UNIDAD 1: Estructura SQL – Estructura NoSQL

1. [UNIDAD 1: Introducción](https://learn.nextu.com/mod/page/view.php?id=3553&pid=P_WEBDEV_V2)

Bienvenido a Interactuando con Bases de Datos

En esta Unidad aprenderás qué es una base de datos y cómo funciona. En computación llamamos base de datos a un conjunto de datos que tienen una estructura organizacional que se guarda en un servidor. Estos servidores cuentan con un sistema gestor de base de datos que permite a los usuarios acceder a la información que se esté almacenando desde diferentes computadores. Además entenderás, con detalle, qué es una base de datos relacional y no relacional e instalarás un Sistema Gestor de Base de Datos orientado a objetos y relaciones llamado PostgreSQL.

Puntos de aprendizaje

**Unidad 1: Estructura SQL – Estructura NoSQL**  
¡Aquí comienza la Unidad 1: Estructura SQL – Estructura NoSQL!

Los objetivos del Programa que corresponden a esta Unidad son:

* **Usar metodologías sincrónicas y asincrónicas para gestionar datos desde el servidor.**
* **Crear ambientes de programación y a gestionar recursos.**

Los objetivos del Curso 7 que corresponden a esta Unidad son:

* **Aprender qué es y cómo funciona una base de datos.**
* **Asimilar, diseñar y crear bases de datos SQL.**
* **Entender, diseñar y construir bases de datos NoSQL.**

1. Lección 1: ¿Qué es una base de datos
   1. [Definición de base de datos](https://learn.nextu.com/mod/lesson/view.php?id=3554&pid=P_WEBDEV_V2)

Conjunto de datos que tienen una estructura organizacional que se guarda en un servidor 🡪 Los servidores cuentan con un gestor.

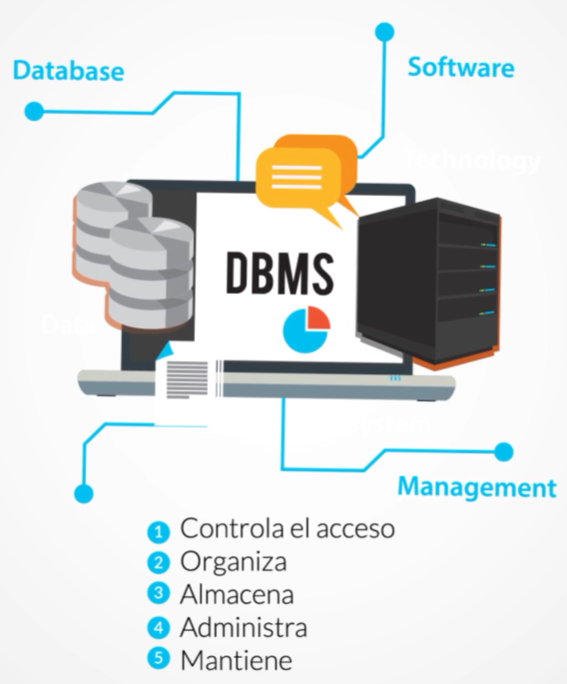
Requerimientos mínimos:



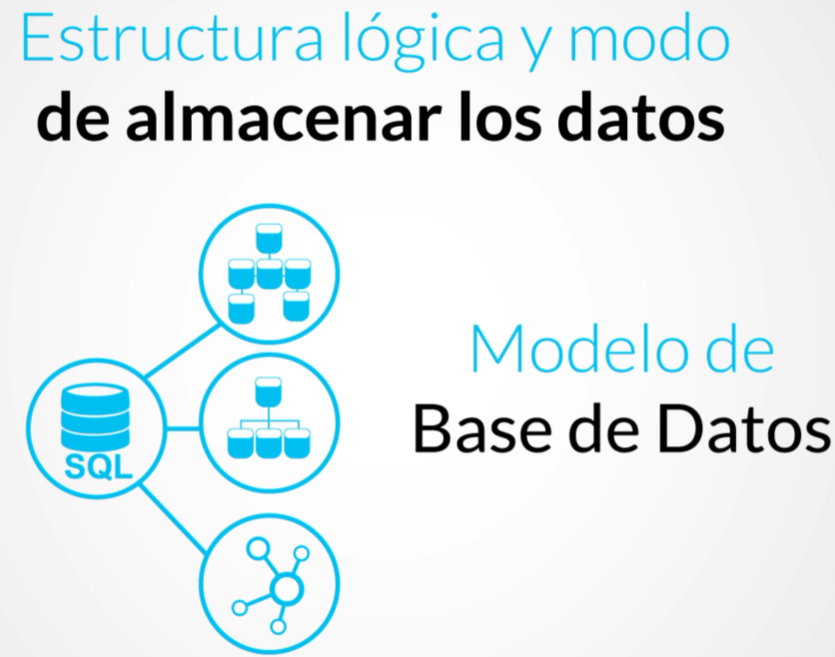
Especialidades:



Gestor de Datos CGBD o DBMS



* Calidad de datos (Redundancia puede provocar pérdida)



Los 3 más utilizados son:

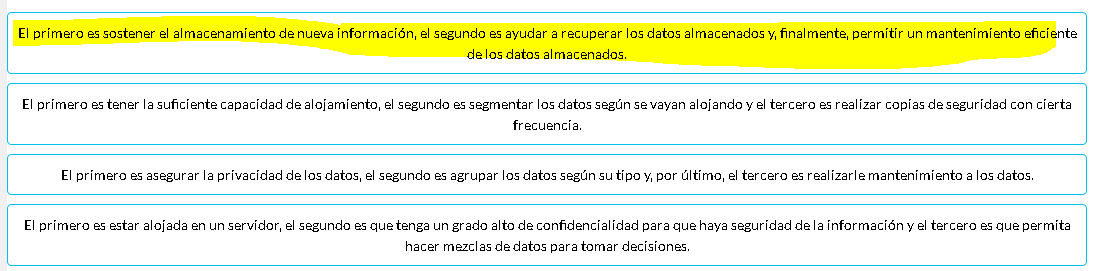
* Modelo Relacional – Es el más común (entidad – relación). Conexiones existentes, un registro es una fila de una tabla de la base de datos.
* Modelo Jerárquico – Organiza los datos en forma de árbol (padre – raíz)
* Modelo en Red - Se Construye en el Modelo Jerárquico (muchos – muchos). Registros padre múltiples



* 1. Actividad 1

Tiempo de pensar

En computación llamamos base de datos a un conjunto de datos que tienen una estructura organizacional que se guarda en un servidor, estos servidores cuentan con un sistema gestor de base de datos que permite a los usuarios acceder a la información que se esté almacenando desde diferentes computadores.  
  
Instrucciones:  
Lee el enunciado y, según las opciones de respuesta que se te plantean a continuación, elige el texto que lo complementa.  
  
Enunciado:  
Las bases de datos deben tener unos requerimientos mínimos:



* 1. Lectura: Por qué utilizar una base de datos

¿Por qué utilizar una Base de Datos?

Traducir:  
-DataBase Management System por Sistema de Gestión de Bases de datos  
-DBMS por SGBD  
-Programming por Programación  
-Data por Datos  
  
Como hemos aprendido en esta lección, las bases de datos son de mucha ayuda para almacenar datos en un servidor. Para que estos datos se puedan administrar de una manera más eficiente, existe un conjunto de aplicaciones que, unidas, forman un Sistema Gestor de Base de Datos (SGBD).  
  
Pero de esto nos surgen varios interrogantes: ¿Por qué utilizar una base de datos? ó ¿Por qué utilizar un SGBD? ¿Por qué no simplemente almacenar los datos en archivos de texto?

Las respuestas a estas preguntas se resuelven analizando las ventajas que tienen las bases de datos y los SGBDs. Entre estas ventajas tenemos:

1. Potencial para reforzar estándares
2. Flexibilidad
3. Aislamiento de los datos y el Software
4. Reducción del tiempo de desarrollo del Software
5. Seguridad
6. Restricción de Acceso No Autorizado
7. Compartir datos y sistema multiusuario
8. Backup y recuperación de datos

1. Potencial para reforzar estándares  
  
Una Base de Datos permite al Administrador (DBA) definir y reforzar estándares entre los usuarios que acceden a los datos en una organización, facilitando la comunicación y la cooperación entre varios departamentos, proyectos y usuarios en la organización.  
Los estándares pueden estar definidos por nombres y formatos de datos, reporte de estructuras, terminología, entre otros.  
En una base de datos, el administrador puede reforzar estándares de una manera centralizada sin tener el problema de asignar las reglas de uso de datos a cada usuario o a cada aplicación.

2. Flexibilidad  
  
En ocasiones, los requerimientos de la aplicación cambian y es necesario cambiar la estructura de la base de datos. Por ejemplo, puede surgir un nuevo grupo de usuarios que necesite información que no está actualmente en la base de datos. Los SGBDs modernos permiten algunos cambios en la estructura sin afectar los datos guardados y los programas de la aplicación existente.

3. Aislamiento de los datos y el Software  
  
Al utilizar una base de datos tienes los datos en un lugar independiente de tu aplicación, lo que permite el acceso desde diferentes lugares o programas. Esto facilita que los cambios en la estructura de datos sean manejados por el SGBD y no estén embebidos en el Software.

4. Reducción del tiempo de desarrollo del Software  
  
Diseñar e implementar una base de datos desde 0 puede tomar más tiempo que escribir un solo archivo especializado. Sin embargo, una vez que la base de datos está configurada y funcionando, generalmente se requiere menos tiempo para crear nuevas aplicaciones utilizando las facilidades de un SGBD.

5. Seguridad  
  
No todos los usuarios tienen los mismos permisos o privilegios de acceso a los datos. Los SGBDs facilitan el manejo y el control de la Seguridad de las Bases de Datos que almacena; permitiendo crear usuarios, grupos de usuarios y roles. A cada uno de estos grupos se les pueden asignar unos permisos y privilegios.

6. Restricción de Acceso No Autorizado  
  
Cuando múltiples usuarios comparten una base de datos es probable que algunos usuarios no estén autorizados para acceder a toda la información. Por ejemplo, para datos confidenciales, algunos usuarios pueden estar autorizados sólo para devolver datos, mientras que otros pueden estar autorizados para devolver y actualizar. Sólo los administradores de la base de datos deben tener permitido ciertos privilegios, como crear nuevas cuentas o asignar permisos de acceso.

7. Compartir datos y sistema multiusuario  
  
Un sistema de base de datos multiusuario debe permitir el acceso de múltiples usuarios a la base de datos al mismo tiempo. Por lo tanto, el sistema gestor de la base de datos debe tener control del acceso al mismo tiempo que la información, garantizando que no se generen datos repetidos con diferente información, que siempre estén actualizados y que éstos sean correctos. A esta característica se le llama **“integridad de datos”.**

8. Backup y recuperación de datos  
  
Un Sistema de Base de Datos proporciona la capacidad de guardar los datos en un momento determinado y esta copia se guarda independiente del SGBD, a este proceso se le conoce como backup o copia de seguridad, lo cual se hace para prevenir la pérdida de datos por fallos en el Software, Hardware o errores del usuario al momento de interactuar con la base de datos.  
El sistema de recuperación y el backup son los responsables de la restauración del sistema. Si un sistema falla en medio de un programa complejo de actualización, el SGBD debe restaurar la base de datos al estado en el que estaba antes de que el programa comenzara a ejecutarse.

2.4. Base de datos relacional

Es un conjunto de datos organizados en tablas que se relacionan para permitir un almacenamiento práctico y útil de la información para la toma de decisiones.

* Tabla 🡪 Información Ej: Clientes 🡪 Número de identificación, Nombre 🡪 Registro o Tupla

Número\_Cuenta 🡪 Cliente y Cuenta es una relación: 1 a 1, muchos a 1 y muchos a muchos

Fr 🡪 Foreign Key

* 1. Actividad 2

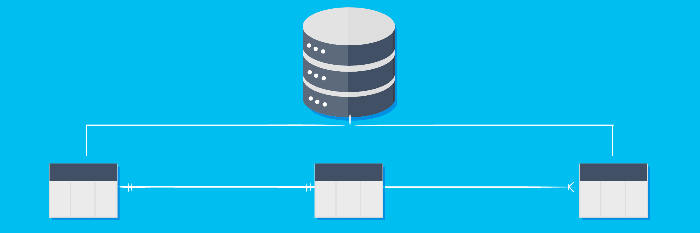
Tiempo de pensar

Recuerda que la tabla es la unidad fundamental de una base de datos relacional, sin embargo por sí sola no significa nada, pues no hay datos en ella. Según lo aprendido en la lección, una relación de uno a uno significa que un sólo elemento de una entidad o tabla se relaciona con un sólo elemento de otra.  
  
Instrucciones:  
  
Lee con atención la pregunta y, una vez lo tengas claro, selecciona dos opciones de respuesta.  
  
Pregunta:  
  
¿Cuáles de las siguientes opciones de respuesta plantean bases de datos relacionales uno a uno?

