ÍNDICE

1. Introdución

Esta unidade ten como finalidade aprender a modificar información de calquera táboa dunha base de datos relacional usando a linguaxe SQL.

Usaremos o SXBD (Sistema Xestor de Base de Datos) SQL Server, polo que a sintaxe que se explicará será a da linguaxe Transact-SQL (T-SQL en adiante), propia do xestor.

A ferramenta gráfica que imos empregar será o SQL Server Management Studio (SSMS en adiante), tanto o editor de consultas como o asistente.

Antes de empezar estableceremos as convencións de escritura para as distintas sintaxes.

Convencións de escritura da sintaxe SQL		
Cláusulas	En maiúsculas	`SELECT, FROM, WHERE, HAVING...`
Predicados	En maiúsculas	`NOT IN, BETWEEN, ANY, SOME...`
Funcións nas consultas	En minúsculas	`isnull, cast...`
Funcións no texto	En cursiva	*`isnull, cast...`*
Palabra reservada para os alias	En minúsculas	`as`
Outras palabras reservadas	En maiúsculas	`ASC, DESC, TOP, DISTINCT...`
Nomes das bases de datos	En maiúsculas	`EMPRESA, SOCIEDADE_CULTURAL...`
Nomes das táboas	En maiúsculas	`ACTIVIDADE, AULA, EMPREGADO...`
Nomes das columnas nas consultas	En minúsculas	`descricion, nome, data_ini,...`
Nomes das columnas no texto	En minúsculas e cursiva	*`descricion, nome, data_ini,...`*
Constantes de texto	En maiúscula	`ADMINISTRATIVO, NON DIRIXE...`
[]	O contido dos corchetes é opcional.	
{}	Entre chaves colócanse listas de valores ou expresións.	
|	Separa distintas opcións a escoller.	

Bases de datos de traballo

Para seleccionar a BD de traballo como por exemplo a BD SOCIEDADE_CULTURAL, bastará con facer unha soa instrución USE:

```
--Seleccionamos a BD SOCIEDADE_CULTURAL
USE SOCIEDADE_CULTURAL;
```

- **Resultado:** Comprobaremos como despois de executar a sentenza, na lista despregable do SSMS aparece o nome da BD SOCIEDADE_CULTURAL.

2. Creación de táboas novas a partires de consultas (SELECT...INTO)

A instrución SELECT ... INTO crea unha nova táboa no grupo de arquivos predeterminado da BD e engade as filas resultantes da consulta nesa nova táboa.

A táboa resultante non terá ningunha das claves nin primarias, nin foráneas, nin índices, nin as restricións, das táboas orixinais.

As táboas xeradas con esta sentenza poden ser:

- **permanentes**,
- **temporais locais**, só visibles na sesión actual, ou,
- **temporais globais**, visibles en todas as sesións conectadas á BD onde se crea a táboa.

Sintaxe sentenza SELECT...INTO	

`SELECT {lista_de_campos|*}`
`INTO [#][#] nome_táboa_nova`
`FROM ...;`
	▪ **SELECT...FROM:** A cláusula SELECT será unha consulta de recuperación de información coa que se desexa crear unha nova táboa, tanto no que se refire a campos como a filas, é dicir a estrutura e contido.
	▪ **INTO [#][#]nome_táboa_nova:** Despois de INTO temos que indicar o nome da táboa a crear. Se antes do nome poñemos un só **símbolo #** a táboa nova será **temporal local**. Se poñemos antes do nome 2 cancelos **## será temporal global**.
	▪ **Nomes dos campos:** Para que os campos da táboa nova teñan diferentes nomes que os das táboas orixinais, deberanse usar alias nas columnas do SELECT.

Se antes do nome (sen deixar espazos) poñemos o símbolo # (cancelo), estaremos a crear unha táboa temporal local. Se poñemos antes do nome 2 cancelos ## crearemos unha táboa temporal global. Os cancelos formarán parte do nome, de tal xeito que para referirnos a elas indicaremos sempre un ou os dous cancelos diante.

A continuación realizaranse, executaranse e comprobaranse os resultados das consultas de exemplo 1 e 2, no editor de consultas do SSMS.

