

Bases de Datos I

Modelo Físico

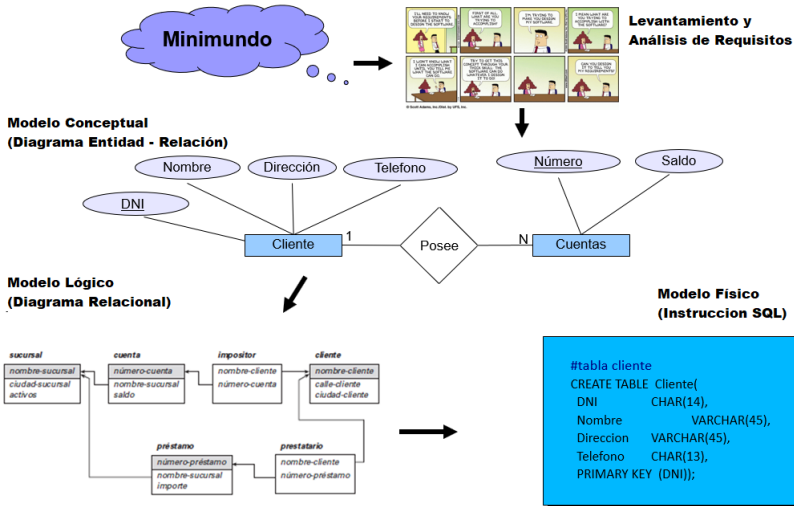
ORACLE®



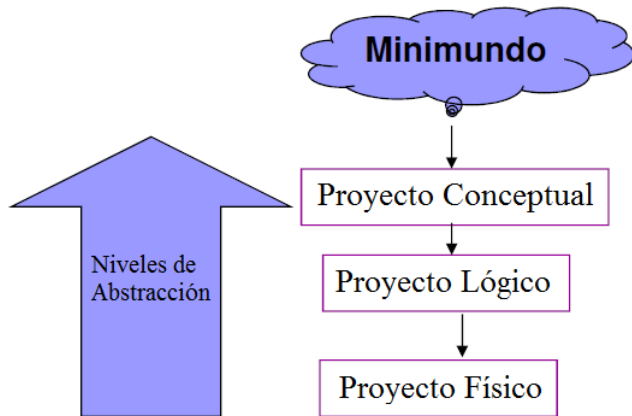
1 Fases del Proyecto de BD

2 Modelo Físico

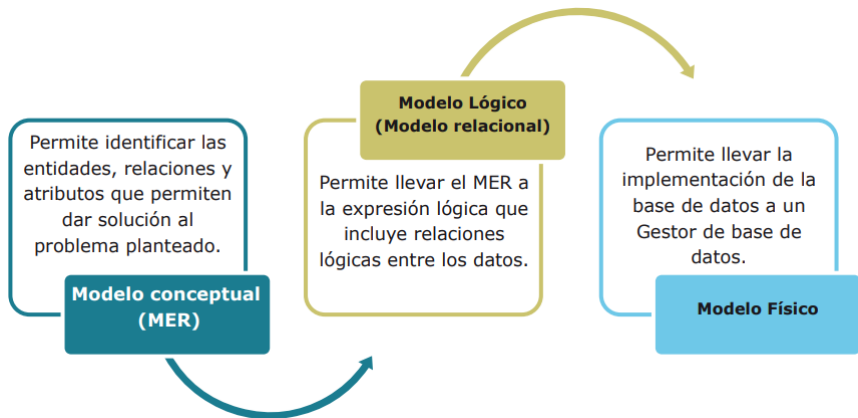
Fases del Proyecto de BD



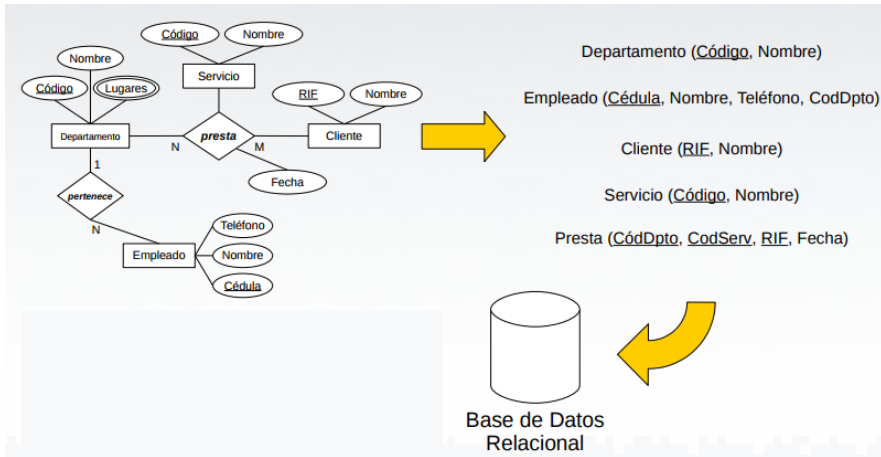
Fases del Proyecto de BD



Fases del Proyecto de BD



Fases del Proyecto de BD



Modelo Físico

- Describe, por medio de un lenguaje, como se almacenarán los datos y las relaciones de la base de datos.
- Debería partir del Modelo Relacional.
- En este nivel se escoge que Sistema Gestor de Base de Datos (SGBD) utiliza.

Sistema Gestor de Base Datos



Lenguaje SQL

- Structured Query Language (SQL) o Lenguaje de Consulta Estructurado.
- Lenguaje más utilizado por los SGBD.
- Aunque nos referimos al lenguaje SQL como un "lenguaje de consulta", puede hacer mucho más que simplemente consultar una BD. Puede definir la estructura de los datos, modificar los datos en la BD, especificar restricciones de seguridad, entre otros.

Lenguaje SQL

- Fue desarrollado originalmente a inicios de los años 70 en los laboratorios de IBM en San Jose, dentro del proyecto System R, que tenía como objetivo demostrar la viabilidad de la implementación del modelo relacional propuesto por E. F. Codd.



E.F. Codd

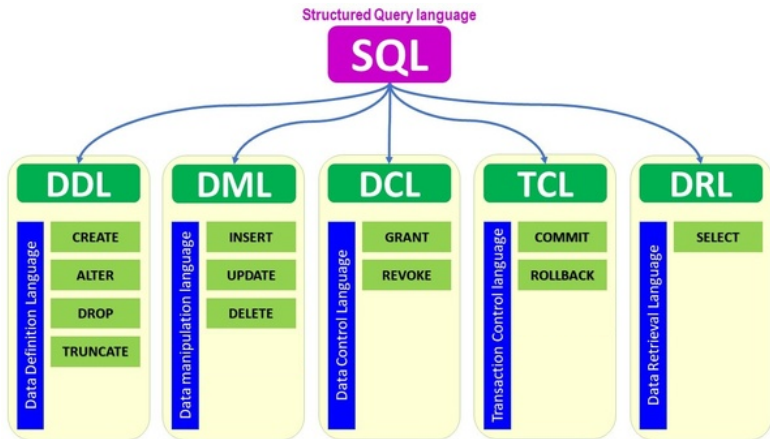
Lenguaje SQL

- El nombre original del lenguaje era SEQUEL (Structured English Query Language) o Lenguaje de Consulta Estructurado en Inglés, de ahí el hecho, que hasta ahora, la sigla sql, en inglés, es comúnmente pronunciada como "síquel".
- El lenguaje es un standard de las bases de datos relacionales. Ello debido a su simplicidad y facilidad de uso. Ello reduce el ciclo de aprendizaje para aquellos que se inician en el lenguaje.

