Guía ejercicios Base de Datos

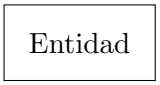
Pablo González-Cantergiani* and Carolina Bonacic*

*Departamento de Ingeniería informática, Universidad de Santiago de Chile

1 Introducción

1.1 Modelo entidad relación

El modelo de Entidades-Relaciones se utiliza para describir el mundo real y sirve como medio de comunicación entre usuarios y desarrolladores [1]



Cualquier objeto físico o abstracto distinguible de un sistema y del cual se desee guardar información. Ejemplos de entidades pueden ser empleados de una empresa, productos que se fabrican en linea de producción o categorías de empleadores.

Se representa con una circunferencia y permiten describir y caracterizar a las entidades. Por ejemplo, los atributos de un alumno son su Nombre, Apellido, Teléfono de Contacto, Correo, Dirección, etc.

Es el elemento principal de las bases de datos relacionales. El vinculo que relaciona las entidades entidades y dice cuantas ocurrencias de una entidad se relacionas con la otra.

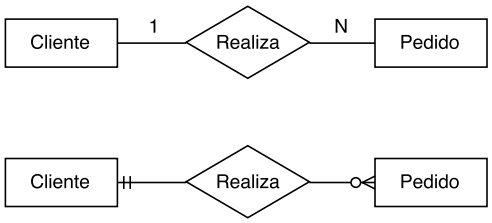


Figure 1: Un cliente realiza 0 o más pedidos y un pedido siempre pertenece a un solo cliente.

1.2 Simbología

 Uno y solo uno.

 Cero o uno.

 Uno o más de uno.

 Cero o más.

[1] Mauricio Marín, *Bases de Datos Relacionales*, Universidad de Santiago de Chile, 2007.

2 Modelo Relacional

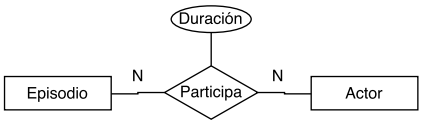
2.1 Algoritmo de Conversión de MER a MR

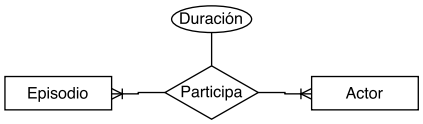
1. Una tabla por cada conjunto de entidades.
2. Una tabla por cada conjunto de relaciones N-N.
3. Definición de columnas para cada tabla:
   (a) Conjunto de entidades: Llave Primaria ∪ Nombre atributo.
   (b) Conjunto de relaciones *r(M-N)* entre 2 entidades A y B:
   Columnas(r) = llave_primaria(A) ∪ llave_primaria(B) ∪ atributos(r)
   (c) Conjunto de relaciones *r(1-1)* entre 2 entidades A y B:
   Columnas(A) = atributos(A) ∪ llave_primaria(B) ∪ atributos(b) [2]
   (d) Conjunto de relaciones *r(1-n)* entre 2 entidades A y B:
   Columnas(B) = atributos(B) ∪ llave_primaria(A) ∪ atributos(r)

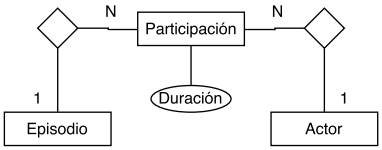
2.2 Ejemplos


```
Cliente(idCliente, rut, nombre, teléfono)

Pedido(idPedido, idCliente, fecha, comentarios)
```







Los tres modelos son equivalentes, su modelo relacional es el siguiente:

```
Episodio(idEpisodio, nombreEpisodio, rate, comentarios)

Episodio_Actor(idEpisodio, idActor, DuraciónParticipacion)

Actor(idActor, nombreActor, fechaNacimiento, teléfono)
```

[2] Generalmente las relaciones 1-1 se agrupan en una sola tabla. Solo casos muy particulares pueden quedar en dos tablas.

3 Patrones

3.1 Documentos

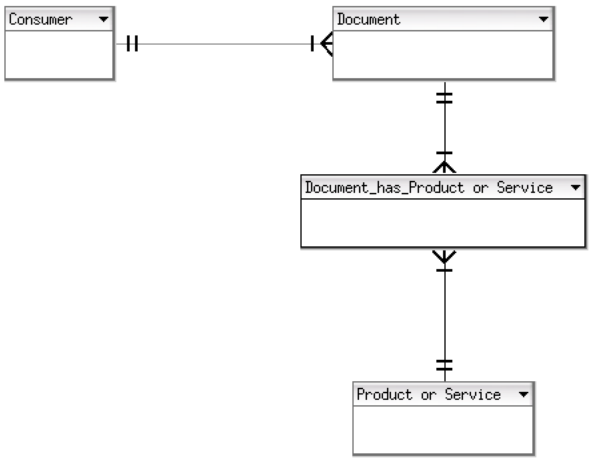
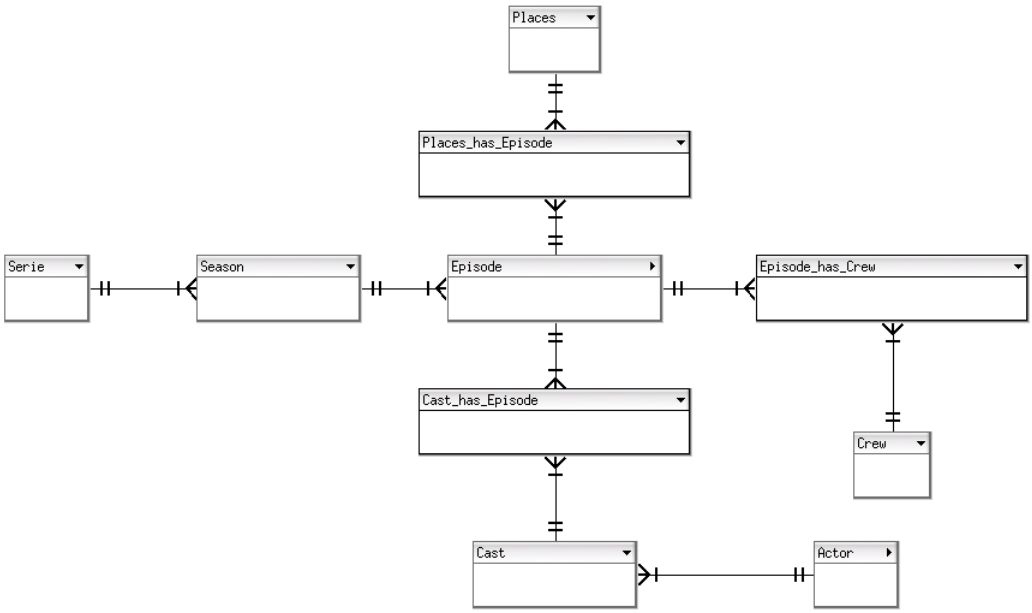


Figure 2: Venta de servicios, facturas, boletas, etc.