- **Consulta de exemplo 1:** Crearase unha táboa permanente de nome COTA_DE_BALDE cos mesmos campos que COTA pero só coas filas das cotas gratis. Os campos da nova táboa terán por nomes *cod*, *descricion* e *importe*. Ao rematar eliminarase a táboa nova coa cláusula DROP TABLE para non modificar o deseño orixinal da BD.

```
--Consultamos o contido da táboa COTA
SELECT * FROM COTA;
--Creamos a táboa nova COTA_DE_BALDE
SELECT codigo as cod,
       nome as descricion, importe INTO COTA_DE_BALDE
FROM COTA
WHERE importe=0;
--Consultamos o contido da táboa COTA_DE_BALDE
SELECT * FROM COTA_DE_BALDE;
--Eliminamos a táboa nova para manter o deseño orixinal da BD
DROP TABLE COTA_DE_BALDE;
```

Táboa COTA		
codigo	**nome**	**importe**
11 12 13 99	DE HONRA FAMILIAR HABITUAL GRATUITA	100.00 30.00 50.00 0.00
Táboa COTA_DE_BALDE		
cod	**descricion**	**importe**
99	GRATUITA	0.00

- **Consulta de exemplo 2:** Crearase unha táboa temporal local de nome SOCIO_MOROSO co nif e nome completo dos socios que deben algunha actividade. Ao rematar eliminarase a táboa nova coa cláusula DROP TABLE para non modificar o deseño orixinal da BD.

```
SELECT DISTINCT nif,
       rtrim(ape1+' '+isnull(ape2,''))+', '+nome as nome_completo
       INTO #SOCIO_MOROSO
FROM SOCIO s INNER JOIN SOCIO_REALIZA_ACTI sr
     ON s.numero=sr.num_socio
WHERE sr.pagada='N';
--Consultamos o contido da táboa #SOCIO_MOROSO
SELECT * FROM #SOCIO_MOROSO;
--Eliminamos a táboa nova para manter o deseño orixinal da BD
DROP TABLE #SOCIO_MOROSO;
```

Táboa #SOCIO_MOROSO	
nif	**nome_completo**
22222223B 33333334C	SIEIRO CAMPOS, MANUEL DEL CARMEN LÉREZ, JORGE

3. Inserción de filas novas (INSERT)

Para engadir filas nunha táboa utilízase a instrución INSERT. As inserciónss de filas poden facerse dunha en unha indicando os valores de cada columna, ou engadindo filas que son o resultado dunha consulta SELECT.

A continuación amósanse as dúas sintaxes.

Sintaxe1 con VALUES	

`INSERT INTO táboa [(col1,...,colN)]`
`VALUES (valor1,...,valorN)`
`[,(valor1,...,valorN),...];`
	▪ **táboa:** Nome da táboa na que se desexa engadir información nova.
	▪ **[(col1,...,colN)]:** A continuación do nome da táboa, é recomendable, aínda que non obrigatorio, indicar entre parénteses os nomes das columnas nas que se van engadir valores. Fai a consulta máis lexible e así non será necesario lembrar a orde dos campos na táboa cando se lle asignen os valores na cláusula VALUES. Se algunha columna da táboa non se especifica na lista gardarase o valor por defecto, ou NULL no caso de que o campo admita nulos.
	▪ **VALUES (valor1,...,valorN):** Especifica o valor a gardar en cada campo da táboa. Poden engadirse varias filas á vez se poñemos varias listas de valores entre parénteses, separadas por comas na cláusula.Por exemplo: VALUES(ListaColFilaNova1), (ListaColFilaNova2), (ListaColFilaNova2);

Sintaxe2 con SELECT	
`INSERT INTO táboa [(col1,...,colN)]` `SUBSELECT;`	▪ **táboa:** Nome da táboa na que se desexa engadir información nova. ▪ **[(col1,...,colN)]:** *Ver explicación na sintaxe1*. ▪ **SUBSELECT:** Consulta de recuperación de información coa que se obteñen as filas que se queren engadir á táboa especificada

A continuación realizaranse, executaranse e comprobaranse os resultados das consultas de exemplo 3 e 4, no editor de consultas do SSMS.

- **Consulta de exemplo 3:** Engadiranse dúas novas cotas de balde, unha con código 20 e nome COTA GRATIS, e a outra con código 30 e nome OUTRA GRATIS. Antes e despois do INSERT faremos a consulta que nos amosa a información da táboa en cada momento.

```
SELECT codigo, nome, importe
FROM COTA;
GO
INSERT INTO COTA (codigo, nome, importe)
VALUES (20, 'COTA GRATIS', 0),
       (30, 'OUTRA GRATIS', 0);
SELECT codigo, nome, importe
FROM COTA;
```

Táboa COTA antes do INSERT da consulta de exemplo 3		
codigo	**nome**	**importe**
11	DE HONRA	100.00
12	FAMILIAR	30.00
13	HABITUAL	50.00
99	GRATUITA	0.00
Táboa COTA despois do INSERT da consulta de exemplo 3		
codigo	**nome**	**importe**
11	DE HONRA	100.00
12	FAMILIAR	30.00
13	HABITUAL	50.00
20	**COTA GRATIS**	**0.00**
30	**OUTRA GRATIS**	**0.00**
99	GRATUITA	0.00