Lenguaje SQL

- SQL cuenta con varios tipos de sentencias que se pueden utilizar para realizar diversas tareas.
- Dependiendo de las tareas, estas sentencias se pueden clasificar en cinco grupos principales:

Lenguaje SQL



Lenguaje SQL: DDL (Data Definition Language)

- Permite crear y modificar la estructura de una BD:
 - CREATE: Utilizado para crear nuevas tablas, campos e índices.
 - ALTER: Utilizado para modificar las tablas agregando campos o cambiando la definición de los campos.
 - DROP: Empleado para eliminar tablas e índices.
 - TRUNCATE: Empleado para eliminar todos los registros de una tabla.

Lenguaje SQL: DML (Data Manipulation Language)

- Permite recuperar, almacenar, modificar, eliminar, insertar y actualizar datos de una BD:
 - INSERT: Utilizado para cargar de datos en la base de datos en una única operación.
 - UPDATE: Utilizado para modificar los valores de los campos y registros especificados
 - DELETE: Utilizado para eliminar registros de una tabla de una base de datos.

Lenguaje SQL: DCL (Data Control Language)

- Permite crear roles, permisos e integridad referencial, así como el control al acceso a la BD:
 - GRANT: Usado para otorgar privilegios de acceso de usuario a la base de datos.
 - REVOKE: Utilizado para retirar privilegios de acceso otorgados con el comando GRANT.

Lenguaje SQL: TCL (Transactional Control Language)

- Permite administrar diferentes transacciones que ocurren dentro de una BD:
 - COMMIT: Empleado para guardar el trabajo hecho.
 - ROLLBACK: Utilizado para deshacer la modificación que hice desde el último COMMIT.

Lenguaje SQL: DRL (Data Retrieval Language)

- Permite obtener datos o información de la BD:
 - SELECT: Utilizado para consultar registros de la base de datos que satisfagan un criterio determinado.

Sistema Gestor de Base de Datos: MySQL



Herramienta Visual: MySQL Workbench



MySQL y MySQL Workbench: Versión

MySQL Workbench

LocalHost x

File Edit View Query Database Server Tools Scripting Help

Navigator

MANAGEMENT

- Server Status
- Client Connections
- Users and Privileges
- Status and System Variables
- Data Export
- Data Import/Restore

INSTANCE

- Startup / Shutdown
- Server Logs
- Options File

PERFORMANCE

- Dashboard
- Performance Reports
- Performance Schema Setup

Administration Schemas

Information

No object selected

Administration - Server Status

Connection Name
LocalHost

Host: **DESKTOP-9HS72UA**
Socket: **MySQL**
Port: **3306**
Version: **8.0.31 (MySQL Community Server - GPL)**
Compiled For: **Win64 (x86_64)**
Configuration File: **C:\ProgramData\MySQL\MySQL Server 5.5\my.ini**
Running Since: **Sat Oct 22 08:18:19 2022 (0:01)**

Refresh

Available Server Features

Performance Schema:	<input checked="" type="radio"/> On
Thread Pool:	<input type="radio"/> n/a
Memcached Plugin:	<input type="radio"/> n/a
Semisync Replication Plugin:	<input type="radio"/> n/a
SSL Availability:	<input checked="" type="radio"/> On

Server Directories

Base Directory:	C:\Program Files\MySQL\MySQL Server 8.0\
Data Directory:	C:\ProgramData\MySQL\MySQL Server 8.0\Data\
Disk Space in Data Dir:	Could not determine
Plugins Directory:	C:\Program Files\MySQL\MySQL Server 8.0\lib\plugin\
Tmp Directory:	C:\WINDOWS\SERVIC~1\NETWOR~1\AppData\Local\Temp\
Error Log:	<input checked="" type="radio"/> On .\DESKTOP-9HS72UA.err

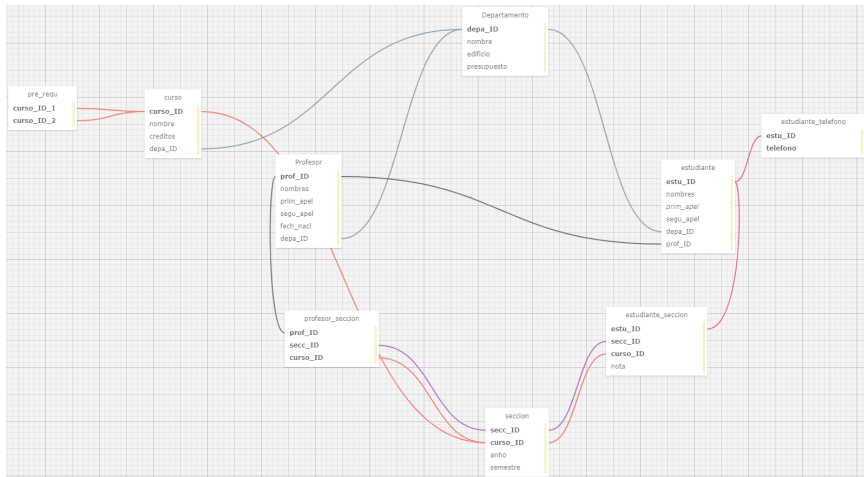
MySQL Workbench 8.0

Version 8.0.31 build 2235049 CE (64 bits) Community

Copyright © 2008, 2022 Oracle and/or its affiliates. All Rights Reserved.
Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

ORACLE

Ejemplo: Creación de la Base de Datos Universidad

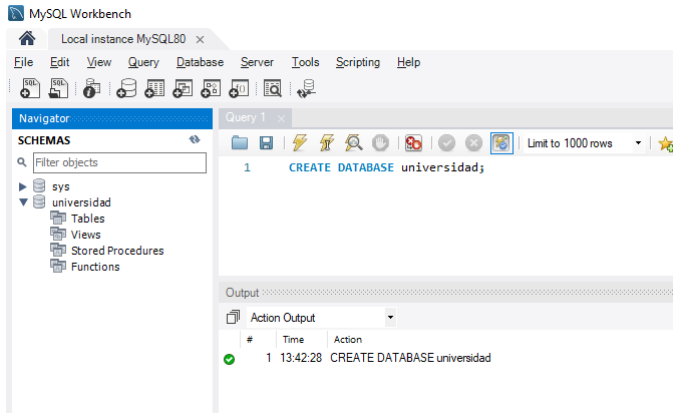


Crear la Base de Datos

The screenshot shows the MySQL 8.0 Reference Manual page for "Creating and Selecting a Database". The page is part of the MySQL 8.0 Reference Manual, version 8.0. The main heading is "3.3.1 Creating and Selecting a Database". Below the heading, it states: "If the administrator creates your database for you when setting up your permissions, you can begin using it. Otherwise, you need to create it yourself:". A code block shows the SQL command: `mysql> CREATE DATABASE menagerie;`. Below the code block, it explains: "Under Unix, database names are case-sensitive (unlike SQL keywords), so you must always refer to your database as `menagerie`, not as `Menagerie`, `MENAGERIE`, or some other variant. This is also true for table names. (Under Windows, this restriction does not apply, although you must refer to databases and tables using the same lettercase throughout a given query. However, for a variety of reasons, the recommended best practice is always to use the same lettercase that was used when the database was created.)". A "Note" box at the bottom states: "If you get an error such as `ERROR 1044 (42000): Access denied for user 'micah'@'localhost' to database 'menagerie'` when".

