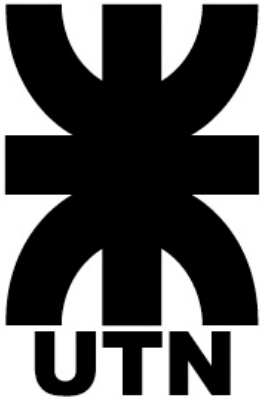
Trabajo práctico Nº 1



Materia

Administración y Diseño de Base de Datos

Docentes

Sato, Fernando
Zapata Icart, Ernesto

Carrera

Tecnicatura Superior en Programación
T.S.P.

Alumnos

Caro, Alejandro
Barraco Mármol, Jerónimo

Introducción

Para generar nuestra base de datos hicimos un archivo *.bat que ejecuta los comandos necesarios y los SQLs dentro de la base de datos.
El contenido del archivo es el siguiente:

```
@echo off

set isql="C:\Archivos de programa\Firebird\Firebird_2_5\bin\isql.exe"
set DATABASE="C:\tp.fdb"
set usuario=SYSDBA
set pwd=masterkey

echo isql es %isql%
echo database es %DATABASE%

del %database%
echo Haciendo macana
echo Creando tablas ...
rem pongo las opciones largas para que se entienda
%isql% -user %usuario% -password %pwd% -q -i "punto1.sql"

echo Creando triggers ...
%isql% -user %usuario% -password %pwd% -q -i "punto1b.sql"

echo Insertando datos ...
%isql% -user %usuario% -password %pwd% -q -i "punto1c.sql"

echo Creando un indice ...
%isql% -user %usuario% -password %pwd% -q -i "punto2.sql"

echo Creando vistas ...
%isql% -user %usuario% -password %pwd% -q -i "punto3.sql"

echo Mas triggers ...
%isql% -user %usuario% -password %pwd% -q -i "punto4.sql"

echo Stored Procedures ...
%isql% -user %usuario% -password %pwd% -q -i "punto5.sql"

echo Mas vistas ...
%isql% -user %usuario% -password %pwd% -q -i "punto6.sql"

echo Ultima vista ...
%isql% -user %usuario% -password %pwd% -q -i "punto7.sql"
```

```
echo Datos extra ...
%isql% -user %usuario% -password %pwd% -q -i "extra.sql"

echo Listo el pollo, se feliz :D
```

Como se verá, el ejecutable usa los archivos llamado puntoX.sql para determinar el contenido de la base de datos.
Nosotros al escribir dichos archivos, lo hicimos ordenado en función de las consignas del trabajo práctico.
A continuación iremos mostrando el contenido de cada archivo sql donde se verá los dml, ddl y dcl que generan, agregan y administran la base de datos.

Punto 1)

- Analizar e identificar los dominios propios de la base creando los tipos respectivos (dominios) y las tablas base.
- Analizar e identificar los dominios propios de la base creando los tipos respectivos (dominios) y las tablas base.
- Se deben entregar las DDL respectivas, según las siguientes restricciones:
- Para los casos de entidades que contengan apellido y nombre, se debe hacer una columna computada con el

apellido/s “, “ nombres. Ej apellido: “Fuentes”, nombres “Maria de los Angeles” debe quedar “Fuentes, Maria
de los Ángeles”.

- Evitar los atributos del tipo varchar y codificarlos en una tabla de referencia, ejemplo ocupación.
- Crear un identificador homogéneo para cada tabla de nombre id de tipo entero y definirlo como clave

primaria.

- Realizar un análisis exhaustivo de PK naturales del sistema, las cuales tienen que ser definidas como

restricción de UNIQUE.

