
 WHERE   D0.NUMDE = D1.DEPDE AND
    D0.TIDIR = 'P' AND D1.TIDIR = 'P' AND
    D1.DIREC = EM.NUMEM
 GROUP BY D0.NUMDE, D0.DIREC
```

4. Masa salarial de los directores en propiedad de los departamentos que dependen de cada director en propiedad a nivel 2.

```
INSERT INTO TRA1
        SELECT   D0.NUMDE, D0.DIREC, 2, SUM (SALAR)
        FROM    TDEPTO D0, TDEPTO D1, TDEPTO D2,  TEMPLE EM
        WHERE   D0.NUMDE = D1.DEPDE AND
                D1.NUMDE = D2.DEPDE AND
                D0.TIDIR = 'P' AND D1.TIDIR = 'F' AND
                D2.TIDIR = 'P' AND
                D2.DIREC = EM.NUMEM
        GROUP BY D0.NUMDE, D0.DIREC
```

5. Masa salarial de los directores en propiedad de los departamentos que dependen de cada director en propiedad a nivel 3.

```
INSERT INTO TRA1
        SELECT   D0.NUMDE, D0.DIREC, 3, SUM (SALAR)
        FROM    TDEPTO D0, TDEPTO D1, TDEPTO D2,
                TDEPTO D3, TEMPLE EM
        WHERE   D0.NUMDE = D1.DEPDE AND
                D1.NUMDE = D2.DEPDE AND
                D2.NUMDE = D3.DEPDE AND
                D0.TIDIR = 'P' AND D1.TIDIR = 'F' AND
                D2.TIDIR = 'F' AND D3.TIDIR = 'P' AND
                D3.DIREC = EM.NUMEM
        GROUP BY D0.NUMDE, D0.DIREC
```

6. Masa salarial de los directores en propiedad de los departamentos que dependen de cada director en propiedad a nivel 4.

```
INSERT INTO TRA1
        SELECT   D0.NUMDE, D0.DIREC, 4, SUM (SALAR)
        FROM    TDEPTO D0, TDEPTO D1, TDEPTO D2,
                TDEPTO D3, TDEPTO D4, TEMPLE EM
        WHERE   D0.NUMDE = D1.DEPDE AND
                D1.NUMDE = D2. DEPDE AND
                D2.NUMDE = D3. DEPDE AND
                D3.NUMDE = D4. DEPDE AND
                D0.TIDIR = 'P' AND D1.TIDIR = 'F' AND
                D2.TIDIR = 'F' AND D3.TIDIR = 'F' AND
                D4.TIDIR = 'P' AND
                D4.DIREC = EM.NUMEM
        GROUP BY D0.NUMDE, D0.DIREC
```

7. Comprobar que no hay en nuestra organización dependencias de departamentos a nivel mayor que 2.

```
SELECT MAX (NUNIV)
FROM   TRA1
```

8. Hallar la masa salarial total, a cualquier nivel, de cada director en propiedad.

```
SELECT      NOMEM, SUM (MASAL)
FROM        TRA1, TEMPLE
WHERE               NUDIR = NUMEM
GROUP BY    NUDIR, NOMEM
ORDER BY    1
```

9. Destruir la tabla intermedia.

```
DROP TABLE TRA1
```

Solución 2 (con recursividad):

```
WITH TRA1 (D1, D2, N, DIR, TIP, SALA) AS
    (SELECT D.NUMDE, E.NUMDE, 0, DIREC, TIDIR, SUM (SALAR)
        FROM TDEPTO D, TEMPLE E
        WHERE D.NUMDE = E.NUMDE AND D.DIREC <> E.NUMEM
        GROUP BY D.NUMDE, E.NUMDE, DIRE, TIDIR),
  TRA (DEPP, DEPH, NIV, NUDIRP, TIDIRH, SALA) AS
        (SELECT *  FROM TRA1

        UNION ALL

        SELECT P.DEPP, H.NUMDE, NIV + 1, NUDIRP, TIDIR, E.SALAR
        FROM TRA P, TDEPTO H, TEMPLE E
        WHERE P.DEPH = H.DEPDE AND
            ((NIV = 0 AND TIDIRH = 'P') OR (NIV > 0 AND TIDIRH = 'F'))
            AND H.DIREC = E.NUMEM
        )

    SELECT NOMEM, SUM (SALA) AS MASA_SAL
    FROM TRA, TEMPLE
    WHERE (NIV = 0 OR TIDIRH = 'P') AND NUDIRP = NUMEM
    GROUP BY NOMEM
    ORDER BY NOMEM
```

Resultado:

```
NOMEM                   MASA_SAL
----------------------  ----------
CAMPS, AURELIO               11700
GARCIA, AUGUSTO               6900
GARCIA, OCTAVIO              14900
LOPEZ, ANTONIO               26200
PEREZ, JULIO                 10700
PEREZ, MARCOS                25400
```

19.13). Crear una tabla con dos columnas, una *smallint* y otra *integer*. Dar de alta filas en la tabla de manera que los datos de la columna primera vayan del 1 al 20 y los de la segunda del 20 al 1.

Solución:

```
CREATE TABLE TABLA2 (COL1 SMALLINT, COL2 INTEGER) ;

INSERT INTO TABLA2
        WITH TTEMP (C1,C2) AS
            (VALUES (1,20)
```

```
        UNION ALL
    SELECT (C1+1), (C2-1)
    FROM TTEMP
    WHERE C1< 20)
SELECT * FROM TTEMP ;
```

Solucionario Capítulo 20

"USO DE OTROS TIPOS DE DATOS (UDT, LOB) Y FUNCIONES DE USUARIO"

20.1). Definir un UDT "KMHORA" que se base en un tipo de datos DEC(4,0). Definir las operaciones básicas de suma y resta sobre el UDT. Crear una tabla que contenga las siguientes columnas (tipos de datos): LOCALIZACIÓN_1 (VARCHAR(20)), LOCALIZACIÓN_2 (VARCHAR(20)), TIPO_DE_CARRETERA (CHAR(1)), VELOCIDAD_MAXIMA (KMHORA).

Solución:

```
CREATE DISTINCT TYPE KMHORA AS DECIMAL(4,0) WITH COMPARISONS;

CREATE FUNCTION "+" (KMHORA, KMHORA)
RETURNS KMHORA
SOURCE SYSIBM."+" (DECIMAL(4,0), DECIMAL(4,0));

CREATE FUNCTION "-" (KMHORA, KMHORA)
RETURNS KMHORA
SOURCE SYSIBM."-" (DECIMAL(4,0), DECIMAL(4,0));

CREATE TABLE CARRETERAS
(LOCALIZACION_1 VARCHAR(20),
LOCALIZACION_2 VARCHAR(20),
TIPO_DE_CARRETERA CHAR(1),
VELOCIDAD_MAXIMA KMHORA) ;
```

20.2). Definir las funciones escalares MAX y MIN sobre el UDT anterior. Escribir una sentencia SQL que permita obtener cuales son las dos localidades comunicadas por la carretera en la que se puede viajar a más velocidad.

Solución:

```
CREATE FUNCTION MAX (KMHORA)
RETURNS KMHORA SOURCE SYSIBM.MAX (DECIMAL (4,0));

CREATE FUNCTION MIN (KMHORA)
RETURNS KMHORA SOURCE SYSIBM.MIN (DECIMAL (4,0));

SELECT LOCALIZACION_1, LOCALIZACION_2
FROM CARRETERAS
WHERE VELOCIDAD_MAXIMA = (SELECT MAX (VELOCIDAD_MAXIMA)
                          FROM CARRETERAS)
```

Solucionario Capítulo 21

"VISTAS Y AUTORIZACIONES"

21.1). Crear una vista donde aparezcan todas las filas de la tabla TDEPTO pero no aparezca la columna PRESU. Hacerla pública para consultas.

Solución:

```
CREATE VIEW VDEPTO
        AS SELECT NUMDE, NUMCE, DIREC,
                  TIDIR, DEPDE, NOMDE
           FROM   TDEPTO ;

GRANT  SELECT ON VDEPTO TO PUBLIC ;
```

21.2). Crear una vista llamada VEMCOM que contenga las columnas NUMEM, NUMDE, EXTEL y NOMEM, de los empleados que trabajan a comisión. Hacerla pública para consultas.

Solución:

```
CREATE VIEW VENCOM AS
            SELECT NUMEM, NUMDE, EXTEL, NOMEM
            FROM   TEMPLE
            WHERE  COMIS IS NOT NULL ;

GRANT  SELECT ON VENCOM TO PUBLIC ;
```

***21.3).** Crear una vista llamada VJUBIL1 en la que aparezcan todos los datos de los empleados que cumplen 65 años de edad este año de modo que puedan ser consultados solamente por el director de Personal, suponiendo que éste tiene como identificador U0150.

Solución:

```
CREATE VIEW VJUBIL1 AS
        SELECT *
        FROM    TEMPLE
        WHERE  YEAR (CURRENT DATE) - YEAR (FECNA) = 65 ;

GRANT  SELECT ON VJUBIL1 TO U0150 ;
```

***21.4).** (Variante del ejercicio anterior). Crear una vista llamada VJUBIL2 en la que aparezcan todos los datos de los empleados que cumplen 65 años o más en este año de modo que puedan ser consultados solamente por el director de Personal, suponiendo que los identificadores de todos los empleados están almacenados en la tabla TIDPER descrita en un ejemplo del párrafo sobre "Utilidad de las vistas" en este capítulo.

Solución:

```
CREATE VIEW VJUBIL2 AS
        SELECT E.*
        FROM    TIDPER I, TDEPTO D, TEMPLE E
        WHERE  IDPEM = USER AND
                  I.NUMEM = DIREC AND
                  NOMDE LIKE '%PERSONAL%' AND
                  YEAR (CURRENT DATE) - YEAR (FECNA) >= 65;

GRANT  SELECT ON VJUBIL2 TO PUBLIC ;
```

Esta vista está vacía para todos los usuarios excepto el director de Personal. Si éste cambia, la vista refleja este cambio automáticamente en cuanto se haya actualizado TDEPTO.

***21.5).** Crear una vista llamada VJUBIL3 en la que aparezcan todos los datos de los empleados que cumplen 65 años o más este año de modo que estos datos puedan ser consultados solamente por los empleados del departamento de Personal.

Solución:

```
CREATE VIEW VJUBIL3  AS
        SELECT J.*
        FROM    TIDPER I, TEMPLE E, TDEPTO D,  TEMPLE J
        WHERE  IDPEM = USER AND
                    I.NUMEM = E.NUMEM AND
                    E.NUMDE = D.NUMDE AND
                    D.NOMDE LIKE '%PERSONAL%' AND
                   YEAR (CURRENT DATE) - YEAR (J.FECNA) >= 65 ;

GRANT  SELECT ON VJUBIL3 TO PUBLIC ;
```

21.6). Crear una vista llamada VINIDE, con columnas llamadas NUMDE1, NIVEL, NUMDE2, NOMDIR y MASALA. Las columnas NUMDE1 y NUMDE2 son números de departamento. NUMDE2 depende administrativamente de NUMDE1, a un nivel que viene dado por el contenido de la columna NIVEL. En NOMDIR está el nombre del director del departamento NUMDE2. En MASALA la masa salarial total de NUMDE2. La vista debe contener los departamentos dependientes administrativamente de otros desde el nivel 0 hasta el 4, como máximo.

Solución 1 (sin recursividad):

```
CREATE VIEW VINIDE (NUMDE1, NIVEL, NUMDE2, NOMDIR, MASALA)

