Contenido

[Integridad de datos 1](#_Toc92704365)

[Restricciones en create table 3](#_Toc92704366)

[La clave primaria. La restricción primary key 6](#_Toc92704367)

[Claves ajenas. La restricción foreign key 9](#_Toc92704368)

[Valores por defecto. La especificación default 15](#_Toc92704369)

[Verificación de condiciones. La restricción check 16](#_Toc92704370)

[La restricción unique 19](#_Toc92704371)

[El motor de almacenamiento 22](#_Toc92704372)

CREAR UNA BASE DE DATSOS:

**https://dev.mysql.com/doc/refman/5.7/en/creating-database.html**

mysql> CREATE DATABASE empresa;

La creación de una base de datos no la selecciona para su uso; debes hacerlo explícitamente. Para hacer empresa base de datos actual, use esta declaración:

mysql> USE empresa;

# ***Integridad de datos***

Cuando almacenamos datos en una tabla, se ajustan una serie de restricciones predeterminadas, por ejemplo, una columna no puede ser nula, una cadena sólo pueda almacenar datos en mayúscula, etc.

La integridad hace referencia al hecho de que os datos de la base de datos han de ajustarse a restricciones antes de almacenarse en ella.. Así pues, una restricción de integridad será una regla que restringe el rango de valores para una o más columnas en la tabla.

Si se produce cualquier fallo mientras un usuario está cambiando los datos en la base de datos, ésta tiene la capacidad de deshacer o cancelar cualquier transacción sospechosa.

Existe otro tipo de integridad, que es la integridad referencial, la cual, la cual garantiza que los valores de una columna ( o columnas) de una tabla dependan de los valores de otra columna o columnas de otra tabla. Así por ejemplo, si tenemos dos tabla (ventas y artículos) y se define integridad referencial para esas tablas, nunca se dará la situación de insertar una venta con un artículo inexistente.

***Una clave es un conjunto de uno o más atributos que, considerados conjuntamente, nos permiten identificar de forma única a una entidad en el Conjunto de entidades.***

***En el caso del conjunto de entidades Alumno el atributo DNI puede funcionar como clave, ya que cada entidad dentro del conjunto de entidades tiene un DNI distinto.***

***En un conjunto de entidades pueden existir más de una clave. A todas las posibles claves existentes se las denominará Claves Candidatas. Se usará el término Clave Primaria para denotar una clave candidata que elige el diseñador de la BD como medio principal de identificar entidades dentro de un conjunto de entidades.***

***Clave Ajena (foreing key) conjunto de atributos de una entidad que son clave primaria en otra entidad.***

**TIPOS DE DATOS RECORDAMOS:**

[**https://dev.mysql.com/doc/refman/5.7/en/data-types.html**](https://dev.mysql.com/doc/refman/5.7/en/data-types.html)

* [TINYINT[(***M***)] [UNSIGNED] [ZEROFILL]](https://dev.mysql.com/doc/refman/5.7/en/integer-types.html)

Un número entero muy pequeño. El rango con signo es -128a 127. El rango sin firmar es 0hasta 255.

* [SMALLINT[(***M***)] [UNSIGNED] [ZEROFILL]](https://dev.mysql.com/doc/refman/5.7/en/integer-types.html)

Un pequeño entero. El rango con signo es -32768a 32767. El rango sin firmar es 0hasta 65535.

* [MEDIUMINT[(](https://dev.mysql.com/doc/refman/5.7/en/integer-types.html" \o "11.1.2 Tipos de enteros (valor exacto): INTEGER, INT, SMALLINT, TINYINT, MEDIUMINT, BIGINT)***[M](https://dev.mysql.com/doc/refman/5.7/en/integer-types.html" \o "11.1.2 Tipos de enteros (valor exacto): INTEGER, INT, SMALLINT, TINYINT, MEDIUMINT, BIGINT)***[)] [UNSIGNED] [ZEROFILL]](https://dev.mysql.com/doc/refman/5.7/en/integer-types.html" \o "11.1.2 Tipos de enteros (valor exacto): INTEGER, INT, SMALLINT, TINYINT, MEDIUMINT, BIGINT)

Un número entero de tamaño mediano. El rango con signo es -8388608a 8388607. El rango sin firmar es 0hasta 16777215.

* [INT[(](https://dev.mysql.com/doc/refman/5.7/en/integer-types.html" \o "11.1.2 Tipos de enteros (valor exacto): INTEGER, INT, SMALLINT, TINYINT, MEDIUMINT, BIGINT)***[M](https://dev.mysql.com/doc/refman/5.7/en/integer-types.html" \o "11.1.2 Tipos de enteros (valor exacto): INTEGER, INT, SMALLINT, TINYINT, MEDIUMINT, BIGINT)***[)] [UNSIGNED] [ZEROFILL]](https://dev.mysql.com/doc/refman/5.7/en/integer-types.html" \o "11.1.2 Tipos de enteros (valor exacto): INTEGER, INT, SMALLINT, TINYINT, MEDIUMINT, BIGINT)

Un número entero de tamaño normal. El rango con signo es -2147483648a 2147483647. El rango sin firmar es 0hasta 4294967295.

