FUNDAMENTOS DE BASES DE DATOS
Examen Diciembre 2016
Solución

La duración del examen es de 3 horas.
Presentar la resolución del examen:

- Con las hojas numeradas y escritas de un solo lado.
- Con la cantidad de hojas entregadas en la primer hoja.
- Con cédula de identidad y nombre en cada hoja.
- Escrita a lápiz y en forma prolija.
- Comenzando cada ejercicio en una nueva hoja

Ejercicio 1 (25 puntos).

Una empresa de venta de zapatos a gran escala desea desarrollar un sistema informático para gestionar sus productos, clientes, reservas y ventas. Para esto se necesita realizar un modelo conceptual de su base de datos, que represente la siguiente realidad.

La empresa cuenta con tres tipos de tiendas: sucursales, tiendas móviles (camiones que se instalan en distintos puntos y realizan sus ventas allí) y tienda virtual (que es una sola). Cada tienda tiene un identificador, un nombre y un encargado. Las sucursales tienen además una dirección y un teléfono, las tiendas móviles tienen la matrícula del camión (que también identifica la tienda), su modelo y marca. Finalmente, de la tienda virtual se registra la url (que la identifica).

La empresa comercializa zapatos, los cuales son manejados como productos. Un producto tiene un identificador, un talle y un color. Además, de cada producto se conoce su modelo. Los modelos tienen un código que los identifica y un nombre. Por otro lado, de cada tienda se registra qué productos tiene en stock y la cantidad.

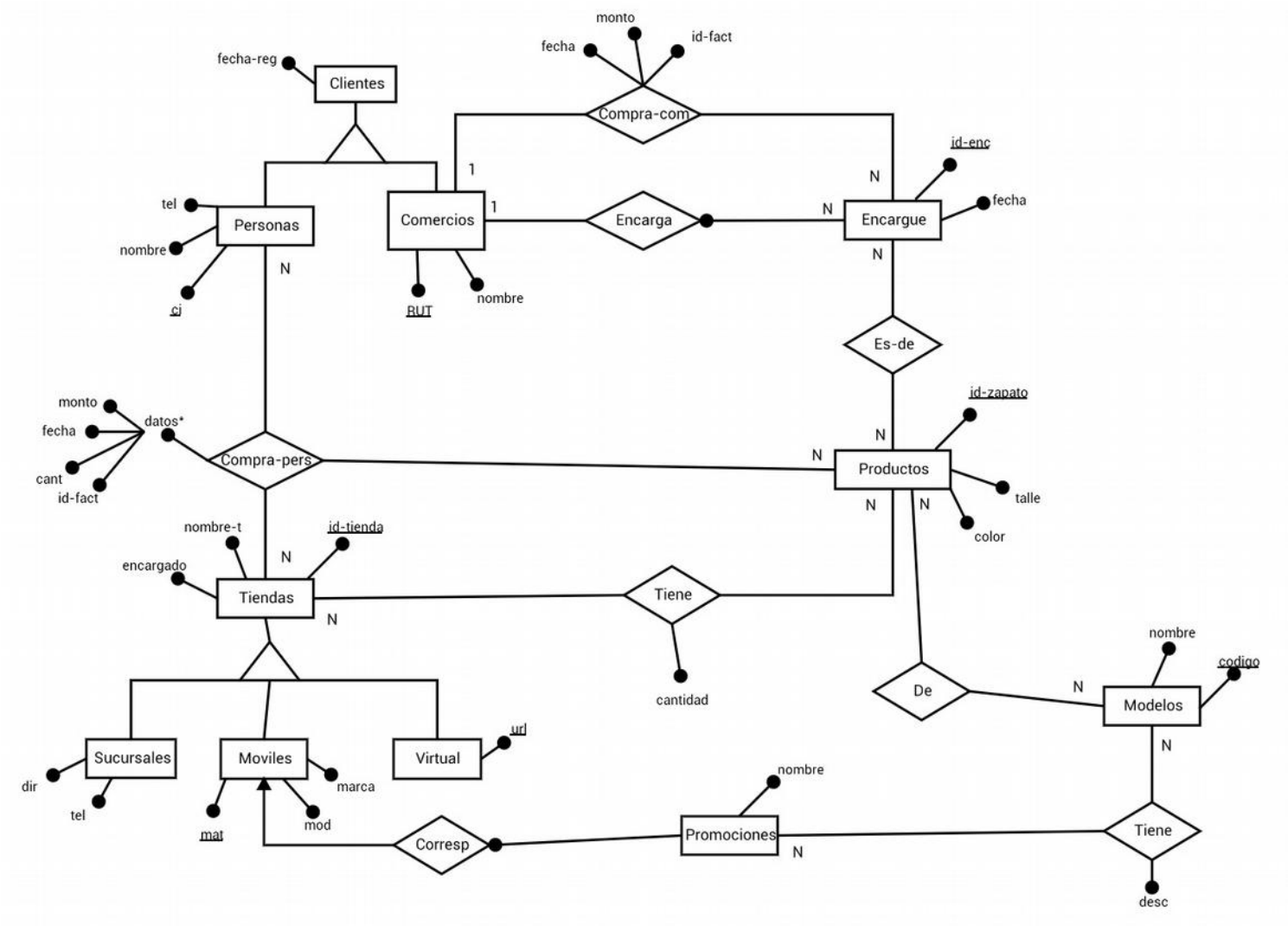
La empresa registra dos tipos de clientes: personas y comercios. Las personas se identifican por su cédula de identidad y los comercios por su número de RUT. Además, de las personas se conoce su nombre y teléfono, y de los comercios su nombre. De los clientes se registra una fecha en la cual se registraron como tales.