        AS
                SELECT   D0.NUMDE, 0, D0.NUMDE, ED.NOMEM, SUM (EM.SALAR)
                FROM     TDEPTO D0, TEMPLE ED, TEMPLE EM
                WHERE D0.DIREC = ED.NUMEM AND D0.NUMDE = EM.NUMDE
                GROUP BY D0.NUMDE, ED.NOMEM
        UNION
                 SELECT   D0.NUMDE, 1, D1.NUMDE, ED.NOMEM, SUM (EM.SALAR)
                 FROM     TDEPTO D0, TDEPTO D1, TEMPLE ED, TEMPLE EM
                 WHERE    D0.NUMDE = D1.DEPDE AND
                             D1.DIREC = ED.NUMEM AND D1.NUMDE = EM.NUMDE
                 GROUP BY D0.NUMDE, D1.NUMDE, ED.NOMEM
        UNION
                 SELECT   D0.NUMDE, 2, D2.NUMDE, ED.NOMEM, SUM (EM.SALAR)
                 FROM     TDEPTO D0, TDEPTO D1, TDEPTO D2, TEMPLE ED,TEMPLE EM
                 WHERE    D0.NUMDE = D1.DEPDE AND
                             D1. NUMDE = D2. DEPDE AND
                             D2.DIREC = ED.NUMEM AND D2.NUMDE = EM.NUMDE
                 GROUP BY D0.NUMDE, D2.NUMDE, ED.NOMEM
        UNION
                SELECT   D0.NUMDE, 3, D3.NUMDE, ED.NOMEM, SUM (EM.SALAR)
                FROM     TDEPTO D0, TDEPTO D1, TDEPTO D2, TDEPTO D3,
                         TEMPLE ED, TEMPLE EM
                WHERE    D0.NUMDE = D1.DEPDE AND
                             D1.NUMDE = D2.DEPDE AND
                             D2.NUMDE = D3.DEPDE AND
                             D3.DIREC = ED.NUMEM AND D3.NUMDE = EM.NUMDE
                GROUP BY D0.NUMDE, D3.NUMDE, ED.NOMEM
        UNION
                SELECT   D0.NUMDE, 4, D4.NUMDE, ED.NOMEM,  SUM (EM.SALAR)
                FROM     TDEPTO D0, TDEPTO D1, TDEPTO D2, TDEPTO D3, TDEPTO D4,
                             TEMPLE ED, TEMPLE EM
```

```
WHERE    D0.NUMDE = D1.DEPDE AND
         D1.NUMDE = D2. DEPDE AND
         D2.NUMDE = D3. DEPDE AND
         D3.NUMDE = D4. DEPDE AND
         D4.DIREC = ED.NUMEM AND D4.NUMDE = EM.NUMDE
GROUP BY D0.NUMDE, D4.NUMDE, ED.NOMEM
```

Solución 2 (con recursividad):

```
CREATE VIEW VINIDE (NUMDE1, NIVEL, NUMDE2, NOMDIR, MASALA) AS
    WITH TRA1  AS
        (SELECT D.NUMDE, DEPDE, ED.NOMEM AS NOMBR, SUM (EM.SALAR) AS MASALA
         FROM TDEPTO D, TEMPLE ED, TEMPLE EM
         WHERE D.DIREC = ED.NUMEM AND D.NUMDE = EM.NUMDE
         GROUP BY D.NUMDE, DEPDE, ED.NOMEM),
       TRA2 (DEPP, NIV, DEPH, NOMDIRH, MASALA) AS
         (SELECT NUMDE, 0, NUMDE, NOMBR, MASALA
         FROM TRA1

         UNION ALL

         SELECT P.DEPP, NIV + 1, H.NUMDE, H.NOMBR, H.MASALA
         FROM TRA2 P, TRA1 H
         WHERE P.DEPH = H.DEPDE AND NIV< 4
         )
    SELECT *
    FROM TRA2
```

21.7). Vaciar de filas la tabla TRABA1 creada en los ejercicios del capítulo sobre "Definición de tablas" y llenarla de nuevo con las dependencias entre departamentos hasta cuatro niveles como máximo.

Solución:

```
DELETE FROM TRABA1 ;

INSERT INTO TRABA1 SELECT NUMDE1, NUMDE2, 2, NOMDIR, MASALA
                          FROM   VINIDE ;
```

21.8). Crear una vista sobre TRABA1 llamada VIORDE (acrónimo de VIsta de la ORganización de los DEpartamentos) donde aparezcan las columnas NUMDE1, NUMDE2, NUMNIV y NOMDIR de TRABA1 y todas sus filas. Hacer pública para consultas esta vista.

Solución:

```
CREATE VIEW VIORDE AS
        SELECT NUMDE1, NUMDE2, NUMNIV, NOMDIR
        FROM   TRABA1 ;

GRANT  SELECT ON VIORDE TO PUBLIC ;
```

21.9). Utilizando la tabla de usuarios TIDPER mencionada más arriba crear una vista llamada VIORDI (VIsta de la Organización para DIrectores) tal que aparezcan en ella todas las columnas de TRABA1, pero sólo las filas de los departamentos de los que el usuario sea director, inmediato o no. Hacer pública para consultas esta vista.

Solución:

```
CREATE VIEW VIORDI  AS
        SELECT T.*
        FROM   TIDPER U, TDEPTO D, TRABA1 T
        WHERE  IDPEM = USER AND
                U.NUMEM = D.DIREC AND
```

```
                    D.NUMDE = T.NUMDE1 ;

        GRANT  SELECT ON VIORDI TO PUBLIC ;
```

De este modo la vista está vacía para usuarios que no sean directores, y para los que lo sean sólo contiene filas de los departamentos que dirige o que dependen de éstos.

Obsérvese sin embargo que un cambio en TDEPTO o TEMPLE (por ejemplo, si cambia el director de un departamento), no se refleja automáticamente en esta vista. Para que ésta represente la última situación hay que actualizar el contenido de TRABA1 cada vez que haya cambios que la puedan afectar. Si se desea evitar esta situación, se podría hacer de TRABA1 una tabla temporal local definida dentro del CREATE VIEW, como ya se hizo en el ejercicio 21.6 . Dejamos este ejercicio al lector.

21.10). Se desea autorizar a cada director de departamento a ver todos los datos de los empleados de los departamentos que dirige, tanto en propiedad como en funciones. Crear una vista para ello y autorizarla al público para consultas.

Solución:

```
        CREATE VIEW VEMPLE AS
                SELECT E.*
                FROM   TIDPER U, TDEPTO D, TEMPLE E
                WHERE  IDPEM = USER AND
                       U.NUMEM = D.DIREC AND
                       D.NUMDE = E.NUMDE ;

        GRANT  SELECT ON VEMPLE TO PUBLIC ;
```

Como en el ejercicio anterior, la vista está vacía para usuarios que no sean directores, y para los que lo sean sólo contiene filas de los empleados de los departamentos que dirige. Pero, al contrario que en el ejercicio anterior, los cambios en TDEPTO, TIDPER o TEMPLE, son automáticamente reflejados en la vista puesto que ésta está definida directamente sobre estas tablas (en vez de sobre una tabla intermedia como era el caso en el ejercicio anterior).

21.11). Se desea autorizar a cada director de departamento a ver todos los datos de los empleados de los departamentos que dirige o de los que dependen de ellos a cualquier nivel. Crear una vista para ello y autorizar para que pueda ser consultada.

Solución:

Para acceder a los departamentos dependientes podemos usar la TRABA1 con el contenido almacenado en ella en el ejercicio 21.7 anterior. Puesto que usaremos una tabla intermedia, es aplicable la observación que hacíamos en el ejercicio 21.9, a saber, que las actualizaciones en las tablas que afecten a las dependencias o directores de los departamentos no se reflejan automáticamente en TRABA1, ni por lo tanto tampoco en nuestra vista. Para que la vista estuviera actualizada en todo momento habría que definir TRABA1 como una vista temporal local en el CREATE VIEW, como hicimos en el ejercicio 21.6, lo cual dejamos al lector.

```
        CREATE VIEW VEMPLED AS
                SELECT E.*, NUMNIV
                FROM    TIDPER U, TRABA1 D, TEMPLE E
                WHERE  IDPEM = USER AND
                       U.NUMEM = D.NUMDE1 AND
                       D.NUMDE2 = E.NUMDE ;

        GRANT  SELECT ON VEMPLED TO PUBLIC ;
```

Solucionario Capítulo 23

"PROGRAMACIÓN SIN CURSORES"

23.1). Escribir un programa que lea de un fichero secuencial que contenga números de empleados, y vaya borrando de la tabla TEMPLE. En caso de no existir el empleado, guárdese ese número en otro fichero de salida emitiéndose un mensaje de error.

Solución:

```
/************************************************
EJERCICIO: 23.1

ENUNCIADO: Escribir un programa que lea de un fichero
secuencial que contenga números de empleados, y vaya
borrando de la tabla TEMPLE. En caso de no existir el
empleado, guárdese ese número en otro fichero de
salida emitiéndose un mensaje de error.

************************************************/

/* Librerías estándares de C */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/* Librería para el API sqlaintp( ) */
#include <sql.h>

/* Inclusión del SQL Communication Area */
EXEC SQL INCLUDE SQLCA;

/* Prototipo de la función error() */
void error (char *);

/************************************************************
               PROGRAMA PRINCIPAL
************************************************************/
void main() {

 /* Declaración de variables huéspedes */
 EXEC SQL BEGIN DECLARE SECTION;
   long numem;
 EXEC SQL END DECLARE SECTION;

 /* Declaración de las variables para los ficheros */
 FILE *entrada;
 FILE *salida;

 /* Apertura de los fichero de entrada y salida*/
 entrada=fopen ("entrada.txt","r");
 salida=fopen ("salida.txt","w");

 /* Conexión a la base de datos BDEJE */
 EXEC SQL CONNECT TO BDEJE;

 /* Tratamiento de errores */
 error ("Conexión");

 printf ("BORRADO DE EMPLEADOS\n\n");
```

```
 /* Se recorre en un bucle el fichero de entrada */
 do
 {

   /* lectura de un regitro del fichero de entrada*/
   fscanf(entrada,"%d", &numem);

   /* borrado de dicho empleado */
   EXEC SQL DELETE FROM TEMPLE WHERE NUMEM= :numem;

   /* Tratamiento de errores */
   error ("Borrado");

   /* Si no existe se escribe en el fichero de salida */
   if (sqlca.sqlcode==100)
     fprintf (salida,"%d\n",numem);
   else
     printf ("Borrado el empleado número: %d\n",numem);

 } while (!feof (entrada));

 /* Desconexión de la base de datos */
 EXEC SQL CONNECT RESET;

 /* Tratamiento de errores */
 error ("Desconexión de la base de datos");

} /* Fin del main  */

/************************************************************
               FUNCION ERROR ()
************************************************************/

/* Recibe una hilera indicando donde se ha producido un error
   y hace uso del api sqlaintp() para mostrar el texto del error
   correspondiente */
void error (char * hilera )
{

  char texto [1024];

  if (sqlca.sqlcode<0)
  {
    printf ("Error en %s\n", hilera);
    sqlaintp (texto,1024,80,&sqlca);
    printf ("%s",texto);
    exit (-1);
  }
}
```

23.2). Modifíquese el programa ejemplo de tal manera que después de un SQLERROR se guarde la información del registro donde se produjo el error y se continúe el procesamiento de los demás registros.

Solución:

```
/************************************************
EJERCICIO: 23.2

ENUNCIADO: Modifíquese el programa ejemplo de tal
manera que después de un SQLERROR se guarde la
información del registro donde se produjo el error
```

```
y se continúe el procesamiento de los demás
registros.

