Guía de Ejercicios # 7:
SQL Avanzado
Versión del 14/05/2010

Introducción a las bases de datos

UNQ

1. Productos, clientes y facturas

Se parte de este esquema

```
producto <nomProd, stkMinimo>
stock <nomProd, deposito, cantidad>
precioProducto <nomProd, desdeFecha, hastaFecha, precio>
cliente <codCli, nomCli, localidad, compraMaxima, codCliContacto>
(codCliContacto puede ser null)
factura <numFact, fecha, importe, codCli, deposito>
itemFactura <numFact, nomProd, cantidad>
deposito <deposito, direccion, codPostal>
```

1.1. Queries para practicar joins y cuestiones básicas

Qué practicamos: condiciones, join natural, operaciones en una fila, ordenamiento, IFNULL, join.

Serie 1

1. código y nombre de los clientes que compraron Bidu, sin repetidos, ordenados x nombre.
   idem que compraron Bidu en junio de 2008
   idem que hicieron facturas sobre el deposito de Dominico.
2. `<codCli, nomCli, numFact, fecha, importe, compraMaxima, diferencia>` para las facturas que superan el máximo del cliente, ordenadas x nro de factura.
3. `<codCli, nomCli, numFact, fecha, cantidad>` para las compras de Bidu, ordenadas x fecha y cod cliente.
   - lo mismo ordenado x cod cliente y fecha.
   - lo mismo ordenado x cantidad de la compra más grande a la más chica.

Serie 2

1. `<prod, stkmin, codCli, nomCli, numFact, fecha, cant>` para los productos de stk minimo menor a 20, ordenado x producto y fecha.
2. código y nombre de clientes que compraron Bidu o Vitina, sin repetidos.
3. para cada cliente: código, nombre, código de contacto o "SIN CONTACTO".

Serie 3

1. `<numFact, deposito, producto, cantidad en la factura, stock en el depósito de la factura>` para cada item de cada factura de menos de 100 pesos, ordenado por numFact y nombre producto.
2. `<codCli, nomCli, compraMaxima, nomProd, precio * 1000>` para cada combinación tal que el cliente puede comprar 1000 unidades del producto, tomando los precios al 01/05/2008.
3. `<prod, cant, unitario, total item>` para los items de la factura 1.

1.2. Queries para practicar alias de tablas y unión

1. `<nro de factura, fecha, cantidad de bidu, cantidad de vitina>` para las facturas que incluyen ambos productos.
2. `<numFact, prod1, prod2, cantidad>` para las facturas que tienen la misma cantidad de dos productos (me los va a tirar al derecho y al revés, y todas las combinaciones, todo OK)
3. `<código, nombre, código del contacto, nombre del contacto>` para los clientes con contacto.
4. nombre y stock mínimo para todos los productos que: o bien tienen más de 20 unidades en el depósito de Dominico, o bien el cliente 1 hizo una compra por más de 20 unidades. Ordenar por nombre del producto. Sin repetidos.
5. Comparativo del stock de Dominico y Bernal, o sea `<nomProd, stockDominico, stockBernal>`, para los productos con stock en ambos depósitos.

1.3. Queries para practicar left join y right join

1. Comparativo del stock de Dominico y Bernal, o sea `<nomProd, stockDominico, stockBernal>`, incluyendo todos los productos, poniendo 0 donde no tengan stock.
2. `<código, nombre, código del contacto, nombre del contacto>` para todos los clientes, dejando null en la info de contacto para los clientes que no tienen contacto.
3. lista de precios al 01/06/2008 incluyendo los productos sin precio, ordenada por nombre del producto.

1.4. Queries para practicar agrupamiento

1. `<fecha, numFact, importe, cant items, cant total unidades, suma de los importes a precios del día de la factura>` ordenado por número de factura.
   - Fijarse qué pasa si algún producto no tiene precio a la fecha de la factura.
2. `<nombre producto, cantidad total de unidades, en cuántos depósitos hay stock>` ordenado por nombre de producto.
   - Fijarse qué pasa con los productos sin stock.
3. ranking de facturación `<cliente, importe total facturado>`
   - idem para junio de 2008
   - idem incluyendo sólo aquellos clientes a los que se les facturó al menos 1000 pesos
   - idem incluyendo todos los clientes, aún los que no tienen facturas

1.5. Queries para practicar subselect

1. nombre de los productos con stock en Dominico y no en Bernal.
2. código y nombre de los clientes cuya compra máxima supera a la suma de los importes de las facturas que se les hicieron.
3. clientes a los que se les hizo al menos una factura, y que no se les hizo ninguna factura de menos de 20 pesos.

1.6. Queries para practicar vistas

1. crear una vista con nombre de cliente, localidad del cliente y facturación total para todos los clientes cuya facturación en el año 2008 haya superado los 2000 pesos
2. crear una vista con nombre de producto, stock mínimo y stock actual para todos los productos

1.7. Desafío

`<depósito, producto con más unidades, unidades de ese producto, total unidades en el depósito>` para cada depósito.

2. ONG

Una ONG tiene que manejar la información sobre las tareas que llevan a cabo sus voluntarios y colaboradores externos. De cada tarea se sabe: el nombre, dónde se hace, cuántas horas por semana conviene dedicarle como mínimo, y cuántas horas por semana se le puede dedicar como máximo. Puede pasar que varias tareas se hagan en el mismo lugar, pero no se contemplas que una misma tarea se distribuya en varios lugares.

Cada tarea puede tener un coordinador, que es un voluntario,que puede o no trabajar en esa tarea.

Cada voluntario puede trabajar en varias tareas, para cada una se sabe cuántas horas le dedica por semana. En una tarea pueden trabajar varios voluntarios. De cada voluntario se sabe: nombre, apellido, email, país donde vive, e idiomas que habla. De algunos voluntarios se desconoce el país donde viven.

