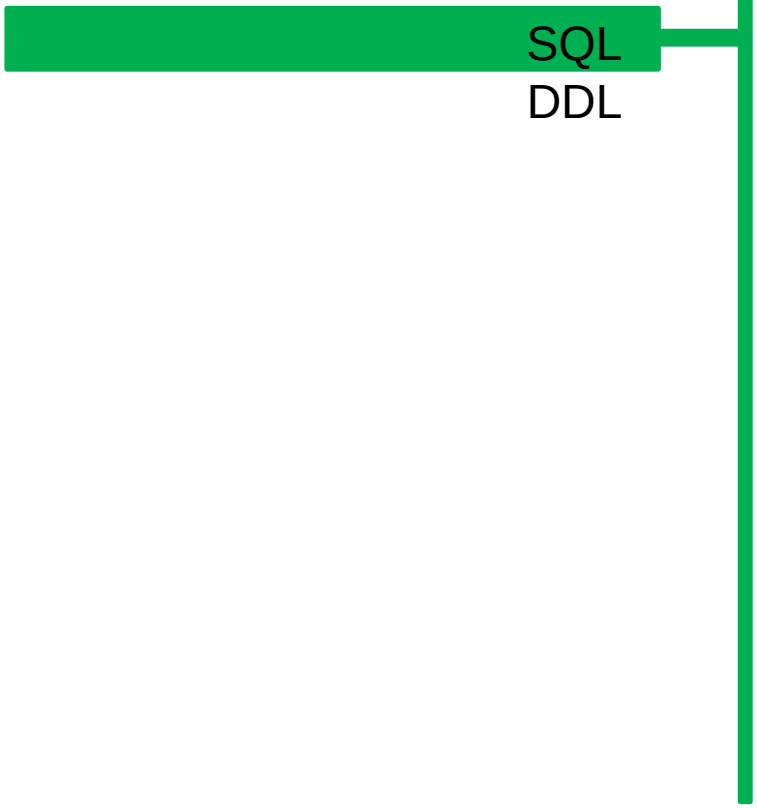
Maestría en Bioinformática

Bases de Datos y Sistemas de Información

SQL: DDL

Ing. Alfonso Vicente, PMP
alfonso.vicente@logos.com.uy

Agenda



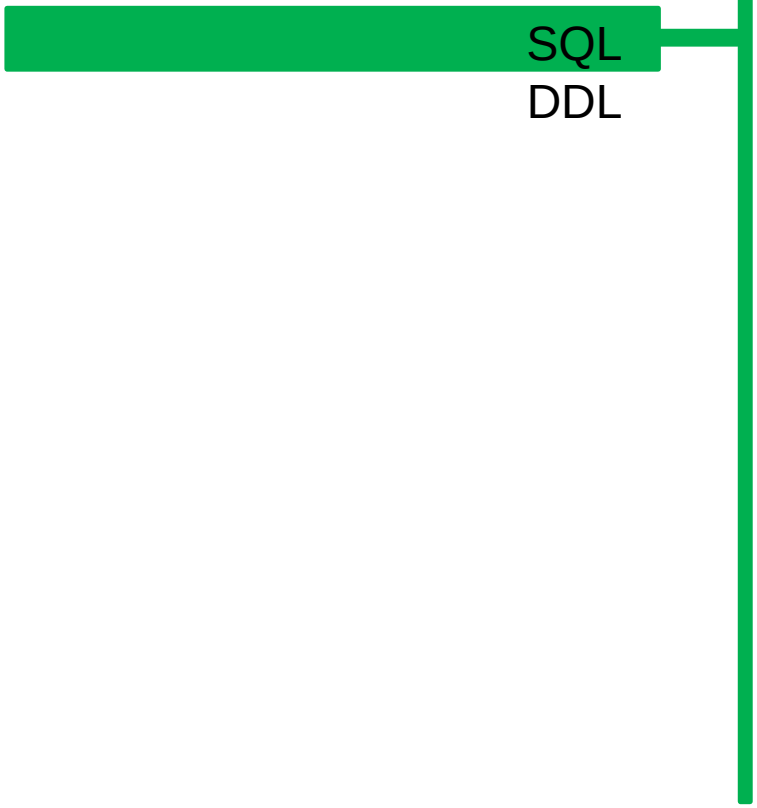
- Lenguaje SQL
- Sub-lenguajes
- Dialectos