Las personas realizan compras de productos en todos los tipos de tiendas, y de estas compras se debe registrar el número de factura, el monto, la fecha y la cantidad comprada.

Por otro lado, los comercios antes de comprar deben realizar encargues. Un encargue tiene un identificador y una fecha, es de cierta cantidad de cada producto y tiene un comercio correspondiente. Los comercios realizan compras y cada compra debe corresponderse con un encargue ya realizado por el mismo comercio. De cada compra se registra la fecha, el monto y el número de factura. Un encargue tiene un solo comercio correspondiente. Un comercio puede realizar muchos encargues y cada compra que realice se va a corresponder con un encargue.

Finalmente, las tiendas móviles realizan diferentes promociones. Las promociones tienen un nombre (por ej. “Navideña” o “Reyes”), el cual la identifica dentro de cada tienda movil. Las promociones están asociadas a los modelos, y dependiendo de la promoción y el modelo hay un descuento determinado.

Se pide Modelo Entidad Relación completo.



RNE:

- Tiendas = Sucursales U Moviles U Virtual
- Virtual ∩ Sucursales = Ø
- Virtual ∩ Moviles = Ø
- Sucursales ∩ Moviles = Ø
- |Virtual| = 1
- Clientes = Personas U Comercios
- Personas ∩ Comercios = Ø
- Compra-com ⊆ Encarga

Ejercicio 2 (30 puntos).

Se desea construir una base de Datos para almacenar datos de Compra - Venta de casas.
De los propietarios, de los compradores y de los escribanos que intervienen, se conoce la cédula de identidad y el nombre .
De las casas se conoce su dirección (como un único string), además de su padrón y ciudad , además de la fecha en que se realiza la transacción. Tenga en cuenta que una casa se puede identificar por su dirección, o bien, por su padrón y ciudad. Se conoce además, el monto que aporta cada comprador en esa transacción.
En cada transacción de compra-venta, deben intervenir al menos un vendedor, al menos un comprador pero sólo un escribano.
Se asume que una casa no se puede vender más de una vez en una fecha dada. Por otro lado, cada vendedor de una propiedad le vende a todos los compradores y viceversa: cada comprador le compra a todos los vendedores.

La tabla universal resultante es la siguiente:

R (CiV, nomV, CiC, nomC, CiE, nomE, dirProp, padron, ciudad, fecha, aporte)

Se pide:

a) Escriba el conjunto de dependencias funcionales que se cumplen. Justifique su respuesta.

SOL:
CiV → nomV /* Estas tres surgen de las personas que intevienen */
CiC → nomC
CiE → nomE
dirProp → padron, ciudad /* Esta y la siguiente se cumplen porque tanto la dirección como el padrón y la ciudad identifican a la propiedad */
padron, ciudad → dirProp
dirProp, fecha → CiE /* la transacción queda identificada por la propiedad y la fecha. Además para esa transacción sólo puede haber un escribano */
dirProp,fecha,CiC → aporte /* el precio es por cada comprador de la transacción */

b) Determine todas las claves. Justifique su respuesta.