- Podemos borrar la Base de Datos usando `DROP DATABASE IF EXISTS` universidad

Crear la Base de Datos



Crear las Tablas

String Data Types

Data type	Description
CHAR(size)	A FIXED length string (can contain letters, numbers, and special characters). The size parameter specifies the column length in characters - can be from 0 to 255. Default is 1
VARCHAR(size)	A VARIABLE length string (can contain letters, numbers, and special characters). The size parameter specifies the maximum column length in characters - can be from 0 to 65535
BINARY(size)	Equal to CHAR(), but stores binary byte strings. The size parameter specifies the column length in bytes. Default is 1
VARBINARY(size)	Equal to VARCHAR(), but stores binary byte strings. The size parameter specifies the maximum column length in bytes.
TINYBLOB	For BLOBs (Binary Large Objects). Max length: 255 bytes
TINYTEXT	Holds a string with a maximum length of 255 characters
TEXT(size)	Holds a string with a maximum length of 65,535 bytes
BLOB(size)	For BLOBs (Binary Large Objects). Holds up to 65,535 bytes of data
MEDIUMTEXT	Holds a string with a maximum length of 16,777,215 characters
MEDIUMBLOB	For BLOBs (Binary Large Objects). Holds up to 16,777,215 bytes of data
LONGTEXT	Holds a string with a maximum length of 4,294,967,295 characters
LONGBLOB	For BLOBs (Binary Large Objects). Holds up to 4,294,967,295 bytes of data
ENUM(val1, val2, val3, ...)	A string object that can have only one value, chosen from a list of possible values. You can list up to 65535 values in an ENUM list. If a value is inserted that is not in the list, a blank value will be inserted. The values are sorted in the order you enter them
SET(val1, val2, val3, ...)	A string object that can have 0 or more values, chosen from a list of possible values. You can list up to 64 values in a SET list

Crear las Tablas

Numeric Data Types

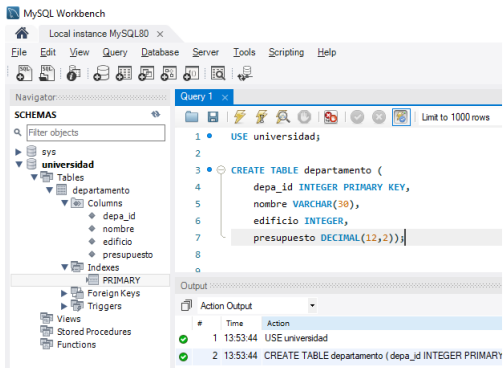
Data type	Description
<code>BIT(size)</code>	A bit-value type. The number of bits per value is specified in <i>size</i> . The size parameter can hold a value from 1 to 64. The default value for <i>size</i> is 1.
<code>TINYINT(size)</code>	A very small integer. Signed range is from -128 to 127. Unsigned range is from 0 to 255. The size parameter specifies the maximum display width (which is 255)
<code>BOOL</code>	Zero is considered as false, nonzero values are considered as true.
<code>BOOLEAN</code>	Equal to <code>BOOL</code>
<code>SMALLINT(size)</code>	A small integer. Signed range is from -32768 to 32767. Unsigned range is from 0 to 65535. The size parameter specifies the maximum display width (which is 255)
<code>MEDIUMINT(size)</code>	A medium integer. Signed range is from -8388608 to 8388607. Unsigned range is from 0 to 16777215. The size parameter specifies the maximum display width (which is 255)
<code>INT(size)</code>	A medium integer. Signed range is from -2147483648 to 2147483647. Unsigned range is from 0 to 4294967295. The size parameter specifies the maximum display width (which is 255)
<code>INTEGER(size)</code>	Equal to <code>INT(size)</code>
<code>BIGINT(size)</code>	A large integer. Signed range is from -9223372036854775808 to 9223372036854775807. Unsigned range is from 0 to 18446744073709551615. The size parameter specifies the maximum display width (which is 255)
<code>FLOAT(size, d)</code>	A floating point number. The total number of digits is specified in <i>size</i> . The number of digits after the decimal point is specified in the <i>d</i> parameter. This syntax is deprecated in MySQL 8.0.17, and it will be removed in future MySQL versions
<code>FLOAT(p)</code>	A floating point number. MySQL uses the <i>p</i> value to determine whether to use <code>FLOAT</code> or <code>DOUBLE</code> for the resulting data type. If <i>p</i> is from 0 to 24, the data type becomes <code>FLOAT()</code> . If <i>p</i> is from 25 to 53, the data type becomes <code>DOUBLE()</code>
<code>DOUBLE(size, d)</code>	A normal-size floating point number. The total number of digits is specified in <i>size</i> . The number of digits after the decimal point is specified in the <i>d</i> parameter
<code>DOUBLE PRECISION(size, d)</code>	
<code>DECIMAL(size, d)</code>	An exact fixed-point number. The total number of digits is specified in <i>size</i> . The number of digits after the decimal point is specified in the <i>d</i> parameter. The maximum number for <i>size</i> is 65. The maximum number for <i>d</i> is 30. The default value for <i>size</i> is 10. The default value for <i>d</i> is 0.
<code>DEC(size, d)</code>	Equal to <code>DECIMAL(size, d)</code>

Crear las Tablas

Date and Time Data Types

Data type	Description
DATE	A date. Format: YYYY-MM-DD. The supported range is from '1000-01-01' to '9999-12-31'
DATETIME(<i>fsp</i>)	A date and time combination. Format: YYYY-MM-DD hh:mm:ss. The supported range is from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'. Adding DEFAULT and ON UPDATE in the column definition to get automatic initialization and updating to the current date and time
TIMESTAMP(<i>fsp</i>)	A timestamp. TIMESTAMP values are stored as the number of seconds since the Unix epoch ('1970-01-01 00:00:00' UTC). Format: YYYY-MM-DD hh:mm:ss. The supported range is from '1970-01-01 00:00:01' UTC to '2038-01-09 03:14:07' UTC. Automatic initialization and updating to the current date and time can be specified using DEFAULT CURRENT_TIMESTAMP and ON UPDATE CURRENT_TIMESTAMP in the column definition
TIME(<i>fsp</i>)	A time. Format: hh:mm:ss. The supported range is from '-838:59:59' to '838:59:59'
YEAR	A year in four-digit format. Values allowed in four-digit format: 1901 to 2155, and 0000. MySQL 8.0 does not support year in two-digit format.

Crear las Tablas: departamento



The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' pane shows a tree view with 'universidad' expanded, containing 'Tables' and 'Indexes'. The 'departamento' table is listed under 'Tables'. The main editor shows a SQL query in 'Query 1':

```

1 • USE universidad;
2
3 • CREATE TABLE departamento (
4     depa_id INTEGER PRIMARY KEY,
5     nombre VARCHAR(30),
6     edificio INTEGER,
7     presupuesto DECIMAL(12,2));
8
9

```

Below the query, the 'Output' pane shows the execution results:

#	Time	Action
1	13:53:44	USE universidad
2	13:53:44	CREATE TABLE departamento (depa_id INTEGER PRIMARY



- Podemos borrar la Tabla usando `DROP TABLE IF EXISTS departamento`

Crear las Tablas: profesor

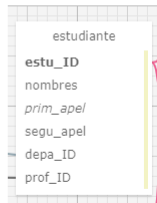
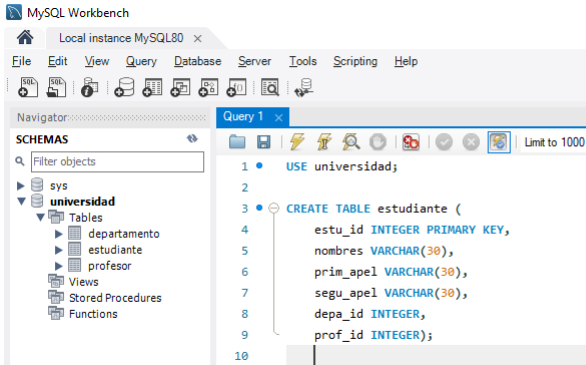
The screenshot displays the MySQL Workbench interface. On the left, the 'Navigator' pane shows the 'universidad' schema with a tree view containing 'departamento', 'profesor', 'Views', 'Stored Procedures', and 'Functions'. The 'profesor' table is highlighted. The main editor window, titled 'Query 1', contains the following SQL code:

```
1 • USE universidad;
2
3 • CREATE TABLE profesor (
4     prof_id INTEGER PRIMARY KEY,
5     nombres VARCHAR(30),
6     prim_apel VARCHAR(30),
7     segu_apel VARCHAR(30),
8     fech_naci DATE,
9     depa_id INTEGER);
10
```

To the right of the SQL editor, a diagram of the 'Profesor' table structure is shown, listing the following fields:

- prof_ID
- nombres
- prim_apel
- segu_apel
- fech_naci
- depa_ID

Crear las Tablas: estudiante



Crear las Tablas: curso

The screenshot displays the MySQL Workbench interface for a local instance named 'MySQL80'. The 'Navigator' pane on the left shows the 'universidad' database selected, with a tree view containing 'Tables', 'Views', 'Stored Procedures', and 'Functions'. The 'Tables' list includes 'curso', 'departamento', 'estudiante', and 'profesor'. The main editor window, titled 'Query 1', contains the following SQL code:

```
1 • USE universidad;  
2  
3 • CREATE TABLE curso (  
4     curso_id INTEGER PRIMARY KEY,  
5     nombre VARCHAR(30),  
6     credits INTEGER,  
7     depa_id INTEGER);  
8  
9
```

To the right of the query editor, a preview of the 'curso' table structure is shown, listing the columns: 'curso_ID', 'nombre', 'credits', and 'depa_ID'.

Crear las Tablas: pre_requ

MySQL Workbench

Local instance MySQL80 x

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- sys
- universidad
 - Tables
 - curso
 - departamento
 - estudiante
 - pre_requ
 - profesor
 - Views
 - Stored Procedures
 - Functions

Query 1 x

```
1 • USE universidad;  
2  
3 • CREATE TABLE pre_requ (  
4     curso_id_1 INTEGER,  
5     curso_id_2 INTEGER,  
6     PRIMARY KEY (curso_id_1, curso_id_2));  
7  
8
```

Limit to 1000 rows

pre_requ
curso_ID_1
curso_ID_2

Crear las Tablas: estudiante_tel

The screenshot displays the MySQL Workbench interface. On the left, the 'Navigator' pane shows the 'universidad' schema with a list of tables including 'estudiante_telefono'. The main editor window, titled 'Query 1', contains the following SQL code:

```
1 • USE universidad;  
2  
3 • CREATE TABLE estudiante_telefono (  
4     estu_id INTEGER,  
5     telefono INTEGER,  
6     PRIMARY KEY (estu_id, telefono));  
7  
8
```

To the right of the SQL editor, a preview of the 'estudiante_telefono' table structure is shown, listing the columns 'estu_ID' and 'telefono'.

Crear las Tablas: seccion

The screenshot displays the MySQL Workbench interface. On the left, the 'Navigator' pane shows the 'universidad' schema with a list of tables including 'seccion'. The main editor window, titled 'Query 1', contains the following SQL code:

```
1 • USE universidad;  
2  
3 • CREATE TABLE seccion (  
4     secc_id INTEGER,  
5     curso_id INTEGER,  
6     anho INTEGER,  
7     semestre INTEGER,  
8     PRIMARY KEY (secc_id, curso_id));  
9  
10
```

To the right of the query editor, a preview of the 'seccion' table structure is shown, listing the columns: secc_ID, curso_ID, anho, and semestre.

Crear las Tablas: profesor_seccion

MySQL Workbench

Local instance MySQL80 x

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- sys
- universidad
 - Tables
 - curso
 - departamento
 - estudiante
 - estudiante_telefono
 - pre_requ
 - profesor
 - profesor_seccion
 - seccion
 - Views
 - Stored Procedures
 - Functions

Query 1 x

```
1 • USE universidad;  
2  
3 • CREATE TABLE profesor_seccion (  
4     prof_id INTEGER,  
5     secc_id INTEGER,  
6     curso_id INTEGER,  
7     PRIMARY KEY (prof_id, secc_id, curso_id));  
8  
9
```

profesor_seccion

- prof_ID
- secc_ID
- curso_ID

Crear las Tablas: estudiante_seccion

The screenshot displays the MySQL Workbench interface. On the left, the 'Navigator' pane shows the 'universidad' database schema with a list of tables including 'curso', 'departamento', 'estudiante', 'estudiante_seccion', 'estudiante_telefono', 'pre_requ', 'profesor', 'profesor_seccion', and 'seccion'. The main editor window, titled 'Query 1', contains the following SQL code:

```
1 • USE universidad;
2
3 • CREATE TABLE estudiante_seccion (
4     estu_id INTEGER,
5     secc_id INTEGER,
6     curso_id INTEGER,
7     nota INTEGER,
8     PRIMARY KEY (estu_id, secc_id, curso_id));
9
10
```

To the right of the query editor, a preview of the 'estudiante_seccion' table structure is shown:

estudiante_seccion
estu_ID
secc_ID
curso_ID
nota

Modificar Tablas

- Una vez que se crea la tabla en la base de datos, existen muchas ocasiones donde uno puede desear cambiar la estructura de la tabla. Los casos típicos incluyen los siguientes:
 - Agregar una columna
 - Cambiar el nombre de una columna
 - Cambiar el tipo de dato de una columna
 - Eliminar una columna
 - Adicionar o eliminar una cable primaria
 - Adicionar o eliminar una cable foránea

Modificar Tablas

```

ALTER TABLE tbl_name
[alter_option [, alter_option] ...]
[partition_options]

alter_option: {
  table_options
| ADD [COLUMN] col_name column_definition
  [FIRST | AFTER col_name]
| ADD [COLUMN] (col_name column_definition,...)
| ADD [INDEX | KEY] [index_name]
  [index_type] (key_part,...) [index_option] ...
| ADD [FULLTEXT | SPATIAL] [INDEX | KEY] [index_name]
  (key_part,...) [index_option] ...
| ADD [CONSTRAINT [symbol]] PRIMARY KEY
  [index_type] (key_part,...)
  [index_option] ...
| ADD [CONSTRAINT [symbol]] UNIQUE [INDEX | KEY]
  [index_name] [index_type] (key_part,...)
  [index_option] ...
| ADD [CONSTRAINT [symbol]] FOREIGN KEY
  [index_name] (col_name,...)
  reference_definition
| ADD [CONSTRAINT [symbol]] CHECK (expr) [[NOT] ENFORCED]
| DROP [CHECK | CONSTRAINT] symbol
| ALTER [CHECK | CONSTRAINT] symbol [NOT] ENFORCED
| ALGORITHM [=] {DEFAULT | INSTANT | INPLACE | COPY}
| ALTER [COLUMN] col_name
  {SET DEFAULT {literal} | (expr)} | DROP DEFAULT}
| ALTER INDEX index_name {VISIBLE | INVISIBLE}
| CHANGE [COLUMN] old_col_name new_col_name column_definition
  [FIRST | AFTER col_name]
| [DEFAULT] CHARACTER SET [=] charset_name [COLLATE [=] collation_name]
| CONVERT TO CHARACTER SET charset_name [COLLATE collation_name]
| [DISABLE | ENABLE] KEYS
| {DISCARD | IMPORT} TABLESPACE
| DROP [COLUMN] col_name
| DROP [INDEX | KEY] index_name
| DROP PRIMARY KEY
| DROP FOREIGN KEY fk_symbol
| FORCE
| LOCK [=] {DEFAULT | NONE | SHARED | EXCLUSIVE}
| MODIFY [COLUMN] col_name column_definition
  [FIRST | AFTER col_name]
| ORDER BY col_name [, col_name] ...
| RENAME COLUMN old_col_name TO new_col_name
| RENAME [INDEX | KEY] old_index_name TO new_index_name
| RENAME [TO | AS] new_tbl_name
| {WITHOUT | WITH} VALIDATION
}