- **Consulta de exemplo 4:** Engadiranse na táboa COTA unha nova de nome NOVA e código 80 co mesmo importe que a cota de nome HABITUAL. Antes e despois do INSERT faremos a consulta que nos amosa a información da táboa en cada momento.

```
SELECT codigo, nome, importe
FROM COTA;
GO
INSERT INTO COTA (codigo,nome,importe)
SELECT 80, 'NOVA',importe
FROM COTA
WHERE nome='HABITUAL';
SELECT codigo, nome, importe
FROM COTA;
```

Táboa COTA antes do INSERT da consulta de exemplo 4		
codigo	**nome**	**importe**
11	DE HONRA	100.00
12	FAMILIAR	30.00
13	HABITUAL	50.00
20	COTA GRATIS	0.00
30	OUTRA GRATIS	0.00
99	GRATUITA	0.00

Táboa COTA despois do INSERT da consulta de exemplo 4		
codigo	**nome**	**importe**
11	DE HONRA	100.00
12	FAMILIAR	30.00
13	HABITUAL	50.00
20	COTA GRATIS	0.00
30	OUTRA GRATIS	0.00
80	**NOVA**	**50.00**
99	GRATUITA	0.00

4. Eliminación de filas (DELETE e TRUNCATE TABLE)

4.1. Sentenza DELETE

A sentenza DELETE permite eliminar filas dunha táboa especificada.

Sintaxe sentenza DELETE	

`DELETE [FROM] táboa`
`FROM combinación`
`[WHERE predicado];`
	▪ **táboa:** Nome da táboa da que se desexa eliminar información.
	▪ **[FROM combinación]:** Permite utilizar as columnas doutras táboas na cláusula WHERE.
	▪ **[WHERE predicado]:** Condicións que deben cumprir as filas a eliminar da táboa indicada no DELETE.
	▪ **Prescindir da cláusula WHERE:** Se escribimos unha instrución DELETE sen cláusula WHERE, eliminaranse tódalas filas da táboa.

A continuación realizaranse, executaranse e comprobaranse os resultados das consultas de exemplo 5 e 6, no editor de consultas do SSMS.

- **Consulta de exemplo 5:** Eliminaremos as tres cotas de códigos 20, 30 e 80 engadidas nas consultas de exemplo 3 e 4. Antes e despois do DELETE faremos a consulta que nos amosa a información da táboa en cada momento.

```
SELECT codigo, nome, importe
FROM COTA;
GO
DELETE FROM COTA
WHERE codigo IN (20,30,80);
SELECT codigo, nome, importe
FROM COTA;
```

Táboa COTA antes do DELETE da consulta de exemplo 5		
codigo	**nome**	**importe**
11 12 13 20 30 80 99	DE HONRA FAMILIAR HABITUAL COTA GRATIS OUTRA GRATIS NOVA GRATUITA	100.00 30.00 50.00 0.00 0.00 50.00 0.00
Táboa COTA despois do DELETE da consulta de exemplo 5		
codigo	**nome**	**importe**
11 12 13 99	DE HONRA FAMILIAR HABITUAL GRATUITA	100.00 30.00 50.00 0.00

- **Consulta de exemplo 6:** Para facer esta consulta primeiro crearemos unha táboa temporal local SOCIO2 copia da táboa SOCIO empregando a sentenza SELECT...INTO. Despois teranse que eliminar os socios da táboa SOCIO2 que deben algunha actividade.

  A eliminación farase de dous xeitos, nunha primeira solución empregando unha consulta subordinada e, nunha segunda cunha combinación interna.

```
--Creación de #SOCIO2
SELECT * INTO #SOCIO2
FROM SOCIO;
--Comprobamos o contido de #SOCIO2 antes de borrar
SELECT * FROM #SOCIO2;
--Solución 1
DELETE FROM #SOCIO2
WHERE numero IN (SELECT num_socio
                 FROM SOCIO_REALIZA_ACTI
                 WHERE pagada='N');
--Comprobamos o contido de #SOCIO2 despois de borrar
SELECT * FROM #SOCIO2;
```

  Para probar a solución seguinte eliminarase a táboa temporal #SOCIO2 e volverase crear.

```
--Eliminamos e voltamos a crear #SOCIO2 para
--comprobar a segunda solución
DROP TABLE #SOCIO2;
--Creación de #SOCIO2
SELECT * INTO #SOCIO2
FROM SOCIO;
--Comprobamos o contido de #SOCIO2 antes de borrar
SELECT * FROM #SOCIO2;
--Solución 2
DELETE FROM #SOCIO2
FROM #SOCIO2 s2, SOCIO_REALIZA_ACTI sr
WHERE sr.pagada='N' AND
      s2.numero=sr.num_socio;
--Comprobamos o contido de #SOCIO2 despois de borrar
SELECT * FROM #SOCIO2;
```

Tamén se podería facer a consulta da solución 2 empregando a sintaxe do INNER JOIN como podemos ver a continuación.

```
--Solución 3
DELETE FROM #SOCIO2
FROM #SOCIO2 s2 INNER JOIN SOCIO_REALIZA_ACTI sr
     ON s2.numero=sr.num_socio
WHERE sr.pagada='N';
```

As solucións 2 e 3 serían as que estarían optimizadas xa que é menos custoso para o xestor facer unha combinación interna que unha subconsulta.

