2. Bases de datos relacionales

Class	Bases de datos
Column	Ⓧ Xerach Casanova
Lass Edited time	@Mar 29, 2021 9:22 PM

1. Modelo de datos

Un modelo de datos es un conjunto de métodos y reglas que indican como se ha de almacenar la información y como se han de manipular los datos.

En informática el modelo de datos se implementa con un lenguaje utilizado para la descripción de la BBDD. Se pueden describir las estructuras de datos (tipos y relaciones), restricciones de integridad (condiciones que se deben cumplir según las necesidades de nuestro modelo en la realidad) y las operaciones de manipulación de datos (insertar, modificar, borrar).

Este lenguaje está dividido en dos sublenguajes:

- **Lenguaje de definición de datos (DDL).** Describe de una forma abstracta las estructuras de datos y las restricciones de integridad.
- **Lenguaje de manipulación de datos DML.** Describe las operaciones de manipulación de los datos.

Las fases de modelado se caracterizan por el nivel de abstracción de las mismas (lo alejado que estén del mundo real).

1. **Modelo de datos conceptual.** Representación normalmente gráfica de estructuras de datos y restricciones de integridad. Se realizan en fase de análisis y representan los elementos que intervienen y sus relaciones. Cualquier error en esta fase se arrastra a las siguientes.
2. **Modelo lógico.** Criterios de almacenamiento y operaciones de manipulación de datos dentro de un tipo de entorno informático.
3. **Modelo Físico:** se utiliza en el diseño físico y es la implementación física del modelo anterior. Estructuras de bajo nivel dentro del sistema gestor de bases de datos.

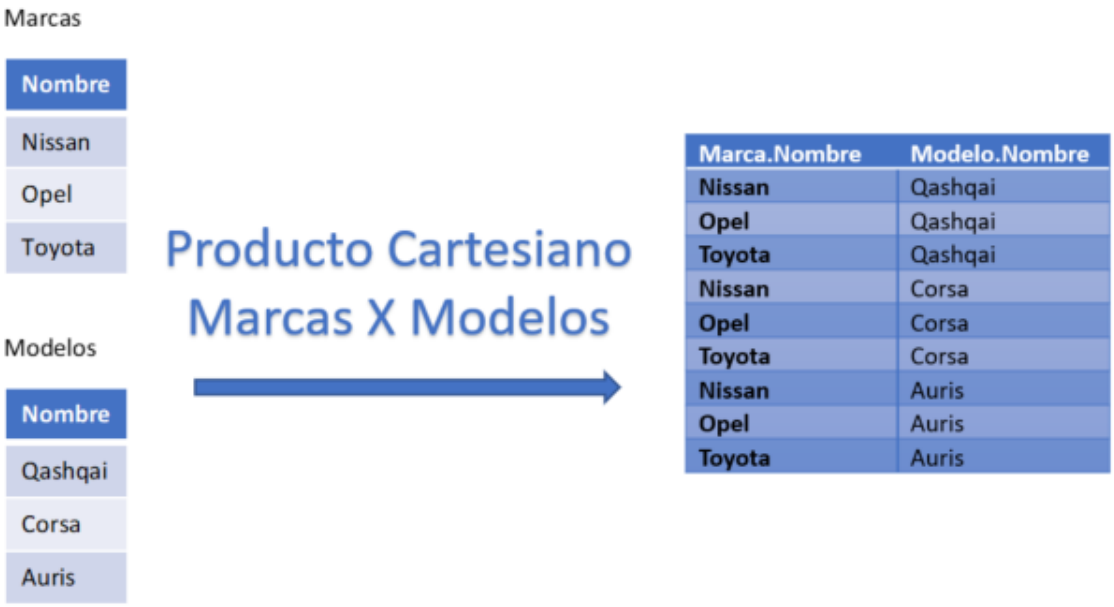
2. Terminología del modelo relacional

Se trata de un modelo lógico que establece una estructura sobre los datos, independientemente del modo en que luego los almacenemos.

El diseño de las tablas del modelo relacional es la segunda fase del diseño de la BBDD, ya que previamente se habrá realizado el diseño conceptual.

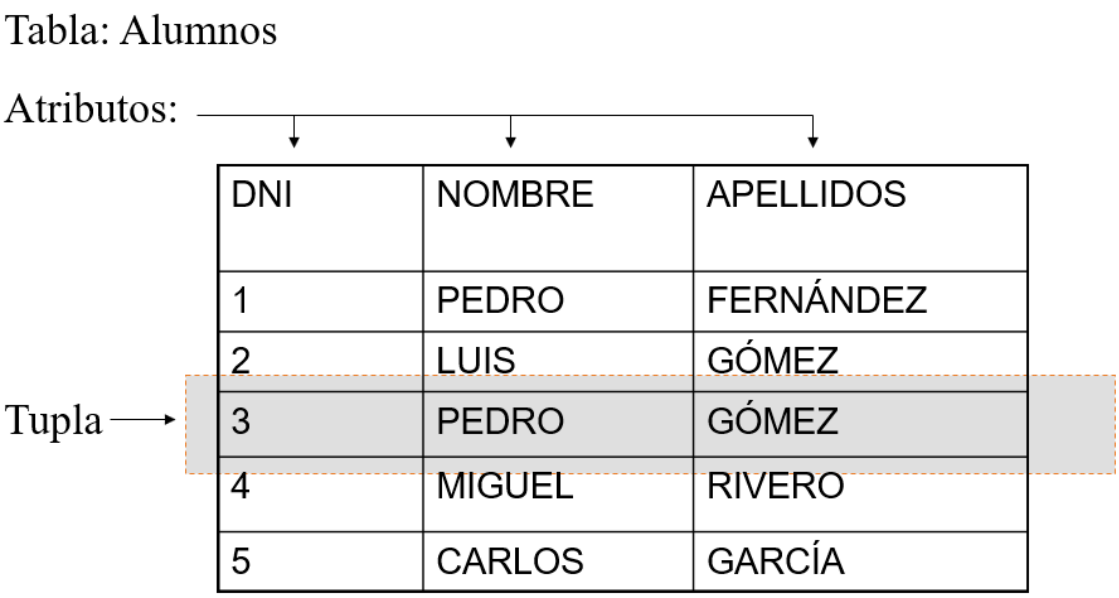
El modelo relacional se basa en los subconjuntos del producto cartesiano resultante de dos o más conjuntos.