SOL:
Sólo a la Izquierda= {CiC,CiV,fecha}
Nunca a la Izquierda= {nomV,nomC,nomE}
En los dos lados = {dirProp,padron,ciudad,CiE}

{CiC,CiV,fecha}+={CiC,CiV,fecha,nomV,nomC} por lo que CiC,CiV,fecha no es superclave y por lo tanto tampoco es clave.

Combinando con atributos que pueden estar de los dos lados, calculamos los siguientes:

{ CiC,CiV,fecha,dirProp}+= { CiC,CiV,fecha,dirProp,nomV,nomC,aporte,CiE,nomE}

Por lo que encontramos una clave: CiC,CiV,fecha,dirProp
Como dirProp y la pareja padron-ciudad se determinan mutuamente, entonces CiC,CiV,fecha,padron,ciudad también es una clave.

Sólo queda verificar la siguiente clausura:
{ CiC,CiV,fecha,CiE}+={ CiC,CiV,fecha,CiE,nomC,nomV,nomE}

con lo cual se verifica que las únicas claves son:

{ CiC,CiV,fecha,dirProp} y { CiC,CiV,fecha,padron,ciudad}

c) Determine la forma normal en función de sus dependencias. Justifique su respuesta.

SOL:
dirProp,fecha,CiV,CiC es una clave. Por este motivo, la dependencia
CiV-→nomV es una dependencia parcial de una clave sobre un atributo no primo. Por esto, está en 1NF

d) Indique si en R se cumple o no la siguiente dependencia multivaluada embebida.

dirProp, fecha -->> CiV | CiC

Justifique.

SOL:

Esto es porque en una transacción, cada vendedor de una propiedad le vende a todos los compradores.

e) Dado el siguiente subesquema de R: R_1, proyecte las dependencias funcionales y multivaluadas.

R_1 (CiV, CiC, dirProp, padron, ciudad, fecha, aporte)

SOL:
F_1={ dirProp → padron, ciudad; padron, ciudad→dirProp; dirProp,fecha,CiC → aporte }

Todas las dependencias se proyectan directamente.

f) Aplique el algoritmo para obtener una descomposición de R_1 en 4nf. Justifique cada paso. Si las hay, tenga en cuenta las dependencias multivaluadas embebidas. Indique qué dependencias se pierden o si no se pierden dependencias.

SOL:

R_1(CiV,CiC, dirProp,padron,ciudad,fecha,aporte)
R_{11}(dirProp,fecha,CiC,aporte)
F_{11}={ dirProp,fecha,CiC → aporte}
FN: 4NF porque sólo hay dependencias funcionales de clave y no hay multivaluadas.

R_{12}(CiV,CiC, dirProp,padron,ciudad,fecha)
F_{12} ={ dirProp → padron, ciudad; padron, ciudad→dirProp; dirProp,fecha-→ > CiC}
FN: Por argumentos similares a los usados en las descomposiciones anteriores, las claves siguen siendo las mismas. Por esto, los lados derecho de todas las dependencias son primos por lo que no hay dependencias que violen 3NF. Sin embargo, ninguna de las dependencias tiene una clave del lado izquierdo por lo que todas viola BCNF.

La siguiente descomposición la hacemos según dirProp → padron, ciudad obteniendo:

R_{22}(CiV,CiC, dirProp,padron,ciudad,fecha)
R_{221}(dirProp,padron,ciudad)
F_{221}={dirProp → padron, ciudad; padron, ciudad→dirProp }
FN: 4NF dado que las dos dependencias son de clave y no hay multivaluadas.

R_{222}(CiV,CiC, dirProp,fecha)
F_{222}={ dirProp,fecha-→ > CiC} Esta dependencia es la multivaluada embebida dado que estamos en el esquema correcto para que se convierta en efectiva.

FN: La dependencia multivaluada que queda viola 4NF por lo tanto está en BCNF.

Pérdida de dependencias: no se perdieron dependencias.