De los colaboradores externos también se sabe nombre, apellido y email, y en qué tareas trabajan. No se controla las horas que trabajan, los cargamos sólo para poder contactarlos. Se toma como supuesto semántico que no puede haber dos personas que compartan su email.

De los lugares donde se hacen tareas se debe manejar: sus coordenadas (latitud norte/sur y longitud este/oeste), su medio de acceso (auto, 4x4, camión, avión, barco, tren), si tienen lugar de alojamiento o no, y en qué país están. De cada lugar se conoce un nombre que lo identifica.

Se tiene que manejar también: el continente en que está cada país, los idiomas que se hablan en cada país, y el grado de conflictividad de algunos países (1: todo tranquilo, 10: guerra civil, de algunos países no se tiene esta información).

El esquema relacional que se eligió para representar esta información es

```
   tarea     <nomTarea, codLugar, horasMin, horasMax,
             emailVolCoord>
 voluntario  <emailVol, nombre, apellido, nomPais>
 volIdioma   <emailVol, idioma>
colaborador  <emailCol, nombre, apellido>
 volTrabaja  <nomTarea, emailVol, horas>
 colTrabaja  <nomTarea, emailCol>
   lugar     <codLugar, latitud, latNS, longitud, longEW,
             medioAcceso, tieneAlojamiento, nomPais>
    pais     <nomPais, gradoConflictividad, continente>
 paisIdioma  <nomPais, idioma>
```

Esto es lo que hay que hacer:

1. De todos los atributos incluidos en la BD, indicar de acuerdo a la explicación sobre la semántica: cuáles no pueden ser null, cuáles sí pueden ser null, y en cuáles el enunciado deja la decisión al diseñador de la BD.

2. Indicar cómo hacer las siguientes consultas en SQL. Consejo: háganlo en máquina, eso los va a ayudar a corregir errores.

   En el blog hay un script que genera las tablas y un juego de datos de ejemplo que da respuestas a todas las consultas que se piden, se llama `ong.sql`.
   Les puede ser muy útil estudiar los datos que están incluidos en la instancia de ejemplo, para cada consulta llegar "por afuera" a la conclusión de qué resultado debería dar, y compararlo con lo que efectivamente les da.

   Algunas consultas son más difíciles que otras, y no están en orden de dificultad, si una te cuesta mucho, pasá a la siguiente, y después podemos ver en clase las que no hayan salido.

   a) El nombre y continente de los países conflictivos (con grado de conflictividad > 7)

   b) Nombre y apellido de los voluntarios y colaboradores externos (todos juntos) que trabajan en tareas a las que se accede en 4x4.

   c) Nombre de los países en los que se hacen tareas en las que algún voluntario trabaje más de 20 horas por semana.

d) Nombre y país de las tareas en las que trabajan tanto voluntarios como colaboradores externos, sin repetidos, ordenados por nombre de país y nombre de tarea dentro del país.

e) Nombre, apellido y email de las personas que trabajan sin coordinador en al menos una de sus tareas, sin repetidos, ordenados por apellido y nombre.
Para esta consulta, indicar

- qué condición/es sobre los datos provocarían respuestas repetidas de no pedir explícitamente que no aparezcan repetidos.
- qué debería cambiarse en esta consulta si el atributo de la fk de tarea a voluntario se llamara `emailVol` en lugar de `emailVolCoord`.

f) Nombre y continente de los países en los que hay al menos un voluntario que no está trabajando en ninguna tarea.

g) Nombre y apellido de los colaboradores que trabajan en más de dos tareas.

h) Una tabla de asignaciones de voluntarios con estas columnas: nombre, apellido y email del voluntario; nombre, coordenadas y modo de acceso del lugar donde se hace la tarea; nombre, grado de peligrosidad, y continente del país.
En esta consulta hay un atributo compartido entre tablas que imposibilita usar solamente join natural. Resolverlo de dos formas distintas (o sea, armar dos consultas con el mismo resultado): una usando subselect, y la otra usando join moñito.

i) Indicar a qué países podría ir un voluntario, o sea pares `<emailVol, nomTarea>`. Un voluntario puede ir a un país si habla algún idioma que se hable en ese país, y no es su país de residencia.

j) Armar las posibles asignaciones de voluntarios a tareas (pares `<emailVol, nomTarea>`), donde el voluntario debe hablar alguno de los idiomas del país de la tarea, en este caso no importa si es su país de residencia o no. No importan las asignaciones actuales.

k) Agregar a la consulta anterior la condición de que el voluntario debe vivir en el mismo continente donde se hace la tarea.

l) Agregar a la consulta anterior la condición de que el voluntario no esté ya asignado a la tarea.

m) Una tabla `<nombre tarea, cantidad total de horas que trabajan los voluntarios en la tarea, horasMax, apellido del coordinador>` para cada tarea.
Sobre esta consulta indicar: qué tareas podrían no aparecer en el resultado, cómo hacer para que sí aparezcan.

n) Nombre de las tareas que no tienen coordinador y que además están pasadas de horas, o sea, en la que los voluntarios trabajan en total más horas del máximo indicado para la tarea.
Nota que puede ser útil: en el `having` sólo se pueden poner los atributos que aparecen en el `select`, o funciones de agrupación onda `sum, max, min, avg`. Entonces, si quiero poner un atributo en el `having` que no le pasa ninguna de las dos cosas, en este caso `horasMax`, una forma es poner `min(horasMax)` que va a ser lo mismo, porque para cada grupo va a ser el mismo valor en todas sus filas componentes.

