Universidad del Este - La Plata

Facultad de Ciencias Humanas

Actividad de Extensión

DIPLOMATURA EN SOFTWARE LIBRE

Docente coordinador: Mg. Lic. Mariano Reingart

Base de Datos PostgreSQL

Examen

Francisco Silva Garcés

2015

Consigna

Realizar un documento similar al “Caso de Estudio2” (licencia GNU FDL 1.2+ para su eventual publicación) contemplando (10 puntos c/u):

1. Presentación y relevamiento del caso de estudio
2. Diagrama Entidad-Relación y transformación al Modelo Relacional (normalizado)
3. Esquema Relacional y Álgebra relacional
4. Lenguaje SQL
    4.1. Sentencias DDL de creación de tablas
    4.2. Sentencias DML de manipulación de datos.
5. Análisis comparativo PostgreSQL vs MySQL, FireBird, Oracle, MS SQL Server, u otro (licenciamiento, soporte, comunidad, características, valoración, etc.)
6. Consideraciones implementación en PostgreSQL (respaldo y restauración, tipos de datos, seguridad, integridad, procedimientos almacenados, índices y análisis de rendimiento, mantenimiento, etc.)
7. Desarrollo de aplicación en Python con conexión a la base de datos (importación, procesamiento, búsqueda, reportes, visualización, exportación, o similar)
8.Análisis DAFO y conclusiones finales

Adicionalmente al apunte de UOC, se pide (de ser posible):

- Los datos iniciales deberían poder importarse con COPY, y luego utilizar INSERT /
- UPDATE / DELETE para normalizar esos datos en sus respectivas tablas. El esquema
- inicial debe contener todos los campos, y luego deben ejecutarse los ALTER / DROP
- TABLE para eliminar los campos o tablas redundantes.
- Realizar y crear una vista para cada consultas SQL relevantes
- Resultado de la ejecución de dichos ejercicios (incluir SQL y datos devueltos -ej-)
- Convertir las consultas de SQL a Álgebra Relacional
- Definir las Dependencias Funcionales y Normalizar intuitivamente (3FN o FNBC)
- Análisis de ejecución de consulta (EXPLAIN ANALIZE gráfico explicando cada
- nodo) de una consulta no trivial. Indicar que índices se podrían agregar (SQL).
- Integridad, Validaciones y Restricciones: claves primarias (PRIMARY KEY) y
- foráneas (FOREIGN KEY), no nulo (NOT NULL), (CHECK), política de
- actualización: en cascada (CASCADE), política de eliminación: asignación de valores
- nulos (SET NULL), que correspondan
- Disparador (trigger) para actualizar automáticamente campos calculados o similares
- SQL de Creación de Usuarios (por ej. CREATE USER) y SQL para otorgar/denegar
- Permisos de Acceso (GRANT/REVOKE):
- En el caso de los campos calculados, recomendaciones a realizar al desarrollador
- cuando tenga que consultarlos o actualizarlos. Indicar casos de uso para SELECT ...
- FOR UPDATE/SHARE y LOCK TABLE.
- Realizar un pequeño script SQL de mantenimiento, que limpie las filas obsoletas y
- actualice las estadísticas (usando VACUUM). Ejecútelo y comente la salida.

Índice de contenido

1. Presentación y relevamiento del caso de estudio

Se toman los datos de los resultados definitivos de las elecciones a legisladores porteños desarrollados en el 2013, que están subidos en la página del Gobierno de la Ciudad. (http://data.buenosaires.gob.ar/dataset/elecciones-2013)

Los datos proporcionados están subidos en archivos de formato cvs, texto plano separado por ";" en tres archivos:

1. Codigo elecciones 2013: Listado de códigos de partidos políticos y votos para las elecciones.
2. Establecimientos_Elecciones_Legisladores_2013: Establecimientos donde se llevaron a cabo las elecciones a legisladores.
3. Resultados Elecciones Legisladores 2013: Resultados definitivos de las elecciones a legisladores porteños 2013.

Los archivos tienen la siguiente estructura:

codigo-elecciones-caba-2013.csv

codigo: código del partido político

descripcion: nombre del partido político.

	A	B
1	**codigo**	**descripcion**
2	5	DEMOCRATA_CRISTIANO
3	63	ES_POSIBLE
4	68	NUEVA_IZQUIERDA
5	179	ACCION_CIUDADANA
6	187	AUTODETERMINACION_Y_LIBERTAD
7	262	EL_MOVIMIENTO

establecimientos-caba-2013.csv

x: Cordenada geoespacial (longitud)

y: Cordenada geoespacial (latitud)

distrito: Nombre del distrito

comuna: Número de la comuna

circuito: Número del circuito

establecimiento: Nombre del establecimiento

mesa desde: El numero inicial del rango de mesas del establecimiento

mesa hasta: El numero final del rango de mesas del establecimiento

cantidad de mesas: Cantidad de mesas que tiene el establecimiento

direccion: Dirección del establecimiento.

