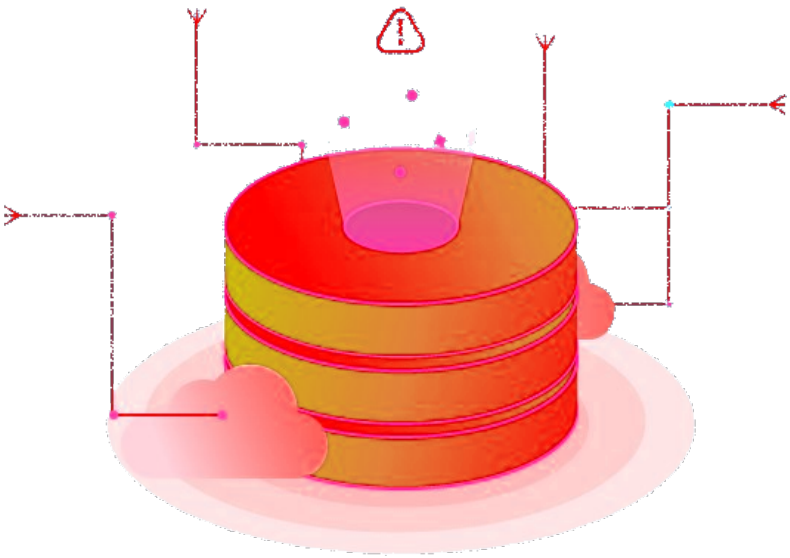
Universidad Nacional Autónoma De México
Facultad De Ciencias, 2024-I
Fundamentos De Bases De Datos



Práctica 05:

Lenguaje para Definición de Datos. DDL, (Data Definition Language).

Profesor:
Gerardo Avilés Rosas

Ayudantes de teoría:
Gerardo Uriel Soto Miranda
Valeria Fernanda Manjarrez Angeles

Ayudantes de laboratorio:
Ricardo Badillo Macías
Jerónimo Almeida Rodríguez

DDL

Un esquema de base de datos se especifica mediante un conjunto de definiciones expresadas mediante un lenguaje especial llamado **lenguaje de definición de datos (DDL)**.

Se debe especificar el almacenamiento y los métodos de acceso usados por el sistema de bases de datos por un conjunto de instrucciones en un tipo especial de DDL denominado **lenguaje de almacenamiento y definición de datos**. Estas instrucciones definen los detalles de implementación de los esquemas de base de datos, que se ocultan usualmente a los usuarios.

Tablas

Una tabla consiste de renglones y columnas. Las columnas definen los datos que queremos almacenar y las filas contienen los valores para esas columnas.

Existen los siguientes tipos de tablas en *PostgreSQL*:

- **Tabla Ordinaria:** Una tabla permanente. La vida útil de la tabla comienza con la creación de la tabla y termina con la eliminación de la tabla. Las tablas ordinarias deben pertenecer a un esquema, si no se especifica un *schema* a la tabla, la tabla se asigna el esquema actual.
- **Tabla Temporal:** Tablas no permanentes. La vida útil de la tabla es la sesión del usuario. Tablas temporales se eliminan al finalizar la sesión del usuario o al termino de una transacción. Estas tablas pertenecen a un *schema* especial, por lo que no se debe especificar el esquema al momento de crear la tablas. El nombre de la tabla debe ser diferente a cualquier objeto de la base de datos.
- **Unlogged table:** Tienen un tiempo de escritura menor que las tablas ordinarias o temporales, esto se debe a que los datos no se escriben en los archivos *WAL*. *Unlogged tables* no son *crash-safe*, por lo que si la base de datos tiene algún fallo, los datos almacenados en este tipo de tablas no se pueden recuperar.
- **Tabla hija:** Son tablas que heredan de una o mas tablas ordinarias. Modificaciones a los padres se propagan a las tablas hijas. La herencia a menudo se uso con exclusión de restricciones para particionar fisicamente los datos en el disco duro y mejorar el rendimiento al recuperar un subconjunto de datos que tiene un cierto valor.