ñ) La mínima latitud de los lugares en Argentina. Dado que Argentina está toda en el hemisferio sur, es la latitud del lugar en Argentina que esté más al norte.

o) El nombre del lugar más al norte de Argentina.

p) El nombre de los países que tienen al menos un lugar en el hemisferio norte y otro en el hemisferio sur.

q) Apellido de los voluntarios que trabajan en la tarea 'tb1' la misma cantidad de horas que las que ocupa el voluntario cuyo mail es 'lucia@abc.com' en la tarea 't90'.

r) Una tabla `<emailVol, tarea1, horas1, tarea2, total horas 2>` para cada caso en el que el voluntario trabaje en la tarea 1 más que el total de horas ocupadas (o sea, suma de las horas que trabajan los voluntarios) en la tarea 2.
P.ej. si Pepe trabaja 15 horas en la tarea 1, y en la tarea 2 trabajan solamente Juan 5 horas y Lucas 7 horas, entonces la tupla `<Pepe, tarea 1, 15, tarea 2, 12>` debe incluirse en la respuesta.

s) Crear una vista con país de tarea, nombre de tarea, horas máximas de tarea, cantidad de voluntarios y total de horas de trabajo de los voluntarios para todas las tareas cuyo total de horas supere la máxima.

t) Una tabla `<codLugar, seHablaCastellano, seHablaPortugues>` donde para cada lugar se indique "si" o "no" en los atributos `seHablaCastellano` y `seHablaPortugues`; se establece que en todos los lugares de un país se hablan todos los idiomas que se hablan en ese país.

u) Una tabla `<emailVol, continente, horas>` indicando cuántas horas tiene asignadas cada voluntario en cada continente. Incluir todas las combinaciones `<emailVol, continente>`, poniendo 0 en los casos en que un voluntario no haga tareas en un continente.

3. Transporte marítimo

Tenemos esta base que refleja parte de la operatoria de la compañia de transporte marítimo "La segunda del docke"

```
barco <nomBarco, paisBandera>
bodega <nomBarco, numBodega, capacidad, nivelProf>
puerto <nomPto, pais>
viaje <nomBarco, nomPtoOrigen, nomPtoDestino, fini, ffin>
producto <nomProd, precioKg, gradoPel>
carga <nomBarco, numBodega, fini, nomProd, kg>
```

En el blog hay un script que genera las tablas y un juego de datos de ejemplo, tal vez no muy completo, pero que al menos les sirve como base. Se llama `lasegundadeldocke.sql`.

Se pide resolver las siguientes consultas en SQL

1. barcos (nombre y país de bandera) que zarparon de Brasil en 2008 sin repetidos, ordenados por nombre.
2. para todos los viajes de un determinado barco: `<fini, ffin, ptoOrigen, ptoDestino, totalKg>`
   Sobre esta consulta indicar bajo qué condición un viaje no aparecería, y qué agregado hacer a la consulta para que aparezca.
3. agregar a la consulta anterior el precio total de cada viaje, según el precio por kg de cada producto
4. Aplicar las siguientes variantes a la consulta anterior por separado.
   - que sólo aparezcan los viajes cuyo precio total es mayor a 10000 pesos
   - que sólo aparezcan los viajes de más de 10 días de duración (función `datediff`)
   - tener en cuenta para cada viaje solamente los productos peligrosos (grado de peligrosidad mayor a 7) Indicar qué viajes dejan de aparecer al agregar esta condición
   - tener en cuenta para cada viaje solamente los productos no peligrosos; si no se informa grado de peligrosidad, se presume que un producto no es peligroso.
5. Una lista `<nomBarco,nomPto>` que relaciona cada barco con cada puerto que tocó (ya sea habiendo zarpado o partido), sin repetidos, y ordenado por barco y puerto.
6. Los nombres de los puertos por los que pasó (o sea, zarpó y/o llegó) mercadería peligrosa.
7. Los barcos que nunca tocaron puertos por los que haya pasado mercadería peligrosa
8. Una lista `<nomBarco, cantidad de bodegas, capacidad de la bodega más grande>`
9. Para un determinado viaje, una lista `<numBodega, capacidad, total kg>`, incluyendo todas las bodegas del barco.
10. Para un determinado viaje, una lista `<nivelProf, total kg>`.
11. Una lista `<nomBarco, máxima carga que llevó>` para cada barco, 0 para los barcos que no hayan hecho ningún viaje.
12. El nombre del barco que tiene la bodega más grande.
13. Una tabla `<nomBarco, primer puerto que tocó, último puerto que tocó>`
    O sea, para cada barco, origen del primer viaje y destino del último viaje.
14. Los nombres de los puertos que nunca hayan sido tocados por un barco de bandera brasileña (o sea, que nunca haya ni zarpado ni llegado un barco de bandera brasileña a ese puerto).

15. Los nombres de los puertos por los que pasaron más de 5 barcos.

16. Los nombres de los barcos que el 01/06/2008 estaban en viaje.

17. Los nombres de los barcos que el 01/06/2008 **no** estaban en viaje.

18. La lista `<nomBarco, nivelProf, capacidad total>` para cada combinación barco/nivel de profundidad cuya capacidad total sea mayor que la capacidad total del barco *granlily* (reemplazar por algún nombre de barco que esté en tu base).
    P.ej. si la capacidad total del barco *granlily* (sumando todas sus bodegas) es de 25000 kg, y el barco *zanganga* tiene estas bodegas

Nro. bodega	nivel de profundidad	capacidad
101	1	13500
104	1	8200
203	2	7400
208	2	9200
284	2	9500
304	3	1500
305	3	4500
306	3	5500
434	4	3900

    entonces la fila `<zanganga,2>` debe estar en la respuesta, porque el total de capacidad del nivel de profundidad 2 de *zanganga* es mayor a la capacidad total de *granlily* ($26100 > 25000$).