El producto cartesiano nos da la relación de todos los elementos de un conjunto con todos los elementos de otros conjuntos de ese producto:



2.1. Relación o tabla. Tuplas. Dominios

Una relación es una tabla con filas (tuplas o registros) y columnas (atributos).



- **Atributo:** Nombre de cada dato que se almacena en la relación. En la imagen de arriba hay 3 atributos: DNI, nombre y apellidos. Debe describir el significado de la información que representa y a veces debe añadir una pequeña descripción para aclarar más el contenido.
- **Tuplas (registros):** Cada elemento de la relación o tabla. Deben cumplir dos requisitos:
  - Debe corresponderse con un elemento del mundo real.
  - No puede ver dos tuplas con todos los valores iguales.

Un atributo tiene asociado un **dominio** de valores, que son valores pertenecientes a un conjunto previamente establecido. Por ejemplo, en el atributo "Sexo", solo se podrá definir un dominiuo de valores posibles que sean "M" o "F".

Los dominios deben ser atómicos. Los valores contenidos en un atributo no pueden separarse en valores de dominios más simples.

Un dominio debe tener **nombre, definición lógica, tipo de datos y formato.**

Ejemplo:

- Nombre: Sueldo
- Def. Lógica: Sueldo neto del empleado
- Tipo de datos: Entero.
- Formato: 9.999€

2.2. Cardinalidad

- **Cardinalidad:** número de tuplas de una tabla.
- **Grupo:** tamaño de una tabla en base a sus atributos.

En la siguiente tabla se muestra el producto cartesiano de tres relaciones o tablas (A,B,C), donde A es el nombre de los alumnos, B son las asignaturas y C es si están aprobados o suspendidos.

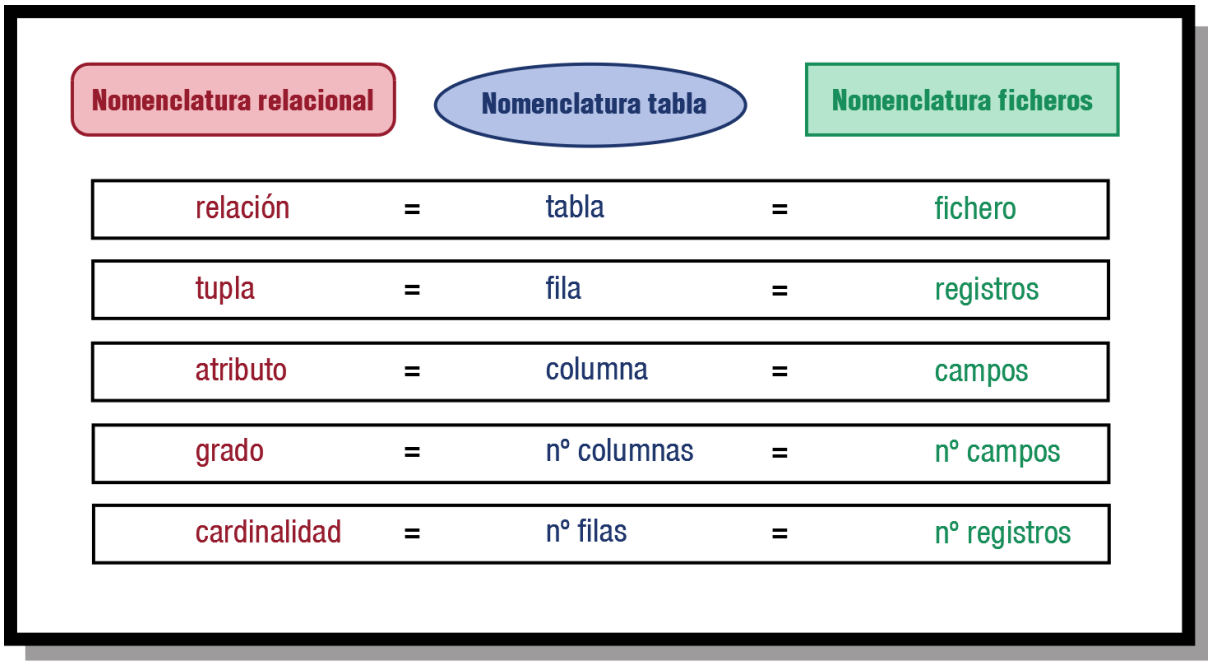
Producto Cartesiano AxBxC.

A={Carlos, María}	B={Matemáticas, Lengua}	C={Aprobado, Suspenso}
CARLOS	MATEMÁTICAS	APROBADO
CARLOS	MATEMÁTICAS	SUSPENSO
CARLOS	LENGUA	APROBADO
CARLOS	LENGUA	SUSPENSO
CARLOS	INGLÉS	APROBADO
CARLOS	INGLÉS	SUSPENSO
MARÍA	MATEMÁTICAS	APROBADO
MARÍA	MATEMÁTICAS	SUSPENSO
MARÍA	LENGUA	APROBADO
MARÍA	LENGUA	SUSPENSO
MARÍA	INGLÉS	APROBADO
MARÍA	INGLÉS	SUSPENSO

Subconjunto del Producto Cartesiano AxBxC con cardinalidad 5.

