DEPARTAMENTO DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Trabajo práctico Nro. 1

xx/09/2010 Bases de Datos

Integrante	LU	Correo electrónico
Facundo Carrillo	693/07	facu.zeta@gmail.com
Rodrigo Castaño	602/07	castano.rodrigo@gmail.com
Dardo Marasca	227/07	dmarasca@yahoo.com.ar
Federico Pousa	221/07	fedepousa@gmail.com

Facultad de Ciencias Exactas y Naturales

Universidad de Buenos Aires

Ciudad Universitaria - (Pabellón I/Planta Baja)

Intendente Güiraldes 2160 - C1428EGA

Ciudad Autónoma de Buenos Aires - Rep. Argentina

Tel/Fax: (54 11) 4576-3359

http://www.fcen.uba.ar

Índice

1. Modelo de Entidad Relacion

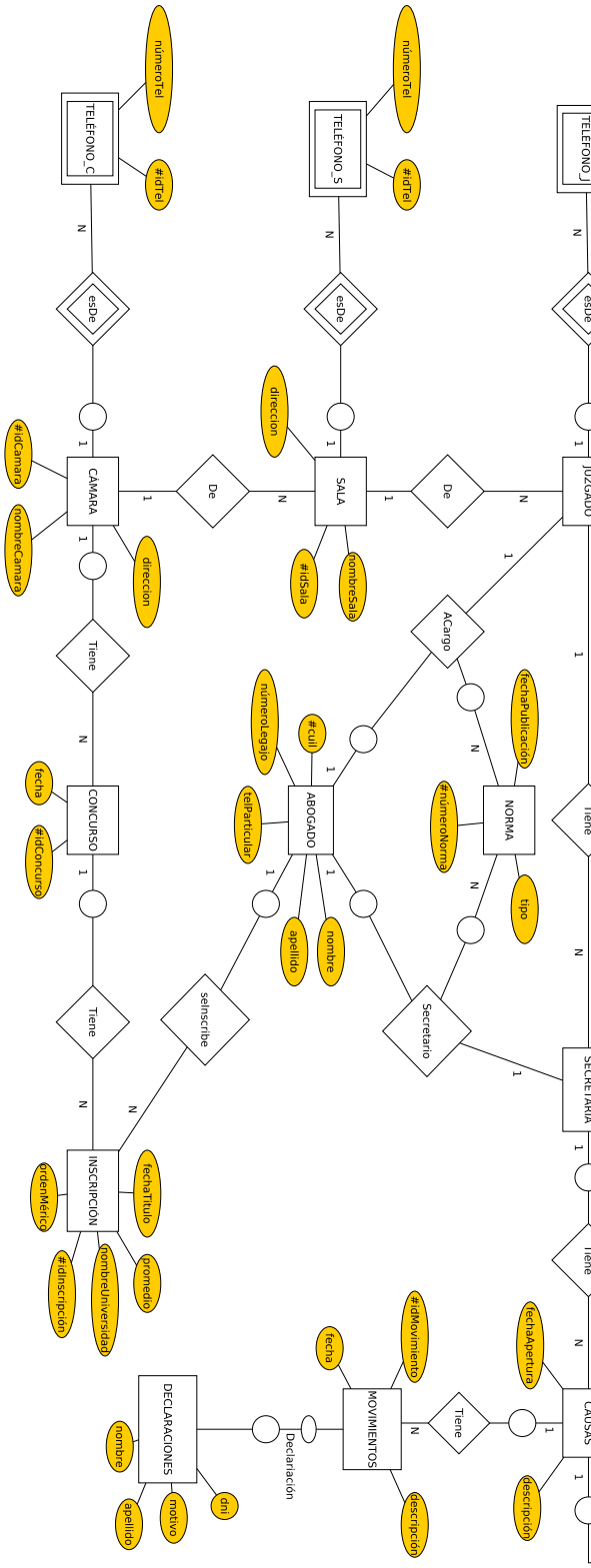

Figura 1: DER

En primer lugar, mencionaremos dos asunciones sobre el dominio del problema que fueron importantes a la hora de realizar el trabajo. Por un lado, se asumió, previa consulta con el docente, que las causas se pueden unir directamente a una secretaría y no al juzgado. Esto se debe a que se puede asumir que cuando una causa llega a un juzgado, esta se deriva automaticamente a una secretaría con algún criterio externo al trabajo, como podría ser el tema de las causas.