- Realizar un análisis exhaustivo de FK fundamentando el criterio de propagación que elija “CASCADE, etc”.

```
CREATE DATABASE 'c:\tp.fdb' page_size 8192;
-- user 'sysdba' password 'masterkey';

create domain DTRev as varchar(10) check ( value in ('PERIODICA', 'PEDIDA') );
create domain DTTel as varchar(10) check ( value in ('CASA', 'TRABAJO', 'MOVIL',
'OTRO') );
create domain DTBool as smallint check (value in (0,1));
create domain DTCausa as varchar(15) check (value in ('VACACIONES',
'ENFERMEDAD', 'OTROS'));
CREATE DOMAIN DTDOC AS VARCHAR(30) CHECK (VALUE IN('LIBRETA', 'DNI',
'PASAPORTE','OTRO'));

--Tablas
create table GENEROS(
        id integer primary key,
        nombre varchar(50) unique not null
);
create generator id_generos;
set generator id_generos to 0;

CREATE TABLE AUTORES (
    ID      INTEGER primary key,
    NOMBRE  VARCHAR(50) NOT NULL
);
CREATE GENERATOR ID_AUTORES;
```

```
set generator id_autores to 0;

--- obra
create table OBRAS(
    id integer primary key,
    nombre varchar(20) not null unique,
    duracion integer not null,
    id_autor integer not null,
    id_genero integer not null
);
create generator id_obras;
set generator id_obras to 0;

---- LUTHIERES
create table LUTHIERES(
        id integer primary key,
        nombre varchar(50),
        apellido varchar(50),
        dni integer unique,
        nom_completo computed by(nombre || ', ' || apellido)
);
create generator id_luthieres;
set generator id_luthieres to 0;

-- paises
create table paises (
        id integer primary key,
        nombre varchar(50) not null
);
create generator id_paises;
set generator id_paises to 0;

--- musicos
create table musicos(
        id integer primary key,
        nombre varchar(50) not null,
        apellido varchar(50) not null,
        nom_completo computed by(nombre || ', ' || apellido),
        tipo_doc dtdoc not null,
        numero_doc varchar(20) not null,
        domic varchar(50) not null,
        alta date not null,
        baja date,
        id_t_i_contratado integer, -- puede ser null para poder ingresar primero las
habilidades y luego ser contratado, lo que dispara la comprobacion
        id_pais integer not null
);
create generator id_musicos;
set generator id_musicos to 0;
```

```
--- TELEFONO
create table TELEFONOS(
        id integer primary key,
        id_musico integer not null,
        numero varchar(50) unique not null,
        tipo DTTel not null
);
create generator id_telefonos;
set generator id_telefonos to 0;

--- TITULOS
create table titulos (
        id integer primary key,
        nombre varchar(40) not null,
        fecha date not null,
        id_musico integer not null
);
create generator id_titulos;
set generator id_titulos to 0;

----- Licencias
create table licencias (
        id integer primary key,
        inicio date not null,
        fin date,
        causa DTCausa not null,
        id_musico integer not null
);
create generator id_licencias;
set generator id_licencias to 0;

----- tipoInstrumento
Create table TIPOS_INSTRUMENTO(
    id integer primary key,
    tipo varchar(20) not null unique,
    dias_revision integer not null,
    costo_rev decimal(18, 4) not null, --aka money
    id_luthier integer not null
);
create generator id_tipos_instrumento;
set generator id_tipos_instrumento to 0;

--- HABILIDADES
create table habilidades (
        id integer primary key,
        nivel smallint not null,
        id_tipo_inst integer not null,
        id_musico integer not null
```

```
);
create generator id_habilidades;
set generator id_habilidades to 0;

---- Instrumentos necesarios
create table INST_NECESARIOS(
        id integer primary key,
        nivel smallint not null,
        id_obra integer not null,
        id_tipo_inst integer not null,
        cant integer not null
);
create generator id_inst_necesarios;
set generator id_inst_necesarios to 0;

--- Instrumento
create table INSTRUMENTOS(
        id integer primary key,
        n_serie varchar(30) not null unique,
        disponible DTBool not null,
        id_tipo_inst integer not null
);
create generator id_instrumentos;
set generator id_instrumentos to 0;

---Instr
create table inst_favoritos(
        id integer primary key,
        id_musico integer not null,
        id_inst integer not null
);
create generator id_inst_favoritos;
set generator id_inst_favoritos to 0;

---REGISTRO
create table registro (
        id integer primary key,
        id_musico integer not null,
        id_inst integer not null,
        fecha date not null
);
create generator id_registro;
set generator id_registro to 0;

---- revision
create table REVISIONES(
        id integer primary key,
        tipo DTREV not null,
        inicio DATE not null,
```

```
        fin DATE,
        costo DECIMAL(18,4) not null,
        id_instrumento integer not null,
        id_luthier integer not null
);
create generator id_revisiones;
set generator id_revisiones to 0;

---- reparacion
create table REPARACIONES(
        id integer primary key,
        inicio date not null,
        fin date,
        garantia date not null,
        costo DECIMAL(18,4), --el costo puede estar dado despues de la reparacion,
por ende puede ser puesto al final. al terminar de reparar
        detalle varchar(50) not null,
        id_luthier integer not null,
        id_revision integer not null,
        id_instrumento integer not null
);
create generator id_reparaciones;
set generator id_reparaciones to 0;

--Foreign keys de todas las tablas.
--obras
alter table OBRAS add foreign key (id_autor) references AUTORES(id) on delete
cascade on update cascade;
alter table OBRAS add foreign key (id_genero) references GENEROS(id) on delete
cascade on update cascade;

--musicos
alter table musicos add foreign key (id_t_i_contratado) references
tipos_instrumento(id) on delete set null on update cascade;
-- si se llegara a borrar un tipo, no borramos el musico
alter table musicos add foreign key (id_pais) references paises(id) on delete set
null on update cascade;
--si llegaramos a borrar un pais no borramos al musico

--telefonos
alter table telefonos add foreign key (id_musico) references musicos(id) on delete
cascade on update cascade;

--titulos
alter table titulos add foreign key (id_musico) references musicos(id) on delete
cascade on update cascade;

--licencias
alter table licencias add foreign key(id_musico) references musicos(id) on delete
```

```
cascade on update cascade;

-- tipos instrumentos
alter table TIPOS_INSTRUMENTO add foreign key (id_luthier) references
luthieres(id) on update cascade on delete set null;
--el set null es necesario, ya que podriamos borrar un luthier, pero no borraremos
el tipo de instrumento, solo ponemos su luthier default a null

--habilidades
alter table habilidades add foreign key(id_tipo_inst) references
tipos_instrumento(id) on update cascade on delete cascade;
alter table habilidades add foreign key(id_musico) references musicos(id) on
update cascade on delete cascade;

-- instrumentos necesarios
alter table inst_necesarios add foreign key (id_obra) references OBRAS(id) on
delete cascade on update cascade;
alter table inst_necesarios add foreign key (id_tipo_inst) references
TIPOS_INSTRUMENTO(id) on delete cascade on update cascade;

--instrumentos
alter table instrumentos add foreign key (id_tipo_inst) references
TIPOS_INSTRUMENTO(id) on delete cascade on update cascade;

--instrumentos favoritos
alter table inst_favoritos add foreign key(id_musico) references musicos(id) on
update cascade on delete cascade;
alter table inst_favoritos add foreign key(id_inst) references instrumentos(id) on
update cascade on delete cascade;

--registro
alter table registro add foreign key(id_musico) references musicos(id) on update
cascade on delete cascade;

--revisiones
alter table revisiones add foreign key (id_instrumento) references
INSTRUMENTOS(id) on delete cascade on update cascade;
alter table revisiones add foreign key (id_luthier) references LUTHIERES(id) on
delete cascade on update cascade;

--reparaciones
alter table reparaciones add foreign key (id_revision) references REVISIONES(id)
on delete cascade on update cascade;
alter table reparaciones add foreign key (id_luthier) references LUTHIERES(id) on
delete cascade on update cascade;
alter table reparaciones add foreign key (id_instrumento) references
INSTRUMENTOS(id) on delete cascade on update cascade;

Aquí termina el archivo punto1.sql
```

```
CONNECT "C:\tp.fdb" user 'SYSDBA' password 'masterkey';

-- triggers
SET term ^ ;

create trigger TR_ID_GENEROS for GENEROS ACTIVE before insert
as
begin
        if (new.id IS NULL) then new.id = gen_id(id_generos, 1);
end ^


create trigger TR_ID_REPARACIONES for REPARACIONES active before insert
as
begin
    if (new.id is null) then new.id = gen_id(id_reparaciones, 1);
end^

create trigger TR_ID_OBRAS for OBRAS active before insert
as
begin
        if (new.id is null) then new.id = gen_id(id_obras, 1);
end ^

create trigger TR_ID_LUTHIERES for LUTHIERES ACTIVE before insert
as
begin
        if (new.id IS NULL) then new.id = gen_id(id_luthieres, 1);
end^


create trigger TR_id_paises for paises ACTIVE before insert
as
begin
        if (new.id IS NULL) then new.id = gen_id(id_paises, 1);
end^

create trigger tr_id_musicos for musicos active before insert position 0
 as
 begin
    if(new.id is null) then new.id = gen_id(id_musicos, 1);
 end^

create trigger TR_ID_TELEFONOS for TELEFONOS ACTIVE before insert
as
begin
        if (new.id is null) then new.id = gen_id(id_telefonos, 1);
end^
```

```
create trigger tr_id_titulos for titulos active before insert position 0
as
begin
  if(new.id is null) then new.id = gen_id(id_titulos, 1);
end^

create trigger tr_id_licencias for licencias active before insert position 0
as
begin
   if(new.id is null)then new.id = gen_id(id_licencias, 1);
end^

create trigger TR_ID_TIPOS_INSTRUMENTO for TIPOS_INSTRUMENTO active
before insert
as
begin
       if (new.id is null) then new.id = gen_id(id_tipos_instrumento, 1);
end^

create trigger tr_id_habilidades for habilidades active before insert position 0
as
begin
   if(new.id is null) then new.id=gen_id(id_habilidades,1);
end^

create trigger TR_ID_INST_NECESARIOS for INST_NECESARIOS active before
insert
as
begin
   if (new.id is null) then new.id = gen_id(id_inst_necesarios, 1);
end^

create trigger TR_ID_INSTRUMENTOS for INSTRUMENTOS active before insert
as
begin
   if (new.id is null) then new.id = gen_id(id_instrumentos, 1);
end^

create trigger tr_id_inst_fav for inst_favoritos active before insert position 0
as
begin
   if(new.id is null) then new.id = gen_id(id_inst_favoritos,1);
end^

create trigger tr_id_registro for registro active before insert position 0
as
begin
   if(new.id is null) then new.id = gen_id(id_registro, 1);
end^
```

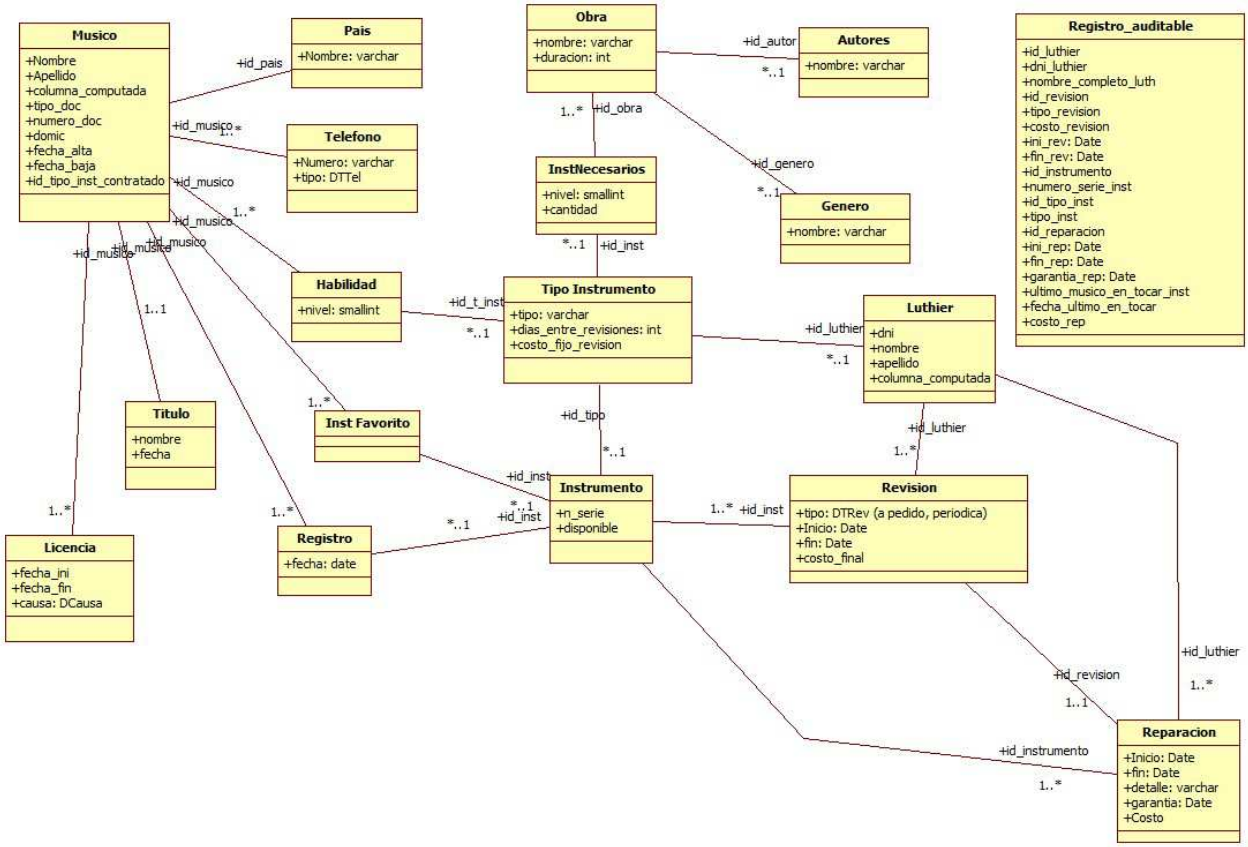
```
create trigger tr_id_AUTORES for AUTORES ACTIVE before insert
as
begin
        if (new.id IS NULL) then new.id = gen_id(ID_AUTORES, 1);
end^

create trigger TR_ID_REVISIONES for REVISIONES active before insert
as
begin
   if (new.id is null) then new.id = gen_id(id_revisiones, 1);
end^

set term ; ^
```

