

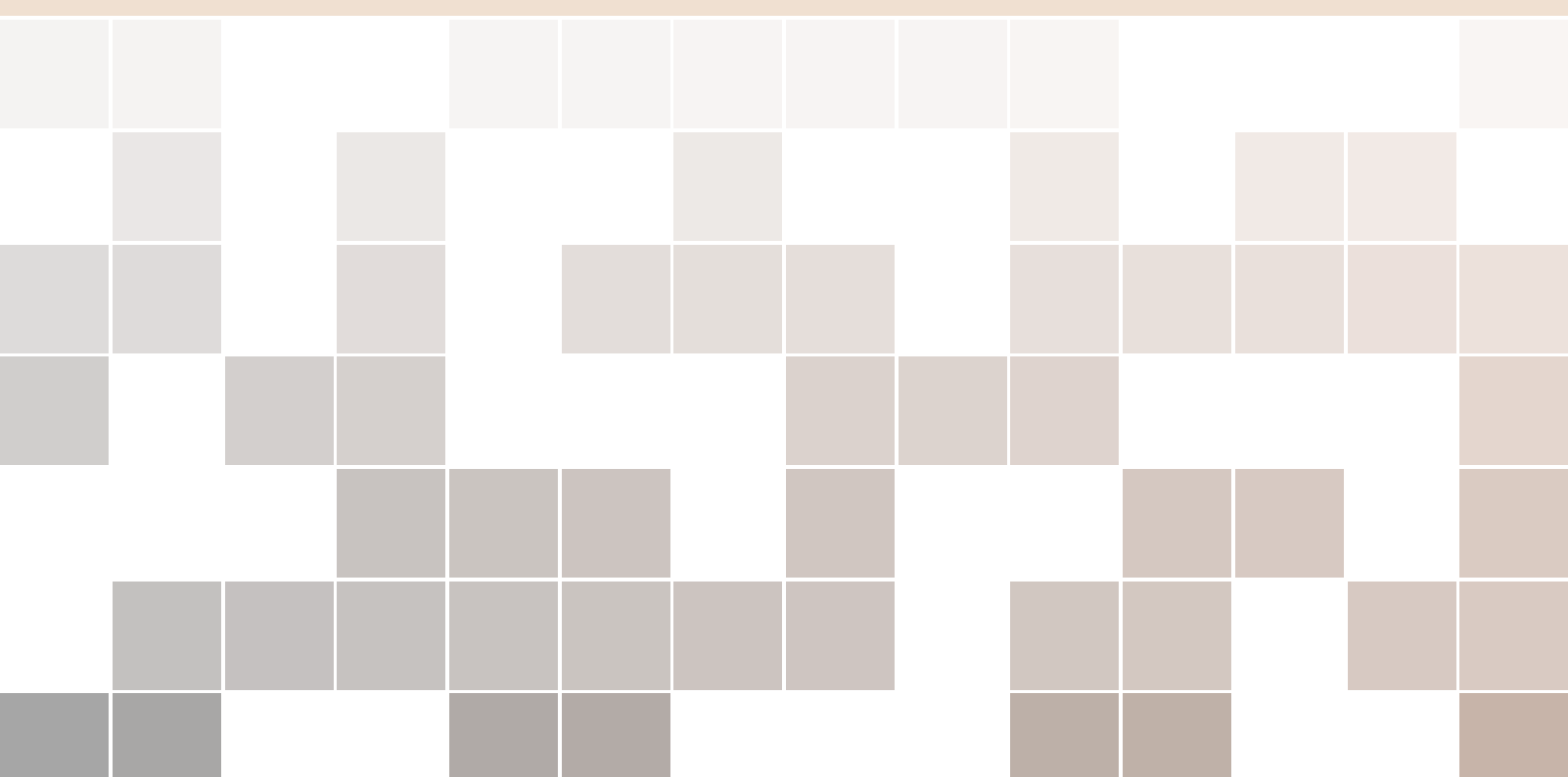


# Base de Datos (75.15 - 75.28 - 95.05)

Dpto. de Computación (Fac. de Ingeniería - UBA)

## Guía de Ejercicios

Docentes: Mariano G. Beiró, Gustavo Dejean, Alberto Fasce, Lucas Román, Yicheng Zhang



## GUIA 1: MODELADO CONCEPTUAL

1. (*AFA - Modelado de datos*) La *Asociación del Fútbol Argentino (AFA)* mantiene una serie de datos históricos acerca de sus jugadores y equipos inscriptos, encuentros y torneos disputados, resultados y goles:
  - Cada jugador asociado a la AFA recibe un número de inscripción, y se registran su apellido y nombres, ciudad y provincia de origen, y fecha de nacimiento.
  - Cada equipo asociado se registra con su nombre, ciudad y provincia en que se encuentra, y fecha de alta como institución. No pueden registrarse dos equipos con idéntico nombre.
  - La AFA organiza una serie de torneos, cada uno compuesto por una serie de fechas que se identifican dentro de cada torneo con números consecutivos. En cada fecha de un torneo se producen una serie de encuentros en que un equipo local enfrenta a un equipo visitante. Los encuentros también se encuentran numerados en forma consecutiva dentro de cada fecha. La AFA mantiene registro de todos los encuentros jugados en cada fecha, indicando qué equipos se enfrentaron, en qué día y horario, y cuál fue el resultado.
  - En cada torneo, cada equipo participante presenta una formación compuesta por una serie de jugadores. Un mismo jugador no puede formar parte de la formación de más de un equipo durante un mismo torneo. Cada jugador inscripto en un torneo es inscripto para jugar en una determinada posición.
  - En cada encuentro se convierte una serie de goles, que se identifican con números consecutivos dentro del encuentro. Para cada gol convertido se registra cuál fue el equipo que lo convirtió, qué jugador lo hizo, y en qué minuto del partido.Proponga un diagrama Entidad-Interrelación adecuado para modelar los datos que mantiene la AFA.

2. (*Obra Social - Modelado de interrelaciones*) La *Superintendencia de Seguros de Salud* mantiene una base de datos con información sobre los *Servicios* que distintas *Obras Sociales* prestan, y a través de qué *Clínicas* los ofrecen. La operatoria se rige por las siguientes restricciones:
  - Cada obra social presta a sus clientes una serie de servicios (p. ej., *Ameba Salud* presta *Cirugía Ocular* entre otros), y tiene convenio con una o más clínicas (p. ej., *Ameba Salud* tiene convenio con *Clínica San Vito* entre otras).
  - A su vez, cada clínica ofrece uno o más servicios (la *Clínica San Vito* ofrece *Neonatología*, por ejemplo) y tiene convenio con una o más obras sociales.
  - Un mismo servicio puede ser ofrecido por una o más clínicas, y ser prestado por una o más obras sociales.
  - Adicionalmente, si una obra social *O* presta un servicio *S*, tiene convenio con una clínica *C*, y la clínica *C* ofrece el servicio *S*, entonces los clientes de *O* podrán recibir el servicio *S* en *C*. En cambio, si *O* no presta el servicio *S*, sus clientes no podrán recibirlo a pesar de que *O* tenga convenio con *C* y *C* ofrezca *S*. De la misma forma, si *O* presta el servicio *S* y tiene convenio con la clínica *C*, pero *C* no ofrece ese servicio, entonces los clientes de *O* no podrán recibir el servicio *S* en *C*.Proponga un diagrama Entidad-Interrelación adecuado para modelar el problema planteado. Utilice para ello los tipos de entidad *Obra Social*, *Clínica* y *Servicio*, y suponga que cada uno de ellos posee únicamente los atributos **código** y **nombre**.

3. (*Aerolínea - Modelado de datos*) Una aerolínea comercial guarda información de los aeropuertos en los que opera, los vuelos que ofrece y los pasajeros que transporta. La aerolínea registra los siguientes datos:

- Para cada aeropuerto: su código IATA de tres letras, su nombre, sus coordenadas de latitud y longitud, la ciudad y el país al que pertenece.
- Para cada vuelo ofrecido: su número de vuelo (ej., 4330), sus aeropuertos de partida, llegada, y aquellos en los que hace escalas intermedias (si es que las hay), y los horarios programados de partida y llegada (cada uno en la hora local del aeropuerto respectivo).
- Cada vuelo se ofrece a lo sumo una vez al día, y la aerolínea denomina viaje a una realización concreta de un vuelo en un día determinado. Para cada viaje se registra la fecha, hora de partida y hora de llegada a destino.
- Se registra también cierta información de cada uno de los pasajeros del viaje. Para cada pasajero se registra su pasaporte (indicando país de emisión y número), apellido, nombres y fecha de nacimiento, y el asiento asignado en el viaje.

Proponga un diagrama Entidad-Interrelación adecuado para modelar el problema planteado.

4. (*Facultad de Ingeniería - Análisis de requerimientos incompletos*) Considere el diseño de una base de datos para la *Facultad de Ingeniería*, que permita llevar el registro de los alumnos, docentes, asignaturas y departamentos, así como de las asignaturas que cada alumno aprobó.

Adicionalmente la base de datos debe satisfacer los siguientes requerimientos:

- Se debe registrar no sólo la aprobación de una asignatura por parte de un alumno, sino también su inscripción a la cursada de la asignatura. Asimismo, se debe considerar que un alumno puede aprobar una asignatura o bien en el contexto de una cursada, o bien en calidad de libre.
- Se debe registrar la institución secundaria de la cual el alumno egresó.
- Cada asignatura dictada pertenece a un departamento, y puede tener correlativas en el mismo u otro departamento.
- Cada asignatura es dictada por un plantel docente.
- Cada departamento tiene un director, que puede o no ser docente.

Proponga un diagrama Entidad-Interrelación adecuado para modelar los datos de la Facultad de Ingeniería.

#### Nota

Este enunciado no hace una enumeración exhaustiva de todos los datos involucrados (entidades, atributos e interrelaciones). Tome sus propias decisiones respecto a los datos y sus restricciones en base a su experiencia en esta facultad.

5. (*Megacciones - Análisis de requerimientos*) *Megacciones S.A.* es un *broker* dedicado a la intermediación de operaciones financieras, que ofrece a sus clientes la posibilidad de comprar y vender acciones en distintas plazas del mundo. El Broker nos ha solicitado realizar el diseño e implementación de una base de datos para registrar, gestionar y consultar las operaciones de compra y venta que realizan sus clientes. En una entrevista personal con el gerente general de la empresa obtenemos la siguiente información sobre los requerimientos:

Los clientes de Megacciones S.A. pueden ser personas físicas o jurídicas. En ambos casos, cuando un cliente se registra debe presentar su identificador fiscal, un domicilio legal en Argentina, y un teléfono y mail de contacto. Asimismo, debe indicar el número de CBU de la cuenta corriente con la cual operará. Opcionalmente se puede ingresar un segundo teléfono de contacto. En el momento de la registración se provee al cliente de un nombre de usuario y una contraseña. Esta última se almacenará en forma segura en la base de datos utilizando un hash.

El Broker opera en el Merval y en dos plazas internacionales: el NYSE y el FWB, aunque no se descarta incorporar nuevas plazas en un futuro cercano. En cada una de ellas se cotizan distintas acciones. Cada acción comercializada en una plaza posee un código de entre 1 y 5 letras, único dentro de la plaza. A su vez, cada plaza se identifica con una sigla y cotiza en una moneda particular. El Merval, el NYSE y el FWB cotizan en pesos argentinos, dólares y euros, respectivamente.

Los clientes accederán al sistema del Broker a través de una interfaz web en la que se loguearán con su usuario y contraseña. Esta interfaz se conectará directamente con la base de datos, y desde ella los clientes podrán solicitar comprar o vender acciones, o visualizar su tenencia de acciones actualizada.

**Ordenes de compra y venta:** Cuando un cliente desee realizar una operación de compra, cargará una orden de compra conteniendo como datos la acción a comprar, la cantidad, y el precio máximo que esté dispuesto a pagar. Para las operaciones de venta se registrará una orden de venta indicando la acción a vender, la cantidad y el precio mínimo exigido. En el momento de registrar una orden, la misma quedará en estado "Procesando". Si la misma lograra ejecutarse durante el día, se registrará como "Ejecutada". Si la orden no lograra ser ejecutada en todo el día, quedará registrada como "Rechazada". Si en algún momento del día el cliente cancelara la operación, la misma se registrará como "Cancelada". Las ordenes ejecutadas implicarán el cobro de una comisión que determinará el sistema de Operaciones y que quedará registrada en la orden.