La última descomposición es la siguiente:
R_{2221}(CiC, dirProp,fecha)
F_{2221}={ dirProp,fecha-→ > CiC}
FN; 4NF dado que la única dependencia es una multivaluada trivial.

R_{2222}(CiV,dirProp,fecha)
F_{2222}={}
FN: 4NF dado que no hay dependencias que violen esta forma normal.

Conclusión. La descomposición que se obtiene es la siguiente:

R_{11}(dirProp,fecha,CiC,aporte) /* Aportes de compradores */
F_{11}={ dirProp,fecha,CiC → aporte}

R_{221}(dirProp,padron,ciudad) /* Propiedades */
F_{221}={dirProp → padron, ciudad; padron, ciudad→dirProp }

R_{2221}(CiC, dirProp,fecha) /* Compras */
F_{2221}={ dirProp,fecha-→ > CiC}

R_{2222}(CiV,dirProp,fecha) /* Ventas */
F_{2222}={}

Ejercicio 3 (30 puntos).

Una empresa de desarrollo de software posee un sistema para manejar los proyectos que realiza y el personal asignado a los mismos. Este sistema usa las siguientes tablas:

PROYECTOS (codProy, nombreProy, fechaInicio, tipoProy)

PERSONAL (codFunc, nombreFunc, fechaIngreso)

TAREAS (codTarea, descripción, tipoTarea)

TAREAS_PROYECTOS (codProy, codTarea)

ASIGNACION (codFunc, codProy, codTarea)

REGISTRO_HORAS (codFunc, codProy, codTarea, fecha, cantHoras)

En la base de datos no hay tablas vacias y se cumplen las siguientes dependencias de inclusión:

$\Pi_{codProy}(TAREAS\_PROYECTOS) \subseteq \Pi_{codProy}(PROYECTOS)$

$\Pi_{codTarea}(TAREAS\_PROYECTOS) \subseteq \Pi_{codTarea}(TAREAS)$

$\Pi_{codFunc}(ASIGNACION) \subseteq \Pi_{codFunc}(PERSONAL)$

$\Pi_{codProy,codTarea}(ASIGNACION) \subseteq \Pi_{codProy,codTarea}(TAREAS\_PROYECTOS)$

$\Pi_{codFunc,codProy,codTarea}(REGISTRO\_HORAS) \subseteq \Pi_{codFunc,codProy,codTarea}(ASIGNACION)$

Se pide:

1. Resolver las siguientes consultas en **álgebra relacional**.

a) Obtener el código y nombre de los proyectos que sólo tienen tareas de tipo *Desarrollo*.

Proyectos con tareas de tipo desarrollo

$A = \Pi_{codProy}(TAREAS\_PROYECTOS * (\sigma_{tipoTarea="Desarrollo"} TAREAS))$

Proyectos con tareas de tipo diferente a desarrollo

$B = \Pi_{codProy}(TAREAS\_PROYECTOS * (\sigma_{tipoTarea<>"Desarrollo"} TAREAS))$

$R = \Pi_{codProy,nombreProy}(PROYECTOS * (A-B))$

b) Obtener el nombre de los funcionarios que en cada uno de los proyectos de tipo *Consultoría* existentes se encuentran asignados a todas las tareas de tipo *Gestión* de ese proyecto.

Proyectos de consultoría

$A = \Pi_{codProy}(\sigma_{tipoProy="Consultoria"} PROYECTOS)$

Tareas de gestión

$B = \Pi_{codTarea}(\sigma_{tipoTarea="Gestión"} TAREAS)$

Parejas (proy,tarea) con tareas de Gestión en proyectos de Consultoria

C= (TAREAS_PROYECTOS *A)*B

$R = \Pi_{nombreFunc}(PERSONAL * (ASIGNACION \% C))$

2. Resolver las siguientes consultas en **cálculo relacional**.

c) Obtener el nombre y fecha de inicio de proyectos que tienen tareas definidas para las cuales aún no hay personal asignado.

{p.nombreProy, p.fechaInicio / PROYECTOS(p) ∧

(∃ tp) (TAREAS_PROYECTO(tp) ∧ tp.codProy=p.codProy ∧

¬ (∃ a) (ASIGNACION(a) ∧ a.codProy=tp.codProy ∧ a.codTarea=tp.codTarea))}

d) Obtener el nombre de los funcionarios que en el mes de noviembre de 2016 han registrado horas para todas las tareas que tienen asignadas. Su solución sólo debe considerar a los funcionarios que tienen alguna tarea asignada. *(*Asuma que cuenta con las funciones *mes* y *año* que reciben una fecha y devuelven respectivamente el mes y año de la misma).