Crear tablas:

Crea una nueva tabla en *PostgreSQL*.

```
-- Tipo de base de datos.
CREATE [ [GLOBAL | LOCAL] {TEMPORARY | TEMP} | UNLOGGED]
-- Nombre de base de datos y definición de columnas.
TABLE [ IF NOT EXISTS ] [database_name].[schema_name].table_name ( [
    { column_name data_type [ COLLATE collation ] [ column_constraint [ ... ] ]
        | table_constraint
        | LIKE source_table [ like_option ... ] }
        [, .... ]
] )
-- Referencia a una tabla padre.
[ INHERITS ( parent_table [, ... ] ) ]
-- Tablespace a almacenar la tabla.
[ TABLESPACE tablespace_name ]
```

database_name: Es el nombre de la base de datos en la que se crea la tabla. `database_name` debe especificar el nombre de una base de datos existente. Si no se especifica, `database_name` el valor predeterminado es la base de datos actual.

schema_name: Es el nombre del esquema al que pertenece la nueva tabla. Si no se especifica un esquema, los objetos creados se agregan a el esquema **public**.

table_name: Es el nombre de la nueva tabla. Los nombre de tabla deben seguir las convenciones de nombrado para identificadores.

Sintaxis completa para crear una tabla se puede encontrar en:

`https://www.postgresql.org/docs/current/sql-createtable.html`.

Tipos de datos nativos.

En *PostgreSQL*, cada columna, variable local, expresión y parámetro tiene un tipo de datos relacionado. Un tipo de datos es un atributo que especifica el tipo de datos que el objeto puede contener.

Se debe de considerar el tipo de dato de cada columna antes de crear una tabla en la base de datos. Cambiar un tipo de dato en un ambiente productivo es una operación costosa, la cual puede bloquear la tabla y en algunos casos, es necesario reescribir la tabla. Por lo que escoger el tipo de dato correcto puede resultar complejo, aquí algunas aspectos a considerar al momento de escoger los tipos de datos para tu tabla:

- **Extensibilidad:** Considerar si es posible cambiar la longitud del tipo de dato sin necesidad de construir la tabla.
- **Tamaño del tipo de dato:** Escoger un tamaño que se ajuste a la necesidad del dato. Asignar tipos de datos muy grandes pueden consumir más recursos de los necesarios.
- **Soporte:** Considerar que las tablas pueden ser consumidas por aplicaciones escritas en diferentes lenguajes de programación. Por lo que se recomienda utilizar tipos de datos soportados por estos lenguajes, de otra forma se tendrá que trabajar extra para serializar y desearizar los datos.

Númericos:

- **Enteros:** Tipos de datos numéricos exactos que utilizan datos enteros.

Tipo de datos.	Intervalo.	Peso.
bigint.	De -9,223,372,036,854,775,808 a 9,223,372,036,854,775,807.	8 bytes.
integer.	De -2,147,483,648 a 2,147,483,647.	4 bytes.
smallint.	De -32,768 a 32,767.	2 bytes.

- **Decimal o númericos:** Tipos de datos numéricos que tienen precisión y escala fijas. *Decimal* y *numeric* son sinónimos y se pueden usar indistintamnete.
  - **decimal[(p[,s])]** y **numeric[(p[,s])]**: Números de precisión y escala fijas. Cuando se utiliza la precisión máxima, los valores válidos se sitúan entre $-10^{38}+1$ y $10^{38}-1$ numérico es funcionalmente equivalente a decimal.
  - **p (precisión)**: El número total máximo de dígitos decimales que almacenará, tanto a la izquierda como a la derecha del separador decimal. La precisión debe ser un valor comprendido entre 1 y la precisión máxima de 38. La precisión predeterminada es 18.
  - **s (escala)**: El número de dígitos decimales que se almacenará a la derecha del separador decimal. Este número se resta de p para determinar el número máximo de dígitos a la izquierda del separador decimal. El número máximo de dígitos decimales que se puede almacenar a la derecha del separador decimal. Escala debe ser un valor comprendido entre 0 y p. Sólo es posible especificar la escala si se ha especificado la precisión. La escala predeterminada es 0.
- **real**: Al menos 6 dígitos de precisión y puede variar dependiendo de la de la plataforma. 4 bytes.
- **double precision**: Al menos 15 dígitos de precisión y puede variar dependiendo de la plataforma. 8 bytes.