* [INTEGER[(](https://dev.mysql.com/doc/refman/5.7/en/integer-types.html" \o "11.1.2 Tipos de enteros (valor exacto): INTEGER, INT, SMALLINT, TINYINT, MEDIUMINT, BIGINT)***[M](https://dev.mysql.com/doc/refman/5.7/en/integer-types.html" \o "11.1.2 Tipos de enteros (valor exacto): INTEGER, INT, SMALLINT, TINYINT, MEDIUMINT, BIGINT)***[)] [UNSIGNED] [ZEROFILL]](https://dev.mysql.com/doc/refman/5.7/en/integer-types.html" \o "11.1.2 Tipos de enteros (valor exacto): INTEGER, INT, SMALLINT, TINYINT, MEDIUMINT, BIGINT)

Este tipo es sinónimo de [INT](https://dev.mysql.com/doc/refman/5.7/en/integer-types.html).

* [BIGINT[(](https://dev.mysql.com/doc/refman/5.7/en/integer-types.html" \o "11.1.2 Tipos de enteros (valor exacto): INTEGER, INT, SMALLINT, TINYINT, MEDIUMINT, BIGINT)***[M](https://dev.mysql.com/doc/refman/5.7/en/integer-types.html" \o "11.1.2 Tipos de enteros (valor exacto): INTEGER, INT, SMALLINT, TINYINT, MEDIUMINT, BIGINT)***[)] [UNSIGNED] [ZEROFILL]](https://dev.mysql.com/doc/refman/5.7/en/integer-types.html" \o "11.1.2 Tipos de enteros (valor exacto): INTEGER, INT, SMALLINT, TINYINT, MEDIUMINT, BIGINT)

Un entero grande. El rango con signo es -9223372036854775808a 9223372036854775807. El rango sin firmar es 0hasta 18446744073709551615.

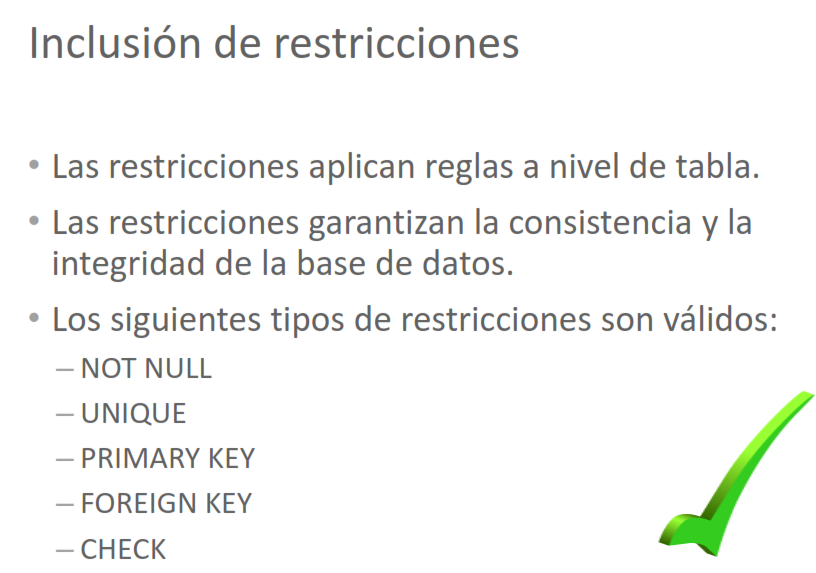
SERIALes un alias de BIGINT UNSIGNED NOT NULL AUTO\_INCREMENT UNIQUE.

**Mas información en el manual**

## **Restricciones en create table**

<https://dev.mysql.com/doc/refman/5.7/en/create-table.html>

La orden create table permite definir distintos tipos de restricciones sobre una tabla, clave primaria, claves ajenas, obligatoriedad, valores por defecto y verificación de condiciones. Para definir las restricciones en la orden create table usamos la cláusula constraint. Una cláusula constraint puede restringir una sola columna (**restricción de columna**) o un grupo de columnas de una misma tabla (**restricción de tabla**). Hay dos modos de especificar restricciones: como parte de la definición de columnas (una restricción de columna) o al final, una vez especificadas todas las columnas (una restricción de tabla).



A continuación parece el formato de la orden create table con restricciones de columna. La restricción forma parte de la definición de columna.