4 Problemas

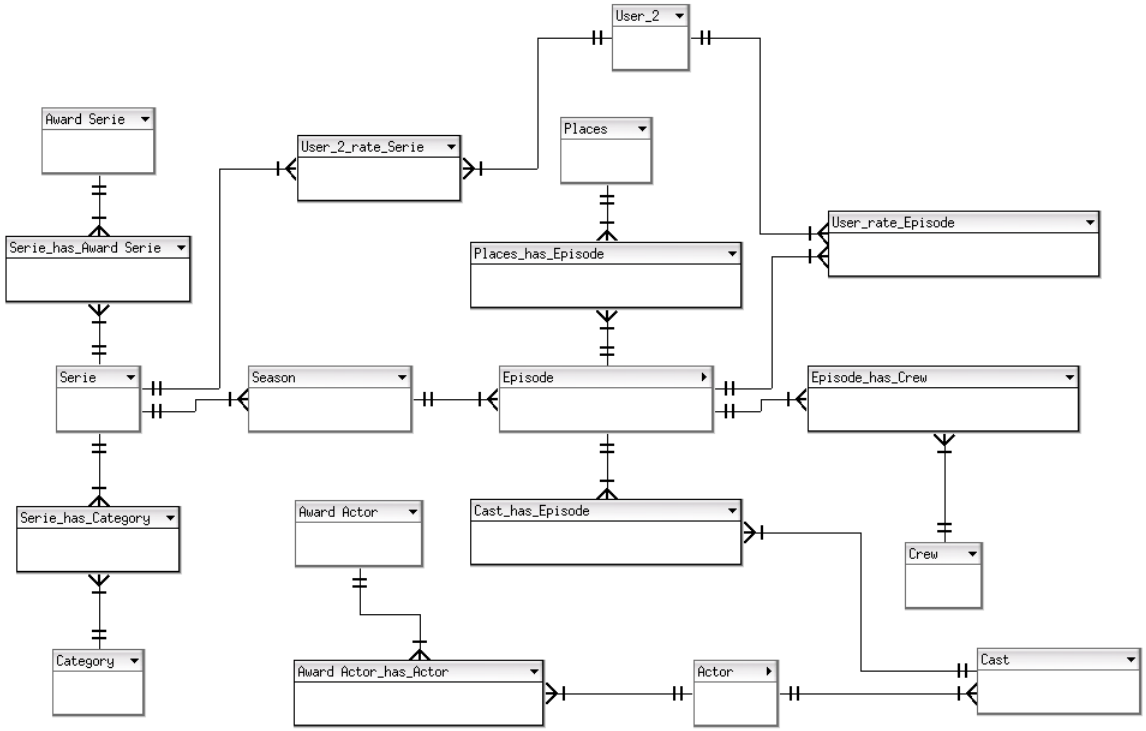
4.1 Catálogo de series

Una serie es una obra audiovisual compuesta de episodios, que mantienen una unidad argumental y con continuidad. Los capítulos se organizan en temporadas, y son estrenados en una fecha concreta. En cada capítulo aparecen diferentes personajes que son interpretados por actores. Además cada capítulo tendrá un director, uno o más guionistas, uno o más productores, y uno o más lugares de rodaje. Una serie tiene una lista de especificaciones técnicas, duración de los capítulos, mezcla de sonido, formato, relación de aspecto (formato de alto y ancho que se emite).



Cada serie puede estar asociada a una o varias categorías: drama, ciencia ficción, etc. Por cada serie se tiene una lista de premios a los que ha sido nominado y los que ha ganado. Los actores de una serie también pueden tener una lista de los premios a los que han sido nominados y los que han ganado por su rol en la serie.

Por cada capítulo de una serie, un usuario podrá hacer una valoración numérica y una escrita. Lo mismo ocurre para la serie en general. Además, el sistema permite realizar valoraciones sólo a usuarios registrados. Para los usuarios registrados se almacenan el nombre, apellido, correo electrónico y valoraciones realizadas.

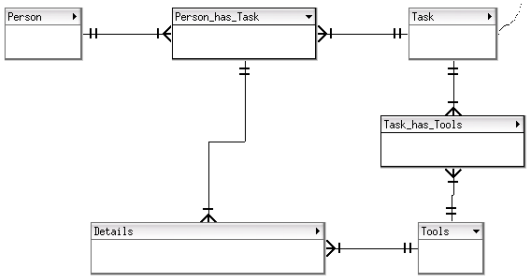


4.2 Arcoch

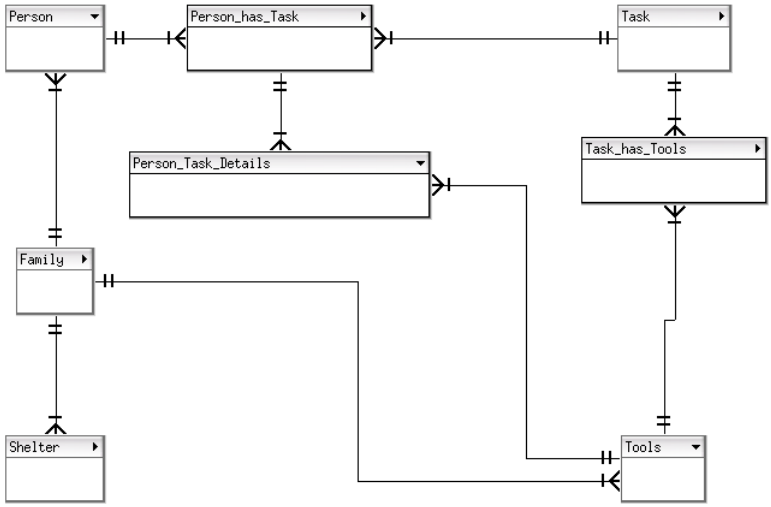
La tribu de los Arcoch, habitantes del Planeta Nak y únicos sobrevivientes de la guerra del año 2084, que destruyó los cuatro planetas de su sistema solar, se han propuesto construir una nueva forma de organización social para proteger su raza, actualmente en extinción.

Cada miembro de la tribu desarrolla alguna actividad, la cual debe aprender y luego perfeccionar.Por ejemplo, cazar, cultivar la tierra, educar, etc.. Cada niño es educado e inicia alguna actividad en la adolecencia, cada anciano puede dejar de trabajar si lo desea.

De acuerdo a la actividad que desarrolle cada Arcoch, se le entregan los elementos necesarios. Si se cambia de actividad estos deben ser devueltos. Se conoce que elementos deben ser entregados por actividad.



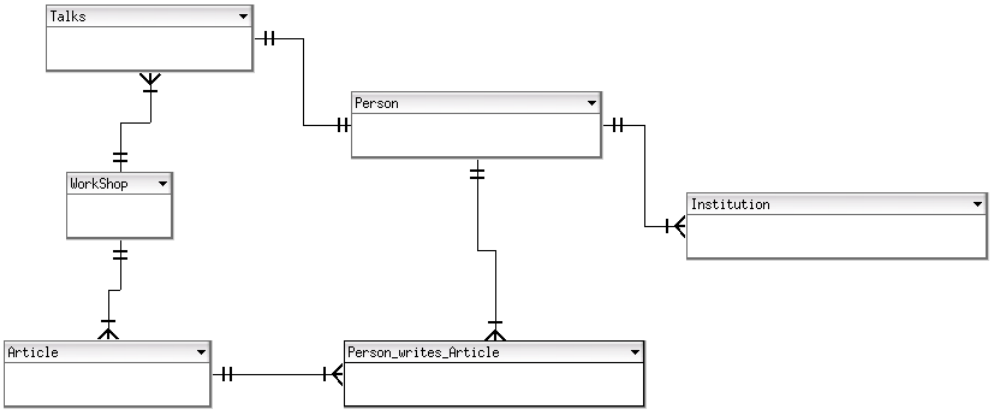
Tal como se otorgan bienes por actividad, se otorgan bienes a las familias de acuerdo a la edad de la familia (tiempo que lleva constituida la familia) y al número de integrantes que tenga. Por ejemplo, cuando dos Arcoch deciden formar una familia, la Tribu les da una choza y otros utensilios.