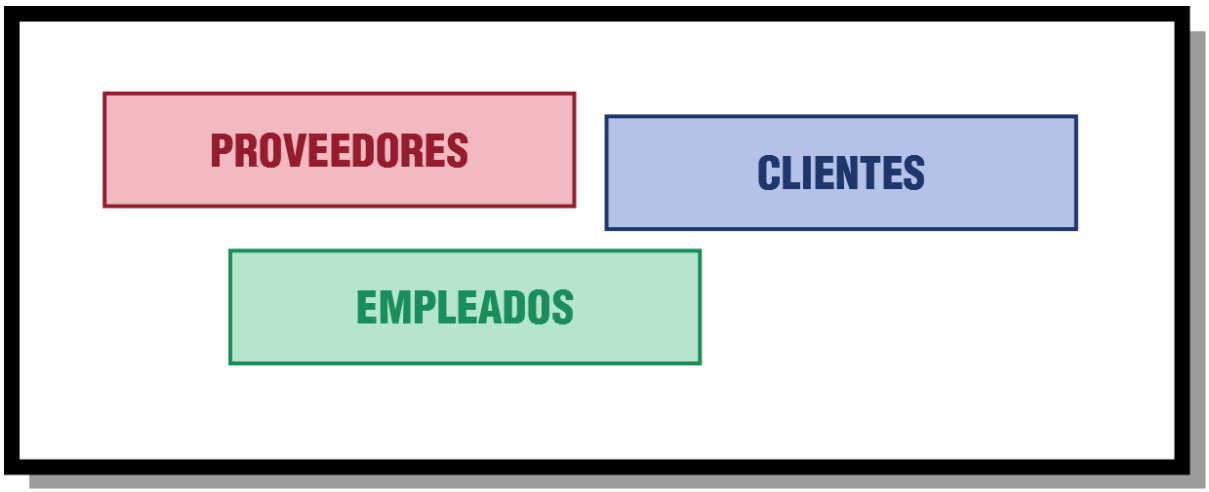
A={Carlos, María}	B={Matemáticas, Lengua}	C={Aprobado, Suspenso}
CARLOS	MATEMÁTICAS	APROBADO
CARLOS	LENGUA	APROBADO
CARLOS	INGLÉS	APROBADO
MARÍA	MATEMÁTICAS	APROBADO
MARÍA	INGLÉS	SUSPENSO

Los términos según la nomeclatura utilizada se resume en la siguiente tabla:



3. Relaciones. Características de una relación (tabla)

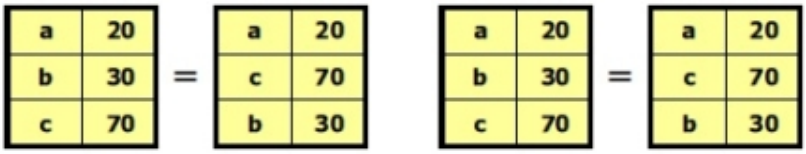
- Cada tabla o relación debe tener un nombre distinto.



- Cada atributo o columna toma un solo valor en cada tupla o fila.
- Cada atributo o columna tiene un nombre distinto en cada tabla pero puede ser el mismo en distintas.
- No puede haber dos tuplas o filas completamente iguales.

Carlos	Matemáticas	Aprobado
Carlos	Matemáticas	Suspenso
Carlos	Lengua	Aprobado
~~Carlos~~	~~Lengua~~	~~Aprobado~~

- El orden de las tuplas o filas y el de los atributos o columnas no importa.



- Todos los datos de un atributo o columna deben ser del mismo dominio. Si por ejemplo designamos que el atributo Nota solo admite valores "Aprobado" o "Suspenso" el dato "Notable" es incorrecto.

Carlos	Matemáticas	Aprobado
Carlos	Lengua	Suspenso
Carlos	Inglés	~~NOTABLE~~
María	Matemáticas	Suspenso
María	Lengua	Suspenso

3.1. Tipos de relaciones (tablas)

Existen varios tipos de relaciones o tablas:

- **Persistentes:** Solo pueden ser borradas por los usuarios.
  - **Base:** independientes.. Se crean indicando su estructura y sus ejemplares (conjunto de tuplas o filas).
  - **Vistas:** Son tablas que solo almacenan una definición de consulta. Este resultado procede de otras tablas base o de otras vistas e instantáneas.

Cuando cambian los datos de las tablas base, también cambiarán los de la vista que los utilizan.

- **Instantáneas:** son vistas, pero sí almacenan los datos que muestran además de la consulta que las creó. Modifican su resultado cada cierto tiempo. Es una fotografía de la tabla o relación que dura un tiempo concreto.

- **Temporales:** son tablas eliminadas automáticamente por el sistema.

4. Tipos de datos

Los atributos se mueven dentro de un dominio o conjunto de valores. A ese conjunto de valores se le especifica **el tipo de dato de forma general** y el **conjunto de valores que puede tomar de forma restringida** (estos últimos se indican en la definición de la tabla si el SGBD lo permite)**.** Por ejemplo:

> En el atributo "Género" el tipo de dato será "carácter" o "texto de longitud 1". El conjunto de valores que puede tomar es "M" o "F".

Cada campo o atributo debe poseer nombre que esté relacionado con los datos que va a contener y debe tener un tipo de dato que determinará que valores puede tomar y qué operaciones se pueden realizar con ellos.

Los tipos de datos más comunes son:

- **Texto:** cadenas y números que no van a realizar operaciones.
- **Numérico.**
- **Fecha.**
- **Sí/No:** almacena datos que solo tienen posibilidad de verdadero/falso.
- **Autonumérico:** numéricos secuenciales automáticos en cada registro nuevo.
- **Memo** (texto largo).
- **Moneda.**
- **Objeto OLE:** gráficos, imágenes o textos creados por otras apps.

Para determinar el tipo de dato de cada atributo se tiene en cuenta el conjunto de valores y las operaciones que se van a realizar.

5. Claves

Es un atributo o un conjunto de atributos que identifiquen de modo único a cada tupla o fila de la relación o tabla. A ese conjunto se le llama **superclave.**

Una tupla completa, al no poderse repetir se puede considerar una superclave.

En una tabla usuario podríamos utilizar distintas superclaves:

- {Nombre, Apellidos, Login, E-Mail, F_nacimiento}
- {Nombre, Apellidos, Login, E-mail]
- {Login, E-Mail}
- {Login}

5.1. Clave candidata, clave primaria, clave alternativa

Se debe elegir la clave que mejor se adapta a tus necesidades.

