# Base de Datos I

## NORMALIZACIÓN PARA BASES DE DATOS RELACIONALES

INTRODUCCIÓN:

Cada relación consiste en varios atributos. Existen normas "formales" para determinar que una agrupación de atributos en una tabla sea mejor que otra.
Como pautas informales de diseño encontramos:

- La relación que existe entre los valores de los atributos de una tupla.
- Evitar los valores nulos en las tuplas.

- **Significado de los atributos en una relación**

Se definen ciertas reglas para optimizar las relaciones:
1. Si la entidad A contiene tuplas que se relacionan de a una con más de una tupla de la entidad B (o sea una relación de uno a muchos), la relación correspondiente a "muchos" (en este caso la entidad B) contendrá un atributo que la relacione con la otra.
Ej. Tenemos una tabla-relación que contiene datos de las provincias disponibles para residir en una convención laboral. Por otro lado, la tabla que contiene datos de los becados que concurrirán a dicha convención.
Esto me indica la relación:

| | | |
|---|---|---|
| A una provincia | concurren | varios becados |
| Un becado | puede concurrir a | una provincia |

Entidad A: Provincias

| Cod-Prov | Descripción |
|---|---|
| 1 | La Pampa |
| 2 | Corrientes |
| 3 | Neuquén |
| 4 | La Rioja |

Entidad B: Becados

| ID | Nombre | Cod-Prov |
|---|---|---|
| 1 | Farina, Ana | 1 |
| 2 | Perez, Ariel | 4 |
| 3 | Gómez, Luis | 2 |
| 4 | Valdez, María | 1 |

No hizo falta la creación de una tercera tabla, ya que la vinculación se produce cuando se agrega en la entidad B el atributo COD-PROV (el cual es clave ajena referenciando a la entidad A).

2. Si la entidad A contiene más de una tupla que se relaciona con tuplas de la entidad B de a una (o sea una relación de muchos a uno), sucede lo mismo que en el caso anterior, en la relación de "muchos" (en este caso la entidad A) contendrá un atributo que la relacione con la otra.

Ej. Tenemos una tabla-relación que contiene datos de los empleados de una empresa. Por otro lado, tenemos la tabla que contiene los datos de las gerencias de esa empresa

Esto me indica la relación:

| | | |
|---|---|---|
| Un empleado | pertenece a | una gerencia |
| A una gerencia | pueden pertenecer | varios empleados |

Entidad A: Empleados

| Legajo | Nombre | IdGerencia |
|---|---|---|
| 1010 | Torres, José | 01 |
| 1012 | Perez, Sergio | 03 |
| 1015 | Funes, Diego | 02 |
| 1020 | López, Alan | 01 |

Entidad B: Gerencias

| IdGerencia | Nombre |
|---|---|
| 01 | Sistemas |
| 02 | Contaduría |
| 03 | Finanzas |
| 04 | Recursos Humanos |

En este caso tampoco hizo falta la creación de una tercera tabla, ya que la vinculación se produce cuando se agrega en la entidad A el atributo IDGERENCIA (el cual es clave ajena referenciando a la entidad B)

3. Si la entidad A contiene más de una tupla que se relaciona con más de una tupla de la entidad B (o sea una relación de muchos a muchos), se justifica la creación de una tercera tabla que interrelacione las otras 2.

Ej. Contamos con la entidad A: PACIENTES, que contiene más de una tupla que se relaciona con más de una tupla de la entidad B: MEDICAMENTOS (o sea una relación de muchos a muchos), se justifica la creación de una tercera tabla PAC-MED que interrelacione las otras 2.

Esto me indica la relación:

Un paciente puede consumir varios medicamentos
Un medicamento puede ser consumido por varios pacientes

Entidad A: Pacientes

| Id-Pac | Nombre | Edad |
|---|---|---|
| 21010 | Torres, José | 51 |
| 21012 | Alonso, Sergio | 46 |
| 31015 | Funes, Diego | 34 |
| 51020 | López, Alan | 18 |

Entidad B: Medicamentos

| Id-Med | Descripción |
|---|---|
| 01 | Calmante |
| 02 | Colirio |
| 03 | Cicatrizante |
| 04 | Anticoagulante |

Interrelación C: PAC-MED

| Id-Pac | Id-Med | Dosis-Diaria |
|---|---|---|
| 21010 | 01 | 2 |
| 21010 | 03 | 1 |
| 21012 | 04 | 2 |
| 51020 | 02 | 3 |

- **Valores nulos en las tuplas**

En una relación se agrupan determinada cantidad de atributos que a veces conforman una relación demasiado "grande". Si hubiera atributos que no se aplican a la mayoría de las tuplas de esta relación, aparecerán gran número de nulos. Por eso, se trata de evitar incluir en una relación, atributos cuyos valores puedan ser nulos, y de existir, que sea en su minoría.

Ej: Si sólo el 10% de los alumnos trabajan, no se justifica incluir un atributo CUIL o CUIT. Se puede crear una relación ALU-TRAB que contenga sólo tuplas con los alumnos que están empleados.

ALU-TRAB(LEGAJO, CUIT-CUIL)

## FORMAS NORMALES

Edgard Codd propuso 3 formas normales, las cuales se conocen como primera (1FN), segunda (2FN) y tercera (3FN) formas normales. Todas estas formas se definen bajo las restricciones de las dependencias funcionales entre los atributos de una relación.