4.3 Jornadas Chilenas de la Computación

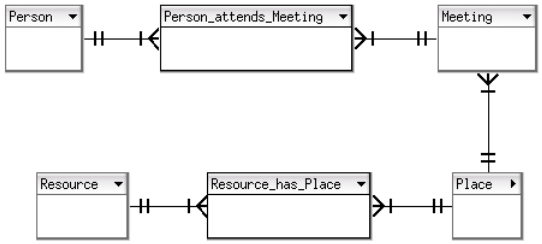
Para las Jornadas Chilenas de Computación es indispensable tener una base de datos donde se guarde la información necesaria para la correcta organización de actividades.

En esta base de datos se tiene información sobre los diferentes Workshops, los artículos que se van a presentar en cada uno y las charlas que se darán. Cada artículo puede estar escrito por más de una persona a diferencia de las charlas donde solo hay un expositor. Tanto las charlas como los artículos son únicos por cada Workshop.



4.4 Reuniones

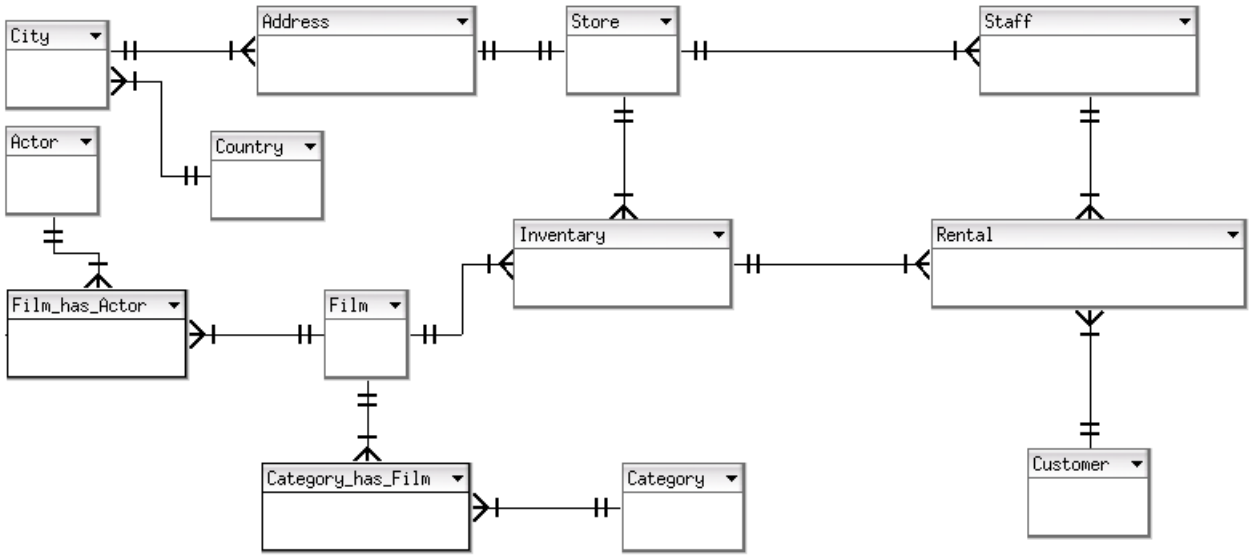
Se desea guardar el registro de reuniones que se realizan en un Hospital. Cada persona es invitada a una o más reuniones, las cuales se realizan en un único lugar. La base de datos permite tambien saber los recursos que se encuentrán en cada lugar.



4.5 Arriendo de películas

Se necesita una base de datos para la administración de una cadena de locales de arriendo de películas. La cadena de arriendo tiene locales en varios países donde comparten una variedad de películas globalmente, cada local cuenta con un *stock* propio a través de un inventario. Cada película tiene una o más categorías como uno o más actores. Se debe conocer la categoría y actor principal para ambos casos.

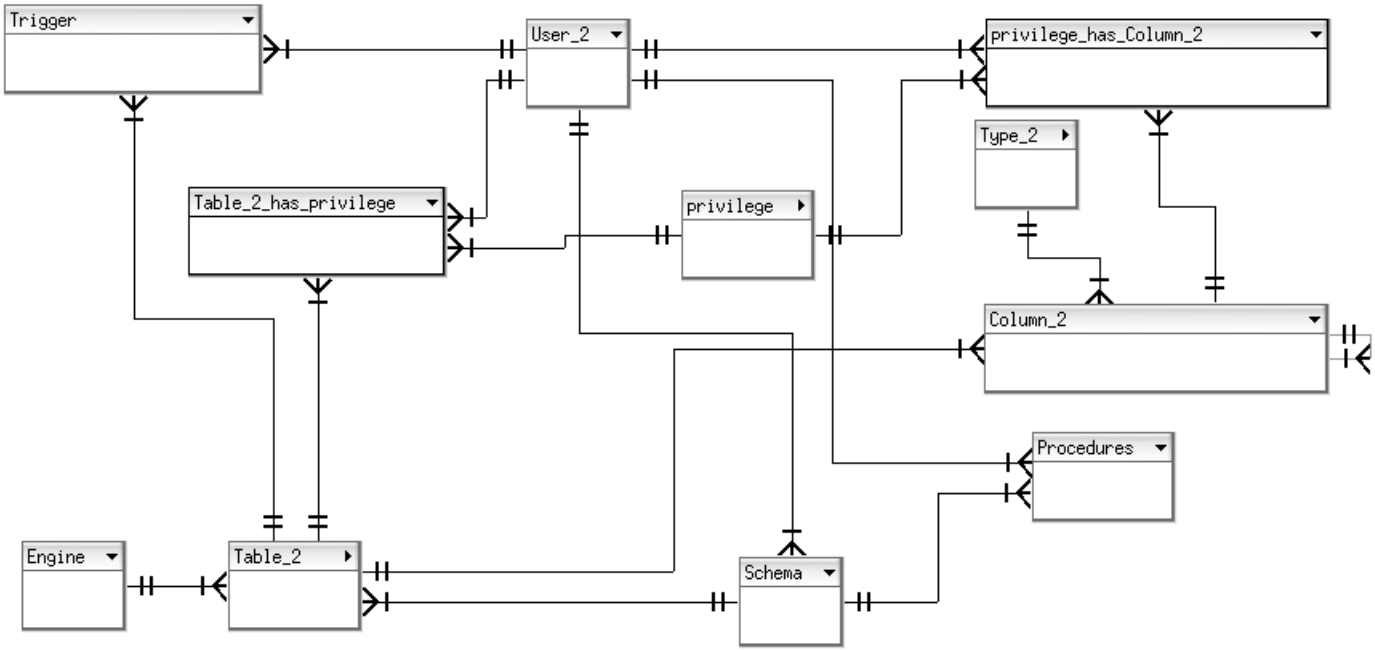
Cuando un usuario realiza un arriendo es necesario saber de que tienda se realizó, que película y que persona atendió al cliente.



4.6 DBMS

Un DBMS es un sistema de gestión que se encarga de administrar bases de datos. En un DBMS un usuario define un conjunto de Esquemas que contienen una o más tablas (con un motor de almacenamiento) y a la vez cada tabla contiene una o más columnas. Cada columna puede ser de un tipo de datos (*Integer*, *Varchar*, etc.) y puede hacer referencia a otra columna de otra tabla, pasando a ser una llave foránea.

Además de los objetos anteriores, un usuario puede definir *triggers* que son acciones asociadas a una tabla y procedimientos almacenadas que son acciones asociadas a un esquema. Cada tabla, esquema y columna tiene asociados privilegios con los usuarios inscritos en el DBMS.



4.7 Pabellón

El Pabellón Quirúrgico es un recinto especialmente diseñado y equipado para garantizar la seguridad en la atención del paciente, que va a ser sometido a una actividad anestésica o quirúrgica.