```

Modificar Tablas: Agregar una columna

MySQL Workbench

Local instance MySQL80 x

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- sys
- universidad
 - Tables
 - curso
 - departamento
 - estudiante
 - estudiante_seccion
 - Columns
 - estu_id
 - secc_id
 - curso_id
 - nota
 - porcentaje_asistencia
 - Indexes
 - Foreign Keys
 - Triggers

Query 1 x

```
1 • USE universidad;  
2  
3 • ALTER TABLE estudiante_seccion ADD porcentaje_asistencia INTEGER;  
4
```

Limit to 1000 rows

Modificar Tablas: Cambiar el nombre de una columna

MySQL Workbench

Local instance MySQL80 x

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- sys
- universidad
 - Tables
 - curso
 - departamento
 - estudiante
 - estudiante_seccion
 - Columns
 - estu_id
 - secc_id
 - curso_id
 - nota
 - porc_asis
 - Indexes

Query 1 x

Limit to 1000 rows

```
1 • USE universidad;
2
3 • ALTER TABLE estudiante_seccion RENAME COLUMN porcentaje_asistencia TO porc_asis;
4
```


Modificar Tablas: Cambiar el tipo de dato de una columna

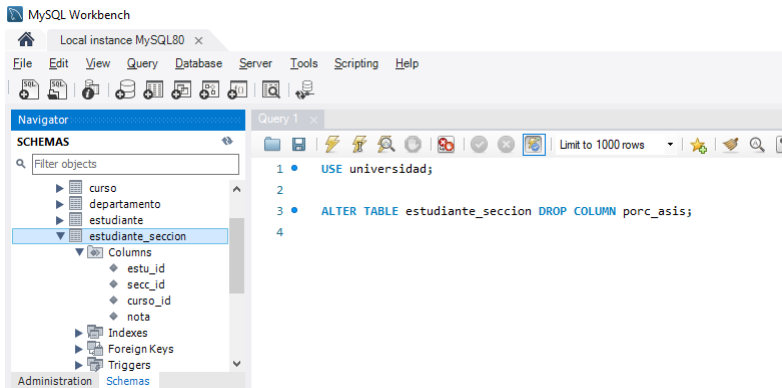
The screenshot shows the MySQL Workbench interface. In the left sidebar, the 'Navigator' pane displays the 'SCHEMAS' tree. The 'estudiante_seccion' table is selected, and its columns are listed: 'estu_id', 'secc_id', 'curso_id', 'nota', and 'porc_asis'. The 'porc_asis' column is highlighted. Below the Navigator, the 'Information' pane shows the table's structure: 'Columns: estu_id int PK, secc_id int PK, curso_id int PK, nota int, porc_asis varchar(2)'. The main query editor displays the following SQL code:

```
1 • USE universidad;
2
3 • ALTER TABLE estudiante_seccion MODIFY porc_asis VARCHAR(2);
4
5
```

The 'Output' pane at the bottom shows the execution results:

#	Time	Action
1	17:00:31	USE universidad
2	17:00:31	ALTER TABLE estudiante_seccion MODIFY porc_asis VARCHAR(2)

Modificar Tablas: Eliminar una columna



The screenshot displays the MySQL Workbench interface. On the left, the 'Navigator' pane shows the 'SCHEMAS' tree with 'estudiante_seccion' selected. Under 'Columns', the columns 'estu_id', 'secc_id', 'curso_id', and 'nota' are listed. The 'Indexes' section is also visible. The main query editor on the right shows a SQL script with two statements: 'USE universidad;' and 'ALTER TABLE estudiante_seccion DROP COLUMN porc_asis;'. The 'ALTER TABLE' statement is highlighted with a blue box. The interface includes a menu bar (File, Edit, View, Query, Database, Server, Tools, Scripting, Help) and a toolbar with various icons for file operations, execution, and navigation.

MySQL Workbench

Local instance MySQL80 x

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- curso
- departamento
- estudiante
- estudiante_seccion
 - Columns
 - estu_id
 - secc_id
 - curso_id
 - nota
 - Indexes
 - Foreign Keys
 - Triggers

Administration Schemas

Query 1 x

Limit to 1000 rows

```
1 • USE universidad;
2
3 • ALTER TABLE estudiante_seccion DROP COLUMN porc_asis;
4
```

Modificar Tablas: Eliminar una cable primaria

The screenshot shows the MySQL Workbench interface. In the left sidebar, the 'Schemas' pane is open, showing a tree view of the database structure. The 'estudiante_seccion' table is selected, and its columns (estu_id, secc_id, curso_id, nota) and indexes are visible. The 'Administration' tab is active, and the 'Schemas' sub-tab is selected. Below the tree view, the 'Table: estudiante_seccion' is displayed with its columns and data types.

In the center, the 'Query 1' editor shows the following SQL script:

```
1 • USE universidad;  
2  
3 • ALTER TABLE estudiante_seccion DROP PRIMARY KEY;  
4
```

On the right, the 'Output' pane shows the execution results of the SQL script. The 'Action Output' tab is selected, displaying a table with columns '#', 'Time', and 'Action'. The results show two successful actions:

#	Time	Action
✓ 1	17:06:25	USE universidad
✓ 2	17:06:25	ALTER TABLE estudiante_seccion DROP PRIMARY KEY

Modificar Tablas: Adicionar una cable primaria

MySQL Workbench

Local instance MySQL80 x

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- estudiante
 - estudiante_seccion
 - Columns
 - estu_id
 - secc_id
 - curso_id
 - nota
 - Indexes
 - PRIMARY
 - Foreign Keys
 - Triggers
 - estudiante_telefono
 - pre_requ
 - profesor
 - profesor_seccion

Administration Schemas

Information

Table: estudiante_seccion

Columns:

estu_id	int PK
secc_id	int PK
curso_id	int PK
nota	int

Query 1 x

Limit to 1000 rows

```
1 • USE universidad;
2
3 • ALTER TABLE estudiante_seccion ADD PRIMARY KEY (estu_id, secc_id, curso_id);
4
```

Output

Action Output

#	Time	Action
✓ 1	18:17:20	USE universidad
✓ 2	18:17:20	ALTER TABLE estudiante_seccion ADD PRIMARY KEY (estu_id, secc_id, curso_id)

Modificar Tablas: Adicionar una cable foránea

The screenshot displays the MySQL Workbench interface. On the left, the 'SCHEMAS' pane shows a tree view of the database structure. The 'Foreign Keys' folder is expanded, showing a list of foreign keys, including 'seccion_ibfk_1'. The 'Information' pane at the bottom provides details for the selected foreign key, including its definition and target table.