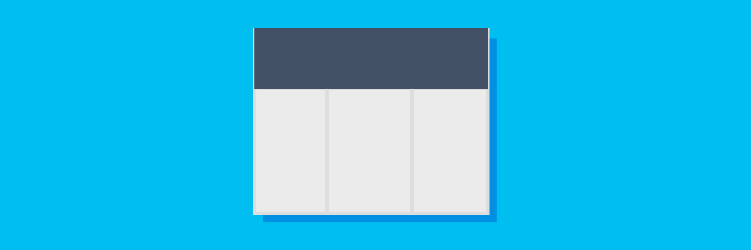


* 1. Lectura: Conceptos y restricciones

Bases de datos: Conceptos y restricciones



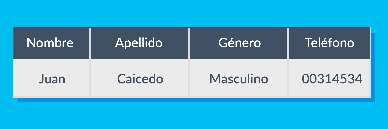
Recordemos que una base de datos relacional es una colección de elementos de datos organizados en múltiples tablas descritas detalladamente, desde las que podemos acceder y/o gestionar los datos de diversas maneras sin tener la necesidad de reorganizar la base de datos. Después de mencionar lo anterior, es momento de que veamos los conceptos y restricciones que nos van a garantizar el buen funcionamiento de nuestra base de datos.



**CONCEPTOS**  
  
**1. Tablas:** dentro de una base de datos relacional, las tablas son las estructuras que se encargan de almacenar y/o alojar la información de la base de datos. Las tablas son usadas para ordenar y entregar la información almacenada, éstas están compuestas por filas y columnas, las cuales pueden ser llenadas con cualquier tipo de datos.



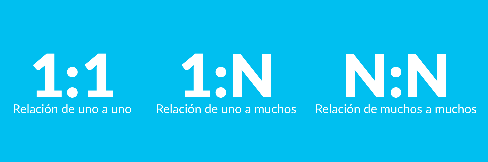
**2. Campos o atributos:** los campos o atributos de una tabla son cada una de las columnas que la componen, donde cada campo almacena un dato en concreto, por ejemplo, el campo **‘Nombre’.** A cada campo se le asigna el tipo de dato que almacenará, es decir, si el dato es un número, una cadena de texto, un dato booleano, etc., con el fin de darle un orden a la información que se está almacenando.



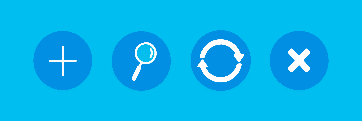
**3. Registros:** los registros son cada una de las filas que componen la tabla y su función es agrupar toda la información de un mismo elemento.



**4. Llaves o Claves:** una clave o llave es un campo en el cual su valor o contenido no puede estar duplicado dentro de la misma tabla, ya que éste nos permitirá identificar y diferenciar cada registro de los demás. Los principales tipos de llaves o claves que existen son:  
**- Llave o clave primaria (Primary Key - PK):** es un campo único e irrepetible que nos permite identificar y diferenciar la información de una tabla.  
**- Llave o clave foránea (Foreign Key - FK):** la clave foránea es una clave secundaria que se utiliza en una tabla, haciendo referencia a la clave primaria de la tabla con la cual está relacionada.  
**- Llave o clave candidata:** también conocida como clave única, esta llave es usada cuando aún no se define una clave o llave primaria. Se conoce como clave o llave candidata ya que es uno de los datos que no tiene tendencia a repetirse en otros registros.



**5. Relación:** dentro de una base de datos relacional, una relación es un vínculo que se establece entre dos o más tablas, el cual permite trabajar con todos los datos de éstas como si estuvieran en una sola tabla. Existen 3 tipos de relaciones entre tablas y son los siguientes:  
**- Relación de uno a uno:** es el tipo de relación menos frecuente entre tablas y sucede cuando un solo elemento de una entidad o tabla se relaciona con un solo elemento de otra. Un ejemplo de esta relación lo encontramos entre países y capitales: un país sólo puede tener una capital y una capital sólo puede estar en un país.  
**- Relación de uno a muchos:** es el tipo más frecuente de relación entre tablas. Una relación uno a muchos se da cuando un registro de una tabla se relaciona con más de un registro de otra. Por ejemplo, país y ciudad es una relación de uno a muchos porque un país tiene muchas ciudades, pero una ciudad sólo puede pertenecer a un país.  
**- Relación de muchos a muchos:** es aquella en la que los registros de dos tablas pueden relacionarse con más de un elemento de otra tabla. Por ejemplo, la relación entre personas y ciudades de residencia: una persona puede vivir en más de una ciudad durante su vida y, a su vez, una ciudad alberga más de una persona.



**6. Consultas:** en las bases de datos, una consulta es el método por medio del cual podemos acceder a los datos de nuestra base de datos; éstas nos permiten insertar, eliminar, modificar y borrar los datos de nuestras bases de datos a través de un lenguaje de consultas. En la actualidad el más utilizado es el SQL, cuyas siglas en inglés significan Structured Query Lenguage, y en español significa Lenguaje de Consulta Estructurada.  
  
Como vimos anteriormente, existen múltiples conceptos a tener presentes en el momento de trabajar con bases de datos relacionales, pero es necesario saber qué restricciones se presentan.  
  
**Restricciones:** las restricciones en las bases de datos relacionales son un conjunto de reglas y/o condiciones que se deben cumplir al pie de la letra para el manejo adecuado de los datos dentro de las tablas que la componen, con el fin de garantizar la integridad de los datos para realizar cualquier tipo de operación con ellos. Algunas de estas restricciones no son definidas por el usuario, ya que al trabajar con bases de datos relacionales, éstas ya tienen definidas ciertas restricciones que el usuario debe respetar, mientras que otras son definidas por el mismo usuario, por ejemplo: el tipo de dato de un campo, la longitud de caracteres, si el campo es obligatorio o no.

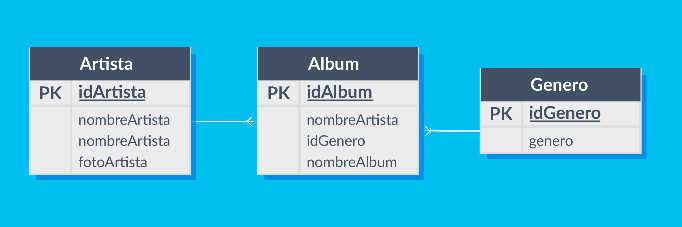
Estas son algunas de las restricciones más comunes:

1. Ninguna llave primaria puede ser un valor nulo.
2. Para que una llave foránea exista dentro de una tabla, debe ser delegada previamente como la llave primaria en la tabla con la cual se relaciona.
3. Los valores que se ingresan en una columna de la tabla deben corresponder al tipo de dato que se le asignó previamente al campo, es decir, si se creó el campo ‘Fecha’, el valor que éste debe almacenar no debe ser numérico o de texto, sino que debe ser de tipo fecha, por ejemplo: 11-11-1992 o 11/11/1992, según el formato que éste maneje.
4. Los campos que se marquen como obligatorios no pueden ser nulos o estar vacíos.
5. Los campos que forman una clave candidata deben tomar siempre valores distintos para cada posible registro.
   1. Lectura: Normalización

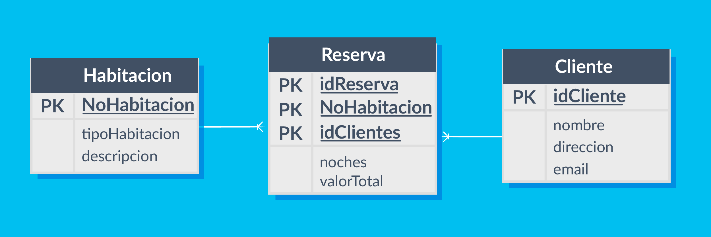
Normalización

Como mencionamos anteriormente, una base de datos relacional está compuesta de tablas que se relacionan una a otra a través de llaves. Las columnas de la base de datos se llaman atributos y las tablas se relacionan a través de una o varias llaves, esto depende de la base de datos que estemos trabajando. Cuando una llave primaria está compuesta de varios campos la llamamos **Llave Compuesta o Compuound Key.**

**Llave primaria**



**Llave Compuesta (Compound Key)**



**Llave Compuesta → idReserva+NoHabitacion+idCliente**

Pero para llegar a tener una base de datos en la que las tablas tengan información que no se repita, hay que seguir un proceso que se llama Normalización, que tiene como propósito organizar las columnas de las tablas con sus respectivas relaciones para eliminar la redundancia y mejorar la integridad de los datos.  
  
El proceso de Normalización está dividido en tres conjuntos de reglas:  
1. Primera Forma Normal (1FN)  
2. Segunda Forma Normal (2FN)  
3. Tercera Forma Normal (3FN)

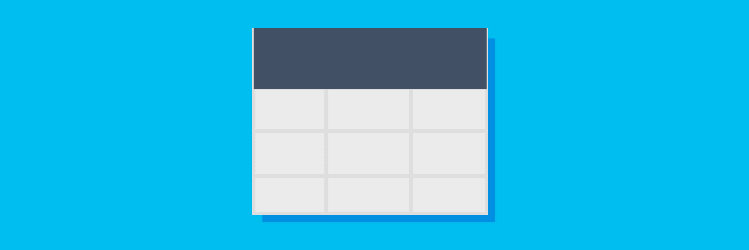


Según Edgar Frank Codd, cada uno de estos conjuntos de reglas tienen el propósito de solucionar o prevenir los siguientes problemas:

* Quitar de la colección de datos las dependencias no deseadas al momento de insertar, actualizar o eliminar datos.
* Reducir la necesidad de reestructurar las relaciones entre los datos cuando se añaden nuevos atributos o nuevas tablas a la base de datos.
* Hacer que la colección de relaciones sea neutral a las estadísticas de consulta, donde estas estadísticas pueden cambiar a lo largo del tiempo.
* Hacer que el modelo relacional proporcione más información a los usuarios de las bases de datos.

Aunque en la práctica no todas las reglas se cumplen, entre más se cumplan se diría que una base de datos es ‘más relacional’.  
Para entender cómo ejecutar el proceso de normalización, partiremos de la siguiente base de datos que está representada por esta tabla no normalizada.





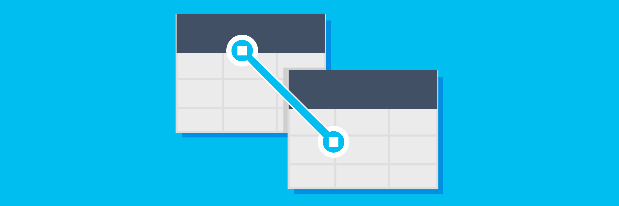
1. Primer nivel de normalización:

* Eliminar atributos repetidos dentro de una relación.
* Agrupar los elementos repetidos en una tabla.
* Identificar cada fila de la tabla con una columna.

Para nuestro ejemplo, esto se transforma en reducir los atributos “dirección1” y “dirección2”, que pertenecen al mismo contacto. Como resultado tenemos la siguiente tabla:



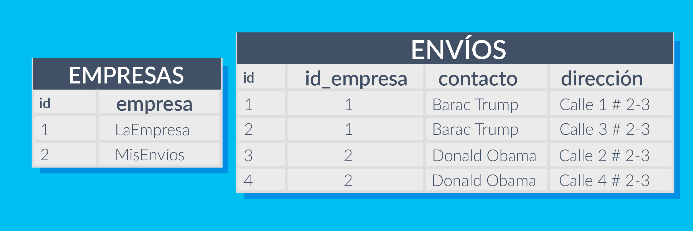
Por lo que podemos ver, para la relación “envíos” no existen campos repetidos pero hay filas que generan redundancia, como la repetición del nombre de un contacto o el nombre de la empresa. Para esto aplicaremos el segundo nivel de normalización.



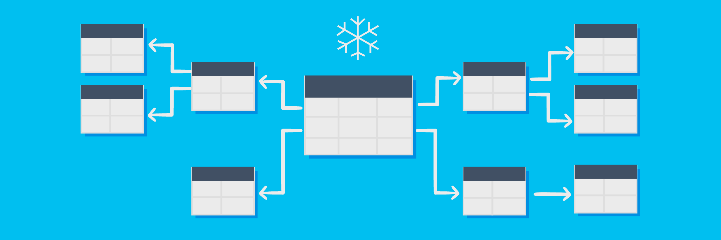
2. Segundo nivel de normalización:

* Separar en una tabla los datos que se repiten en varias filas que puedan generar errores en la integridad de los datos.
* Identificar la nueva tabla de relaciones con claves primarias.

Para el caso de la relación “envíos”, la columna “empresa” contiene datos que pueden corromper la integridad cuando se vaya a actualizar el nombre de una empresa. Esto genera como resultado una tabla de “empresas” y otra tabla de “envíos”.

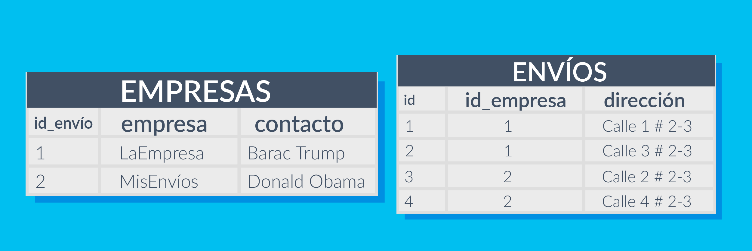


Vemos que se agregó la relación “empresas” y que se relaciona con la tabla principal de envíos mediante una clave foránea. Ahora veamos cómo aplicar el tercer nivel de normalización.



3. Tercer nivel de normalización:

* Eliminar los campos que pueden ser claves únicas y que no están relacionados con la clave primaria en tablas aparte.

En el caso de la base de datos de envíos, este campo sería el atributo “contacto” y puede ser agrupado en la relación existente entre las empresas, así quedamos con las siguientes tablas:  


Estos tres niveles de normalización son los que se usan principalmente en el diseño de las bases de datos, pero puedes consultar por tu cuenta: la forma normal de Boyce-Codd, la cuarta Forma Normal (4FN) o la quinta Forma Normal (5FN).