El módulo de pabellón tiene por finalidad gestionar las diversas intervenciones que se van a realizar a los pacientes, el equipo médico que va a trabajar, materiales, horarios de las cirugías entre otros. La especificación de los requisitos se encuentra en el siguiente listado:

1. Cada Hospital tiene un número determinado de pabellones, los cuales poseen instrumentaría especial para los diversos tipos de cirugías.

2. El sistema debe permitir el ingreso de pacientes, médicos, pabellones, instrumentaría, insumos y cirugías. Además, el sistema debe permitir especificar la instrumentaría necesaria para cada cirugía, así como gestionar la instrumentaría existente en cada pabellón.

3. Cuando se desee agendar una cirugía, es necesario especificar el equipo médico (agregados previamente al sistema) y la cirugía a realizar, automáticamente el sistema devolverá los horarios disponibles de los pabellones que cumplan con los requisitos.

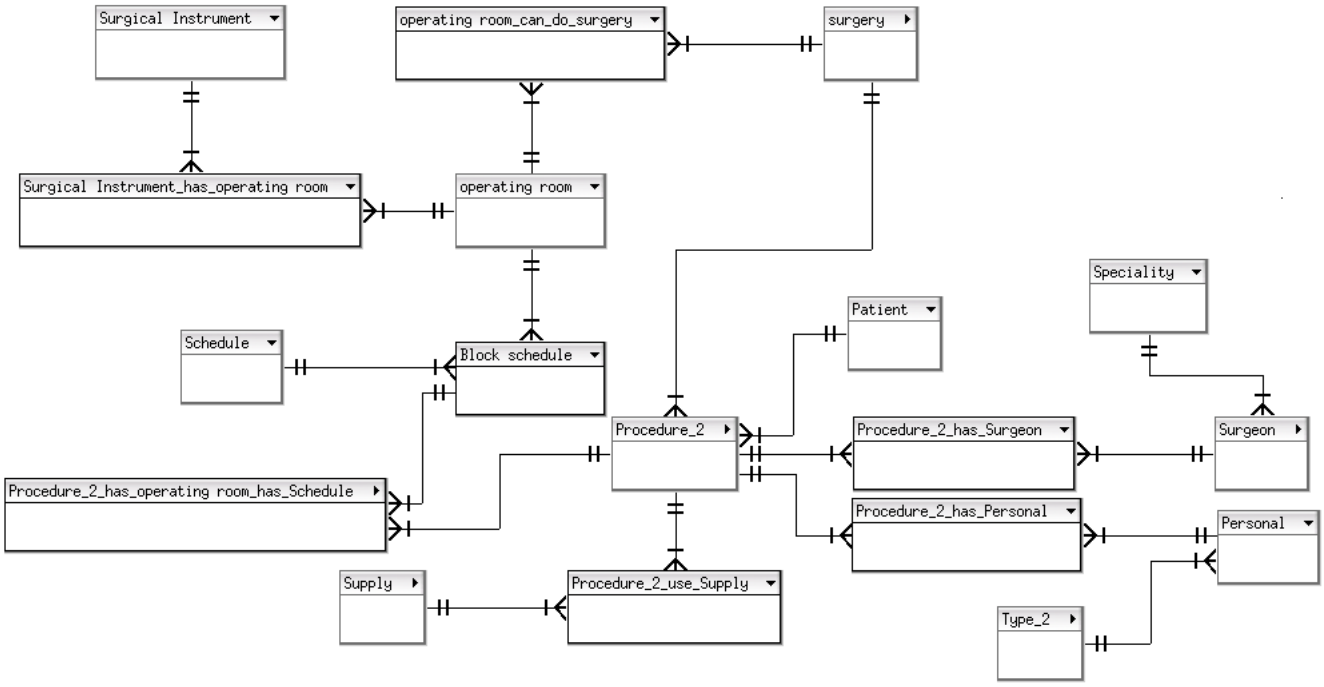
4. Al momento de agendar una cirugía, se especifica el tiempo aproximado de duración. Como es posible que ocurran imprevistos, el sistema debe permitir manejar éstos, asignándole mayor tiempo en caso de ser necesario y replanificando las cirugías que seguían a continuación de la que está en transcurso.

5. El sistema debe permitir ver todas las cirugías a la cual fue sometido un paciente, con su respectivo detalle, como la instrumentaría, insumos, equipo médico y duración, con el fin de realizar la facturación una vez finalice.

6. Cada insumo tiene un precio por unidad y la instrumentaría un precio por hora. En el caso especial de los insumos, cuando se realiza el ingreso al sistema se da un *stock* que se va descontando en cada utilización.

7. Se debe permitir la visualización de todas las cirugías realizadas por los médicos, las cirugías que están en pabellón, las cirugías a realizarse en un futuro, entre otra información que se considere importante para el módulo.

8. Al momento de agendar una cirugía, se debe agregar cirujano, un arsenalero y un anestesista. Si no se encuentran alguno de estos profesionales, la cirugía no podrá ser llevada a cabo.


Surgical Instrument
operating room_can_do_surgery
surgery
Surgical Instrument_has_operating room
operating room
Speciality
Patient
Schedule
Block schedule
Procedure_2
Procedure_2_has_Surgeon
Surgeon
Procedure_2_has_operating room_has_Schedule
Procedure_2_has_Personal
Personal
Supply
Procedure_2_use_Supply
Type_2

5 Ejercicios Propuestos

5.1 Gestión hotelera

Un pequeño hotel rural necesita una aplicación software que le permita gestionar su negocio. Las primeras reuniones se han resumido en un documento expresado en lenguaje natural, que recoge a grandes rasgos la lógica de negocio del sistema a construir. Este documento se presenta a continuación:

El software a construir debe cumplir las siguientes funcionalidades:

1. Gestionar las reservas de habitaciones
2. Gestionar las habitaciones libres/ocupadas
3. Gestionar gastos extras
4. Facturar a los clientes
5. Permitir listados y estadísticas

5.1.1 Reservas

Las reservas se hacen por teléfono, y el cliente debe dar su nombre, RUT, teléfono de contacto. La reserva tendrá validez hasta las 0:00 horas del día siguiente a la supuesta entrada en el hotel. Un cliente puede reservar tantas habitaciones como sean necesarias.

5.1.2 Habitaciones

El hotel tiene tres pisos con habitaciones para los clientes. Cada habitación tiene un número de habitación y pertenece a un tipo de habitación (los tipos existentes son individual, doble y matrimonial). El precio de la habitación depende del tipo de la habitación y de la temporada. Existen tres temporadas: baja, alta y normal. Las habitaciones pueden estar libres, reservadas u ocupadas. Una habitación está ocupada sólo en el momento que el cliente ha llegado al hotel y hay un responsable de la habitación del cual se necesita su nombre, RUT y teléfono de contacto. Si la habitación estaba reservada, la reserva ya no se necesita y puede eliminarse. Sin embargo, si debe existir información histórica del uso del hotel.

5.1.3 Gastos extras

Los gastos extras que se van a poder cargar a la habitación son de naturaleza variada, y se denotará por un concepto y valor.

5.1.4 Facturación

Al abandonar el hotel se realizará una factura asociada al responsable de la habitación, aunque puede llevar otro nombre y otro RUT. La factura reflejará todos los gastos asociados a la habitación o habitaciones perfectamente desglosados en las líneas de la factura. Las facturas no pueden borrarse, aunque si podrían modificarse los gastos involucrados en caso de error.

5.1.5 Listados

El usuario no ha definido todos los listados que requiere, pero si le interesa obtener estadísticas de ocupación por fechas (calendario).