{p.nombreFunc / PERSONAL(p) ∧

(∀ a)(ASIGNACION (a) ∧ a.codFunc =p.codFunc →

(∃ r) (REGISTRO_HORAS(r) ∧ r.codFunc = a.codFunc ∧ r.codProy = a.codProy ∧ r.codTarea = a.codTarea and MES(r.∧)=”Noviembre” ∧ AÑO(r.fecha)=2016)) ∧

(∃ a1)(ASIGNACION (a1) and a1.codFunc =p.codFunc)}

3. Resolver la siguiente consulta en **SQL** (sin utilizar vistas)

a) Obtener para cada proyecto su nombre, la cantidad de personas asignadas por tipo de tarea, y el total de horas registradas por tipo de tarea. Sólo incluir tareas para las cuales hay registros de horas en ese proyecto.

```
SELECT nombreProy, tipoTarea, count(distinct codFunc), sum(cantHoras)
FROM TAREAS T NATURAL JOIN ( PROYECTOS P NATURAL JOIN REGISTRO_HORAS)
GROUP BY codProy,  nombreProy, tipoTarea
```

Ejercicio 4 (15 puntos)

La siguiente realidad corresponde a una cadena de cines para la cual se registran todas las funciones que son emitidas. Las tablas más importantes de la base son las siguientes:

Películas (idPel, nombre, genero, origen, actorPpal): contiene la información de las películas.
Cines (idCine, nombre, ciudad, direccion): contiene la información de todos los cines.
Funciones (idCine, idPel, dia, hora): contiene información de todas las películas que se proyectan en los cines.

Sea la siguiente consulta SQL:

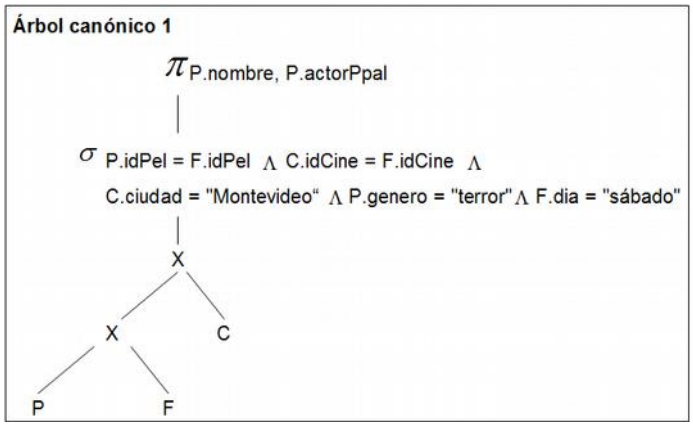
SELECT P.nombre, P.actorPpal
FROM Películas P, Cines C, Funciones F
WHERE P.idPel = F.idPel and C.idCine = F.idCine
and C.ciudad = "Montevideo" and P.genero = "terror" and F.dia = “sábado”;

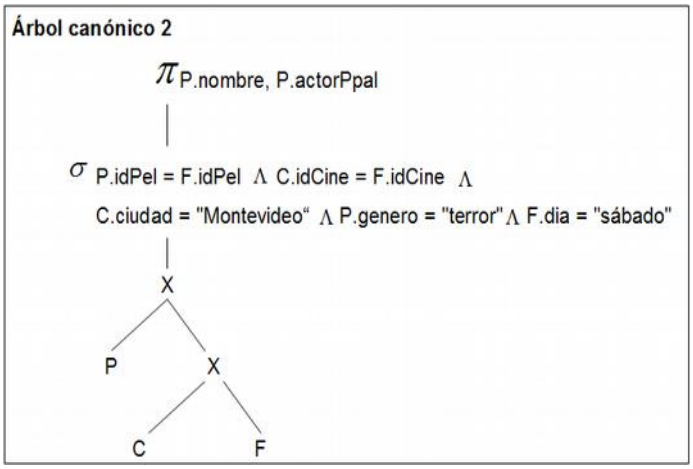
Además, se cuenta con la siguiente información:

Relación R	n_R	Dist. Unif. de valores	Índices
Películas P	900	V(genero, P) = 15	- Índice primario sobre idPel - Indice Sec. árbol b+ sobre genero
Cines C	100	V(ciudades, C) = 19	- Índice primario sobre idCine - Indice Sec. árbol b+ sobre ciudad
Funciones F	16000	V(día, F) = 7	- Indice Primario sobre idCine, idPel, dia, hora

Se pide:

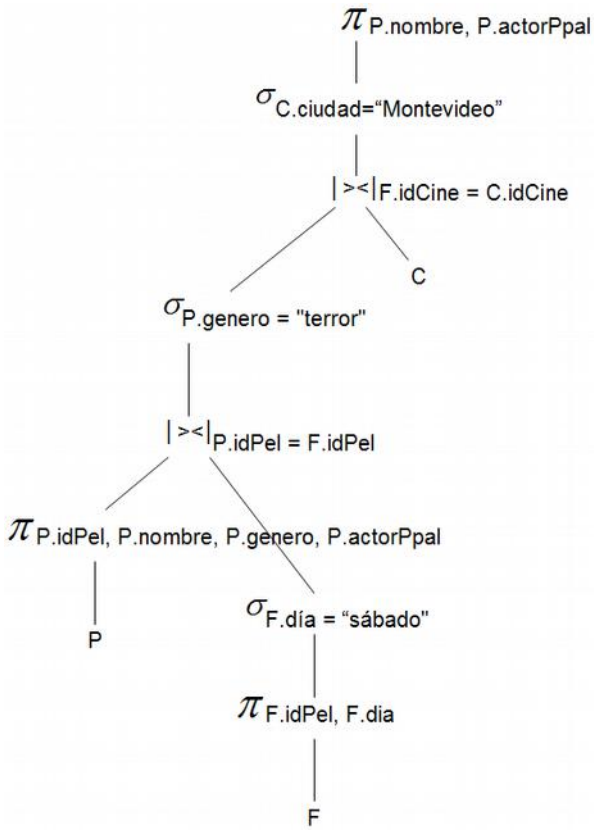
1. Indicar si alguno de los siguientes árboles es el árbol canónico de la consulta sql dada. En caso negativo justifique su respuesta.





Ninguna de las 2 opciones es correcta. En el primer caso, porque no respeta el orden de las tablas, según la consulta SQL. En el segundo caso, porque el árbol está recostado a la derecha, mientras que el árbol canónico siempre debe estar recostado a la izquierda.

2. Se sabe que un optimizador generó el plan lógico que se presenta a continuación. ¿Considera que este plan lógico es el plan lógico optimizado según las heurísticas? En caso afirmativo, justifique su respuesta. En caso negativo, describa los cambios que ud. haría para obtener el plan lógico optimizado según las heurísticas, mostrando el plan lógico final.



Este plan lógico no es el optimizado según las heurísticas. Para lograr el plan lógico optimizado es necesario aplicar las heurísticas al árbol canónico correcto. En primer lugar, debemos cambiar las selecciones conjuntivas por una cascada de selecciones simples, esto ya lo cumple el plan dado. Luego, debemos mover las selecciones lo más abajo que se pueda en el árbol. P.genero="terror" y C.ciudad="Montevideo" no lo cumple, por lo que estas selecciones deben ser bajadas lo más posible. Se deben poner a la izquierda de los productos las hojas que generen menos tuplas. Para esto, debemos tener en cuenta los siguientes valores:

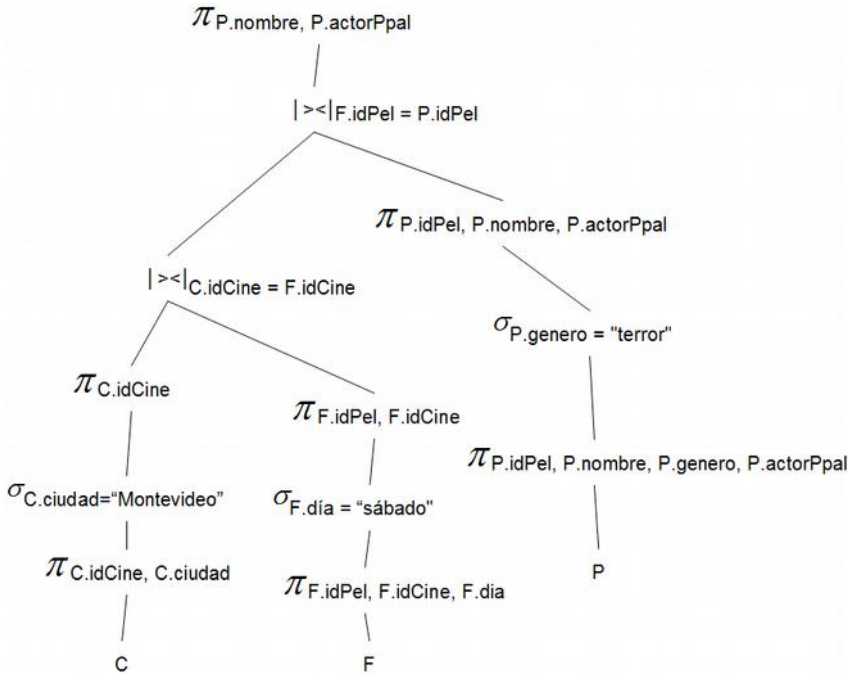
|P| = 900
V(genero, P) = 15
900/15 = 60 tuplas

|C| = 100
V(ciudades, C) = 5
100/19 =

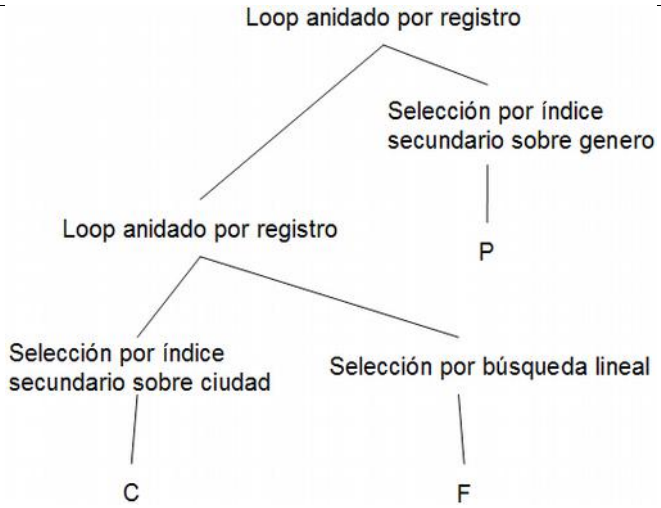
|F| = 16000
V(día, F) = 7
16000/7 = 2286

A partir de estos resultados y asegurando que el orden de las hojas no cause operaciones de producto cartesiano, se llega a la conclusión que el orden de las tablas debe ser *Cines, Funciones* y *Películas.* Además, se debe tener en cuenta que el árbol debe continuar estando recostado a la izquierda.
Unaa vez establecido el orden de las tablas, se cambia la secuencia de selecciones y productos por joins, este paso ya está realizado en el plan dado. Finalmente, se mueven las proyecciones lo más abajo posible en el árbol, agregando todas las proyecciones que sean necesarias. Se debe tener en cuenta que la última proyección sobre *Funciones* es incorrecta, ya que falta proyectar *idCine*, por lo tanto, la misma debe ser corregida.

Por lo tanto, el plan lógico correcto es el que se presenta a continuación:



3. Para el plan lógico optimizado que ud. considera correcto, dar un plan físico en el cual, siempre que sea posible, se utilicen los índices dados.



Para el plan físico contamos con índices secundarios árbol b+ sobre *genero* y *ciudad* en las tablas *Películas* y *Cines* respectivamente, por lo tanto es posible realizar una selección por índice secundario en ambos casos. En la tabla Funciones sólo tenemos un índice primario sobre la clave *idCine, idPel, dia, hora* de esta manera, solo podemos aplicar una selección por búsqueda lineal. Para los operadores de join, debemos tener en cuenta que no conocemos el número de buffers, por lo tanto no podemos considerar el loop anidado por bloques. Por otro lado, después de realizar la selección se pierden los índices, por tanto tampoco es posible utilizar el operador index join. Finalmente, la opción correcta es el loop anidado por registros.