19. Crear una vista con nombre de barco, cantidad de viajes y total de kilos de carga para cada barco en el 2008

20. Crear una vista con nombre de barco y capacidad ociosa total en sus bodegas

4. Biblioteca

Tenemos esta base que refleja parte de la operatoria de una biblioteca pública

```
socio <numSocio, nomSocio, nomBarrio>
libro <isbn, nomLibro, nomAutor, genero, idioma, anioPublicacion>
prestamo <numSocio, isbn, fechaPrestamo, dias, fechaDevolucionReal>
autor <nomAutor, nomPaisNac, anioNac>
lecturaEnSala <numSocio, isbn, fecha>
```

Los géneros que tendremos en cuenta son: 'novela', 'cuentos', 'poesia', 'historia', 'arte'.

En el blog hay un script que genera las tablas y un juego de datos muy chiquito, sí, hay que completarlo (y qué bien vendrían voluntarios). Se llama `biblio.sql`.

Se pide

1. Se quiere hacer estas consultas acerca de los hábitos de lectura de los socios de la biblioteca.

   a. Para cada libro de Cortázar, nombre y cuántas lecturas tuvo en 2008.

   b. Nombre de los socios que leyeron a Cortázar, pero no a Borges, en 2007.

   c. Nombre de los socios que leyeron novelas de autor argentino en junio de 2007.

   d. Ranking 2007 de socios por cantidad de lecturas, `<nombre, cant. de lecturas>`.

   e. Nombre de los socios que leyeron tanto a Benedetti como a Joyce en 2007.

   En todos los casos, y también en lo que resta del ejercicio, "leer un libro" quiere decir: o bien haberlo retirado, o bien haberlo leido en sala.
   Una "lectura" es un usuario que leyó un libro (según el criterio de "leer" recién definido), o sea, si dos usuarios leyeron el mismo libro, o incluso si el mismo usuario leyó dos veces el mismo libro, se cuenta como dos lecturas. Para los préstamos se toma la fecha de préstamo.

   Resolver estas cinco consultas, definiendo una vista que resuelva una parte que es común a las cinco.

2. Se quiere hacer estas consultas acerca de las preferencias de lectura en cada barrio, consideradas por géneros:

   a. Barrios en los que hubo más lecturas de más novelas que cuentos en 2007.

   b. En qué años hubo más de 5 lecturas de novelas en Bernal.

   c. Ranking de los barrios según la cantidad de lecturas de novelas en 2007.

   d. evolución anual de la lectura de poesía en Bernal.

   Resolver estas cuatro consultas, definiendo una vista que resuelva una parte que es común a las cuatro. Vale usar la vista definida en el punto anterior,

3. Resolver estas consultas, en las que vale usar las vistas definidas en los puntos anteriores. Cuando se pide "libro" es `<isbn, nomLibro>`; cuando se pide "socio" es `<numSocio, nomSocio>`.

   a. Libros que o bien son de autor francés, o bien se leyeron en sala al menos una vez en junio de 2008.

   b. Libros leidos por el usuario 1 y que no están en la lista anterior.

   c. `<numSocio, nomSocio, fecha en que retiró Ficciones, fecha en que retiró Rayuela>` para los socios que hayan retirado ambos libros.

   d. Socios que hayan retirado algún libro más de una vez.

   e. `<autor, novela, año de publicación, libro de cuentos, año de publicación>` para cada libro de cuentos de un autor publicado después de una novela del mismo autor (si un libro de cuentos se publicó después de 3 novelas, OK que aparezcan 3 filas en el resultado).

f. `<autor, libro de cuentos, año de publicación>` para los libros de cuentos de un autor publicados después de su última novela.

g. `<nomSocio, libro, fecha en que lo tenía que devolver>` para los préstamos vencidos. Tener en cuenta estas funciones de MySQL: `date(sysdate())` que devuelve la fecha actual, `addDate(fecha, cant. dias)` que suma una cantidad de días a una fecha.

h. Socios y cantidad de lecturas de cada uno en 2007, poniendo cero en los que no hayan leido nada.

i. Socios que en 2007 leyeron más que todo Paternal.

j. `<libro, año primer libro de cuentos, año primera novela>` incluyendo todos los autores (aunque no hayan escrito cuentos y/o novelas).

5. Cajas chicas

ACME inc. quiere hacer un seguimiento de su operatoria de cajas chicas.
¿Cómo se manejan?

- Hay muchas cajas chicas, se las identifica por un número.
- Una persona puede hacer un gasto adelantando el importe, y después lo rinde. Eso genera una fila en la tabla `gasto`.
- Una persona también puede hacer un retiro a rendir más adelante. Cuando hace el retiro se genera una fila en la tabla *retiro* donde se indica que está pendiente de ser rendido poniendo `estaRendido = 'N'`, y cuando la persona rinde el gasto se marca el retiro como rendido poniendo `estaRendido = 'S'` y se insertan los gastos correspondientes.

Quieren incluir en la misma BD un sistema de seguimiento de los toners de impresora láser: cuando se compra un toner se inserta una fila en la tabla `toner`, cuando se instala se le asigna a esa fila su `fechaInstalacion`, y cuando se cambia por uno nuevo, al que se está sacando se le asigna su `fechaReemplazo`.

El esquema completo es

```
caja <numCaja, nomAdmistrador, saldo>
retiro <numCaja, fecha, nomResp, importe, estaRendido>
gasto <numGasto, numCaja, fecha, nomResp, concepto, importe>
toner <modelo, numSerie, codImpresora, fechaCompra, fechaInstalacion, fechaReemplazo>
modeloToner <modelo, precioAproximado>
dondeTrabaja <nomEmpleado, nomDepto>
```

Los datos `nomAdministrador`, `nomResp` (por "responsable") y `nomEmpleado` (en `dondeTrabaja`) se refieren al mismo concepto. Administrador y responsable no son fk a `dondeTrabaja` porque hay personas de las que no se conoce el departamento donde trabaja, o trabaja en varios y no se puede determinar uno principal.
La pk de `gasto` es un número que se asigna automáticamente para cada gasto.
El `nomDepto` de cada empleado es el nombre del departamento de ACME Inc. donde trabaja.