1. Lección 2: Lenguaje SQL – DDL
   1. Instalación de PostgreSQL

Es un gestor de base de datos orientado a Objetos y Relaciones, conocido por su estabilidad, fácil uso y multiplataforma 🡪 Global Development Group

Instalación: Descargar el instalador Postgresql para el sistema operativo donde se desea trabajar

En consola Windows:

C:\Users\jrodrigue253> "..\..\Program Files\PostgreSQL\13\bin\psql.exe" -h localhost -U postgres

Password for user postgres:

psql (13.0)

WARNING: Console code page (437) differs from Windows code page (1252)

8-bit characters might not work correctly. See psql reference

page "Notes for Windows users" for details.

Type "help" for help.

postgres=# **\list**

List of databases

Name | Owner | Encoding | Collate | Ctype | Access privileges

-----------+----------+----------+----------------------------+----------------------------+-----------------------

postgres | postgres | UTF8 | English\_United States.1252 | English\_United States.1252 |

template0 | postgres | UTF8 | English\_United States.1252 | English\_United States.1252 | =c/postgres +

| | | | | postgres=CTc/postgres

template1 | postgres | UTF8 | English\_United States.1252 | English\_United States.1252 | =c/postgres +

| | | | | postgres=CTc/postgres

(3 rows)

postgres=#

PgAdmin 🡪 herramienta para facilitar el uso del PostgreSQL

El puerto por defecto en PostgreSQL es: 5432

* 1. Actividad 1

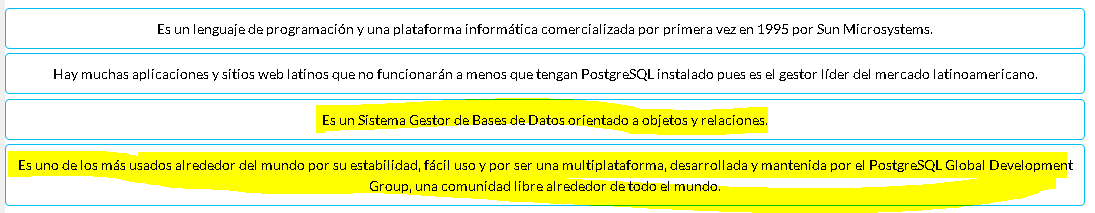
Tiempo de pensar

Hemos estudiado PostgreSQL como gestor de las bases de datos. Pero un momento… ¿Recuerdas qué es PostgreSQL? Resuelve la siguiente actividad.  
  
Instrucciones:

Lee la pregunta y elige las respuestas que consideres correctas.

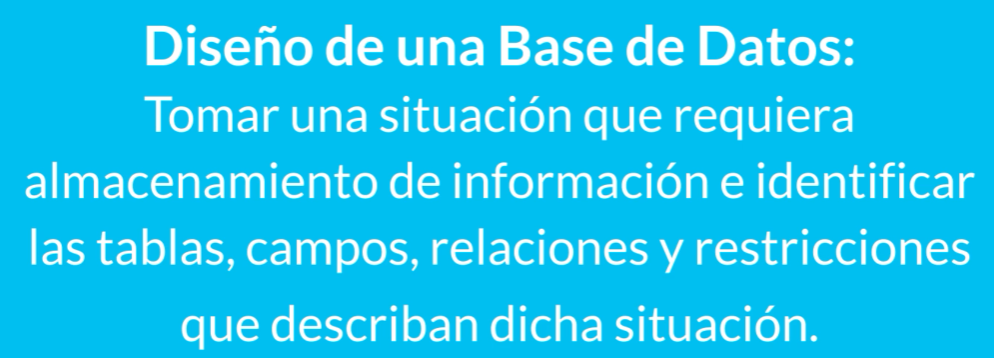
Ten presente que son dos.

Pregunta:  
  
¿Cuáles de las siguientes opciones plantea enunciados relacionados con PostgreSQL?



* 1. Diseño de Base de datos Relacional

Diseño antes de construir una DB. 🡪 Debe satisfacer los requerimientos

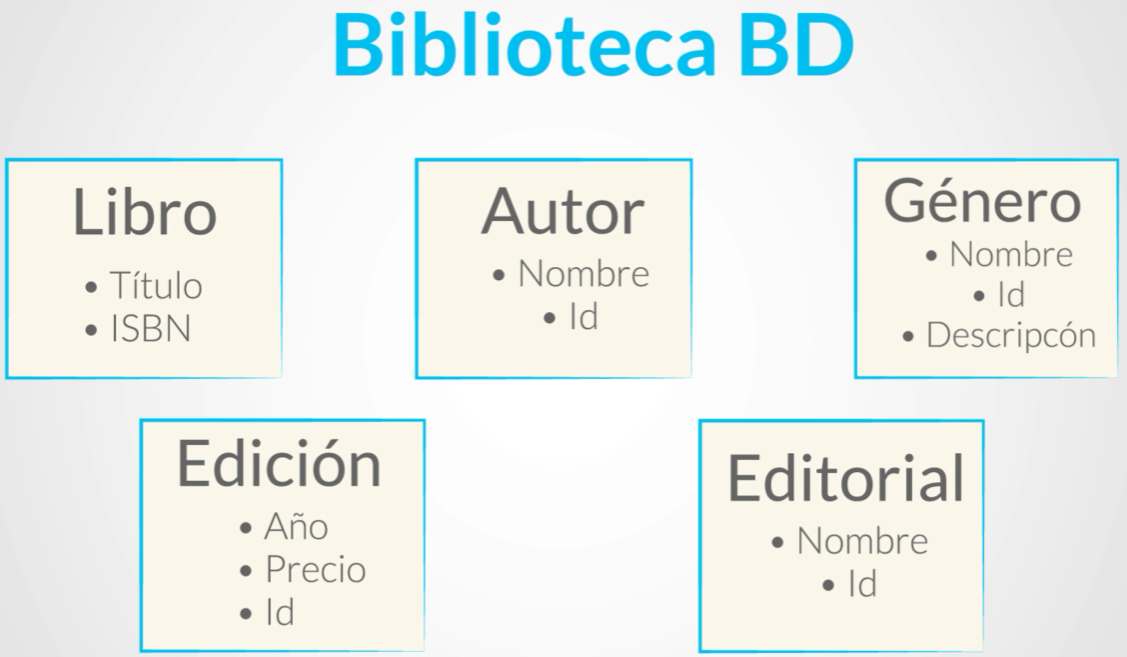


Identificar las entidades y tablas.

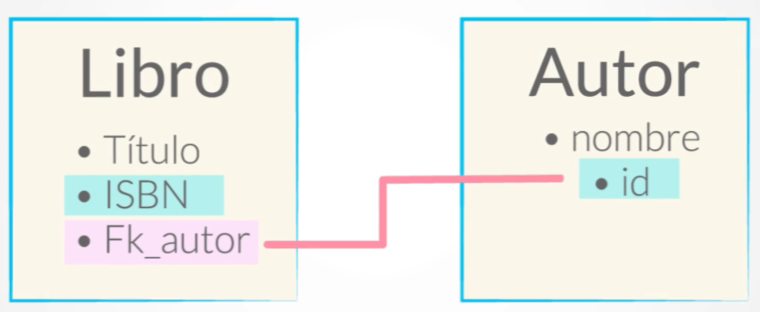
Ejemplo:



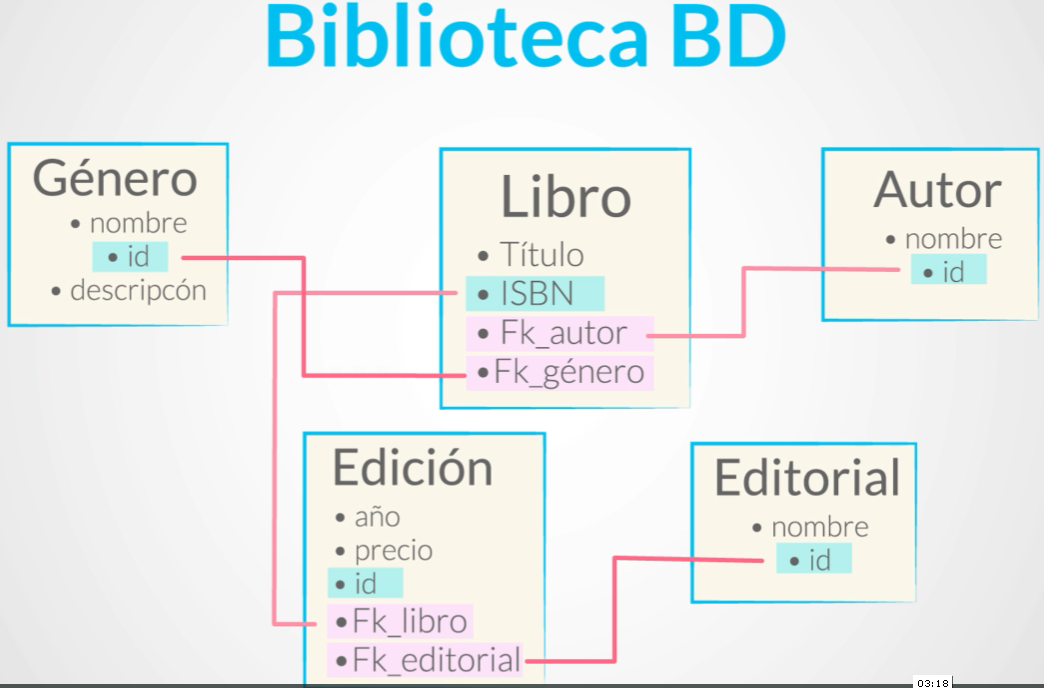
Cada tabla deberá tener sus campos o columnas. Cada tabla debe tener un Id (Llave primaria)



Llave primaria y llave foránea



Después de Diseño



* 1. Actividad 2

Tiempo de pensar

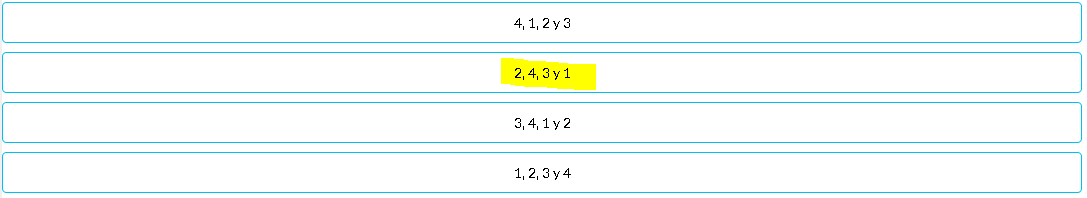
Hemos estudiado que diseñar una base de datos consiste en tomar una situación que requiera almacenamiento de información e identificar las tablas, campos, relaciones y restricciones que describan dicha situación. Realiza la siguiente actividad para poner en práctica lo aprendido.  
  
Instrucciones:

* Lee los pasos para diseñar una base de datos.
* Ordénalos mentalmente.
* Una vez tengas claro el paso a paso adecuado, selecciona la opción de respuesta que plantee el orden correcto.

Pasos:

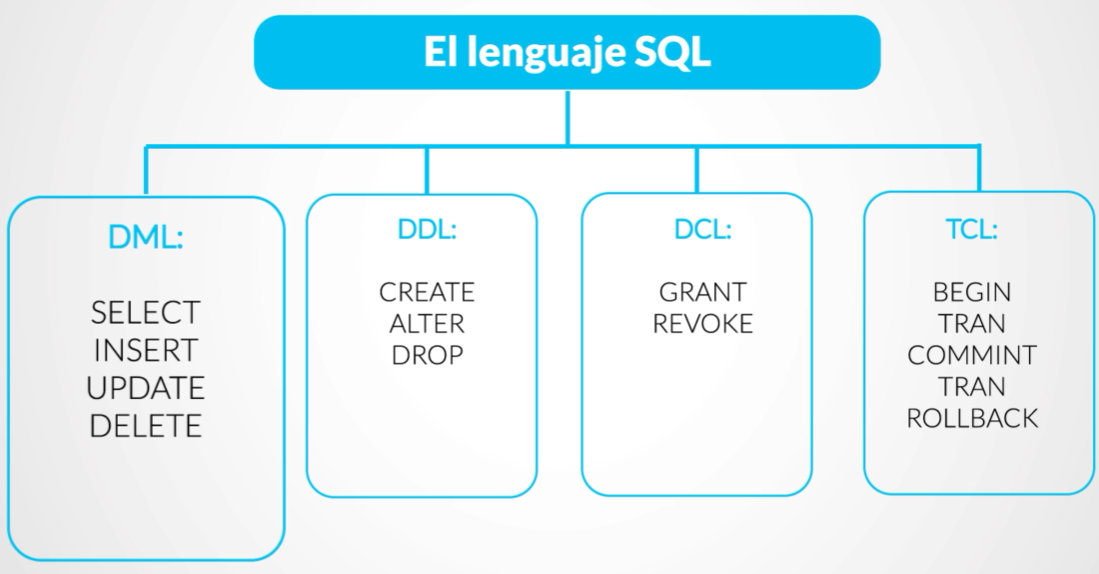
1. Cuando establecemos las relaciones, tendremos un diseño.
2. Identificar las entidades o tablas.
3. Definir las relaciones, es decir, la unión de varias tablas que tengan algo que ver, por ejemplo, libro y autor.
4. Definir, por cada una de las tablas, sus campos o columnas. En este paso es importante saber qué datos nos interesa almacenar de cada entidad.