**Create table nombretabla (**

**columna1 tipo de dato**

**[constraint nombrerestricción]**

**[not null]**

**[unique]**

**[primary key]**

**[default valor]**

**[references nombretabla [ (columna [,columna] ) ] [on delete cascade] ]**

**[check condición] ,**

**columna2 tipo de dato**

**[constraint nombrerestricción]**

**[not null]**

**[unique]**

**[primary key]**

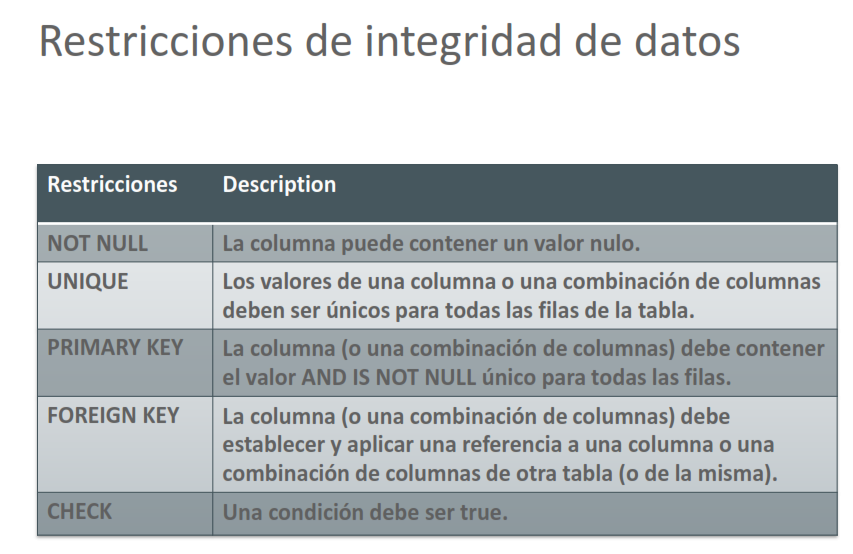
**[default valor]**

**[references nombretabla [ (columna [,columna] ) ] [on delete cascade] ]**

**[check condición],**

**................**

**);**

****

Ejemplo

**Create table USUARIOS**

**(**

**nombre varchar(25) primary key,**

**edad tinyint(2) check (edad between 18 and 40),**

**cod\_provi tinyint(2) references provincias (cod\_provincia)**

**);**

aquí se definen las siguientes restricciones

**clave primaria** nombre

**clave ajena** cod\_privi, que referencia a la tabla provincias

en el campo edad, verificación de ***condición check***( veremos más adelante), ha de estar comprendida entre 18 y 40.

Las restricciones de la orden create table que aparecen al final de la definición de las columnas (o tabla) se diferencian de la anterior en que se puede hacer referencia a varias columnas en una única restricción (por ejemplo, declarando dos columnas como clave primaria o ajena).

**Create table nombretabla**

**(**

**columna1 tipo de dato,**

**columna2 tipo de dato,**

**columna3 tipo de dato,**

**....................................**

**[constraint nombre restricción]**

**{[unique] | [primary key] (columna [,columna])}**

**[constraint nombre restricción]**

**[foreing key (columna [,columna]) references nombretabla [(columna [,columna])]**

**[on delete cascade]],**

**[constraint nombre restricción]**

**[check (condición)]**

**....................................**

**);**

**Create table T1**

**(**

**nombre varchar(25),**

**edad tinyint(2),**

**cod\_provi tinyint(2),**

**constraint keynom primary key (nombre),**

**constraint chedad check (edad between 18 and 40),**

**constraint foranipro foreign key (cod\_provi) references provincias(provincias)**

**);**

El nombre de las restricciones es opcional, también es válida esta sentencia create table