**Tenencia y valoración:** Cuando una orden se ejecute, se actualizará la tenencia de acciones del cliente, incorporando o eliminando la cantidad de acciones que acaba de comprar o vender. Asimismo, deberá quedar registrado el precio efectivo de compra/venta, teniendo en cuenta que la cantidad total comprada/vendida puede provenir de uno o más lotes con precios ligeramente distintos. Se deberá tener en cuenta que al Broker le interesará hacer un seguimiento de los activos del cliente en el tiempo para analizar, por ejemplo, cómo evolucionó su tenencia de una acción particular o la valoración total de sus activos financieros hasta una fecha determinada. Es por ello que no deberá perderse el rastro de las operaciones realizadas.

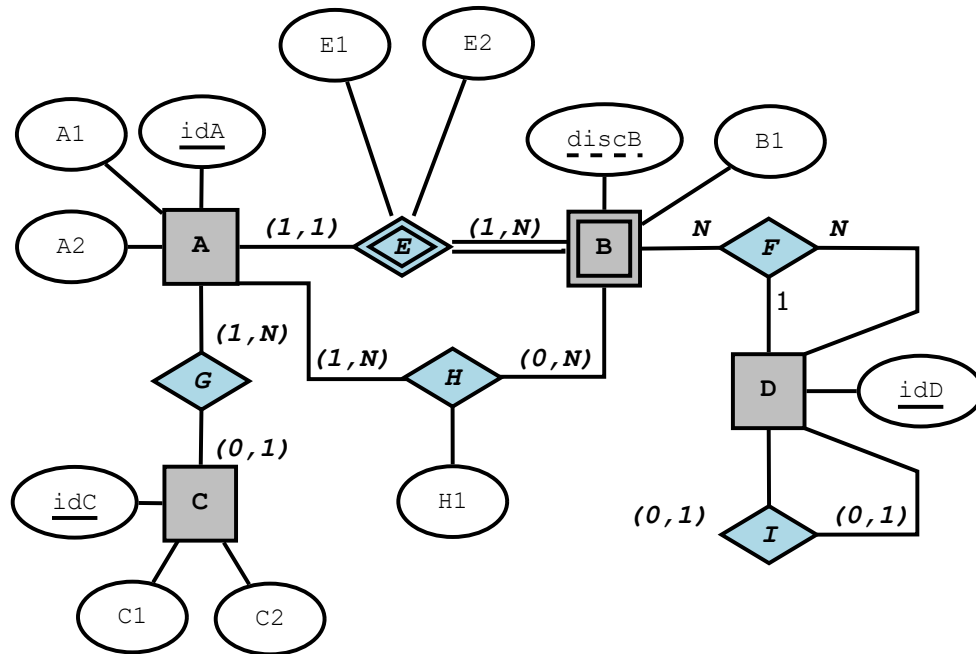
La valoración de los activos de los clientes se realizará consultando periódicamente la cotización de las distintas acciones, y almacenando todas estas cotizaciones históricas en una tabla.

**Interacción con sistemas externos:** La actualización de la tabla de cotizaciones se realizará a través de un subsistema que interactúa con los sistemas de las distintas plazas. Esta interacción no deberá ser considerada por nosotros. Este mismo subsistema se ocupará del cobro de las operaciones de compra y el pago de las operaciones de venta, incluidas las comisiones, interactuando con el sistema bancario. Asimismo, las órdenes de compra/venta serán ejecutadas por un subsistema interactúa con los sistemas de las distintas plazas, y que utilizará nuestra base de datos como soporte. Esta interacción no deberá ser considerada por nosotros. Este mismo subsistema se ocupará de actualizar la tabla de cotizaciones en forma periódica.

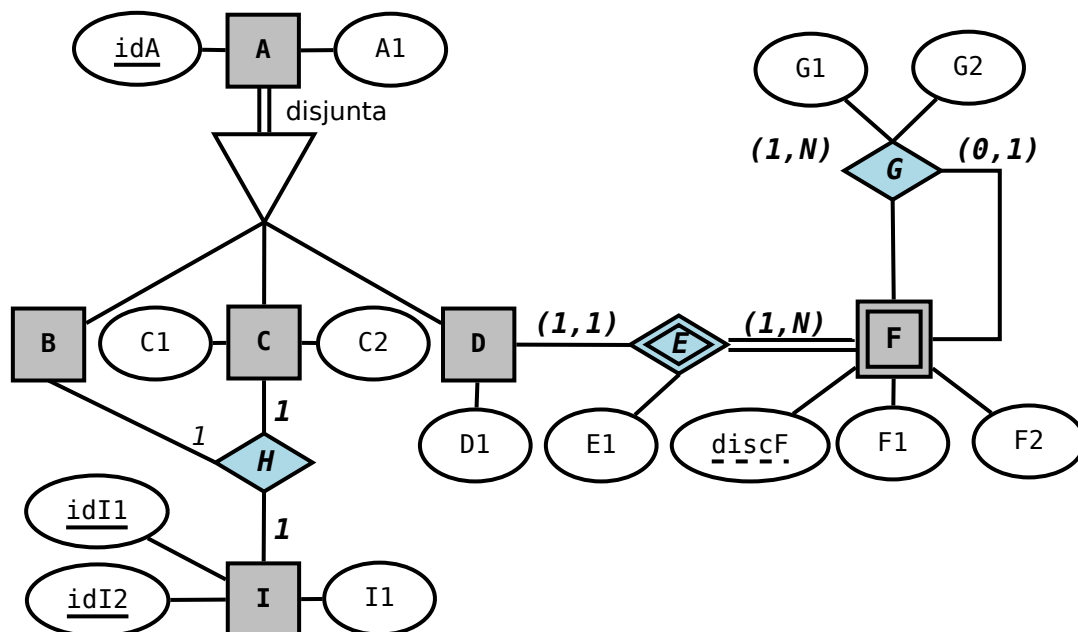
## GUIA 2: MODELO RELACIONAL

1. (*Pasaje de modelo*) Para los siguientes diagramas Entidad-Interrelación, realice el pasaje al modelo relacional indicando para cada relación cuáles son las claves primarias, claves candidatas, claves foráneas y atributos descriptivos.

a)



b)



2. (*Pasaje de modelo*) Dadas las siguientes relaciones pertenecientes a un sistema de reserva de alojamientos, proponga un diagrama Entidad-Interrelación adecuado para modelar la estructura de datos.
- Habitación(cod\_habitación, cantidad\_personas)
  - Hotel(cod\_hotel, cantidad\_estrellas, calle, altura, ciudad, país)
  - CuartoHotel(cod\_habitación, cod\_hotel, nro\_habitación)
  - Departamento(cod\_habitación, metros\_cuadrados, calle, altura, ciudad, país)
  - Servicios(cod\_habitación, nombre\_servicio)
  - Disponibilidad(cod\_habitación, fecha)
  - Reserva(cod\_reserva, cod\_cliente, cod\_habitación, fecha\_in, fecha\_out)
  - Clientes(cod\_cliente, apellido, nombre, correo\_electrónico, tarjeta\_crédito)
3. (*Restricciones de integridad*) Dadas dos relaciones  $R(\underline{A_1}, A_2, \dots, A_n)$  y  $S(\underline{B_1}, \underline{B_2}, \dots, B_m)$ , en donde  $B_2$  hace referencia a  $R$ , indique si las siguientes afirmaciones son verdaderas ó falsas, justificando cada una de sus respuestas.
- a) La regla de integridad referencial exige que  $\forall s \in S : s[B_2] \neq \text{NULL}$ .
  - b) La regla de integridad de entidad exige que  $\forall r \in R : r[A_1] \neq \text{NULL}$ .
  - c) La regla de integridad referencial exige que  $\forall r \in R : \exists s \in S / s[B_2] = r[A_1]$ .
  - d) La regla de integridad de entidad exige que  $\forall s \in S : s[B_2] \neq \text{NULL}$ .
  - e) La regla de integridad referencial exige que  $\forall s \in S : \exists r \in R / s[B_2] = r[A_1]$ .
  - f) La regla de unicidad exige que  $\forall s_1, s_2 \in S, s_1 \neq s_2 : s_1[B_2] \neq s_2[B_2]$ .
4. (*Integridad referencial en SQL*) Una compañía de seguros guarda información sobre sus clientes, las pólizas que poseen y los incidentes que las mismas cubren. El esquema de base de datos relacional fue creado a través de las siguientes consultas SQL, siendo su estado actual el descrito en el Cuadro 1:

```
CREATE TABLE Clientes (
    legajo INT PRIMARY KEY,
    nombre VARCHAR(30),
    fecha_alta DATE,
    email VARCHAR(30)
);

CREATE TABLE Pólizas (
    nro_poliza INT PRIMARY KEY,
    legajo_cliente INT,
    patente VARCHAR(10),
    FOREIGN KEY (legajo_cliente)
    REFERENCES Clientes(legajo) ON DELETE RESTRICT
    ON UPDATE CASCADE
);

CREATE TABLE Coberturas (
    nro_poliza INT,
    incidente VARCHAR(20),
    PRIMARY KEY(nro_poliza, incidente),
    FOREIGN KEY (nro_poliza)
    REFERENCES Pólizas(nro_poliza) ON DELETE CASCADE
    ON UPDATE CASCADE
);
```

**CLIENTES**

legajo	nombre	fecha_alta	email
271490	Ángeles Hudson	17-01-2011	ahudson@mimail.com
287315	Rosario Sierra	26-08-2013	sierarr@outmail.com
372088	Mario García	12-07-2016	mgarcia@abase.com
380040	Úrsula Gómez	03-12-2017	ursu_15_g@bmail.com.ar

**POLIZAS**

nro_poliza	legajo_cliente	patente
152703	271490	MHL316
192041	271490	FZU570
130515	287315	AA030MK
613219	372088	DAB918
908821	380040	AB218CB

**COBERTURAS**

nro_poliza	incidente
152703	TERCEROS
192041	TERCEROS
130515	TERCEROS
130515	ROBO
613219	TERCEROS
908821	TERCEROS
908821	ROBO
908821	INCENDIO

---

CUADRO 1. Instancias de relación para el Ejercicio 4 (Integridad referencial en SQL).

- a) Para cada una de las consultas siguientes, indique si podrá ejecutarse exitosamente o será restringida por el SGBD. Tenga en cuenta el estado actual de la base de datos mostrado en el Cuadro 1 y las restricciones de integridad definidas a través de SQL.
- 1) DELETE FROM Polizas WHERE legajo\_cliente=271490;
  - 2) DELETE FROM Coberturas WHERE nro\_poliza=130515;
  - 3) UPDATE Polizas SET legajo\_cliente=391996 WHERE legajo\_cliente=372088;
  - 4) UPDATE Clientes SET legajo=391996 WHERE legajo=372088;
  - 5) DELETE FROM Clientes WHERE legajo=380040;
- b) Ejecute aquellas consultas que no serán restringidas, y muestre el resultado final en formato de tablas, en base al estado actual mostrado en el Cuadro 1.