************************************************/


#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/* Declaración de los ficheros de Entrada/Salida*/
FILE *inciden;
FILE *datos;
FILE *salida;

/* Funciones auxiliares */
void tratamiento_de_error (int*,int*);

/* Variables auxiliares */
int codigo;
char texto [70];

/* Definición del Area de Comunicaciones de SQL */
EXEC SQL INCLUDE SQLCA;

/* Dentro de la Declare Section se definen todas
   las variables necesarias para las sentencias SQL */
EXEC SQL BEGIN DECLARE SECTION;
  long numem;
  long numde;
  char fecna[11];
  long salar;
  short numhi;
  struct
  {  short longitud;
     char  datos[20];
  } nomem;
EXEC SQL END DECLARE SECTION;

/* En caso de AVISO de las sentencias SQL
   se continua */
EXEC SQL WHENEVER SQLWARNING CONTINUE;

/************************************************************
               PROGRAMA PRINCIPAL
************************************************************/
main( )
{

  /* Definición de variables auxiliares */
  int* error;
  int* existe;

  error=(int*) malloc(sizeof(int));
  existe=(int*) malloc(sizeof(int));

  *error=0;
  *existe=0;

  /* Conexión a la base de datos BDEJE */
  EXEC SQL CONNECT TO BDEJE;
```

```
 if (sqlca.sqlcode < 0)
 {
   printf ("Error en la conexión\n");
   exit (-1);
 }

 /* Apertura de los ficheros */
 inciden = fopen ("incidencias.txt","r");
 datos   = fopen ("datos.txt","r");
 salida  = fopen ("salida.txt","w");

 /* Lectura del primer registro del fichero de incidencias */
 fscanf(inciden,"%3d%1d", &numem, &codigo);

/* Con un bucle se recorre el fichero de incidencias
    hasta que se llegue al final del fichero */
 while (!feof (inciden))
 {

   /* Si el codigo es 1 borro el registro de la tabla TEMPLE */
   if (codigo == 1)
   {
     EXEC SQL DELETE FROM TEMPLE WHERE NUMEM=:numem;

     /* Llamada a la rutina tratamiento_de_error */
     tratamiento_de_error(error,existe);

     if (!(*error))
     {
     /* Actualizo la variable texto */
     if (!*existe) strcpy(texto,"No existe en la Tabla TEMPLE");
       else strcpy (texto,"causa baja en la empresa");
     }

   }

   /*Si no es baja tiene que ser alta en la empresa -> Insert */
   else
   {
     /* Lectura de los datos del fichero de datos */
     fscanf (datos, "%3d%3d%10s%4d%1d%2d%20s", &numem, &numde, fecna,
     &salar, &numhi, &nomem.longitud, nomem.datos);

     /* Llamada a la rutina tratamiento_de_error */
     EXEC SQL INSERT INTO TEMPLE
      (NUMEM,NUMDE,EXTEL,FECNA,FECIN,SALAR,COMIS,NUMHI,NOMEM)
      VALUES (:numem,:numde, 0, :fecna, CURRENT DATE, :salar, NULL,
      :numhi, :nomem);

     tratamiento_de_error(error,existe);

     if (!(*error))
     /* Actualización de la variable texto */
     strcpy(texto, "causa alta en la empresa");
   }

   if (!(*error))
   /* Escribo un registro en el fichero de Salida */
   fprintf (salida, "El empleado número %d %s%\n", numem ,texto);
```

```
    /* lectura del siguiente registro del fichero de incidencias*/
    fscanf(inciden,"%3d%1d", &numem, &codigo);

  } /* Fin del while */

  /* Cierre de los ficheros usados */
  fclose (inciden);
  fclose (datos);
  fclose (salida);

  /* Desconexión de la base de datos */
   EXEC SQL CONNECT RESET;

   if (sqlca.sqlcode < 0)
  {
    printf ("Error en la desconexión\n");
    exit (-1);
  }

  return 0;

} /* Fin del main */


/*************************************************************
               FUNCION TRATAMIENTO_DE_ERROR ()
*************************************************************/
void tratamiento_de_error (int* error, int* existe)
{

    /* Si hay un error escribo el registro y continuo */
    /* Si no hay error confirmo la operación insert o delete */
    if (sqlca.sqlcode < 0)
    {
     *error=1;
     if (sqlca.sqlcode == -803)
     {
       strcpy (texto,"EMPLEADO YA EXISTENTE -> SE CONTINUA");
       *existe=1;
     }
     else
     {
      strcpy(texto,"ERROR SQL AL PROCESAR EMPLEADO -> SE CONTINUA");
      existe=0;
     }

     fprintf  (salida, "Empleado %d %s\n", numem,texto);
     EXEC SQL ROLLBACK;
     if (sqlca.sqlcode < 0) printf ("Fallo en Rollback\n");
    }
    else
    {
      *error=0;
      *existe=0;
      EXEC SQL COMMIT;
      if (sqlca.sqlcode < 0) printf ("Fallo en Commit\n");
    }
}
```

23.3). Prepárese una rutina de tratamiento de errores que imprima toda la información contenida en el SQLCA.

Solución:

```
/************************************************
EJERCICIO: 23.3

ENUNCIADO: Prepárese una rutina de tratamiento de
errores que imprima toda la información contenida
en el SQLCA.

************************************************/

/* Inclusion del SQLCA */
EXEC SQL INCLUDE SQLCA;

tratamiento_de_errores ( )
{

  /* Impresión de la estructura SQLCA */

  printf ("sqlcaid: %.8s\n",sqlca.sqlcaid);
  printf ("sqlcabc: %d\n",sqlca.sqlcabc);
  printf ("sqlcode: %d\n",sqlca.sqlcode);
  printf ("sqlerrml: %d\n",sqlca.sqlerrml);
  printf ("sqlerrmc: %.*s%\n",sqlca.sqlerrml,sqlca.sqlerrmc);
  printf ("sqlerrp: %.8s\n",sqlca.sqlerrp);
  printf ("sqlerrd[0]: %d\n",sqlca.sqlerrd[0]);
  printf ("sqlerrd[1]: %d\n",sqlca.sqlerrd[1]);
  printf ("sqlerrd[2]: %d\n",sqlca.sqlerrd[2]);
  printf ("sqlerrd[3]: %d\n",sqlca.sqlerrd[3]);
  printf ("sqlerrd[4]: %d\n",sqlca.sqlerrd[4]);
  printf ("sqlerrd[5]: %d\n",sqlca.sqlerrd[5]);
  printf ("sqlwarn: %.11s\n",sqlca.sqlwarn);
  printf ("sqlstate: %.5s\n",sqlca.sqlstate);

}
```

Solucionario Capítulo 24

"PROGRAMACIÓN CON CURSORES"

24.1). Escribir un programa para listar todos los empleados de la tabla TEMPLE que trabajan en la calle de Atocha, con el siguiente formato:

```
                                        FECHA:&&-&&-&&&&

        LISTADO DE EMPLEADOS

  NOMBRE          DPT       ANT    SUELDO          COMENTARIOS
XXXXXXXXXXXX  999         999    99.999.99       XXXXXXXXXXXX
```

El listado debe estar en orden alfabético. DPT es el número de Departamento, ANT es la antigüedad en la empresa (en número de años), SUELDO es la suma del sueldo más comisiones, y comentarios indicará (CON COMISION) o (SIN COMISION).

Solución:

```
/************************************************
EJERCICIO: 24.1

ENUNCIADO: Escribir un programa para listar todos
los empleados de la tabla TEMPLE que trabajan en
la calle de Atocha, con el siguiente formato:

                                  FECHA:&&-&&-&&&&

      LISTADO DE EMPLEADOS

   NOMBRE       DPT       ANT      SUELDO        COMENTARIOS
XXXXXXXXXXXX    999       999     99.999.99     XXXXXXXXXXXX

El listado debe estar en orden alfabético. DPT es el
número de Departamento, ANT es la antigüedad en la
empresa (en número de años), SUELDO es la suma del
sueldo más comisiones, y comentarios indicará
(CON COMISION) o (SIN COMISION).

************************************************/


/* Librerías estándares de C */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/* Librería para el API sqlaintp( ) */
#include <sql.h>

/* Inclusión del SQL Communication Area */
EXEC SQL INCLUDE SQLCA;

/* Prototipo de la función error() */
void error (char *);
```

```
/*************************************************************
               PROGRAMA PRINCIPAL
*************************************************************/
void main() {

 /* Declaración de variables huéspedes */
 EXEC SQL BEGIN DECLARE SECTION;
   char  nomem [21] ;
   long numde;
   long anios;
   double salar;
   double comis;
   short ind_comis;
   char fecha [11];
 EXEC SQL END DECLARE SECTION;

 /* Conexión a la base de datos BDEJE */
 EXEC SQL CONNECT TO BDEJE;

 /* Tratamiento de errores */
 error ("Conexión");

 /* Almaceno la fecha actual en la variable fecha */
 EXEC SQL VALUES CURRENT DATE INTO :fecha;

 printf ("                                        FECHA:%s\n\n",fecha);
 printf ("                LISTADO DE EMPLEADOS\n\n");

 printf ("     NOMBRE        DPT   ANT    SUELDO   COMENTARIOS\n");

 /* Tratamiento de errores */
 error ("Values");

 /* Declaración de un cursor */
 EXEC SQL DECLARE c1 CURSOR FOR
  SELECT A.NOMEM, A.NUMDE, YEAR (CURRENT DATE) - YEAR(FECIN), A.SALAR,
  A.COMIS
   FROM TEMPLE A, TDEPTO B, TCENTR C
    WHERE A.NUMDE=B.NUMDE AND B.NUMCE=C.NUMCE
     AND SEÑAS LIKE '%ATOCHA%' ORDER BY NOMEM FOR READ ONLY;

 /* Apertura del cursor */
 EXEC SQL OPEN c1;

 /* Tratamiento de errores */
 error ("Apertura del cursor");

 do{

     /* Fetch de un registro de la tabla de resultados */
     EXEC SQL FETCH c1 INTO :nomem, :numde, :anios, :salar, :comis
     INDICATOR :ind_comis;

     /* Tratamiento de errores */
     error ("Fetch");

     /* Si se ha devuelto alguna fila se imprime por pantalla */
     if (sqlca.sqlcode!=100)
     {
      if (ind_comis<0)
```

```
       printf( "%-20s %d   %d   %3.2f   SIN
COMISION\n",nomem,numde,anios,salar);
      else
       printf( "%-20s %d   %d   %3.2f   CON
COMISION\n",nomem,numde,anios,salar+comis);
     }

 } while (sqlca.sqlcode!=100);

 /* Cierre del cursor */
 EXEC SQL CLOSE c1;

 /* Tratamiento de errores */
 error ("Cierre del Cursor");

 /* Desconexión de la base de datos */
 EXEC SQL CONNECT RESET;

 /* Tratamiento de errores */
 error ("Desconexión de la base de datos");

} /* Fin del main  */


/************************************************************
              FUNCION ERROR ()
************************************************************/
/* Recibe una hilera indicando donde se ha producido un error
   y hace uso del api sqlaintp() para mostrar el texto del error
   correspondiente */
void error (char * hilera )
{

  char texto [1024];

  if (sqlca.sqlcode<0)
  {
    printf ("Error en %s\n", hilera);
    sqlaintp (texto,1024,80,&sqlca);
    printf ("%s",texto);
    exit (-1);
  }
}
```

24.2). Escribir un programa que actualice la tabla de empleados TEMPLE sumando al salario una prima de 20 euros por hijo a aquellas personas que no tienen comisión, utilizando la cláusula FOR UPDATE OF.

Solución:

```
/************************************************
EJERCICIO: 24.2

ENUNCIADO: Escribir un programa que actualice la
tabla de empleados TEMPLE sumando al salario una
prima de 20 euros por hijo a aquellas personas
que no tienen comisión, utilizando la cláusula
FOR UPDATE OF.