Caracteres:

Son tipos de datos de cadena de longitud fija o de longitud variable.

- **char**: Equivalente a **char[1]**. Longitud máxima 1.
- **name**: Equivalente a **varchar[64]**. Longitud máxima 64.
- **char [(n)]**: Datos de cadena no `UNICODE` de longitud fija `n`. `n` define la longitud de cadena y debe ser un valor entre 1 y 10485760. El tamaño de almacenamiento es n bytes.
- **varchar [(n — max)]**: Datos de cadena no UNICODE de longitud variable. **n** define la longitud de cadena y puede ser un valor entre 1 y 10485760. **max** indica que el tamño máximo de almacenamiento es 1 GB. El tamaño de almacenamiento es la longitud real de los datos especificados + 2 *bytes*.
- **text**: Longitud limitada.

Fechas:

Tipo de datos que almacenan datos de fechas.

- **time:** Define un tiempo de un día. La hora es sin conocimiento de zona horaria y se basa en un reloj de 24 horas.
- **date:** Define una fecha con el formato `YYYY-MM-DD`.

Para completa descripción de tipos de datos, referirse a la siguiente documentación:

`https://www.postgresql.org/docs/11/datatype.html`.

Integridad de datos

Restricciones:

Las restricciones le permiten definir la manera en que el motor de base de datos exigirá automáticamente la integridad de una base de datos. Las restricciones definen reglas relativas a los valores permitidos en las columnas y constituyen al mecanismo estándar para exigir la integridad. El uso de restricciones es preferible al uso de desencadenadores, reglas y valores predeterminados. El optimizador de consultas también utiliza definiciones de restricciones para generar planes de ejecución de consultas de alto rendimiento.

Restricciones de Dominio:

- **NOT NULL**: La restricción *NOT NULL* **especifica que la columna no acepta valores NULL**. Un valor *NULL* no es lo mismo que cero (0), en blanco o que una cadena de caracteres de longitud cero. **NULL, signfica que no hay ninguna entrada**.
  La presencia de un valor *NULL* suele implicar que el valor es desconocido o no está definido. Se recomienda evitar la aceptación de valores *NULL*, dado que pueden implicar una mayor complejidad en las consultas y actualizaciones.
  Por otra parte hay opciones para las columnas como las restricciones **PRIMARY KEY**, que no pueden utilizarse con columnas que aceptan valores *NULL*.
- **CHECK:** La restricción *CHECK* exige la integridad del dominio mediante la **limitación de los valores que se pueden asignar a una columna**. Una restricción *CHECK* especifica una condición de búsqueda *booleana* (se evalúa como **TRUE o FALSE**) que se aplica a todos los valores que se indican en la columna. Se rechazan todo los valores que se evalúan como **FALSE**. En una misma columna se pueden especificar varias restricciones **CHECK**. Las condiciones de búsqueda booleanas son las siguientes: $=$, $<>$, $>$, $<$, $<=$, $>=$, `between`, `in`.
- **DEFAULT:** A una columna se le puede asignar un valor predeterminado. Cuando se crea una nueva fila y no se especifican los valores para algunas de las columnas, esas columnas se rellenarán con sus respectivos valores predeterminados. Si no se declara explícitamente ningún valor predeterminado, el valor predeterminado es el valor *NULL*.