Aquí termina el archivo punto1b.sql

```
CONNECT "C:\tp.fdb" user 'SYSDBA' password 'masterkey';

--- inserts
insert into paises(nombre) values('ARGENTINA');
insert into paises(nombre) values('CHILE');

insert into generos (nombre) values ('clasico');
insert into autores (nombre) values('mozart');
insert into obras(nombre, duracion, id_autor, id_genero)
        values ('Requiem', 75, 1 , 1);
insert into luthieres(nombre,apellido, dni) values ('jorge', 'ruiz', 312458);

insert into Tipos_instrumento (id, tipo, dias_revision, costo_rev, id_luthier) values
(1 , 'flauta', 120, 30.5, 1);
insert into Tipos_instrumento (id, tipo, dias_revision, costo_rev, id_luthier) values
(2 , 'piano', 300, 70, 1);

insert into INST_NECESARIOS(nivel, id_obra, id_tipo_inst, cant)
        values (3, 1, 1, 1);

insert into INST_NECESARIOS(nivel, id_obra, id_tipo_inst, cant)
        values (2, 1, 2, 1);

insert into INSTRUMENTOS(n_serie, disponible, id_tipo_inst)
        values ('ASDLF123345', 1, 1);

insert into INSTRUMENTOS(n_serie, disponible, id_tipo_inst)
        values ('9182323', 1, 2);

insert into INSTRUMENTOS(n_serie, disponible, id_tipo_inst)
        values ('2833 14 K', 1, 2);

insert into INSTRUMENTOS(n_serie, disponible, id_tipo_inst)
        values ('629824499', 1, 1);

insert into INSTRUMENTOS(n_serie, disponible, id_tipo_inst)
        values ('YFL 381 HII', 1, 2);

insert into musicos(nombre, apellido, tipo_doc, numero_doc, domic, alta, id_pais)
        values('Juan', 'Perez', 'DNI', '24114324', 'Irigoyen 399', '12/11/2011', 1);
insert into habilidades(id_tipo_inst, id_musico, nivel) values (1, 1, 5);
update musicos set id_t_i_contratado = 1 where id = 1;

insert into inst_favoritos(id_musico, id_inst) values(1, 1);

insert into telefonos (id_musico, numero, tipo) values (1, '12345', 'TRABAJO');
insert into titulos(nombre, fecha, id_musico) values ('Flautista', '10/15/2004', 1);
```

```
insert into licencias(id_musico, inicio, fin, causa) values (1,
'01/25/2011','02/08/2011', 'VACACIONES');

insert into registro(id_musico, id_inst, fecha) values(1,1,'05/26/2011');

insert into REVISIONES(tipo, inicio, fin, costo, id_instrumento, id_luthier)
        values ( 'PEDIDA', CURRENT_DATE, CURRENT_DATE +1 , 12.20, 1, 1);

insert into REPARACIONES(inicio, garantia, detalle, id_instrumento, id_luthier,
id_revision)
        values ( CURRENT_DATE, CURRENT_DATE+60, 'se rompio un boton', 1, 1,
1);

insert into musicos (nombre, apellido, tipo_doc, numero_doc, domic, alta, id_pais)
        values('Pedro', 'Quiroga', 'DNI', '23446557', 'loma altamira', current_date,
2);

insert into titulos(nombre, fecha, id_musico) values ('Pianista', '10/15/2004', 2);

insert into habilidades (id_tipo_inst, id_musico, nivel) values(2, 2, 6);
update musicos set id_t_i_contratado = 2 where id = 2;

Aquí termina el archivo punto1c.sql
```

- Generar el Diagrama de Entidad-Relación completo.



Punto 2)

a) Incorporar índices que permitan agilizar búsquedas por apellidos y nombre teniendo en cuenta que si el músico
esta vigente, considerar para esto que la orquesta maneja mas de 8000 músicos de los cuales un 6 % se
encuentra activo.
b) Responder: Que sucede con los atributos definidos como claves alternativas con UNIQUE respecto de su
indización, es necesario crear índices?

```
CONNECT "C:\tp.fdb" user 'SYSDBA' password 'masterkey';

-- Punto 2 a)
--- indices
create index ix_musicos on musicos (baja, nombre, apellido);

--b)
--- No es necesario crear indices para las columnas con constraint PK o UNIQUE ya la base de datos crea indices implicitos en estos casos

Aquí termina el archivo punto2.sql
```

Punto 3)

```
CONNECT "C:\tp.fdb" user 'SYSDBA' password 'masterkey';

--Requerimiento del ejercicio, llevar cuenta de los titulos de los musicos
argentinos
create or alter view v_Mus_arg
(nombre, tipod, documento, titulo, fecha)
as select
                m.nom_completo, m.tipo_doc, m.numero_doc,
                t.nombre, t.fecha
                from musicos m
                join paises p on m.id_pais = p.id and p.nombre = 'ARGENTINA'
                join titulos t on t.id_musico = m.id;


-- a) Crear una vista que integre los músicos vigentes, los instrumentos que son
contratados y en caso de ser
-- caprichosos el instrumento concreto que toca.
create or ALTER view v_musicos_inst
    (nombre, apellido, tipo, n_serie)
    as
    select m.nombre, m.apellido, t.tipo, ins.n_serie from musicos m
    join tipos_instrumento t on t.id = m.id_t_i_contratado
    LEFT join inst_favoritos i on i.id_musico = m.id
    LEFT join instrumentos ins on ins.id = m.id_t_i_contratado
    where m.baja is null;

-- b) Crear una vista que muestre los “musicos estrella”. En la orquesta llaman
asi a los musicos que saben interpretar todos los instrumentos existentes en la
orquesta.

create view v_estrellas ( id_musico, nombre, apellido, cant_instrumentos )  as
   select m.id, m.nombre, m.apellido, count (h.id) as cuenta from musicos m
      join habilidades h on (m.id = h.id_musico)
   group by m.id, m.nombre, m.apellido
   having count(h.id)= (select count(1) from tipos_instrumento);

-- c) Generar una vista con la informacion de instrumentos revisados, que luthier
lo tiene, y a que tipo pertenece.
CREATE OR ALTER VIEW V_LUTH_INST_REV(
   NOMBRE,
   DNI,
   TIPO_Revision,
   FECHA_inicio,
   COSTO,
   N_SERIE,
```

```
    TIPO_inst,
    COSTO_REV)
AS
    select l.nom_completo, l.dni, r.tipo, r.inicio, r.costo, ins.n_serie,
    t.tipo, t.costo_rev
    from luthieres l
        join revisiones r on r.id_luthier = l.id
        join instrumentos ins on ins.id = r.id_instrumento
        join tipos_instrumento t on t.id = ins.id_tipo_inst
        where r.fin is null;


--d) Generar una vista que contenga, una fila por cada tabla del sistema y 2
columnas, una con el nombre de tabla y
--otra con la cantidad de filas. Nota: Vista estática, cuando se agrega una nueva
tabla se debe agregar
--explícitamente a la vista.

CREATE VIEW V_TABLAS
(NOMBRE, CUENTA) AS
select 'AUTORES' as nombre, count(1) as cuenta from autores
union
select 'GENEROS' as nombre, count(1) as cuenta from generos
union
select 'HABILIDADES' as nombre, count(1) as cuenta from HABILIDADES
union
select 'INSTRUMENTOS' as nombre, count(1) as cuenta from INSTRUMENTOS
union
select 'INST_FAVORITOS' as nombre, count(1) as cuenta from INST_FAVORITOS
union
select 'INST_NECESARIOS' as nombre, count(1) as cuenta from
INST_NECESARIOS
union
select 'LICENCIAS' as nombre, count(1) as cuenta from LICENCIAS
union
select 'LUTHIERES' as nombre, count(1) as cuenta from LUTHIERES
union
select 'MUSICOS' as nombre, count(1) as cuenta from MUSICOS
union
select 'OBRAS' as nombre, count(1) as cuenta from OBRAS
union
select 'PAISES' as nombre, count(1) as cuenta from PAISES
union
select 'REGISTRO' as nombre, count(1) as cuenta from REGISTRO
union
select 'REPARACIONES' as nombre, count(1) as cuenta from REPARACIONES
union
```

```
select 'REVISIONES' as nombre, count(1) as cuenta from REVISIONES
union
select 'TELEFONOS' as nombre, count(1) as cuenta from TELEFONOS
union
select 'TIPOS_INSTRUMENTO' as nombre, count(1) as cuenta from
TIPOS_INSTRUMENTO
union
select 'TITULOS' as nombre, count(1) as cuenta from TITULOS;
```