Se pide resolver los siguientes requerimientos mediante consultas SQL

1. Código de las impresoras en las que se haya instalado más de tres toners durante 2007.
2. Modelo y número de serie de los toners que ya fueron reemplazados, y que duraron más de 90 días (entre instalación y reemplazo).
3. Un reporte de gastos anuales de cada caja de la forma
   `<numCaja, gastos totales en 2006, gastos totales en 2007>` Incluir **todas** las cajas. Ordenar por la suma del gasto en los dos años.
4. El disponible de cada caja, que es su saldo más el importe total de los retiros para esa caja que no hayan sido rendidos. Deben aparecer todas las cajas.
5. Todos los movimientos (gastos y retiros) hechos por el responsable "Juan Darthes" en 2008, ordenados por fecha. Para cada movimiento: tipo ("gasto" o "retiro"), fecha e importe.
6. Nombres de las personas que tengan retiros sin rendir tanto en la caja 1 como en la caja 2, al menos un retiro sin rendir en cada una. Que no aparezcan nombres repetidos en la lista. Ordenar por nombre.
7. Para cada toner, rastreo de qué gasto pudo haber sido el correspondiente a su compra. Las condiciones son: que la fecha del gasto coincida con la fecha de compra del toner, y que el importe del gasto no difiera en más de 20 pesos (en más o en menos) del precio aproximado del modelo de toner. Puede ser más de un gasto posible para cada toner, eso está OK. Si para un toner no hay ningún gasto que pudiera corresponder a su compra, que no

aparezca en el listado.
Del toner incluir modelo y número de serie. Del gasto: número de caja, importe y nombre del responsable.

8. Reporte de gastos del 2007 discriminados por departamento y concepto, o sea una tabla con este esquema: `<nomDepto, concepto, total de gastos>`
Agrupar los responsables de los que no se conoce el departamento donde trabajan bajo el nombre ‘‘`desconocido`’’.

9. Nombres de los administradores de las cajas cuyo saldo es mayor al total gastado por el departamento “Marketing” durante 2007.

10. Para la caja número 1, cuántos gastos hizo en 2008 cada persona que hizo al menos un gasto en el 2007, un esquema `<nomResponsable, cantidad de gastos>`. Ojo que el número puede ser 0, deben incluirse todas las personas responsables de gastos en el 2007.

11. Crear una vista con número de caja, nombre de administrador, saldo y monto total de retiros sin rendir para las cajas cuyos retiros sin rendir superan el saldo.

6. Trámites

Un ministerio quiere llevar un control adecuado de los trámites que hace. Para cada trámite hay que hacer varias acciones (afectar presupuestos, hacer revisión legal, hacer licitaciones, organizar reuniones, tomar determinadas decisiones, hacer estudio de impacto, pedir aprobaciones, etc.). Para hacer lo que haya que hacer, están las oficinas del ministerio, y también hay profesionales que el ministerio puede contratar puntualmente para una determinada tarea sobre un trámite.

La BD que se armó es esta

```
oficina <numOficina, nombre, nomCiudad>
profesional <codProfesional, nombre, nomCiudad>
oficinaPuedeHacer <numOficina, nomAccion>
profesionalPuedeHacer <codProfesional, nomAccion, aniosExperiencia>
tramite <numTramite, nomResponsable, fechaInicio, fechaFin>
accionNecesariaTramite <numTramite, nomAccion>
tramitePasoPor <numTramite, numOficina, fechaEntrada, fechaSalida>
```

El circuito para cada trámite es el siguiente:

- Cuando se crea un trámite, se agrega la fila correspondiente en la tabla `tramite` y se registra su fecha de inicio y responsable; la fecha de fin queda en `null`.
- Cuando el trámite entra en una oficina, se agrega la fila correspondiente en `tramitePasoPor`, registrando la fecha de entrada; la fecha de salida queda en `null`.
- Cuando el trámite sale una oficina, se registra la fecha de salida en la fila correspondiente en `tramitePasoPor`.
- Cuando el trámite se termina, se registra la fecha de fin en la fila correspondiente en `tramite`.
- Si el trámite se deriva a un profesional, eso no se registra en la BD.

Hacer las consultas en SQL que permitan obtener

1. Lista de trámites iniciados en abril de 2008, indicando en qué oficina estaban al 15/06/2008, con: número de trámite, fecha de inicio, responsable, número de oficina, nombre de la oficina; ordenado por fecha de inicio y responsable. Debe incluir todos los trámites iniciados en ese mes; para los que no estaban en ninguna oficina el 15/06/2008, poner null en número y nombre de oficina.
2. Una ficha de la historia registrada de los trámites 125 y 126, o sea de las oficinas por donde fueron pasando estos dos trámites, con estas columnas: número de trámite, responsable, número de oficina, nombre de la oficina, nombre de la ciudad donde está la oficina, fecha de entrada, fecha de salida. Ordenado por número de trámite (o sea, primero la historia del 125 y después la del 126) y fecha de entrada.
3. Supongamos que el trámite 125 debe "recomenzar de cero", o sea hacer todas las acciones necesarias para el mismo, porque las oficinas por donde pasó se equivocaron.
   Obtener los nombres de todas las oficinas y profesionales que pueden hacer algo sobre el trámite 125; o sea, que pueden hacer alguna acción necesaria para el trámite. No incluir las oficinas por las que el trámite ya haya pasado.
   Para cada fila: el tipo que puede ser 'Oficina' o 'Profesional', el nombre, la acción que puede hacer. Si una oficina o profesional puede hacer varias acciones necesarias para el trámite 125, que aparezcan varias filas, una por cada acción.
4. Código y nombre de las oficinas de Mendoza por las que pasaron más de 10 trámites cuya fecha de inicio esté en el año 2008.
5. Código y nombre de los profesionales que tienen más años de experiencia en zurcir que en bordar, o que tienen experiencia en zurcir y no en bordar. 'zurcir' y 'bordar' son dos acciones.