### GUIA 3: ÁLGEBRA Y CÁLCULO RELACIONALES

1. (*Álgebra relacional básica*) Considerando los siguientes esquemas de relaciones:

- Artículos(nro\_art, nombre\_art, color)
- Proveedores(nro\_prov, apellido)
- Pedidos(nro\_prov, nro\_art, cantidad)

Traduzca las siguientes consultas al lenguaje del *Álgebra Relacional*:

- a) Obtener los números de aquellos proveedores que suministran el producto 34.
  - b) Obtener el número y el apellido de aquellos proveedores que suministran el producto 34 o el producto 12.
  - c) Obtener los números de los productos que son tuercas o que son suministrados por el proveedor número 328.
  - d) Obtener los números de los productos que son tuercas y que son suministrados por el proveedor número 328.
  - e) Obtener los apellidos de aquellos proveedores que suministran al menos un producto que no sea el producto 33.
  - f) Obtener los apellidos de aquellos proveedores que no suministran el producto 33.
  - g) Obtener los apellidos de aquellos proveedores que suministran al menos un producto y que no suministran el producto 33.
  - h) Obtener el número y el apellido de aquellos proveedores que suministran más de 100 tuercas en algún pedido.
  - i) Obtener los números de aquellos proveedores que suministran todos los productos.
  - j) Obtener los apellidos de aquellos proveedores que suministran todos los productos menos el 12.
  - k) Obtener los apellidos de aquellos proveedores que suministran, al menos, todos los productos suministrados por el proveedor número 128.
  - l) Obtener los números de aquellos proveedores que suministran todos los productos, y en igual cantidad (es decir, que no suministran dos productos en cantidades distintas).
2. (*Cálculo Relacional de Tuplas*) Samuel invitó a algunos amigos a su casa para disfrutar de la piscina y refrescarse con algunos *cocktails*. Para prepararse descargó de Internet unas tablas que detallan la composición de 1000 *cocktails* distintos, indicando el nombre de cada *cocktail* y los ingredientes que lo componen. A su vez, agregó a la tabla de ingredientes un atributo *booleano* que indica si el ingrediente se encuentra en su alacena ó no:

- Cocktails(nombre\_cocktail, tamaño\_ml, instructivo)
- Ingredientes(nombre\_ingredient, descripción, en\_alacena)
- Composiciones(nombre\_cocktail, nombre\_ingredient, cantidad)

Escriba una consulta en *Cálculo Relacional de Tuplas* que devuelva los nombres de aquellos *cocktails* que pueden ser preparados con los ingredientes que Samuel tiene en la alacena.

Nota: Para los ingredientes que sí se encuentran en la alacena, suponga que Samuel tiene cantidad suficiente como para preparar cualquier cantidad de *cocktails*.



3. (*C.R.T. y álgebra relacional*) Dados los esquemas de relación:

- $\text{Platos}(\underline{\text{cod\_plato}}, \text{nombre\_plato}, \text{calorias}, \text{precio})$
- $\text{Composiciones}(\underline{\text{cod\_plato}}, \underline{\text{ingred}})$

que reflejan los ingredientes que componen los platos de un restaurante, se realiza la siguiente consulta escrita en el lenguaje del Cálculo Relacional de Tuplas:

$$\{ p_1.\text{nombre\_plato}, p_1.\text{precio} \mid \text{Platos}(p_1) \wedge (\nexists c_1)(\text{Composiciones}(c_1) \wedge c_1.\text{cod\_plato} = p_1.\text{cod\_plato} \wedge c_1.\text{ingred} = \text{'QUESO'}) \wedge (\nexists p_2)(\text{Platos}(p_2) \wedge p_2.\text{calorias} < p_1.\text{calorias} \wedge (\nexists c_2)(\text{Composiciones}(c_2) \wedge c_2.\text{cod\_plato} = p_2.\text{cod\_plato} \wedge c_2.\text{ingred} = \text{'QUESO'})) \}$$

Se pide:

- a) Explique en palabras qué hace esta consulta.
- b) Para las instancias de relación de *Platos* y *Composiciones* que se muestran en formato de tabla en el *Cuadro 1*, muestre –también en formato de tabla– cuál será la relación resultante al ejecutar la consulta.
- c) Traduzca la consulta al lenguaje del *Álgebra Relacional*.

**PLATOS**

cod_plato	nombre_plato	calorías	precio
NP01	Pizza margherita	830	130,00
NP02	Lomo a la pimienta	950	178,00
NP03	Ensalada Waldorf	590	145,00
NP04	Tarta de brócoli	490	110,00
NP05	Suprema a la suiza	830	230,00
NP06	Gambas al ajillo	590	170,00

**COMPOSICIONES**

cod_plato	ingred
NP01	HARINA
NP01	TOMATE
NP01	QUESO
NP01	ALBAHACA
NP02	CARNE
NP02	VINO
NP02	PIMIENTA
NP03	MAYONESA
NP03	LECHUGA
NP03	APIO
NP03	MANZANA
NP03	NUEZ

(...continuación)

cod_plato	ingred
NP04	HARINA
NP04	BROCOLI
NP04	HUEVO
NP04	CREMA
NP04	QUESO
NP05	POLLO
NP05	PAN RALLADO
NP05	QUESO
NP05	HUEVO
NP06	GAMBAS
NP06	AJO
NP06	PEREJIL

CUADRO 1. Instancias de relación para el Ejercicio 3.

4. (*Potencia expresiva*) Dadas las siguientes relaciones que almacenan información sobre tenistas, torneos y los tenistas vencedores de cada uno:

- Tenistas(nombre\_tenista, país, altura, diestro)
- Torneos(nombre\_torneo, tipo\_torneo)
- Campeones(nombre\_tenista, nombre\_torneo, modalidad, año)

Para cada una de las siguientes consultas, indique: (i) si la misma es expresable en el lenguaje del *Cálculo Relacional de Tuplas (CRT)*, (ii) si es expresable en el lenguaje del *Álgebra Relacional* con los operadores básicos, y (iii) si es expresable en lenguaje *SQL*.

- a) 'Listar los nombres de los tenistas que ganaron al menos 5 torneos de tipo *Grand Slam*'.
- b) 'Listar para cada tenista su nombre y la cantidad de torneos de tipo *Grand Slam* que ganó'.

Nota: No es necesario que traduzca las consultas a ninguno de estos lenguajes.

5. (*Álgebra relacional básica*) Dadas las siguientes relaciones que almacenan información sobre llamadas telefónicas:

- Clientes(CUIT, apellido, nombre, domicilio)
- Líneas(número\_línea, CUIT\_titular)
- Llamadas(número\_línea\_origen, número\_línea\_destino, fecha, duración)

Traduzca las siguientes consultas al lenguaje del *Álgebra Relacional*:

- a) Obtener el CUIT, apellido y nombre de los clientes que participaron de la/s llamada/s de mayor duración.
- b) Obtener el CUIT, apellido y nombre de los clientes que poseen más de una línea.
- c) Obtener el CUIT, apellido y nombre de los clientes que mantuvieron al menos una llamada con todas las líneas del cliente cuyo CUIT es '27-40315228-3'.

## GUIA 4: STRUCTURED QUERY LANGUAGE (SQL)

1. (*NBA*) Kobe Bryant ha sido uno de los más grandes basquetbolistas de todos los tiempos. Accederemos a una base de datos de la NBA para extraer algunos datos sobresalientes acerca de su carrera.

A continuación se muestran las tablas de las que disponemos, y que contienen información sobre los equipos de la NBA, los partidos disputados en cada temporada, los equipos de los que cada jugador formó parte, y las acciones realizadas por cada jugador en cada partido que jugó (cantidad de puntos anotados, minutos jugados, etcétera):

- Seasons(name, starting\_date, end\_date)
- Teams(name, city, conference)
- Matches(match\_id, local\_team, visiting\_team, date, stage\_name, season\_name)
- Players(name, birth\_date, height)
- TeamsCompositions(player\_name, season\_name, team\_name)
- Actions(player\_name, match\_id, minutes, points, rebounds, steals, blocks)

Nota: Por simplicidad se asume que los jugadores no cambian de equipo en medio de la temporada.

Para cada ítem indicado a continuación, escriba una consulta SQL que permita encontrar la respuesta a partir de la información contenida en las tablas anteriores. Sólo a modo de cultura general, podrá encontrar la respuesta a cada estadística girando la página.

- a) Encuentre la cantidad de partidos en que Kobe Bryant marcó al menos 50 puntos, devolviendo únicamente dicha cantidad.

Rta: Lo hizo en **25** partidos (siendo únicamente superado por Wilt Chamberlain en 18 ocasiones- y Michael Jordan en 31 ocasiones-).

- b) Kobe Bryant llegó a convertir 5640 puntos en los *playoffs*. Encuentre a aquellos jugadores que hayan superado la marca de 5640 puntos, indicando para cada uno su nombre y la cantidad de puntos que convirtió en *playoffs*.

Nota: Los *playoffs* son una etapa de la temporada. La etapa a la que corresponde cada partido se indica bajo el atributo **stage**.

Rta: Kobe Bryant fue únicamente superado por **LeBron James** (6911 puntos), **Michael Jordan** (5987 puntos) y **Kareem Abdul-Jabbar** (5762 puntos).

- c) Encuentre la cantidad de puntos que marcó Kobe Bryant en el último partido oficial que jugó, indicando el nombre del equipo local, el nombre del equipo visitante y la cantidad de puntos marcados por Kobe.

Rta: En su último partido oficial, que enfrentó a **Los Angeles Lakers** contra los **Utah Jazz**, Kobe Bryant señaló **60** puntos (era la séptima vez que alcanzaba dicha marca).

2. (*Fórmula 1*) Considere las siguientes tablas que almacenan información sobre los distintos circuitos de Fórmula 1<sup>®</sup>, las carreras que se corrieron en cada uno de ellos a lo largo de los años, los pilotos que han participado de cada carrera y las escuderías a las que pertenecían, y el tiempo que cada piloto ha marcado en cada vuelta que logró terminar:

- Circuitos(nombre\_circuito, ciudad, país)
- Carreras(nombre\_circuito, temporada)
- Pilotos(nombre\_piloto, fecha\_nacimiento, nacionalidad)
- Participación(nombre\_piloto, nombre\_circuito, temporada)
- EscuderíasPilotos(nombre\_piloto, temporada, escudería)
- Timings(nombre\_piloto, nombre\_circuito, temporada, nro\_vuelta, tiempo)

Escriba una consulta SQL que encuentre para cada circuito cuál fue la escudería que logró marcar el tiempo de vuelta histórico más corto, indicando el nombre del circuito y el nombre de la escudería que posee dicho récord de tiempo de vuelta. Si varias escuderías ostentan el mismo récord, devuelva una línea por cada una, pero no devuelva más de una vez a la misma escudería para un mismo circuito.

3. (*Consultas - Investigadores*) Considerando los siguientes esquemas de relaciones:

- Facultades(código, nombre)
- Investigadores(DNI, nombre, facultad)
- Reservas(DNI, num\_serie, comienzo, fin)
- Equipos(num\_serie, nombre, facultad)

Traduzca las siguientes consultas al lenguaje SQL:

- a) Obtener el DNI y nombre de aquellos investigadores que han realizado más de una reserva.
- b) Obtener un listado completa de reservas, incluyendo los siguientes datos:
  - DNI y nombre del investigador, junto con el nombre de su facultad.
  - Número de serie y nombre del equipo reservado, junto con el nombre de la facultad a la que pertenece.
  - Fecha de comienzo y fin de la reserva.
- c) Obtener el DNI y el nombre de los investigadores que han reservado equipos que no son de su facultad.
- d) Obtener los nombres de las facultades en las que ningún investigador ha realizado una reserva.
- e) Obtener los nombres de las facultades con investigadores ‘ociosos’ (investigadores que no han realizado ninguna reserva).
- f) Obtener el número de serie y nombre de los equipos que nunca han sido reservados.

$$A = \begin{bmatrix} 0 & 1 & 2 \\ 1 & 0 & 4 \\ 0 & 1 & 0 \end{bmatrix} \quad B = A^2 = \begin{bmatrix} 1 & 2 & 4 \\ 0 & 5 & 2 \\ 1 & 0 & 4 \end{bmatrix}$$

A			B		
i	j	v	i	j	v
1	2	1	1	1	1
1	3	2	1	2	2
2	1	1	1	3	4
2	3	4	2	2	5
3	2	1	2	3	2
			3	1	1
			3	3	4

CUADRO 1. Ejemplo correspondiente al Ejercicio 5. Matrices  $A$  y  $B$  (arriba) y su representación como tabla (abajo).

4. (*Histograma*) Una heladería almacena información sobre sus clientes, las ventas que realiza, y los gustos pedidos en cada venta.