As tres consultas de eliminación eliminarán aos mesmos socios, 1001 e 1002.

Táboa #SOCIO2 antes do DELETE da consulta de exemplo 6		
numero	**nif**	**...**
1000 1001 1002 1003	11111112A 22222223B 33333334C 44444445D	...
Táboa #SOCIO2 despois do DELETE da consulta de exemplo 6		
codigo	**nome**	**...**
1000 1003	11111112A 44444445D	...

4.2. Sentenza TRUNCATE TABLE

A sentenza TRUNCATE TABLE quita todas as filas dunha táboa.

Sintaxe sentenza TRUNCATE TABLE	
`TRUNCATE TABLE táboa;`	▪ **táboa:** Nome da táboa da que se desexan eliminar tódalas filas.

4.3. Comparación de TRUNCATE con DELETE

TRUNCATE TABLE é similar á instrución DELETE sen cláusula WHERE, pero é máis rápida e usa menos recursos do sistema.

DELETE elimina unha a unha as filas e grava unha entrada no rexistro de transaccións por cada fila. TRUNCATE TABLE elimina os datos cancelando a asignación das páxinas de datos para almacenar os datos da táboa, e no rexistro só grava as cancelacións das páxinas.

DELETE usa máis bloqueos xa que bloquea cada fila da táboa para a súa eliminación, e pola contra, TRUNCATE TABLE a táboa e a páxina, pero non cada fila.

Eliminando con TRUNCANTE TABLE se a táboa ten unha columna de identidade (definida con IDENTITY), o contador desa columna volverá ao valor de inicio que se indicou na súa definición, e no caso de non existir usarase o valor predeterminado 1. Se queremos conservar o valor de identidade utilizarase DELETE.

A continuación realizaranse, executaranse e comprobaranse os resultados da consultas de exemplo 7, no editor de consultas do SSMS.

- **Consulta de exemplo 7:** Para facer esta consulta primeiro crearase unha táboa EMPREGADO2, copia da táboa EMPREGADO coas mesmas filas e columnas. A continuación eliminaremos todas as filas da táboa nova do xeito máis rápido e eficiente posible. Antes e despois do borrado faremos a consulta que nos amosa o número de da táboa EMPREGADO2 en cada momento. Ao rematar eliminarase a táboa nova coa cláusula DROP TABLE para non modificar o deseño orixinal da BD.

```
--Creamos a táboa EMPREGADO2 copia de EMPREGADO
SELECT * INTO EMPREGADO2
FROM EMPREGADO;
GO
--Comprobamos o contido de #EMPREGADO2 antes de borrar
SELECT count(*) as num_filas FROM EMPREGADO2;
--Baleiramos EMPREGADO2 do xeito máis rápido
TRUNCATE TABLE EMPREGADO2;
--Comprobamos o contido de #EMPREGADO2 despois de borrar
SELECT count(*) as num_filas FROM EMPREGADO2;
--Eliminamos a táboa EMPREGADO2;
DROP TABLE EMPREGADO2;
```

Número de filas da táboa EMPREGADO2 antes do TRUNCATE TABLE
num_filas
4
Número de filas da táboa EMPREGADO2 despois do TRUNCATE TABLE
num_filas
0

5. Modificación de contido das filas (UPDATE)

A sentenza UPDATE permite modificar o contido dunha ou máis filas da táboa indicada. Deberanse indicar os campos que queremos cambiar e o novo valor que conterá cada campo modificado.

Os novos valores deberán satisfacer as restricións dos campos da táboa. Por exemplo, non poderemos poñer un importe a 3000 se existe unha restrición no campo que indica que os importes deben ser inferiores a 2000.

Sintaxe sentenza UPDATE	

`UPDATE táboa`
`SET col1=expr1,...,colN=exprN`
`[FROM combinación]`
`[WHERE predicado];`
	▪ **táboa:** Nome da táboa da que se desexa modificar a información.
	▪ **SET col1=expr1,...,colN=exprN:** Nesta cláusula escribirase a listaxe de pares columna=expresión. Expresión pode ser un valor, unha operación matemática, o resultado dunha función integrada e incluso unha consulta SELECT.
	▪ **[FROM combinación]:** Permite utilizar as columnas doutras táboas na cláusula WHERE.
	▪ **[WHERE predicado]:** Condicións que deben cumprir as filas a modificar da táboa indicada no UPDATE.
	▪ **Prescindir da cláusula WHERE:** Se escribimos unha instrución UPDATE sen cláusula WHERE, modificaranse as columnas indicadas en tódalas filas da táboa.