El orden correcto es:



* 1. Actualizar registros

SQL 🡪 Structure Query Language 🡪 Bases de datos relacionales 🡪 comandos y palabras reservadas



DML: Lenguaje de manipulación de datos

DDL: Lenguaje de definición de datos

DCL: Lenguaje de control de datos

TCL: Lenguaje de control de transacciones

Creación de DB:

Create database database\_name

En PostgreSQL para conectarse a una base de datos en terminal se debe correr el comando:

\c database\_name

Creación de una Tabla:

Create table nombre\_tabla ( nombre\_campo1 INTEGER(3),

nombre\_campo2 CHARACTER VARYING(45)

);

Para ver las tablas creadas en un DB PostgreSQL se envía el comando \d

Ejemplo:

postgres=# \c biblioteca\_db

You are now connected to database "biblioteca\_db" as user "postgres".

biblioteca\_db=# create table libro (

biblioteca\_db(# isbn varchar(20),

biblioteca\_db(# titulo varchar (100),

biblioteca\_db(# fk\_autor integer,

biblioteca\_db(# fk\_genero integer);

CREATE TABLE

biblioteca\_db=# \d

List of relations

Schema | Name | Type | Owner

--------+-------+-------+----------

public | libro | table | postgres

(1 row)

biblioteca\_db=#

Para ver las columnas de una tabla se envía el comando: \d nombre\_tabla

biblioteca\_db=# \d libro

Table "public.libro"

Column | Type | Collation | Nullable | Default

-----------+------------------------+-----------+----------+---------

isbn | character varying(20) | | |

titulo | character varying(100) | | |

fk\_autor | integer | | |

fk\_genero | integer | | |

biblioteca\_db=#

* 1. Creación de relaciones y restricciones

Llave Primaria 🡪 Campo único para todos sus registros y sirve como identificador

Comandos:

Alter table 🡪 Sirve cambios a una entidad en una base de datos

Alter table nombre\_tabla Add Primary Key (nombre\_campo);

Ejemplo:

biblioteca\_db=# alter table libro Add primary key (isbn);

ALTER TABLE

biblioteca\_db=# \d libro

Table "public.libro"

Column | Type | Collation | Nullable | Default

-----------+------------------------+-----------+----------+---------

isbn | character varying(20) | | not null |

titulo | character varying(100) | | |

fk\_autor | integer | | |

fk\_genero | integer | | |

Indexes:

"libro\_pkey" PRIMARY KEY, btree (isbn)

Para definir que un campo no puede ser nulo:

Not null

Ejemplo:

golf\_db=# alter table paises alter column nombre set not null

golf\_db-# ;

ALTER TABLE

Modificando una tabla ya creada se realiza de la siguiente forma:

Alter table nombre\_tabla alter columna nombre\_campo Set not null

Ejemplo:

biblioteca\_db=# alter table libro Add primary key (isbn);

ALTER TABLE

biblioteca\_db=# \d libro

Table "public.libro"

Column | Type | Collation | Nullable | Default

-----------+------------------------+-----------+----------+---------

isbn | character varying(20) | | not null |

titulo | character varying(100) | | |

fk\_autor | integer | | |

fk\_genero | integer | | |

Indexes:

"libro\_pkey" PRIMARY KEY, btree (isbn)

biblioteca\_db=#

Relación entre dos tablas:

Alter table nombre\_tabla add constraint nombre\_relacion Foreign key (campo\_tabla) reference table\_destino (campo\_destino)

Ejemplo:

biblioteca\_db=# \d autor

Table "public.autor"

Column | Type | Collation | Nullable | Default

--------+------------------------+-----------+----------+---------

id | integer | | not null |

nombre | character varying(100) | | not null |

Indexes:

"autor\_pkey" PRIMARY KEY, btree (id)

biblioteca\_db=# alter table libro add constraint libro\_autor foreign key (fk\_autor) references autor (id);

ALTER TABLE

biblioteca\_db=# \d libro

Table "public.libro"

Column | Type | Collation | Nullable | Default

-----------+------------------------+-----------+----------+---------

isbn | character varying(20) | | not null |

titulo | character varying(100) | | not null |

fk\_autor | integer | | |

fk\_genero | integer | | |

Indexes:

"libro\_pkey" PRIMARY KEY, btree (isbn)

Foreign-key constraints:

"libro\_autor" FOREIGN KEY (fk\_autor) REFERENCES autor(id)

biblioteca\_db=#

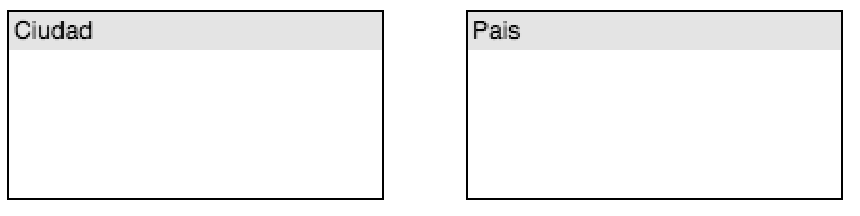
* 1. Ejercicio 1

HORA DE  
codificar

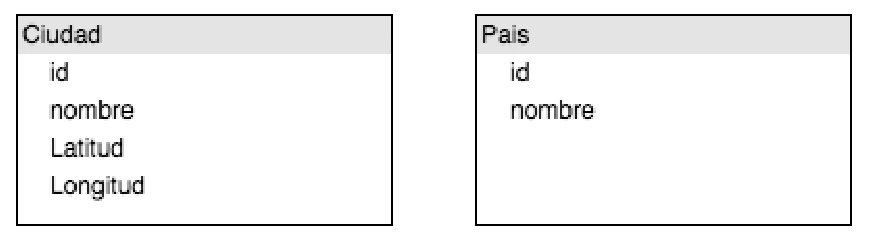
En todo proceso de desarrollo e incluso en un proyecto de cualquier índole, la etapa de diseño es una de las más relevantes en todo el proceso de ejecución, debido a que ésta permite determinar específicamente lo que se debe hacer para solucionar el problema propuesto inicialmente. Los sistemas de información no son la excepción y por lo tanto es muy importante que adquieras habilidad en este ámbito y particularmente en el diseño de bases de datos relacionales.  
  
Como bien sabes, las bases de datos relacionales brindan seguridad en los datos y proporcionan una estructura escalable e integral para almacenar datos correspondientes a un modelo específico. Sin embargo, ninguna de estas ventajas puede ser explotadas si la base de datos no ha sido definida correctamente desde la fase de diseño.  
  
En este ejercicio te será presentado una situación real que requiere manejo de información. Presta mucha atención a la descripción del caso para que identifiques las necesidades específicas de almacenamiento de información y puedas crear una propuesta válida.  
  
Para el desarrollo de este ejercicio debes leer con mucha atención el siguiente caso. Luego sigue las instrucciones para que construyas un modelo de bases de datos correspondiente a la situación presentada. Puedes ir realizando el modelo en una hoja de papel, o en cualquier herramienta de diseño que prefieras.  
  
Golf internacional.  
La federación internacional de Golf decidió sistematizar la información correspondiente a este deporte a nivel mundial, para lo cual necesita un portal web en el que las personas puedan acceder a toda la información. La base de datos que almacene dicha información, debe contener todos los campos de golf alrededor del mundo con su código único, nombre del campo, dirección y ciudad a la que pertenece. Deben estar almacenados los países en los que tiene presencia la federación y las ciudades correspondientes. Cada campo debe tener asociado la lista de todos su hoyos. El identificador único del hoyo es su número dentro del campo y el campo al que está asociado (Por ejemplo, el hoyo 6 del Campo Charlstone de Nueva York); adicionalmente se debe guardar por cada hoyo, el par y una dificultad descrita por palabras. La base de datos debe almacenar los datos de todos los jugadores afiliados, guardando su identificación única ante la federación, su nombre y apellido, la ciudad de nacimiento, la fecha de nacimiento y la categoría del jugador definida por una palabra.  
  
Adicionalmente, la base de datos debe almacenar los torneos oficiales de la federación de golf, incluyendo un código único de identificación del torneo, su nombre, la fecha de inicio, el premio que ofrece y la categoría de requisito para inscribirse. Por cada torneo se deben almacenar el número de golpes que realiza un jugador por cada hoyo del campo y la puntuación, la cual se calcula mediante la diferencia positiva o negativa del número de golpes en un hoyo con su respectivo par (Por ejemplo, el hoyo 3 tiene un par de 3 y el jugador realizó 4 golpes, por lo tanto su puntuación es de +1). Por último se deben almacenar las puntuaciones generales de cada jugador en un torneo, con la suma total de golpes realizados y la suma de todas sus puntuaciones.

Instrucciones

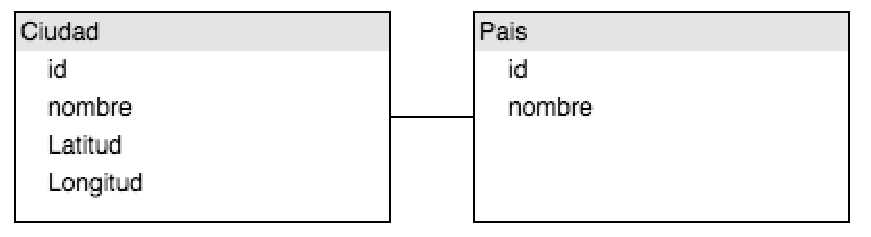
1. Lee detenidamente el caso y comprende la situación general que se te presenta. Familiarízate con los términos y acciones involucrados en la situación.
2. Identifica las tablas de la base de datos, obteniendo todas las entidades importantes dentro del caso presentado. Ejemplo:



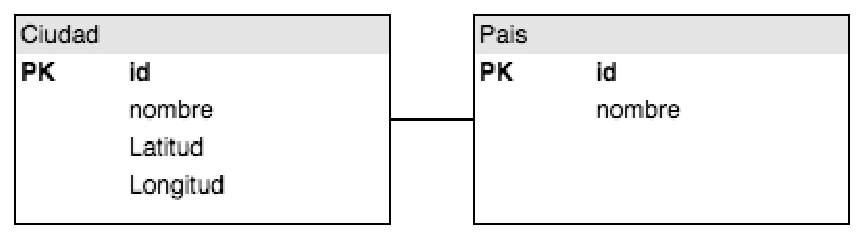
1. Por cada una de las tablas identificadas, lista los campos o columnas que tendrían, de acuerdo con la información que se debe almacenar de cada una de las entidades. Ubícalas al interior de la tabla correspondiente. Ejemplo:



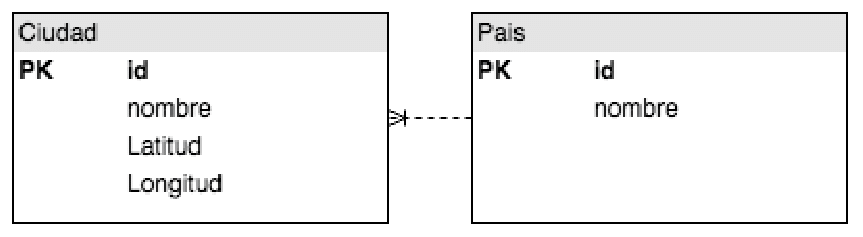
1. Identifica las relaciones entre cada una de las tablas y únelas con una línea. Por ejemplo:



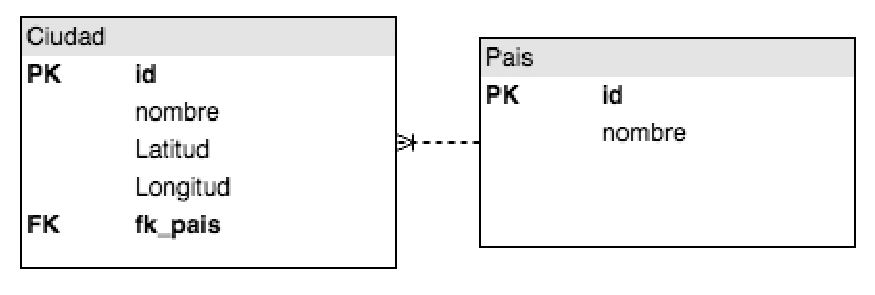
1. Para cada tabla, identifica la llave o el conjunto de llaves primarias y señálalas en el modelo. Por ejemplo:



1. Para cada relación entre tablas, determina el tipo de relación y utiliza una notación específica en tu diagrama para mostrar relaciones de uno a uno, de uno a muchos y de muchos a muchos. Por ejemplo. Relación de uno a muchos.



1. Para cada tabla con relaciones, identifica las llaves foráneas correspondientes, asígnales un nombre y resáltalas de los demás campos.



Buenas Prácticas

Dedicar tiempo en el buen diseño de una base de datos, garantiza un funcionamiento óptimo y seguro en el manejo de los datos, los cuales son el recurso más importante de un sistema de información. Una de las prácticas más recomendadas al diseñar bases de datos relacionales, es evitar el exceso de relaciones y la creación de tablas innecesarias, ya que en ocasiones se tienden a crear más de las necesarias.