**Create table T1**

**(**

**nombre varchar(25),**

**edad tinyint(2),**

**cod\_provi tinyint(2),**

**primary key (nombre),**

**check (edad between 18 and 40),**

**foreign key (cod\_provi) references provincias(cod\_provincias)**

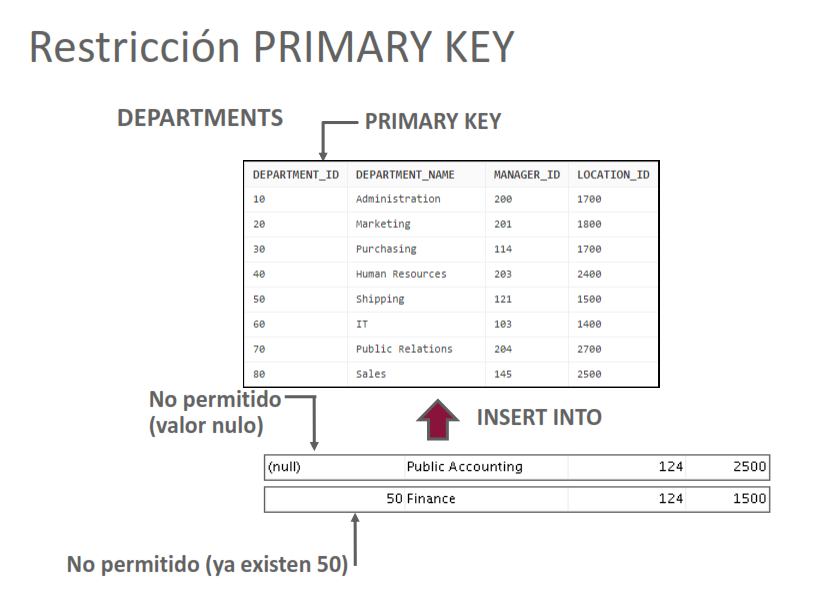
**);**

VAMOS UNA A UNA

## **La clave primaria. La restricción primary key**

Una clave primaria dentro de una tabla es una columna o conjunto de columnas que identifican unívocamente a cada fila. Debe ser única, no nula y obligatoria. Como máximo podemos definir una clave primaria por tabla. Esta clave se puede referenciar por una columna o columnas de otra tabla; llamamos clave ajena a esta columna o columnas.

Cuando se crea una clave primaria, automáticamente se crea un índice que facilita el acceso a la tabla. Para definir una clave primaria en una tabla usamos la restricción **primary key**.



Los formatos de la orden create table para definir claves primarias es:

Formato de restricción de columna:

**Create table nombretabla**

**(**

**colum1 tipo de dato [constaint nombrerestriccón] primary key,**

**colum2 tipo de dato**

**..............................**

**);**

Formato de restricción de tabla

**Create table nombretabla**

**(**

**colum1 tipo de dato,**

**colum2 tipo de dato,**

**.................................**

**[constraint nombrerestricción] primary key (columna[,columna]),**

**...........................**

**);**

Ejemplo, creamos una tabla llamada pisos con las siguientes columnas.

|  |  |  |  |
| --- | --- | --- | --- |
| **Nombre columna** | **Representa** | **Tipo de dato** | **create table pisos**  **(**  **calle varchar(30) not null,**  **numero tinyint(3) not null,**  **piso tinyint(2) not null,**  **puerta char(1) not null,**  **codigo postal tinyint(5),**  **metros tinyint(5),**  **comentarios varchar(60),**  **zona tinyint(2),**  **dni tinyint(10),**  **primary key (calle, numero, piso, puerta)**  **);** |
| Calle | Calle donde está el bloque | Varchar(30) |
| Numero | Número donde está el bloque | Tinyint(3) |
| Piso | Número planta | Tinyint(2) |
| Puerta | Puerta | Char(1) |
| Codigo postal | Codigo postal de la ciudad | Tinyint(5) |
| Metros | Metros cuadrados | Tinyint(5) |
| Comentarios | Otros datos de la vivienda | Varchar(60) |
| Zona | Zona donde está el bloque | Tinyint(2) |
| Dni | Deni del propietario | Tinyint(10) |

La clave de esta tabla es la combinación de calle, número, piso y puerta, que están especificadas como not null. Esto permite evitar la introducción de datos enla tabla sin dar valores a determinadas columnas. Si no ponemos not null en alguno de los atributos de la clave, oracle lo pone automá-ticamente en ese atributo***.(CUANDO EN LA ORDEN CREATE TABLE APARECE LA CLÁUSULA PRIMARY KEY, SÓLO DEBE ESPECIFICARSE UNA VEZ)***

Si en el ejemplo anterior, decidimos que el dni sea la clave primaria y la definimos de esta forma:

**create table pisos**

**(**

**dni varchar(10) primary key,**

**calle varchar(30) not null,**

**numero tinyint not null,**

**piso tinyint not null,**

**puerta varchar(1) not null,**

**código\_ p varchar(5),**

**metros tinyint(5),**

**comentarios varchar(60),**

**zona tinyint(2)**

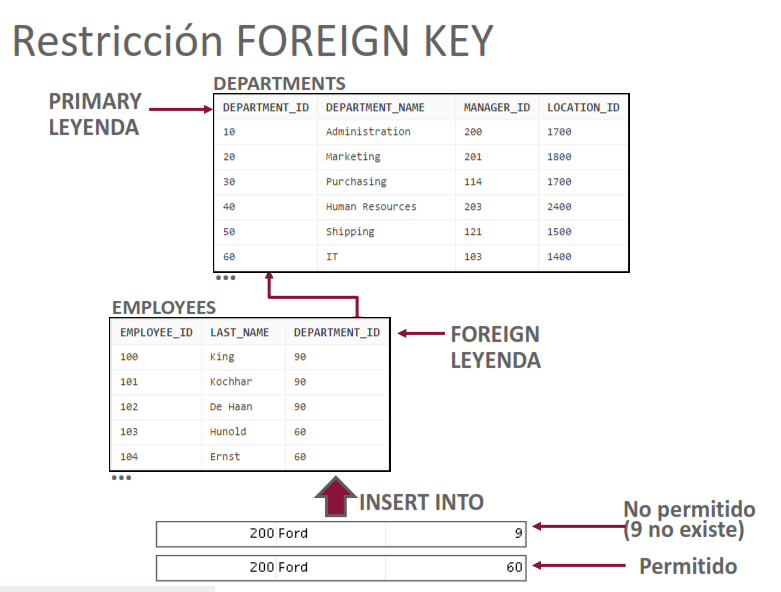
**);**

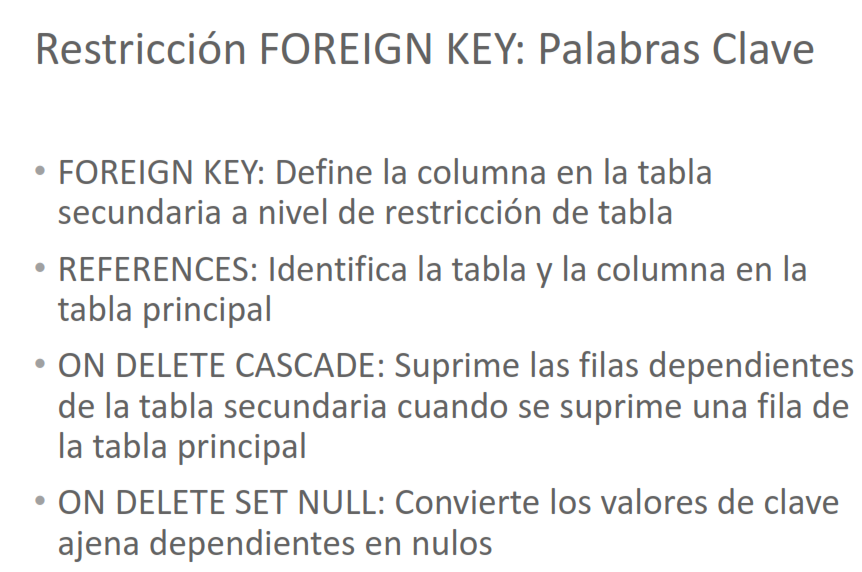
La clave la definimos formando parte de la columna, se dice entonces que es restricción de columna.

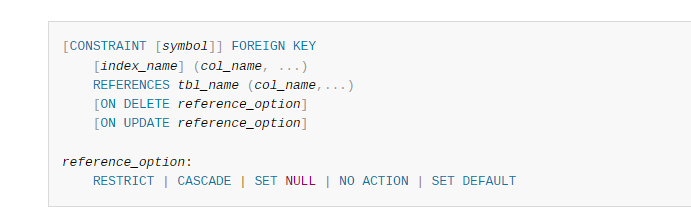
## **Claves ajenas. La restricción foreign key**

<https://dev.mysql.com/doc/refman/5.7/en/create-table-foreign-keys.html>

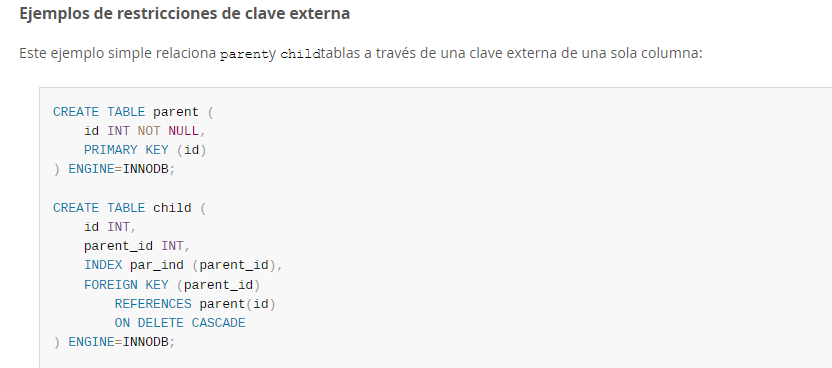
Una clave ajena está formada por una o varias columnas que están asociadas a una clave primaria de otra o de la misma tabla. Se puede definir tantas claves ajenas como sea preciso y pueden estar o no en la misma tabla que la clave primaria. El valor de la columna o columnas que son claves ajenas debe ser NULL o igual a un valor de la clave referenciada (regla de integridad referencial)







Por ejemplo:



Otro ejemplo:

Sean las tabla personas y provincias. La tabla personas contiene datos sobre las personas de una comunidad, mientras que la tabla provincias contiene el código y nombre de cada provincia.

|  |  |
| --- | --- |
| Tabla personas | **Tabla provincias** |
| Create table personas  (  dni varchar (9)primary key,  nombre varchar(15),  dire varchar(20),  pobla varchar(20),  codprovin tinyint(2) not null references provincias(codprovin)  ); | **Create table provincias**  **(**  **Codprovin timyint primary key,**  **nombreprovin varchar(15)**  **);** |

Dado que personas necesita información de la tabla provincias, se definen las siguientes restricciones para estas tabla.