Foreign Key: seccion_ibfk_1

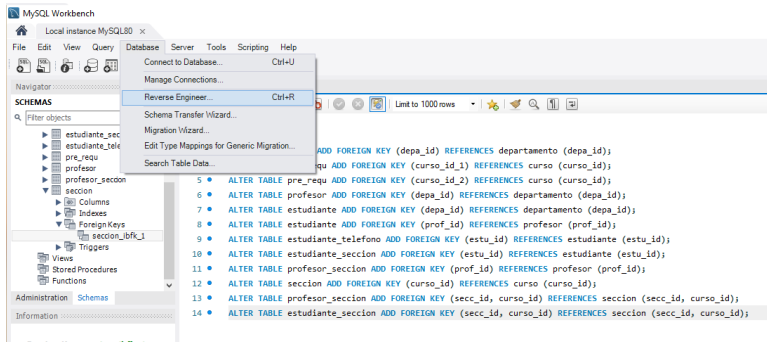
Definition:

Target	curso (curso_id →
On Update	RESTRICT
On Delete	RESTRICT

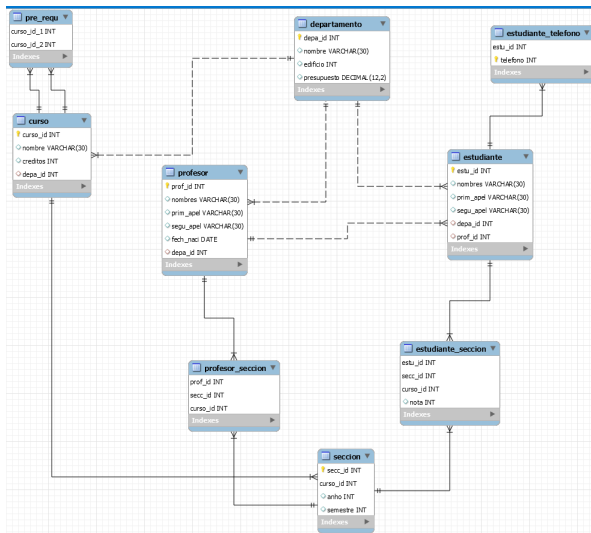
The main query editor on the right contains a list of SQL statements for creating and modifying the database schema. The statements are as follows:

```
1 • USE universidad;
2
3 • ALTER TABLE curso ADD FOREIGN KEY (depa_id) REFERENCES departamento (depa_id);
4 • ALTER TABLE pre_requ ADD FOREIGN KEY (curso_id_1) REFERENCES curso (curso_id);
5 • ALTER TABLE pre_requ ADD FOREIGN KEY (curso_id_2) REFERENCES curso (curso_id);
6 • ALTER TABLE profesor ADD FOREIGN KEY (depa_id) REFERENCES departamento (depa_id);
7 • ALTER TABLE estudiante ADD FOREIGN KEY (depa_id) REFERENCES departamento (depa_id);
8 • ALTER TABLE estudiante ADD FOREIGN KEY (prof_id) REFERENCES profesor (prof_id);
9 • ALTER TABLE estudiante_telefono ADD FOREIGN KEY (estu_id) REFERENCES estudiante (estu_id);
10 • ALTER TABLE estudiante_seccion ADD FOREIGN KEY (estu_id) REFERENCES estudiante (estu_id);
11 • ALTER TABLE profesor_seccion ADD FOREIGN KEY (prof_id) REFERENCES profesor (prof_id);
12 • ALTER TABLE seccion ADD FOREIGN KEY (curso_id) REFERENCES curso (curso_id);
13 • ALTER TABLE profesor_seccion ADD FOREIGN KEY (secc_id, curso_id) REFERENCES seccion (secc_id, curso_id);
14 • ALTER TABLE estudiante_seccion ADD FOREIGN KEY (secc_id, curso_id) REFERENCES seccion (secc_id, curso_id);
```

Modelo Físico

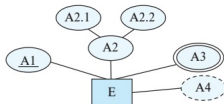


Modelo Físico



Modelo Físico

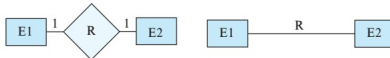
entity set E with
simple attribute A1,
composite attribute A2,
multivalued attribute A3,
derived attribute A4,
and primary key A1



many-to-many
relationship



one-to-one
relationship



many-to-one
relationship



participation
in R: total (E1)
and partial (E2)



weak entity set



generalization



total
generalization



Figure 6.27 Alternative E-R notations.

Modelo Físico

Cardinality



Zero or one



Many



One



One (and only one)



Zero or many

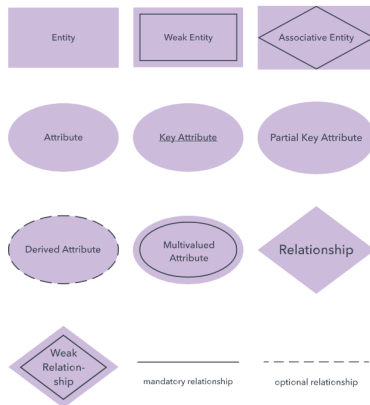


One or many

Modelo Físico

ERD symbols and notations

Chen notation style



Inserción de Datos

MySQL Workbench

Local instance MySQL80 x MySQL Model* x EER Diagram x

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- sys
- universidad
 - curso
 - departamento
 - Columns
 - depa_id
 - nombre
 - edificio
 - presupuesto
 - Indexes
 - PRIMARY
 - Foreign Keys
 - Triggers
 - estudiante
 - estudiante_seccion
 - estudiante_telefono
 - pre_requ
 - profesor
 - profesor_seccion
 - seccion
- Views
- Stored Procedures

Query 1 x

Limit to 1000 rows

```

3 • INSERT INTO departamento (depa_id, nombre, edificio, presupuesto)
4   VALUES (1, 'Computacion', 1, 4785.30);
5
6 • INSERT INTO departamento (depa_id, nombre, edificio, presupuesto)
7   VALUES (4, 'Ingenieria', 2, 1748.85);
8
9 • INSERT INTO departamento (depa_id, nombre, edificio, presupuesto)
10  VALUES (5, 'Biomedicas', 3, 21447.587);
11
12 • SELECT * FROM departamento;
13

```

Result Grid

	depa_id	nombre	edificio	presupuesto
▶	1	Computacion	1	4785.30
	4	Ingenieria	2	1748.85
	5	Biomedicas	3	21447.59
*	NULL	NULL	NULL	NULL

Inserción de Datos

MySQL Workbench

Local instance MySQL80 x MySQL Model* x EER Diagram x

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- sys
- universidad
 - Tables
 - curso
 - Columns
 - curso_id
 - nombre
 - creditos
 - depa_id
 - Indexes
 - Foreign Keys
 - Triggers
 - departamento
 - estudiante
 - estudiante_seccion
 - estudiante_telefono
 - pre_requ
 - profesor
 - profesor_seccion
 - seccion
 - Views
 - Stored Procedures
 - Functions