La **normalización** de los datos consiste en descomponer las relaciones distribuyendo sus atributos en relaciones más pequeñas satisfaciendo un cierto conjunto de restricciones. Es el proceso de producción de grupos óptimos de atributos en las relaciones.

Ventajas de llevar las relaciones hasta la tercera forma normal:

- Se evitan anomalías en la inserción, borrado y modificación.
- Se facilita la extensión: si en un futuro se llevan a cabo ampliaciones, se tendrán menos cambios en la estructura de la base de datos.

## Primera Forma Normal (1fn)

Una relación está en 1FN si y sólo si todos sus dominios subyacentes contienen sólo valores atómicos. Significa la eliminación de grupos repetitivos.

Ej. EMPLEADOS

| Legajo | Nombre | Cod-Idioma | Nombre-Idioma | Nivel-Idioma | Sección |
|---|---|---|---|---|---|
| 120 | Juan | 01 | Inglés | B | Sistemas |
| 121 | José | 03 | Portugués | A | Contaduría |
| 121 | José | 04 | Italiano | B | Contaduría |
| 122 | Ana | 01 | Inglés | C | Ventas |

Los atributos referidos al "idioma" del empleado son los que provocan la repetición de los legajos. Entonces, eliminar los grupos repetitivos significa "llevarme a otra relación los atributos causantes de tal iteración".

Así quedan dos relaciones entidades dispuestas de la siguiente manera:

EMPLEADOS (LEGAJO, NOMBRE, SECCION)
Con los atributos propios del empleado.

IDIOMAS (COD-IDIOMA, NOMBRE-IDIOMA)
Con los atributos propios del idioma.

Y una interrelación EMP-IDIOMA para indicar el NIVEL de cada empleado en cada idioma.

EMP-IDIOMAS (LEGAJO, COD-IDIOMA, NIVEL-IDIOMA)

## Segunda Forma Normal (2fn)

Una relación está en 2FN si y sólo si está en 1FN y todos los atributos no clave tienen dependencia funcional completa con la clave primaria, no existen dependencias parciales.

Ejemplo 1:

EMP-IDIOMA (LEGAJO, COD-IDIOMA, NIVEL-IDIOMA)

NIVEL-IDIOMA es un atributo no clave que tiene dependencia funcional completa con toda la clave: el nivel del idioma depende del empleado que lo tiene y del idioma que sabe. Esto significa que la relación está en 2FN.

Ejemplo 2:

EMPLEADOS (LEGAJO, COD-IDIOMA, NIVEL-IDIOMA, NIVEL-ESTUDIO)

La relación se encuentra en 1FN porque la tupla se identifica en forma única, pero no se cumple la 2 FN, ya que el atributo NIVEL-ESTUDIO depende del empleado independientemente

del idioma que sepa. Por lo tanto, ese atributo debe desaparecer de esta relación e incluirse en la relación entidad del empleado.

Las tablas correctas quedan:

EMPLEADOS ( LEGAJO, NOMBRE, SECCION, NIVEL-ESTUDIO)
Con los atributos propios del empleado.

IDIOMAS (COD-IDIOMA, NOMBRE-IDIOMA)
Con los atributos propios del idioma.

EMP-IDIOMA (LEGAJO, COD-IDIOMA, NIVEL-IDIOMA)
Con los atributos de la interrelación.

**Tercera Forma Normal (3fn)**

Una relación está en 3FN si y sólo si está en 2FN y todos los atributos no clave son mutuamente independientes entre sí.

Ejemplo:

EMPLEADOS (LEGAJO, NOMBRE, NRO-SECCION, OFIC-SECCION)

Para la 3FN, además de estar en 1FN y 2FN, ningún dominio no clave debe tener dependencia funcional completa con otro dominio no clave.

NOMBRE y SECCION no tienen ninguna dependencia.
NOMBRE y OFICINA-SECCION no tienen ninguna dependencia.
NRO-SECCION y OFIC-SECCION dependen entre sí, ya que si cambia de sección, cambia de oficina. Por este caso, la relación no se encuentra en 3FN

Se debe quitar de la tabla el atributo OFIC-SECCION, y armar otra donde no interfiera en la normalización de la tercera forma.

EMPLEADOS (LEGAJO, NOMBRE, NRO-SECCION)

SECCIONES (NRO-SECCION, OFICINA-SECCION)

Si la relación contiene un solo atributo no clave, el análisis de la 3FN sería innecesario.

Ej.: IDIOMAS (COD-IDIOMA, NOMBRE-IDIOMA)

Esta tabla está en 1FN, 2FN y 3FN respectivamente.

**EJERCICIOS**

1) Tenemos los siguientes requerimientos para una base de datos universitaria con que se manejan las boletas de notas de los estudiantes. Para cada alumno consta su nombre, número de legajo, dirección, teléfono, otro teléfono opcional, fecha de nacimiento, sexo, departamento de carrera, departamento de especialidad.
Para cada departamento, figura su respectivo código con su nombre.
Para cada curso, el nombre del curso, su código, número de horas semanales, nivel, profesor a cargo, año.
Las boletas se conforman del número de legajo, curso, nota y fecha.

A) Diseñar un conjunto de relaciones normalizadas.
B) Llevar todas las relaciones hasta 2FN.
C) Especificar los atributos clave de cada relación.