Agenda

SQL
DDL

- Esquemas
- Objetos de Base de Datos
- Sentencias de DDL
- Sentencia CREATE TABLE
- Tipos de datos
- Primary Keys
- Unique Keys
- Foreign Keys
- Checks

Agenda



- Lenguaje SQL
- Sub-lenguajes
- Dialectos

SQL

Lenguaje SQL

- Presentado en 1974 por Chamberlin y Boyce (SEQUEL)
- Objetivos:
  - Lenguaje declarativo de alto nivel, con el mismo poder expresivo que el álgebra y el cálculo relacional
  - Será procesado por un compilador-optimizador
  - Parecido al inglés, sin términos técnicos
  - Unificar acceso a datos y tareas administrativas

SQL

Lenguaje SQL

- 1986 – ANSI Standard
- 1987 – ISO Standard
- SQL-89 (mejoras en la integridad)
- SQL-92 (tipos de datos, operadores de conjuntos)
- SQL:1999 (triggers, LOBS, recursión)
- SQL:2003 (merge, XML, secuencias, autonumerados)
- SQL:2008 y SQL:2011

SQL

Sub-lenguajes

- Data Definition Language (DDL)

  `CREATE / ALTER / DROP`

- Data Manipulation Language (DML)

  `INSERT / UPDATE / DELETE`

- Query Language (QL)

  `SELECT`

- Data Control Language (DCL)

  `GRANT / REVOKE`

SQL

Dialectos

- SQL es un estándar, una especificación
- Los RDBMSs lo implementan y lo respetan en mayor o menor medida, agregando elementos propios
- Es necesario conocer el dialecto de SQL del RDBMS que se utiliza (Oracle, DB2, Informix, SQL Server, MySQL, PostgreSQL, etc.)

Agenda

SQL
DDL

- Esquemas
- Objetos de Base de Datos
- Sentencias de DDL
- Sentencia CREATE TABLE
- Tipos de datos
- Primary Keys
- Unique Keys
- Foreign Keys
- Checks

DDL

Esquemas

- La mayoría de los RDBMSs soportan múltiples esquemas, que son una forma de agrupar lógicamente los objetos
- Un ERP podría tener esquemas GL (contabilidad general), AP (cuentas a pagar), AR (cuentas a cobrar), HR (recursos humanos), etc
- Normalmente hay esquemas Information_schema, SYS, SYSCAT, etc para los objetos del sistema, como el catálogo
- Los RDBMSs tienen objetos de base y objetos de esquema

DDL

Esquemas

- Objetos globales: tablespaces, bufferpools, usuarios, roles
- Objetos de esquema: tablas, constraints, índices, vistas, triggers, secuencias, código almacenado (procedimientos, funciones, paquetes)
- Ejemplos:
  - Mediawiki
  - Homozygositymapper
- Un esquema define un namespace, como un apellido para los objetos, aunque se indica primero (esquema.objeto)

DDL

Objetos de Base de Datos

- Tablespaces: Objetos que permiten gestionar el almacenamiento físico de los objetos en archivos
- Bufferpools: Objetos que permiten especificar detalles sobre la gestión de la memoria
- Usuarios y roles: Objetos para implementar mecanismos de autenticación (mediante usuario y password) y autorización (los roles simplifican la gestión de los privilegios)

DDL

Objetos de Base de Datos

- Tablas: Los objetos centrales del MR, mantienen los datos en estructuras de filas y columnas (con nombre y tipo de datos)
- Constraints: Permiten implementar las restricciones del modelo (NOT NULL, PKs, UKs, FKs, CHECKs)
- Índices: Permiten mejorar el desempeño en el acceso a los datos
- Vistas: Permiten almacenar una consulta con un nombre