Aquí termina el archivo punto3.sql

Punto 4)

```
CONNECT "C:\tp.fdb" user 'SYSDBA' password 'masterkey';

SET term ^ ;

--requerimiento del practico, llevar cuenta  de los instrumentos disponibles

create or alter trigger tr_revision for revisiones
active after insert or update position 0
as
begin
   update instrumentos
      set disponible = iif( new.fin is null, 0, 1)
      where new.id_instrumento = instrumentos.id;
end^

create or alter trigger tr_reparacion for reparaciones
active after insert or update position 0
as
begin
   update instrumentos
      set disponible = iif( new.fin is null, 0, 1)
      where new.id_instrumento = instrumentos.id;
end^

-- a) los triggers para los indices autoincrementales se encuentran en ddl.sql

-- b) Implementar un/os trigger/s que administre/n la regla de negocio “Es
necesario llevar registro de los
--instrumentos para los cuales fue contratado cada músico en la orquesta. Nunca
puede ser contratado para un
--instrumento que no sabe interpretar”.
--este verifica que no toque algo que no sabe
create exception EX_REGISTRO 'El musico no esta capacitado para este
instrumento'^
create trigger TR_Registro_Negocio for registro
active
before insert or update  position 0
as begin
   if (not exists(
      Select
         m.id
      from
         musicos m
         join habilidades h on m.id = h.id_musico
         join tipos_instrumento ti on ti.id = h.id_tipo_inst
         join instrumentos i on i.id_tipo_inst = ti.id
      where
```

```
          m.id = new.id_musico
          and i.id = new.id_inst
       )
   ) then begin
      exception EX_REGISTRO;
   end
end^
--este verifica que no lo contraten para algo que no sabe
create trigger TR_Registro_Negocio_contrato for musicos
active
before update  position 0
as
begin
        if(not exists( select * from musicos m
                        join habilidades h on (m.id = h.id_musico)
                        where h.id_tipo_inst = new.id_t_i_contratado
        )) then begin
                exception EX_REGISTRO;
        end
end^

-- c) Implementar un/os trigger/s que administren la historia de los cambios de
niveles de un músico en el tiempo.
-- se crea la tabla junto al trigger, ya que su existencia queda vinculada al trigger
create table REG_HABILIDADES(
   id_tipo_inst integer not null,
   id_musico integer not null,
   nivel integer not null,
   fecha timestamp not null,
   foreign key (id_tipo_inst) references instrumentos(id),
   foreign key (id_musico) references musicos(id)
) ^

create trigger TR_REG_HABILIDADES for habilidades
active before insert or update
as begin
        insert into reg_habilidades(id_tipo_inst, id_musico, nivel , fecha)
                values(new.id_tipo_inst, new.id_musico, new.nivel,
current_timestamp);
end^

--d) Crear un esquema básico genérico que permita registrar la auditoria de una
tabla del sistema en cuanto a alta,
--ultima modificación. De estos eventos se debe conocer el instante de tiempo en
que se produjo el evento y el
--usuario de base de datos. Implemente un trigger “modelo” para una tabla.
Ampliado en el punto siguiente.
create table auditoria (tiempo timestamp, usuario varchar(31), accion
varchar(10), tabla varchar(20))^
```

```
create trigger tr_audit_musico for musicos
active after insert or update
as
declare variable acciones varchar(10);
begin
    if(inserting)then acciones = 'insert';
    else acciones = 'update';
    insert into auditoria (tiempo, usuario, accion, tabla) values
(current_timestamp, current_user, :acciones, 'MUSICOS');
end^

set term ; ^

insert into habilidades (id_tipo_inst, id_musico, nivel) values (1, 1, 3);
insert into habilidades (id_tipo_inst, id_musico, nivel) values (2, 2, 3);

insert into registro(id_musico, id_inst, fecha) values (1, 1, current_date);
insert into registro(id_musico, id_inst, fecha) values (2, 2, current_date);

insert into musicos (nombre, apellido, tipo_doc, numero_doc, domic, alta, id_pais)
    values('Oscar', 'Rabiola', 'DNI', '12444666', 'Strada L Oro', current_date, 1);
update musicos set id_t_i_contratado=1 where id = 1;
update musicos set id_t_i_contratado=2 where id = 2;
```

Aquí termina el archivo punto4.sql

Punto 5)

```
CONNECT "C:\tp.fdb" user 'SYSDBA' password 'masterkey';

SET term ^ ;

--a) Cree un procedimiento almacenado que pida como único argumento un
nombre de tabla y --genere la ddl que realiza el código para generar el/los
trigger/s de autoincremento según el modelo --efectuado en el punto anterior
inciso a.

create or alter procedure P_AUTOINCDDL(tabla varchar(16))
        returns (ddl varchar (255))
as
begin
        ddl = 'create generator id_' || tabla || '^
                set generator id_' || tabla || ' to 0^
                create trigger TR_ID_' || tabla || ' for ' || tabla || '
                active before insert as begin
                if (new.id is null) then new.id = gen_id(id_' || tabla || ', 1);
                end^';
        suspend;
end^

--b) Cree un procedimiento almacenado que pida como único argumento un
nombre de tabla y --genere la ddl que realice el código para generar el/los
trigger/s de auditoria según el modelo --efectuado en el punto anterior inciso d.

create or alter procedure P_AUDIT(tabla varchar(16))
    returns (ddl varchar (500))
as
begin
    ddl = 'create trigger tr_audit_' || tabla || ' for ' || tabla || '
    after insert or update as
declare acciones varchar(10);
begin
    if(inserting)then acciones=''insert'';
    else acciones=''update'';
    insert into auditoria(tiempo,usuario,accion,tabla) values (current_timestamp,
current_user, :acciones, ''' || tabla || ''');
end';
    suspend;
end^

--- Devuelve todos los posibles musicos con instrumentos segun la necesidad de
una obra, siempre que el instrumento este disponible
-- y no sea esa una de las ultimas 2 combinaciones
```

```
--c) Realice un sp que se llame rotación que reciba como parametro la clave de
una partitura y --proponga un nueva combinación de músicos e instrumentos
distinta a la últimas 2 interpretaciones --registradas.
CREATE GLOBAL TEMPORARY TABLE temp_elegidos
  (
  idm integer,
  idi integer)
  oN COMMIT DELETE ROWS ^

create or alter procedure rotacion(pid_obra integer) returns (vti integer, vniv
integer, vmid integer, viid integer)
as
declare vaux integer;
declare vcant_nec integer;
declare vcant_ins intege R;
begin
   for select id_tipo_inst, nivel, cant --itera nivel/tipo
      from inst_necesarios where id_obra = :pid_obra
      into :vti, :vniv, :vcant_nec
   do begin
            --reiniciamos la basura necesaria para los ifs
            vcant_ins = 0;
      for select m.id, rand() pos --iteramos los musicos para cada nivel/tipo, el pos
es para randomizar
         from
            musicos m
            join habilidades h on (m.id= h.id_musico) -- podriamos busar solo los
contratados, pero eso reduce mucho las posibilidades
         where
                        m.baja is null and -- musicos que quieran trabajar
            h.nivel >= :vniv and
            h.id_tipo_inst = :vti and
            m.id not in (select idm from temp_elegidos) -- no lo hayamos elegido
anteriormente
         order by pos
         into :vmid, :vaux
      do begin
               viid = null;
         select first 1 i.id, rand() pos --nos fijamos si tiene un inst favorito para
este tipo
         from
            instrumentos i
            join inst_favoritos inf on (
                  inf.id_musico = :vmid and
                  inf.id_inst = i.id
               )
         where
```

```
            i.id_tipo_inst = :vti and
            i.disponible = 1 and
            i.id not in (select idi from temp_elegidos) -- no lo este usando otro
                    order by pos
        into :viid, :vaux;

        if (viid is null) then begin --si no tiene un favorito para este tipo...
                        --necesito hacerlo en dos partes porque sino el select se
hace muy largo y no tengo tipo_inst en la tabla instfavoritos
            select first 1
                i.id, rand() pos
            from
                instrumentos i
            where
                i.id_tipo_inst = :vti and
                i.disponible = 1 and
                i.id not in (select idi from temp_elegidos ) and
                i.id not in ( --diferente a las ultimas 2 interpretaciones , notar que
no lo hacemos con inst favorito porque no tiene sentido ya que todas las
interpretaciones son iguales
                    select first 2 id_inst
                    from registro
                    where
                        id_musico = :vmid
                    order by fecha DESC
                )
                        order by pos
            into :viid, :vaux;
        end --fin sin favorito
        if (viid is not null) then begin --nunca fue tan complicado ponerla (sujeto
tacito "informacion") (aunque se que son solo datos no info)
            insert into temp_elegidos (idm, idi) values (:vmid, :viid); --yay.....
            --deberia iterar la tabla elegidos al final... pero cometiendo esta
atrocidad, puedo beneficiarme en cuanto al tiempo de retorno del sp y que quizas
no se usen todos los datos retornados por este sp...
                        vcant_ins = vcant_ins+1;
            suspend;
                        if (vcant_ins >= vcant_nec) then break;
        end
    end    --fin musico/inst/nivel
            --if (vcant_ins < vcant_nec ) then begin -- no lanzo excepcion, para
eos esta la otra funcion
            while (vcant_ins < vcant_nec) do begin -- solo para senialar que faltan
musicos
                    vmid = null;
                    viid = null;
                    vcant_ins = vcant_ins +1;
```

```
                suspend;
            end
        --aca podria hacer un select count (1) from @elegidos where tipo  > cant y
ver si tenemos tantos musicos como necesitamos
    end -- fin nivel/tipo
    delete from temp_elegidos; -- por las dudas si hacen otro select en esta
transaccion
end^

--d) Producir un sp con el nombre  init_tabla  que ejecute las DML necesarias
para borrar las tablas en el orden
--necesario para que no se produzcan excepciones de foreign key. Nota no debe
borrar las tablas de tipo, ejemplo
--tipo o grupos de instrumentos (en principio, vientos, cuerdas o percusión).
create or alter procedure init_tabla (a integer) as begin
        delete from luthieres;
        delete from musicos;
        delete from obras;
        delete from instrumentos;
        --el resto de las tablas no se borran por considerarse genericas para todos.
        --las demas tablas se borran por el constraint del foreign key.
        --en tipos_intrumento tiene un "on delete" especial para evitar que se borre
el dato.
end^


--e) Producir un sp que genere 5 insert como mínimo a fin de generar un ejemplar
de la base para pruebas, que tenga
--consistencia.
create or alter procedure generar_info_default as begin
            if(not exists(select nombre from paises where nombre = 'p')) then
                insert into paises (nombre) values ('p');
                --La consigna asume que la DB esta vacia, pero el if esta puesto
como ejemplo de si no lo estuviera.
        insert into musicos (nombre, apellido, tipo_doc, numero_doc, domic, alta,
id_pais)
                values ('p', 'p', 'DNI', 'p', 'p', current_date, (select id from paises
where nombre = 'ARGENTINA'));
        insert into habilidades (id_tipo_inst, id_musico, nivel)
                values (1, (select first 1 id from musicos where nombre = 'p'), 1);

            update musicos set id_t_i_contratado = 1 where id = (select first 1 id
from musicos where nombre = 'p');

        insert into telefonos (id_musico, numero, tipo)
                values ((select id from musicos where nombre = 'p'), 'p', 'CASA');
        insert into instrumentos (n_serie, disponible, id_tipo_inst)
```

```
                values('p', 1, 1);
        insert into registro (id_musico, id_inst, fecha)
                values ((select id from musicos where nombre = 'p'), (select id
from instrumentos where n_serie = 'p'), current_date);
            insert into luthieres (nombre, dni) values('p', 0);
                update tipos_instrumento set id_luthier = (select id from
luthieres where nombre = 'p') where id = 0;
end^
SET term ; ^
```