Por otro lado, es importante destacar que a los concursos se les adhirió una fecha ya que resultó necesario para confeccionar el segundo store procedure que se pidió en el enunciado; de esta forma, se puede saber cual es el concurso vigente para una norma.

Restricciones adicionales al DER:

- Toda causa esta en un y solo un rol de la interrelación unaria. Es decir, o es la original para un tema dado, o es copia de otra anterior.
- Un Abogado no puede tener dos inscripciones diferentes para el mismo concurso. Esta restricción se podría evitar con un modelado diferente, pero en las explicaciones subsecuentes se verá porque se optó por este modelo.
- Una norma debe designar al menos un juez o un secretario.

A continuación, se detallará cuales fueron las decisiones importantes a la hora de realizar el Diagrama de Entidad Relación:

En primer lugar, el grupo se encontró con diferentes opciones para modelar las normas que rigen los nombramientos de los jueces y los secretarios. Se explicará solamente el caso de las interrelacion entre Abogado, Norma y Juzgado, dado que la relación entre Abogado, Norma y Secretaría es análoga.

Las dos opciones principales consistían en una interrelación ternaria por un lado, y una agregación por el otro. En el caso de la agregación, la idea se basaba en relacionar Abogado con Juzgado y, al hacer una agregación entre estos dos, interrelacionarlos con las normas. Sin embargo, este modelo traía aparejado una interpretación distinta a la requerida por el enunciado. En caso de utilizar esta forma, se hubiese permitido tener una norma relacionada dos veces con el mismo abogado, pero con diferente juzgado, lo cual no es valido.

Dado lo explicado anteriormente, se optó por utilizar una relación ternaria, dado que se ajustaba mejor a los requerimientos elicitados. Es importante destacar que era necesario una relación de este tipo para lograr *tener historia* en las normas; ya que dado que una norma tiene fecha de publicación, podría ser que dos normas diferentes hagan referencia al mismo par de Juzgado y Abogado.

En segundo lugar, se mostrará porque las inscripciones se encuentra modeladas de la forma presentada.

Al tener las inscripciones como una entidad relacionada con los Abogados y con los Concursos, necesitamos la restricción adicional que dice que un Abogado no puede tener dos Inscripciones para el mismo Concurso. Existe otra forma de modelar esto que sería unir directamente Abogado con Concursos, y poner todos los atributos respectivos a la Inscripción en la interrelación. Si bien esto es valido y hubiese evitado la restricción adicional, se consideró que la inscripción tenia fuerza y una cantidad de atributos necesaria como para ser considerada una entidad aparte.

En último lugar, se explicará la interrelación unaria en las causas.

Dado que las causas se deben agrupar por los hechos a los que hacen referencia, se barajó la idea de poner el motivo de la causa como una entidad aparte, pero se consideró que esto no tenía una relevancia necesaria como para ser modelado con una entidad nueva.

De esta manera, se optó una interralación unaria que relaciona las causas entre sí dependiendo del motivo de las mismas. Para realizar esto, se vieron diferentes maneras de hacerlo. Las diferentes maneras estaban basadas en la cardinalidad y la participación de las causas. Finalmente, se eligió utilizar una interrelación con cardinalidad 1:N, en donde la interpretación de la misma es que para un motivo dado hay una causa original que fue la primera en llegar, y luego estan las causas del mismo hecho que llegaron despues que tiene una foreign key a la causa original.

2. Modelo Relacional Derivado

- TELÉFONO_J(idTel, númeroJuzgado, númeroTel)
  - CK = {(idTel, númeroJuzgado)}
  - PK = {(idTel, númeroJuzgado)}
  - FK = {(númeroJuzgado)}
  - númeroJuzgado hace referencia a JUZGADO.númeroJuzgado.
  - Restricciones:
    - TELÉFONO_J.númeroJuzgado debe estar en JUZGADO.númeroJuzgado.