- **Clientes**(nro\_cliente, nombre, domicilio)
- **Ventas**(nro\_factura, fecha, nro\_cliente, monto)
- **Degustación**(nro\_factura, gusto)

A la heladería le interesa saber si sus clientes son propensos a probar gustos variados, o si en general se aferran cada uno a unos pocos gustos. Para ello se desea construir un histograma que indique para cada número entero positivo  $i$  la cantidad de clientes que han probado  $i$  gustos distintos desde el 1 de enero de este año. Se pide:

- a) Escriba una consulta en *SQL* que construya una vista **NroGustosPorCliente**(nro\_cliente, nro\_gustos) que para cada cliente que probó algún gusto desde el 1 de enero de 2017 devuelva el número de cliente y la cantidad de gustos distintos que probó.
- b) A partir de la vista anterior, escriba una consulta en *SQL* que devuelva, para cada número entero positivo  $i$ , la cantidad de clientes que probó esa cantidad de gustos distintos desde el 1 de enero de 2017.

Nota: Omitiremos los  $i$  para los cuales la cantidad de clientes es cero.

5. (*Suma de matrices ralas*) Una base de datos guarda una matriz cuadrada rala  $A$  de grandes dimensiones a través de una tabla  $A(\underline{i}, \underline{j}, v)$ , en donde los atributos  $i$  y  $j$  identifican al elemento  $(i, j)$  de la matriz, mientras que  $v$  representa su valor,  $a_{ij}$ . La tabla sólo almacena aquellas posiciones para las cuales el valor  $a_{ij}$  es no nulo.

Escriba una consulta en lenguaje *SQL* que devuelva como resultado una tabla  $B(\underline{i}, \underline{j}, v)$  que represente, con el mismo formato, a la matriz  $B$  que resulta de elevar  $A$  al cuadrado (es decir,  $B = A^2 = A \cdot A$ ). Sólo almacene aquellas posiciones para las cuales el valor  $b_{ij}$  es no nulo. En el Cuadro 1 encontrará un pequeño ejemplo ilustrativo.

Nota: Recuerde que los elementos de la matriz  $B = A^2$  se calculan como  $b_{ij} = \sum_k a_{ik} \cdot a_{kj}$ .

6. (*Club de golf*) Un club de golf mantiene en una base de datos un registro de todos sus hoyos, las personas afiliadas al club, las distintas partidas jugadas entre ellas y la puntuación de cada jugador para cada hoyo en cada una de las partidas:

- Hoyos(nro\_hoyo, hándicap, par)
- Socios(nro\_socio, apellido, nombre)
- Partidas(cod\_partida, fecha, hora)
- JugadoresPartidas(cod\_partida, nro\_socio)
- Puntuaciones(cod\_partida, nro\_hoyo, nro\_socio, puntuación)

Escriba una consulta SQL que liste el número de socio, apellido y nombre de las personas que hayan logrado alguna vez realizar al menos 3 birdies en una misma partida. Se dice que una persona realiza un *birdie* cuando obtiene una puntuación 1 unidad menor al par del hoyo (por ejemplo, si un hoyo tiene par 4 y el jugador lo completa en 3 golpes).

*Nota:* En una *partida* de golf pueden jugar entre dos y cuatro personas, cada una de las cuales deberá completar una cantidad de *hoyos* (generalmente 9 ó 18). Cada hoyo de un campo de golf posee un *par* predeterminado –que es la cantidad de golpes esperada en que un profesional metería la pelota en dicho hoyo– y un *hándicap* predeterminado que mide la dificultad. Para cada persona y cada hoyo se registra la *puntuación* obtenida en la partida, definida como la cantidad de golpes realizados por la persona hasta meter la pelota en ese hoyo.

7. (*Subconsultas correlacionadas*) Dadas las siguientes relaciones que guardan información sobre los alumnos de una facultad y sus calificaciones:

- Alumnos(padrón, apellido, nombre)
- Notas(padrón, materia, fecha, nota)

Considere la siguiente consulta SQL que busca para cada alumno su mejor nota histórica:

```
SELECT DISTINCT a.padron, a.apellido, a.nombre, n.nota as mejor_nota
FROM Alumnos a, Notas n
WHERE a.padron = n.padron
AND n.nota >= ALL (SELECT n2.nota
                  FROM Alumnos a2, Notas n2
                  WHERE a2.padron=a.padron AND a2.padron=n2.padron);
```

Esta consulta incluye una subconsulta *correlacionada* (es decir, que hace referencia a atributos de las tablas de la consulta externa). Típicamente la existencia de una subconsulta correlacionada genera un mayor costo de procesamiento porque la misma debe reejecutarse para cada una de las tuplas obtenidas en la consulta externa, a menos que el motor de base de datos logre optimizar la ejecución para evitarlo.

Escriba una consulta SQL equivalente a la anterior, que no contenga subconsultas correlacionadas.

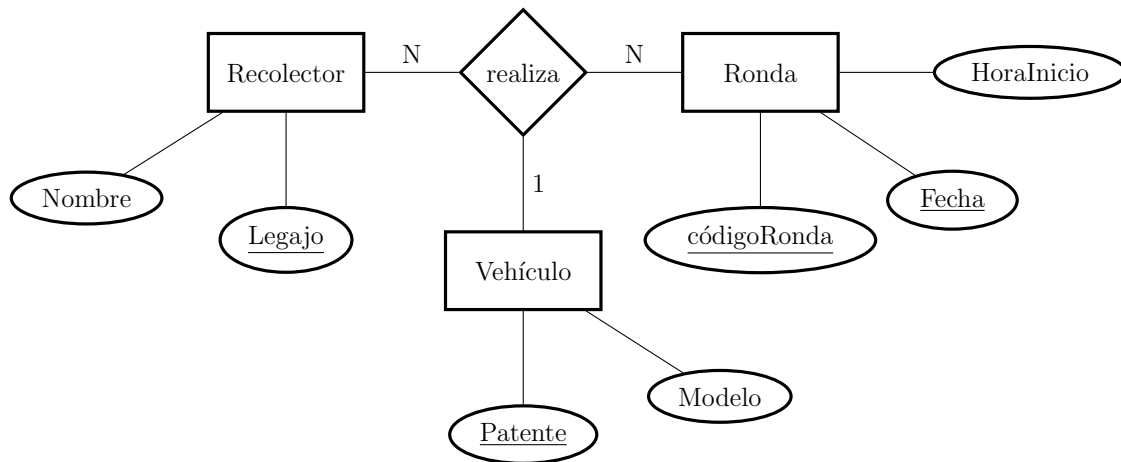
## GUIA 5: DISEÑO RELACIONAL

1. (*Clausura de conjuntos de atributos y de conjuntos de dependencias*) Dado el esquema de relación  $R(A, B, C, D, E)$  y el siguiente conjunto de dependencias funcionales asociado  $F = \{AB \rightarrow D, CD \rightarrow E, A \rightarrow C, D \rightarrow B, BC \rightarrow A\}$ , indique si cada una de las siguientes dependencias funcionales pertenece a la clausura de  $F$ .
  - a)  $BC \rightarrow AD$
  - b)  $BD \rightarrow DC$
  - c)  $CA \rightarrow B$
  - d)  $BC \rightarrow DE$
2. (*Diseño conceptual y relacional*) Una firma de agentes de bolsa ofrece a sus clientes la posibilidad de invertir en acciones. La firma está compuesta por varios agentes de bolsa (B) y tiene varias oficinas (O). Cada agente de bolsa opera en a lo sumo una oficina. Cada cliente inversor (I) es gestionado por un y sólo un agente de bolsa, y puede poseer acciones de muchas compañías (A). La firma almacena para cada inversor la cantidad de acciones (Q) que posee de una compañía dada, y almacena también para cada acción de una compañía su cotización actual (P) y el monto de dividendos anuales (D) que la misma paga. Se pide:
  - a) Escriba un conjunto de dependencias funcionales  $F$  que represente todas las restricciones de la empresa.
  - b) A partir del conjunto  $F$  definido, encuentre las claves candidatas de la relación universal  $R(B, O, I, A, Q, P, D)$ .
  - c) Utilizando el algoritmo correspondiente, encuentre una descomposición de  $R$  en FNBC sin pérdida de información. Indique si se preservaron todas las dependencias funcionales.
  - d) Considere la descomposición de  $R$  en  $R_1(I, A, Q, P, D)$  y  $R_2(I, B, O)$ . Indique en qué forma normal se encuentra cada una de estas dos relaciones, si se preserva la información, y si se preservan todas las dependencias funcionales.
3. (*Claves candidatas y FNBC*) Dado el esquema de relación  $R(A, B, C, D, E)$  y el siguiente conjunto de dependencias funcionales asociado  $F = \{A \rightarrow BC, CD \rightarrow E, B \rightarrow D, E \rightarrow A\}$ :
  - a) Halle todas las claves candidatas de  $R$ .
  - b) Indique si la descomposición de  $R$  en  $R_1(A, B, C)$  y  $R_2(C, D, E)$  preserva la información, y si preserva todas las dependencias funcionales.
  - c) Encuentre una descomposición de  $R$  en FNBC sin pérdida de información. Indique si se preservaron todas las dependencias funcionales.
4. (*Conjuntos minimales y 3FN*) Dado el esquema  $R(A, B, C, D, E, F)$  y el siguiente conjunto de dependencias funcionales asociado  $F = \{A \rightarrow BC, B \rightarrow C, C \rightarrow A, AD \rightarrow E, AE \rightarrow F, CD \rightarrow E\}$ .
  - a) Halle un cubrimiento minimal  $F_{\min}$  de  $F$ .
  - b) Encuentre todas las claves candidatas de  $R$ .
  - c) Descomponga la relación  $R$  a 3FN usando el algoritmo correspondiente. Indique si la descomposición obtenida se encuentra en FNBC.

5. (Diseño conceptual y relacional) Una empresa de recolección de residuos organiza sus rondas diarias de recolección de la siguiente manera:

- Cada *Ronda* se identifica con un código y la fecha, y tiene además una hora de inicio. Para cada *Ronda* se asigna un mínimo de 3 *Recolectores* y un y sólo un *Vehículo*.
- Un *Recolector* se identifica con su legajo. Un *Recolector* puede participar de muchas rondas, inclusive en un mismo día.
- Un *Vehículo* se identifica con su patente. Un *Vehículo* puede participar de muchas rondas, inclusive en un mismo día.

Basado en estas reglas, un diseñador de base de datos propone el siguiente diagrama Entidad-Interrelación para modelar la situación:



Y luego traduce el diagrama al siguiente *esquema de base de datos relacional*:

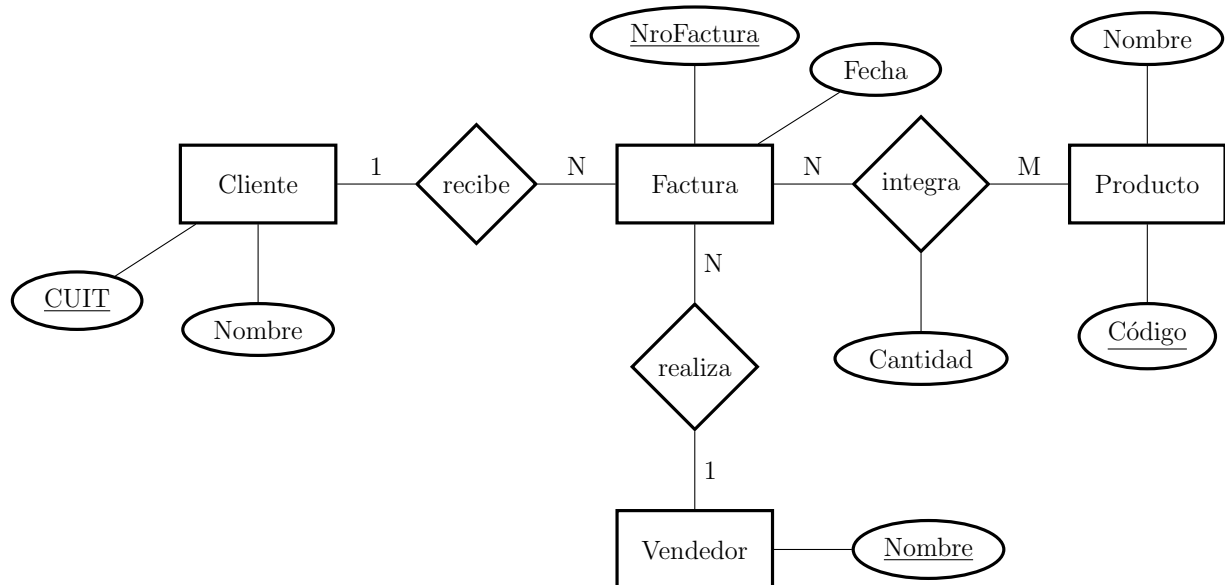
- Recolectores(legajo, nombre)
- Rondas(código\_ronda, fecha, hora\_inicio)
- Vehiculos(patente, modelo)
- Realiza(legajo, código\_ronda, fecha, patente)

Se pide:

- a) Encuentre un conjunto de dependencias funcionales en la relación universal  $U(\text{legajo}, \text{nombre}, \text{patente}, \text{modelo}, \text{código\_ronda}, \text{fecha}, \text{hora\_inicio})$  que represente todas las restricciones de la empresa.
- b) Projete este conjunto de dependencias en cada una de las 4 relaciones del esquema.  
¿Se perdió alguna dependencia?
- c) Muestre que el esquema no se encuentra en FNBC.
- d) Proponga una descomposición a FNBC del esquema dado.
- e) Explique cuál fue el error conceptual del diseñador que derivó en un diseño lógico con redundancia.
- f) Proponga un diagrama Entidad-Interrelación más adecuado.