A continuación realizarase, executarase e comprobarase o resultado das consultas de exemplo 8, 9, 10 e 11, no editor de consultas do SSMS.

- **Consulta de exemplo 8:** Nesta consulta incrementarase o prezo das actividades en 4 euros. Para deixar os datos orixinais da BD, faremos unha segunda modificación de redución do prezo en 4 euros. Faremos unha consulta antes e despois do incremento do prezo.

```
--Comprobamos o contido de ACTIVIDADE antes de modificar
SELECT identificador, prezo
FROM ACTIVIDADE;
--Incremento do prezo das actividades
UPDATE ACTIVIDADE
SET prezo=prezo+4;
--Comprobamos o contido de ACTIVIDADE despois de modificar
SELECT identificador, prezo
FROM ACTIVIDADE;
--Revertemos os cambios
UPDATE ACTIVIDADE
SET prezo=prezo-4;
```

Datos da táboa ACTIVIDADE antes do UPDATE da consulta de exemplo 8	
identificador	**prezo**
10 20 30 40	301.55 50.00 80.00 0.00
Datos da táboa ACTIVIDADE despois do UPDATE da consulta de exemplo 8	
identificador	**prezo**
10 20 30 40	305.55 54.00 84.00 4.00

Pódese observar como nestas consultas ao non usar a cláusula WHERE tódalas filas da táboa se ven afectadas polos cambios. Neste caso modifícase o prezo de tódalas actividades.

- **Consulta de exemplo 9:** Nesta consulta incrementarase en 7 o número de prazas da actividade con número 10, o seu nome pasará a ser CURSO TENIS e aumentarase o seu prezo en 5'14%. Para deixar os datos orixinais da BD, faremos unha segunda modificación e a táboa quedará como estaba antes da modificación. Faremos unha consulta antes e despois da modificación para comprobar os cambios.

```
--Comprobamos o contido de ACTIVIDADE antes de modificar
SELECT identificador, nome, num_prazas, prezo
FROM ACTIVIDADE
WHERE identificador=10;
--Modificación das prazas, nome e prezo da actividade 10
UPDATE ACTIVIDADE
SET  num_prazas=num_prazas+7,
     nome='CURSO TENIS',
     prezo=prezo*1.0514
WHERE identificador=10;
--Comprobamos o contido de ACTIVIDADE despois de modificar
SELECT identificador, nome, num_prazas, prezo
FROM ACTIVIDADE
WHERE identificador=10;
--Revertemos os cambios
UPDATE ACTIVIDADE
SET  num_prazas=num_prazas-7,
     nome='TENIS PARA PRINCIPIANTES',
     prezo=prezo/1.0514
WHERE identificador=10;
```

Datos da táboa ACTIVIDADE 10 antes do UPDATE da consulta de exemplo 9			
identificador	**nome**	**num_prazas**	**prezo**
10	TENIS PARA PRINCIPIANTES	15	301.55
Datos da táboa ACTIVIDADE 10 despois do UPDATE da consulta de exemplo 9			
identificador	**prezo**		
10	CURSO TENIS	22	317.05

Ao incluír a cláusula WHERE só se verán afectadas pola modificación as filas que cumpren as condicións dos predicados do WHERE.

- **Consulta de exemplo 10:** Modificarase o prezo das cotas gratis co valor do prezo máis alto das actividades. Para deixar os datos orixinais da BD, faremos unha segunda modificación e a táboa quedará como estaba antes da modificación. Faremos unha consulta antes e despois da modificación para comprobar os cambios.

```
--Comprobamos o contido de COTA antes de modificar
SELECT codigo, nome, importe
FROM COTA;
--Modificamos
UPDATE COTA
SET importe=(SELECT max(prezo) FROM ACTIVIDADE)
WHERE importe=0;
--Comprobamos o contido de COTA despois de modificar
SELECT codigo, nome, importe
FROM COTA;
--Revertemos os cambios
UPDATE COTA
SET importe=0
WHERE codigo=99;
```

Táboa COTA antes do UPDATE da consulta de exemplo 10		
codigo	**nome**	**importe**
11 12 13 99	DE HONRA FAMILIAR HABITUAL GRATUITA	100.00 30.00 50.00 0.00
Táboa COTA despois do UPDATE da consulta de exemplo 10		
codigo	**nome**	**importe**
11 12 13 99	DE HONRA FAMILIAR HABITUAL GRATUITA	100.00 30.00 50.00 301.55

Obsérvase como na cláusula SET podemos asignar como valor dun campo, ou de varios, o resultado dunha consulta SELECT. É importante usar unha consulta que devolva un único valor.