Aquí termina el archivo punto5.sql

Punto 6)

```
CONNECT "C:\tp.fdb" user 'SYSDBA' password 'masterkey';

--a)Las partituras que no podrian ser interpretadas ya sea porque no se cuenta
con  los musicos con la experiencia
--requerida o porque no hay suficientes instrumentos para interpretarla. En este
caso seria adecuado que ademas
--se indique qur es lo que falta para poder interpretar la mencionada partitura.
Para poder interpretar una
--partitura el nivel en que un musico interpreta un instrumento debe coincidir
exactamente con el requerido para ese
--instrumento.
create or alter view v_obras_imposibles (id, nombre, tipo, nivel, necesarios,
disponibles, problema) as
        select o.id, o.nombre, t.tipo, n.nivel, n.cant, count(i.n_serie) as disponibles,
                'Se necesitan mas instrumentos' as msg
                from obras o
                join inst_necesarios n on o.id = n.id_obra
                join tipos_instrumento t on n.id_tipo_inst = t.id
                left join instrumentos i on i.id_tipo_inst = t.id
                group by o.id, o.nombre, t.tipo, n.cant, n.nivel, msg
                having count(i.id) < n.cant
        union
        select o.id, o.nombre, t.tipo, n.nivel, n.cant, count(h.nivel) as disponibles,
                'Se necesitan mas musicos' as msg
                from obras o
                join inst_necesarios n on o.id = n.id_obra
                join tipos_instrumento t on n.id_tipo_inst = t.id
                left join habilidades h on h.id_tipo_inst = t.id and h.nivel >= n.nivel
                group by o.id, o.nombre, t.tipo, n.cant, n.nivel, msg
                having count(h.nivel) < n.cant ;

--b)La cantidad de dias de licencia de cada tipo que solicitaron los músicos
durante el último año.
--El siguiente dml es generico.
create or alter view v_tot_licencias (dias, causa) as
        select sum(l.fin - l.inicio) as dias, l.causa
                from licencias l
                where  extract (year from l.fin) = extract (year from current_date)
                group by causa;

--- Este es para el caso de querer ver las licencias de cada musico por separado
(no esta explicito en la consigna)
create or alter view v_licencias (nombre, dias, causa) as
        select m.nom_completo, sum(l.fin - l.inicio) as dias, l.causa
                from licencias l join musicos m on l.id_musico = m.id
```

```
where  extract (year from l.fin) = extract (year from current_date)
group by  m.nom_completo, l.causa;
```