- **Clave candidata:** es el atributo o conjunto de atributos que se identifican de manera única en cada tupla. Una tabla debe tener al menos una clave candidata. E-Mail podría ser clave candidata, también podría serlo el conjunto de los atributos Nombre, Apellidos y F_nacimiento. Las claves candidatas deben cumplir los siguientes requisitos:
  - **Unicidad:** No puede haber dos tuplas con los mismos valores para esos atributos.
  - **Irreductibilidad:** si se elimina alguno de los atributos deja de ser única.

  La manera de identificar las claves candidatas es conociendo el significado real de los atributos o campos De esa manera se pueden desechar claves como candidatas fijándonos en valores que podemos llegar a tener. Por ejemplo, sabemos que Nombre + Apellidos no pueden ser clave candidata porque se pueden repetir.
  - **Clave primaria:** es aquella que se escoge para identificar sus tuplas de modo único. Siempre hay una clave candidata ya que las tuplas no pueden repetirse y por tanto siempre hay clave primaria. Normalmente son un pequeño subconjunto de los atributos de la tupla. También podemos crear un campo único que identifique las tuplas (por ejemplo código de usuario) que pueden estar constituidos por valores autonuméricos.
  - **Claves alternativas.** Son las claves candidatas que no son escogidas como clave primaria. Por ejemplo, si tenemos login como clave primaria, Nombre, Apellidos y F_nacimiento pueden ser claves alternativas

5.3. Clave externa, ajena o secundaria.

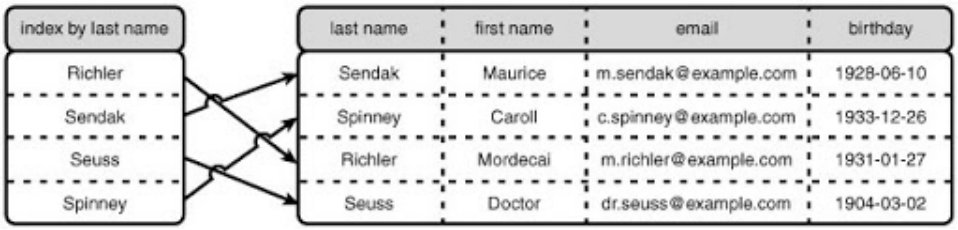
Una clave ajena, externa, foránea o secundaria es un atributo o conjunto de ellos cuyos valores coinciden con los valores de la clave primaria de alguna otra relación o de sí misma. Representan relaciones entre datos.

Las claves ajenas pueden repetirse en la tabla y tienen como objetivo establecer una conexión con la clave primaria que referencian.

Los valores de una clave ajena deben estar presentes como clave primaria en la tabla a la que hacen referencia, o bien deben ser valores nulos. De lo contrario supondría información inconsistente (no fiable).

6. Índices. Características

Un índice es una estructura de datos que permite acceder a diferentes filas de una tabla a través de uno o varios campos definidos como tales, permitiendo acceso mucho más rápido ya que se busca a partir de ellos. Ejemplo:



Los índices son independientes, lígca y físicamente de los datos. Se crean y eliminan sin afectar a las tablas o a otros índices.

Los cambios en los datos de las tablas (agregar, actualizar o borrar) son incorporados de manera automática en los índices con total transparencia. El inconveniente es que estas operaciones se ralentizan al tener que modificar tanto la tabla como el índice. A cambio, se gana velocidad en el acceso a datos.

No hay límite de columnas a indexar, sin embargo debemos elegir columnas en las que vamos a realizar operaciones de búsqueda grandes.

El SGBD utiliza índices para gestión de claves ajenas y primarias y en el caso de claves primaria serán índices únicos que no admiten valores repetidos.

7. El valor null. Operaciones con este valor

Null significa ausencia de dato. Todos los campos pueden tomar este valor independientemente del dominio al que pertenezca.

En claves secundarias, un valor nulo indica que esa tupla no está relacionada con otra.

Un valor nulo NO ES un espacio en blanco (insertar un espacio en un valor perteneciente al dominio texto es distinto a "ausencia de valor) y tampoco será lo mismo que el valor CERO.

Es imprescindible saber como actúa un NULL en operaciones lógicas. Si ambos campos son nulos no podremos obtener ni verdadero ni falso.

- Verdadero AND nulo = Nulo.
- Falso AND nulo = Falso.
- Verdadero OR nulo = Verdadero.
- Falso OR Nulo = Falso.
- Not nulo = Nulo.
- IS NULL devuelve verdadero si el dato que se compara es falso.

8. Vistas

Una vista es una tabla virtual cuyas filas y columnas se obtienen a partir de una o varias tablas de nuestro modelo. No se almacena la tabla en sí, sino su definición, actuando como filtro de las tablas a las que referencia.

La consulta que define la vista puede provenir de una o varias tablas o incluso de otras vistas de la base de datos actual u otras bases de datos.

No hay restricciones para crear vistas pero sí para modificar datos de manera que estas restricciones establecen integridad y consistencia de datos.

Las vistas se crean por dos razones:

- **Seguridad.** Los usuarios solo tienen acceso a parte de la información.
- **Comodidad:** en lenguaje de bbdd trabajar con vistas es menos complejo.

Las vistas no tienen copia física, de manera que los cambios en las tablas afectan a la vista y viceversa, pero no siempre se pueden actualizar datos de una vista. Dependerá de su complejidad y del SGBD.

9. Usuarios. Roles. Privilegios.

Un usuario es un conjunto de permisos que se aplican a una conexión de base de datos. Puede tener además otras características:

- Es el propietario de ciertos objetos (tablas, vistas, etc.).
- Realiza copias de seguridad.
- Tiene asignada una cuota de almacenamiento.
- Tiene asignado un tablespace (una lógica de almacenamiento) por defecto para los objetos Oracle.

Los privilegios son un permiso dado a un usuario para realizar operaciones que pueden ser de dos tipos:

- **De sistema:** necesitará el permiso de sistema correspondiente.
- **Sobre objeto:** necesitará el permiso sobre el objeto en cuestión.

Un rol de BBDD es una agrupación de permisos de sistemas y de objeto, de manera que si X usuarios tienen el mismo rol, cuando se añaden o quitan permisos se hace a todos los usuarios a la vez.