6. Cantidad total de tiempo en días que cada trámite iniciado en 2007 y no terminado estuvo en oficinas sumando el tiempo que estuvo en todas las oficinas por las que pasó; si el trámite está actualmente en una oficina, tomar para esa oficina el período hasta la fecha de sistema (`date(sysdate())`).
   El resultado debe tener dos columnas: número de trámite y cantidad total de días.
   Para obtener la cantidad de días entre dos fechas: `dateDiff(fechaPosterior, fechaAnterior)`.
   Verificar que aparezcan en el resultado los trámites que no hayan pasado por ninguna oficina.

7. Acciones que saben hacer las oficinas por las que pasó el trámite 126 pero que no son necesarias para ese trámite. Se espera que el resultado tenga una sola columna, las acciones sin repetidos y ordenadas alfabéticamente.
   P.ej. si el trámite 126 pasó por expedición que sabe hacer envío de material, empaquetado e impresión de etiquetas, y de estas tres acciones el trámite 126 solamente necesita empaquetado, las otras dos acciones deben incluirse en el resultado.

8. Números de los trámites en curso (i.e. no terminados) que pasaron la oficina número 9 más tiempo de lo que cualquier trámite haya pasado por la oficina número 7. No tener en cuenta en este cálculo los trámites que están actualmente en las oficinas 7 ó 9.

9. Crear una vista con el nombre y la ciudad de la oficina, cantidad de tramites y morosidad promedio para el año 2008

7. Aulas - Ejercicio de Parcial

Tenemos estas tablas del log de uso de aulas de una universidad del conurbano.
Las actividades se dividen en clases y reuniones, cada clase es de un curso, las reuniones no. Algunas reuniones corresponden a una carrera, otras a ninguna (tendrán el valor correspondiente en null).
Las horas son enteras (o sea, no vale p.ej. 16.30, es 16 o 17).

```
curso<codCurso, codMateria, nomProfe, cantInscriptos>
materia<codMateria, nomMateria, carrera>
profesor<nomProfe, fnac>
aula<numAula, piso, capacidad>
clase<numAula, fecha, desdeHora, hastaHora, codCurso, cantAlumnos>
reunion<numAula, fecha, desdeHora, hastaHora, cantAsistentes, carrera, nombreReunion>
```

Se pide resolver las siguientes consultas, teniendo en cuenta estas aclaraciones

- Les va a venir muy (pero muy) bien una vista, defínanla.
- La función `datediff(fecha2,fecha1)` devuelve la diferencia en días entre fecha1 y fecha2.
- Hay que poner estrictamente los `left` y `right` que se necesiten, no vale poner de más.

Ahí vamos

1. `<nombre de materia, cantidad de cursos, total de inscriptos>` para las materias de la TPI `(carrera = 'tpi')` con más de un curso. O sea, las materias sin cursos, o con un solo curso, **no** deben aparecer.
2. número y piso de las aulas que estaban vacías el 17/08/2008 a las 17 horas.
3. asignaciones incoherentes, o sea, que estén indicando que se ocupa la misma aula, el mismo día a horas solapadas (p.ej. de 16 a 18 y de 15 a 17) para cosas distintas.
   La tabla resultante debe tener el siguiente formato
   `<fecha, aula,`
   `tipo act 1, desdeHora act 1, hastaHora act 1, nombre act 1,`
   `tipo act 2, desdeHora act 2, hastaHora act 2, nombre act 2>`
   donde el tipo es 'C' (por curso) o 'R' (por reunión), el nombre de actividad para los cursos es la materia y para las reuniones es el nombre de reunión. Ordenar por fecha, aula y hora de la actividad 1.

   La cuenta de las condiciones es:
   (desdeHora1 > desdeHora2 y desdeHora1 < hastaHora2) ó
   (hastaHora1 > desdeHora2 y hastaHora1 < hastaHora2)
4. informe de uso de aulas
   `<piso, numAula, total gente 17/08/2008, total gente 18/08/2008>`.
   Deben incluirse todas las aulas, poner 0 las aula/día donde no haya habido actividad. Ordenado por piso y aula dentro de cada piso.
5. Reporte de cantidad de gente discriminado por carrera y piso, del 17/08/2008, o sea
   `<carrera, piso, total de gente en eventos en ese piso y para esa carrera>`. Para clases, tomar la carrera de la materia del curso. Para reuniones, la carrera informada en la reunión.
   Que aparezcan todas las combinaciones (carrera,piso), 0 si no hay actividades de esa carrera en ese piso.
   Va a haber eventos que no van a sumar, ¿cuáles son?.
6. código de curso, nombre de materia, y nombre del profesor, para los cursos a los que haya ido menos de un tercio de los inscriptos a la clase del 17/08/2008. No tener en cuenta los cursos en los que no haya habido clase ese día.

7. Informe comparativo de dos días
   `<codCurso, nombre materia, cantidad alumnos en la clase del 17/08/2008, cantidad alumnos en la clase del 21/08/2008>`
   para los cursos a los que haya ido más gente el 21/08/2008 que el 17/08/2008, o que el 21 haya habido clase y el 17 no. Si el 17 no hubo clase, poner 0 en el lugar correspondiente.

8. Tabla `<codCurso, nombre materia, cantidad inscriptos>` para los cursos que tengan más inscriptos que cualquier curso de la TPI.