2) Se trata de hacer una versión normalizada de la estructura que describe una compra de libros a una compañía XZ.
Un pedido consiste en el nombre del cliente, la fecha de pedido, el ISBN (código internacional único) del libro pedido, el título, el autor, la cantidad que ha sido pedida, y el importe total del período para un libro determinado.

3) Tenemos el siguiente conjunto de datos que se va a grabar en una BD de personal de una compañía:

- La compañía tiene un conjunto de departamentos.
- Cada departamento tiene un conjunto de empleados, un conjunto de proyectos y un conjunto de oficinas.
- Cada empleado tiene una historia de empleos. Para cada empleo, tiene una historia de salarios.
- Cada oficina tiene un conjunto de teléfonos.

La base de datos debe contener la siguiente información:

- Para cada departamento: Nro. de departamento (único), presupuesto de cada departamento y número de empleado del gerente del departamento (único)
- Para cada empleado: Nro. de empleado (único), nro. de proyecto actual, nro. de oficina y nro. de teléfono; además el título de cada trabajo que ha tenido el empleado, junto con la fecha y el salario para cada salario distinto recibido en ese trabajo.
- Para cada proyecto: Nro. de proyecto (único) y presupuesto del proyecto.
- Para cada oficina: Nro. de oficina (único), área en m2. y números de todos los teléfonos de esa oficina.

Diseñar un conjunto apropiado de relaciones normalizadas hasta 3FN.

4) Supongamos que tenemos una relación para proveer la asignación de empleados temporarios a los proyectos:

PROY-ASIGN (NRO-EMP, TEL, SUELDO-HORA, NRO-PROY, FECHA-FINAL)

A) ¿Está en 1FN?
B) ¿Está en 2FN?
C) ¿Está en 3FN?

## PROCESO DE DISEÑO DE BASES DE DATOS

### Ciclo de vida de una Base de Datos

1) **Definición del Sistema**: En esta etapa se define el alcance del sistema de bases de datos, los usuarios que utilizarán la misma y sus aplicaciones generales y específicas.
   Durante la recolección y análisis de requerimientos, los diseñadores mantienen entrevistas con los usuarios de la base de datos para poder interpretar y documentar sus necesidades en lo que respecta a requerimientos de información.

   En paralelo con esto, se especificarán los requerimientos funcionales, o sea, las necesidades del usuario con respecto a las transacciones que se realizarán, incluyendo la obtención de datos y su actualización.

2) **Diseño**: Se confecciona un diseño completo lógico y físico del sistema de bases de datos en en sistema de gestión de bases de datos (SGBD) elegido.

3) **Implementación**: Se debe implementar la base de datos diseñada en un SGBD comercial (se elige el más apropiado). Se crean los archivos de base de datos vacíos y se implementan las aplicaciones de software.
   En paralelo con estas actividades, se diseñan e implementan programas de aplicación que van a interactuar con esta nueva base de datos.

4) **Carga o conversión de los datos**: en algunos proyectos, la conversión de la base de datos lleva mucho más trabajo (y más planeación estratégica) que el desarrollo de programas. En otros casos, puede no haber existido una base de datos y esto da lugar a la carga completa de la misma.

5) **Conversión de las aplicaciones**: luego de esta etapa se está en condiciones de efectuar la instalación.

6) **Prueba y validación**: se probará con datos reales en paralelo con el sistema existente para no provocar, en caso de fallas, problemas en la organización existente.

7) **Operación**

8) **Supervisión y mantenimiento**

**Pautas para el diseño físico**

Los factores que se deben tener en cuenta para el diseño físico son:

- Análisis de las consultas y transacciones (en relación con su calidad)
- Análisis de las consultas y transacciones (con respecto a la frecuencia esperada de invocación)
- Análisis de las consultas y transacciones (con relación a las restricciones de tiempo sobre ellas)
- Análisis de la frecuencia esperada de las operaciones de actualización.

Basándonos en dichos factores influyentes, las pautas para el diseño físico de sistemas relacionales constan de:

- Aplicación de técnicas para optimizar las operaciones hechas sobre la base de datos.
- Prestar un cuidado especial en la organización de los archivos de la base de datos y la selección de los índices.

**MODELO DE DATOS**

Es una colección de herramientas conceptuales para la descripción de datos, relaciones entre datos, semántica de los datos y restricciones de consistencia.

Veremos dos modelos de datos:

- Modelo Entidad
- Modelo Relacional

El **modeloentidad-relación** (E-R) es un modelo de datos de alto nivel. Está basado en una percepción de un mundo real que consiste en una colección de objetos básicos, denominados entidades, y de relaciones entre estos objetos.

El **modelo relacional** es un modelo de menor nivel. Usa una colección de tablas para representar tanto los datos como las relaciones entre los datos.

Los diseñadores formulan generalmente el diseño del esquema de la base de datos modelando primero los datos en alto nivel, usando el **modelo E-R**, y después traduciéndolo al **modelo relacional**.

## MODELO ENTIDAD-RELACIÓN

El modelo de datos entidad-relación (E-R) está basado en una percepción del mundo real consistente en objetos básicos llamados entidades y de relaciones entre estos objetos.

Se desarrolló para facilitar el diseño de bases de datos permitiendo la especificación de un esquema de la empresa que representa la estructura lógica completa de una base de datos.