************************************************/
```

```
/* Librerías estándares de C */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/* Librería para el API sqlaintp( ) */
#include <sql.h>

/* Inclusión del SQL Communication Area */
EXEC SQL INCLUDE SQLCA;

/* Prototipo de las función error() */
void error (char *);

/************************************************************
               PROGRAMA PRINCIPAL
************************************************************/
void main() {

 /* Declaración de variables huéspedes */
 EXEC SQL BEGIN DECLARE SECTION;
   char  nomem [21] ;
   short numhi;
 EXEC SQL END DECLARE SECTION;

 /* Conexión a la base de datos BDEJE */
 EXEC SQL CONNECT TO BDEJE;

 /* Tratamiento de errores */
 error ("Conexión");

 printf (" PRIMA POR HIJOS PARA LOS EMPLEADOS QUE NO TIENEN COMISIÓN\n");

 /* Declaración de un cursor */
 EXEC SQL DECLARE c1 CURSOR FOR
    SELECT NOMEM,NUMHI
      FROM TEMPLE
      WHERE COMIS IS NULL
      FOR UPDATE OF SALAR;

 /* Apertura del cursor */
 EXEC SQL OPEN c1;

 /* Tratamiento de errores */
 error ("Apertura del cursor");

 do{

     /* Fetch de un registro de la tabla de resultados */
     EXEC SQL FETCH c1 INTO :nomem, :numhi;

     /* Tratamiento de errores */
     error ("Fetch");

     /* Si se ha devuelto alguna fila se actualiza el salario y se
        imprime por pantalla */
     if (sqlca.sqlcode!=100)
     {
       EXEC SQL UPDATE TEMPLE SET salar=salar+(:numhi*20) WHERE CURRENT OF
C1;
```

```
       /* Tratamiento de errores */
       error ("Update");

       printf ("Al empleado %s se le ha incrementado el sueldo en %d
euros\n",nomem,numhi*20);
     }

 } while (sqlca.sqlcode!=100);

 /* Cierre del cursor */
 EXEC SQL CLOSE c1;

 /* Tratamiento de errores */
 error ("Cierre del Cursor");

 /* Desconexión de la base de datos */
 EXEC SQL CONNECT RESET;

 /* Tratamiento de errores */
 error ("Desconexión de la base de datos");

} /* Fin del main  */


/************************************************************
               FUNCION ERROR ()
************************************************************/
/* Recibe una hilera indicando donde se ha producido un error
   y hace uso del api sqlaintp() para mostrar el texto del error
   correspondiente */
void error (char * hilera )
{

  char texto [1024];

  if (sqlca.sqlcode<0)
  {
    printf ("Error en %s\n", hilera);
    sqlaintp (texto,1024,80,&sqlca);
    printf ("%s",texto);
    exit (-1);
  }
}
```

Solucionario Capítulo 25

"PREPARACIÓN DE PROGRAMAS"

25.1). Escribir un programa que cree a partir de la tabla TEMPLE otra con la misma estructura y que contenga todos los empleados del departamento 112.

Solución:

```
/************************************************
EJERCICIO: 25.1

ENUNCIADO: Escribir un programa que cree a partir
de la tabla TEMPLE otra con la misma estructura y
que contenga todos los empleados del departamento
112.

************************************************/

/* Librerías estándares de C */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/* Librería para el API sqlaintp( ) */
#include <sql.h>

/* Inclusión del SQL Communication Area */
EXEC SQL INCLUDE SQLCA;

/* Prototipo de la función error() */
void error (char *);

/************************************************************
               PROGRAMA PRINCIPAL
************************************************************/
void main() {

 /* Conexión a la base de datos BDEJE*/
 EXEC SQL CONNECT TO BDEJE;

 /* Tratamiento de errores */
 error ("Conexión");

 /* Creación de la tabla TEMPLE_112 con la misma
    estructura que la tabla TEMPLE */
 EXEC SQL CREATE TABLE TEMPLE_112 LIKE TEMPLE;

 /* Al existir un DDL (CREATE TABLE)  que posteriormente se utiliza
 en sentencias SQL es necesario Precompilar o Vincular el
 programa con el parámetro VALIDATE RUN. Esto es debido a que el
 optimizador del gestor al no existir la tabla no puede validarla y
 crear el plan de acceso hasta el momento de ejecución del programa
 (como ya se explicó en el capítulo de preparación de programas). */

 /* Tratamiento de errores */
 error ("Creación de tabla");

 /* Inserción en la tabla TEMPLE2 de los empleados
    pertenecientes al departamento 112 */
```

```
 EXEC SQL INSERT INTO TEMPLE_112
            SELECT * FROM TEMPLE
              WHERE NUMDE=112;

 /* Tratamiento de errores */
 error ("Select/insert");

 /* Desconexión de la base de datos */
 EXEC SQL CONNECT RESET;

 /* Tratamiento de errores */
 error ("Desconexión de la base de datos");

 printf ("Tabla TEMPLE_112 creada con los empleados del departamento
112\n");

} /* Fin del main  */


/************************************************************
               FUNCION ERROR ()
************************************************************/
/* Recibe una hilera indicando donde se ha producido un error
   y hace uso del api sqlaintp() para mostrar el texto del error
   correspondiente */
void error (char * hilera )
{

  char texto [1024];

  if (sqlca.sqlcode<0)
  {
    printf ("Error en %s\n", hilera);
    sqlaintp (texto,1024,80,&sqlca);
    printf ("%s",texto);
    exit (-1);
  }
}
```

25.2). Crear mediante un programa una vista que realice la misma función que el programa creado en el ejercicio anterior (25.1).

Solución:

```
/************************************************
EJERCICIO: 25.2

ENUNCIADO: Crear mediante un programa una vista que
realice la misma función que el programa creado en
el ejercicio anterior (25.1).

************************************************/


/* Librerías estándares de C */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/* Librería para el API sqlaintp( ) */
#include <sql.h>
```

```
/* Inclusión del SQL Communication Area */
EXEC SQL INCLUDE SQLCA;

/* Prototipo de la función error() */
void error (char *);

/************************************************************
               PROGRAMA PRINCIPAL
************************************************************/
void main() {

 /* Conexión a la base de datos BDEJE */
 EXEC SQL CONNECT TO BDEJE;

 /* Tratamiento de errores */
 error ("Conexión");

 /* Creación de la vista VISTA_TEMPLE con los
    empleados del departamento 112 */
 EXEC SQL CREATE VIEW VISTA_TEMPLE
            AS SELECT * FROM TEMPLE
                 WHERE NUMDE=112;

 /* Tratamiento de errores */
 error ("Creación de vista");

 /* Desconexión de la base de datos */
 EXEC SQL CONNECT RESET;

 /* Tratamiento de errores */
 error ("Desconexión de la base de datos");

 printf ("Vista VISTA_TEMPLE creada con los empleados del departamento
112\n");

} /* Fin del main  */


/************************************************************
               FUNCION ERROR ()
************************************************************/
/* Recibe una hilera indicando donde se ha producido un error
   y hace uso del api sqlaintp() para mostrar el texto del error
   correspondiente */
void error (char * hilera )
{

  char texto [1024];

  if (sqlca.sqlcode<0)
  {
    printf ("Error en %s\n", hilera);
    sqlaintp (texto,1024,80,&sqlca);
    printf ("%s",texto);
    exit (-1);
  }
}
```

Solucionario Capítulo 26

"PROGRAMACIÓN PARA CONCURRENCIA"

26.1). Modificar el programa del ejemplo al final del capítulo 23 incluyendo COMMIT/ROLLBACK.

Solución:

```
/************************************************
EJERCICIO: 26.1

ENUNCIADO: Modificar el programa del ejemplo al
final del capítulo 23 incluyendo COMMIT/ROLLBACK.

************************************************/


#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/* Declaración de los ficheros de Entrada/Salida*/
FILE *inciden;
FILE *datos;
FILE *salida;

/* Funciones auxiliares */
void tratamiento_de_error (void);

/* Variables auxiliares */
int codigo;
char texto [70];

/* Definición del Area de Comunicaciones de SQL */
EXEC SQL INCLUDE SQLCA;

/* Dentro de la Declare Section se definen todas
   las variables necesarias para las sentencias SQL */
EXEC SQL BEGIN DECLARE SECTION;
  long numem;
  long numde;
  char fecna[11];
  long salar;
  short numhi;
  struct
  {  short longitud;
     char  datos[20];
  } nomem;
EXEC SQL END DECLARE SECTION;

/* En caso de fallo de las sentencias SQL
   se saltará a la etiqueta TERROR */
EXEC SQL WHENEVER SQLERROR GOTO TERROR;

/* En caso de AVISO de las sentencias SQL
   se continua */
EXEC SQL WHENEVER SQLWARNING CONTINUE;


main( )
```

```
{

  /* Conexión a la base de datos BDEJE */
  EXEC SQL CONNECT TO BDEJE;

  /* Apertura de los ficheros */
  inciden = fopen ("incidencias.txt","r");
  datos   = fopen ("datos.txt","r");
  salida  = fopen ("salida.txt","w");

  /* Lectura del primer registro del fichero de incidencias */
  fscanf(inciden,"%3d%1d", &numem, &codigo);

 /* Con un bucle se recorre el fichero de incidencias
    hasta que se llegue al final del fichero */
  while (!feof (inciden))
  {

    /* Si el codigo es 1 borro el registro de la tabla TEMPLE */
    if (codigo == 1)
    {
      EXEC SQL DELETE FROM TEMPLE WHERE NUMEM=:numem;

      /* Actualizo la variable texto */
      if (SQLCODE == 100) strcpy(texto,"No existe en la Tabla TEMPLE");
        else strcpy (texto,"causa baja en la empresa");
    }

    /*Si no es baja tiene que ser alta en la empresa -> Insert */
    else
    {
      /* Lectura de los datos del fichero de datos */
      fscanf (datos, "%3d%3d%10s%4d%1d%2d%20s", &numem, &numde, fecna,
      &salar, &numhi,
                  &nomem.longitud, nomem.datos);

      EXEC SQL INSERT INTO TEMPLE
       (NUMEM,NUMDE,EXTEL,FECNA,FECIN,SALAR,COMIS,NUMHI,NOMEM)
       VALUES (:numem,:numde, 0, :fecna, CURRENT DATE, :salar, NULL,
       :numhi, :nomem);

      /* Actualización de la variable texto */
      strcpy(texto, "causa alta en la empresa");
    }

     /* Confirmación de la inserción o del borrado */
     EXEC SQL COMMIT;

    /* Escribo un registro en el fichero de Salida */
    fprintf (salida, "El empleado número %d %s%\n", numem ,texto);


    /* lectura del siguiente registro del fichero de incidencias*/
    fscanf(inciden,"%3d%1d", &numem, &codigo);

  } /* Fin del while */

  /* Cierre de los ficheros usados */
  fclose (inciden);
  fclose (datos);
  fclose (salida);
```

```
  /* Desconexión de la base de datos */
   EXEC SQL CONNECT RESET;

  return 0;

  TERROR:
    tratamiento_de_error ();

  return -1;

} /* Fin del main */


/* Rutina de Tratamiento de errores */
void tratamiento_de_error ( )
{


   /* En caso de fallo de las sentencias SQL
   se saltará a la etiqueta TERROR2 */
   EXEC SQL WHENEVER SQLERROR GOTO TERROR2;

  if (SQLCODE == -803)
    strcpy (texto,"EMPLEADO YA EXISTENTE -> FINALIZACION DEL PROCESO");
  else
    strcpy(texto,"ERROR SQL AL PROCESAR EMPLEADO -> FINALIZACION DEL
PROCESO");

  fprintf  (salida, "Empleado %d %s\n", numem,texto);

  /* Se deshace la transacción */
  EXEC SQL ROLLBACK;

  TERROR2:
   if (sqlca.sqlcode<0) printf ("Fallo en el Rollback\n");

  /* Cierre de los ficheros usados */
  fclose (inciden);
  fclose (datos);
  fclose (salida);
}
```

26.2). Modificar el programa del ejemplo al final del capítulo 24 incluyendo COMMIT/ROLLBACK.

Solución:

```
/************************************************
EJERCICIO: 26.2

ENUNCIADO: Modificar el programa del ejemplo al
final del capítulo 24 incluyendo COMMIT/ROLLBACK.

