1. A02. Modificación da estrutura de bases de datos relacionais

1.1 Introdución

1.1.1 Obxectivos

O obxectivo desta actividade é modificar a estrutura de bases de datos e táboas seguindo as especificacións dos diagramas conceptuais e lóxicos ou atendendo a novos requirimentos, utilizando a linguaxe SQL de definición de datos (LDD ou DDL), e asistentes e ferramentas gráficas.

1.2 Actividade

1.2.1 Modificación dunha base de datos

A sintaxe da sentenza que permite cambiar as características globais dunha base de datos é:

```
ALTER {DATABASE | SCHEMA}  nome_da_base  [opcións_a_modificar] ...
```

Consideracións sobre a sintaxe anterior:

- As opcións para modificar son as mesmas que as opcións de creación:

```
[DEFAULT] CHARACTER SET [=] nome_xogo_carácteres
[DEFAULT] COLLATE [=] nome_sistema_ordenación
```

- O cambio do xogo de carácteres ou as opcións de cotexamento para unha base de datos na que xa existan táboas, non afecta a estas e só afecta ás que se creen no futuro.

1.2.2 Modificación do esquema dunha táboa

Co paso do tempo é normal que se teñan que facer adaptacións no esquema das táboas das bases de datos, pola aparición de novos requirimentos, novas restricións, ou a desaparición dalgunhas das existentes. Algunhas veces, ter que facer cambios débese a non terlle dedicado o tempo suficiente ás fase de deseño conceptual e lóxico; é moi recomendable pararse a facer un bo deseño antes de empezar a escribir código para crear a base de datos.

A sentenza ALTER TABLE permite facer modificacións no esquema dunha táboa que xa existe na base de datos. Sintaxe:

```
ALTER TABLE nome_táboa
[especificación_alter [, especificación_alter] …]
```

A *especificación_alter* pode ser:

```
opcións de táboa
| ADD [COLUMN] nome_columna definición_columna [FIRST | AFTER nome_columna]
| ADD [COLUMN] (nome_columna definición_columna, …)
| CHANGE [COLUMN] nome_columna nome_novo nova_definición_columna
          [FIRST|AFTER nome_columna],
| MODIFY [COLUMN] nome_columna <nova_definición_columna>] [FIRST | AFTER nome_columna],
| ALTER [COLUMN] nome_columna {SET DEFAULT valor | DROP DEFAULT},
| ADD {INDEX|KEY} [nome_índice] (columnas_índice),
| ADD [CONSTRAINT [nome_restrición]] PRIMARY KEY (lista_columnas),
| ADD [CONSTRAINT [nome_restrición]] UNIQUE {INDEX|KEY} [nome_indice](lista_columnas),
| ADD [CONSTRAINT [nome_restrición]] FOREIGN KEY [nome_índice](lista_columnas)
```

```
      REFERENCES nome_táboa (lista_de_columnas)[ON DELETE opción][ON UPDATE opción]
| DROP [COLUMN] nome_columna,
| DROP {INDEX | KEY} nome_índice,
| DROP PRIMARY KEY,
| DROP FOREIGN KEY nome_restrición,
| RENAME [TO | AS] nome_táboa_nova
| CONVERT TO CHARACTER SET xogo_carácteres [COLLATE sistema_colación]
```

Consideracións sobre a sintaxe:

- As opcións de táboa afectan ás características da táboa e son as mesmas que se utilizan na sentenza CREATE TABLE. Algunhas opcións de táboa son:

```
[DATA DIRECTORY= 'directorio']
[INDEX DIRECTORY= 'directorio']
[{ENGINE | TYPE} = {ISAM, MyISAM, InnodDB, ...}]
[[DEFAULT] CHARACTER SET nome_xogo_carácteres] [COLLATE nome_sistema_colación]]
[AUTO_INCREMENT = número]
```