- JUZGADO(númeroJuzgado, fechaCreación, dirección, idSala)
  - CK = {(númeroJuzgado)}
  - PK = {(númeroJuzgado)}
  - FK = {(idSala)}
  - Restricciones:
  - idSala hace referencia a SALA.idSala.
    - JUZGADO.númeroJuzgado puede no estar en TELÉFONO_J.númeroJuzgado.
    - JUZGADO.idSala no puede ser NULO.
    - JUZGADO.númeroJuzgado debe estar en ACARGO.númeroJuzgado.

- TELÉFONO_S(idTel, idSala, númeroTel)
  - CK = {(idTel, idSala)}
  - PK = {(idTel, idSala)}
  - FK = {(idSala)}
  - idSala hace referncia a SALA.idSala.
  - Restricciones:
    - TELÉFONO_S.idSala debe estar en SALA.idSala.

- SALA(idSala, nombreSala, idCámara,dirección)
  - CK = {(idSala)}
  - PK = {(idSala)}
  - FK = {(idCámara)}
  - idCámara hace referencia a CÁMARA.idCámara.
  - Restricciones:
    - SALA.idSala puede no estar en TELÉFONO_S.idSala.
    - SALA.idCámara no puede ser NULO.

- TELÉFONO_C(idTel, idCámara, númeroTel)
  - CK = {(idTel, idCámara)}

- PK = {(idTel, idCámara)}
- FK = {(idCámara)}
- idCámara hace referencia referencia a CÁMARA.idCámara.
- Restricciones:
  - TELÉFONO_C.idCámara debe estar en CÁMARA.idCámara.

- CÁMARA(idCámara, nombreCámara, dirección)
  - CK = {(idCámara)}
  - PK = {(idCámara)}
  - FK = { }
  - Restricciones:
    - CÁMARA.idCámara debe estar en TELÉFONO_C.idTeléfono.
    - CÁMARA.idCámara puede no estar en CONCURSO.idCámara.

- CONCURSO(idConcurso,fecha,idCámara)
  - CK = {(idConcurso)}
  - PK = {(idConcurso)}
  - FK = {(idCámara)}
  - idCámara hace referencia a CÁMARA.idCámara.
  - Restricciones:
    - CONCURSO.idCámara no puede ser NULO.
    - CONCURSO.idConcurso puede no estar en INSCRIPCIÓN.idConcurso.

- INSCRIPCIÓN(idInscripción, ordenMérito, nombreUniversidad, promedio, fechaTitulo, idConcurso, cuil)
  - CK = {(idInscripción)}
  - PK = {(idInscripción)}
  - FK = {(idConcurso),(cuil)}
  - idConcurso hace referencia a CONCURSO.idConcurso.
  - cuil hace referencia a ABOGADO.cuil.
  - Restricciones:
    - INSCRIPCIÓN.idConcurso no puede ser NULO.
    - INSCRIPCIÓN.cuil no puede ser NULO.

- ABOGADO(cuil, númerodeLegajo, telParticular, apellido, nombre)
  - CK = {(cuil)}
  - PK = {(cuil)}
  - FK = {}
  - Restricciones:

- ABOGADO.cuil puede no estar en INSCRIPCIÓN.cuil.
- ABOGADO.cuil puede no estar en ACARGO.cuil.
- ABOGADO.cuil puede no estar en SECRETARIO.cuil.

- NORMA(<u>númeroNorma</u>, fechaPublicación, tipo)
  - CK = {(númeroNorma)}
  - PK = {(númeroNorma)}
  - FK = {}
  - Restricciones:
    - NORMA.númeroNorma puede no estar en ACARGO.númeroNorma.
    - NORMA.númeroNorma puede no estar en SECRETARIO.númeroNorma.