************************************************/

# include <stdio.h>
# include <stdlib.h>
# include <string.h>

/* Se incluye el Area de Comunicación de SQL */
```

```
EXEC SQL INCLUDE SQLCA;

/* Se declaran las variables huéspedes. */
/* No se definen variables indicadoras
   ya que ninguno de los 2 campos pueden
   contener nulos */
EXEC SQL BEGIN DECLARE SECTION;
  long salar_var;
  long numem_var;
EXEC SQL END DECLARE SECTION;

/* Se declara un cursor C1 con la clausual WITH HOLD
para que no se cierre el cursor al ejecutar un COMMIT */
EXEC SQL DECLARE C1 CURSOR WITH HOLD FOR
SELECT NUMEM,SALAR FROM TEMPLE
WHERE FECIN + 16 YEARS > CURRENT DATE
FOR UPDATE OF SALAR;

/* Se establecen las condiciones de error y aviso */
/* Estas sentencias rompen la programación estructurada
    (mediante el uso de sentencias GOTO). Puede ser
    preferible usar sentencias de C tipo if comprobando el contenido
    del sqlca.sqlcode */
EXEC SQL WHENEVER SQLERROR GO TO ERROR;
EXEC SQL WHENEVER SQLWARNING CONTINUE;
EXEC SQL WHENEVER NOT FOUND GO TO FIN;

void main () {

  /* Se establece una conexión a la base de datos BDEJE */
  EXEC SQL CONNECT TO BDEJE;

  /* Se abre el Cursor */
  EXEC SQL OPEN C1;

  /* Se establece un bucle para recorrer el cursor */
  while (sqlca.sqlcode==0)
  {

    EXEC SQL FETCH C1 INTO :numem_var, :salar_var;

    EXEC SQL UPDATE TEMPLE
      SET SALAR= :salar_var * 1.09
      WHERE CURRENT OF C1;

    /* Confirmación del UPDATE */
    EXEC SQL COMMIT;

    printf ("Actualizado el Empleado: %d \n",numem_var);
  }

  /* Etiquetas de error y fin de programa */
  ERROR:
    printf ("sqlcode: %d sqlerrmc: %s \n ",sqlca.sqlcode, sqlca.sqlerrmc);

    /* Se deshace la transacción */
    EXEC SQL ROLLBACK;

    exit (-1);

  FIN:
```

```
    EXEC SQL CLOSE C1;
    EXEC SQL CONNECT RESET;
} /* Fin del programa */
```

Solucionario Capítulo 27

"ACCESO A DATOS DISTRIBUIDOS"

27.1). Definir todos los pasos que se deben realizar para ejecutar una transacción que modifique tres gestores de bases de datos distintos.

Solución:

A continuación se especifican los pasos de la transacción, los pasos de la confirmación de la transacción (COMMIT) y se incluye un ejemplo de un programa en C que realiza una transacción distribuida que afecta a tres bases de datos distintas.

- Pasos de la transacción:

1. Antes de la conexión al primer gestor de bases de datos que interviene la aplicación cliente se conecta internamente al gestor de transacciones que devuelve una confirmación. Existe la posibilidad de que el gestor de transacciones sea el primer gestor de bases de datos al que la aplicación se conecta (esto es lo que ocurre en el ejemplo más abajo).

2. La aplicación se conecta al primer gestor de bases de datos que devuelve una confirmación.

3. La aplicación realiza la primera operación SQL que marca el comienzo de la unidad de trabajo. El gestor de transacciones provee un identificador para esta unidad de trabajo.

4. La aplicación registra el identificador, provisto en el paso anterior, en el primer gestor de bases de datos que interviene.

5. Las sentencias SQL se realizan contra el primer gestor de bases de datos que devuelve, para cada una información del resultado de su ejecución (correcta o no).

6. Sin terminar la unidad de trabajo se realiza una conexión contra el segundo gestor de bases de datos que devuelve una confirmación.

7. La aplicación registra el identificador, provisto en el paso 3, en el segundo gestor de bases de datos que interviene.

8. Las sentencias SQL se realizan contra el segundo gestor de bases de datos que devuelve, para cada una información del resultado de su ejecución (correcta o no).

9. Sin terminar la unidad de trabajo se realiza una conexión contra el tercer gestor de bases de datos que devuelve una confirmación.

10. La aplicación registra el identificador, provisto en el paso 3, en el tercer gestor de bases de datos que interviene.

11. Las sentencias SQL se realizan contra el tercer gestor de bases de datos que devuelve, para cada una información del resultado de su ejecución (correcta o no).

- Pasos para la confirmación de la transacción:

1. En este paso comienza la primera fase de la confirmación en 2 Fases. La aplicación cliente confirma la transacción mediante la ejecución de la sentencia de SQL commit. Al ejecutarse esta sentencia se manda a los gestores de bases de datos un mensaje de "preparados para confirmar" todos los gestores confirman la recepción de este mensaje.

2. La aplicación cliente, una vez recibidos los *ok* de cada gestor, envia el mismo mensaje de preparado al gestor de transacciones que responde con una confirmación.

3. En este paso comienza la segunda fase (de la confirmación en 2 fases ó two-phase commit en inglés). La aplicación cliente envia a los 3 gestores de base de datos que ejecuten la setencia commit. Los gestores la ejecutan y devuelven la confirmación al cliente.

4. Por ultimo el cliente, una vez recibida la confirmación de los gestores de base de datos, comunica al gestor de transacciónes que la unidad de trabajo ha finalizado.

- Programa C en SQL Embebido:

Para entender mejor estos pasos, a continuación se incluye un programa en SQL Embebido estático que ejecuta una transacción que afecta a tres bases de datos distintas. Las bases de datos se llaman BDatos1, BDatos2 y BDatos3. La transacción consiste en insertar un registro en las tablas T1, T2 y T3. T1 pertenece a BDatos1, T2 pertenece a BDatos2 y T3 a BDatos3. Las tablas tienen un único campo de tipo entero.

```
/* Librerías estándares de C */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/* Librería para el API sqlaintp( ) */
#include <sql.h>

/* Inclusión del SQL Communication Area */
EXEC SQL INCLUDE SQLCA;

/* Prototipo de la función error() */
void error (char *);

/*************************************************************
            PROGRAMA PRINCIPAL
*************************************************************/
void main() {

 /* Variables auxiliares */
 char a;

 /* Conexión a la base de datos. */
 EXEC SQL CONNECT TO  BDatos1;

 /* Tratamiento de errores. */
 error ("Conexión BDatos1");

 /* Inserción en Tabla T1. */
 EXEC SQL insert into T1 values (1);

 /* Tratamiento de errores. */
 error ("Inserción en T1");

 /* En este punto se puede comprobar desde otra sesión como se ha
 insertado (no confirmado todavia , leer en UR -> select * from T1 with ur)
 un registro en la base  de datos BDatos1. Se realiza una pausa por si se
quiere finalizar  el programa y comprobar que la inserción se deshace.*/
 printf ("Inserción realizada en Base de Datos BDatos1\n");
 printf ("Introduzca 's' para finalizar o cualquier otra tecla para continuar: ");
 fflush(stdin);
 a=getch();
 if (a=='s') exit(-1);

/* Conexión a la base de datos BDatos2. */
 EXEC SQL CONNECT TO BDatos2;

 /* Tratamiento de errores. */
 error ("Conexión BDatos2");
```

```
 /* Inserción en Tabla T2. */
 EXEC SQL insert into T2 values (2);

 /* Tratamiento de errores. */
 error ("Inserción en T2");

 /* En este punto se puede comprobar como se ha
 insertado (no confirmado todavía, leer en UR) un registro en la base
 de datos BDatos2. Se realiza una pausa por si se quiere finalizar
 el programa y comprobar que la inserción se deshace.*/
 printf ("\nInserción realizada en Base de Datos BDatos2\n");
 printf ("Introduzca 's' para finalizar  o cualquier otra tecla para continuar: ");
 fflush(stdin);
 a=getch();
 if (a=='s') exit(-1);

/* Conexión a la base de datos BDatos3. */
 EXEC SQL CONNECT TO BDatos3;

 /* Tratamiento de errores. */
 error ("Conexión BDatos3");

 /* Inserción en Tabla T3. */
 EXEC SQL insert into T3 values (3);

 /* Tratamiento de errores. */
 error ("Inserción en T3");

 /* En este punto se puede comprobar como se ha
 insertado (no confirmado todavía, leer en UR) un registro en la base
 de datos BDatos3. Se realiza una pausa por si se quiere finalizar
 el programa y comprobar que la inserción se deshace.*/
 printf ("\nInserción realizada en Base de Datos BDatos3\n");
 printf ("Introduzca 's' para finalizar  o cualquier otra tecla para continuar: ");
 fflush(stdin);
 a=getch();
 if (a=='s') exit(-1);

 /* Confirmación de la transacción distribuida */
 EXEC SQL COMMIT;

 /* Tratamiento de errores. */
 error ("COMMIT");

 /* Desconexión de la base de datos. */
 EXEC SQL CONNECT RESET;

 /* Tratamiento de errores. */
 error ("Desconexión de la base de datos");

 } /* Fin del main  */

/*************************************************************
            FUNCION ERROR ()
*************************************************************/
/* Recibe una hilera indicando donde se ha producido un error
   y hace uso del api sqlaintp() para mostrar el texto del error
   correspondiente */
void error (char * hilera )
```

```
{

  char texto [1024];

  if (sqlca.sqlcode<0)
  {
    printf ("Error en %s\n", hilera);
    sqlaintp (texto,1024,80,&sqlca);
    printf ("%s",texto);
    exit (-1);
  }
}
```

La preparación de este programa requiere especificar, en tiempo de precompilación, que se trata de una transacción distribuida. Si por ejemplo el programa anterior se llama 2phase.sqc y se utiliza el Visual C++ de Microsoft los pasos para generar el ejecutable (2phase.exe) son:

1) Precompilar con los parámetros: CONNECT 2 y SYNCPOINT TWOPHASE

```
db2 prep 2phase.sqc bindfile connect 2 syncpoint twophase
```

2) COMPILAR Y LINKAR NORMALMENTE

```
cl -c 2phase.c
link %2phase.obj db2api.lib
```

3) REALIZAR EL BIND SOBRE LA PRIMERA BASE DE DATOS

```
db2 connect to BDatos1
db2 bind 2phase.bnd validate run
```

Nota: El parámetro VALIDATE RUN es necesario ya que todas las tablas (T1, T2 y T3) no existen en todas las bases de datos.

4) REALIZAR EL BIND SOBRE LA SEGUNDA BASE DE DATOS

```
db2 connect to BDatos2
db2 bind 2phase.bnd validate run
```

5) REALIZAR EL BIND SOBRE LA TERCERA BASE DE DATOS