* 1. Ejercicio 2

HORA DE  
**codificar**

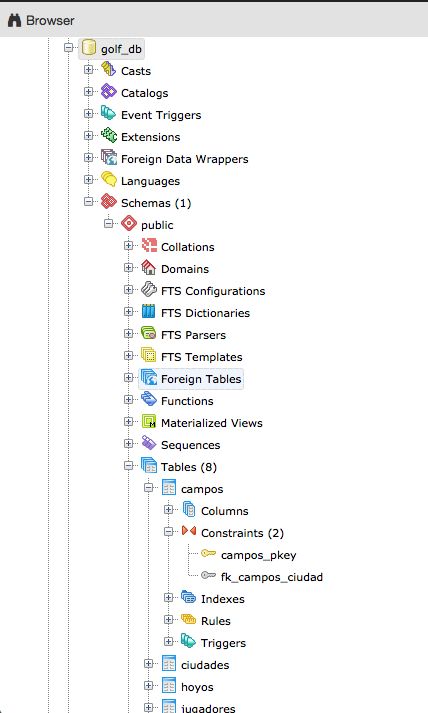
SQL es la herramienta por excelencia para manipular las bases de datos relacionales. Al momento de finalizar la fase de diseño de una base de datos, debemos recurrir a SQL para construirla en el servidor que vayamos a usar. A todos los comandos de SQL que se encargan de crear una base de datos y construir sus elementos, se agrupan en la categoría DDL (Data Definition Language).  
  
En este ejercicio usaremos el diseño de la base de datos para el caso propuesto en el ejercicio anterior. A partir de este diagrama, deberás crear la base de datos en el servidor local PostgreSQL en tu máquina.

Contenido

Para el desarrollo de este ejercicio debes usar el diseño de la base de datos que creaste en el ejercicio anterior. Sin embargo, si no cuentas con este diagrama, puedes descargarlo del codBase.

Instrucciones

1. 1A través de la consola del servidor PostgreSQL, ejecuta una sentencia SQL para crear una base de datos con el nombre golf\_db.
2. Ejecuta una sentencia SQL para la creación de cada tabla en el diseño de base de datos. Ten en cuenta la definición de las llaves primarias, los tipos de datos correctos y los valores no nulos en todos los campos.
3. Ejecuta una sentencia SQL para la creación de cada relación, asignando las llaves foráneas correspondientes que apunten a los campos de las tablas de referencia de manera correcta.
4. Accede a PgAdmin4 y verifica que la base de datos se creó exitosamente. Observa las tablas y sus restricciones para identificar si todas las relaciones fueron creadas adecuadamente. Alfinalizar podrás ver algo como esto en el navegador de PgAdmin4:



Buenas Prácticas

El lenguaje DDL corresponde a uno de los más poderosos grupos de SQL, ya que la estructura de una base de datos es en sí, el lugar en el que se almacenarán los datos. Cuando uses sentencias SQL de definición DDL, presta mucha atención a lo que ejecutas en el servidor, puesto que los errores que se produzcan a partir de un mal uso de este tipo de comandos pueden ser muy perjudicial para los datos que se almacenan o serán almacenados allí.

1. Lección 3: Lenguaje SQL – DML
   1. [Inserción de datos](https://learn.nextu.com/mod/lesson/view.php?id=3556&pid=P_WEBDEV_V2)

DML – Data Management Language

Inserción de datos

Insert Into nombre\_tabla (campo1, campo2) Values (‘valor1’,’valor2’)

Si se omiten los campos se va a ingresar los valores de todas las columnas en el orden en que se encuentran definidas las columnas.

Tabla fuerte: ninguna columna es llave foránea.

Tabla débil: posee al menos una columna de tipo llave foránea.

Primero se llenan tablas fuertes y luego tablas débiles.

Ej:

biblioteca\_db=# insert into autor (id,nombre) values (1,'William Shakespeare');

INSERT 0 1

biblioteca\_db=#

Varias filas en una sentencia insert:

biblioteca\_db=# insert into autor (id,nombre) values (2,'Paulo Cohelo'), (3,'Daniel Rodriguez');

INSERT 0 2

biblioteca\_db=#

* 1. Actividad 1

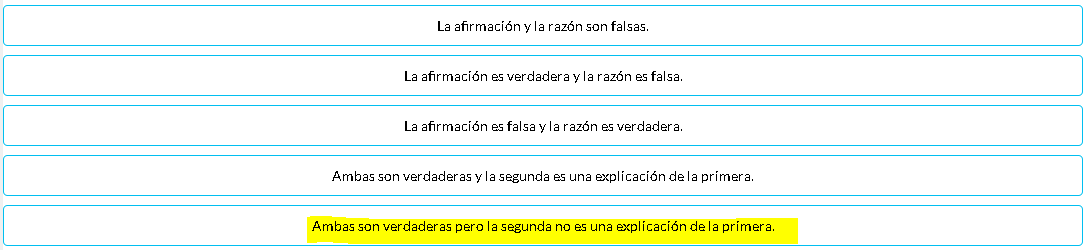
Tiempo de pensar

Ahora ya sabes cómo crear la estructura de una base de datos, sus tablas, relaciones y restricciones. Además, estudiamos sobre el Lenguaje de Manipulación de Datos (DML).  
  
Instrucciones:

Lee con atención la afirmación y la razón e identifica si son enunciados verdaderos o falsos.

Determina si la primera es una explicación de la segunda.

Afirmación:  
Para ingresar registros en una tabla es muy importante saber si ésta es débil o fuerte…  
... porque...  
  
Razón:  
En una base de datos relacional siempre se deben poblar primero las tablas fuertes y por último las débiles.



* 1. Actualizar y eliminar registros

Actualizar:

Update table\_nombre set nombre\_campo = ‘nuevo\_valor’ where nombre\_campo = 2

Eliminar registros de una tabla:

Delete from nombre\_tabla where campo=3 🡪 Se debe tener cuidado con esta sentencia

* 1. Consultas básicas

Select \* from nombre\_tabla where condiciones

Operador Like

* 1. Actividad 2

Tiempo de pensar

Hemos estudiado que una consulta es ejecutar una sentencia en la base de datos para obtener información almacenada en ella. Es como una pregunta que le haces a la base de datos para obtener una respuesta.  
  
Instrucciones:

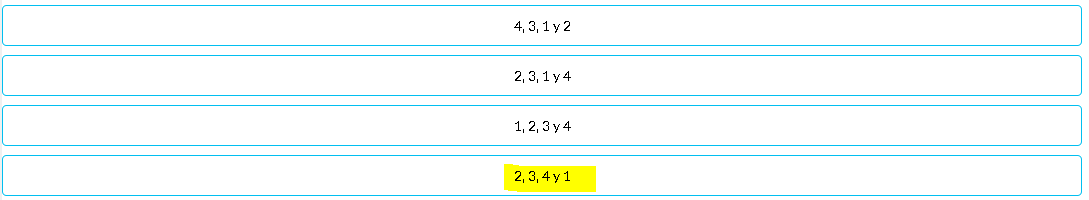
Lee con atención los pasos que se plantean a continuación, que sirven para realizar una consulta SQL.

Ordénalos mentalmente y, una vez creas identificar el paso a paso adecuado, selecciona la opción de respuesta correcta.

Pasos:

1. De manera opcional, se puede añadir un condicional para filtrar los resultados.
2. Se usa la sentencia SELECT.
3. Se indica el o los campos que se requiere obtener.
4. Se utiliza la sentencia FROM y luego el nombre de la tabla de la que se obtendrá la información.

El orden correcto es:



* 1. Consultas avanzadas

Ordenamiento de resultados

Order by nombre\_campo

DESC para invertir el sentido de ordenamiento

Función de conteo:

COUNT(nombre\_campo)

COUNT(\*)

COUNT(DISTINCT nombre\_campo)

Operador de agrupación

Group By

* 1. Consultas basadas en relaciones

En bases de datos relacionales

JOIN ON

Nombres de los campos:

Ejemplo:





Inner Join 🡪 significa que se obtendrán todos los registros comunes a ambas tablas de la relación

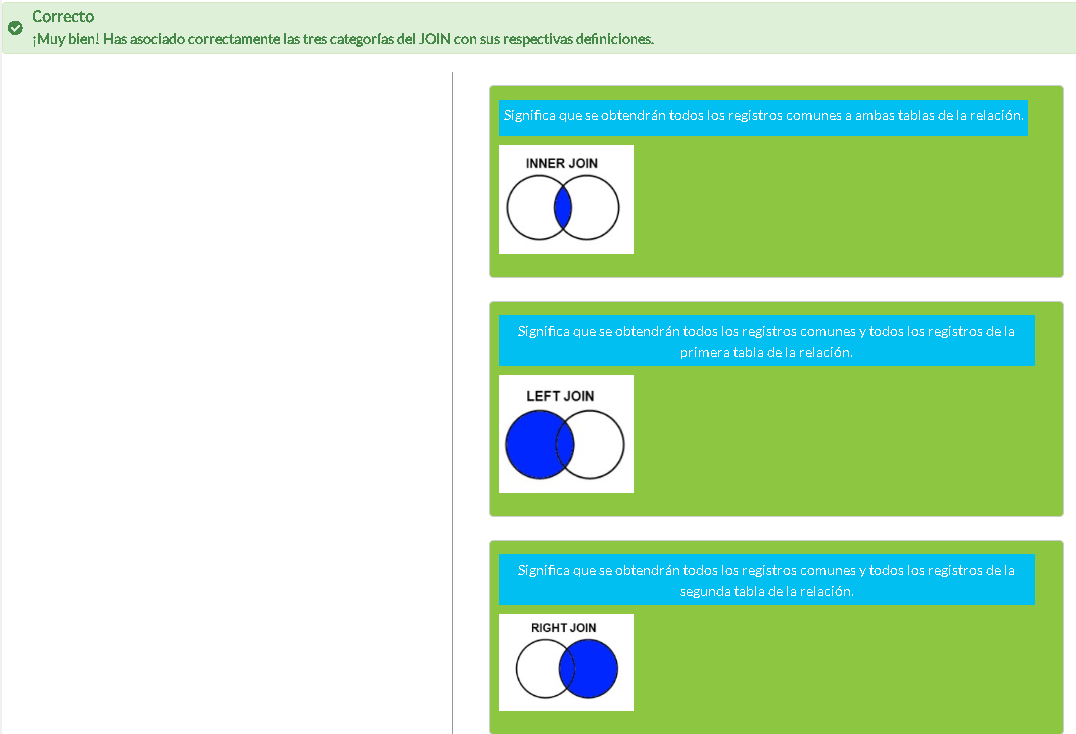
Left Join 🡪 Significa que se obtendrán todos los registros comunes y todos los registros de la primera tabla de la relación

Right Join 🡪 Significa que se obtendrán todos los registros comunes y todos los registros de la segunda tabla de la relación

* 1. Actividad 3

Tiempo de pensar

Como ya lo hemos estudiado, la característica principal de SQL es que se fundamenta en bases de datos relacionales en las que, gracias a los enlaces entre tabla y tabla, se adquieren propiedades asociativas importantes. Y, para realizar una consulta que involucra a dos o más tablas relacionadas, se usan los operadores JOIN y ON. Realicemos la siguiente actividad para evaluar los conocimientos.  
  
Instrucciones:  
  
Asocia las tres categorías del JOIN, presentadas en la columna izquierda, a las definiciones correspondientes, en la columna derecha. ¡Adelante!



* 1. Ejercicio 1

HORA DE  
**codificar**

La construcción de una base de datos SQL, tiene como propósito almacenar datos relevantes al modelo de negocio de un sistema de información. Sin embargo, el proceso de inserción de cada registro debe tener en cuenta la definición de los campos de cada tabla y de las restricciones correspondientes. La identificación de las tablas fuertes y débiles en el modelo de base de datos, permite una inserción de registro sin presentar errores y dispone a la información para que sea almacenada limpia y ordenadamente.  
  
En este ejercicio usaremos la base de datos que construiste en el ejercicio anterior, con el objetivo de poblarla con unos cuantos registros que sigan adecuadamente el modelo propuesto inicialmente para el case del sistema de información para la federación de golf.

Contenido

Para el desarrollo de este ejercicio debes usar la base de datos que creaste en el ejercicio anterior. Sin embargo si no cuentas con ella, puedes ejecutar en tu servidor PostgreSQL el archivo SQL del codBase para construir una base de datos que te servirá para el desarrollo de este ejercicio.

Instrucciones

1. Almacena en la base de datos las siguientes ciudades:  
   a. París y Marsella en Francia.  
   b. Madrid y Barcelona en España.  
   c. Lisboa y Oporto en Portugal.
2. Ingresa los siguientes campos de golf:  
   a. Campo de Golf “Le Champ” ubicado en la dirección Av4 87-21, París.  
   b. Campo de Golf “Petites” ubicado en la dirección Av1 11-40, Marsella.  
   c. Campo de Golf “Los Robles” ubicado en la dirección Av7 22-11, Madrid.  
   d. Campo de Golf “Antienes” ubicado en la dirección Av2 25-25, Barcelona.  
   e. Campo de Golf “Ouro Preto” ubicado en la dirección Av9 10-05, Lisboa.  
   f. Campo de Golf “Pedras Brancas” ubicado en la dirección Av23 12-51, Oporto.
3. Para el campo de golf “Le Champ” debes ingresar 18 hoyos y para “Los robles” 9 hoyos. Eres libre de decidir cuál es el par de cada hoyo (un número entre 2 y 5) y su dificultad entre baja, media y alta.
4. Añade 10 jugadores a la base de datos. Puedes crear sus datos como quieras indicando su categoría de la siguiente manera:  
   a. 5 jugadores deben ser Profesionales  
   b. 3 jugadores deben ser Semi-Profesionales  
   c. 2 jugadores deben ser Amateur.
5. Ingresa 2 torneos a la base de datos con las siguientes características:  
   **a. Masters de París:** Torneo celebrado el 12 de Abril de 2016 en la categoría Semi-profesional, entregando un premio de $400.000 dólares al primer lugar.  
   **b. Abierto de Madrid:** Torneo celebrado el 30 de Agosto de 2016 en la categoría Profesional, entregando un premio de $800.000 dólares al primer lugar.
6. Registra las tarjetas de 3 jugadores que participaron en el Masters de París, el cual se jugó en el Campo de Golf de esta ciudad. Cada participante jugó los 18 hoyos de este campo. La base de datos debe registrar el número de golpes de cada jugador por cada hoyo, puedes elegir el número que desees entre 1 y 9. La puntuación debe ser cero para todos los registros.
7. A través de la herramienta PgAdmin4 corrobora que los registros se hayan ingresado correctamente y existan la información dentro de la base de datos.