5.2 Distribucción de Vinos

Un distribuidor de vinos ha decidido crear una tienda virtual en Internet a través de la cual vender sus productos. Las primeras reuniones se han resumido en un documento expresado en lenguaje natural, que recoge a grandes rasgos la lógica de negocio del sistema a construir. Este documento se presenta a continuación:

El software a construir debe cumplir las siguientes funcionalidades:

1. Mantenimiento de los productos
2. Mantenimiento de proveedores
3. Mantenimiento de los clientes
4. Gestionar el carrito de la compra de vinos
5. Facturación de los pedidos
6. Permitir listados y estadísticas

5.2.1 Productos

El distribuidor en cuestión comercializa diferentes productos relacionados con el vino. Cada producto (llamémosle tipo de vino), viene definido por un nombre, una denominación de origen, una categoría opcional (cosecha,, reserva, gran reserva, reserva especial), la variedad de uva y su porcentaje, la crianza, una añada, un precio por botella sin IVA, la cata, la gastronomía recomendada, la temperatura a la que se debe servir y los comentarios destacables de ese tipo de vino.

Cada tipo de vino puede distribuirse en diferentes formatos siendo los más habituales (aunque pueden aparecer más) media botella, tres cuartos, litro y medio y cinco litros. No todo tipo de vino tiene porque distribuirse en todos los formatos. Cada tipo de vino de un formato determinado puede venderse en una (y sólo en una) de las dos siguientes posibilidades: por botellas o por cajas de madera de n unidades, de forma que el precio de la caja será el de cada botella multiplicado por el número de botellas más un plus por la caja de madera.

De cada tipo de vino se debe tener constancia del número de unidades de que se dispone, haciendo referencia la unidad al formato de distribución (botella o cajas de n botellas). Cada tipo de vino se compra en una bodega, de forma que de cada bodega se debe conocer el nombre, la dirección, el correo electrónico y una lista de teléfonos de contacto. Además, el cliente podrá configurar cajas de madera con las botellas compradas individualmente (los tipos de cajas disponibles son de 1, 2, 3, 4 y 6 botellas) para poder adquirir una caja así conformada debe llenarse la caja. La caja de madera tendrá un valor adicional y variará en función del tamaño.

Opcionalmente, el cliente puede elegir una dirección diferente a la que enviar el pedido. Si se elige la opción "Regalo", la factura se enviará a la dirección del cliente y el pedido a la dirección indicada

5.2.2 Clientes

Para que un cliente pueda comprar tiene que estar dado de alta en el sistema. Por ello, de cada uno se conocerá su RUT, fecha de nacimiento (no se venderá vino a los menores de 18 años), nombre, apellidos, dirección, correo electrónico y lista de teléfonos. Se contempla la posibilidad de que el cliente sea una empresa, pero entonces se almacenará su RUT, y, obviamente, no hará falta la fecha de nacimiento. Una vez que el cliente está dado de alta se le asignará un nombre de usuario y una clave.

5.2.3 El carro de la compra

El usuario irá seleccionando los productos e incorporarlos a su carrito. Este carrito se podrá vaciar en cualquier momento, o bien confirmar su contenido para conformar el pedido final. No se desea guardar información histórica de los carritos de la compra.

5.2.4 Facturación

Cuando el cliente ha confirmado su carrito, se emite una factura que se le enviará con la mercancía, excepto si el pedido era para regalo. Debe tenerse constancia de la dirección a la que se envió la factura. La factura siempre se paga con VISA en el momento de confirma el pedido. La factura detallará perfectamente todos los productos comprados, más una cantidad fija por gastos de envío. Las facturas no se borrarán, ni podrán modificarse, pero podrán imprimirse tantas veces como sea necesario.

5.2.5 Listados

El usuario no ha definido todos los listados que requiere, pero si le interesa obtener estadísticas de compras por tipos de vino.

5.3 Clínica

Se tiene una clínica que tiene 16 *box* de consultas ambulatorias de tres especialidades: cirugía, medicina y pediatría. Las horas de atención pueden reservarse vía telefónica o por Web, y pueden ser solicitadas para un médico en particular o sólo indicando la especialidad que se necesita, con lo que se asigna al profesional que posee menor carga horaria.

Los pacientes que van a la consulta, pueden ser dados de alta y enviados a su domicilio con tratamiento ambulatorio o ser ingresados para hospitalización con lo cual se realiza una solicitud de cama a la enfermera encargada del turno.

Para realizar el ingreso de un paciente es necesario poseer los datos de su previsión. La enfermera encargada debe revisar los diferentes mapas de piso y según la patología, asignar la cama que corresponde a cada especialidad.

Al ingresar un paciente, el médico que lo deriva indica el tratamiento a seguir el cual se divide en: tipo de régimen alimenticio, tratamiento farmacológico, exámenes o medios de diagnostico y cuidados de enfermería.

La clínica cuenta con una farmacia para los pacientes ambulatorios y otra aparte para los pacientes hospitalizados. Las indicaciones del médico se envían a farmacia y un técnico para-médico las lleva al piso tres veces al día. Es sumamente importante tener registro cada vez que un medicamento se despacha de la farmacia y quién realiza el transporte hacia las distintas áreas de especialidad. Una vez entregado en piso, si la enfermera recepciona conforme los fármacos, los administra al paciente en los horarios correspondientes.

Al momento de realizar el alta a un paciente se le genera una cuenta, la cual se calcula en base a los días de estadía, los cuales cambian según el tipo de habitación, los procedimientos realizados, los fármacos e insumos empleados y cada una de las visitas médicas las cuales varían su valor según especialidad.

5.4 Empresa subconstractivas

Se necesita crear una BD para la municipalidad de una región, que permita llevar la mantención de las empresas contratistas que realizan trabajos para ella.

Para lo anterior tome en cuenta lo siguiente: las empresas contratistas realizan obras, como son: pavimentación de calles, mantenimiento de edificios, mantenimiento de computadores, etc. Las empresas tienen empleados, los cuales realizan labores, sólo una labor (ingeniero, operario, soldador, capataz, etc) estos están asignados a una obra (sólo una). Las empresas tienen maquinarias, las cuales pueden ser usadas en distintas obras. También es importante saber que las empresas pueden subcontratar a otras.

De la base de datos se puede obtener información tal como: el listado de obreros según obra, el listado de obras que está ejecutando una empresa, una máquina determinada en cuántas obras participa, datos personales de los obreros o instituciones.

Otros datos interesantes son: toda obra tiene un presupuesto, las empresas tienen un único representante, las máquinas tienen peso, marca, modelo, etc

6 Normalización

Es el proceso de organizar de manera eficiente los datos dentro de una Base de Datos. Incluye la creación de tablas, relaciones entre ellas, para proteger los datos; eliminar redundancia y dependencias incoherentes.

Objetivos de la normalización:

1. Eliminación de datos redundantes.
2. Evitar problemas de actualización de datos en las tablas.
3. Garantizar que las dependencias de los datos sean lógicos.

6.1 Primera Forma Normal (1FN)

Los datos deben ser atómicos o escalares, es decir, no se deben utilizar varios campos en una sola tabla para almacenar datos similares (tener solo dos dimensiones).

6.2 Dependencias Funcionales

Es la relación que existe entre los atributos no llaves y la llave candidata.

```
rut -> nombre, apellido
código-> modelo, patente, color, capacidad, año
```

Por ejemplo, sea R una relación con los siguientes atributos:

```
(rut, nombre, dirección, código_departamento, nombre_departamento,
    código_proyecto, nombre_proyecto, dirección_proyecto, horas)
```

Las dependencias funcionales son:

```
rut -> nombre, dirección
código_departamento -> nombre_departamento
código_proyecto -> nombre_proyecto, dirección_proyecto
rut, código_proyecto -> horas
```

6.3 Segunda Forma Normal (2FN)

Cada atributo que no es llave, depende funcionalmente de toda la llave. Por ejemplo, sea la relación:

```
Proveedores(código_proveedor, nombre_proveedor, código_insumo, precio)
```

Las dependencias funcionales son:

```
código_proveedor->nombre_proveedor
código_proveedor, código_insumo->precio
```

Si la llave es (código_proveedor, código_insumo), la relación Proveedores, no está en 2FN, ya que el atributo nombre_proveedor depende de parte de la llave y no de toda la llave.