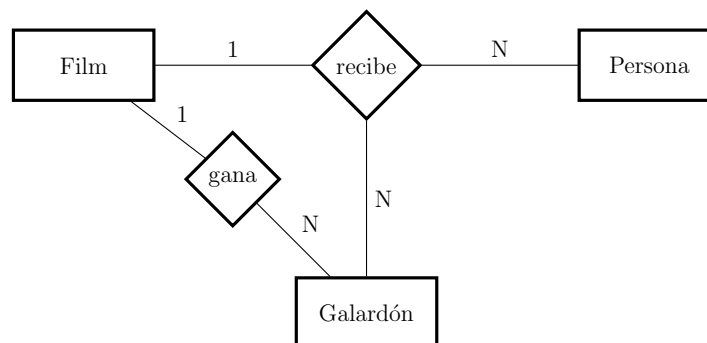


6. (*Diseño conceptual y relacional*) El siguiente diagrama Entidad-Interrelación registra las ventas de un supermercado, identificando a los clientes, los productos vendidos en cada factura y el vendedor que realizó la operación:



Se pide:

- Considerando la relación universal  $U(\text{CUIT}, \text{nombre\_cliente}, \text{nro\_factura}, \text{fecha}, \text{nombre\_vendedor}, \text{cod\_producto}, \text{nombre\_producto}, \text{cantidad})$ , proponga un conjunto de dependencias funcionales  $F$  que capture todas las dependencias funcionales existentes de acuerdo al diagrama Entidad-Interrelación presentado.
  - Aplicando el algoritmo de descomposición a Forma Normal Boyce-Codd (FNBC), halle una descomposición a FNBC del esquema universal dado.
  - Indique si la descomposición que obtuvo preserva todas las dependencias funcionales halladas en el punto a). Justifique su respuesta.
  - El esquema construido, ¿se encuentra en Cuarta Forma Normal (4FN)? Justifique su respuesta.
7. (*Diseño relacional*) Considere el siguiente diagrama Entidad-Interrelación que representa los galardonados en la última entrega de *Premios Oscar*:



En la misma, cada *Galardón* ( $G$ ) fue obtenido por un y sólo un *Film* ( $F$ ). Algunos de estos *Galardones* son recibidos por una o más *Personas* ( $P$ ) (una actriz, el/los director/es, el/los guionista/s, etc.) por un *Film* en particular, mientras que otros se entregan directamente al *Film* (ej., Mejor Película). El diagrama intenta representar esta situación a través de dos interrelaciones: una binaria que vincula al *Film* con el *Galardón*, y una ternaria que vincula al *Film*, el *Galardón* y la *Persona*.

En términos de diseño relacional, este diagrama indicaría que  $G \rightarrow F$  y que  $GP \rightarrow F$  y nos sugiere construir dos tablas para representar la información contenida en las interrelaciones. Hemos consultado a algunas personas y nos han dado opiniones variadas:

*“La interrelación binaria debería ser eliminada porque puede deducirse de la ternaria. Es redundante. Dejen sólo la ternaria con la dependencia funcional  $GP \rightarrow F$  y generen una sola tabla. Cuando un Galardón se entrega directamente al Film, pongan **NULL** en la columna correspondiente a la Persona.”*

*“Dejar la interrelación ternaria no está bien, porque nos obliga a repetir el Film galardonado por cada persona que recibió el mismo Galardón. Esto es redundante. Dejen una interrelación binaria entre Galardón y Film, y dibujen una binaria entre Film y Persona para representar a todos los miembros de cada Film. Eliminen la interrelación ternaria”.*

¿Alguna de estas dos posturas le resulta convincente? Utilice sus conocimientos de diseño relacional para analizar si es necesario mejorar el diagrama de manera de disminuir la redundancia. Luego indique cómo traduciría al modelo relacional el diagrama que escogió, indicando las claves primarias y foráneas existentes en cada relación.

Nota: Por simplicidad, asuma en este ejercicio que cada una de las entidades se identifica con su nombre. Ej.: para *Galardón*, `nombre_galardón`.

## GUIA 6: CONCURRENCIA, TRANSACCIONES Y RECUPERACIÓN

1. (*Serializabilidad y recuperabilidad*) Considere el siguiente solapamiento de dos transacciones  $T_1$  y  $T_2$  que operan sobre los ítems de datos  $X$  e  $Y$ .

$b_{T_1}; R_{T_1}(X); b_{T_2}; W_{T_1}(X); R_{T_2}(X); R_{T_1}(Y); W_{T_2}(X); W_{T_1}(Y); c_{T_1}; c_{T_2};$

- a) Construya el grafo de precedencias para este solapamiento e indique si el mismo es serializable.
  - b) Indique si el solapamiento es recuperable, justificando su respuesta.
  - c) Indique si podrían ocurrir rollbacks en cascada ante un eventual aborto de una transacción en medio del solapamiento, justificando su respuesta.
2. (*2PL*) Un SGBD soporta el manejo de transacciones ACID en forma concurrente utilizando el *Protocolo de Lock de 2 Fases (2PL)*. En este contexto, indique si las siguientes afirmaciones relativas a la ejecución de transacciones son verdaderas ó falsas, justificando cada una de sus respuestas.
- a) Una transacción  $T_i$  no tomará un lock sobre un ítem  $X$  hasta tanto toda otra transacción  $T_j$  que tomó un lock sobre el ítem  $X$  previamente haya hecho su *commit*.
  - b) Una transacción  $T_i$  no volverá a tomar un lock sobre un ítem  $X$  luego de haberlo liberado.
  - c) Una transacción  $T_i$  no tomará un lock sobre un ítem  $X$  luego de haber liberado un lock sobre otro ítem  $Y$ .
  - d) Una transacción  $T_i$  no leerá un ítem  $X$  sin tener en posesión un lock sobre dicho ítem.
  - e) Una transacción  $T_i$  nunca poseerá locks sobre dos ítems distintos,  $X$  e  $Y$ , simultáneamente.
3. (*Lectura Sucia*) Se dice que una transacción  $T_2$  realiza una Lectura Sucia ó *Dirty Read* cuando la misma lee un ítem cuyo último valor fue escrito por una transacción  $T_1$  que aún no hizo su *commit*. La Lectura Sucia da lugar a potenciales anomalías si no se toman medidas preventivas. Indique si las siguientes afirmaciones sobre la misma son verdaderas ó falsas, justificando sus respuestas.
- a) En un solapamiento recuperable de transacciones puede ocurrir una Lectura Sucia.
  - b) Aplicando el Protocolo de Lock de Dos Fases (2PL) puede ocurrir una Lectura Sucia.
  - c) Aplicando el Protocolo de Lock de Dos Fases Riguroso (R2PL) puede ocurrir una Lectura Sucia.
  - d) Bajo el nivel de aislamiento *Read Committed* definido en el estándar SQL puede ocurrir una Lectura Sucia.

4. (*Control de concurrencia basado en timestamps*) El algoritmo de control de concurrencia basado en timestamps asigna a cada transacción  $T_i$  un *timestamp* único  $TS(T_i)$  y mantiene para cada ítem  $X$  información sobre el timestamp de la última transacción que lo leyó,  $read\_TS(X)$ , y de la última transacción que lo escribió,  $write\_TS(X)$ .

Para garantizar serializabilidad el algoritmo utiliza una serie de reglas para determinar si en algún momento una transacción debe ser abortada.

Complete el siguiente cuadro referente un solapamiento *tentativo* de dos transacciones bajo el esquema de control de concurrencia basado en timestamps, indicando en cada fila qué cambios se producen en los  $read\_TS$  y  $write\_TS$  a partir de la operación respectiva. Detenga el análisis cuando encuentre que una transacción deberá ser abortada con el fin de garantizar la serializabilidad, indicando de qué transacción y operación se trata, y cuál es la regla que viola. A modo de ejemplo se ha completado la primera fila,  $R_{T_2}(X)$ .

	$T_1$	$T_2$	$T_3$	$X$	$Y$	$Z$
Valores inic.	TS=5	TS=8	TS=4	read_TS=0 write_TS=0	read_TS=0 write_TS=0	read_TS=0 write_TS=0
		$R(X)$ $W(Z)$	$R(Z)$	read_TS(X)=8	—	—
	$R(X)$ $W(Y)$ $R(Z)$ $W(Z)$		$W(X)$			

5. (*Control de concurrencia multiversión*) El algoritmo de control de concurrencia basado en timestamps tiene una desventaja: ante una lectura o escritura “fuera de orden” de un ítem se aborta la transacción.

Uno de los primeros métodos de control multiversión, propuesto por Reed en 1978, intentó paliar este problema. Este método, además de mantener los timestamps asociados a cada transacción,  $TS(T_i)$ , propone crear una nueva versión  $X_k$  de cada ítem  $X$  cada vez que una transacción  $T_k$  modifica ese ítem, manteniendo registros  $read\_TS(X_k)$  y  $write\_TS(X_k)$  para cada versión.

La idea del algoritmo es que cada transacción  $T_i$  vea una snapshot del momento en que se creó, reflejado en su timestamp asociado,  $TS(T_i)$ .

Complete a continuación la lógica de este algoritmo de manera de garantizar la serializabilidad de los solapamientos:

- a) Ante una lectura  $R(X)$  por parte de una transacción  $T_i$ , se debe \_\_\_\_\_
- b) Ante una escritura  $W(X)$  por parte de una transacción  $T_i$ , se consideran dos casos:
- 1) Si alguna transacción  $T_j$  posterior a  $T_i$  (es decir, tal que  $TS(T_j) > TS(T_i)$ ) leyó una versión  $k$  del ítem escrita por una transacción  $T_k$  anterior a  $T_i$  ( $TS(T_k) < TS(T_i)$ ), entonces se debe \_\_\_\_\_
  - 2) De lo contrario se debe \_\_\_\_\_

Utilice en cada caso una ó más de las siguientes opciones: ejecutar la lectura - ejecutar la escritura - creando una nueva versión del ítem  $X$ ,  $X_k$  - devolviendo la versión  $X_k$  de  $X$  más recientemente modificada anterior a  $TS(T_i)$  - actualizar  $read\_TS(X_k)$  con  $TS(T_i)$  - actualizar  $write\_TS(X_k)$  con  $TS(T_i)$  - abortar la transacción.

6. (*★Snapshot Isolation*) En el método de control de concurrencia conocido como *Snapshot Isolation* cada transacción ve una imagen de la base de datos correspondiente al instante en el cual la transacción se inició. Para lograr esto, el SGBD mantiene múltiples versiones  $X_k$  de los ítems de datos  $X$ .