- **Consulta de exemplo 11:** Poránselle como pagadas todas as actividades aos socios que teñan abonada a cota anual. Para deixar os datos orixinais da BD, faremos unha segunda modificación e a táboa quedará como estaba antes da modificación. Faremos unha consulta antes e despois da modificación para comprobar os cambios.

```
--Comprobamos o contido de SOCIO e
--SOCIO_REALIZA_ACTI antes de modificar
SELECT s.numero, s.abonada, sr.id_actividade, sr.pagada
FROM SOCIO s INNER JOIN SOCIO_REALIZA_ACTI sr
     ON s.numero=sr.num_socio
WHERE s.abonada='S';
--Modificamos
UPDATE SOCIO_REALIZA_ACTI
SET pagada='S'
FROM SOCIO s INNER JOIN SOCIO_REALIZA_ACTI sr
     ON s.numero=sr.num_socio
WHERE s.abonada='S';
--Comprobamos o contido de SOCIO e
--SOCIO_REALIZA_ACTI despois de modificar
SELECT s.numero, s.abonada, sr.pagada
FROM SOCIO s INNER JOIN SOCIO_REALIZA_ACTI sr
     ON s.numero=sr.num_socio
WHERE s.abonada='S';
--Revertemos os cambios
UPDATE SOCIO_REALIZA_ACTI
SET pagada='N'
WHERE num_socio=1002 AND
      id_actividade=40;
```

Datos da táboa SOCIO e SOCIO_REALIZA_ACTI antes do UPDATE da consulta de exemplo 11			
numero	**abonada**	**id_actividade**	**pagada**
1000 1002 1003	S S S	10 40 30	S N S
Datos da táboa SOCIO e SOCIO_REALIZA_ACTI despois do UPDATE da consulta de exemplo 11			
numero	**abonada**	**id_actividade**	**pagada**
1000 1002 1003	S S S	10 40 30	S **S** S

Nesta modificación necesítase facemos unha combinación interna para modificar unicamente os socios que pagaron a cota anual. Ese dato non está na táboa que se vai modificar e por iso procuramos os datos no FROM.

6. Medidas para manter a integridade da información

Unha BD mantén a integridade da información se as súas restricións foron ben definidas e ademais conséguese que estas non se vulneren.

No modelo relacional clasifícanse as restricións en dous tipos:

- Restricións inherentes (propias do modelo):
  - Obrigatoriedade de clave primaria.
  - Orde de filas e atributos non é relevante.
  - Os atributos toman un único valor do seu dominio.
  - Cada un dos atributos que forman parte dunha clave primaria non poden tomar valor nulo (integridade de entidade).
- Restricións semánticas ou de usuario (establecidas polo deseñador da BD):
  - De clave primaria.
  - De unicidade.
  - De obrigatoriedade.
  - De integridade referencial.
  - De verificación.
  - De aserción.
  - Disparadores.

Cando executamos consultas DML de actualización da información da BD, debemos asegurarnos de que vai seguir mantendo a integridade da información. Se as restricións foron ben establecidas, o servidor fará ese traballo por nós, avisando cando unha das nosas consultas vulnere algunha das restricións impostas. De aí a importancia do deseño correcto nunha BD relacional.

Casos que vulnerarían a integridade dunha BD relacional

- Crear una táboa sen establecer a clave primaria. Vulnera a obrigatoriedade de clave primaria e non se podería distinguir unha fila doutra.
- Intentar engadir unha fila repetindo a clave primaria. Sería o mesmo caso, non poderiamos diferenciar unha fila doutra.
- Intentar modificar un campo poñendo o seu valor a NULL cando o campo é obrigatorio.
- Modificar o valor dunha columna que é clave candidata de tal xeito que o valor se repite en 2 filas ou máis.
- Engadir unha fila, ou varias, que conteñen unha clave foránea que non existe como clave primaria en ningunha fila da táboa pai coa que se relaciona.

- Borrar filas dunha táboa pai se existen filas relacionadas nas táboas fillas (sempre que o borrado sexa restrinxido, porque se é en cascada permitirase).
- Modificar o valor dun campo de tal xeito que o CHECK ou verificación establecido non se cumpra.

Na seguinte tarefa proporanse código de consultas T-SQL que vulnerarían a integridade da BD SOCIEDADE_CULTURAL. Deberase analizar o erro que lanza o servidor ao executalas e dicir que restrición da BD se vulneraría en cada caso.

