1. Para empezar

1. En este taller utilizaremos PostgreSQL, un DBMS de código abierto. Éste se encuentra instalado en Linux en la sala Ada Lovelace y tiene creado un usuario *lovelace* y una base de datos por defecto del mismo nombre.

2. Para empezar cree una nueva base de datos desde la terminal con el comando

   ```
   createdb unidb
   ```

3. Ahora inicie Postgresql conectándose a la base de datos que acaba de crear con el comando

   ```
   psql -d unidb
   ```

4. Verifique la información de conexión a la base de datos con el comando

   ```
   \conninfo
   ```

5. Ahora está listo para realizar operaciones CRUD (Create, Read, Update, Delete). Recuerde que en cualquier momento puede terminar la conexión con el comando

   ```
   \q
   ```

2. Definición de Datos en SQL

El DDL de SQL nos permite especificar el esquema de las relaciones, los tipos de los valores de cada atributo, restricciones de integridad, índices, información de seguridad, entre otros.

Tipos de datos

1. **char(n)**: cadena de caracteres de tamaño fijo n (se completa con espacios si se usan menos caracteres o se compara con otra cadena de tamaño mayor)
2. **varchar(n)**: cadena de caracteres de tamaño variable con máximo n caracteres
3. **int**: entero
4. **smallint**: entero con menor dominio
5. **numeric(p,d)**: número de punto fijo con p digitos, de los cuales d están después del punto decimal
6. **real**: número de punto flotante
7. **double**: número de punto flotante de doble precisión
8. **float(n)**: número de punto flotante con precisión de al menos n dígitos

Definición de esquemas

1. El comando `create table` crea una relación. El siguiente comando crea la relación *unid_acad* para registrar las unidades académicas de la universidad.

```
create table unid_acad
  (nombre_unid varchar (20),
  edificio varchar (15),
  presupuesto numeric (12,2),
  primary key (nombre_unid));
```

   Note que se definen los atributos en la relación, su dominio o tipo y la llave primaria.

2. La última línea `primary key (A1,...,Ak)` es una restricción de integridad. Define la combinación de atributos $A_1, \ldots, A_k$ como la llave primaria. Estos atributos deben ser no nulos y únicos en los registros en la relación.

3. Otra restricción de integridad que se puede agregar sirve para especificar la llave foránea. La restricción en este caso es

```
foreign key (A1,...,Ak) references rel
```

   Esto quiere decir que los atributos $A_1, \ldots, A_k$ en *esta* relación constituyen una llave primaria en la relación *rel*.

4. Para asegurar que un campo sea no nulo se puede incorporar una restricción de integridad así

```
create table unid_acad
  (nombre_unid varchar (20),
  edificio varchar (15) not null,
  presupuesto numeric (12,2),
  primary key (nombre_unid));
```

   Los campos de la llave primaria siempre deben ser no nulos.

Insertar registros, eliminar y alterar relaciones

1. El siguiente comando inserta un registro en la relación.

```
insert into unid_acad
        values ('MACC', 'Cabal', 100.0);
```

   Note que los valores deben insertarse en el orden en que se definieron los campos en el esquema.

2. Para eliminar todos los registros de una relación se usa el comando

```
delete from unid_acad;
```

3. Para eliminar una relación de la base de datos se usa el comando

```
drop table unid_acad;
```

4. Para agregar un atributo a una relación usamos el comando

```
alter table unid_acad add num_est int;
```

Todos los registros existentes fijan el valor del nuevo campo como null para los registros existentes.

5. Para eliminar un atributo de una relación usamos el comando

```
alter table unid_acad drop num_est;
```

3. Crear algunas tablas

En su base de datos existente borre todas las tablas, o cree una nueva base de datos, y cree las siguientes tablas:

```
create table unid_acad
  (nombre_unid varchar (20),
  edificio varchar (15) not null,
  presupuesto numeric (12,2),
  primary key (nombre_unid));


create table curso
  (curso_cod varchar (7),
  nombre varchar (50),
  nombre_unid varchar (20),
  creditos numeric (2,0),
  primary key (curso_cod),
  foreign key (nombre_unid) references unid_acad);


create table instructor
  (inst_ID varchar (5),
  nombre varchar (20) not null,
  nombre_unid varchar (20),
  salario numeric (8,2),
  primary key (inst_ID),
  foreign key (nombre_unid) references unid_acad);


create table grupo
  (curso_cod varchar (8),
  grupo_cod varchar (8),
  semestre varchar (6),
  ano numeric (4,0),
  primary key (curso_cod, grupo_cod, semestre, ano),
  foreign key (curso_cod) references curso);


create table dicta
  (inst_ID varchar (5),
  curso_cod varchar (8),
  grupo_cod varchar (8),
```

```
semestre varchar (6),
ano numeric (4,0),
primary key (inst_ID, curso_id, grupo_id, semestre, ano),
foreign key (curso_id, grupo_id, semestre, ano) references grupo,
foreign key (inst_ID) references instructor);
```

Agregue algunos registros a cada una de las tablas.