Este método puede en principio dar lugar a anomalías, por lo que distintas implementaciones lo complementan con técnicas adicionales para garantizar un cierto nivel de aislamiento. Por ejemplo, la implementación de *snapshot isolation* de *Oracle* utiliza el siguiente criterio: “En el momento en que una transacción  $T_i$  hace su commit, si se observa que modificó algún ítem que también ha sido modificado por una transacción  $T_j$  que hizo su commit después de que  $T_i$  inició, entonces  $T_i$  se deshace y aborta (esta regla se conoce como *first-committer-wins*)”.

Aún así, esta implementación no garantiza la serializabilidad. Se pide:

- a) Indique si el mecanismo de *snapshot isolation* puede padecer alguna de las siguientes situaciones:
- Lectura sucia:  $W_{T_1}(X) \dots R_{T_2}(X) \dots (c_{T_1} \text{ ó } a_{T_1})$
  - Lectura no repetible:  $R_{T_1}(X) \dots W_{T_2}(X) \dots (c_{T_1} \text{ ó } a_{T_1})$
  - Fantasma:  $R_{T_1}(X_{\text{cond}}) \dots W_{T_2}(X \in X_{\text{cond}}); \dots (c_{T_1} \text{ ó } a_{T_1})$
  - Write-skew:  $R_{T_1}(X) \dots R_{T_2}(Y) \dots W_{T_1}(Y) \dots W_{T_2}(X) \dots (c_{T_1} \text{ y } c_{T_2})$
- b) Encuentre un solapamiento de 2 transacciones bajo la implementación de *Snapshot Isolation* de *Oracle* que no sea serializable.
7. (*Recuperación*) Un SGBD implementa el algoritmo de recuperación UNDO con checkpoint activo. Luego de una falla, el sistema encuentra el siguiente archivo de log:

```
01 (BEGIN, T1);
02 (BEGIN, T2);
03 (WRITE T1, X, 18);
04 (WRITE T2, Y, 4);
05 (BEGIN, T3);
06 (WRITE T1, Z, 20);
07 (COMMIT, T1);
08 (WRITE T2, Z, 6);
09 (BEGIN CKPT, T2, T3);
10 (WRITE T3, X, 0);
11 (COMMIT, T2);
12 (WRITE T3, Z, 8);
13 (BEGIN, T4);
14 (WRITE T4, Y, 0);
```

Explique cómo se llevará a cabo el procedimiento de recuperación, indicando qué cambios deben ser realizados en disco y en el archivo de log.

## GUIA 7: BASES DE DATOS DISTRIBUIDAS Y NOSQL

1. (*Consistencia*) Una base de datos distribuida almacena datos de tipo (*clave: valor*) replicados en 2 nodos. En un instante inicial  $t_0$ , ambos nodos contienen la misma información:

(A: 15), (B: 12), (C: 16).

En ese momento, dos procesos  $P_1$  y  $P_2$  comienzan a ejecutar en forma concurrente los siguientes programas, que involucran operaciones de tipo *put()* y *get()* sobre el *data store*. El primer proceso opera sobre el nodo 1, mientras que el segundo lo hace sobre el nodo 2:

Proceso $P_1$ (Nodo 1)	Proceso $P_2$ (Nodo 2)
$vA = \text{get}(A)$	$vC = \text{get}(C)$
$vC = \text{get}(C)$	$vB = \text{get}(B)$
$r = vC + vA$	$r = vB + vC$
$\text{put}(B, vA)$	$\text{put}(B, r)$
$\text{put}(C, r)$	

Si bien el *data store* se ocupa de propagar la información entre sus nodos, lo hace de manera asincrónica y sólo nos garantiza *consistencia eventual*. La siguiente figura ilustra los efectos que tuvo la ejecución concurrente de los dos procesos sobre el *data store*, indicando el valor de cada lectura y escritura.

$P_1$	R(A)15	R(C)16	W(B)15	W(C)31
$P_2$	R(C)16	R(B)15	W(B)31	

Se pide:

→ Tiempo local

- a) Indique si la ejecución presenta consistencia secuencial. Justifique su respuesta.
  - b) Indique si la ejecución es equivalente a alguna ejecución serial de los procesos (es decir si los procesos, vistos como transacciones, se ejecutaron de manera serializable). Justifique su respuesta.
2. (*MongoDB*) Una cadena de supermercados almacena información sobre las ventas que realiza en una base de datos MongoDB. El siguiente documento JSON ejemplifica la estructura almacenada por cada venta, que incluye los siguientes campos:
- El número de ticket, que se utiliza como *\_id*.
  - La fecha y hora de la venta.
  - El DNI del comprador.
  - El nombre del comprador.
  - Un vector de documentos embebidos conteniendo cada uno un código de producto, nombre de producto, cantidad de unidades y precio por dicha cantidad de unidades.
  - El monto total de la venta.

```

1  {
2    "_id": "0057-46290015",
3    "fecha_venta": Date("2018-02-06 11:53:28"),
4    "DNI_comprador": 38115218,
5    "nombre_comprador": "BENITEZ, JUAN ANTONIO",
6    "productos": [
7      { "codigo_producto": 108431,
8        "nombre_producto": "pañales makako 20 un.",
9        "cantidad_producto": 3,
10       "precio_por_cantidad": 241.50
11     },
12     { "codigo_producto": 210160,
13       "nombre_producto": "papas fritas boom 150 gr.",
14       "cantidad_producto": 1,
15       "precio_por_cantidad": 41.00
16     }
17   ],
18   "monto_total": 282.50
19 }

```

La gerencia identifica a las familias que tienen bebés como aquellas que adquieren pañales “*Makako*” en sus compras. Recientemente se ejecutó la siguiente consulta para analizar la evolución mensual de la cantidad de compradores únicos que han adquirido dichos pañales:

```

[
  { $match: { "productos.codigo_producto": { $in: [108431] } } },
  { $project: { mes: { $month: "$fecha_venta" },
    a o: { $year: "$fecha_venta" },
    comprador: "$DNI_comprador" } },
  { $group: { _id: { "a o": "$a o", "mes": "$mes" },
    compradores: { $addToSet: "$comprador" } } },
  { $project: { cant_compradores: { $size: "$compradores" } } },
  { $sort: { "_id.a o": -1, "_id.mes": -1 } },
  { $limit: 10 }
]

```

Lamentablemente, esta consulta mostró que en los últimos 10 meses la cantidad de compradores de pañales ha ido en disminución, y la gerencia lo atribuye a que las familias con bebés están frecuentando menos el supermercado.

Se quiere entonces proponer ofertas interesantes para las familias que tienen bebés, y para ello se desea saber cuáles son los productos más frecuentemente adquiridos junto con pañales *Makako*. Escriba una consulta en MongoDB que encuentre los cinco productos que aparecen más frecuentemente en tickets de compra que incluyen pañales *Makako*, indicando el código del producto y la cantidad de tickets de compra distintos en los que aparece combinado con dichos pañales. El formato de los documentos resultantes deberá ser:

```

1  {
2    "_id": 210160,
3    "cantidad_tickets_compra": 3804
4  }
5  ...

```

3. (*MongoDB*) La *Facultad de Ingeniería* quiere obtener un histograma del promedio con que egresan los alumnos, calculando para cada nota promedio (truncada a un dígito) la cantidad de alumnos que egresan con esa nota. Para ello dispone de una base de datos en *MongoDB*, en cuya colección *Graduados* se encuentran cargadas las notas finales de cada alumno en cada materia. Los documentos de la colección *Graduados* poseen la siguiente estructura:

```
1 {
2   "_id": ObjectId("207ca8372c83b6b092d12512"),
3   "padron": 97294,
4   "apellido": "Mart nez",
5   "nombres": "Gonzalo",
6   "notas": [
7     { "codigo_materia": 7541,
8       "nombre_materia": "ALGORITMOS Y PROGRAMACI N II",
9       "nota_final": 8
10    },
11    { "codigo_materia": 6620,
12      "nombre_materia": "ORGANIZACI N DE COMPUTADORAS",
13      "nota_final": 9
14    },
15    ...
16  ]
17 }
```

Escriba una consulta en *MongoDB* que permita obtener como resultado el histograma de notas. El formato del resultado deberá ser:

```
1 {
2   "nota_promedio": 7.7,
3   "cantidad_alumnos": 184
4 }
5 {
6   "nota_promedio": 7.8,
7   "cantidad_alumnos": 240
8 }
9 ...
```

Para recordar parte de la sintaxis de *MongoDB* puede ayudarse con la siguiente consulta que cuenta la cantidad de alumnos egresados, agrupándolos por su número de padrón truncado por las decenas de mil (es decir, con padrón 8xxxx, 9xxxx, etc.). Observe que el operador `$trunc` no permite especificar la posición decimal en que se debe truncar el número.

---

```
[
  { $project: { padron_truncado: { $multiply:
                                [ { $trunc: { $divide: [ "$padron", 10000 ] } },
                                  10000 ] } } },
  { $group:   { _id: "$padron_truncado", cantidad_graduados: { $sum: 1 } } }
]
```

---



4. (*Neo4j*) Una red social almacena datos sobre sus usuarios y las páginas que les gustan en una base de datos Neo4J. Los usuarios (*User*) y las páginas (*Page*) tienen la siguiente estructura:

```
1 CREATE (p:User {name: 'Pablo', place: 'Mar del Plata'})
2       (l:User {name: 'Luisa', place: 'Escobar'})
3       ...
4       (teg:Page {name: 'TEG', type: 'Juegos'})
5       (aldosivi:Page {name: 'Club Atletico Aldosivi', type: 'Clubes'})
6       ...
```

Las relaciones de amistad entre los usuarios se almacenan a través de la interrelación IS\_FRIEND:

```
1 MATCH (p:User {name: 'Pablo'}),
2       (l:User {name: 'Luisa'})
3 CREATE (p)-[:IS_FRIEND]->(l)
4 MATCH ... CREATE ...
```

Mientras que la información sobre qué páginas le gustan a cada usuario se almacena a través de la interrelación LIKES:

```
1 MATCH (p:User {name: 'Pablo'}),
2       (teg:Page {name: 'TEG'})
3 CREATE (p)-[:LIKES]->(teg)
4 MATCH (p:User {name: 'Pablo'}),
5       (a:Page {name: 'Club Atletico Aldosivi'})
6 CREATE (p)-[:LIKES]->(a)
7 MATCH ... CREATE ...
```

La siguiente consulta en Cypher obtiene la cantidad de páginas que agradan en común a Pablo con cada uno de sus amigos:

```
1 MATCH (amigo:User)-[:IS_FRIEND]-(p:User),
2       likes_comun_amigo = (amigo:User)-[:LIKES]->(pag1:Page)<-[:LIKES]-(p:
3       User)
4 WHERE p.name='Pablo'
5 RETURN amigo.name, count(likes_comun_amigo)
```

Nos interesa recomendar a Pablo nuevos amigos. Modifique la consulta anterior para obtener la cantidad de páginas que agradan en común a Pablo con los amigos de sus amigos, pero que no son amigos de Pablo. El formato del resultado debería ser (*nombre\_amigo\_amigo*, *páginas\_común*).

5. (*Neo4j*) Una clínica utiliza una base de datos Neo4J para realizar consultas sobre los medicamentos a recetar a sus pacientes. Esta base de datos almacena a los Medicamentos, Acciones Terapéuticas y Drogas como nodos del grafo:

```
1 (med1:Medicamento {nombre: 'Lotrial'})
2 (med2:Medicamento {nombre: 'Alloboxal'})
3 (med3:Medicamento {nombre: 'Benzetacil'})
4 (dr1:Droga {nombre: 'Enalapril'})
5 (dr2:Droga {nombre: 'Alopurinol'})
6 (dr3:Droga {nombre: 'Penicilina'})
7 (at1:AccionTerapeutica {nombre: 'Antihipertensivo'})
8 (at2:AccionTerapeutica {nombre: 'Antigotoso'})
9 (at3:AccionTerapeutica {nombre: 'Antibiotico'})
```

Además, se almacenan las siguientes interrelaciones:

- **(Medicamento) - [:CONTIENE] -> (Droga)**

Indica que un Medicamento contiene en su composición una determinada Droga. Cada medicamento puede contener una o más drogas.