  - DATA DIRECTORY e INDEX DIRECTORY permiten cambiar as rutas absolutas nas que se almacenan os datos e os índices.
  - ENGINE permite cambiar o motor de almacenamento asociado á táboa. Ver o apartado "Motores de almacenamento en MySQL" que está máis adiante
  - CHARACTER SET e COLLATE permiten cambiar o conxunto de carácteres e o sistema de colación predeterminados para as columnas que se crean nesa táboa. Non afecta ás columna que xa están creadas. Para cambiar o contido das columnas tipo cadea de carácteres, hai que utilizar a cláusula CONVERT TO CHARACTER SET.
  - AUTO_INCREMENT permite cambiar o número de comezo para a columna de tipo autoincremental.
- En xeral, as cláusulas ADD permiten engadir novas propiedades e as cláusulas DROP permiten eliminalas.
- Se ao engadir unha columna (ADD) non se especifica a cláusula FIRST | AFTER, a nova columna engádese ao final da táboa.
- Non é posible engadir unha columna tipo autoincremento se a táboa non está baleira.
- Non se pode modificar NULL por NOT NULL se a táboa contén valores nulos para a columna a modificar; a operación inversa non presenta ningún problema.
- Para cambiar a definición dunha columna, hai que utilizar a cláusula MODIFY, pero se ademais da definición tamén se quere cambiar o nome, entón hai que utilizar a cláusula CHANGE. O editor da actual versión de Workbench non recoñece a cláusula MODIFY e marca a liña como un erro aínda que se executa correctamente.
- A cláusula ALTER permite modificar o valor por defecto para unha columna.
- A cláusula RENAME permite cambiar o nome da táboa.
- A cláusula CONVERT TO CHARACTER SET permite cambiar o xogo de carácteres e o sistema de colación por defecto, e ademais, cambiar os valores de todas as columnas tipo cadea de carácteres (CHAR, VARCHAR e TEXT). Hai que ter en conta que non todos os sistemas de colación utilizan o mesmo número de bytes polo que pode ser necesario cambiar antes o tipo ou tamaño da columna.

  Pódese facer a conversión só en algunha columna e non en todas as tipo cadea de carácteres, nese caso pódese utilizar a cláusula MODIFY. Exemplo:

```
alter table proba
  modify column texto_latin1 varchar(200) character set utf8,
```

No manual pódense consultar todas as opcións que ten esta sentenza.

Exemplo de modificación da estrutura da táboa *fabricante* da base de datos *practicas1*:

```
alter table fabricante
   add column pais varchar(60) default null,
   add column enderezo varchar(200) not null after idFabricante,
```

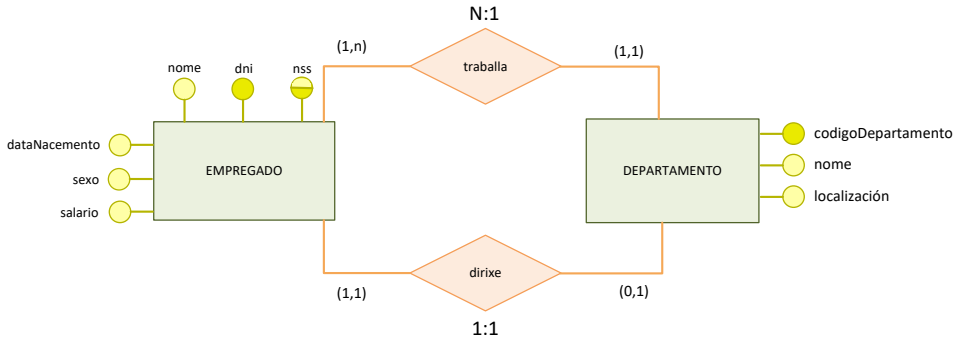
```
add index idx_fabricante_nome (nome),
engine = Innodb;
```

1.2.3 Engadir relacións e restricións de clave foránea

Pódense engadir relacións e restricións de clave foránea de dúas formas:

- Crear ao mesmo tempo as táboas e as relacións, empregando a sentenza CREATE TABLE. Neste caso hai que ter en conta a orde en que se crean as táboas, xa que non se pode crear unha táboa que conteña unha clave foránea se aínda non está creada a táboa á que fai referencia. Isto pode representar un problema no caso de relacións bidireccionais.
- Crear primeiro as táboas e establecer as relacións despois empregando sentenzas ALTER TABLE. Recoméndase esta segunda opción.

Por exemplo, supóñase que a entidade *empregado* está relacionada coa entidade *departamento* cunha relación *traballa* de tipo N:1, e *departamento* está relacionada con *empregado* coa relación *dirixe*, de tipo 1:1, con cardinalidade mínima 0. Ao crear primeiro a táboa *empregado* e despois *departamento*, provocaríase un erro porque na orde de creación da táboa *empregado* se define unha clave foránea que fai referencia á táboa *departamento*, que aínda non existe. O mesmo ocorre se empeza creando *departamento*.



En MySQL pódese solucionar o problema anterior desactivando a verificación de claves foráneas, poñendo o valor 0 ou OFF na variable *foreign_key_checks*.

```
/*desactivar a verificación de claves foráneas */
set FOREIGN_KEY_CHECKS = 0;
/*activar a verificación de claves foráneas*/
set FOREIGN_KEY_CHECKS = 1;
```