Query 1 x

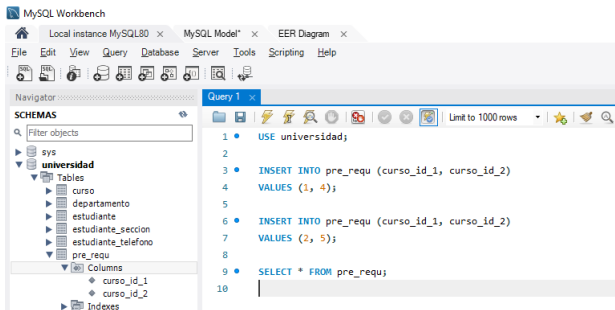
Limit to 1000 rows

```

1 • USE universidad;
2
3 • INSERT INTO curso (curso_id, nombre, creditos, depa_id)
4   VALUES (1, 'Base de Datos I', 4, 1);
5
6 • INSERT INTO curso (curso_id, nombre, creditos, depa_id)
7   VALUES (2, 'Inteligencia Artificial I', 6, 1);
8
9 • INSERT INTO curso (curso_id, nombre, creditos, depa_id)
10  VALUES (3, 'Ingenieria del Software', 5, 1);
11
12 • INSERT INTO curso (curso_id, nombre, creditos, depa_id)
13   VALUES (4, 'Base de Datos II', 2, 1);
14
15 • INSERT INTO curso (curso_id, nombre, creditos, depa_id)
16   VALUES (5, 'Inteligencia Artificial II', 3, 1);
17
18 • SELECT * FROM curso;
19

```

Inserción de Datos



Inserción de Datos

- El valor NULL representa a un valor desconocido.
- Este valor NULL puede ser asignado como valor a cualquier columna de una tabla.
- Si el valor de una columna es opcional, quiere decir, que podemos insertar una fila en la tabla sin asignarle ningún valor a esa columna opcional, así que esa columna tomará el valor NULL.
- El valor NULL es un valor especial, y por tanto, no se puede comparar con los operadores aritméticos normales, en su lugar debemos utilizar los operadores IS y IS NOT.
- Se puede omitir o cambiar el orden de inserción de las columnas.

Inserción de Datos

MySQL Workbench

Local instance MySQL80 x MySQL Model* x EER Diagram x

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- sys
 - universidad
 - Tables
 - curso
 - departamento
 - estudiante
 - Columns
 - estu_id
 - nombres
 - prim_apel
 - segu_apel
 - depa_id
 - prof_id
 - Indexes
 - Foreign Keys
 - Triggers
 - estudiante_seccion
 - estudiante_telefono
 - pre_requ
 - profesor
 - profesor_seccion
 - seccion
 - Views
 - Stored Procedures
 - Functions

Query 1 x

Limit to 1000 rows

```

1 • USE universidad;
2
3 • INSERT INTO profesor (prof_id, nombres, prim_apel, segu_apel, fech_naci, depa_id)
4   VALUES (1, 'Edward', 'Hinojosa', 'Cardenas', NULL, 1);
5
6 • INSERT INTO profesor
7   VALUES (2, 'Edgar', 'Sarmiento', 'Calisaya', '2020-12-26', 1);
8
9 • INSERT INTO profesor (depa_id, prof_id, prim_apel, nombres, segu_apel, fech_naci)
10  VALUES (1, 3, 'Gutierrez', 'Juan Carlos', 'Caceres', NULL);
11
12 • SELECT * FROM profesor;
13
14

```

Result Grid

	prof_id	nombres	prim_apel	segu_apel	fech_naci	depa_id
1	1	Edward	Hinojosa	Cardenas	NULL	1
2	2	Edgar	Sarmiento	Calisaya	2020-12-26	1
3	3	Juan Carlos	Gutierrez	Caceres	NULL	1
4	NULL	NULL	NULL	NULL	NULL	NULL

Inserción de Datos

MySQL Workbench

Local instance MySQL80 x MySQL Model x EER Diagram x

File Edit View Query Database Server Tools Scripting Help

Navigators

SCHEMAS

Filter objects

- sys
- universidad
 - Tables
 - curso
 - departamento
 - estudiante
 - Columns
 - estu_id
 - nombres
 - prim_apel
 - segu_apel
 - depa_id
 - prof_id
 - Indexes
 - Foreign Keys
 - Triggers
 - estudiante_seccion
 - estudiante_telefono
 - pre_requ
 - profesor
 - profesor_seccion
 - seccion
 - Views
 - Stored Procedures
 - Functions

Administration Schemas

Information

Column: **prof_id**

Query 1

```

1  USE universidad;
2
3  ALTER TABLE estudiante CHANGE depa_id depa_id INTEGER NOT NULL;
4  ALTER TABLE estudiante CHANGE prof_id prof_id INTEGER NOT NULL;
5
6  INSERT INTO estudiante VALUES (123456789, 'Juan', 'Perez', 'Rodriguez', 1, 1);
7  INSERT INTO estudiante VALUES (333445555, 'Frank', 'Velazquez', 'Flores', 1, 1);
8  INSERT INTO estudiante VALUES (999887777, 'Alice', 'Jimenez', 'Portugal', 1, 1);
9  INSERT INTO estudiante VALUES (987654321, 'Luisa', 'Santos', 'Ferrel', 1, 2);
10 INSERT INTO estudiante VALUES (666884444, 'Pedro', 'Lima', 'Maldonado', 1, 2);
11 INSERT INTO estudiante VALUES (453453453, 'Daniela', 'Acco', 'Olvarez', 1, 2);
12 INSERT INTO estudiante VALUES (987987987, 'Mateo', 'Vela', 'Marruecos', 1, 2);
13 INSERT INTO estudiante VALUES (888665555, 'Francisco', 'Linares', 'Gomez', 1, 1);
14
15 SELECT * FROM estudiante;
```

Result Grid

estu_id	nombres	prim_apel	segu_apel	depa_id	prof_id
123456789	Juan	Perez	Rodriguez	1	1
333445555	Frank	Velazquez	Flores	1	1
453453453	Daniela	Acco	Olvarez	1	2
666884444	Pedro	Lima	Maldonado	1	2
888665555	Francisco	Linares	Gomez	1	1
987654321	Luisa	Santos	Ferrel	1	2
987987987	Mateo	Vela	Marruecos	1	2
999887777	Alice	Jimenez	Portugal	1	1
NULL	NULL	NULL	NULL	NULL	NULL

Inserción de Datos

MySQL Workbench

Local instance MySQL80 x MySQL Model* x EER Diagram x

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

sys

universidad

Tables

curso

departamento

estudiante

estudiante_seccion

estudiante_telefono

Columns

estu_id

telefono

Indexes

Foreign Keys

Triggers

pre_requ

profesor

profesor_seccion

seccion

Views

Stored Procedures

Functions

Query 1

Limit to 1000 rows

```

1 • USE universidad;
2
3 • INSERT INTO estudiante_telefono VALUES (123456789, 999999999);
4 • INSERT INTO estudiante_telefono VALUES (123456789, 888888888);
5 • INSERT INTO estudiante_telefono VALUES (987654321, 555555555);
6 • INSERT INTO estudiante_telefono VALUES (987654321, 333333333);
7 • INSERT INTO estudiante_telefono VALUES (987654321, 777777777);
8
9 • SELECT * FROM estudiante_telefono;
10
11
12
13

```

Result Grid

estu_id	telefono
123456789	888888888
123456789	999999999
987654321	333333333
987654321	555555555
987654321	777777777
NULL	NULL

Inserción de Datos

MySQL Workbench

Local instance MySQL80 x MySQL Model* x EER Diagram x

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- sys
- universidad
 - Tables
 - curso
 - departamento
 - estudiante
 - estudiante_seccion
 - estudiante_telefono
 - pre_requ
 - profesor
 - profesor_seccion
 - seccion
 - Columns
 - secc_id
 - curso_id
 - anho
 - semestre
 - Indexes
 - Foreign Keys
 - Triggers
 - Views
 - Stored Procedures
 - Functions

Query 1 x

```
1 • USE universidad;
2
3 • INSERT INTO seccion VALUES (1, 1, 2019, 4);
4 • INSERT INTO seccion VALUES (2, 1, 2019, 4);
5 • INSERT INTO seccion VALUES (3, 3, 2020, 6);
6 • INSERT INTO seccion VALUES (4, 4, 2020, 5);
7 • INSERT INTO seccion VALUES (5, 5, 2020, 5);
8
9 • SELECT * FROM seccion;
10
11
12
13
```