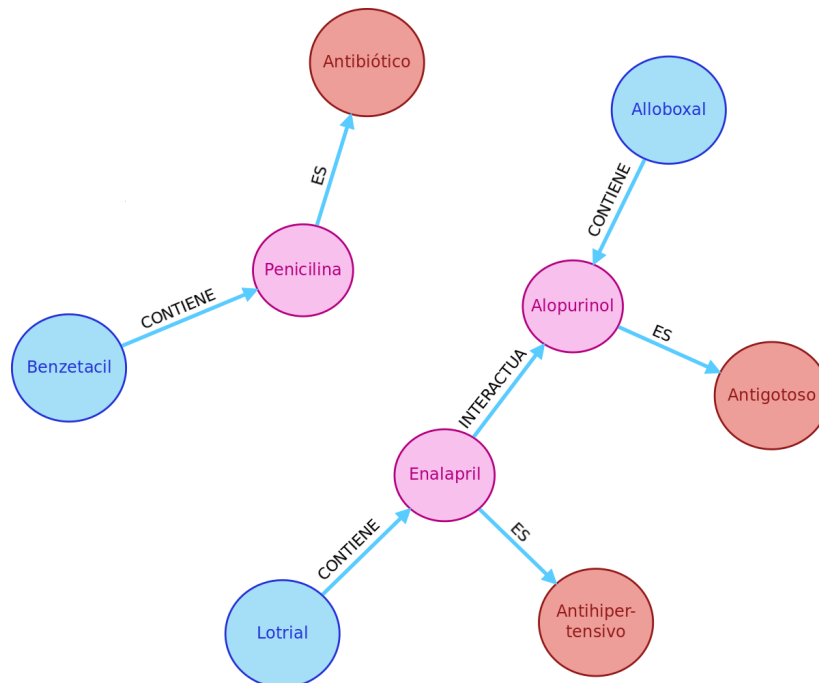
- **(Droga) - [:ES] -> (AccionTerapeutica)**

Indica que una Droga tiene una determinada Acción Terapéutica. Una droga puede tener una o más acciones terapéuticas.

- **(Droga) - [:INTERACTUA] - (Droga)**

Indica que dos Drogas tienen una interacción medicamentosa entre sí. Esto quiere decir que, suministradas simultáneamente a un paciente, pueden tener efectos colaterales.

A continuación se muestra como ejemplo un subgrafo de la base de datos con algunos nodos e interrelaciones:



Para recetar un antibiótico a un paciente alérgico a la penicilina, el doctor Gregorio realizó ayer la siguiente consulta a la base de datos:

```
1 MATCH (m:Medicamento)-[:CONTIENE]->(:Droga)-[:ES]->(a:AccionTerapeutica)
2 WHERE NOT (m)-[:CONTIENE]->(:Droga {nombre: 'Penicilina'})
3 AND a.nombre = 'Antibiotico'
4 RETURN DISTINCT m.nombre
```

Acaba de llegar un paciente hipertenso con una infección, y el doctor Gregorio quiere recetarle dos medicamentos distintos: uno que contenga un antihipertensivo y otro que contenga un antibiótico. Además quiere asegurarse de que las drogas de un medicamento no tengan interacciones medicamentosas con las drogas del otro medicamento.

Escriba la consulta en Cypher que encuentre los posibles pares de medicamentos en la base de datos. El formato del resultado debería ser *(nombre\_medicamento\_1, nombre\_medicamento\_2)*.

6. (*Cassandra*) Los vikingos se están preparando para las próximas celebraciones de verano y se han repartido la recolección de provisiones como leña, hongos, frutos y peces. Para llevar el control de lo recolectado instalaron una base de datos en *Cassandra* en la que definieron las familias de columnas **EspecialidadesPorVikingo**, **RecursosPorVikingoTipo** y **RecursosPorTipoVikingo** de la siguiente manera:

```
CREATE TABLE EspecialidadesPorVikingo (  
    nombre_vikingo text,  
    especialidad text,  
    asentamiento_origen text static,  
    primary key ((nombre_vikingo), especialidad));
```

```
CREATE TABLE RecursosPorVikingoTipo (  
    nombre_vikingo text,  
    tipo_recurso text,  
    cantidad int,  
    primary key ((nombre_vikingo), tipo_recurso));
```

```
CREATE TABLE RecursosPorTipoVikingo (  
    tipo_recurso text,  
    nombre_vikingo text,  
    cantidad int,  
    primary key ((tipo_recurso), nombre_vikingo));
```

Para cada una de las siguientes consultas, indique si la misma es válida en lenguaje *CQL* y si puede ser ejecutada o no a partir del diseño anterior. En aquellos casos en que no sea posible, reformule la consulta y/o construya una nueva familia de columnas para que se pueda obtener el resultado pedido.

- a) Listar los nombres de los vikingos especialistas en caza:

```
SELECT nombre_vikingo  
FROM EspecialidadesPorVikingo  
WHERE especialidad='CAZA';
```

- b) Encontrar la cantidad total de leña recolectada por vikingos provenientes de Kattegat:

```
SELECT sum(cantidad)  
FROM RecursosPorVikingoTipo, EspecialidadesPorVikingo  
WHERE RecursosPorVikingoTipo.nombre_vikingo =  
    EspecialidadesPorVikingo.nombre_vikingo  
AND EspecialidadesPorVikingo.asentamiento_origen = 'KATTEGAT'  
AND RecursosPorVikingoTipo.tipo_recurso = 'LEÑA';
```

- c) Para cada vikingo que proveyó hongos, encontrar su nombre y la cantidad provista:

```
SELECT nombre_vikingo, cantidad  
FROM RecursosPorTipoVikingo  
WHERE tipo_recurso = 'HONGOS';
```

## GUIA 8: PROCESAMIENTO Y OPTIMIZACIÓN DE CONSULTAS

1. (*Loops anidados por bloques*) El método de junta de loops anidados por bloques utiliza dos *loops* para comparar uno a uno todos los bloques de una de las tablas contra todos los bloques de la otra. Suponiendo la disponibilidad de únicamente 2 bloques de memoria RAM (*worst-case*), el costo del método es de  $\min(B(R), B(S)) + B(R) \cdot B(S)$ . Con memoria infinita, en cambio, el costo sería de  $B(R) + B(S)$ .

Generalice el cálculo de la junta con este método, suponiendo disponibles  $M$  de bloques de memoria RAM, y verifique que en los dos escenarios extremos su cálculo coincida con los resultados mencionados anteriormente.

2. (*Uso de índices*) Dado el siguiente esquema de base de datos que guarda información de los clientes morosos de una empresa y de los abogados que llevan adelante los casos:

- **Clientes**(cod\_cliente, nombre, domicilio, deuda, matricula\_abogado)
- **Abogados**(matricula, nombre, telefono)

encontramos que la siguiente consulta SQL se realiza de forma muy frecuente:

```
SELECT a.nombre, a.telefono, c.nombre, c.deuda
FROM Abogados a, Clientes c
WHERE a.matricula = c.matricula_abogado
AND c.cod_cliente = x;
```

en donde  $x$  es un valor variable entre consultas. ¿Cuál de las siguientes alternativas de creación de índices propondría con el objetivo de agilizar las consultas? Justifique su respuesta.

- Un índice por **Abogados.matricula** y otro por **Clientes.matricula\_abogado**
- Un índice compuesto por (**Abogados.nombre**, **Clientes.nombre**)
- Un índice por **Abogados.matricula** y otro por **Clientes.cod\_cliente**
- Un índice compuesto por (**Cliente.cod\_cliente**, **Cliente.matricula\_abogado**)
- Un índice por **Abogados.nombre** y otro por **Clientes.nombre**

3. (*Costo de la junta con índice*) Dados los siguientes esquemas de relación sobre los clientes de una empresa y las facturas de las ventas realizadas:

- **Clientes**(cod\_cliente, nombre, apellido)
- **Facturas**(numero\_factura, cod\_cliente, fecha\_factura, monto)

se propone construir un índice de árbol por código de cliente (*cod\_cliente*) en la tabla *Facturas* para mejorar el costo de resolución de la junta natural en términos de accesos a disco.

- a) Calcule el nuevo costo de la junta natural suponiendo que el índice a construir fuera de agrupamiento (*clustering*).
- b) Calcule el nuevo costo de la junta natural en caso que el índice a construir no fuera de agrupamiento.

Asuma en ambos casos que un índice de árbol tendría una altura de 3, y considere la siguiente información de catálogo. Muestre todos los cálculos involucrados.

CLIENTES	FACTURAS
$n(\text{Clientes}) = 1500$	$n(\text{Facturas}) = 45000$
$B(\text{Clientes}) = 150$	$B(\text{Facturas}) = 9000$
	$V(\text{cod_cliente}, \text{Facturas}) = 1500$

4. (*Cardinalidad de la junta*) Los siguientes esquemas de relación almacenan los discos preferidos de los usuarios de un sistema de música *online*, y las canciones que componen cada *disco*:

- Usuarios(nombre\_usuario, password, nombre, apellido, edad, localidad)
- DiscosUsuarios(nombre\_usuario, nombre\_disco, puntaje)
- ComposiciónDiscos(nombre\_disco, nombre\_canción, duración)

Se desea estimar el tamaño de la junta natural DiscosUsuarios \* ComposiciónDiscos. Para ello se cuenta con la siguiente información de catálogo:

DISCOS USUARIOS	COMPOSICIÓN DISCOS
n(DiscosUsuarios) = 14000	n(ComposiciónDiscos)=300
B(DiscosUsuarios) = 1400	B(ComposiciónDiscos)=15
V(nombre_disco, DiscosUsuarios) = 100	V(nombre_disco, ComposiciónDiscos)=40

- a) Estime la cardinalidad del resultado de la junta natural entre ambas tablas, en términos de cantidad de tuplas.
- b) Estime la cantidad de bloques en disco que ocupará el resultado.
- Ayuda: Considere que una tupla del resultado ocupa un espacio igual al espacio ocupado por una tupla de DiscosUsuarios más el espacio ocupado por una tupla de ComposiciónDiscos.
5. (*Cardinalidad con histograma*) Estime el tamaño de la junta  $R(A, B) \bowtie S(B, C)$  utilizando histogramas para  $R.B$  y  $S.B$ . Asuma que  $V(B, R) = 20$ ,  $V(B, S) = 21$  y considere el siguiente histograma almacenado por el SGBD, en que se incluyen los 4 valores más frecuentes en cada tabla:

	0	1	2	3	4	otros
R.B	5	6	4	5		32
S.B	10	8	5		7	31

6. (*Junta sort-merge*) En este ejercicio compararemos en una situación concreta el método de *sort-merge* para el cálculo de la junta natural con el método de loops anidados por bloques.

(*Parte I*) Suponga que dispone de 4GB de memoria RAM, y debe ordenar una tabla con 4 millones de bloques de 4KB, que ocupa 16GB.

- a) Considere que se divide la tabla en cuatro partes de 4GB cada una, que denominaremos P1, P2, P3 y P4. El primer paso (*sort*) consistirá en leer cada una de las cuatro partes aprovechando toda la memoria disponible, ordenarlas y guardarlas en disco. Calcule el costo de este paso en términos de cantidad de accesos a disco.
- b) El segundo paso (*merge*) consiste en combinar las 4 partes ya ordenadas, guardando el resultado ordenado en disco. Calcule el costo de este paso en términos de cantidad de accesos a disco. Indique cuál es la mínima cantidad de memoria RAM que debe tener disponible para poder mantener ese costo en este paso.
- c) Calcule el costo total que le insumió ordenar la tabla y guardarla en términos de cantidad de accesos a disco.

(Parte II) Considere ahora los esquemas de relación  $R(A, B, C)$  y  $S(C, D, E)$ , con 4 millones de bloques de 4KB (16GB) y 2 millones de bloques de 4KB (8GB) respectivamente.

- Calcule el costo total de la junta natural  $R * S$  utilizando el método de *sort-merge* disponiendo de 4GB de RAM, en base a la fórmula calculada en la Parte I. Recuerde que además de ordenar cada tabla, luego debe juntarlas.
- Calcule en forma aproximada el costo de la junta natural  $R * S$  utilizando el método de loops anidados por bloques, suponiendo que emplea todos los bloques de memoria menos uno ( $\approx 4GB$ ) para la lectura de la tabla  $S$ , y el bloque restante para recorrer  $R$ .
- ¿Qué método resultó más conveniente? Si aumenta el tamaño de las tablas manteniendo la cantidad de memoria RAM, ¿cree que esta relación se reafirma o se revierte? Justifique su respuesta.