10. SQL

Structured Query Language es un lenguaje de dominio diseñado para administrar y recuperar información de SGBD relacionales. Permite trabajar con cualquier lenguaje y con cualquier BBDD.

Es recomendable revisar la documentación del SGBD con el que se está trabajando para conocer su sintaxis concreta, ya que no es idéntico en todas.

SQL es potente y versátil y fácil de aprender, con un lenguaje bastante natural. Se considera un lenguaje de cuarta generación.

SQL no es solo un lenguaje de consulta ya que además tiene las siguientes capacidades:

- Definición de la estructura de los datos (DDL).
- Manipulación de datos (DML).
- Especificación de conexiones seguras (DCL).

Se puede trabajar de dos formas con SQL:

- **SQL embebido:** sentencias dentro de un programa escrito en otro lenguaje.
- **SQL interpretado:** entorno gráfico para escribir y ejecutar sentencias o programa en línea de comandos para ejecutar comandos SQL.

10.1. Elementos del lenguaje. Normas de escritura

SQL está compuesto por elementos que se combinan en instrucciones para crear, actualizar y manipular bases de datos. Estos elementos se dividen en:

- **Comandos:** son las instrucciones SQL que se dividen en tres grupos:
  - **De definición de datos (DDL).** Permiten crear datos, tablas, campos, etc.
  - **De manipulación de datos (DML).** Permiten generar consultas para ordenar, filtrar y extraer datos.

- **De control de seguridad.** Administran derechos y restricciones a usuarios.

Estos comandos a su vez están construidos por:

- **Cláusulas:** Condiciones o criterios, palabras especiales para modificar el funcionamiento de un comando. (FROM, WHERE, GROUP, ORDER, HAVING).
- **Operadores:** Aritméticos o lógicos, permiten crear expresiones complejas.
- **Funciones:** Para obtener valores complejos (por ejemplo la función promedio para obtener la media de un salario).
- **Literales:** o constantes, son valores concretos, un número, fecha, conjunto de caracteres...

11. Lenguaje de descripción de datos (DDL).

Se utilizan las sentencias DDL para definir las estructuras donde almacenar información. Modifican, crean y eliminan objetos de la base de datos (metadatos).

En Oracle cada usuario de una base de datos tiene un esquema que tiene el mismo nombre que el usuario y sirve para almacenar objetos que posee ese usuario.

Estos objetos pueden ser: tablas, vistas, índices u otros objetos relacionados con la definición de la BBDD. Solo el propietario y los administradores tienen permiso para manipularlos, a no ser que se modifiquen privilegios para permitir acceso a otros usuarios.

Las instrucciones DDL no se pueden deshacer. Conviene usarlas con precaución y tener copias de seguridad.

11.1. Creación de bases de datos. Objetos de la base de datos

Una base de datos es un conjunto de objetos que nos servirán para gestionar los datos. Estos objetos están contenidos en esquemas y a su vez están asociados a un usuario.

La creación de la base de datos consiste en crear las tablas que la componen, pero antes se debe definir un espacio de nombres separado para cada conjunto de tablas **(esquemas o usuarios).**

Crear una base de datos implica indicar los archivos y ubicaciones y otras indicaciones técnicas y administrativas que se van a utilizar. Solo lo puede realizar el administrador.

```
CREATE DATABASE NombreDeLaBaseDeDatos;
```

En Oracle no utilizaremos esta sentencia, ya que tiene un sentido distinto a otros SGBD.

11.2. Creación de tablas

Antes de crear la tabla es conveniente recordar que solo se pueden crear si se poseen los permisos necesarios, además, se debe tener a mano la siguiente información que se obtiene de la fase de diseño:

- Nombre de la tabla.
- Nombre para cada columna.
- Tipo y tamaño de datos que vamos a almacenar en cada columna.
- Restricciones sobre los datos.
- Información adicional que necesitemos.

Además se deben cumplir ciertas reglas para los nombres de las tablas:

- Los nombres de las tablas no se pueden duplicar en un mismo esquema.
- Deben comenzar por carácter alfanumérico.
- Longitud máxima de 30 caracteres.
- Solo letras del alfabeto inglés, dígitos o guión bajo.
- No se pueden usar palabras reservadas de SQL.
- No distingue mayúsculas y minúsculas.
- En caso de espacios se entrecomilla. No todas las BBDD lo permiten. En Oracle las comillas se utilizan para hacer sensible el nombre de la tabla a mayús/minus.

CREATE TABLE NombreDeLaTabla (NombreColumna1 Tipo_Dato, NombreColumna2 Tipo_Dato, ... NombreColumnaN Tipo_Dato);

Ejemplo:

```
CREATE TABLE USUARIOS (Nombre VARCHAR(25));
```

11.3. Restricciones

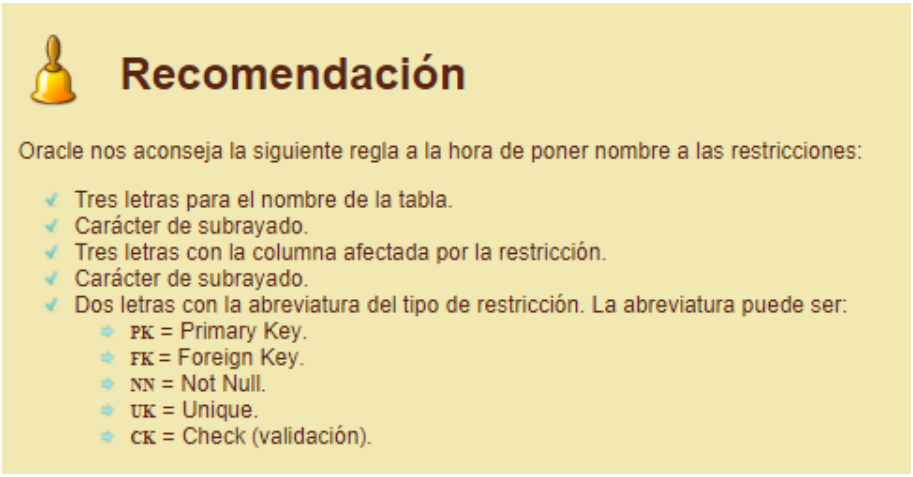
Una restricción es una condición que una o varias columnas deben cumplir obligatoriamente. Cada restricción debe llevar un nombre. Si no lo ponemos, lo hará el SGBD, pero es conveniente utilizar un nombre único para cada esquema que nos ayude a identificarla.

```
CREATE TABLE NombreDeLaTabla (
  Columna1 Tipo_Dato
    [CONSTRAINT NombreDeRestricción]
    [NOT NULL]
    [UNIQUE]
    [PRIMARY KEY]
    [FOREIGN KEY]
    [DEFAULT valor]
    [REFERENCES NombreTabla [(columna[, columna])]]
    [ON DELETE CASCADE]]
    [CHECK condición],...);
```