Limit to 1000 rows

Result Grid

	secc_id	curso_id	anho	semestre
1	1	1	2019	4
2	1	1	2019	4
3	3	3	2020	6
4	4	4	2020	5
5	5	5	2020	5
*	HULL	HULL	HULL	HULL

Inserción de Datos

MySQL Workbench

Local instance MySQL80 x MySQL Model* x EER Diagram x

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

sys

universidad

Tables

curso

departamento

estudiante

estudiante_seccion

estudiante_telefono

pre_requ

profesor

profesor_seccion

Columns

prof_id

secc_id

curso_id

Indexes

Foreign Keys

Triggers

seccion

Views

Stored Procedures

Functions

Query 1 x

```
1 • USE universidad;
2
3 • INSERT INTO profesor_seccion VALUES (1, 1, 1);
4 • INSERT INTO profesor_seccion VALUES (2, 2, 1);
5 • INSERT INTO profesor_seccion VALUES (2, 3, 3);
6 • INSERT INTO profesor_seccion VALUES (3, 4, 4);
7
8 • SELECT * FROM profesor_seccion;
9
10
11
12
13
```

Limit to 1000 rows

Result Grid

Filter Rows:

Edit: Exp

	prof_id	secc_id	curso_id
▶	1	1	1
	2	2	1
	2	3	3
	3	4	4
▲	NULL	NULL	NULL

Inserción de Datos

MySQL Workbench

Local instance MySQL80 x MySQL Model x EER Diagram x

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

sys

universidad

Tables

curso

departamento

estudiante

estudiante_seccion

Columns

estu_id

secc_id

curso_id

nota

Indexes

Foreign Keys

Triggers

estudiante_telefono

pre_requ

profesor

profesor_seccion

seccion

Views

Stored Procedures

Functions

Administration Schemas

Information

Table: estudiante_seccion

Columns:

estu_id int PK

secc_id int PK

curso_id int PK

nota int

Query 1

```

1 USE universidad;
2
3 INSERT INTO estudiante_seccion VALUES (987654321, 1, 1, NULL);
4 INSERT INTO estudiante_seccion VALUES (123456789, 1, 1, 17);
5 INSERT INTO estudiante_seccion VALUES (333445555, 1, 1, 14);
6 INSERT INTO estudiante_seccion VALUES (453453453, 2, 1, 09);
7 INSERT INTO estudiante_seccion VALUES (987987987, 2, 1, 18);
8 INSERT INTO estudiante_seccion VALUES (123456789, 3, 3, 14);
9 INSERT INTO estudiante_seccion VALUES (333445555, 3, 3, NULL);
10 INSERT INTO estudiante_seccion VALUES (987654321, 3, 3, 13);
11 INSERT INTO estudiante_seccion VALUES (888665555, 3, 3, 11);
12 INSERT INTO estudiante_seccion VALUES (987654321, 4, 4, 19);
13 INSERT INTO estudiante_seccion VALUES (453453453, 4, 4, 12);
14 INSERT INTO estudiante_seccion VALUES (123456789, 4, 4, NULL);
15
16 SELECT * FROM estudiante_seccion;

```

Result Grid

estu_id	secc_id	curso_id	nota
123456789	1	1	17
123456789	3	3	14
123456789	4	4	NULL
333445555	1	1	14
333445555	3	3	NULL
453453453	2	1	9
453453453	4	4	12
888665555	3	3	11
987654321	1	1	NULL
987654321	3	3	13
987654321	4	4	19
987987987	2	1	18
NULL	NULL	NULL	NULL

Actualizar de Datos

- Es uno de los comandos con lo que debemos tener mucho cuidado.
- Es importante utilizar la cláusula WHERE junto al comando UPDATE.
- Si existieran múltiples filas que satisfacen la condición, todas ellas se modificarán. Por ello, debemos tener mucho cuidado.
- También es posible UPDATE múltiples columnas al mismo tiempo.

Actualizar la fecha de nacimiento de un profesor

MySQL Workbench

Local instance MySQL80 x MySQL Model* x EER Diagram x

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

sys

universidad

Tables

- curso
- departamento
- estudiante
- estudiante_seccion
- estudiante_telefono
- pre_requ
- profesor**

Columns

- prof_id
- nombres
- prim_apel
- segu_apel
- feh_naci
- depa_id

Query 1 x

```
1 • USE universidad;  
2  
3 • UPDATE profesor SET feh_naci = '1980-12-26' WHERE prof_id = 2;  
4  
5 • SELECT * FROM profesor;
```

Limit to 1000 rows

Result Grid

	prof_id	nombres	prim_apel	segu_apel	feh_naci	depa_id
▶	1	Edward	Hinojosa	Cardenas	NULL	1
	2	Edgar	Sarmiento	Calisaya	1980-12-26	1
	3	Juan Carlos	Gutierrez	Caceres	NULL	1
*	NULL	NULL	NULL	NULL	NULL	NULL

Actualizar la apellido y fecha de nacimiento de un profesor

MySQL Workbench

Local instance MySQL80 x MySQL Model* x EER Diagram x

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

sys

universidad

Tables

curso

departamento

estudiante

estudiante_seccion

estudiante_telefono

pre_requ

profesor

Columns

prof_id

nombres

prim_apel

segu_apel

fech_naci

depa_id

Indexes

Foreign Keys

Triggers

Query 1 x

Limit to 1000 rows

```

1 • USE universidad;
2
3 • UPDATE profesor SET segu_apel = 'M', fech_naci = '1983-10-17' WHERE prof_id = 3;
4
5 • SELECT * FROM profesor;
6
7
8

```

Result Grid

	prof_id	nombres	prim_apel	segu_apel	fech_naci	depa_id
▶	1	Edward	Hinojosa	Cardenas	NULL	1
	2	Edgar	Sarmiento	Calisaya	1980-12-26	1
	3	Juan Carlos	Gutierrez	M	1983-10-17	1
*	NULL	NULL	NULL	NULL	NULL	NULL

Actualizar la cantidad de créditos en cursos

MySQL Workbench

Local instance MySQL80 x MySQL Model* x EER Diagram x

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- sys
- universidad
 - Tables
 - curso
 - Columns
 - curso_id
 - nombre
 - creditos
 - depa_id
 - Indexes
 - Foreign Keys
 - Triggers
 - departamento
 - estudiante
 - estudiante_seccion
 - estudiante_telefono
 - pre_requ
 - profesor
 - profesor_seccion

Query 1

```
1 • USE universidad;  
2  
3 • UPDATE curso SET creditos = 4 WHERE curso_id = 3 OR curso_id = 5;  
4  
5 • SELECT * FROM curso;
```

Limit to 1000 rows

Result Grid

curso_id	nombre	creditos	depa_id
1	Base de Datos I	4	1
2	Inteligencia Artificial I	6	1
3	Ingenieria del Software	4	1
4	Base de Datos II	2	1
5	Inteligencia Artificial II	4	1
* NULL	NULL	NULL	NULL

Eliminar Datos

- Tener MUCHISIMO cuidado.
- DELETE: Este comando elimina las filas de una tabla. Podemos utilizar la clausula WHERE para filtrar las filas que necesitemos eliminar. Tarea.
- TRUNCATE: Este comando elimina todas las filas de una tabla. Tarea.
- Considerar las relaciones antes de borrar los registros.

¡GRACIAS!