- Intentamos engadir unha cota de código 11

```
INSERT INTO COTA (codigo, nome, importe)
VALUES (11, 'PRUEBA', 56);
```

O INSERT lanza o seguinte erro no servidor

```
Infracción de la restricción PRIMARY KEY 'PK_COTA_codigo'.
No se puede insertar una clave duplicada en el objeto 'dbo.COTA'.
El valor de la clave duplicada es (11).
```

A restrición que se vulnera é a de clave primaria, xa que na táboa COTA xa existe unha fila con código 11.

- Cambiamos a NULL o nome do socio 1001.

```
UPDATE SOCIO
SET nome=NULL
WHERE numero=1001;
```

O UPDATE lanza o erro

```
No se puede insertar el valor NULL en la columna 'nome',
tabla 'SOCIEDADE_CULTURAL.dbo.SOCIO'.
La columna no admite valores NULL. Error de UPDATE.
```

Vulnérase a restrición de obrigatoriedade xa que o campo *nome* é obrigatorio.

- Asígnase a aula da actividade 10 á actividade número 40.

```
UPDATE ACTIVIDADE
SET num_aula= (SELECT num_aula
               FROM ACTIVIDADE
               WHERE identificador=10)
WHERE identificador=40;
```

O UPDATE lanza o erro

```
Infracción de la restricción UNIQUE KEY 'UQ_NUM_AULA'.
No se puede insertar una clave duplicada en el objeto 'dbo.ACTIVIDADE'.
El valor de la clave duplicada es (1).
```

Vulnérase a restrición de unicidade UQ_NUM_AULA, xa que o campo *num_aula* de ACTIVIDADE é clave candidata e non se poden repetir valores na columna, é dicir, dúas actividades non poden compartir aula.

- Engádese un docente con código 500 na táboa PROFESORADO, que é un subtipo de EMPREGADO.

```
INSERT INTO PROFESORADO (num_prof, especialidade)
VALUES (500, 'Francés');
```

O INSERT lanza o erro

```
Instrucción INSERT en conflicto con la restricción FOREIGN KEY
"FK_PROFESORADO_EMPREGADO". El conflicto ha aparecido en la base de datos
"SOCIEDADE_CULTURAL", tabla "dbo.EMPREGADO", column 'numero'.
```

Vulnérase a restrición de clave foránea FK_PROFESORADO_EMPREGADO porque non existe a fila pai co código 500 en EMPREGADO.

- Inténtase eliminar a provincia de código 15.

```
DELETE FROM PROVINCIA
WHERE codigo=15;
```

O DELETE lanza o erro

```
Instrucción DELETE en conflicto con la restricción REFERENCE "FK_SOCIO_PROVINCIA".
El conflicto ha aparecido en la base de datos "SOCIEDADE_CULTURAL",
tabla "dbo.SOCIO", column 'cod_provincia_enderezo'.
```

Vulnérase a restrición de clave foránea FK_SOCIO_PROVINCIA porque na táboa SOCIO existen socios que viven nesa provincia que pretendemos borrar. Se o servidor o permitise non poderiamos coñecer o nome da provincia na que viven eses socios.

- Inténtase poñer como data de inicio o día despois da data de fin na actividade 10.

```
UPDATE ACTIVIDADE
SET data_ini = DATEADD(day, +1, data_fin)
WHERE identificador=10;
```

O UPDATE lanza o erro

```
Instrucción UPDATE en conflicto con la restricción CHECK "CHK_control_datas".
El conflicto ha aparecido en la base de datos "SOCIEDADE_CULTURAL",
tabla "dbo.ACTIVIDADE".
```

Vulnérase a restrición de verificación ou CHECK CHK_control_datas que controla que ((data_fin >= data_ini)), e con esta modificación, se o servidor o permitise, a data de inicio sería maior que a de fin na actividade 10.

7. Introdución ao uso das transaccións para evitar modificacións nas BBDD

Unha transacción é unha unidade de traballo formada por unha ou máis sentencias SQL que deben executarse na súa totalidade. É dicir, unha transacción para que sexa correcta debe realizarse completamente, isto é, deberán executarse sen error todas as consultas SQL que a compoñen. Ou se realizan todas ou ningunha.

Non afondaremos no seu uso, xa que non é un tema desta unidade. Centrarémonos só en dúas das sentenzas da linguaxe de procesamento de transaccións, que usaremos para que os cambios das nosas consultas SQL, non se fagan efectivas na BD. Evitaremos así ter que facer unha consulta de modificación para reverter os cambios na BD.

7.1. Sentenza BEGIN TRANSACTION

Esta sentenza marca o inicio dunha transacción. Indicaremos BEGIN TRANSACTION ou BEGIN TRAN xusto ao principio da nosa consulta de modificación. Poderanse desfacer tódalas modificacións realizadas nos datos despois de BEGIN TRANSACTION.

7.2. Sentenza ROLLBACK

Cada transacción dura ata que se pecha con COMMIT ou con ROLLBACK. Con COMMIT as modificacións feitas durante a transacción serán permanentes na BD. Con ROLLBACK desfaranse todas as modificacións da transacción.