La estructura lógica general de una base de datos se puede expresar gráficamente mediante un **diagrama E-R**. Los diagramas son simples y claros.

Se han propuesto varias metodologías de diseño por distintos autores. Analizaremos dos de dichas propuestas: el modelo entidad-relación de **CHEN** y el diagrama de entidad-relación de **MARTIN**

## ELEMENTOS DEL DIAGRAMA ENTIDAD RELACIÓN:

### Entidad:

Cualquier objeto, real o abstracto, que existe en un contexto determinado o puede llegar a existir y del cual deseamos guardar información, que con posterioridad nos interesará recuperar.

Una entidad puede ser un objeto con existencia física como: una persona, un animal, una casa, etc. (real); o un objeto con existencia abstracta como: un puesto de trabajo, una asignatura de clases, etc.

Se representan gráficamente mediante rectángulos y su nombre se muestra en el interior:

| EMPLEADOS | MATERIAS | PUESTO LABORAL |
|---|---|---|

**Atributo:**

Definen o identifican las características de la entidad. Cada entidad contiene distintos atributos, que dan información sobre esta entidad. Estos atributos pueden ser de distintos tipos (numéricos, texto, fecha). También pueden asumir distintos valores.

Se los representa gráficamente mediante elipses.


**Clasificación de los atributos:**

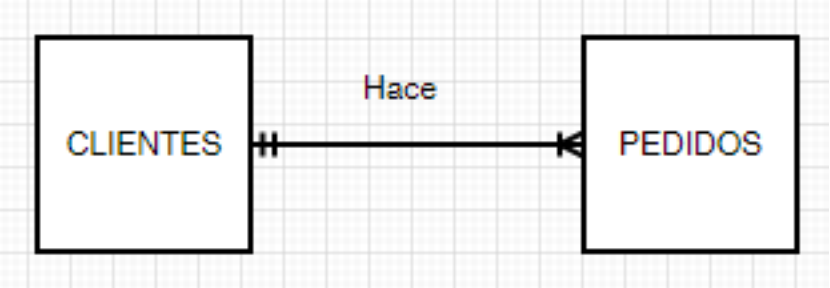
- Simples (Ejemplo: nombre-cliente)
- Compuestos (Ejemplo: dirección)
- Clave (es decir, únicas)
- Univaluadas o Multivaluadas (permiten grupos repetitivos, ejemplo: teléfono)
- Base o Derivadas (Ejemplo: cantidad-total)

**Relaciones:**

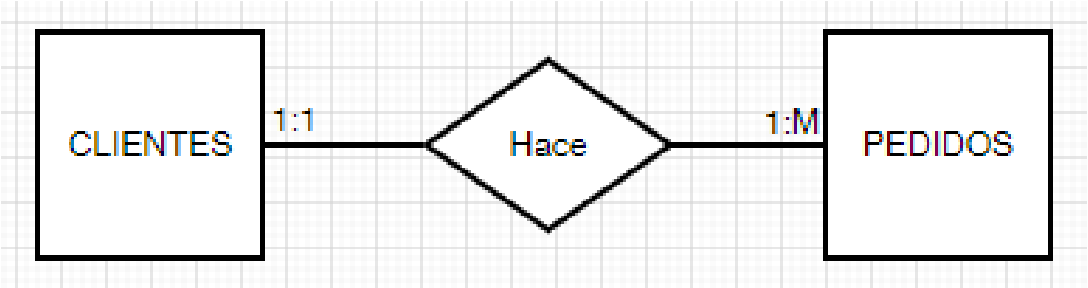
Es la correspondencia o asociación entre dos o más entidades.
Cada relación tiene un nombre que describe su función.

Las relaciones se representan gráficamente mediante rombos y su nombre se muestra en el interior:

**MARTIN:**




**CHEN:**




Las relaciones recursivas son relaciones binarias que conectan una entidad consigo misma.

Ejemplo:

Entidad: Empleados

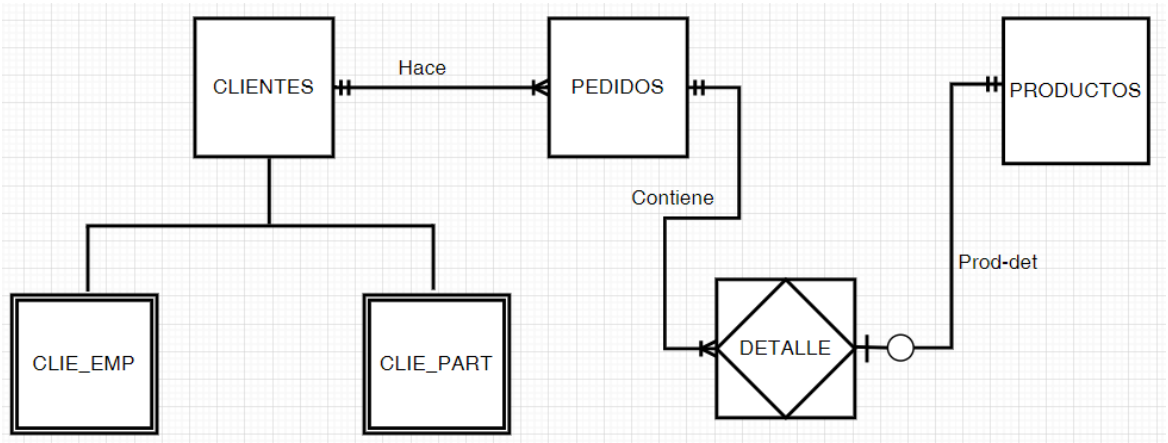
Relación: Trabaja para

**Cardinalidad:**

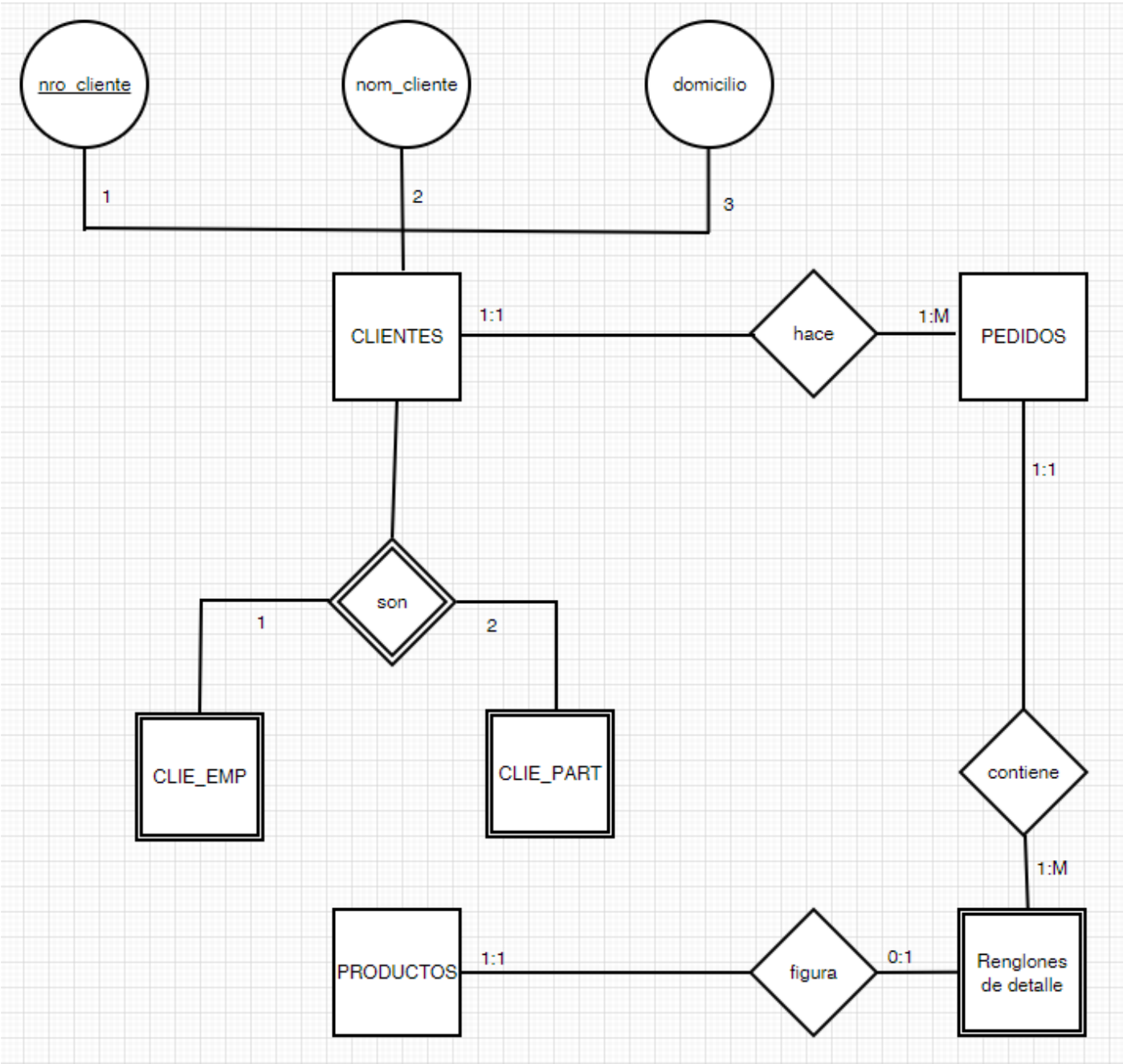
La cardinalidad se refiere a dejar indicado para cada instancia de una entidad, cuántos elementos le puede corresponder de la otra.

Ejemplo ilustrado en la imagen anterior.

**DER DE MARTIN:**



**DER DE CHEN:**


nro_cliente
nom_cliente
domicilio
1
2
3
CLIENTES
1:1
hace
1:M
PEDIDOS
1:1
son
1
2
CLIE_EMP
CLIE_PART
contiene
1:M
PRODUCTOS
1:1
figura
0:1
Renglones de detalle

**EJERCICIOS**

1. Crear el diagrama Entidad / Relación y transformarlo a tablas del modelo relacional.

| ENTIDAD | ATRIBUTO |
|---|---|
| Proyectos | código |
| | nombre |

| Empleados | legajo |
|---|---|
| | nombre |

| Relación | Asignado a |
|---|---|

a) Cardinalidad 1 : 1
b) Cardinalidad 1 : N
c) Cardinalidad N : N

2. Crear el diagrama Entidad / Relación y transformarlo a tablas del modelo relacional.

Entidades:

a) Municipalidades
b) Vivendas
c) Personas

Relaciones

a) Habita
b) Empadronado en
c) Propietario de

Supuestos datos:

a) Cada persona sólo puede habitar en una vivienda.
b) Cada persona puede ser propietaria de más de una vivienda.
c) Una persona está empadronada en una sola muncipalidad
d) En una vivienda pueden habitar más de una persona
e) Una vivienda puede ser propiedad de más de una persona.
f) Una vivienda pertenece a una sola municipalidad.
g) Una municpalidad puede tener muchas viviendas.
h) En una Municipalidad pueden estar empadronadas muchas personas.