6.4 Tercera Forma Normal (3FN)

Elimina los atributos que no dependan de las llaves, no acepta dependencias transitivas. Cada atributo que no es llave, no depende funcionalmente de forma transitiva de la llave.

Por ejemplo, sea la relación `R(ciudad, calle, comuna)`, con sus dependencias funcionales:

```
ciudad, calle -> comuna
comuna -> ciudad
```

con llave `(ciudad, calle)`. R está en 3FN. Cada atributo de la tabla no depende funcionalmente de forma transitiva de la llave primaria. El atributo ciudad, es parte de la llave.

6.5 Forma Normal de Boyce Codd (FNBC)

Una tabla está en FNBC cuando no existen dependencias funcionales transitivas entre atributos no primos y atributos primos. Siendo un atributo primo aquellos que forman parte de la llave primaria, y los atributos no primos, aquellos que no forman parte de la esta llave.

Por ejemplo, sea la relación `Tutorías(rut, asignatura, tutor)` y cuyas dependencias funcionales son:

```
rut, asignatura -> tutor
tutor -> asignatura
```

Si la llave es `(rut, asignatura)` la relación Tutorías, no está en FNBC. Para que sí lo esté el esquema de relación debe quedar de la siguiente manera:

```
Tutorías1 (rut, tutor)
Tutorías2 (tutor, asignatura)
```

En FNBC, se debe tener cuidado al momento de descomponer, ya que se podría perder información por mala descomposición. Al final del proceso de normalización, no debe haber anomalías ni pérdida de información.

6.6 Dependencias Multivaluadas

Son los atributos que tienen varios posibles valores para una sola instancia de la entidad. Por ejemplo, sea la relación `Cursos(curso, profesor, texto)`. Existe una relación entre curso y profesor y que es totalmente independiente de la relación que existe entre curso y texto. Por lo tanto, existe una dependencia multivaluada entre los atributos.

```
curso ->-> profesor | texto
```

6.7 Cuarta Forma Normal (4FN)

Una relación está en 4FN, si y sólo sí para cualquier combinación llave-atributo, no existen valores duplicados. Es decir, una relación está en 4FN si está en FNBC y no tiene dependencias multivaluadas.

Por ejemplo, sea la siguiente relación `R(restaurante, variedad_pizza, area_envío)`. Cada fila (información de la tabla) indica que un restaurante puede entregar una variedad de pizza en un área de envío determinada.

La tabla tiene una llave única y ningún atributo no llave, entonces no viola ninguna forma normal hasta FNBC. Pero, debido a que variedad de pizza que ofrece un restaurante, es independiente del área de envío, existe redundancia en la tabla.

Para satisfacer la 4FN, la relación de atributos debe dividirse en tablas distintas:

```
R1 (restaurante, variedad_pizza)
R2 (restaurante, área_envío)
```

6.8 Ejemplos relacionados

Sea la relación R, `Estudiante(rut, especialidad, actividad)`.

rut	especialidad	actividad
100	música	natación
100	contabilidad	natación
100	música	tenis
100	contabilidad	tenis
200	matemática	running
⋮	⋮	⋮

Si se quiere agregar la tupla (100, música, esquí), para mantener la consistencia de los datos, se debería agregar la tupla (100, contabilidad, esquí) y así para cada nuevo cambio.

Para evitar las anomalías, se construyen las dependencias multivaluadas, donde se almacenan los datos para cada uno de los atributos de valores múltiples. Es decir, la nueva relación Estudiante, quedaría de la siguiente manera:

```
E1 (rut, especialidad)
E2(rut, actividad)
```

Sea la relación `Agenda(nombre, teléfono, correo)`

Esta relación puede representar el caso como el siguiente: una misma persona puede tener varios teléfonos y varios correos. Entonces, Agenda no está en 4FN, porque se se fija el atributo nombre y un teléfono, se pueden tener varios correos. Lo mismo ocurre en el caso de fijar el atributo nombre y correo.

La solución es descomponer en:

```
A1(nombre, teléfono)
A2(nombre, correo)
```

Nota: la situación descrita depende del sistema que se está modelando. Por ejemplo, si se exige que cada persona tenga sólo un teléfono y un solo correo, entonces la relación Agenda, sí está en 4FN.

1. Sea la relación R(bebe, madre, enfermera, médico). ¿R está en 4FN?
2. Sea la relación R(facultad, carrera, electivos, número_electivos). ¿R, está en 4FN?

6.8.1 Ejemplos

Sea la relación `Vacaciones(id_lugar, nombre_lugar, id_cliente, nombre_cliente, fecha)`.

Normalizar hasta 4FN.

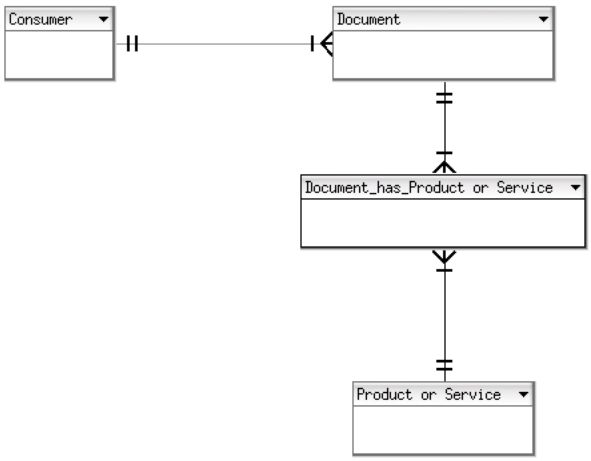
1. La relación Vacaciones, ¿tiene atributos atómicos?, sí, por lo tanto se encuentra en 1FN.
2. ¿Cuál es la llave? id_lugar, id_cliente, fecha.
3. La relación Vacaciones, ¿está en 2FN? no está en 2FN, porque nombre_lugar, depende de parte de la llave y no de toda la llave.
   (a) el atributo nombre_lugar, depende de id_lugar. Entonces, se crea la nueva relación Lugar(id_lugar, nombre_lugar).
   (b) el atributo nombre_cliente depende de id_cliente. Entonces, se crea la nueva relación Cliente(id_cliente, nombre_cliente)
   (c) finalmente, se tiene la relación LugarCliente(id_cliente, id_lugar, fecha)
   (d) Ahora sí, está en 2FN.
4. ¿Existen dependencias transitivas?, no existen dependencias transitivas, por lo tanto está en 3FN.
   (a) ¿Satisface la FNBC?, no existen dependencias transitivas entre atributos no primos y atributos primos, por lo tanto está en FNBC.
5. Las tablas resultado que están en FNBC son:

   ```
   Lugar(id_lugar, nombre_lugar)
   Cliente(id_cliente, nombre_cliente)
   LugarCliente(id_cliente, id_lugar, fecha)
   ```

6. Verificar si el nuevo esquema está en 4FN. Para ello se debe verificar la tabla LugarCliente.
   (a) Bajo el supuesto que una persona no puede estar en una misma fecha en dos lugares distintos, la relación LugarCliente, se encuentra en 4FN (no se encuentran dependencias multivaluadas)