--c)Un registro auditable de las operaciones de Alta y Modificaciones de las reparaciones.

```
CREATE TABLE REG_AUDITABLE (
    ID_LUTHIER         INTEGER NOT NULL,
    NOM_COMPLETO_LUTH  VARCHAR(105) NOT NULL,
    ID_REVISION        INTEGER NOT NULL,
    TIPO_REV           DTREV NOT NULL, /* DTREV = VARCHAR(10) check (value
in ('PERIODICA', 'PEDIDA')) */
    INICIO             DATE NOT NULL,
    FIN_REV            DATE,
    COSTO_REV          DECIMAL(18,4),
    ID_INSTRUMENTO     INTEGER NOT NULL,
    N_SERIE            VARCHAR(20) NOT NULL,
    ID_TIPO_INST       INTEGER NOT NULL,
    TIPO_INST          VARCHAR(20) NOT NULL,
    ID_REPARACION      INTEGER,
    FECHA              DATE,
    GARANTIA           DATE,
    FIN_REP            DATE,
    COSTO_REP          DECIMAL(18,4),
    ULTIMO_QUE_TOCO    VARCHAR(105),
    DNI_LUTH           INTEGER,
    FECHA_ULTIMO       DATE,
        FECHA_MODIFICACION TIMESTAMP
);

SET TERM ^ ;

CREATE OR ALTER TRIGGER TR_AUD_REP FOR REPARACIONES
        ACTIVE AFTER INSERT OR UPDATE POSITION 0
        AS
        DECLARE VARIABLE v_id_luth INTEGER;
        DECLARE VARIABLE v_dni_luth INTEGER;
        DECLARE VARIABLE v_nom_compl_luth VARCHAR(105);
        DECLARE VARIABLE v_id_rev INTEGER;
        DECLARE VARIABLE v_tip VARCHAR(10);
        DECLARE VARIABLE v_costo_reparacion DECIMAL(18,4);
        DECLARE VARIABLE v_costo_rev DECIMAL(18,4);
        DECLARE VARIABLE v_id_inst INTEGER;
        DECLARE VARIABLE v_ini_rev DATE;
        DECLARE VARIABLE v_fin_rev DATE;
        DECLARE VARIABLE v_n_serie_inst VARCHAR(20);
```

```
DECLARE VARIABLE v_id_tipo_inst INTEGER;
DECLARE VARIABLE v_tipo_inst VARCHAR(20);
DECLARE VARIABLE v_id_rep INTEGER;
DECLARE VARIABLE v_fecha_rep DATE;
DECLARE VARIABLE v_garan_rep DATE;
DECLARE VARIABLE v_fin_rep DATE;
DECLARE VARIABLE v_ultimo_musico VARCHAR(105);
declare variable v_fecha_ult date;
BEGIN
	--luthier
	v_id_luth = NEW.id_luthier;
	select first 1 dni, nom_completo from luthieres
		where id = :v_id_luth
	into :v_dni_luth, :v_nom_compl_luth;


	--revision
	v_id_rev = NEW.id_revision;
	select first 1 tipo, inicio, costo, fin from revisiones
		where id = :v_id_rev
		into :v_tip, :v_ini_rev, :v_costo_rev, :v_fin_rev;

	--instrumento
	SELECT first 1 id_instrumento FROM revisiones
		WHERE id = :v_id_rev INTO :v_id_inst;

	SELECT first 1 n_serie FROM instrumentos
		WHERE id = :v_id_inst
	into :v_n_serie_inst;

	SELECT first 1 t.tipo, t.id FROM tipos_instrumento t
		JOIN instrumentos i ON (i.id_tipo_inst = t.id )
		where i.id = :v_id_inst
	into :v_tipo_inst, :v_id_tipo_inst;

	--reparacion
	v_id_rep = NEW.id;
	v_fecha_rep = NEW.inicio;
	v_garan_rep = NEW.garantia;
	v_fin_rep = NEW.fin;
	v_costo_reparacion = NEW.costo;


	--musico que toco ultimo el instrumento
	SELECT first 1  m.nom_completo FROM musicos m
		JOIN registro re ON re.id_musico = m.id
		AND re.id_inst = :v_id_inst ORDER BY fecha DESC
```

```
        into :v_ultimo_musico;

SELECT first 1  re.fecha FROM musicos m
        JOIN registro re ON re.id_musico = m.id
        AND re.id_inst = :v_id_inst ORDER BY fecha DESC
        into :v_fecha_ult;

insert into reg_auditable (
                id_luthier,
                dni_luth,
                nom_completo_luth,
                id_revision,
                tipo_rev,
                inicio,
                fin_rev,
                costo_rev,
                id_instrumento,
                n_serie,
                id_tipo_inst,
                tipo_inst,
                id_reparacion,
                fecha,
                garantia,
                fin_rep,
                costo_rep,
                ultimo_que_toco,
                fecha_ultimo
                )
  values(
                :v_id_luth,
                :v_dni_luth,
                :v_nom_compl_luth,
                :v_id_rev,
                :v_tip,
                :v_ini_rev,
                :v_fin_rev,
                :v_costo_rev,
                :v_id_inst,
                :v_n_serie_inst,
                :v_id_tipo_inst,
                :v_tipo_inst,
                :v_id_rep,
                :v_fecha_rep,
                :v_garan_rep,
                :v_fin_rep,
                :v_costo_reparacion,
                :v_ultimo_musico,
```

```
                        :v_fecha_ult
            );
    END  ^
SET TERM ; ^
```

Aquí termina el archivo punto6.sql

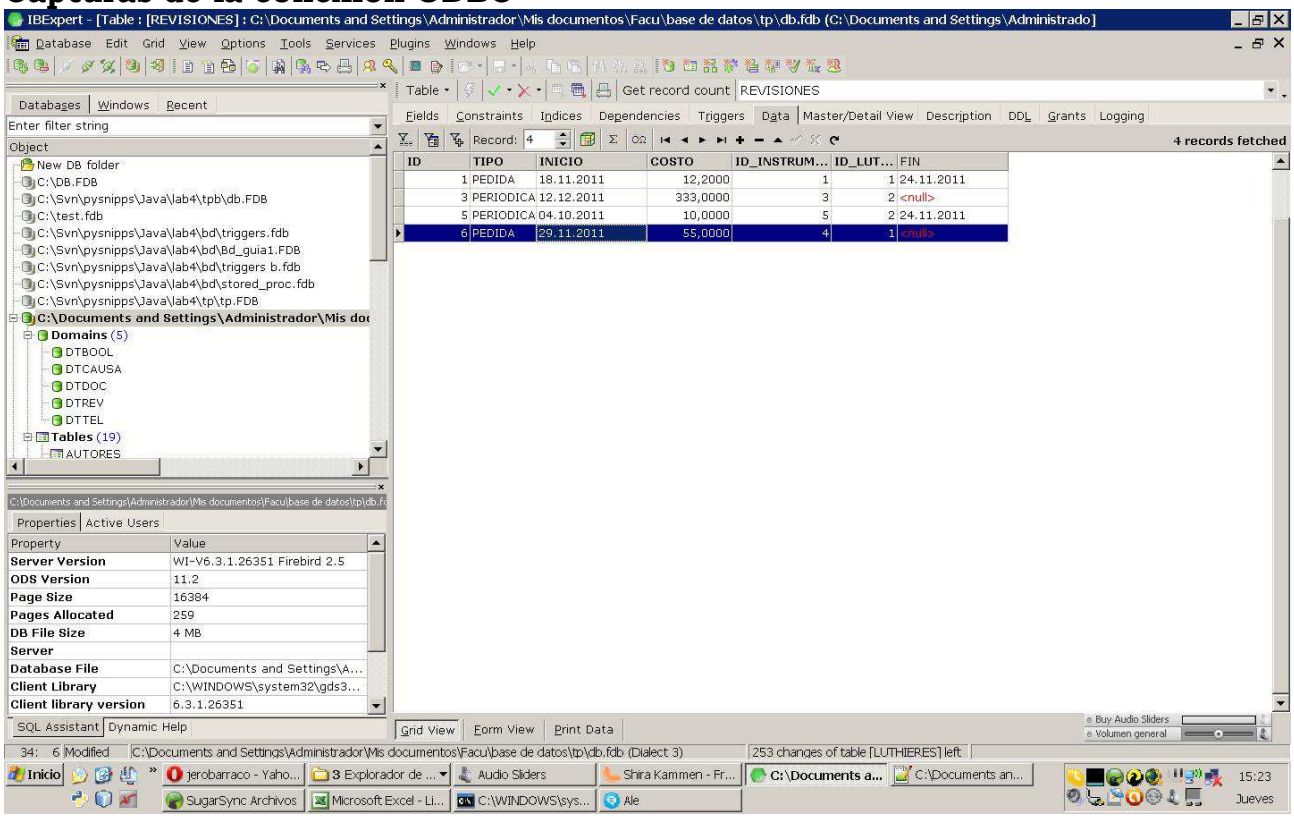
Punto 7)

a) Realice una conexión ODBC, utilice esta conexión desde una planilla de cálculo y genere una tabla dinámica con la
siguiente información.
Dimensiones
• Año
• Mes
• Grupos de Instrumentos (vientos, cuerdas o percusión).
• Tipo de Revisión (periodica, a pedido)
Hecho: cantidad de revisiones.
Considere la posibilidad de utilizar una vista para la información desagregada.