3. Crear el diagrama Entidad / Relación y transformarlo a tablas del modelo relacional.

| **ENTIDAD** | **ATRIBUTO** | **ENTIDAD** | **ATRIBUTO** |
|---|---|---|---|
| Pacientes | Código_paciente | Empleados | Código_empleado |
| | Apellido | | Apellido |
| | Dirección | | Salario |
| | Fecha_nacimiento | | Localidad |
| | Sexo | | |
| | Localidad | Doctores | Código_doctor |
| | | | Apellido |
| Salas | Código_sala | | Localidad |
| | Nombre | | |
| | Numero_de_camas | Especialidades | Código_especialidad |
| | | | Nombre |
| Funciones | Código_función | | |
| | nombre | Turnos | Código_turno |
| | | | Horario |

Supuestos datos:

a) Un paciente puede estar internado en una sola sala.
b) Un paciente puede ser atendido por varios doctores.
c) En una sala puede haber muchos pacientes internados.
d) Un empleado puede tener asignado una sola función.
e) Un empleado puede trabajar en más de un turno.
f) Una función puede ser cumplida por más de un empleado.
g) Un doctor puede atender a más de un paciente.
h) Un doctor puede ejercer varias especialidades.
i) Un doctor puede trabajar en más de un turno.
j) Una especialidad puede ser ejercida por más de un doctor.
k) En un turno pueden trabajar más de un empleado.
l) En un turno pueden trabajar más de un doctor.

4. Crear el diagrama Entidad / Relación y transformarlo a tablas del modelo relacional.

El departamento de formación de una empresa desea construir una base de datos para planificar y gestionar la formación de sus empleados.

La empresa organiza cursos internos de formación de los que desea conocer el código de curso, el nombre, una breve descripción, el número de horas de duración y el costo del curso.
Un curso puede tener como prerequisito haber realizado otro(s) previamente. Un curso que es un prerequisito puede serlo de forma obligatoria o sólo recomendable.
Un mismo curso puede ser impartido en diferentes lugares, fechas y con diferentes horarios (día entero, mañana, tarde). En una misma fecha sólo puede impartirse una edición de un curso.
Los cursos se imparten por personal de la empresa.
De los empleados se desea almacenar su código de empleado, nombre y apellidos, dirección, teléfono, cuit/cuil, fecha de nacimiento, nacionalidad, sexo y salario, así como si está capacitado para impartir cursos.

5. Crear el diagrama Entidad / Relación y transformarlo a tablas del modelo relacional.

La municipalidad de Bariloche desea guardar información sobre las estancias que existen en su jurisdicción y brindan alojamiento a pasajeros. Para ello decide crear una base de datos que contemple las siguientes consideraciones:
Un alojamiento rural (estancia) se identifica por su nombre (“La Tranquila”, “La Rosita”, etc) que no se repite, tiene una dirección, un teléfono y una persona de contacto que pertenece al personal del establecimiento.
En cada establecimiento trabajan una serie de personas que se identifican por un código de personal. Se requiere conocer el nombre completo, la dirección y el CUIL.
Aunque en un establecimiento trabajen varias personas, una persona puede trabajar en un solo establecimiento.
Los alojamientos se alquilan por habitaciones con una fecha de ingreso y la cantidad de días de permanencia; se desea conocer cuántas habitaciones componen la estancia, de qué tipo (single, doble, triple, etc) es cada una de ellas, si posee cuarto de baño y el precio diario.
En algunas de estas estancias se realizan actividades organizadas para los huéspedes (senderismo, bicicleta de montaña, etc). Estas actividades se identifican por un código. Es de interés saber el nombre de la actividad, la descripción y el nivel de dificultad de dicha actividad (de 1 a 10).

Estas actividades se realizan un día a la semana, por ejemplo: en la estancia "La Tranquila" se practica alpinismo los sábados y se desea guardar esta información. Puede haber algún día que no se practique ninguna actividad.

Se pide:
Diseñar el esquema relacional. Indicar las claves primarias y claves externas.

## SEGURIDAD

Seguridad se refiere a la protección de los datos contra una alteración o destrucción no autorizada.
La seguridad implica asegurar que los usuarios están autorizados para llevar a cabo lo que tratan de hacer.

El problema de la seguridad tiene muchos aspectos, como ser:
Aspectos legales, sociales y éticos.
Controles físicos.
Cuestiones de política interna.
Problemas de operación.
Controles del equipo.
Seguridad del sistema.
Materias de relevancia específica para el sistema mismo de BD.

Un usuario dado tendrá diferentes derechos de acceso o autorizaciones sobre diferentes objetos de información. Por supuesto todas las decisiones son de política, no técnicas. Lo único que puede hacer el DBMS es obligar el cumplimiento de esas decisiones una vez tomadas. Para ello cuenta con:

- Los resultados de esas decisiones deben darse a conocer al sistema (en SQL se usa GRANT (conceder) y REVOKE (revocar), y el sistema las graba en el catálogo en forma de restricciones de autorización).