7 SQL

7.1 Data Definition Language (DDL)




```
CREATE TABLE Consumer(
ConsumerID int NOT NULL AUTO_INCREMENT,
Name varchar(255)
Enable TINYINT(1) NOT NULL,
Email varchar(255),
PRIMARY KEY(ConsumerID))

CREATE TABLE Document(
DocumentID int NOT NULL AUTO_INCREMENT,
FolioID int,
Consumer int,
Comments varchar(500)
PRIMARY KEY(DocumentID),
FOREIGN KEY (Consumer) REFERENCES Consumer(ConsumerID))

CREATE TABLE Document_has_Product(
Consumer int,
Product int,
Quantity int NOT NULL,
FOREIGN KEY (Consumer) REFERENCES Consumer(ConsumerID)
FOREIGN KEY (Product) REFERENCES Product(ProductID) )

CREATE TABLE Product(
ProductID int NOT NULL AUTO_INCREMENT,
Sku int UNIQUE, [3]
Name varchar(255),
Color varchar(100),
is_active TINYINT(1) NOT NULL,
Weight int(11) unsigned ) [4]
```

[3] No se permiten repeticiones del valor en la tabla

[4] unsigned solo permite valores positivos

7.2 Structured Query Language (SQL)

```
SELECT [DISTINCT] [ AVG | SUM | COUNT | MAX | MIN] Column_i
FROM Table_j
[ INNER | LEFT | RIGHT | FULL] JOIN Table_k
[ON Table_j.value = Table_k.value]
[ WHERE value [ > | < | >= | =< | <> o =!  | [NOT] IN | BETWEEN] [ table | value | Column] ]
[ GROUP BY Column_x ]
[ HAVING condition ]
[ ORDER BY Column_y ]
```

```
SELECT c.Name
FROM Consumer c
```

Name
Aiden
Jackson
Ethan
Liam
Mason
Noah
Ethan
⋮

```
SELECT DISTINCT c.Name
FROM Consumer c
```

Name
Aiden
Jackson
Ethan
Liam
Mason
Noah
⋮

```
SELECT *
FROM Consumer c
WHERE c.Enable = 0
```

ConsumerID	Name	Enable	Email
1	Aiden	1	aiden@mail.com
2	Jackson	1	jackson@mail.com
3	Ethan	1	ethan@email.com
4	Liam	0	liam@post.com
⋮	⋮	⋮	⋮

```
SELECT * [5]
FROM Consumer c
WHERE c.Enable = 0 AND c.Email LIKE '%usach%' [6]
```

ConsumerID	Name	Enable	Email
35	Anthony	0	anthony@usach.com
48	Chen	0	uusachen@mail.com
165	Atom	0	atom@usach.com
513	Elvis	0	elvis@usach.com
⋮	⋮	⋮	⋮

[5] Tambien es posible escribir c.*

[6] El signo % se utiliza para definir *wildcards*

```
SELECT p.nombre, weight
FROM Product p
GROUP WHERE p.weight BETWEEN 5 AND 1000
```

Nombre	weight
Bow saw	6
Drill	3
Nail	1
Saw	20
⋮	⋮

```
SELECT p.color, SUM(weight) AS weight_color
FROM Product p
GROUP BY (p.color);
ORDER BY (weight_color)
```

Color	weight_color
red	1031
gray	950
Emerald	902
Turquoise	788
⋮	⋮

```
SELECT *
FROM Consumer c, Document d
WHERE c.ConsumerID = d.Consumer

SELECT *
FROM Consumer c
INNER JOIN Document d
ON c.ConsumerID = d.Consumer [7]

SELECT *
FROM Consumer c
JOIN Document d
ON c.ConsumerID = d.Consumer [8]
```

ConsumerID	Name	Enable	Email	DocumentID	FolioID	Consumer	Comments
1	Aiden	1	aiden@...	2	15	1	...
1	Aiden	1	aiden@...	3	17	1	...
35	Anthony	0	anthony@...	17	22	35	...
38	Nadia	1	nadi@...	23	53	38	...
38	Nadia	1	nadi@...	24	59	38	...
39	Jeff	0	jeff@...	25	61	39	...
⋮	⋮	⋮	⋮	⋮	⋮	⋮	...

[7] La consulta *SELECT * FROM T1 INNER JOIN T2 ON T1.ID=T2.ID* y la consulta *SELECT * FROM T1, T2 WHERE T1.ID=T2.ID* son equivalentes. Se recomienda el uso de INNER JOIN por su legibilidad (*Syntactic sugar*)

[8] Por defecto los JOIN se interpretan como INNER JOIN (No se recomienda utilizar esta sintaxis)

```
SELECT *
FROM Consumer c
LEFT JOIN Document d
ON c.ConsumerID = d.Consumer
```
[9]

ConsumerID	Name	Enable	Email	DocumentID	FolioID	Consumer	Comments
1	Aiden	1	aiden@...	2	15	1	...
1	Aiden	1	aiden@...	3	17	1	...
2	Jackson	1	Jackson@...				...
⋮	⋮	⋮	⋮	⋮	⋮	⋮	...
34	Fran	1	fran@...				...
35	Anthony	0	anthony@...	17	22	35	...
⋮	⋮	⋮	⋮	⋮	⋮	⋮	...

```
SELECT *
FROM Document c
WHERE Consumer IN (
    SELECT ConsumerID
    FROM CONSUMER
    WHERE ENABLE = 1)
```
[10]

DocumentID	FolioID	Consumer	Comments
2	15	1	...
3	17	1	...
23	53	38	...
24	59	38	...
⋮	⋮	⋮	⋮

```
SELECT c.Quantity, COUNT(c.Quantity)
FROM Document_has_product c
WHERE Quantity = 50
GROUP BY Quantity
UNION
SELECT c.Quantity, COUNT(c.Quantity)
FROM Document_has_product c
WHERE Quantity = 60
GROUP BY Quantity
```

c.Quantity	COUNT(c.Quantity)
50	6541
60	321

[9]También existe el RIGHT JOIN y el FULL JOIN
[10]También existe el NOT IN

7.3 Ejercicios SQL

7.3.1 Arriendo de películas

```
pelicula(peliculaID, nombre_pelicula,nombre_protagonista, Año_lanzamiento)
censura(censuraID,nombre_censura)
tipo(tipoID, nombre_tipo)
pelicula_censura(peliculaID,censuraID)
pelicula_tipo(peliculaID,tipoID)
cliente(clienteID,nombre_cliente,telefono,direccion)
arriendo(peliculaID,clienteID,fecha_Arriendo,fecha_devolución,esta_devuelto)
```

1. Listado de Clientes (Código y Nombre) no han devuelto películas.

```
SELECT c.nombre_cliente, c.clienteID
FROM Cliente c, Arriendo a
WHERE c.clienteID = a.clienteID
    AND a.esta_devuelto = false
```

2. Listado de películas donde el protagonista fue "Varios" y estrenadas después de 1970.

```
SELECT p.nombre_pelicula, c.peliculaID
FROM pelicula p
WHERE p.nombre_protagonista = 'varios'
    AND Año_lanzamiento > 1970
```

3. Imprima un resumen de películas por censura, el formato de salida es: CodCen, NomCen, Cantidad.

```
SELECT c.CodCen, c.NomCen, COUNT(pc.CodPel) AS NumeroDePeliculas
FROM pelicula_censura pc
INNER JOIN censura c
ON pc.censuraID = c.censuraID
GROUP BY (c.censuraID)
```

4. ¿Cuántas veces se arrendó la película *UP*?

```
SELECT p.nombre_pelicula, COUNT(p.peliculaID)
FROM pelicula p
INNER JOIN arriendo
ON p.peliculaID = a.peliculaID
WHERE p.nombre_pelicula = 'UP'
GROUP BY(a.peliculaID)
```