Restricciones de integridad de entidad

- **PRIMARY KEY**: La restricción *PRIMARY KEY* **identifica la columna o el conjunto de columnas cuyos valores identifican de forma exclusiva a cada una de las filas de una tabla**. Dos filas de la tabla no pueden tener el mismo valor de llave primaria. No se pueden asignar valores NULL a ninguna de las columnas de una llave primaria. Todas las tablas tienen que tener una llave primaria.
- **UNIQUE:** La restricción *UNIQUE* exige **la unicidad de los valores de un conjunto de columnas**. En una restricción *UNIQUE*, dos filas de la tabla NO pueden tener el mismo valor en las columnas. Las claves principales también exigen exclusividad, pero no aceptan *NULL* como uno de los valores exclusivos, en cambio una llave candidata si acepta valores NULL. Una columna o combinación de columnas certificadas con la restriccion UNIQUE se denomina llave candidata.

Restricciones de integridad referencial

- **FOREIGN KEY**: La restricción *FOREIGN KEY* identifica y exige las relaciones entre las tablas. Una llave foránea de una tabla apunta a una llave primaria de otra tabla. No se puede insertar una fila que tenga un valor de llave foránea (excepto *NULL*) si no hay una llave primaria con dicho valor.

Alter table

El comando **ALTER TABLE**, nos permite realizar modificaciones a nuestras tablas, tales como: añadir nuevas columnas, renombrar la estructura o cambiar el tipo de dato.
Para esta práctica lo utilizaremos para manejar **la integridad de datos**. Esto es con la finalidad de mantener una estructura de nuestras tablas con los atributos y el tipo de atributo en la definición de cada tabla, y las restricciones aparte utilizando el comando.

Para mayor información referirse a la siguiente documentación:

`https://www.postgresql.org/docs/current/sql-altertable.html`.

Actividades.

i. Deberán realizar un archivo *SQL* de las instrucciones *DDL* para crear el esquema de base de datos utilizando como base el diagrama relacional que se realizó en la práctica pasada. Y llamarán el archivo como: **DDL**.
IMPORTANTE: Los nombres de las tablas y columnas deben ser las mismas que utilizaron para su modelo relacional. Deberan escoger el tipo de dato correcto para cada columna de acuerdo a las necesidades del caso de uso. También deberán definir las restricciones de dominio necesarias para cada una de las columnas definidas en cada tabla. Y deberán especificar las llaves primarias, compuestas y foráneas para cada tabla.
ii. Deberán realizar un documento *PDF* donde especifiquen las restricciones de dominio para cada columna. Este documento se llamará **Práctica05**.
iii. Deberán incluir los diagramas de las practicas pasadas, el Modelo Entidad Relación y el Modelo Relacional (los archivo `.png` y `.drawio`).

Figura 1: *Actividades.*

Entregables.

Deberán subir un archivo con formato *zip* a *Google Classroom*, de acuerdo a lo indicado en los lineamientos de entrega. Debe de estar organizado de la siguiente manera, (suponiendo que el nombre del equipo que está entregando es *Dream Team*).

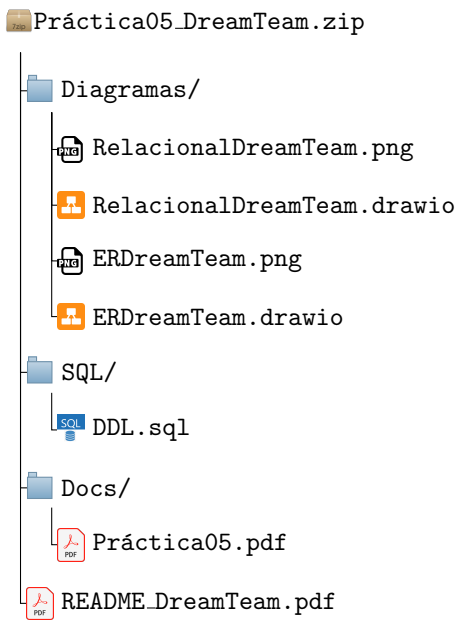


Figura 2: *Entregables.*

Nota.

Para cualquier duda o comentario que pudiera surgirles al hacer este trabajo, recuerden que cuentan con la asignación de este entregable en el grupo de *Classroom*, en donde seguramente encontrarás las respuestas que necesites.