- ACARGO(<u>númeroNorma</u>, <u>númeroJuzgado</u>, cuil)
  - CK = {(númeroNorma,númeroJuzgado),(númeroNorma,Cuil)}
  - PK = {(númeroNorma,númeroJuzgado)}
  - FK = {(cuil)}
  - cuil hace referencia a ABOGADO.cuil.
  - Restricciones:
    - ACARGO.númeroJuzgado debe estar en JUZGADO.númeroJuzgado.
    - ACARGO.númeroNorma debe estar en NORMA.númeroNorma.
    - ACARGO.cuil debe estar en ABOGADO.cuil.

- SECRETARIO(<u>númeroNorma</u>, <u>idSecretaría</u>, cuil)
  - CK = {(númeroNorma,idSecretaría),(númeroNorma,cuil)}
  - PK = {(númeroNorma,idSecretaría)}
  - FK = {(cuil)}
  - cuil hace referencia a ABOGADO.cuil.
  - Restricciones:
    - SECRETARIO.númeroNorma debe estar en NORMA.númeroNorma.
    - SECRETARIO.cuil debe estar en ABOGADO.cuil.
    - SECRETARIO.idSecretaría debe estar en SECRETARÍA.

- SECRETARÍA(<u>idSecretaría</u>, númeroJuzgado)
  - CK = {(idSecretaría)}
  - PK = {(idSecretaría)}
  - FK = {(númeroJuzgado)}
  - númeroJuzgado hace referencia JUZGADO.númeroJuazgado.
  - Restricciones:
    - SECRETARÍA.númeroJuzgado no puede ser NULO.

    - SECRETARÍA.idSecretaría puede no estar en CAUSAS.idSecretaría.
    - SECRETARÍA.idSecretaría debe estar en SECRETARIO.idSecretaria.

- CAUSAS(<u>númeroCausa</u>, fechaApertura, descripción, <u>númeroCausaOriginal</u>, <u>idSecretaría</u>)
  - CK = {(númeroCausa)}
  - PK = {(númeroCausa)}
  - FK = {(númeroCausaOriginal),(idSecretaría)}
  -
  - Restricciones:
    - CAUSAS.idSecretaría no puede ser NULO.
    - CAUSAS.númeroCausa puede no estar en MOVIMIENTOS.númeroCausa.

- MOVIMIENTO(<u>idMovimiento</u>, descripción, fecha, <u>númeroCausa</u>, tipo)
  - CK = {(idMovimiento)}
  - PK = {(idMovimiento)}
  - FK = {(númeroCausa)}
  - númeroCausa hace referencia a CAUSA.númeroCausa.
  - Restricciones:
    - MOVIMIENTO.idMovimiento puede no estar en DECLARACIONES.idMovimiento.

- DECLARACIONES(<u>idMovimiento</u>, dni, motivo, apellido, nombre)
  - CK = {(idMovimiento)}
  - PK = {(idMovimiento)}
  - FK = {(númeroCausa)}
  - Restricciones:
    - DECLARACIONES.idMovimiento debe estar en MOVIMIENTO.idMovimiento.

3. Store Procedures/Triggers

Para realizar las consultas que fueron requeridas por las consignas del trabajo se decidió implementar las soluciones mediante Stored Procedures, para de esta forma poder acceder a las mismas de forma sencilla y cómoda.

Consulta:

El nombre de las salas que más declaraciones tomaron en el último año.

Esta consulta es generada mediante el siguiente código:

```
SELECT nombre 'Nombre de la Sala', declaraciones 'Cantidad de Declaraciones'
FROM sala, (
SELECT id_sala, count( * ) declaraciones
FROM juzgado, (
SELECT id_juzgado
FROM secretaria, (
SELECT id_secretaria
FROM causas, (
SELECT *
FROM movimiento
WHERE id
IN (
SELECT id_movimiento
FROM declaraciones
)
) AS movConDec
WHERE movConDec.id_causa = causas.id
)secConDec
WHERE secConDec.id_secretaria = secretaria.id
) AS juzConDec
WHERE juzConDec.id_juzgado = juzgado.id
GROUP BY id_sala
) AS salaConDec
WHERE sala.id = salaConDec.id_sala
ORDER BY declaraciones DESC
LIMIT 0 , 30
```

La idea que se implemento es ir recorriendo desde las movimientos que fueron declaraciones hasta llegar a las Salas donde estos fueron originados, para ello se fue asociando las identificaciones correspondientes que enlazaban las diferentes tablas durante el camino. De esta forma se obtuvo una tabla con un campo sala y otro que indica la cantidad de declaraciones que fueron tomadas por la misma. Luego se procedió a ordenarlas de forma decreciente según la cantidad de declaraciones.