Atributo dni es la clave primaria de la tabla personas.

Codprovin de la tabla provincias es la clave primaria de dicha tabla.

Codprovin de la tabla personas es la clave ajena, porque se relaciona con la clave primaria de la tabla provincias a traves de la orden references nombre tabla referenciada. La referencia la podíamos haber hecho con la orden foreign key.

En la tabla personas, se podría haber definido la clave ajena con la cláusula foreign key d ela siguiente forma:

**Create table personas**

**(**

**dni tinyint(10) primary key,**

**nombre varchar(15),**

**dire varchar(20),**

**pobla varchar(20),**

**codprovin tinyint(2) not null,**

**foreign key (codprovin) references provincias(codprovin)**

**);**

Existen varias formas de definir las claves ajenas (nosotros nos quedaremos con la vista hasta ahora)

En la cláusula references indicamos la tabla a la cual remite la clave ajena, a la derecha de foreign key y entre paréntesis indicamos la columna o columnas que forman parte de la clave ajena.

***Hemos de crear en primer lugar, la tabla provincias y después la tabla personas, ya que personas refencia a provincias.***

Si creamos primero la tabla personas y la tabla provincias no está creada, oracle dará un error:

**Foreign key (nombrecolumna) references nombretabla**

**Error en línea nº X**

**Ora-022942: la tabla o vista no existe**

Si queremos borrar las tablas, comenzaremos borrando la tabla personas y después la tabla provincias. Si intentamos borrar la tabla provincias antes que la tabla personas, oracle dará un error.

sql > drop table provincias; nombretabla

Para borrar una tabla drop table nombretabla;

error en la línea X:

ora-02449: claves ajenas hacen referencia a las claves única/primaria de la tabla.

Si lo que se desea es borrar alguna provincia de la tabla provincias y que las filas correspondientes dela tabla personas con esa provincia sean eliminadas, se añade la cláusula **on delete cascade** en la opción references.

**Create table personas**

**(**

**Dni varchar(10) primary key,**

**nombre varchar(15),**

**dire varchar(20),**

**pobla varchar(20),**

**codprovin tin(2) not null,**

**foreign key (codprovin) references provincias(codprovin) on delete cascade**

**);**

Oracle borrará la fila dependiente cuando se borre la fila correspondiente en la tabla origen. Esta acción mantiene automáticamente la integridad referencial.

Ejemplo

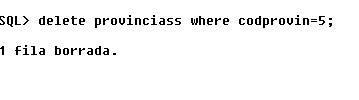
Vamos a crear unas tabla que se tengan que referenciar.

foreign key (codprovin) references provincias on delete cascade

**on delete cascade**

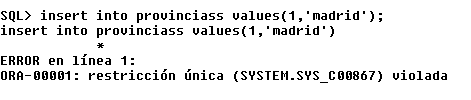
Estas dos tablas tienen una integridad referencial, hagamos una prueba.

Borremos de la tabla provincias, la provincia 5.



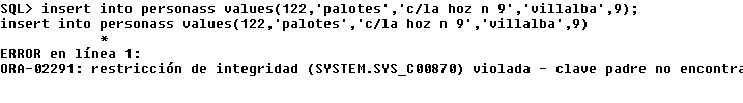
Como de puede comprobar, al borrar de la tabla principal (provincias) una información, queda actualizada la tabla que hace referencia (personas).

También hay que destacar que no podemos dar de alta a una clave primaria que ya exista. Ejemplo:



Indicándonos que ya existe esa clave.

Lo mismo sucede si se pretende dar de alta una fila en la tabla personas a una provincia que no tiene referencia en la tabla provincias.