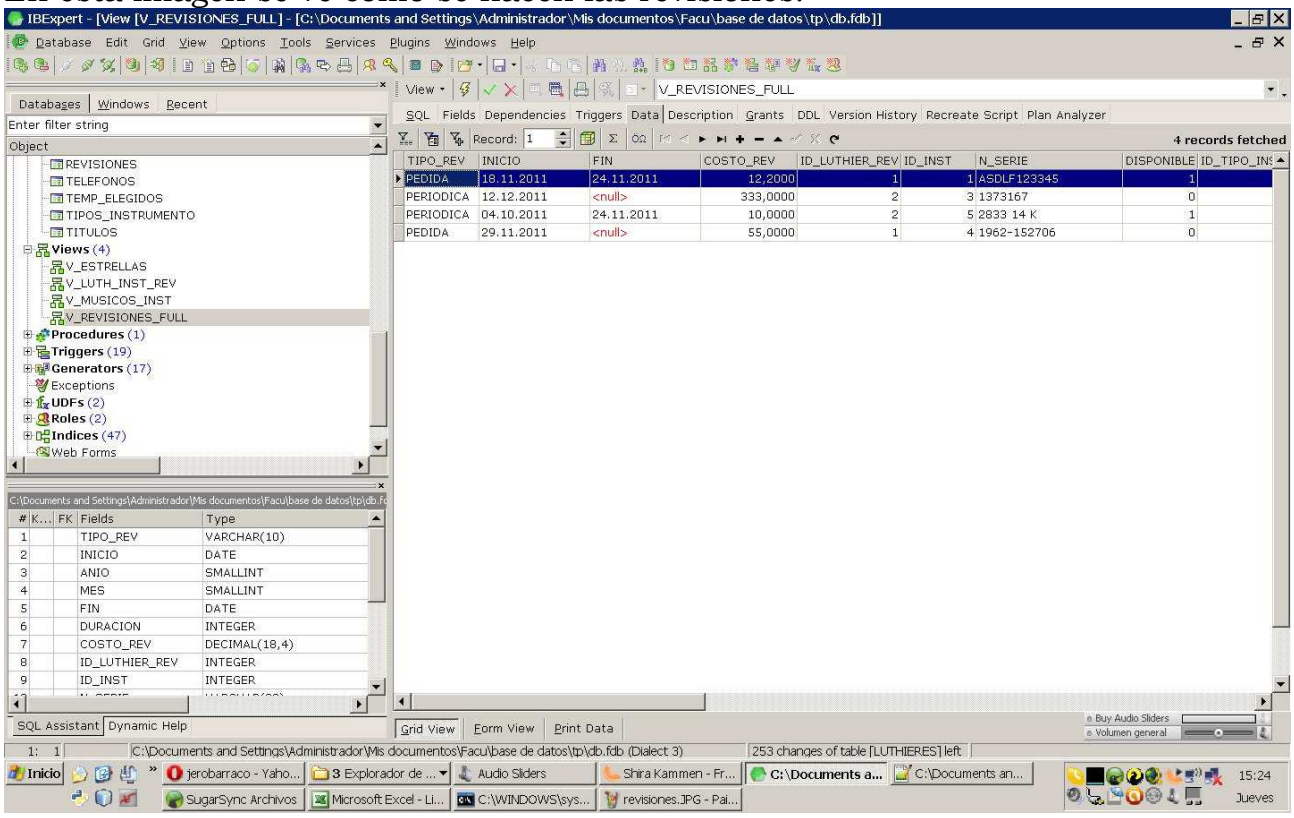
```
CONNECT "C:\tp.fdb" user 'SYSDBA' password 'masterkey';

CREATE OR ALTER VIEW V_REVISIONES_FULL(
    TIPO_REV,
    INICIO,
        anio,
        mes,
    FIN,
        DURACION,
    COSTO_REV,
    ID_LUTHIER_REV,
    ID_INST,
    N_SERIE,
    DISPONIBLE,
    ID_TIPO_INST,
    TIPO_INST,
    DIAS_REV_PERIODICA,
    COSTO_REV_PERIODICA,
    ID_DEFAULT_LUTHIER,
    NOM_LUTHIER,
    DNI_LUTHIER)
AS
select
    r.tipo, r.inicio, extract(year from r.inicio), extract(month from r.inicio), r.fin,
        r.fin-r.inicio, -- a proposito, dara null si fin es null por ende se pueden
contar cuantas se terminaron y cuantas no o sumar las que si terminaron
        r.costo, r.id_luthier, r.id_instrumento,
    i.n_serie, i.disponible, i.id_tipo_inst,
    ti.tipo, ti.dias_revision, ti.costo_rev, ti.id_luthier,
    l.nom_completo, l.dni

    from revisiones r
    join instrumentos i on (i.id = r.id_instrumento)
    join tipos_instrumento ti on (ti.id = i.id_tipo_inst)
    join luthieres l on (l.id = r.id_luthier);
```

Aquí termina el archivo punto7.sql

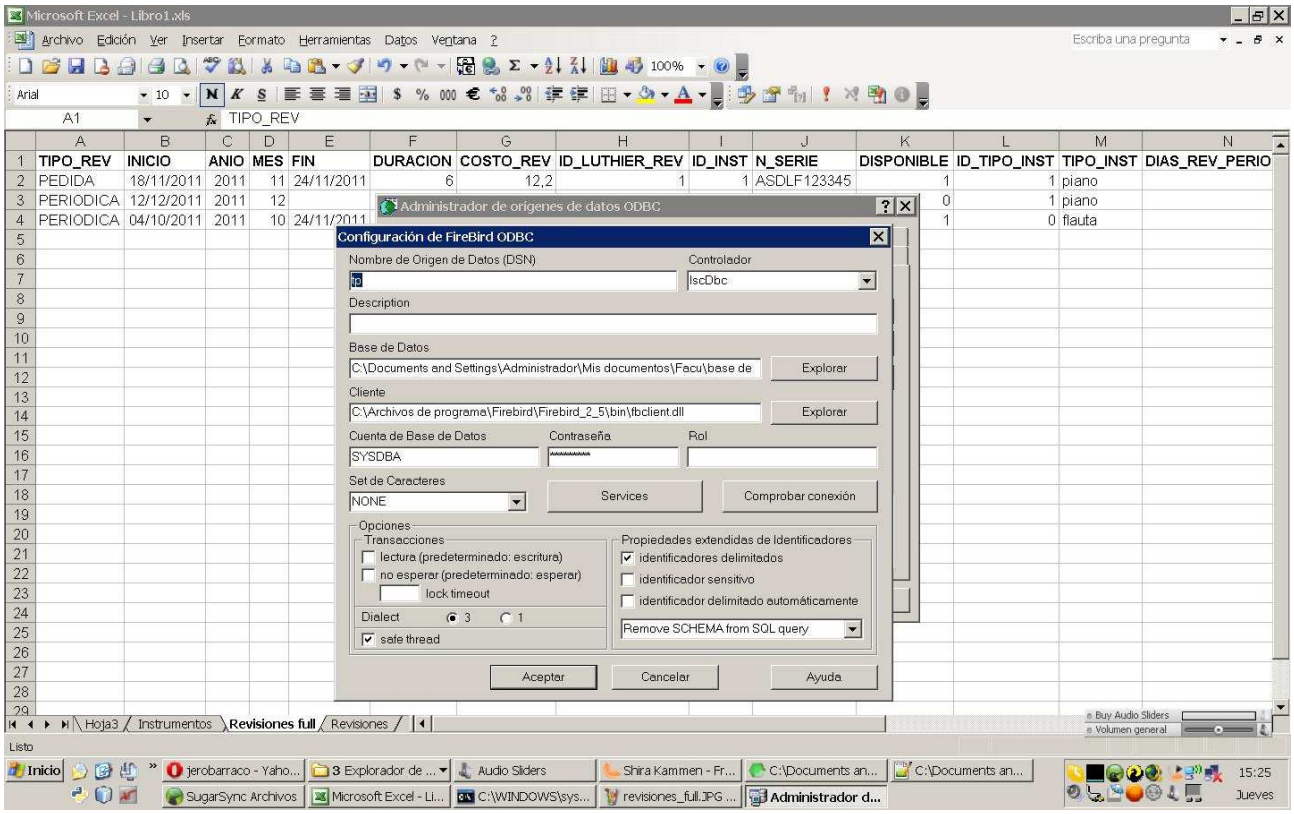
Capturas de la conexión ODBC



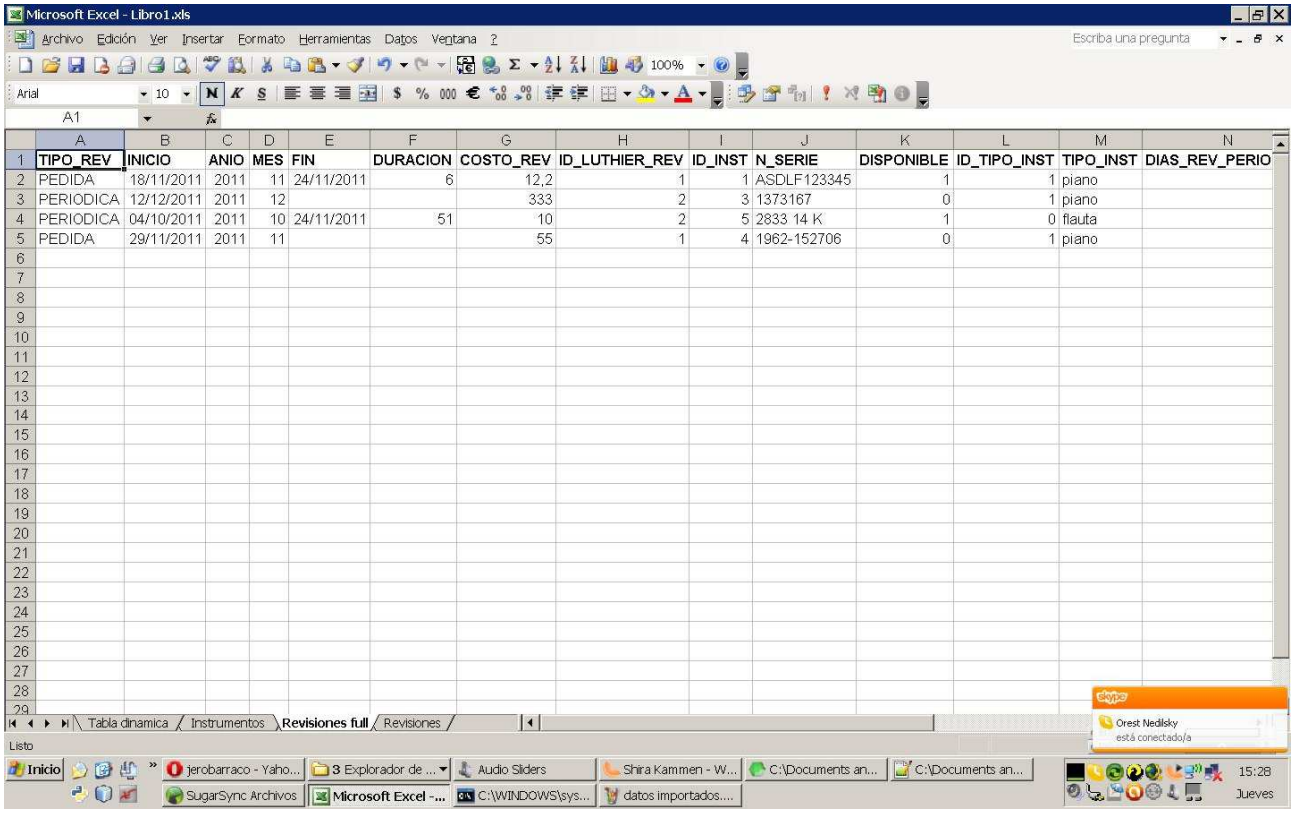
En esta imagen se ve como se hacen las revisiones.