Se denomino **ranking()**, al Stored Procedure que implementa esta consulta.

Consulta:

El nombre de los jueces/secretarios que fueron nombrados por cada cámara a pesar de haber postulantes con un orden de mérito mejor que aún no habían sido designados.

Esta consulta es generada mediante el siguiente código:

```
SELECT DISTINCT nombre
FROM    abogado AS a, concurso AS c, acargo,
        inscripcion AS i, norma AS n, secretario AS s
WHERE
    (a.cuil = acargo.cuil_abogado OR a.cuil = s.cuil_abogado)
    -- El abogado a fue nombrado juez o secretario

    AND a.cuil = i.cuil_abogado
    -- La inscripcion i corresponde al abogado a

    AND c.id = i.id_concurso
    -- La inscripcion i corresponde al concurso c

    AND c.id IN
    -- El concurso c es el que esta vigente para la norma n.
    -- Es decir, c es el ultimo concurso antes de que se publicara la norma n.
    (
    SELECT id
    FROM concurso AS conc
    WHERE
        conc.fecha <= n.fecha_publicacion
        AND (conc.fecha >= ALL (
            SELECT fecha
            FROM concurso conc1
            WHERE conc1.fecha <= n.fecha_publicacion
            ))
    )
    AND EXISTS (
    -- Existe un abogado inscripto en el mismo concurso,
    -- con menor orden de merito, que
    -- no esta entre los nombrados jueces o secretarios.
    SELECT *
    FROM abogado AS abog, inscripcion AS insc
    WHERE
        abog.cuil = insc.cuil_abogado
        -- La inscripcion insc corresponde al abogado abog.
        AND insc.orden_merito < i.orden_merito
        -- El orden de merito de la inscripcion insc
        -- es menor que el de la inscripcion i.
        AND insc.id_concurso = i.id_concurso
        -- Las inscripciones insc e i
        -- corresponden al mismo concurso
        AND NOT EXISTS (
        -- No esta nombrado el abogado abog por una norma que
        -- corresponda al concurso c.
            SELECT *
            FROM norma AS norm, acargo AS acargo2, secretario AS secretario2
            WHERE
               norm.fecha_publicacion <= n.fecha_publicacion
               -- La norma norm es anterior a la norma n
```

```
                AND norm.fecha_publicacion >= c.fecha
                -- El concurso c ya estaba vigente cuando
                -- se hizo la norma norm.
                AND
                -- El abogado aparece nombrado juez o aparece nombrado
                -- secretario bajo la norma norm.
                    (
                        (acargo2.cuil_abogado = abog.cuil
                        -- El abogado abog aparece nombrado juez.
                        AND norm.id = acargo2.id_norma
                        -- El nombramiento corresponde a la norma norm.
                        )
                     OR
                        (secretario2.cuil_abogado = abog.cuil
                        -- El abogado abog aparece nombrado secretario.
                        AND norm.id = secretario2.id_norma
                        -- El nombramiento corresponde a la norma norm.
                        )
                    )
            )
    )
```

Para resolver la consulta se identifican un abogado, la norma que lo nombra como juez o secretario y a su inscripción en el concurso vigente al momento de publicarse esa norma. Logrado esto, se buscan los abogados que tenían mejor posición en el orden de mérito del concurso vigente para los cuales no existía ninguna norma que los nombrara. Si este último conjunto no es vacío, entonces el abogado identificado fue nombrado habiendo todavía postulantes con un orden de mérito mejor que aún no habían sido designados. Se denominó **acomodados()** al Stored Procedure con la intención de reflejar con un poco de humor la principal característica de los abogados cuyos nombres aparecen.