- Debe existir alguna forma de verificar una solicitud de acceso dada contra las restricciones de autorización aplicables. Para poder decidir cuáles restricciones son aplicables a una solicitud dada, el sistema debe ser capaz de reconocer el origen de dicha solicitud: o sea, al usuario específico del cual proviene.

## VISTAS Y SEGURIDAD

Para mostrar el empleo de las vistas con propósitos de seguridad, nos basaremos en la tabla PROVEEDORES.

Para un usuario al cual se permite tener acceso a registros completos de proveedores pero sólo de los situados en Avellaneda:

```
CREATE VIEW  V_PROVEEDORES _ AVELLANEDA  AS
    SELECT NUMERO, NOMBRE, SITUACION, LOCALIDAD
    FROM PROVEEDORES
    WHERE LOCALIDAD = 'AVELLANEDA';
```

Las vistas ofrecen una importante medida de seguridad. Hacen posible dividir conceptualmente la base de datos en fragmentos de distintas maneras con objeto de ocultar información confidencial a usuarios no autorizados. Sin embargo, resulta un poco débil sobre todo en situaciones como si algún usuario en particular necesita diferentes derechos sobre diferentes subconjuntos de la misma tabla al mismo tiempo.

**GRANT (conceder) y REVOKE (revocar)**

Permite especificar las operaciones que los usuarios autorizados pueden ejecutar con esos fragmentos.

En general, para poder realizar cualquier operación en SQL, incluso un simple SELECT, el usuario debe contar con la autorización apropiada.

En un motor de base de datos existen autorizaciones relacionadas con utilerías específicas del sistema, con áreas de almacenamiento temporal específicos de la base de datos, etc.

Para que los usuarios accedan a la base y a los objetos deben tener privilegios otorgados. El DBA debe controlar los privilegios, lo que implica:

- Proveer al usuario los permisos necesarios para ejecutar un tipo de operación.
- Habilitar y restringir los accesos y cambios de los datos.
- Habilitar y restringir la posibilidad de ejecutar funciones del sistema y cambiar la estructura de la base de datos.
- Otorgar privilegios a usuarios individuales y a roles.
- Otorgar privilegios a todos los usuarios (PUBLIC)

Existen 2 tipos de privilegios:

- **Privilegios del sistema (Privilegios sobre operaciones de DDL):**

Permiten al usuario realizar una o varias operaciones especiales. Un privilegio del sistema es el derecho de ejecutar una clase de comando y no son específicos a un esquema. Cada privilegio le permite al usuario realizar una operación específica. El alcance de los privilegios del sistema puede abarcar por ej. Crear una tabla en su esquema, en cualquier esquema, crear usuarios o cambiar la estructura de la base de datos, etc.

- **Privilegios sobre objetos (Privilegios sobre operaciones de DML):**

Permiten que los usuarios puedan manipular las tablas, vistas, secuencias o ejecutar procedimientos almacenados. Los privilegios de objetos se deben otorgar para cada uno de los objetos, no hay forma de otorgarlos agrupadamente por objetos, salvo mediante el uso de roles.

Supongamos que el administrador quiere conceder derechos a algún usuario.
La concesión de esos privilegios se realiza mediante la proposición GRANT (tanto para privilegios del sistema como de objetos).

La sintáxis para otorgar privilegios del sistema es la siguiente:

```
GRANT     priv_sistema     TO     user
          Role                    role
                                  public
```

Ejemplo:

```
GRANT ALTER ANY TABLE TO CARLOS;
        GRANT ALTER DATABASE TO JOSE;
        GRANT CREATE TABLE TO MARIA, CARLOS;
        GRANT DROP ANY TABLE TO JUAN;
```

La sintáxis para otorgar privilegios sobre objetos es la siguiente:

GRANT priv_objeto ON objeto TO user
ALL role
public

Ejemplos:

GRANT SELECT ON PROVEEDORES TO BRUNO;

GRANT SELECT, INSERT (NPROV,NOMBRE), UPDATE (SITUACION) ON PROVEEDORES TO PEDRO, MARIA;

GRANT ALL ON PROVEEDORES, PRODUCTOS TO MAURO, JULIO;

GRANT SELECT ON PRODUCTOS TO PUBLIC;

**SINONIMOS:**

Para realizar una identificación completa de un objeto de base de datos (como una tabla o una vista) es necesario especificar el propietario del objeto y el nombre del objeto. Serán necesarios entre uno y cuatro de estos parámetros en función del desplazamiento del objeto. Los desarrolladores pueden crear sinónimos que apunten al objeto adecuado para ocultar este proceso a los usuarios, que sólo tienen que conocer el nombre del sinónimo.

Los sinónimos públicos los comparten todos los usuarios de una base de datos, mientras que los sinónimos privados pertenecen a propietarios individuales.

Por ejemplo, la tabla EMPLEADOS debe ser propiedad de una cuenta (digamos que el propietario es HR). Desde otra cuenta de usuario de la misma base de datos, dicha tabla podría referenciarse como HR.EMPLEADOS. No obstante, esto implica que la segunda cuenta debe conocer que la tabla EMPLEADOS es propiedad de la cuenta HR. Para evitar esto, puede crearse un sinónimo público llamado EMPLEADOS que apunte a HR.EMPLEADOS.