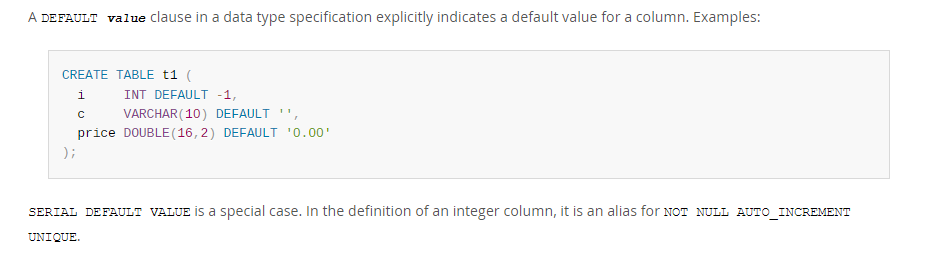


## **Valores por defecto. La especificación default**

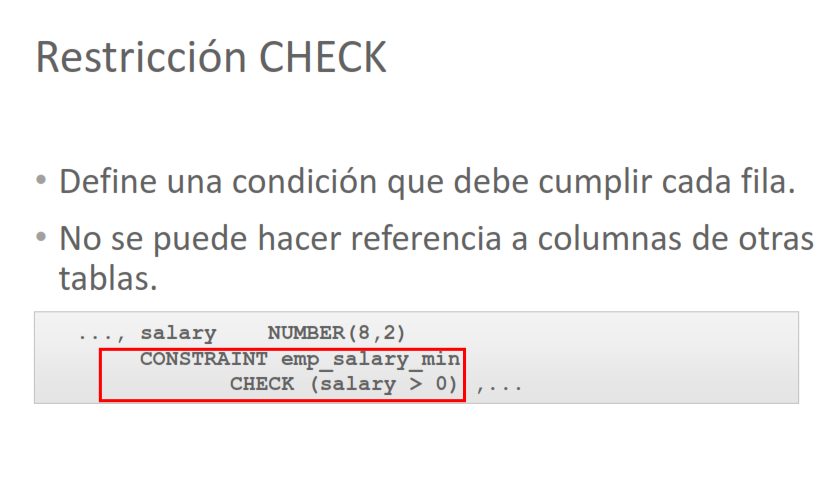
En el momento de crear una tabla podemos asignar valores por defecto a las columnas. Si especificamos la cláusula defautl a una columna, le proporcionamos un valor por omisión cuando el valor de la columna no se especifica en la cláusula insert.

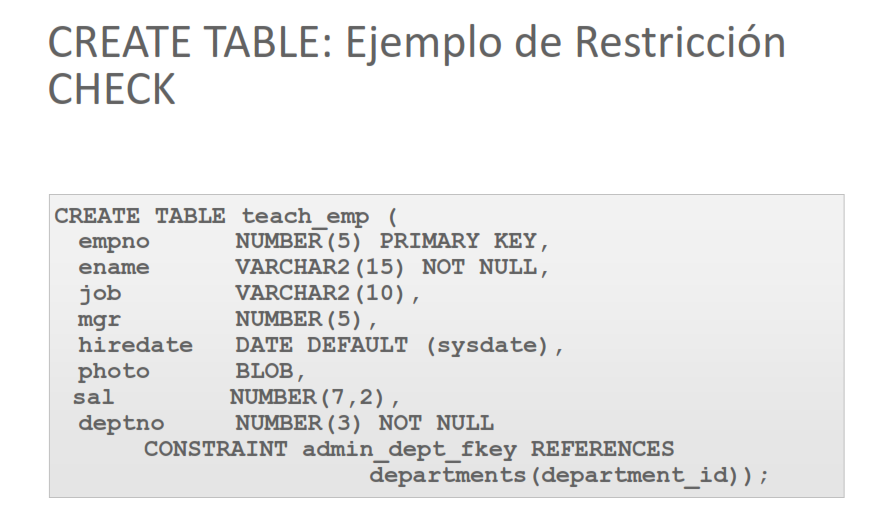
<https://dev.mysql.com/doc/refman/5.7/en/data-type-defaults.html>

Quedando la tabla de la forma:



## **Verificación de condiciones. La restricción check**





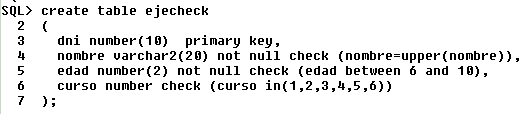
Muchas columnas de tablas requieren valores limitados dentro de un rango o el cumplimiento de ciertas condiciones. Con una restricción de verificación de condiciones se puede expresar una condición que ha de cumplirse para todas y cada una de las filas de la tabla.

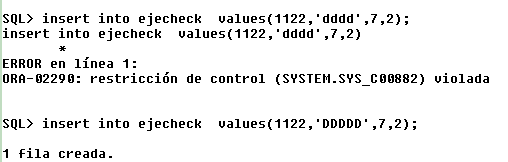
La restricción check actúa como una cláusula where. Puede hacer referencia a una o más columnas, pero no a valores de otras filas. En una cláusula check no cabe incluir subconsultas ni las seudocolumnas sysdate , uid y user

Se desea crear una tabla con los siguientes datos:

Dni, nombre, edad, curso. Con las siguientes restricciones.