Cando rematemos as nosas consultas xa non teremos que executar a consulta de modificación para reverter os cambios feitos, chegará con lanzar unha instrución ROLLBACK.

Na consulta de exemplo 8 incrementárase o prezo de tódalas actividades en 4 euros e despois para desfacer o cambio o que fixemos foi lanzar outro UPDATE pero esta vez para restar os 4 euros engadidos no primeiro UPDATE. Empregando transaccións quedaría deste xeito.

```
/*** CONSULTA DE EXEMPLO 8 MODIFICADA CON TRANSACCIÓNS***/
BEGIN TRAN
--Comprobamos o contido de ACTIVIDADE antes de modificar
SELECT identificador, prezo
FROM ACTIVIDADE;
--Incremento do prezo das actividades
UPDATE ACTIVIDADE
SET prezo=prezo+4;
--Comprobamos o contido de ACTIVIDADE despois de modificar
SELECT identificador, prezo
FROM ACTIVIDADE;
--Revertemos os cambios
ROLLBACK;
```

Tarefa 1. Unha vez copiados, executados e probados os exemplos das explicacións, deberase escribir o código T-SQL necesario para obter a información de cada unha das consultas de modificación propostas na tarefa 1.

8. Tarefas

As tarefas propostas son as seguintes.

- **Tarefa 1. Realización de consultas de modificación en T-SQL.** Nesta tarefa escribirase o código T-SQL necesario para conseguir os cambios que se piden en cada unha das consultas de modificación propostas. Empregaranse transaccións para evitar que os cambios sexan permanentes na BD.

8.1. Tarefa 1. Realización de consultas de modificación en T-SQL

Realizar o código T-SQL adecuado para obter a información que se pide en cada unha das consultas propostas na BD SOCIEDADE_CULTURAL ou na BD EMPRESA. Todas as consultas deberán facerse dentro dunha transacción pechada cun ROLLBACK para desbotar os cambios. Ademais antes e despois das consultas de modificación faranse as consultas de recuperación da información adecuadas a cada caso, para comprobar o antes e o despois da modificación.

- Consultas propostas na BD SOCIEDADE_CULTURAL.
  - **Proposta 1.** Aumentar o número de prazas das actividades nun 15%.
  - **Proposta 2.** Cambiar o estado da aula de nome AULA SUR a regular (R).
  - **Proposta 3.** Engadir unha aula nova de número 9, nome AULA NOVA e con superficie e estado os mesmos que os da aula COCIÑA.
  - **Proposta 4.** Engadir dúas novas cotas, unha cos datos 21, COTA1, 75 e outra cos datos 22, COTA2 e 74.3.
  - **Proposta 5.** Crear unha táboa temporal global PROFE_ASISTENTE_ACTI co nif, nome e primeiro apelido do profesorado que asiste a actividades.
  - **Proposta 6.** Crear unha táboa permanente de nome AULA_MALA coas aulas en mal estado (Estado=M) e coas mesmas columnas da táboa AULA. Os nomes dos campos de AULA_MALA serán *codigo*, *nome*, *m2* e *estado.*
  - **Proposta 7.** Crear unha táboa temporal local que sexa unha copia en canto a contido e columnas da táboa ACTIVIDADE e que se chame ACTI2. Antes de pechar a transacción, farase unha consulta que elimine todas as actividades da táboa nova que non teñan observacións.
  - **Proposta 8.** Crear unha táboa temporal local de nome SOCIO2 copia de SOCIO. A continuación faremos a consulta que elimine de SOCIO2 aqueles socios que non teñen teléfono algún.
  - **Proposta 9.** Substituír os espazos en branco das observacións das actividades, as que asisten docentes, por guións baixos(_).
  - **Proposta 10.** Retrasar nun día a data de inicio de tódalas actividades que aínda non comezaron a día de hoxe.

- Consultas propostas na BD EMPRESA.
  - **Proposta 11.** Eliminar os fabricantes dos que non hai produtos na BD.
  - **Proposta 12.** Incrementar o obxectivo das sucursais da rexión OESTE nun 6% e modificar o nome da rexión por WEST.
  - **Proposta 13.** Crear unha táboa de nome FABRICANTE2 que sexa unha copia de FABRICANTE en número e nome de columnas e en contido. Elimina todas as filas da nova táboa do xeito máis rápido e menos custoso posible.
  - **Proposta 14.** Transferir todos os empregados que traballan na sucursal de BARCELONA á sucursal de VIGO, e cambiar a súa cota de vendas pola media das cotas de vendas de tódolos empregados.
  - **Proposta 15.** Elimina os pedidos de empregados contratados antes do ano 2001.