9. Informe de duración de cursos
   `<codCurso, nombre materia, fecha 1ra clase, fecha última clase, lapso en días, cantidad de clases, máximo de alumnos en una clase>`
   El lapso en días es la cantidad de días entre la fecha de la última clase y la fecha de la primer clase.
   ordenarlos por lapso en días, si dos cursos tienen el mismo lapso por fecha de primera clase.

10. Informe sobre profesores
    `<nombre profesor, cantidad de carreras para las que da cursos>`
    Claro, no hay que repetir carreras.
    Lograr que aparezcan todos los profesores, poner 0 en los que no dan ningún curso.

8. Fabrica de Varillas - Ejercicio de Parcial

Ferrevari S.R.L. es el líder en fabricación de varillas en Bella Vista. Además de varillas, también fabrica chapas. Tiene varias máquinas que le sirven tanto para varillas como para chapas.
Para las varillas se definen modelos con propiedades bien conocidas y su producción se organiza en lotes; en cambio las chapas son de producción individual y no hay modelos standard. Lotes de varillas y chapas se numeran, las varillas se numeran empezando de 1 para cada lote.
Por las características de la organización de Ferrevari, es posible que en el mismo lote entren varillas de distintos modelos.

Para poder obtener estadísticas y organizar su producción, se armó el siguiente esquema de BD:
`varilla <numVarilla, numLote, codModelo, largo>`
`lote <numLote, codMaquina, fecha, responsable>`
`maquina <codMaquina, fechaInstalacion>`
`plancha <numPlancha, codMaterial, codMaquina, fecha, largo, ancho>`
`modeloVarilla <codModelo, codMaterial, ancho>`
`material <codMaterial, nomMaterial, pesoM2>`
donde todas las medidas están en centímetros.

Resolver las siguientes consultas en SQL

1. El ranking de los responsables de lotes de varillas fabricados el 01/10/2008, según el largo total de las varillas de los lotes de cada uno.
2. Todos los pares `<codMaterial, codMaquina>` indicando en qué máquina/s se hicieron varillas de cada material el 01/10/2008; si de un material no se hicieron varillas, entonces debe aparecer una sola fila con codMaquina = null.
3. Reporte de la producción de varillas del 01/10/2008 de las máquinas instaladas entre los años 1991 y 2000, discriminando por máquina y modelo. El resultado debe tener tres columnas: `<codMaquina, codModelo, superficie total de varillas fabricadas>`. Incluir solamente las máquinas que hayan fabricado al menos 1000 centímetros cuadrados de varillas ese día.
4. Qué fabricó la máquina de código 'pepa' el 01/10/2008, para cada cosa el tipo ('Varilla' o 'Plancha'), el código y el nombre del material, el largo y el ancho. Ordenado por nombre de material, ancho y largo.
5. El código de las máquinas que el 01/10/2008 hayan hecho chapas del material de código 'F3', pero no hayan hecho varillas de ese material.
6. Números de los lotes fabricados por la máquina 'pepa' que incluyan varillas de distinto modelo, ordenados de menor a mayor y sin repetidos.
7. Reporte `<codMaquina, cantidad de varillas producidas el 01/10/2008, cantidad de varillas producidas el 02/10/2008>` incluyendo todas las máquinas; poner 0 para los días en que una máquina no haya producido varillas.
8. Para las planchas del mismo material y la misma superficie (ancho * largo) fabricadas por la misma máquina en días consecutivos (p.ej. una el 01/10/2008 y la otra el 02/10/2008), los dos números de plancha, el material, la máquina y la superficie. Tener en cuenta la función
`addDate(fecha,cantDias)`.
9. Los códigos de las máquinas más viejas que 'caty' que hayan trabajado (hecho al menos una varilla o una chapa) el 01/10/2008.
10. Los números de lote fabricados por la máquina 'pepa' el 01/10/2008 cuya longitud promedio sea mayor en 3 o más cm al promedio de longitud de todas las varilas fabricadas por esa máquina en ese día.

9. Vuelos - Ejercicio de Parcial

Partiendo del siguiente esquema de BD:

```
Aeropuerto <nombreAeropuerto, ciudad, pais>
Escalas<numVuelo, numEscala, nombreAeropuerto, horaLlegada, horaPartida>
Vuelos <numVuelo, numeroSerieAvion, nombreAeropuertoSalida, horaSalida, nombreAeropuertoLlegada>,horaLlega
TipoDeAvion<nombreTipoDeAvion, cantMaxDeAsientos, empresa>
PuedeAterrizar <nombreTipoDeAvion, nombreAeropuerto >
Avion <numeroSerieAvion, nombreTipoDeAvion, totalDeAsientos>
```

Donde un vuelo puede tener o no tener escalas, consideramos escala a los puntos intermedios entre el Aeropuerto de salida y el aeropuerto de llegada. Si un vuelo es directo, entonces no tiene ninguna escala. Ej: Si el vuelo desde Ezeiza al aeropuerto Mariscal Sucre de Quito tiene una escala en el aeropuerto Jorge Chávez de Lima, entonces va a existir un Vuelo con salida=Ezeiza y llegada=Mariscal Sucre, y una Escala asociado al vuelo con nombreAeropuerto = Jorge Chávez.