Buenas Prácticas

El poblado de una base de datos es un proceso en ocasiones dispendioso por la cantidad de datos que deben ingresarse y las facilidades que se presentan para cometer errores. Es altamente recomendado que al insertar datos en una base de datos, tengas a la mano el diseño de la misma, para conocer con claridad los campos, las llaves primarias, foráneas, restricciones y demás aspectos que te puedan generar errores en la inserción.

* 1. Ejercicio 2

HORA DE  
**codificar**

La razón por la cual se usa una base de datos se fundamenta en la seguridad y buena organización de la información allí contenida. Sin embargo, el uso de una base de datos no sólo se limita a guardar registros sino también a permitir el acceso a los datos allí contenidos de una manera más práctica y modular. Las consultas SQL permiten obtener los datos de los registros almacenados en una base de datos, así como de filtrarlos y organizarlos para presentarlos de manera óptima ante las necesidades del sistema.  
  
En este ejercicio vamos a obtener algunos datos almacenados en la base de datos que has creado en los ejercicios anteriores. Debes hacer uso de las sentencias SQL del DML para obtener la información requerida.

Contenido

Para el desarrollo de este ejercicio debes usar la base de datos que usaste en el ejercicio anterior. Sin embargo, si no cuentas con ella, puedes ejecutar en tu servidor PostgreSQL el archivo SQL del codBase para construir una base de datos que te servirá para el desarrollo de este ejercicio.

[Descargar archivo base](https://s3.amazonaws.com/nextu-content-production/Desarrollador_Web/07_Interactuando_Base_de_Datos/Ejercicios/WEB16S_C7U1L3_ejercicio2/solucion/codBase.zip)

Instrucciones

1. Realiza una consulta SQL que obtenga el nombre, apellido, fecha de nacimiento y categoría de todos los jugadores en la base de datos.
2. Realiza una consulta SQL que obtenga el nombre de todas las ciudades almacenadas, con su latitud y longitud.
3. Realiza una consulta SQL que obtenga el nombre de todos los campos de golf almacenados en la base de datos, mostrando su nombre, dirección, y el nombre de la ciudad en la que se encuentran.
4. Realiza una consulta SQL que obtenga como resultado la ciudad en la que se juega el torneo “Masters de París”.
5. Realiza una consulta SQL que obtenga como resultado el número de golpes que cada participante del torneo “Masters de París”, realizó en el primer hoyo del campo.
6. Realiza una consulta SQL para obtener el número de golpes de cada jugador por cada hoyo del campo en el que se jugó el torneo “Masters de París”. Los resultados deben estar ordenados por el número del hoyo.
7. Realiza una consulta SQL que muestre el nombre, el apellido y la ciudad natal de todos los jugadores inscritos en los torneos almacenados en la base de datos.
8. Realiza una consulta SQL que obtenga el número de campos de golf en cada país almacenado en la base de datos.

Buenas Prácticas

La construcción de consultas SQL se fundamenta en un buen entendimiento del modelo de la base de datos relacional. El uso adecuado del operador JOIN, te permitirá obtener datos entre tablas y generalmente se usa más de una sentencia JOIN por consulta en una base de datos relacional. Ten siempre presente el diseño de la base de datos antes de realizar cualquier consulta para evitar errores de sintaxis en los nombres de las tablas o campos de la base de datos.

1. Lección 4: Qué es y cómo funciona una base de datos tipo No Relacional
   1. Lectura: Qué es una BD NoSQL y cómo funciona

**¿**Qué es una BD NoSQL y cómo funciona?

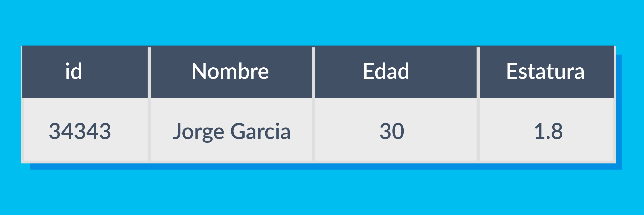


Desde la llegada de la web 2.0 el término de NoSQL hace parte del día a día de la industria. Anteriormente, lo más común era escuchar hablar solo de bases de datos SQL como MySQL, Oracle o SQLServer de Microsoft.  
NoSQL hace referencia a No solo SQL (Not Only SQL), este concepto se presenta por el crecimiento exponencial a nivel de información que estaban afrontando compañías como Twitter o Facebook donde los usuarios están subiendo contenido e información constantemente, y la gestión y escalabilidad de estos datos, representados en bases de datos relacionales, era extremadamente compleja y costosa.  
Entonces cuando hablamos de bases de datos NoSQL nos referimos a estructuras que permiten almacenar información que en una arquitectura SQL genera ciertos problemas de rendimiento y accesibilidad, y donde, por lo general, suceden millones de consultas e ingresan miles de usuarios en concurrencia.

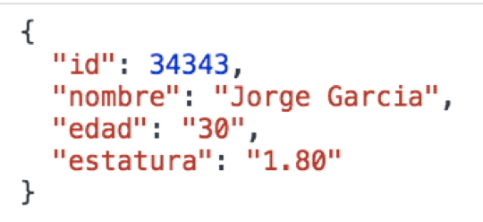


**Cómo funcionan las bases de datos NoSQL**  
  
En los sistemas de bases de datos NoSQL la información no está representada en entidades ni tablas, tampoco sigue el esquema entidad-relación, por el contrario, las bases de datos NoSQL utilizan diferentes estructuras de almacenamiento como objetos, documentos JSON, formatos clave-valor y grafos, entre otros.

**NoSQL orientado a documentos:**  
  
- En una estructura relacional:



- En documento JSON:



Este es el tipo de bases de datos NoSQL más utilizado en reemplazo a las estructuras SQL. Es bastante versátil, ya que está basado en una estructura tipo JSON de un único documento por registro.

* 1. Lectura: Ventajas y desventajas

Ventajas y desventajas

**Ventajas:**  
**- Código abierto:**  
La mayoría de bases de datos NoSQL son de código abierto (open source), esto aporta gran confiabilidad a los desarrolladores, aparte de ser un servicio sin costo alguno.  
**- Nivel de escalabilidad:**  
El escalamiento de este tipo de bases de datos es simple, solo se deben agregar nodos para distribuir la carga de los datos. Una base de datos permite distribuir la carga entre varios hosts.  
**- Diferentes DB’s para diferentes soluciones:**  
En la actualidad existen muchas bases de datos NoSQL especializadas en diferentes cosas; para proyectos en tiempo real, altos volúmenes de información, almacenamientos de sesiones o caché.  
**- Almacenar grandes cantidades de información.**

**Desventajas:**  
**- No utilizan SQL como lenguaje de consultas:**  
Los desarrolladores están acostumbrados a los lenguajes SQL para acceder a la información de sus bases de datos, pues en los sistemas NoSQL se evita utilizar esto y, en su reemplazo, se utilizan lenguajes basados en JSON.  
**- No hay una estructura fija:**  
En los motores NoSQL no hay una estructura definida que deban cumplir cada uno de sus registros, por el contrario, cada documento o registro puede almacenar estructuras diferentes.  
**- Limitaciones en la lógica del negocio:**  
Las estructuras SQL ayudan mucho en cuanto a la lógica de negocio de una aplicación, ya que por la implementación del modelo entidad-relación se obtienen muchas ventajas en esa parte.

* 1. Bases de datos NoSQL más comunes

mongoDB 🡪 NoSQL escrita en C++, documentos tipos JSON

BSON 🡪 evolución de JSON, la cual consiste en almacenamiento de JSON en formato binario

Redis 🡪 Base de datos clave-valor almacena los datos directamente en memoria. Como un array gigante. Sistemas distribuidos donde se tiene varios nodos y estos comparten la info, no permite consultas solo se pueden insertar u obtener datos.

CouchDB 🡪 Por apache, orientada a documentos. Los datos quedan accesibles a través de una API REST. Utiliza JSON para su almacenamiento y permite hacer JOIN típicas de SQL.

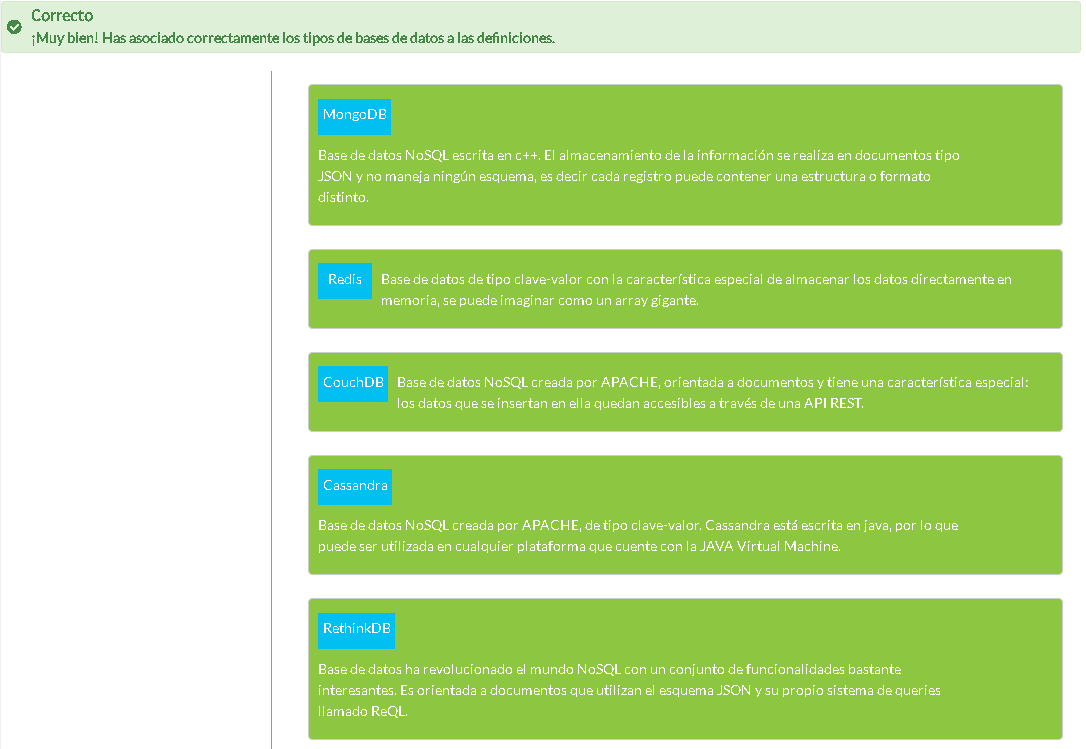
Cassandra 🡪 Apache, clave -vector, está escrita en JAVA. Tiene su propio lenguaje llamado CQL (Cassandra Query Language).

RethinkDB 🡪 Revolucionado el mundo NoSQL, orientada a documentos JSON y tiene su propio sistema de queries llamado ReQL Rethink Query Language. Soporta JOIN y tiene un dashboard muy práctica. Además, notifica.

* 1. Actividad

Tiempo de pensar

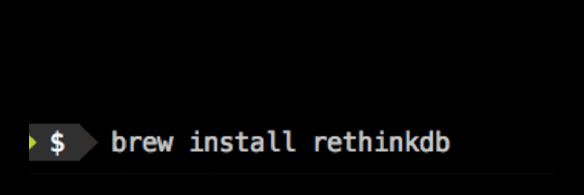
Hemos estudiado las principales bases de datos NoSQL de la actualidad.  
  
Instrucciones:  
  
Asocia los tipos de bases de datos, presentadas en la columna izquierda, a las definiciones y características correspondientes, en la columna derecha. ¡Adelante!