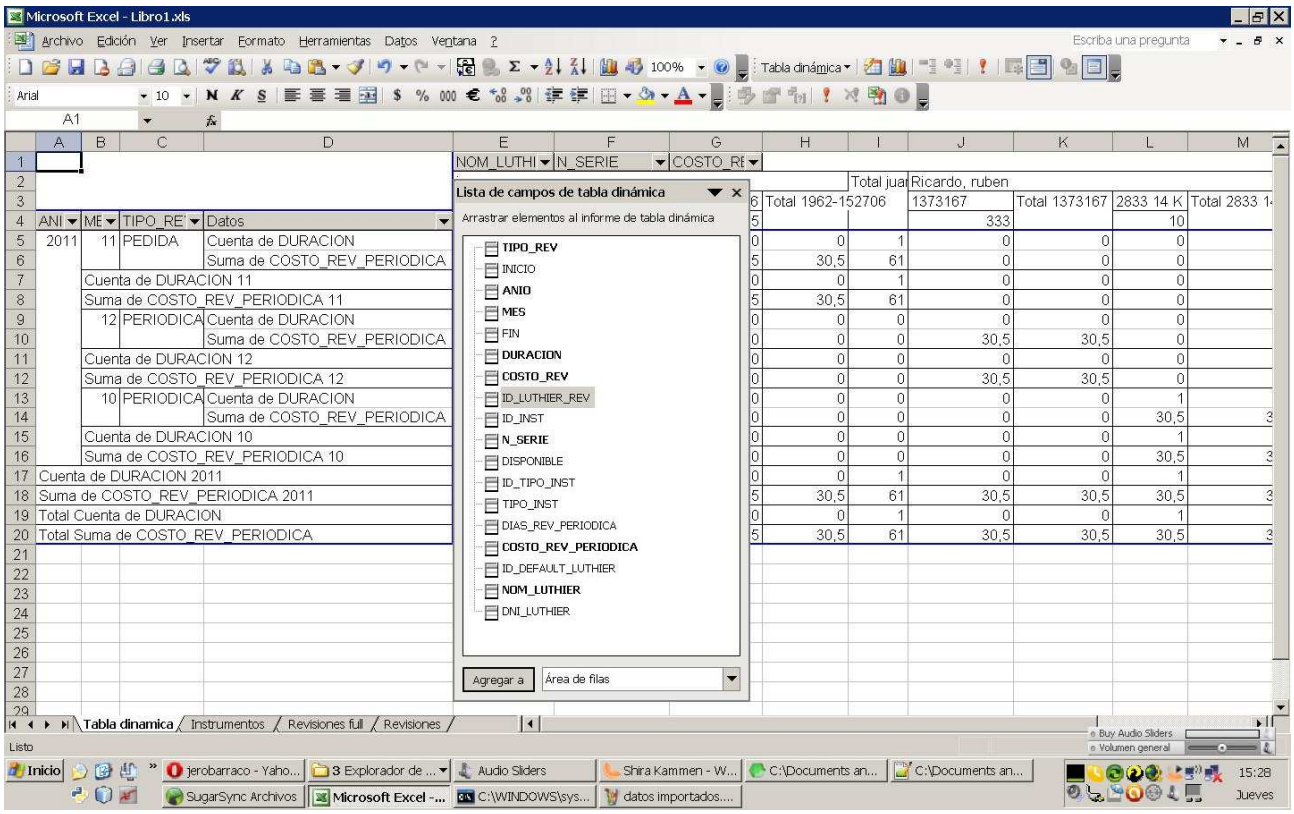
En esta otra se ve como se hacen la totalidad de las revisiones. (revisiones en “full”)



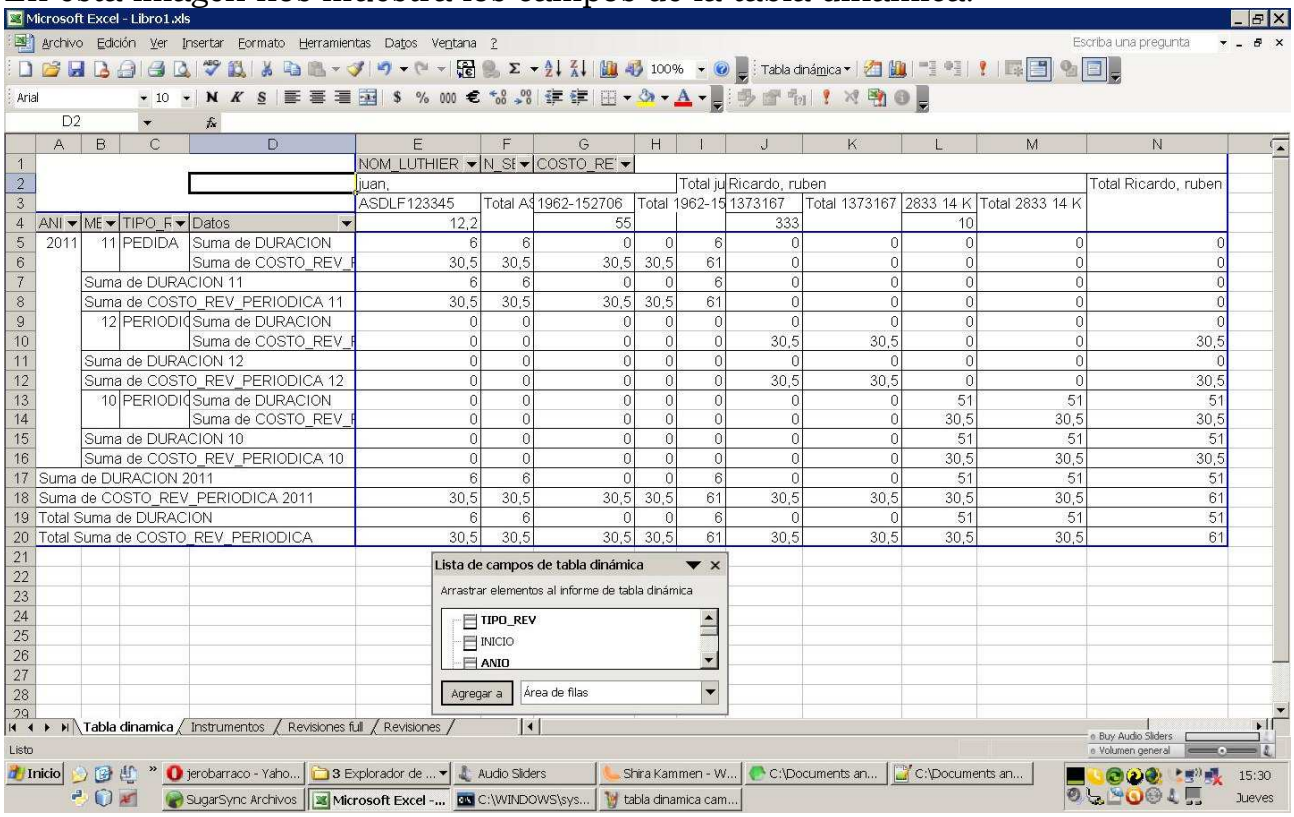
Aquí vemos la configuración de ODBC.



Aquí la imagen nos muestra los datos importados.



En esta imagen nos muestra los campos de la tabla dinámica.



Y en esta última tabla dinámica vemos los datos en la misma.