```
db2 connect to BDatos3
db2 bind 2phase.bnd validate run
```

Solucionario Capítulo 32

"TABLAS TEMPORALES GLOBALES"

32.1). Haciendo uso de las tablas temporales declaradas, se necesita comprobar si se desborda el presupuesto de algún departamento en el supuesto de duplicar el salario de los empleados que lo integran.

Solución:

Creamos una tabla temporal, T1, que tendría sumado por departamento el nuevo salario de los empleados que lo integran. Se alimentaría desde la tabla de Empleados original agrupando empleados por departamento y suma sus salarios multiplicados por dos. El tipo de datos original de salario se tiene que cambiar a DECIMAL(5,0) porque algún departamento podría pasar a tener sumas de salarios superiores a los 9.999 €.

Se crearía una segunda tabla temporal, T2, de departamentos con las siguientes columnas: número de departamento y presupuesto actual multiplicado por mil con el fin de que la cantidad figure también en euros. El tipo de dato del presupuesto se cambia a DECIMAL(6,0) con el fin de acomodar presupuestos en euros superiores a los 99.999 €.

Se llevaría a cabo una yunción de las dos tablas temporales por igual departamento con la condición de que el presupuesto de la tabla temporal declarada T2 sea menor que la columna de nuevos salarios acumulados de la tabla temporal T1.

El conjunto de sentencias SQL que formarían parte de la sesión es el siguiente:

```
DECLARE GLOBAL TEMPORARY TABLE SESSION.T1
(NUMDE INTEGER, NUEVOSALAR DEC(5,0))
NOT LOGGED ON COMMIT PRESERVE ROWS IN TS1;

DECLARE GLOBAL TEMPORARY TABLE SESSION.T2
(NUMDE INTEGER, PRESU DEC(6,0))
NOT LOGGED ON COMMIT PRESERVE ROWS IN TS1;

INSERT INTO SESSION.T1
        SELECT NUMDE, SUM(SALAR*2)
        FROM TEMPLE
        GROUP BY NUMDE;

INSERT INTO SESSION.T2
        SELECT NUMDE, PRESU*1000
        FROM TDEPTO;

SELECT A.NUMDE, PRESU, NUEVOSALAR
FROM SESSION T1 A, SESSION.T2 B
WHERE A.NUMDE=B.NUMDE AND PRESU < NUEVOSALAR;
```

Resultado:

NUMDE	PRESU	NUEVOSALAR
121	20000	24800
130	20000	22200

Solucionario Capítulo 34

"PROGRAMACIÓN DE PROCEDIMIENTOS ALMACENADOS"

34.1). Crear el procedimiento almacenado ACTUALIZAR_DPTO; éste se invoca desde el procedimiento almacenado ACTUALIZAR_SALARIO del apartado de Procedimientos SQL anidados.

Solución:

```
CREATE PROCEDURE ACTUALIZAR_DPTO
(IN DEPARTAMENTO INTEGER,
 IN INCREMENTO)
LANGUAGE SQL
BEGIN

UPDATE TDEPTO
        SET PRESU = PRESU + INCREMENTO
        WHERE NUMDE = DEPARTAMENTO;
END
@
```

34.2). Crear un procedimiento almacenado que devuelva el conjunto de empleados (nombre, número de empleado y presupuesto de su departamento) con más de 2 hijos, que no tengan ingresos por comisiones y correspondientes al departamento que se recibe en la llamada. El resultado se devuelve al cliente.

Solución:

```
CREATE PROCEDURE EMPL_MAS_DE_DOS_HIJOS
(IN DEPARTAMENTO INTEGER)
LANGUAGE SQL
DYNAMIC RESULT SETS 1
BEGIN
        DECLARE CUR1 WITH RETURN TO CLIENT FOR
                SELECT NOMEM, NUMEM, PRESU
                FROM TEMPLE A, TDEPTO B
                WHERE A.NUMDE=B.NUMDE
                AND A.NUMDE = DEPARTAMENTO
                AND NUMHI > 2
                AND COMIS IS NULL;
        OPEN CUR1;
END
@
```

.

Solucionario Capítulo 35

"PROGRAMACIÓN DE FUNCIONES DE USUARIO"

35.1). Escribir el programa para la función POSICION_HILERA definida anteriormente.

Solución:

```
/************************************************
EJERCICIO: 35.1

ENUNCIADO: Escribir el programa para la función
POSICION_HILERA definida anteriormente.

************************************************/


#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <sqludf.h>
#include <sqlca.h>
#include <sqlda.h>
#include <sqlsystm.h>


void SQL_API_FN fun2 (char* hilera1,
                      char* hilera2,
                      long * out,
                      /* Es necesario añadir variables indicadoras para cada
                      una de la varialbes de entrada o salida*/
                      short* hilera1_ind,
                      short* hilera2_ind,
                      short * out_ind,
                      SQLUDF_TRAIL_ARGS_ALL)
{

/* SQLUDF_TRAIL_ARGS_ALLL es la siguiente macro definida en el
 fichero sqludf.h que se encuentra en el directorio sqllib/include:
#define SQLUDF_TRAIL_ARGS_ALL char   sqludf_sqlstate[SQLUDF_SQLSTATE_LEN+1],   \
                              char   sqludf_fname[SQLUDF_FQNAME_LEN+1],        \
                              char   sqludf_fspecname[SQLUDF_SPECNAME_LEN+1], \
                              char   sqludf_msgtext[SQLUDF_MSGTEXT_LEN+1],     \
                              SQLUDF_SCRATCHPAD *sqludf_scratchpad,            \
                              SQLUDF_CALL_TYPE  *sqludf_call_type */


 int* area_scratch;
 char* hilera_aux;

 int longitud1;
 int longitud2;
 int longitud_aux;

 /* Inicialización de las variables  */
 longitud1=strlen (hilera1);
 longitud2=strlen (hilera2);

 /* La variable area_scratch se utiliza para guardar la información entre
 una llamada a la función y la siguiente. SQLUDF_SCRAT se define en el
 fichero sqludf.h que se encuentra en el directorio sqllib/include */
 area_scratch = ( int* ) ( SQLUDF_SCRAT->data );

 /* Una UDF que devuelve una tabla es llamada múltiples veces
```

```
por el gestor. En la primera llamada SQLUDF_CALLT tiene el
valor SQL_TF_OPEN, en las sucesivas llamadas tiene el valor
SQL_TF_FETCH, hasta que la función devuelve en el SQLUDF_STATE
el valor '02000' que significa que no hay más datos. Una vez
recibido este valor el gestor llama por última vez a la función
teniendo esta vez SQLUDF_CALLT el valor SQL_TF_CLOSE.
SQLUDF_CALLT es parte de SQLUDF_TRAIL_ARGS_ALL y en el fichero
sqludf.h se puede obtener su definición. */
switch( SQLUDF_CALLT )
{

 /* Primera llamada a la Apertura de la tabla */
 case SQL_TF_OPEN:
 *area_scratch=0;
 break ;

 /* LLamada normal a la UDF: Fetch de la siguiente fila*/
 case SQL_TF_FETCH:

  /* la función strstr devuelve un puntero a la
  primera ocurrencia de hilera2 en hilera1
  hilera1 es una dirección de memoria que apunta a hilera1
  y se le suma *area_scratch porque en esta variable se guarda
  desde donde debe comenzar la búsqueda de hilera2. La primera
  vez *area_scratch es 0 y las siguientes veces se utiliza para
  eliminar la parte de hilera1 donde ya se realizó la búsqueda.  */
  hilera_aux = strstr( hilera1+(*area_scratch),hilera2 );

  /* Si hilera_aux es nulo es que no hay más ocurrencias
  y se devuelve SQLSTATE='02000'.
  Si no es nulo devuelvo la posición donde se ha encontrado
  la ocurrencia de hilera2 en hilera1 y guardo en la variable
  area_scratch un entero que representa la posición donde tiene
  que comenzar la siguiente búsqueda.*/
  if (hilera_aux == NULL)
    strcpy( SQLUDF_STATE, "02000");
  else
  {
    longitud_aux=strlen (hilera_aux);
    *out=longitud1-longitud_aux+1;
    *area_scratch=longitud1-longitud_aux+longitud2;

  }

 break ;

 /* Ultima llamada a la UDF: Se cierra la tabla */
 case SQL_TF_CLOSE:
 *area_scratch=0;
 break ;


}

} /* fin de la función */
```

Solucionario Capítulo 36

"CONSTRICCIONES Y GATILLOS (TRIGGERS)"

36.1). Definir la Integridad Referencial entre nuestras tablas TDEPTO Y TCENTR.

Solución:

```
CREATE TABLE TDEPTO (
                NUMDE INTEGER NOT NULL ,
                NUMCE INTEGER ,
                DIREC INTEGER NOT NULL ,
                TIDIR CHAR(1) NOT NULL ,
                PRESU DECIMAL(3,0) NOT NULL ,
                DEPDE INTEGER ,
                NOMDE VARCHAR(20) ,
                PRIMARY KEY(NUMDE),
                CONSTRAINT DEPARTSM FOREIGN KEY (DEPDE)
                REFERENCES TDEPTO (NUMDE) ON DELETE NO ACTION,
                CONSTRAINT CENTRO FOREIGN KEY (NUMCE)
                REFERENCES TCENTR (NUMCE) ON DELETE RESTRICT);
CREATE TABLE TCENTR (
                NUMCE INTEGER NOT NULL ,
                NOMCE VARCHAR(25) NOT NULL ,
                SEÑAS VARCHAR(25) NOT NULL ,
                PRIMARY KEY(NUMCE));
```

36.2). ¿Qué columnas de la tabla de Departamentos son susceptibles de formar parte de claves ajenas?. ¿Respecto a qué tablas?.

Solución:

DIREC: clave ajena referenciando la columna NUMEM de la tabla de Empleados.
DEPDE: clave ajena referenciando la columna NUMDE de la propia tabla de Departamentos.
NUMCE: clave ajena referenciando la columna NUMCE de tabla de Centros.

36.3). ¿Qué regla de borrado sería más apropiada para aplicar entre las tablas de Departamentos y la de Empleados respecto a las columnas de DIREC y NUMEM respectivamente?.

Solución:

La opción de CASCADE no parece la más conveniente; si un empleado-director desaparece de la tabla de Empleados no tiene ningún sentido eliminar el departamento que dirigía. La opción RESTRICT puede dificultar el mantenimiento, al haber un ciclo de integridad referencial entre TDEPTO y TEMPLE. La opción SET NULL facilita el mantenimiento (necesita entonces que DIREC admita valores *Nulos*). Por tanto elegiríamos esta opción.