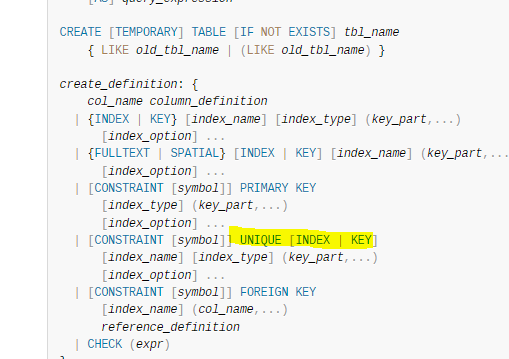
El dni no puede estar sin valor, la clave primaria será dni, el nombre no puede estar vacío, la edad ha de estar comprendida entre 6 y 10 años, el nombre ha de estar en mayúsculas, el curso sólo puede almacenar 1, 2, 3, 4, 5 ó 6.

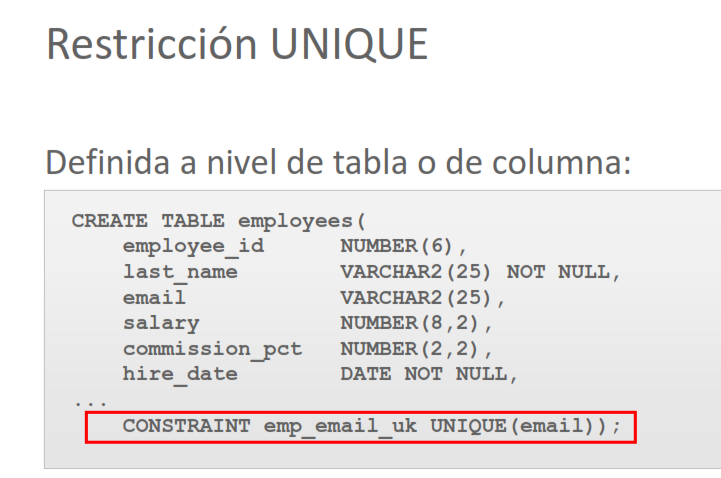




Se ha producido error por no introducir el nombre en mayúscula, eso es una restricción.

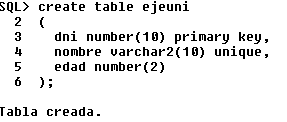
## **La restricción unique**





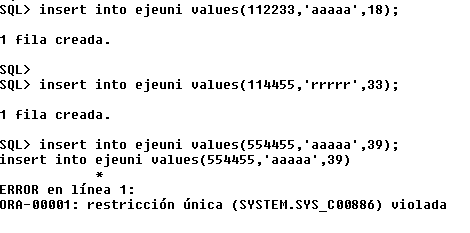
La restricción unique evita los valores repetidos en una misma columna. Puede contener una o varias columnas .

PE, creamos la siguiente tabla.



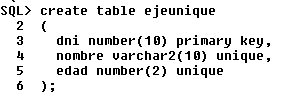
Donde la columna nombre se ha restringido como única.

Insertamos valores, y un valor se pretende almacenar un nombre que ya existe.

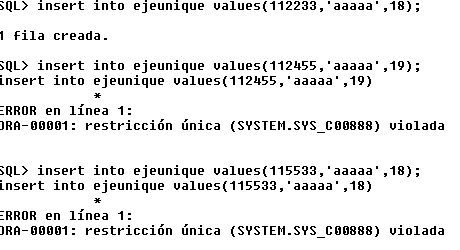


Dando su correspondiente error.

También podemos poner dos columnas ( o varias) con unique.



Al intentar dar datos repetidos en alguna de columna, nos dará error.



## **El motor de almacenamiento**

https://dev.mysql.com/doc/refman/5.7/en/myisam-storage-engine.html

MYISAM;

CREATE TABLE t (i INT) ENGINE = MYISAM;

InnoDB and the ACID Model

The [ACID](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_acid) model is a set of database design principles that emphasize aspects of reliability that are important for business data and mission-critical applications. MySQL includes components such as the InnoDB storage engine that adhere closely to the ACID model so that data is not corrupted and results are not distorted by exceptional conditions such as software crashes and hardware malfunctions. When you rely on ACID-compliant features, you do not need to reinvent the wheel of consistency checking and crash recovery mechanisms. In cases where you have additional software safeguards, ultra-reliable hardware, or an application that can tolerate a small amount of data loss or inconsistency, you can adjust MySQL settings to trade some of the ACID reliability for greater performance or throughput.

The following sections discuss how MySQL features, in particular the InnoDB storage engine, interact with the categories of the ACID model:

* ***A***: atomicity.
* ***C***: consistency.
* ***I:***: isolation.
* ***D***: durability.