Siempre que se haga referencia a este sinónimo apuntará a la tabla adecuada.

La siguiente sentencia SQL permite crear dicho sinónimo:

**CREATE PUBLIC SYNONYM empleados FOR hr.empleados;**

Los sinónimos permiten proporcionar punteros a tablas, vistas, procedimientos, funciones, paquetes y secuencias.

**Revoke**

También se puede revocar dichas autorizaciones o privilegios con REVOKE.

La sintáxis para revocar privilegios del sistema es la siguiente:

| | | | |
|---|---|---|---|
| REVOKE | priv_sistema | FROM | user |
| | role | | role |
| | | | public |

Ejemplo:

```
REVOKE ALTER ANY TABLE FROM CARLOS;
REVOKE ALTER DATABASE FROM JOSE;
```

La sintáxis para revocar privilegios sobre objetos es la siguiente:

| | | | | | |
|---|---|---|---|---|---|
| REVOKE | priv_objeto | ON | objeto | FROM | user |
| | ALL | | | | role |
| | | | | | public |

Ejemplo:

```
REVOKE SELECT ON PROVEEDORES FROM BRUNO;
REVOKE UPDATE ON  PROVEEDORES FROM MANUEL;
REVOKE INSERT, DELETE ON PROV-PROD FROM ANA, JUAN;
REVOKE ALL ON PROVEEDORES, PRODUCTOS, PROV-PROD  FROM  FERNANDO;
```

**SEGURIDAD – ADMINISTRACIÓN POR ROLES**

Cuando se tienen muchas tablas y/o muchos usuarios lo recomendado es utilizar una política de permisos a través de roles.

Los roles sirven para simplificar el otorgamiento de privilegios y permisos. Son un grupo de permisos que son concedidos a usuarios u otros roles.

Las características de los roles son:

- No pertenece a ningún esquema, pertenece a la base de datos.
- Puede ser otorgado a otros roles, excepto a sí mismo. (el role A no se puede otorgar al role B, si el role B ya ha sido otorgado al role A)
- Puede ser habilitado o no por cada usuario.
- Está documentado en el diccionario de datos.

Beneficios de los roles:

- Reducen el tiempo al otorgar/revocar permisos: se pueden otorgar/revocar muchos privilegios con una sola sentencia.
- Administración dinámica: cuando cambia la funcionalidad de un role, basta cambiar el role para que sean alterados los privilegios de todos los usuarios asociados.
- Performance: pocos privilegios individuales para controlar.

La sintáxis para la creación de un role es:

```
CREATE ROLE NOMBRE_ROLE;
```

**Roles por defecto:**

Luego que fueron otorgados los roles al usuario, se puede indicar cuáles serán por defecto y cuáles no. Cuando el usuario se conecte, los roles por defecto serán automáticamente habilitados.

En caso de no indicarse cuáles son los roles por defecto, todos los roles serán considerados por defecto, produciendo que todos los roles sean automáticamente habilitados cuando el usuario se conecte.

Para establecer los roles por defecto se debe usar el comando ALTER USER.

```
ALTER USER   usuario   DEFAULT ROLE      role, role.
                                         All
                                         None
```

**Ejercicios**:

1) Crear una vista para un usuario al cual se permite tener acceso a todos los registros de proveedores pero sin las situaciones. Luego asignarle permisos a un usuario sobre esa vista.

2) Crear una vista para un usuario al cual se permite tener acceso sólo a los registros de los proveedores situados en Avellaneda, sin las situaciones. Luego asignarle permisos a un usuario sobre esa vista.

3) La tabla DATPERS se define así:

```
CREATE TABLE DATPERS (
                IDENTUSUARIO          VARCHAR (8)
                SEXO                  VARCHAR (1)
                DEPENDIENTES          DECIMAL (2)
                OCUPACIÓN             VARCHAR (20)
                SALARIO               DECIMAL (7)
                IMPUESTO              DECIMAL (7)
                AUDITORÍAS        DECIMAL (2)
            PRIMARY KEY (IDENTUSUARIO)   )
```

Escribir proposiciones en SQL para conceder lo siguiente:

A) Al usuario Gómez, autorización de selección sobre toda la tabla.
B) Al usuario López, autorización de inserción y eliminación sobre toda la tabla.
C) Al usuario Sánchez, autorización de selección sobre toda la tabla y autoridad de actualización sobre los campos SALARIO e IMPUESTO (solamente).
D) Al usuario Torres, autorización de selección sobre los campos IDENTUSUARIO, SALARIO e IMPUESTO (únicamente).
E) Al usuario Pérez, autorización de selección igual a la de Torres y autorización de actualización sobre los campos Salario e Impuesto (solamente).

F) Al usuario Villalba, autorización de selección sobre los registros de predicadores (únicamente). (Predicadores es un tipo de ocupación).
G) Al usuario Giménez, autorización de selección igual a la de Torres y autoridad de actualización sobre los campos IMPUESTO y AUDITORIA (únicamente).
H) Al usuario Aguero, autorización de selección sobre los salarios máximos y mínimos por clase de ocupación pero ninguna otra autorización.

4) Para cada una de las partes (a) - (h) del ejercicio anterior escribir proposiciones en SQL para retirar la autorización indicada al usuario en cuestión.