Control:

Registro automático de datos de control de las altas, bajas y modificaciones de concursos y sus resultados

Para realizar esta tarea usamos la funcionalidad que nos provee el motor de la base de datos llamada "disparadores.º "triggers". La idea es llevar el control de los cambios que se realizan en las tablas relacionadas con la gestión de los concursos (Concurso e Inscripción). Para realizar esto creamos un conjunto de disparadores que registran los movimientos que se realizan en estas tablas y guardan dicha información en otras tablas creadas únicamente para este propósito.

El siguiente código implementa los triggers que mencionamos anteriormente:

```
CREATE TRIGGER auditoria_concurso_ins AFTER INSERT ON concurso
FOR EACH ROW
INSERT INTO auditoria_concurso(tipo, new_id_camara, new_id_concurso
,new_fecha, usuario, fecha)
VALUES ('Ins',NEW.id_camara, NEW.id,NEW.fecha, CURRENT_USER(), NOW());

CREATE TRIGGER auditoria_concurso_del AFTER DELETE ON concurso
FOR EACH ROW
INSERT INTO auditoria_concurso(tipo, old_id_camara, old_id_concurso,
old_fecha,usuario, fecha)
```

```
VALUES ('Del',OLD.id_camara, OLD.id,OLD.fecha, CURRENT_USER(), NOW());

CREATE TRIGGER auditoria_concurso_upd AFTER UPDATE ON concurso
FOR EACH ROW
INSERT INTO auditoria_concurso(tipo, new_id_camara, new_id_concurso,
old_id_camara,old_id_concurso, old_fecha,new_fecha,usuario, fecha)
VALUES ('Upd',NEW.id_camara, NEW.id,OLD.id_camara, OLD.id,OLD.fecha,
NEW.fecha, CURRENT_USER(), NOW());

CREATE TRIGGER auditoria_inscripcion_ins AFTER INSERT ON inscripcion
FOR EACH ROW
INSERT INTO auditoria_inscripcion(tipo, new_id, new_orden_merito,
new_nombre_universidad, new_promedio, new_fecha_titulo, new_id_concurso,
 new_cuil_abogado, usuario, fecha)
VALUES ('Ins',NEW.id, NEW.orden_merito, NEW.nombre_universidad,
 NEW.promedio, NEW.fecha_titulo, NEW.id_concurso, NEW.cuil_abogado,
  CURRENT_USER() , NOW());

CREATE TRIGGER auditoria_inscripcion_del AFTER DELETE ON inscripcion
FOR EACH ROW
INSERT INTO auditoria_inscripcion(tipo, old_id, old_orden_merito,
old_nombre_universidad, old_promedio, old_fecha_titulo, old_id_concurso,
 old_cuil_abogado, usuario, fecha)
VALUES ('Del',OLD.id, OLD.orden_merito, OLD.nombre_universidad,
OLD.promedio, OLD.fecha_titulo, OLD.id_concurso, OLD.cuil_abogado,
CURRENT_USER() , NOW());

CREATE TRIGGER auditoria_inscripcion_upd AFTER UPDATE ON inscripcion
FOR EACH ROW
INSERT INTO auditoria_inscripcion(tipo, old_id, old_orden_merito,
old_nombre_universidad, old_promedio, old_fecha_titulo, old_id_concurso,
old_cuil_abogado,new_id, new_orden_merito, new_nombre_universidad, new_promedio,
new_fecha_titulo, new_id_concurso, new_cuil_abogado, usuario, fecha)
VALUES ('Upd',OLD.id, OLD.orden_merito, OLD.nombre_universidad, OLD.promedio,
OLD.fecha_titulo, OLD.id_concurso, OLD.cuil_abogado,NEW.id, NEW.orden_merito,
NEW.nombre_universidad, NEW.promedio, NEW.fecha_titulo, NEW.id_concurso,
NEW.cuil_abogado, CURRENT_USER() , NOW());
```