```
                CONSTRAINT DIRECTOR FOREIGN KEY (DIREC)
                REFERENCES TEMPLE (NUMEM) ON DELETE SET NULL
```

36.4). Crear la tabla de Departamentos con los siguientes controles: a) el presupuesto no debe superar los 150.000 euros, b) el tipo de director sólo puede valer 'P' o 'F'.

Solución:

```
CREATE TABLE TDEPTO
```

```
( NUMDE INTEGER NOT NULL,
  NUMCE INTEGER NOT NULL,
  DIREC INTEGER,
  TIDIR CHAR(1) NOT NULL CONSTRAINT TIDIRCHK CHECK (TIDIR IN ('P', 'F')),
  PRESU DEC(3,0) NOT NULL CONSTRAINT PRESUCHK CHECK (PRESU < 150),
  DEPDE INTEGER,
  NOMDE VARCHAR(20) NOT NULL)
```

36.5). Crear un gatillo para que después de actualizar la columna de número de hijos de la tabla de Empleados, se inserte una fila en la tabla auxiliar ya conocida indicando la razón del alta.

Solución:

```
CREATE TRIGGER GTEMPLEX1
   AFTER UPDATE OF NUMHI ON TEMPLE
   FOR EACH ROW MODE DB2SQL
   BEGIN ATOMIC
         INSERT INTO TABLAAUX
         VALUES( USER,  CURRENT TIMESTAMP,  'ACTUALIZAC. NUM. HIJOS' );
   END
```

36.6). Crear un gatillo para que antes de dar de alta una fila en la tabla de Departamentos compruebe que la suma de los presupuestos de los departamentos ya existentes no supere el millón de euros.

Solución:

```
CREATE TRIGGER GTEMPLEX2
    NO CASCADE BEFORE INSERT ON TDEPTO
    FOR EACH ROW MODE DB2SQL
    WHEN ((SELECT SUM(PRESU) FROM TDEPTO)> 1000)
    BEGIN ATOMIC
              SIGNAL SQLSTATE '70AX2'
              ('Suma total de presupuestos supera el millón de €');
END
```

36.7). Hacer el ejercicio anterior pero la comprobación de la suma de presupuestos debe incluir el del nuevo departamento que se está dando de alta.

Solución:

```
CREATE TRIGGER GTEMPLEX3
    AFTER INSERT ON TDEPTO
    FOR EACH ROW MODE DB2SQL
    WHEN ((SELECT SUM(PRESU) FROM TDEPTO)> 1000)
    BEGIN ATOMIC
              SIGNAL SQLSTATE '70AX3'
              ('Suma total de presupuestos supera el millón de €');
END
```

36.8). Se necesita crear un gatillo que asegure que cada centro que se dé de alta en la tabla de Centros sea mayor que los anteriores. Si no se cumpliera lo anterior, se debe dar un mensaje indicando el error.

Solución:

```
CREATE TRIGGER GTEMPLEX4
  NO CASCADE BEFORE INSERT ON TCENTR
  REFERENCING NEW AS NUEVA
  FOR EACH ROW MODE DB2SQL
  WHEN (NUEVA.NUMCE <= (SELECT MAX(NUMCE) FROM TCENTR))
  BEGIN ATOMIC
```

```
            SIGNAL SQLSTATE '70AX4'
           ('NUMCE debe ser mayor que los ya existentes');
  END
```

36.9). Supongamos que disponemos de una columna adicional en la tabla de Departamentos (TOTALPRESU) que va a contener información del total de los presupuestos de los departamentos existentes antes del alta del nuevo departamento. Si el valor de la suma es inferior a 500.000 €, se le asignará un valor a TOTALPRESU de 'BAJO'; si está entre 500.000 y 1.000.000 €, 'MEDIO'; si mayor de 1 millón de euros, 'ALTO'.

Solución:

```
CREATE TRIGGER GTEMPLEX5
 NO CASCADE BEFORE INSERT ON TDEPTO
 REFERENCING NEW AS NUEVA
 FOR EACH ROW MODE DB2SQL
 BEGIN ATOMIC
  DECLARE TOTAL DEC(5,0);
  SET TOTAL = (SELECT SUM(PRESU) FROM TDEPTO);
  SET NUEVA.TOTALPRESU =
  CASE
       WHEN
            TOTAL < 500 THEN 'BAJO'
       WHEN
            TOTAL BETWEEN 500 AND 1000 THEN 'MEDIO'
       WHEN
            TOTAL > 1000 THEN 'ALTO'
       END;
END
```

Solucionario Capítulo 37

"ORIENTACIÓN A OBJETOS"

37.1). Añadir a la jerarquía de EMPLEADOS y VENDEDORES del capítulo un elemento más que corresponda a los directores. Crear un tipo estructurado que incluya como atributo adicional a los de empleados el bono de compañía.

Solución:

```
CREATE TYPE DIRE_T UNDER EMPLE_T AS
                    (BONO DECIMAL(4,0))
MODE DB2SQL;
```

37.2). Crear una tabla con tipo para albergar a los directores.

Solución:

```
CREATE TABLE DIRECTOR OF DIRE_T UNDER EMPLE
INHERIT SELECT PRIVILEGES;
```

37.3). Dar de alta en la tabla anterior al empleado 150, director del departamento 121, asignándoles identificador de objeto 3 y un bono de 3000€.

Solución:

```
INSERT INTO DIRECTOR
(IDE, NUMEM, NUMDE, EXTEL, FECNA, FECIN, SALAR, NUMHI, NOMEM, BONO)
VALUES
(DIRE_T(4), 150, 121, 340, '10/08/1930', '15/01/1948', 4400, 0, 'PEREZ, JULIO',3000);
```

37.4). Obtener los números de empleado de aquellos empleados que sean vendedores o directores.

Solución:

```
SELECT NUMEM FROM EMPLE
WHERE DEREF(IDE) IS OF DYNAMIC TYPE (VENDE_T, DIRE_T);
```

Resultado:

```
NUMEM
-----------
        600
        150
```

37.5). Crear una tabla de Sectores, SECTOR, para asignar a cada director la responsabilidad de negocio de un sector de clientes determinado (Sector Público, Pequeña y Mediana Empresa, Sector Banca, etc). La tabla se creará alrededor de un tipo estructurado SECTOR_T que contendrá los atributos NUMSE y NOMSE. Dar de alta el sector 1, "BANCA".

Solución:

En primer lugar crearemos el tipo estructurado de datos:

```
CREATE TYPE SECTOR_T AS
        (NUMSE INTEGER,
        NOMSE  VARCHAR(30))
        REF USING INTEGER
```

```
MODE DB2SQL;
```

A continuación crearemos la tabla SECTOR:

```
CREATE TABLE SECTOR OF SECTOR_T
(REF IS IDENSE USER GENERATED);
```

Y daremos de alta el sector 1:

```
INSERT INTO SECTOR (IDENSE, NUMSE, NOMSE)
        VALUES (SECTOR_T(1), 1, 'BANCA');
```

37.6). Modificar la definición de la tabla de Directores para incluir una referencia a la tabla de Sectores.

Solución:

En primer lugar borraremos la tabla de Directores, alteraremos la definición del tipo estructurado DIRE_T para incluirle la referencia a la tabla de Sectores, volviendo a crear posteriormente la tabla de Directores:

```
DROP TABLE DIRECTOR;

ALTER TYPE DIRE_T
ADD ATTRIBUTE SECTOR_REF REF(SECTOR_T);

CREATE TABLE DIRECTOR OF DIRE_T UNDER EMPLE
        INHERIT SELECT PRIVILEGES
        (SECTOR_REF WITH OPTIONS SCOPE SECTOR);
```

Daremos de alta de nuevo al empleado 150, director del departamento 121:

```
INSERT INTO DIRECTOR
(IDE, NUMEM, NUMDE, EXTEL, FECNA, FECIN, SALAR, NUMHI, NOMEM, BONO,
SECTOR_REF)
VALUES
(DIRE_T(4), 150, 121, 340, '10/08/1930', '15/01/1948', 4400, 0, 'PEREZ, JULIO',3000,
SECTOR_T(1));
```

37.7). Obtengamos los nombres de los directores del sector 1 de clientes. Llevar a cabo la misma consulta haciendo uso del operador de desreferencia.

Solución:

```
SELECT NOMEM
FROM DIRECTOR, SECTOR
        WHERE NUMSE= 1 AND SECTOR_REF = IDENSE;
```

Resultado:

```
NOMEM
--------------------
PEREZ, JULIO
```

Solución:

```
SELECT NOMEM
FROM DIRECTOR
WHERE SECTOR_REF -> NUMSE  =  1;
```

37.8). Para la tabla de directores añadir la columna NOMEM que corresponda a un tipo estructurado que contenga dos atributos, el nombre y el apellido, ambos VARCHAR(15). Se debe mantener el mismo empleado existente con anterioridad en la tabla, informando en el alta la nueva columna.

Solución:

En primer lugar se creará el tipo estructurado que irá a la nueva columna de la tabla.

```
CREATE TYPE NOMEM_T AS
        (NOMBRE VARCHAR(15),
        APELLIDO VARCHAR(15))
MODE DB2SQL;
```

Antes de llevar a cabo el cambio de definición del tipo estructurado ya existente, DIRE_T, se tiene que borrar la tabla DIRECTOR. A continuación.

```
ALTER TYPE DIRE_T ADD ATTRIBUTE NOMEM1 NOMEM_T;
```

Recreándose de nuevo la tabla.

Demos de alta de nuevo el empleado 150:

```
INSERT INTO DIRECTOR
(IDE, NUMEM, NUMDE, EXTEL, FECNA, FECIN, SALAR, NUMHI, NOMEM, BONO,
SECTOR_REF, NOMEM1)
VALUES
(DIRE_T(4), 150, 121, 340, '10/08/1930', '15/01/1948', 4400, 0, 'PEREZ, JULIO',3000,
SECTOR_T(1), NOMEM_T( )..NOMBRE('JULIO')..APELLIDO('PEREZ'));
```

Solucionario Capítulo 38

"DISEÑO Y DESARROLLO CON SQL"

38.1). Actualizar un fichero maestro con un fichero de movimientos. Ésta es una aplicación que casi todo programador ha tenido que afrontar una o más veces en su vida.

Se trata de actualizar un fichero maestro de pedidos con las entregas realizadas. Supongamos que la información que trata la aplicación está almacenada en las siguientes tablas y columnas:

```
PEDIDO:  PED_ID (CP)   * identificador del pedido *
       CLI_ID (CA)        * identificador del cliente *
          PED_TIPO              * tipo de pedido *

        LINEAP:  LPD_ID (CP)   * identificador linea pedido *
                 PED_ID (CA)        * identificador pedido  *
                 PZA_ID (CA)   * identificador pieza pedida *
         LPD_CANT              * cantidad pedida *
                 LPD_SERV      * cantidad servida a la fecha *

        ENTREGA:  LPD_ID (CP,CA)* identificador linea pedido *
                  ENT_CANT     * cantidad servida *
           ENT_FECHA                 * fecha de entrega *
```

La tabla PEDIDO contiene los pedidos existentes pendientes de suministro. Por cada una de las distintas piezas solicitadas en un pedido hay una línea de pedido en LINEAP. Cada entrega realizada de una pieza de un pedido es una fila de ENTREGA, debe existir por tanto una fila correspondiente en LINEAP.

A continuación de determinadas columnas se ha indicado que son clave primaria (CP) o clave ajena (CA). En LINEAP la clave primaria es LPD_ID y son claves ajenas PED_ID y PZA_ID. LPD_ID es una clave abstracta, es decir un identificador sin contenido semántico adicional. Otra posibilidad es designar como clave primaria a la clave alternativa compuesta por las columnas PED_ID y PZA_ID.

Solución:

```
/*************************************************
EJERCICIO: 38.1

ENUNCIADO: Actualizar un fichero maestro con un
fichero de movimientos.  Esta es una aplicación que
casi todo programador ha tenido que afrontar una o
más veces en su vida.

Se trata de actualizar un fichero maestro de pedidos
con las entregas realizadas.  Supongamos que la
información que trata la aplicación está almacenada
 en las siguientes tablas y columnas:

PEDIDO:   PED_ID (CP)    * identificador del pedido *
          CLI_ID (CA)    * identificador del cliente *
              PED_TIPO       * tipo de pedido *

      LINEAP:   LPD_ID (CP)    * identificador linea pedido *
                   PED_ID (CA)    * identificador pedido  *
             PZA_ID (CA)    * identificador pieza pedida *
             LPD_CANT       * cantidad pedida *
             LPD_SERV       * cantidad servida a la fecha *

      ENTREGA:   LPD_ID (CP,CA)* identificador linea pedido *
              ENT_CANT       * cantidad servida *
                   ENT_FECHA      * fecha de entrega *
```

```
La tabla PEDIDO contiene los pedidos existentes pendientes de
suministro.  Por cada una de las distintas piezas solicitadas
en un pedido hay una línea de pedido en LINEAP.  Cada entrega
realizada de una pieza de un pedido es una fila de ENTREGA,
debe existir por tanto una fila correspondiente en LINEAP.