Muestre todos los cálculos involucrados.

Nota: No considere el costo de guardado del resultado final en disco.

- (Junta hash GRACE) La red social *Bareando* conecta a personas que frecuentan bares. Para ello almacena las siguientes relaciones:

- Visitas(nombre\_usuario, nombre\_bar, cantidad)
- Usuarios(nombre\_usuario, localidad, fecha\_alta)
- Bares(nombre\_bar, dirección, ciudad, teléfono)
- Amistades(nombre\_usuario\_1, nombre\_usuario\_2)

Se desea invitar a conectarse a aquellas personas que hayan visitado bares en común. Como primer paso, se calcula la siguiente junta por igual con el método de junta *hash GRACE*:

$$Encuentros \leftarrow Visitas \bowtie_{Visitas1.nombre\_bar=Visitas2.nombre\_bar} Visitas$$

Para la tabla de *Visitas* poseemos las siguientes estadísticas:

#### VISITAS

$n(Visitas) = 5.000.000$

$B(Visitas) = 200.000$

$V(nombre\_usuario, Visitas) = 500.000$

$V(nombre\_bar, Visitas) = 50.000$

- Estime la cardinalidad del resultado de esta junta en términos de cantidad de tuplas.
- Estime el costo de la operación en términos de cantidad de accesos a disco.
- Si elijo  $m = 100$  particiones para la función de *hash*, ¿cuántos bloques de espacio en memoria debería tener disponibles para poder realizar la operación de junta?
- Si a continuación se quisiera hacer la proyección  $\pi_{nombre\_usuario1, nombre\_usuario2}(Encuentros)$ , ¿cree que la misma podría hacerse en *pipeline* (es decir, procesando una a una las tuplas a la salida de la junta) ó sería necesario materializar el resultado? Justifique su respuesta.

Nota: Asumimos que el esquema de la relación resultante de la junta es Encuentros(nombre\_usuario1, nombre\_bar1, cantidad1, nombre\_usuario2, nombre\_bar2, cantidad2).

8. (*Junta hash GRACE*) Un popular servicio de streaming de películas bajo demanda está construyendo su sistema de recomendaciones, y quiere calcular para cada par de películas  $(p_1, p_2)$  la cantidad de usuarios que han visto ambas. Para calcular estas co-ocurrencias utilizará la siguiente tabla que indica qué usuarios vieron cada película:

■ VisionesUsuarios(nro\_usuario, cod\_película)

El objetivo inicial es calcular la siguiente junta:

$$VisionesUsuarios \bowtie_{1.nro\_usuario=2.nro\_usuario} VisionesUsuarios$$

La tabla *VisionesUsuarios* tiene un tamaño de  $B(VisionesUsuarios)=10.000.000$ , y la cantidad de memoria disponible para realizar la operación es de  $M=1001$  bloques. Por último, la cantidad de usuarios distintos en la tabla es de  $V(nro\_usuario, VisionesUsuarios) = 10.000.000$ .

La compañía ha probado diversas técnicas para realizar la junta de forma eficiente por el método de *junta hash GRACE*, pero no ha logrado resolverla. A continuación le mostramos como referencia un típico pseudocódigo de una *junta hash GRACE* convencional:

---

**Algoritmo 1: HashearTabla( $T, a, h, MAX$ )**

---

**Entrada** : Una tabla  $T$  en disco, un atributo de junta  $a$ , una función de hash  $h$  y un valor  $MAX$  que es el máximo valor devuelto por la función de hash.

**Salida** : Una partición de  $T$  en subtablas  $T_1, T_2, \dots, T_{MAX}$  en disco.

Alocar  $MAX + 1$  bloques  $M_0, \dots, M_{MAX}$  vacíos en memoria

Crear  $MAX$  subtablas vacías  $T_1, \dots, T_{MAX}$  en disco

**Para** (cada bloque de  $T$ ) {

    Cargar bloque en  $M_0$

**Para** (cada tupla  $t$  en  $M_0$ ) {

$H = h(t.a)$

        Agregar  $t$  al bloque de memoria  $M_H$

**Si** ( $M_H$  está lleno) {

            Enviar  $M_H$  a la subtabla  $T_H$  en disco y vaciar  $M_H$

        }

    }

}

Liberar los  $MAX + 1$  bloques de memoria

Devolver direcciones de  $T_1, T_2, \dots, T_{MAX}$  en disco

---



---

**Algoritmo 2: JuntaHashGRACE( $T_1, T_2, a, h, MAX$ )**

---

**Entrada** : Dos tablas  $T_1$  y  $T_2$  en disco, un atributo de junta  $a$ , una función de hash  $h$  y un valor  $MAX$  que es el máximo valor devuelto por la función de hash.

**Salida** : El resultado de la junta de  $T_1$  y  $T_2$  por el atributo  $a$  en disco.

Crear tabla resultado  $R$  vacía en disco

$T_{11}, \dots, T_{1MAX} = \text{HashearTabla}(T_1, a, h, MAX)$

$T_{21}, \dots, T_{2MAX} = \text{HashearTabla}(T_2, a, h, MAX)$

**Para** (cada  $i \in \{1..MAX\}$ ) {

    Cargar  $T_{1i}$  entera en memoria

/\* Requiere  $B(T_{1i})$  bloques disponibles \*/

    Cargar  $T_{2i}$  entera en memoria

/\* Requiere  $B(T_{2i})$  bloques disponibles \*/

    Agregar  $T_{1i} \bowtie_{1.a=2.a} T_{2i}$  a  $R$

/\* Requiere 1 bloque para armar resultado \*/

    Liberar memoria ocupada por  $T_{1i}$  y  $T_{2i}$

}

Devolver dirección de  $R$  en disco

---

- a) En primer lugar alguien propuso utilizar una función de hash  $h(nro\_usuario)$  que devolviera un valor entre 1 y 10000, pero ésto no funcionó. Explique cuál fue el problema de este enfoque.
- b) Luego alguien propuso utilizar una función de hash  $h(nro\_usuario)$  que devolviera un valor entre 1 y 1000, pero tampoco funcionó. Explique cuál fue el problema ahora.
- c) Finalmente alguien propuso modificar la *junta hash GRACE* utilizando dos funciones de hash *distintas*,  $h_1(nro\_usuario)$  y  $h_2(nro\_usuario)$ , cada una devolviendo un valor entre 1 y 300, y de esta manera pudo resolver la junta de forma eficiente. Para hacerlo, modificó la función `JuntaHashGRACE()` de la siguiente manera:

---

**Algoritmo 3:** JuntaHashGRACE( $T_1, T_2, a, h_1, h_2, MAX$ )

---

**Entrada** : Dos tablas  $T_1$  y  $T_2$  en disco, un atributo de junta  $a$ , dos funciones de hash  $h_1$  y  $h_2$ , y un valor  $MAX$  que es el máximo valor devuelto por cada función de hash.

**Salida** : El resultado de la junta de  $T_1$  y  $T_2$  por el atributo  $a$  en disco.

Crear tabla resultado  $R$  vacía en disco

$T1_1, \dots, T1_{MAX} = \text{HashearTabla}(T1, a, h_1, MAX)$

$T2_1, \dots, T2_{MAX} = \text{HashearTabla}(T2, a, h_2, MAX)$

**Para** (*cada*  $i \in \{1..MAX\}$ ) {

$T1_{i,1}, \dots, T1_{i,MAX} = \text{HashearTabla}(T1_i, a, h_2, MAX)$

$T2_{i,1}, \dots, T2_{i,MAX} = \text{HashearTabla}(T2_i, a, h_2, MAX)$

}

**Para** (*cada*  $i \in \{1..MAX\}$ ) {

**Para** (*cada*  $j \in \{1..MAX\}$ ) {

        Cargar  $T1_{i,j}$  entera en memoria

        /\* Requiere  $B(T1_{i,j})$  bloques disponibles \*/

        Cargar  $T2_{i,j}$  entera en memoria

        /\* Requiere  $B(T2_{i,j})$  bloques disponibles \*/

        Agregar  $T1_{i,j} \bowtie_{1.a=2.a} T2_{i,j}$  a  $R$

        /\* Requiere 1 bloque para result. \*/

        Liberar memoria ocupada por  $T1_{i,j}$  y  $T2_{i,j}$

    }

}

Devolver dirección de  $R$  en disco

---

Estime el costo resultante de esta variante de la *junta hash GRACE* (sin considerar el almacenamiento en disco del resultado) y explique por qué finalmente funciona.



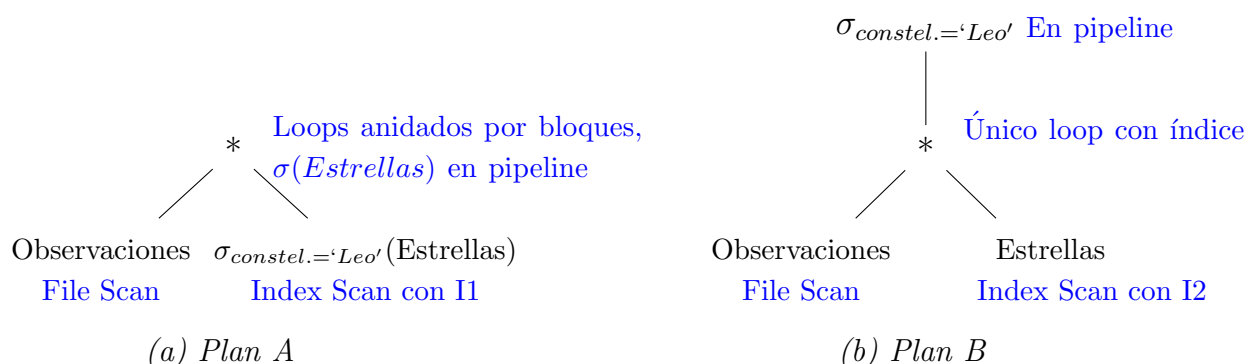
9. (*Ejercicio integrador*) La UFA (*Unión de Fanáticos de la Astronomía*) dispone de una base de datos con información sobre 10000 estrellas y 50000 observaciones realizadas por sus socios. La base de datos cuenta con las siguientes tablas:

- **Estrellas**(nombre\_estrella, constelación, declinación, ascensión)
- **Observaciones**(nro\_socio, fecha, nombre\_estrella, velocidad, intensidad)

También se cuenta con los siguientes dos índices de tipo árbol:

- **I1**(constelación, Estrellas): Índice de *clustering* (agrupamiento) por el atributo constelación en la tabla Estrellas.
- **I2**(nombre\_estrella, Estrellas): Índice secundario por el atributo clave nombre\_estrella en la tabla Estrellas.

Con el objetivo de encontrar todas las observaciones de estrellas que pertenecen a la constelación de 'Leo', el SGBD construye los siguientes dos planes de ejecución:



Como puede observar, el *Plan A* respeta las reglas de optimización algebraica para realizar la selección –utilizando el índice correspondiente– previo a la junta –que se realiza sin índices–, mientras que el *Plan B* realiza primero la junta con índice, y luego realiza la selección.

- a) Estime el costo de cada plan en términos de cantidad de accesos a disco, y determine cuál de los dos es más conveniente en este sentido.
- b) Proponga la construcción de un índice adicional que permita planificar esta consulta de manera de utilizar dos índices, y grafique el plan de ejecución correspondiente.

Puede utilizar para sus cálculos la siguiente información de catálogo:

ESTRELLAS	OBSERVACIONES
n(Estrellas) = 10000	n(Observaciones) = 50000
B(Estrellas) = 1000	B(Observaciones) = 5000
V(constelación, Estrellas) = 10	V(nombre_estrella, Observaciones) = 10000
H(I1(constelación, Estrellas)) = 1	
H(I2(nombre_estrella, Estrellas)) = 4	

Nota 1: Considere la utilización de sólo un bloque de memoria por cada tabla/índice.

Nota 2: No considere el costo de almacenar el resultado final de la consulta en disco.