Se pide escribir las expresiones SQL que permitan obtener:

1. `<numeroSerieAvion,nombreTipoAvion>` de los aviones que no están programados para ningún vuelo.
2. `<numVuelo,numSerieAvion,cantidadDeAsientos,cantMaxDeAsientos>` de los vuelos que tienen más asientos que los permitidos por el tipo de avión del avión asignado al vuelo.
3. `<nombreTipoDeAvion, cantidadDeAviones, totalAsientos>` de los tipos de avion que tienen registrados más de 40 aviones. El `totalAsientos` es de todos los aviones de cada tipo.
4. Nombre de las ciudades de las que hayan salido más de 10 vuelos **directos**.
5. `<numVuelo>` de los vuelos no directos (de aquellos vuelos que tienen escalas) que se encuentra en alguno de los aeropuertos intermedios (o sea, llegó pero todavía no partió) a las 1440 (14:40hs).
6. `<numeroSerieAvion, nombreAeropuerto>` tomando de cada avión los aeropuertos en los que puede aterrizar (según su tipo de avión) pero que no tiene registrado ningún paso por ese aeropuerto (o sea, el aeropuerto no es ni salida ni llegada ni escala de ningún vuelo registrado para el avión).
7. `<numVuelo, cantTramos>` de los vuelos de no mas de dos escalas desde la ciudad de Buenos Aires hasta la ciudad de Caracas ordenados por hora de salida. Si el vuelo es directo, cantTramos es 1.
8. `<numAeropuerto, ciudad, pais, cantidadDeAterrizajes>` de todos los aeropuertos, contando tanto las escalas como el aeropuerto de llegada de cada vuelo. La cantidadDeAterrizajes debería ser 0 si no existe ningún vuelo que aterrice en el aeropuerto, ya sea por ser su destino final o por ser alguna escala.
9. Los números de todos los vuelos que permiten ir desde el aeropuerto de Ezeiza al Aeropuerto Mariscal Sucre, ordenados por número de vuelo. Tanto Ezeiza como Mcal. Sucre pueden ser cabecera (aerop. de salida y llegada respectivamente) como escalas; si ambos son escalas, el vuelo tiene que pasar antes por Ezeiza que por Mcal. Sucre.
10. `<numVuelo,numAeropuerto,estado>` para cada aeropuerto en el que va a aterrizar (escala o destino final) cada vuelo. El estado es 'Permitido' si el avión puede aterrizar en el aeropuerto, y 'No permitido' en caso contrario.

10. Fiestas - Ejercicio de Trabajo Practico

Partiendo del siguiente esquema:

```
persona <nomPersona, mail, dirCalle, dirNro, dirBarrio, dirCiudad, tipoDoc, nroDoc>
salon <nomSalon, dirCalle, dirNro, dirBarrio, dirCiudad>
servicioSalon <nomSalon, nomServicio>
fiesta <nomOrganiz, fecha, motivo, nomSalon, importeAlquiler>
invitado <nomOrganiz, fecha, nomPersona, formaInvitacion, fechaInvitacion>
asistente <nomOrganiz, fecha, nomPersona, objetoRegalo, valorRegalo>
excusado <nomOrganiz, fecha, nomPersona, excusa>
festejado <nomOrganiz, fecha, nomPersona>
prestamo <nomAcreedor, nomDeudor, fechaPrestamo, montoPrestamo>
pago <nomAcreedor, nomDeudor, fechaPago, montoPago>
show <nomShow, mailContacto, precioPorShow, duracionShow, cantidadParticipantes, tieneAnimales>
showEnFiesta <nomOrganiz, fecha, nomShow, hora, minutos>
noDespuesDe <nomArt_noAntes, nomArt_noDespues>
gastoComida <nomOrganiz, fecha, concepto, importe>
```

Se pide escribir las expresiones SQL que permitan obtener:

1. Nombre de personas que han contraído alguna deuda.
2. Nombre y mail de contacto de shows que nunca han sido contratados en ninguna fiesta.
3. Nombre de los que han organizado al menos una fiesta pero que nunca han sido invitados a ninguna.
4. Nombre y mail de contacto de las personas que no asistieron o que asistieron pero con un regalo de valor menor a 2 pesos en la fiesta que organizó Kent Beck el 24/3/2009, ordenados por nombre.
5. Nombre de las personas que asistieron como invitados a alguna fiesta y que hayan sido organizador al menos una vez.
6. Personas que hayan contraído alguna deuda (por prestamos) y que hayan sido invitadas para alguna fiesta posterior al 4/4/2009.
7. Nombre y mail de todos los invitados que han presenciado (en cualquier fiesta que hayan asistido) tanto el show del payaso “Kill Gates” como el del “Mago MS”.
8. `<nomAcreedor, nomDeudor, totalPagado>` para los préstamos con fecha de préstamo en 2008 para los que se hayan hecho al menos 3 pagos.
9. costo total de la fiesta organizada por Linus Torvalds el 24/12/2008, sumando precio del salón, costo de los shows contratados, y costo de la comida que se compró.
10. fecha y nombre del salón de la fiesta organizada por Linus Torvalds en la que se haya gastado más en vino.
11. `<fecha, nombre organizador, gasto en vino, gasto en cerveza, precio del salón>` para las fiestas en las que se haya comprado tanto vino como cerveza. Ordenado por nombre de organizador y fecha.
12. nombre y mail de contacto de las personas que: o bien fueron a alguna fiesta organizada por Kent Beck, o bien contrajeron al menos dos préstamos de más de 2000 pesos, o bien hayan organizado al menos una fiesta en 2008, sin repetidos y ordenados por nombre.
13. `<queEs, nombre, calle, número>` para todas las personas y salones de Bernal. El “queEs” es ‘Persona’ o ‘Salon’. Ordenado por calle y número.
14. `<nombre, total plata que prestó>` para cada persona registrada, 0 si no prestó nada.

15. `<fecha, nombre del salón, precio del salón, cantidad de shows contratados, total de plata gastada en comida>` para cada fiesta organizada por Kent Beck. Tener en cuenta que tanto la cantidad de shows como el total de plata gastada pueden ser 0, tienen que aparecer en el resultado todas las fiestas organizadas por el amigo Kent.

16. (*) `<nombre, status, comentario>` para las personas que hayan sido invitadas a la fiesta que organizó Linus Torvalds el 24/12/2008. El status puede ser: 'Invitado', 'Se excusó' o 'No se sabe'. El comentario es, respectivamente, el nombre del regalo, la excusa, o null.