DDL

Objetos de Base de Datos

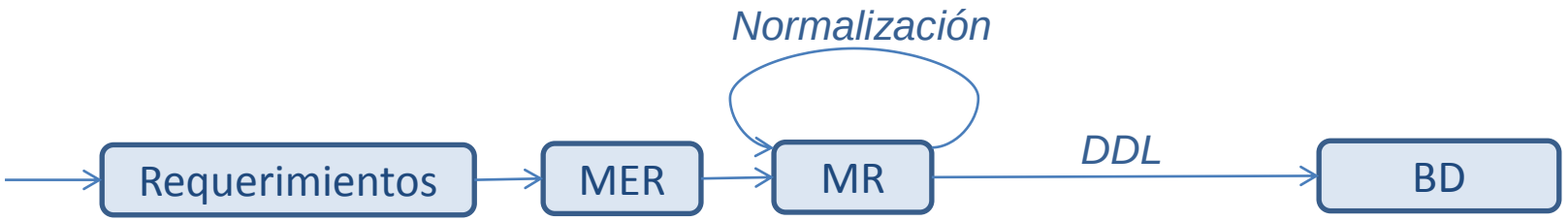
- Triggers: Permiten disparar acciones justo antes o después de ejecutar sentencias
- Secuencias: Permiten generar números siempre distintos
- Código almacenado: Permite escribir código procedural (funciones y procedimientos) que quede almacenado en la base de datos
- Según el RDBMS puede haber otros objetos

DDL

Sentencias de DDL

- En el Data Definition Language hay sentencias para crear (CREATE), modificar (ALTER) y eliminar (DROP) todos los objetos de Base de Datos

- Nos centraremos en la sentencia CREATE TABLE, ya que es la que nos permitirá comenzar a implementar nuestros diseños lógicos



DDL

Sentencia CREATE TABLE

- Los nombres de las tablas y las columnas, así como las constraints, se derivan directamente del modelo lógico

- En caso de no tenerlo ya en el modelo, debemos pensar en el tipo de datos, y si aceptaremos o no valores nulos

```
CREATE TABLE <nombre_tabla> (
   <nombre_columna_1> <tipo_dato_1> [NOT NULL],
   <nombre_columna_2> <tipo_dato_2> [NOT NULL],
   ...
   <nombre_columna_n> <tipo_dato_n> [NOT NULL]
);
```

DDL

Tipos de datos

- Numéricos
  - Exactos
    - Enteros, según el RDBMS: SMALLINT, INTEGER
    - Con precisión y escala, según el RDBMS: NUMERIC(X,Y) o NUMBER(X,Y)
  - Aproximados
    - De punto flotante, según el RDBMS: REAL/DOUBLE

DDL

Tipos de datos

- Texto
  - De largo fijo
    - CHAR(N), los caracteres no usados a la derecha se rellenan con espacios en blanco
  - De largo variable
    - VARCHAR(N), se mantiene el largo en un entero
  - Los caracteres deben ser parte del juego de caracteres de la base (UNICODE es un juego de caracteres que soporta internacionalización)

DDL

Tipos de datos

- Booleanos
  - No todos los RDBMSs lo soportan
- Fecha y hora
  - DATE: año, mes y día
  - TIME: hora, minuto y segundo
  - TIMESTAMP: DATE + TIME
  - El formato dependerá de la configuración del *“National Language”*

DDL

Tipos de datos

- Objetos grandes (LOBs: Long OBjects)
  - CLOB: Character Long Object
    - Permite almacenar textos muy grandes, que superen los límites de VARCHAR
    - Los caracteres deben ser parte del juego de caracteres de la base
  - BLOB: Binary Long Object
    - Permite almacenar archivos binarios, como imágenes o documentos

DDL

Tipos de datos

- Todos los tipos de datos aceptan el valor NULL
- Los límites dependen de la implementación, por lo que varían de un RDBMS a otro
- El tipo de datos de una columna debe permitir almacenar todos los valores previstos, pero usar un tipo demasiado grande puede desperdiciar espacio y limitar el desempeño (no usar TIMESTAMP si alcanza con DATE, no usar CLOB si alcanza con VARCHAR, etc)

DDL

Primary keys

- Podemos definirlas: 1) inline, 2) después de definir las columnas o 3) mediante un ALTER TABLE