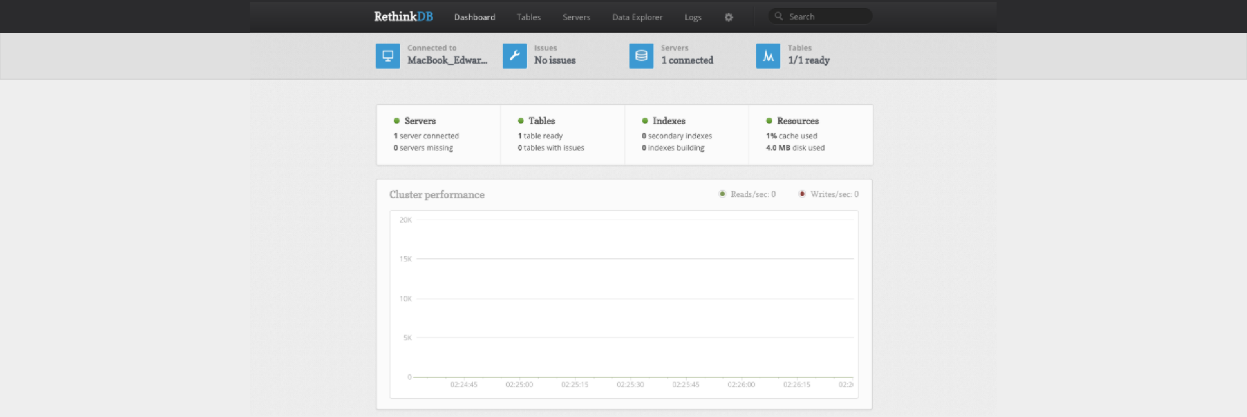
1. Lección 5: Realizar consultas en base de datos No Relacional
   1. [Lectura: Instalar RethinkDB](https://learn.nextu.com/mod/lesson/view.php?id=3558&pid=P_WEBDEV_V2)

Guía de instalación RethinkDB

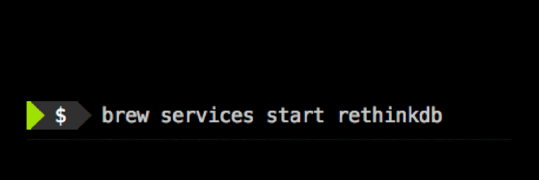
RethinkDB está disponible oficialmente para MacOS, Windows y para distribuciones Linux como Debian, Ubuntu y CentOS, y la comunidad ofrece soporte para otras distribuciones derivadas como Linux Mint, OpenSUSE, Fedora, Arch y Raspbian.  
Si estás en un sistema operativo MacOS te recomiendo usar el manejador de paquetes brew, ejecuta el siguiente comando en la terminal:



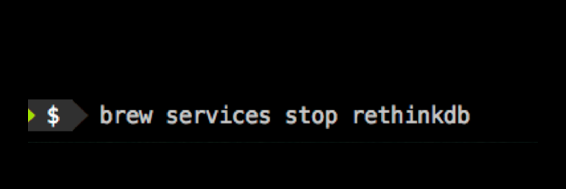
RethinkDB es todo un servidor que se instala en nuestra máquina y se ejecuta por defecto en el puerto 8080. Si vamos al browser y entramos a localhost:8080 debemos ver el dashboard de administración de la base de datos:



Para detener o lanzar el servidor de RethinkDB a través de brew, usamos los siguientes comandos:  
  
Iniciar el servicio:



Detener el servicio:



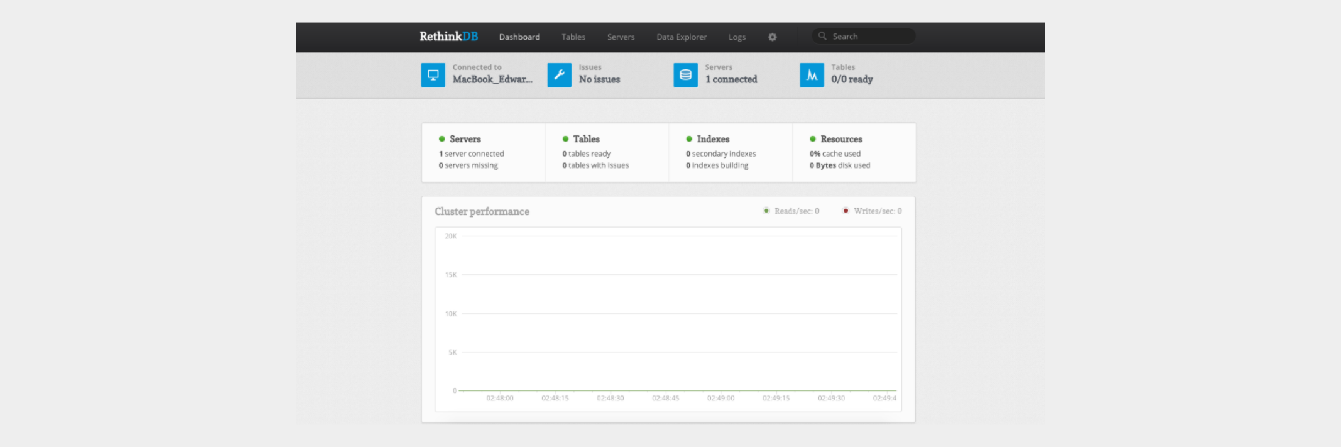
Si estás en un sistema operativo de Windows o Linux, te recomiendo visitar la página oficial de RethinkDB: <https://www.rethinkdb.com/docs/install/> , donde puedes encontrar los instaladores para cada una de las plataformas y, adicionalmente, hay una guía para cada uno.



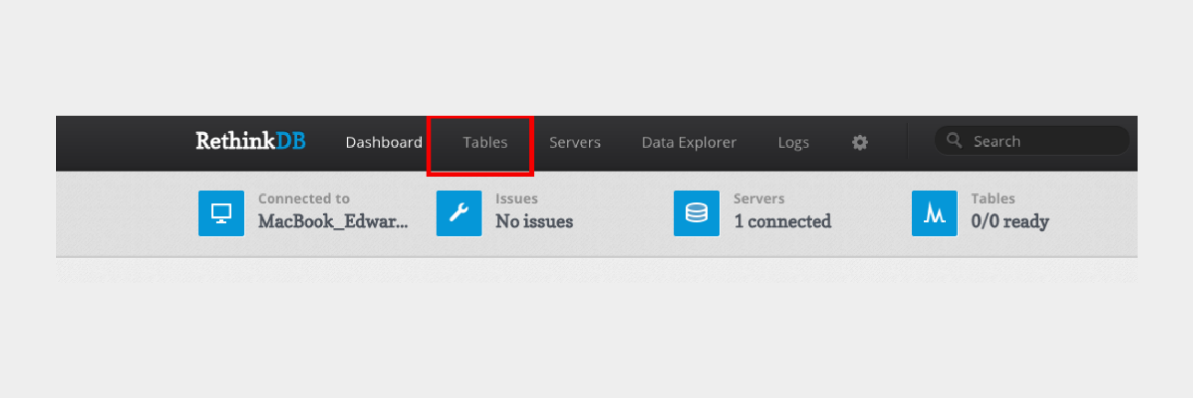
* 1. Lectura: Dashboard RethinkDB

Dashboard RethinkDB

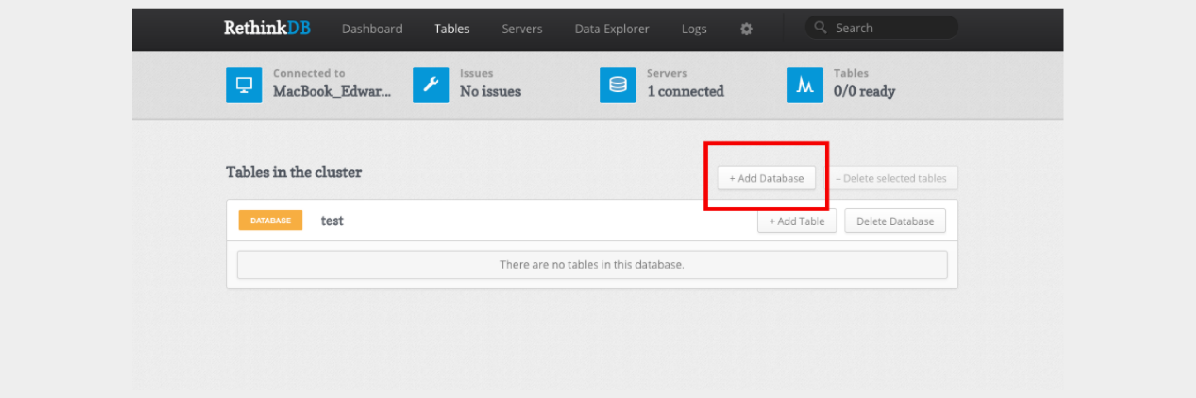
Una vez tengamos instalado el RethinkDB y esté corriendo perfectamente en nuestra máquina, vamos a explorar un poco el dashboard de administración, a crear una base de datos, una tabla y, posteriormente, a insertar algunos registros.  
  
Abriremos el panel de administración en un navegador ingresando a localhost:8080:



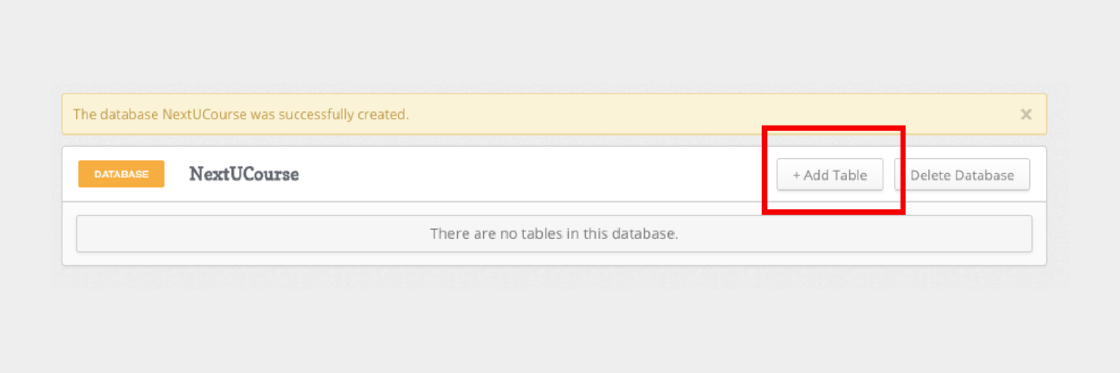
En el menú superior ingresamos en la sección **“Tables”**



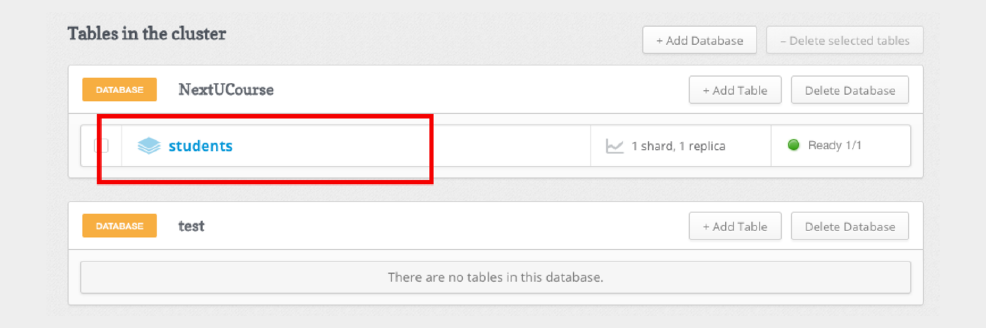
Por defecto, Rethink crea una base de datos llamada test; adicional a esta, vamos a crear una nueva llamada **NextUCourse,** haciendo clic sobre el botón **create database.**



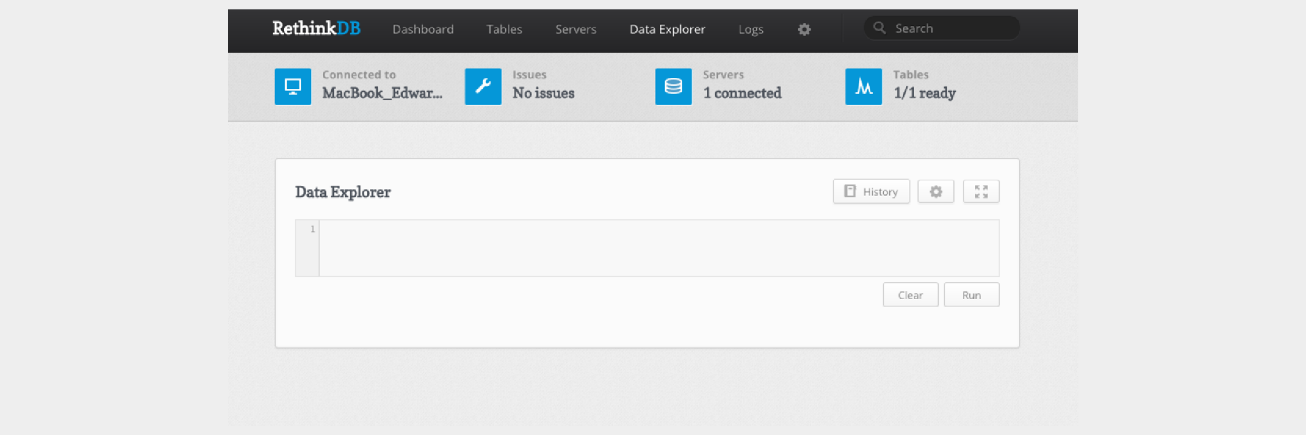
Luego de ingresar el nombre que queremos ponerle a la base de datos, esta aparecerá en el panel y nos habilitará la opción de crear una nueva tabla.