Los corchetes indican opcionalidad. Ejemplo:

```
CREATE TABLE USUARIOS (
Login VARCHAR(15) CONSTRAINT usu_log_PK PRIMARY KEY,
pPassword VARCHAR (8) NOT NULL,
Fecha_Ingreso DATE DEFAULT SYSDATE);
```

También se pueden definir las columnas de la tabla y después especificar las restricciones y así referir varias columnas a una única restricción.



11.3.1. Restricción NOT NULL

Prohibe valores nulos para la columna.

```
CREATE TABLE USUARIOS (
  F_Nacimiento DATE
  CONSTRAINT Usu_Fnac_NN NOT NULL);
```

O bien:

```
CREATE TABLE USUARIOS (
  F_Nacimiento DATE NOT NULL);
```

Se debe tener cuidado con los valores NULL en operaciones (1*NULL = NULL).

11.3.2. Restricción UNIQUE

Hace que no se puedan repetir valores en una misma columna. Oracle crea un índice cuando se habilita esta restricción y lo borra si se deshabilita.

```
CREATE TABLE USUARIOS (
  Login VARCHAR2 (25)
  CONSTRAINT Usu_Log_UK UNIQUE);
```

O bien

```
CREATE TABLE USUARIOS (
  Login VARCHAR (25) UNIQUE);
```

Y para poner esta restricción a varios campos:

```
CREATE TABLE USUARIOS (
  Login VARCHAR2 (25),
  Correo VARCHAR2 (25),
  CONSTRAINT Usuario_UK UNIQUE (Login, Correo));
```

Usando restricciones para varios campos hay que poner "coma" después del último campo definido, ya que la sentencia de la restricción es independiente, llevando los campos que se quieren restringir después entre paréntesis.

11.3.3. Restricción PRIMARY KEY

Se debe indicar en una tabla que columna o columnas son clave primaria. Solo puede haber una y podrá ser referenciada como clave ajena en otras tablas.

Los campos que forman la primary key son **NOT NULL** y de tipo **UNIQUE.**

Clave de único campo:

```
CREATE TABLE USUARIOS (
  Login VARCHAR2 (25) PRIMARY KEY);
```

o bien:

```
CREATE TABLE USUARIOS (
  Login VARCHAR2 (25)
  CONSTRAINT Usu_Log_PK PRIMARY KEY);
```

Clave formada por más de un campo:

```
CREATE TABLE USUARIOS (
  Nombre VARCHAR2 (25),
  aPELLIDOS (VARCHAR2 (30),
  F_Nacimiento DATE,
  CONSTRAINT Usu_PK PRIMARY KEY (Nombre, Apellidos, F_Nacimiento));
```

Nunca olvidar la "coma" después del último campo.

11.3.4. Restricción REFERENCES. FOREIGN KEY

Cuando creamos la tabla debemos indicar la clave ajena haciendo referencia a la tabla y los campos de donde procede. Por ejemplo para unir una tabla llamada Partidas a través del campo Cod_Partida, crearemos en nuestra tabla usuarios un campo Cod_Partida que haga referencia a la tabla y campo que se quiere referenciar:

```
CREATE TABLE USUARIOS (
  Cod_Partida NUMBER(8)
    CONSTRAINT Cod_part_FK
    REFERENCES PARTIDAS(Cod_Partida));
```

Si el campo al que se hace referencia es clave principal no es necesario indicar el nombre del campo.

```
CREATE TABLE USUARIOS (
  Cod_Partida NUMBER(8)
    CONSTRAINT Cod_part_FK
    REFERENCES PARTIDAS);
```

Poniendo la definición de clave al final debemos colocar el texto FOREIGN KEY. En el caso de que la clave ajena estuviese formada por dos campos se haría de la siguiente manera:

```
CREATE TABLE USUARIOS (
Cod_Partida NUMBER(8),
F_Partida DATE,
CONSTRAINT Partida_Cod_F_FK FOREIGN KEY (Cod_Partida, F_Partida)
REFERENCES PARTIDAS);
```

No olvidar la coma después del último campo.

Llamamos **integridad referencial** al hecho de tener la clave secundaria de la tabla incluida en su tabla de procedencia, donde es clave primaria o candidata. Cualquier dato que incluyamos en la clave ajena debe estar previamente en la tabla de la que procede. Esto puede generar algunos errores:

- Si hacemos referencia a una tabla no creada, Oracle la buscará y dará fallo. Se soluciona creando primero las tablas que no tienen claves ajenas.
- Si queremos borrar las tablas primero debemos borrar las que tengan claves ajenas.

Pero existen otras soluciones tras la cláusula REFERENCE:

- **ON DELETE CASCADE:** Borra todos los registros cuya clave ajena sea igual a la clave del registro borrado.
- **ON DELETE SET NULL:** Coloca NULL en claves ajenas relacionadas con la borrada.
- **ON DELETE DEFAULT xxxx:** coloca el valor xxxx en todas las claves ajenas relacionadas con la borrada.

Estas opciones valen también en **ON UPDATE.**

11.3.5. Restricción DEFAULT y CHECK.

Default permite insertar un valor por defecto en un campo:

```
CREATE TABLE USUARIOS (
  Pais VARCHAR(20 DEFAULT 'España');
```

Se puede añadir distintas expresiones: constantes, funciones SQL y variables:

```
CREATE TABLE USUARIOS (
  Fecha_ingreso DATE DEFAULT SYSDATE);
```

CHECK comprueba que los valores que se introducen son los adecuados.