- Inline:

```
CREATE TABLE <nombre_tabla> (
   <columna_1> <tipo_dato_1> [NOT NULL] [PRIMARY KEY],
   <columna_2> <tipo_dato_2> [NOT NULL] [PRIMARY KEY],
   ...
   <columna_n> <tipo_dato_n> [NOT NULL] [PRIMARY KEY]
);
```

- Sólo una de las columnas puede ser definida como PK de esta forma (sólo sirve para claves por una única columna)

DDL

Primary keys

- Después de definir las columnas

```
CREATE TABLE nombre_tabla (
   columna_1 tipo_dato_1 [NOT NULL],
   columna_2 tipo_dato_2 [NOT NULL],
   ...
   columna_n tipo_dato_n [NOT NULL],
   CONSTRAINT PK_nombre_tabla
   PRIMARY KEY (columna_1, columna_2)
);
```

- Esta opción permite definir una PK por dos columnas o más

DDL

Primary keys

- Mediante un ALTER TABLE

```
CREATE TABLE <nombre_tabla> (
   <columna_1> <tipo_dato_1> [NOT NULL],
   <columna_2> <tipo_dato_2> [NOT NULL],
   ...
   <columna_n> <tipo_dato_n> [NOT NULL]
);

ALTER TABLE <nombre_tabla>
ADD CONSTRAINT PK_nombre_tabla
PRIMARY KEY (columna_1, columna_2);
```

- Esta opción también permite definir una PK por dos columnas o más

DDL

Unique Keys

- Se definen de la misma forma que una Primary Key, simplemente cambiando PRIMARY KEY por UNIQUE

```
CREATE TABLE nombre_tabla (
   columna_1 tipo_dato_1 [NOT NULL],
   columna_2 tipo_dato_2 [NOT NULL],
   ...
   columna_n tipo_dato_n [NOT NULL]
);
```

```
ALTER TABLE <nombre_tabla>
ADD CONSTRAINT UK_nombre_tabla
UNIQUE (columna_1, columna_2);
```

DDL

Foreign Keys

- Se definen de forma parecida a una PK o UK mediante ALTER TABLE

```
CREATE TABLE tabla1 (
   columna_1 tipo_dato_1 [NOT NULL],
   columna_2 tipo_dato_2 [NOT NULL],
   ...
   columna_n tipo_dato_n [NOT NULL]
);

ALTER TABLE tabla1
ADD CONSTRAINT FK_tabla1_tabla2
FOREIGN KEY (columna_1) references tabla2(columna_1);
```

DDL

Checks

- Se pueden definir inline o a través de ALTER TABLE

```
CREATE TABLE tabla1 (
   columna_1 tipo_dato_1 [NOT NULL],
   columna_2 tipo_dato_2 [NOT NULL],
   ...
   columna_n tipo_dato_n [NOT NULL]
);

ALTER TABLE tabla1
ADD CONSTRAINT check_columna_1_tabla1
CHECK (columna_1 in ('V1', 'V2', ..., 'Vn'));
```

DDL

Ejemplo en Oracle Database

- Abrimos una consola en la máquina virtual, nos conectamos como DBAs y creamos un usuario:

```
[oracle@localhost ~]$ sqlplus / as sysdba

SQL> create user alfonso identified by mypass;

User created.

SQL> grant connect, resource to alfonso;

Grant succeeded.

SQL> conn alfonso/mypass
Connected.
```

DDL

Ejemplo en Oracle Database

- Creamos una tabla con PK y UK definidas inline:

```
SQL> create table elementos(
  2      simbolo char(2) not null primary key,
  3      nombre varchar(30) not null unique
  4  );

Table created.
```

DDL

Ejemplo en Oracle Database

- Creamos una nueva tabla con una FK a la anterior:

```
SQL> create table imagenes_elemento(
  2     simbolo char(2) not null,
  3     imagen blob not null
  4  );

Table created.

SQL> alter table imagenes_elemento
  2  add constraint fk_imagenes_elemento
  3  foreign key (simbolo) references elementos(simbolo);

Table altered.
```