4. Consultas básicas

Consultas sobre una sola relación

1. Las consultas básicas en SQL se pueden escribir con las cláusulas `select`, `from` y `where`. Recuerde que toda operación genera como resultado una nueva relación. Para seleccionar eñ nombre de todas las unidades académicas podemos ejecutar el siguiente comando

   ```
   select nombre_unid from unid_acad;
   ```

2. Los nombres de todos las unidades académicas a las que están asociados los profesores podemos ejecutar el siguiente comando

   ```
   select nombre_unid from instructor;
   ```

   En este caso el resultado no es una relación porque hay registros repetidos (varios instructores pertenecen a la misma unidad). Si se busca que el resultado sea una relación se pueden eliminar los duplicados con el comando

   ```
   select distinct nombre_unid from instructor;
   ```

3. Para hacer una proyección modificando los atributos podemos por ejemplo ejecutar el siguiente comando

   ```
   select inst_ID, nombre, salario*2 from instructor;
   ```

   Otras operaciones aritméticas también están disponibles ($+$, $-$, $/$). La relación resultante es nueva. La relación *instructor* no se ha modificado.

4. La cláusula `where` permite seleccionar registros que cumplan con una condición. Por ejemplo, para seleccionar todos los cursos de MACC que tienen al menos 3 créditos podemos ejecutar el comando

   ```
   select nombre from curso
   where nombre_unid = 'MACC' and creditos >= 3;
   ```

   Es posible usar los operadores lógicos `and`, `or` y `not`. Se pueden usar los operadores de comparación $<$, $<=$, $>$, $>=$, $=$ y $<>$, para comparar números, cadenas de caracteres, fechas.

Consultas sobre varias relaciones

1. En SQL podemos acceder a varias relaciones para extraer información almacenada en cada una de ellas. Por ejemplo, si queremos listar los nombres de todos los profesores con el edificio donde queda ubicada la respectiva unidad académica podemos ejecutar el siguiente comando

```
select nombre, instructor.nombre_unid, edificio
from instructor, unid_acad
where instructor.nombre_unid = unid_acad.nombre_unid;
```

   Como el atributo `nombre_unid` aparece en las dos relaciones, es necesario usar el nombre de la relación para diferenciarlos. Los nombres de los atributos seleccionados en la cláusula `select` deben ser diferentes.

2. La cláusula `from` se puede entender como un producto cartesiano de las tablas que se utilizan. Sobre este producto la cláusula `where` selecciona los registros que cumplen con un predicado. Finalmente, la cláusula `select` selecciona algunos atributos. En otro ejemplo, para listar los instructores y sus clases podemos ejecutar el comando

```
select nombre, curso_cod
from instructor, dicta
where instructor.inst_ID = dicta.inst_ID;
```

   Como el atributo `nombre_unid` aparece en las dos relaciones, es necesario usar el nombre de la relación para diferenciarlos. Los nombres de los atributos seleccionados en la cláusula `select` deben ser diferentes.

Join natural

1. El join natural une relaciones usando los atributos que tienen en común. El join natural genera una nueva relación con los registros de las dos relaciones combinados siempre que el valor de sus atributos comunes coincida. Por ejemplo, la última operación realizada se puede obtener también con el siguiente comando

```
select nombre, curso_cod
from instructor natural join dicta;
```

   Note que el único atributo en común es precisamente `inst_ID`.

2. El resultado del natural join es una relación y se puede usar para encadenarlo con otras operaciones. Por ejemplo, para listar todos los IDs de los instructores con los códigos de los cursos que dictan podemos ejecutar el siguiente comando

```
select inst_ID, curso_cod
from instructor natural join dicta, curso
where dicta.curso_cod = curso.curso_cod;
```

3. Alternativamente se puede usar `join using` para hacer un join que use solo algunos atributos comunes, no todos. Por ejemplo, para listar todos los IDs de los instructores con los códigos de los cursos que dictan podemos ejecutar el siguiente comando

```
select inst_ID, curso_cod
from (instructor natural join dicta)
join curso using (curso_cod);
```

Renombrar atributos y relaciones

1. La cláusula `as` se puede utilizar para renombrar los atributos en la relación que se obtiene como resultado. Por ejemplo, en el siguiente comando cambiamos el nombre del primer atributo seleccionado

   ```
   select nombre as nombre_inst, curso_cod
   from instructor, dicta
   where instructor.inst_ID = dicta.inst_ID;
   ```

2. La cláusula `as` también se puede utilizar para renombrar las relaciones consideradas. Por ejemplo, el comando anterior lo podríamos escribir como

   ```
   select nombre, curso_cod
   from instructor as I, dicta as D
   where I.inst_ID = D.inst_ID;
   ```

   Esto puede ser útil cuando se requiere comparar una relación consigo misma. Por ejemplo, si se quiere determinar los cursos que tienen al menos tantos créditos como al menos un curso de MACC, ejecutaríamos el comando

   ```
   select distinct A.nombre
   from curso as A, curso as B
   where A.creditos > B.creditos  and B.nombre_unid = 'MACC';
   ```

   Estos nombres para las relaciones se conocen como correlation name, table alias, o correlation variables.