Esta condición se puede construir con columnas de la tabla:

```
CREATE TABLE USUARIOS(
  Credito NUMBER(4) CHECK (Crédito BETWEEN 0 AND 2000));
```

Una misma columna puede tener varios check asociados. Para ello se ponen varios CONSTRAINT seguidos y separados por comas.

11.4. Eliminación de tablas.

Eliminamos tablas cuando ya no son útiles para no ocupar espacio:

```
DROP TABLE NombreDeLaTabla [CASCADE CONSTRAINTS];
```

La opción CASCADE CONSTRAINTS se incluye cuando alguna clave es ajena en otra tabla. De esta forma se borran las restricciones donde es clave ajena y después se elimina la tabla.

- Se borrará la tabla y sus datos (filas). También se borra la información de esa tabla en el diccionario de datos.
- El borrado es irreversible y no hay confirmación antes de ejecutarse.
- Las vistas de una tabla siguen existiendo pero ya no funcionan.
- Oracle tiene la orden **TRUNCATE TABLE**. Borra las filas pero no la estructura.
- Solo se permite borrar tablas con permiso de borrado.

11.5. Modificación de tablas

Se pueden modificar tablas después de crearlas, añadir, eliminar o modificar campos, añadir o eliminar restricciones...

- **Cambiar nombre de tabla:**

```
RENAME Nombreviejo TO NombreNuevo;
```

- **Añadir columnas (se añaden al final).**

```
ALTER TABLE NombreTabla ADD
(ColumnaNueva1 Tipo_Datos [Propiedades]
[, ColumnaNueva2 Tipo_Datos [Propiedades]
...);
```

- **Eliminar columnas.** No se puede deshacer la acción, se elimina la definición y sus datos, no se puede eliminar una columna que es la única que forma la tabla:

```
ALTER TABLE NombreTabla DROP COLUMN (Columna1 [, Columna2, ...]);
```

- **Modificar columnas.** Los cambios son posibles si no contiene datos. Si los tiene, solo podremos aumentar la longitud de la columna, aumentar o disminuir nº de

posiciones decimales en tipo NUMBER o reducir anchura siempre que los datos no ocupen todo el espacio reservado.

```
ALTER TABLE NombreTabla MODIFY
(Columna1 TipoDatos [propiedades]
[, Columna2 TipoDatos [Propieades]...]);
```

- **Renombrar columnas.**

```
ALTER TABLE NombreTabla RENAME COLUMN NombreAntiguo TO NombreNuevo;
```

- **Borrar restricciones.**

```
ALTER TABLE NombreTabla DROP CONSTRAINT NombreRestriccion;
```

- **Modificar nombre de restricciones.**

```
ALTER TABLE NombreTabla RENAME CONSTRAIT NombreViejo to NombreNuevo;
```

- **Activar o desactivar restricciones.** A veces es conveniente desactivarlas temporalmente para hacer pruebas o saltarse la regla.

```
ALTER TABLE NombreTabla DISABLE CONSTRAINT NombreRestriccion [CASCADE];
```

```
ALTER TABLE NombreTabla ENABLE CONSTRAINT NombreRestriccion [CASCADE];
```

La opción CASCADE desactiva/activa las restricciones que dependan de ésta.

Para consultar las restricciones podemos consultar el diccionario de datos all_constants.

11.6. Creación y eliminación de índices.

```
CREATE INDEX NombreIndice ON NombreTabla (Columna1 [, Columna2...]);
```

No se aconseja utilizar índices sobre campos de tablas pequeñas o que se actualicen con frecuencia. Tampoco si esos campos no se usan en consultas de manera frecuente o en expresiones.

Una mala elección ocasiona ineficiencia y un uso excesivo puede dejar la BBDD colgada solo por insertar una fila.

```
DROP INDEX NombreIndice;
```

La mayoría de ínidces se crean implícitamente en restricciones PRIMARY KEY, FOREIGN KEY O UNIQUE.

12. Lenguaje de control de datos.

Se necesitan cuentas de usuario para acceder a los datos de una BBD, estas claves de acceso se establecen cuando se crea el usuario y las modifica el administrador o propietario de la clave. Se almacenan encriptadas en DBA_USERS. Para crear usuarios se debe contar con privilegios de administrador.

Así se crean usuarios:

```
CREATE USER NombreUSuario
IDENTIFIED BY ClaveAcceso
[DEFAULT TABLESPACE tablespace]
[TEMPORARY TABLESPACE tablespace]
[QUOTA int {K | M} ON tablespace]
[QUOTA UNLIMITED ON tablespace]
[PROFILE perfil];
```

- **CREATE USER:** Nombre de usuario
- **IDENTIFIED BY:** Contraseña.
- **DEFAULT TABLESPACE:** Asigna al usuario el nombre del tablespace por defecto para almacenar los objetos que cree. Si no se asigna una, será SYSTEM.
- **TEMPORARY TABLESPACE:** Especifica el nombre del Tablespace para trabajos temporales. Por defecto SYSTEM.
- **QUOTA:** Asigna un espacio en Megabytes o Kilobytes en el Tablespace asignado. Si no se especifica el usuario no tendrá espacio y no podrá crear objetos.
- **PROFILE:** Asigna un perfil al usuario, si no se especifica será el perfil por defecto.

Para ver todos los usuarios creados utilizamos las vistas **ALL_USERS y DBA_USERS**. Y para ver en mi sesión los usuarios que existen pondría: **DESC SYS.ALL_USERS;**

Para modificar usuarios se utiliza:

```
ALTER USER NombreUsuario
IDENTIFIED BY clave_acceso
[DEFAULT TABLESPACE tablespace ]
[TEMPORARY TABLESPACE tablespace]
[QUOTA int {K | M} ON tablespace]
[QUOTA UNLIMITED ON tablespace]
[PROFILE perfil];
```

Un usuario sin privilegios de administrador solo podrá modificar su propia contraseña.

```
DROP USER NombreUsuario [CASCADE];
```

La opción CASCADE borra todos los objetos del usuario antes de eliminarlo. Sin esa opción no deja borrar al usuario si tiene tablas creadas.