5. ¿Cuántas películas han arrendado las personas que viven en "Providencia".

```
SELECT COUNT(a.peliculaID)
FROM arriendo a
INNER JOIN cliente c
ON c.clienteID = a.clienteID
WHERE c.dirección LIKES '%providencia%'
```

6. Listado (Nombres) de películas arrendadas por "Felipe", en marzo de este año y que sean de acción, ordenado por nombre de película.

```
SELECT nombre_pelicula
FROM pelicula
WHERE peliculaID IN (
    SELECT a.peliculaID
    FROM arriendo a
    INNER JOIN cliente c
    ON c.clienteID = a.clienteID
    INNER JOIN pelicula_tipo pt
```

```
    ON pt.peliculaID = a.peliculaID
    INNER JOIN tipo t
    ON pt.tipoID = pt.tipoID
    WHERE c.nombre_cliente = 'Felipe'
    AND a.fecha_Arrriendo BETWEEN '2014-03-01' AND '2014-03-31'
    AND t.nobre_tipo = 'accion'
ORDER BY (nombre_pelicula)
```

7. ¿Cuántas películas de Censura "may. de 18" arrendó "José" durante el 2013?

```
SELECT count(a.peliculaID)
FROM arriendo a
INNER JOIN cliente c
ON c.clienteID = a.clienteID
INNER JOIN pelicula_censura pc
ON pc.peliculaID = a.peliculaID
INNER JOIN censura c
ON pc.censuraID = c.censuraID
WHERE c.nombre_cliente = 'JOSE'
AND a.fecha_Arrriendo BETWEEN '2014-03-01' AND '2014-03-31'
AND c.nombre_censura = "may.  de 18"
```

8. ¿Cuál es el tipo de películas más arrendada?

```
SELECT t.nombre_tipo, count(t.tipoID) as masArrendada
FROM arriendo a
INNER JOIN pelicula p
ON p.peliculaID = a.peliculaID
INNER JOIN pelicula_tipo pt
ON pt.peliculaID = a.peliculaID
INNER JOIN tipo t
ON t.tipoID = pt.tipoID
GROUP BY p.tipoID
ORDER BY masArrendada
LIMIT 1
```

9. Promedio Mensual de películas arrendadas por "Lorena" durante el año 2014.

```
SELECT month(a.fecha_arriendo) as month, avg(t.tipoID)
FROM arriendo a
INNER JOIN cliente c
ON c.clienteID = a.clienteID
WHERE c.nombre_cliente = "Lorena %"
AND year(a.fecha_arriendo) = "2014"
GROUP BY month(a.fecha_arriendo)
```

10. ¿Qué Clientes (nombre y código) han arrendado películas de "Acción" y censura "Todo Espectador"?

```
SELECT c.clienteID, c.nombre_cliente
FROM arriendo a
INNER JOIN cliente c
ON c.clienteID = a.clienteID
INNER JOIN pelicula_tipo pt
ON pt.peliculaID = a.peliculaID
INNER JOIN pelicula_censura pc
ON pc.peliculaID = a.peliculaID
INNER JOIN censura c
ON pc.censuraID = c.censuraID
INNER JOIN tipo t
ON t.tipoID = pt.tipoID
WHERE t.nombre_tipo = "Acción" AND c.nombre_censura = "Todo Espectador"
```

11. ¿Cuáles fueron las Películas que se arrendaron más de 2 veces durante el 2013 (indique el nombre y código)?

```
SELECT p.nombre_pelicula, p.peliculaID
```

```
FROM arriendo a
INNER JOIN pelicula p
ON a.peliculaID = a.peliculaID
WHERE year(a.fecha_arriendo) = "2013"
GROUP BY (p.peliculaID)
HAVING COUNT(p.peliculaID) > 2
```

12. Imprima un listado de todos los clientes, indicando la cantidad de dinero invertido por ellos en arriendos.

```
SELECT c.clienteID, (count(c.clienteID) * PRECIO_ARRIENDO) AS Precio_Invertido
FROM arriendo a, cliente c
WHERE c.clienteID = a.clienteID
GROUP BY(c.clienteID)
```

7.4 Ejercicios Propuestos

7.4.1 Arriendo de propiedades

```
ARRENDATARIO(rut, nombre, apellido)
ARRIENDA(rut, id_casa, deuda)
TELEFONOS(rut, fono)
DUEÑO(rut, nombre, apellido)
CASA(id_casa, rut, nro, calle, comuna)
```

1. Los arrendatarios que arriendan la casa ubicada en la calle Carrera 123, de Santiago.
2. ¿Cuánto le deben a María Perez? se supone que hay solo una persona con ese nombre.
3. ¿Cuál es la deuda total de cada dueño?
4. Listar todas las personas de la base de datos
5. Listar los dueños que poseen 3 o más casas.
6. Listar los dueños que tengan deudores en todas sus casas (dueño con deudas en todas sus casa ó el complemento: dueños con casas sin deudas)

7.4.2 Prestamos

```
sucursal(rut_sucursal, nombre, ciudad, capital)
cliente(rut_cliente, nombre, dirección, ciudad)
cuenta(nro_cuenta, rut_sucursal, saldo)
prestamo(nro_prestamo, rut_sucusal, cantidad)
cliente_cuenta(rut_cliente, nro_cuenta)
cliente_prestamo(rut_cliente, nro_prestamo)
```

1. ¿Cuáles son los clientes (nombres) que tienen una cuenta y un préstamo?
2. Encontrar los nombres de todos los clientes que tienen un préstamo en la sucursal Providencia (nombre) de Santiago.
3. Nombre de los clientes que tienen un préstamo en la sucursal Providencia de Santiago, pero no tienen una cuenta en dicha sucursal.
4. Encontrar el saldo mayor para cualquier cuenta.
5. Clientes que tienen una cuenta en (por lo menos) las sucursales Providencia y Vitacura.
6. Clientes con cuentas en todas las sucursales de la ciudad de Antofagasta.

7.4.3 Vuelos (A.R)

```
vuelos(nro_vuelo, desde, hasta)
avión_utilizado(nro_vuelo, tipo_avión, nro_avión)
info_pasajeros(nro_vuelo, rut, nombre, origen, destino)
```

Los vuelos no pueden tener más de dos escalas y no hay cambio de tipo de avión para un mismo número de vuelo.

1. Listar los números de vuelos desde la A hasta la F.
2. Listar los tipos de avión que no son utilizados en ningún vuelo que pase por B.
3. Listar los números de vuelos para aquellos pasajeros que viajan de A a D pasando por B.
4. Listar los tipos de avión que son utilizados en todos los vuelos que pasan por C.

7.4.4 BAR (A.R.)

```
FRECUENCIA(bebebor, bar, desde)
SIRVE(bar, cerveza)
GUSTA(bebebor, cerveza)
```

1. ¿Qué cervezas sirven en el bar "don lucho"?
2. Listar los bebebores que no frecuentan bares con cervezas de su agrado
3. Sabiendo que hay bares que sirven todos los tipos de cervezas, indicar cuales son.
4. Listar los bebebores que frecuentan todos los bares.
5. Listar los bebebores que gustan de todos los tipos de cervezas.
6. Listar los bebebores que no les gusta la cerveza "austral".
7. Listar los bares que van a quebrar, ya que ningún bebebor los frecuenta.
8. ¿Desde cuando José frecuenta el bar "don lucho"?
9. Listar los bebebores que beben lo mismo que Jorge. Supuesto: un bebebor consumirá sólo la cerveza disponible en los bares que frecuenta y sólo la que le gusta.