Aquí crearemos una tabla llamada **“students”.**



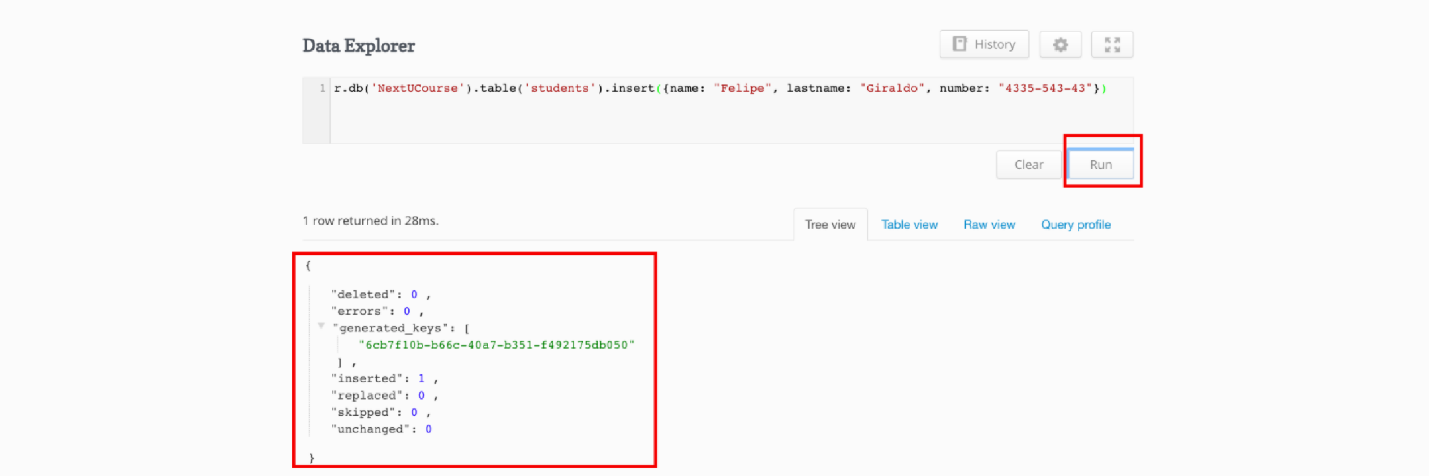
Está todo listo para empezar a insertar registros, ahora en el menú superior vamos a la sección **“Data explorer”**



En este recuadro podemos ejecutar la sentencia ReQL (Rethink Query Language). Por ahora las dos sentencias que utilizaremos serán para insertar y leer registros.  
La sentencia para insertar un nuevo registro es la siguiente:



ReQL funciona encadenando funciones, primero invocamos el objeto “r” que contiene todas las operaciones de Rethink, luego invocamos la base de datos NextUCourse y después hacemos referencia a la tabla Students. Por último, ejecutamos la función INSERT pasándole un objeto como parámetro, este objeto es el registro como tal.  
Escribimos la sentencia y damos clic en el botón **RUN.**



* 1. Realizar consultas

Para consultar:

r.db('NextUCourse').table('students')

ReQL orientado a SQL

Para filtrar

Filter({})

Ejemplo:

r.db('NextUCourse').table('students').filter({name:"Felipe"})

Se pueden ordenar usando order by:

Ejemplo:

r.db('NextUCourse').table('students').orderBy(name)

Límites en la consulta con el operador limit:

Ejemplo:

r.db('NextUCourse').table('students').orderBy(name).limit(2)

* 1. Actividad

Tiempo de pensar

Ahora que sabemos cómo realizar consultas con el lenguaje de queries de RethinkDB queremos invitarte a que realices la siguiente actividad.  
  
Instrucciones:  
Lee con atención las siguientes afirmaciones y determina cuáles de ellas son verdaderas.

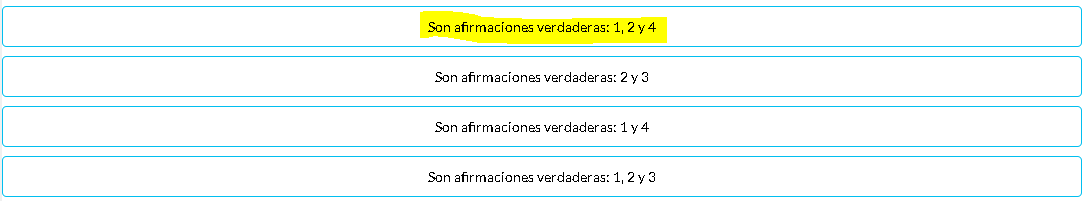
Una vez lo hayas analizado, selecciona la opción de respuesta que las reúne.

Afirmaciones:  
  
1. Para realizar una consulta general a una tabla de una base de datos en Rethink, utilizamos la misma sintaxis que utilizamos para hacer la inserción de registros pero omitiendo la invocación a la función INSERT.

2. La manera de realizar una consulta, de manera adecuada, con Rethink es: escribiendo esta sentencia en el explorador de datos y luego haciendo clic en RUN para obtener los resultados.

3. El lenguaje ReQL es bastante orientado a NoSQL, por lo que es bastante intuitivo y fácil de aprender para los desarrolladores.

4. Podemos filtrar solo un elemento invocando la función FILTER y pasándole como argumento el campo y el valor que queremos filtrar dentro de uno o varios documentos.



* 1. Ejercicio

HORA DE  
**codificar**

RethinkDB es una base de datos NoSQL de código abierto que se caracteriza por su buen desempeño para aplicaciones en tiempo real. Su instalación es sencilla y la gestión de las bases de datos y sus acciones correspondientes se realiza fácilmente por medio de su interfaz gráfica a través del navegador.  
  
En este ejercicio debes crear una base de datos para una tienda de productos tecnológicos sobre la herramienta RethinkDB.  
  
Para el desarrollo de este ejercicio debes seguir las siguientes instrucciones una a una para la creación de la base de datos planteada y sus respectivas acciones.

Instrucciones

1. Crea una base de datos con el nombre Tienda
2. Crea las siguientes tablas en la base de datos:  
   a. productos  
   b. vendedores  
   c. clientes
3. Ingresa 5 productos con los siguientes datos:  
   a. código de producto  
   b. nombre  
   c. precio  
   d. cantidad disponible
4. Ingresa los siguientes vendedores con sus respectivos datos:  
   a. Santiago Gonzáles: Su id es 1023221 y su número de teléfono es 7165239854  
   b. Gabriela Gómez: Su id es 3654873 y su número de teléfono es 6458932543
5. Ingresa 4 clientes con los siguientes datos por cada uno:  
   a. id  
   b. nombre  
   c. teléfono  
   c. email
6. Consulta los registros de cada tabla, verificando que los datos ingresados son correctos.
7. Añade a la consulta de la tabla productos el método filter() para filtrar los resultados de la consulta. Como parámetro indica un objeto con el nombre del campo y el valor que deseas buscar. Realiza una consulta donde obtengas un producto cuyo precio sea 1200.
8. Añade a una consulta filtrada el método delete() para eliminar un registro. Elimina un registro de la tabla clientes, filtrando por el nombre de un cliente registrado.

Buenas Prácticas

Al trabajar con el motor de bases de datos RethinkDB, es muy recomendado utilizar el objeto changes, el cual permite estar al tanto de todo tipo de cambios en la base de datos seleccionada tales como inserciones, actualizaciones y eliminación de registros. Esto nos evita realizar consultas una y otra vez para obtener los registros de las tablas, liberando así la cantidad de tareas que debe ejecutar el motor de base de datos y contribuyendo al rendimiento general del sistema.

1. UNIDAD 1: Prueba

# **Interactuando con Bases de Datos >**UNIDAD 1: Prueba

|  |  |
| --- | --- |
| **Comenzado en** | Monday, 12 de October de 2020, 14:43 |
| **Estado** | Terminados |
| **Finalizado en** | Monday, 12 de October de 2020, 14:48 |
| **Tiempo empleado** | 5 mins 44 segundos |
| **Puntos** | 10/10 |
| **Calificación** | **100** de un total de 100 |
| **Comentario de retroalimentación** | Buen trabajo! |

Top of Form

### Pregunta 1

Correcta

Puntúa 1 sobre 1

Señalar con bandera la pregunta

#### **Texto de la pregunta**

Redis, MongoDB, PostgreSQL y CouchDB son ejemplos de bases de datos NoSQL.

Seleccione una:

a. Falso.

b. Verdadero.

#### **Retroalimentación**

PostgreSQL es una base de datos SQL.

La respuesta correcta es: Falso.

### Pregunta 2

Correcta

Puntúa 1 sobre 1

Señalar con bandera la pregunta

#### **Texto de la pregunta**

¿Qué se obtiene con la siguiente línea de código?  
Código:

r.db(‘database’).table(‘usuarios’).insert({nombre: ‘Patrick’, id: 3002})

Seleccione una:

a. Crea un nuevo registros en la tabla usuarios.

b. Actualiza un registro en la tabla usuarios.

c. Crea la tabla usuarios y añade un registro.

d. La línea de código está incorrecta.

#### **Retroalimentación**

Para agregar un nuevo registro se usa el método insert, y se pasa como parámetro un objeto con los atributos y valores del nuevo registro.

La respuesta correcta es: Crea un nuevo registros en la tabla usuarios.

### Pregunta 3

Correcta

Puntúa 1 sobre 1

Señalar con bandera la pregunta

#### **Texto de la pregunta**

¿Con qué función del lenguaje ReQL con el que trabaja REthinkDB, es posible obtener un sólo registro de una tabla?.

Seleccione una:

a. filter.

b. search.

c. get.

d. find.

#### **Retroalimentación**

La función filter obtiene el registro que cumpla con los parámetros que se le especifiquen en el llamado.

La respuesta correcta es: filter.

### Pregunta 4

Correcta

Puntúa 1 sobre 1

Señalar con bandera la pregunta

#### **Texto de la pregunta**

La siguiente sentencia es correcta: INSERT INTO usuarios VALUES('Charles', 23, 'Nueva York', 'Masculino', 2547368945, 'Falso')

Seleccione una:

a. Verdadero

b. Falso

#### **Retroalimentación**

Los campos de la tabla que se van insertar se pueden omitir si los valores se ubican en el mismo orden en que están las columnas de la tabla.

La respuesta correcta es: Verdadero

### Pregunta 5

Correcta

Puntúa 1 sobre 1

Señalar con bandera la pregunta

#### **Texto de la pregunta**

¿Qué diferencia existe entre LEFT JOIN Y RIGHT JOIN?

Seleccione una:

a. LEFT JOIN selecciona los elementos comunes y todos los elementos de la primera tabla de la relación, RIGHT JOIN selecciona los comunes y todos los de la segunda tabla.

b. LEFT JOIN selecciona los elementos comunes y todos los elementos de la segunda tabla de la relación, RIGHT JOIN selecciona los comunes y todos los de la primera tabla.

c. LEFT JOIN selecciona todos los elementos de la primera tabla de la relación sin seleccionar los comunes, RIGHT JOIN selecciona todos los de la segunda tabla sin seleccionar los comunes.

d. LEFT JOIN selecciona todos los elementos de la segunda tabla de la relación sin seleccionar los comunes, RIGHT JOIN selecciona todos los de la primera tabla sin seleccionar los comunes.

#### **Retroalimentación**

Ambas sentencias seleccionan los elementos comunes entre dos tablas, con la diferencia que LEFT JOIN selecciona adicionalmente los elementos de la primera tabla de la relación y RIGHT JOIN los de la segunda.

La respuesta correcta es: LEFT JOIN selecciona los elementos comunes y todos los elementos de la primera tabla de la relación, RIGHT JOIN selecciona los comunes y todos los de la segunda tabla.

### Pregunta 6

Correcta

Puntúa 1 sobre 1

Señalar con bandera la pregunta

#### **Texto de la pregunta**

¿Cuál es la unidad fundamental de una base de datos relacional?

Seleccione una:

a. Tabla.

b. Relación.

c. Datos.

d. Objetos.

#### **Retroalimentación**

La unidad fundamental de una base de datos relacional son las tablas en las que se estructuran los datos según cada uno de sus atributos.

La respuesta correcta es: Tabla.

### Pregunta 7

Correcta

Puntúa 1 sobre 1

Señalar con bandera la pregunta

#### **Texto de la pregunta**

¿Qué es una base de datos?

Seleccione una:

a. Conjunto de datos estructurados que se almacenan en un servidor

b. Programa de computación que gestiona los datos en un servidor

c. Sistema de información web que permite el almacenamiento de datos

d. Conjunto de reglas para el almacenamiento de programas informáticos

#### **Retroalimentación**

Una base de datos es una colección de datos con estructura definida y que se encuentran en un servidor al que pueden acceder múltiples clientes.

La respuesta correcta es: Conjunto de datos estructurados que se almacenan en un servidor

### Pregunta 8

Correcta

Puntúa 1 sobre 1

Señalar con bandera la pregunta

#### **Texto de la pregunta**

Selecciona dos ventajas de las bases de datos NoSQL

Seleccione una o más de una:

a. Nivel de escalabilidad

b. Código abierto

c. SQL para consultas

d. Estructura fija

#### **Retroalimentación**

Las grandes ventajas de las bases de datos NoSQL es su capacidad de escalar y crecer para almacenar grandes cantidades de información sin hacer más complejo el modelo, así como ser en su gran mayoría de código abierto.

Las respuestas correctas son: Nivel de escalabilidad, Código abierto

### Pregunta 9

Correcta

Puntúa 1 sobre 1

Señalar con bandera la pregunta

#### **Texto de la pregunta**

¿Para qué sirve la sentencia ADD CONSTRAINT en una base de datos relacoinal?

Seleccione una:

a. Crea relaciones entre tablas.

b. Añade una llave primaria a una tabla.

c. Crea una nueva tabla.

d. Elimina un campo de una tabla.

#### **Retroalimentación**

Para crear una relación entre dos tablas se usa la sentencia ADD CONTRAINT nombre\_relacion.

La respuesta correcta es: Crea relaciones entre tablas.

### Pregunta 10

Correcta

Puntúa 1 sobre 1

Señalar con bandera la pregunta

#### **Texto de la pregunta**

¿Qué es una llave primaria?

Seleccione una:

a. Una columna de una tabla que sirve como identificador único de cada registro

b. Un atributo de una tabla que permite el almacenamiento de datos

c. Una columna de una tabla que identifica los campos repetidos

d. Una tabla que sirve para acceder a todos los registros de una base de datos

#### **Retroalimentación**

Una llave primaría es un campo único para cada registro en una tabla, oficia como identificador y siempre debe existir en una tabla de una base de datos relacional.

La respuesta correcta es: Una columna de una tabla que sirve como identificador único de cada registro

Bottom of Form