12.1. Dar permisos

Los usuarios no pueden llevar a cabo operaciones si no tienen permisos para ello. Para acceder a objetos de una BBDD se necesitan privilegios que se pueden agrupar formando **roles.**

Estos roles pueden gestionar los comandos que pueden utilizar los usuarios y pueden activarse, desactivarse o protegerse con clave.

Para dar privilegios sobre objetos ya existentes:

```
GRANT {privilegio_objeto [, privlegio_objeto]... |ALL|[PRIVILEGES]}
ON [usuario.]objeto
FROM {usuario1|rol|PUBLIC} [,{usuario2|rol2|PUBLIC]...
[[WITH GRANT OPTION];
```

- **ON** especifica el objeto sobre el que se conceden privilegios.
- **TO** Señala a los usuarios o roles a los que se conceden privilegios.
- **ALL** concede todos los privilegios sobre el objeto especificado.
- **WITH GRANT OPTION** permite que el receptor del privilegio se lo pueda conceder a otros.
- **PUBLIC** hace que un privilegio esté disponible para todos los usuarios.

Por ejemplo:

```
GRANT INSERT ON usuarios TO Ana; /*concede permisos a Ana para insertar
datos en la tabla Usuarios.*/
```

```
GRANT ALL ON Partidas TO Ana; /*concede todos los privilegios sobre la
tabla Partidas a Ana.*/
```

Para dar privilegios de sistema (estos dan derecho a ejecutar comandos SQL o acciones sobre objetos). La sintaxis es así:

```
GRANT {Privilegio1 | rol1 } [, privilegio2 | rol2}, ...]
TO {usuario1 | rol1| PUBLIC} [, usuario2 | rol2 | PUBLIC} ... ]
[WITH ADMIN OPTION];
```

- **TO** señala usuarios o roles a los que se le conceden privilegios.
- **WITH ADMIN OPTION** permite al receptor que pueda conceder esos mismos privilegios a otros usuarios o roles.
- **PUBLIC** hace que un privilegio esté disponible para todos los usuarios.

Por ejemplo:

```
GRANT CONNECT TO Ana; /*Concede a Ana el rol Connect con todos los privilegios
que tiene asociados.*/
GRANT DROP USER TO Ana WITH ADMIN OPTION; /*Concede a Ana el priv. de borrar
usuarios y además puede conceder el mismo privilegio a otros usuarios.*/
```

12.2. Retirar permisos

Con el comando REVOKE se retiran privilegios.

Sobre objetos.

```
REVOKE {privilegio_objeto [, privilegio_objeto]...|ALL|[PRIVILEGES]}
ON [usuario.]objeto
FROM {usuario|rol|PUBLIC} [,{usuario|rol|PUBLIC] ...;
```

Por ejemplo:

```
REVOKE SELECT, UPDATE ON Usuarios FROM Ana; /*Quitar permiso a Ana de seleccionar y actualizar
la tabla de Usuarios*/
```

Del sistema o roles a usuarios.

```
REVOKE {privilegio_stma | rol} [, {privilegio_stma | rol}]...|ALL|[PRIVILEGES]}
ON [usuario.]objeto
FROM {usuario|rol|PUBLIC} [,{usuario|rol|PUBLIC] ...;
```

Por ejemplo:

```
REVOKE DROP USER FROM Ana; /* Quitamos permiso a Ana para
eliminar usuarios*/
```

Anexo. Elementos del lenguaje SQL

Los elementos SQL se combinan en las instrucciones que se utilizan para crear, actualizar y manipular BBDD.

Comandos

Comandos DDL. Lenguaje de definición de datos:

- **CREATE** (crear nuevas tablas campos e índices).
- **DROP** (eliminar tablas e índices).
- **ALTER** (modificar tablas).

Comandos DML. Lenguaje de manipulación de datos:

- **SELECT** (consultar filas con un criterio determinado).
- **INSERT** (insertar filas en una tabla).
- **UPDATE** (modificar valores de campos y filas específicos).
- **DELETE** (eliminar filas de una tabla).

Comandos DCL. Lenguaje de control de datos.

- **GRANT** (Dar permisos a uno o varios usuarios o roles).
- **REVOKE** (quitar permisos que previamente se han concedido).

Cláusulas

Llamadas también condiciones o criterios. Permiten modificar el funcionamiento de un comando.

- **FROM** (especifica la tabla de la que se seleccionarán las filas).
- **WHERE** (con él se especifican las condiciones que deben reunir las filas que se van a seleccionar).
- **GROUP BY** (se usa para separar las filas en grupos específicos).

- **HAVING** (se usa para expresar la condición que debe tener un grupo).
- **ORDER BY** (Ordena las filas seleccionadas en un orden específico).

Operadores

Permiten crear expresiones complejas

Operadores Lógicos

- **AND** (Evalúa dos condiciones y devuelve un valor de verdad si ambas son ciertas).
- **OR** (Evalúa dos condiciones y devuelve un valor de verdad si alguna de las dos es cierta).
- **NOT** (Devuelve el valor contrario de la expresión).

Operadores de comparación

- **<** (Menor que)
- **>** (Mayor que)
- **<>** (Distinto que)
- **<=** (Menor o igual que)
- **>=** (Mayor o igual que)
- **=** (Igual)
- **BETWEEN** (para especificar un intervalo de valores).
- **LIKE** (para comparar)
- **IN** (para especificar filas de una base de datos).

Funciones

Para conseguir valores complejos. Existen muchas, estos son solo algunos ejemplos.

Funciones de agregado

- **AVG** (Calcula el promedio de los valores de un campo determinado).
- **COUNT** (devuelve el nº de filas de la selección)
- **SUM** (devuelve la suma de los valores de un campo determinado).
- **MAX** (devuelve el valor más alto).
- **MIN** (devuelve el valor más bajo).

Funciones literales

También llamadas constantes. Son valores concretos: un número, una fecha, un conjunto de caracteres...

- **23/03/97** (literal fecha).
- **María** (literal caracteres).
- **5** (literal número).

Mapa Conceptual