A continuación de determinadas columnas se ha indicado que
son clave primaria (CP) o clave ajena (CA).  En LINEAP la
clave primaria es LPD_ID y son claves ajenas PED_ID y PZA_ID.
LPD_ID es una clave abstracta, es decir un identificador sin
contenido semántico adicional.  Otra posibilidad es designar
como clave primaria a la clave alternativa compuesta por las
columnas PED_ID y PZA_ID.
```

Abordaremos el problema con una actualización en modo secuencial. Para ello se desea escribir un programa que lee ENTREGA y actualiza aquellas líneas de pedido para las que ha habido una entrega.

```
************************************************/


/* Inclusión de las librerías de C estándar */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/* Librería para el API sqlaintp */
#include <sql.h>

/* Inclusión del Area de Comunicacion de SQL */
EXEC SQL INCLUDE SQLCA;

/* Declaración de la función para la gestión de errores */
void tratamiento_errores (char *);

/*************************************************************
               PROGRAMA PRINCIPAL
*************************************************************/
void main() {

   /* Definición de variables auxiliares */
   int fin=0;

   /* Declaración de la variables huéspedes */
   EXEC SQL BEGIN DECLARE SECTION;
     long lpd_id;
     long ent_cant;
     long lpd_serv;
     long lpd_cant;
   EXEC SQL END DECLARE SECTION;

   /* Conexión a la base de datos BDEJE */
   EXEC SQL CONNECT TO BDEJE;

   /* Tratamiento de errores */
   tratamiento_errores ("Conexión");

   /* Declaración del cursor */
   EXEC SQL DECLARE c1 CURSOR FOR
     SELECT LPD_ID,ENT_CANT FROM ENTREGA
       WHERE ENT_FECHA=CURRENT DATE;
```

```
   /* Apertura del cursor */
   EXEC SQL OPEN c1;

   /* Tratamiento de errores */
   tratamiento_errores ("Apertura del Cursor");

   /* Se establece un bucle para leer las filas */
   while (!fin)
   {

     /* Se lee una linea entregada */
     EXEC SQL FETCH c1 INTO :lpd_id, :ent_cant;

     /* Tratamiento de errores */
     tratamiento_errores ("Fetch");

     if (sqlca.sqlcode==100) fin=1;
     else
     {
      /* Se actualiza la tabla que contiene las
      las lineas de pedido. Se suma a la cantidad
      servida a la fecha la cantidad servida hoy.*/
      EXEC SQL UPDATE LINEAP SET LPD_SERV=LPD_SERV + :ent_cant
        WHERE LPD_ID= :lpd_id;

      /* Tratamiento de errores */
      tratamiento_errores ("Actualización");

      /* Comprobación que la linea de pedido que se ha entregado
      existe en la tabla LINEAP */
      if (sqlca.sqlcode==100)
       printf ("Error1: No existe el identificador %d en la tabla
LINEAP\n",lpd_id);
      else
      {
        EXEC SQL SELECT LPD_SERV,LPD_CANT INTO :lpd_serv, :lpd_cant
          FROM LINEAP WHERE LPD_ID= :lpd_id;

        /* Tratamiento de errores */
        tratamiento_errores ("Select into");

        /* Comprobación que la cantidad servida no supera la cantidad
        pedida */
        if (lpd_serv > lpd_cant)
          printf ("Error2: Linea de pedido %d cantidad servida superior a
cantidad pedida\n",lpd_id);
        else printf ("Linea de pedido %d actualizada
correctamente\n",lpd_id);
      }
     } /* Fin del else */
   } /* Fin del while */

   /* Cierre del cursor */
   EXEC SQL CLOSE c1;

   /* Tratamiento de errores */
   tratamiento_errores ("Cierre del cursor");

   /* Desconexión de la base de datos*/
   EXEC SQL CONNECT RESET;
```

```
   /* Tratamiento de errores */
   tratamiento_errores ("Desconexión");

} /* Fin del main */


/************************************************************
               FUNCION TRATAMIENTO_ERRORES ()
************************************************************/
/* Rutina para el tratamiento de los errores que
   recibe un mensaje que muestra por pantalla
   y mediante el API sqlaintp() muestra el
   mensaje en caso de error                  */

void tratamiento_errores (char *mensaje)
{
  char texto [1024];

  if (sqlca.sqlcode<0)
  {
    printf ("Error en %s\n",mensaje);
    sqlaintp (texto,1024,80,&sqlca);
    printf ("%s",texto);
    exit (-1);
  }
}
```

38.2). Resolver el mismo problema anterior, pero sin desarrollar programas, es decir, utilizando solamente sentencias SQL. Analizar ventajas e inconvenientes de esta solución.

Solución:

En este caso, el programa no requiere lógica y se puede expresar exclusivamente como una serie de sentencias SQL, que podría formar un procedimiento ejecutable interactivamente desde un terminal. La idea es explotar más a fondo la capacidad del SQL de procesar conjuntos de filas.

Como **primera opción** podríamos recurrir a la creación de una tabla a través de la sentencia CREATE TABLE con la opción LIKE. Esta opción permite crear una tabla fija (en contraposición a la tabla temporal) tomando otra como modelo. La nueva tabla queda entonces definida con las mismas columnas que la que sirve de modelo, y cuyo nombre se especifica detrás de la palabra LIKE.

```
CREATE TABLE TEMPORAL LIKE LINEAP;
COMMIT;
INSERT INTO TEMPORAL
        SELECT A.LPD_ID, A.PED_ID, A.PZA_ID, A.LPD_CANT,
                A.LPD_SERV + B.ENT_CANT
        FROM LINEAP A, ENTREGA B
        WHERE A.LPD_ID = B.LPD_ID
        AND ENT_FECHA = CURRENT DATE
         AND A.LPD_CANT >= A.LPD_SERV + B.ENT_CANT;
DELETE FROM LINEAP
        WHERE LPD_ID IN (SELECT LPD_ID FROM TEMPORAL);
INSERT INTO LINEAP
        SELECT * FROM TEMPORAL;
DROP TABLE TEMPORAL;
COMMIT WORK;
```

Como puede verse, el procedimiento consiste en:

1) Crear una tabla TEMPORAL semejante (LIKE) a LINEAP.
2) Añadir a TEMPORAL las líneas de pedido para las que hay una entrega a procesar. Además, aprovecha la operación para sumar a LPD_SERV la cantidad entregada. Obsérvese la condición de que la suma de la cantidad ya servida (LPD_SERV) más la cantidad de la entrega (ENT_CANT), no pueden superar la cantidad a servir en la línea del pedido (LPD_CANT).
3) Borrar de LINEAP aquellas filas para las que existe una fila con el mismo LPD_ID en TEMPORAL, es decir, las que se van a modificar.
4) Finalmente añadir a LINEAP las líneas de pedido modificadas por las entregas y destruir la tabla temporal.

Como se recordará de un capítulo anterior, se pueden definir tablas temporales cuya existencia se limita a la duración de la sesión de trabajo donde se crean. Veamos esta **segunda opción** que sólo se diferencia de la primera en el uso de tabla temporal declarada frente al uso de una tabla fija:

```
DECLARE GLOBAL TEMPORARY TABLE SESSION.TEMPORAL LIKE LINEAP WITH
REPLACE ON COMMIT PRESERVE ROWS NOT LOGGED;
        INSERT INTO SESSION.TEMPORAL
                SELECT A.LPD_ID, A.PED_ID, A.PZA_ID, A.LPD_CANT,
                        A.LPD_SERV + B.ENT_CANT
                FROM LINEAP A, ENTREGA B
                WHERE A.LPD_ID = B.LPD_ID
                AND ENT_FECHA = CURRENT DATE
         AND A.LPD_CANT >= A.LPD_SERV + B.ENT_CANT;
DELETE FROM LINEAP
        WHERE LPD_ID IN (SELECT LPD_ID FROM SESSION.TEMPORAL);
INSERT INTO LINEAP
        SELECT * FROM SESSION.TEMPORAL;
COMMIT WORK;
```

Como puede verse la creación de la tabla temporal declarada también se ha hecho en este caso semejante (LIKE) a LINEAP. La definición de la tabla se ha hecho para que las filas se mantengan en ésta después de ejecutar una sentencia COMMIT. Después del COMMIT WORK ejecutado al final del conjunto anterior de sentencias SQL, la tabla temporal declarada desaparecerá cuando se produzca la desconexión de la base de datos, y acabe la sesión de trabajo con el gestor.

Hay todavía una **tercera opción** que consiste en llevar a cabo todo el proceso visto en las opciones anteriores, pero en una sola sentencia SQL. La potencia del lenguaje SQL nos permite ejecutar una sentencia SQL de UPDATE como la que sigue:

```
UPDATE LINEAP A SET LPD_SERV = LPD_SERV +
        COALESCE((SELECT ENT_CANT FROM ENTREGA B
                        WHERE B.LPD_ID = A.LPD_ID
                        AND B.ENT_FECHA = CURRENT DATE),0)
WHERE LPD_CANT > = LPD_SERV +
         COALESCE((SELECT ENT_CANT FROM ENTREGA B
                        WHERE B.LPD_ID = A.LPD_ID
                        AND B.ENT_FECHA = CURRENT DATE),0)
```

Podemos especificar un SELECT COMPUESTO correlacionado con la sentencia de UPDATE; este select sólo devuelve una fila cuando existe una fila correlacionada con la que se pretende modificar, de lo contrario el valor devuelto es *Nulo*. Lo anterior nos obliga a especificar el valor 0 a través de la función COALESCE para estos casos (líneas de pedido para las que no hay una entrega). La única condición a especificar en la cláusula WHERE es la relacionada con la suma de cantidad servida en ENTREGA frente a la cantidad necesaria y a la ya existente en LINEAP. Por la razón anteriormente explicada, vuelve a ser necesario hacer uso de la función COALESCE.

Las distintas opciones explotan a fondo la potencia del SQL y de un gestor de base de datos relacional, suprimiendo la necesidad de programación convencional. Es muy importante resaltar que requieren que los datos

no tengan errores, pues en caso contrario no se pueden detectar situaciones como la de una entrega sin la línea de pedido correspondiente.

Los registros donde se produzca la situación ya comentada en la que la cantidad de la entrega más lo ya servido supere la cantidad necesitada de la pieza, no son procesados y no se detectan. Con programación convencional, y conforme se van procesando las filas de las dos tablas, se podría evaluar una situación como la anterior y emitir un mensaje de error. Para el caso de estas tres opciones anteriores, nos veríamos obligados a llevar a cabo operaciones SQL adicionales específicas que nos permitieran detectar este extremo.

Por último, estas opciones que no usan la programación convencional no permiten que, dentro del proceso completo de tratamiento de todas las entregas, se puedan tomar puntos de control (COMMITs) intermedios, es decir cada un cierto número de registros procesados. Por el contrario, un error a mitad de proceso, desharía todos los cambios hechos, es decir todas las entregas procesadas hasta ese momento. En consecuencia, no parece recomendable usar estas técnicas cuando se manejen volúmenes grandes de datos. En contrapartida, la formulación y prueba de estos procedimientos resultan sencillas.