	A	B	C	D	E	F	G	H	I	J
1	X	Y	DISTRITO	COMUNA	CIRCUITO	ESTABLECIMIENTO	MESA DESDE	MESA HASTA	CANTIDAD DE MESAS	DIRECCIÓN
2	-58.441351539196674	-34.598703651498163	CAPITAL FEDERAL	15	160	COLEGIO BUENOS AIRES	6987	6991	5	ACEVEDO 357
3	-58.43875197064046	-34.596808161709312	CAPITAL FEDERAL	15	159	COLEGIO PAIDEIA	6955	6957	3	ACEVEDO 566
4	-58.443644043058718	-34.589337439627464	CAPITAL FEDERAL	15	163	ENET Nº34 E MARTIN HERMITTE	7140	7155	16	AGUIRRE 1473
5	-58.439040380325828	-34.601088038028351	CAPITAL FEDERAL	15	160	ESC Nº22 DR ROMULO S NAON	7018	7022	5	ARAOZ 234
6	-58.372807959838362	-34.622409328219327	CAPITAL FEDERAL	1	1	ESC N°3 BERNARDINO RIVADAVIA	18	31	14	BOLIVAR 1235
7	-58.373318132700348	-34.611859603403147	CAPITAL FEDERAL	1	14	ESC POLITECNICA Nº5 M BELGRANO	378	381	4	BOLIVAR 346
8	-58.411245459498083	-34.651117916343388	CAPITAL FEDERAL	4	47	NTRA SRA DE LA DIV PROVIDENCIA	1584	1588	5	CACHI 724
9	-58.413157883962207	-34.644218528803385	CAPITAL FEDERAL	4	57	ESC N°14 PROVINCIA DE SAN LUIS	1925	1936	12	CACHI 77
10	-58.43627945305164	-34.593225124493081	CAPITAL FEDERAL	15	158	INST SAN J DE LA PALABRA DIOS	6885	6895	11	CASTILLO 767
11	-58.509247852385052	-34.621746511735402	CAPITAL FEDERAL	10	104	ESCUELA N°2 ALEJANDRO AGUADO	4263	4270	8	CERVANTES 1911
12	-58.457414916678246	-34.557899464244258	CAPITAL FEDERAL	13	139	ENET N°28 REP FRANCESA	5854	5869	16	CUBA 2410
13	-58.4606286352697	-34.615308508015232	CAPITAL FEDERAL	7	80	ESC Nº12 FACUNDO ZUVIRIA	3085	3100	16	FRANKLIN 1836

resultado-caba-elecciones-2013.csv

comuna: Número de la comuna

circuito: Número del circuito

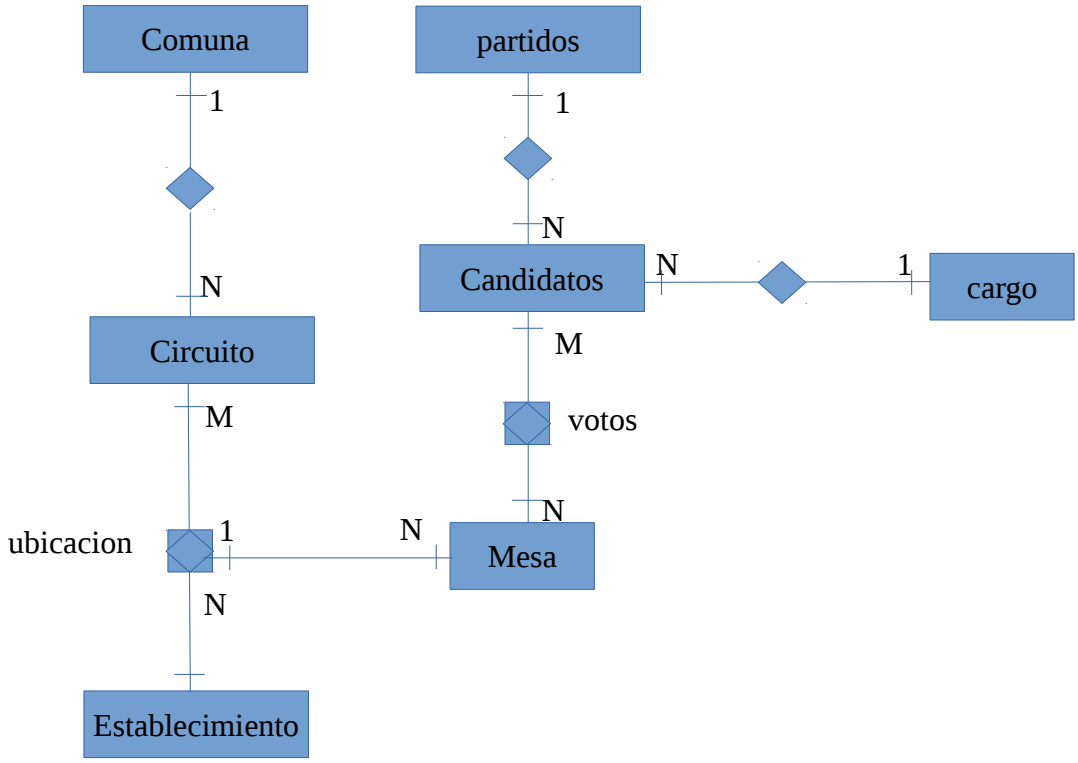
mesa: Número de la mesa

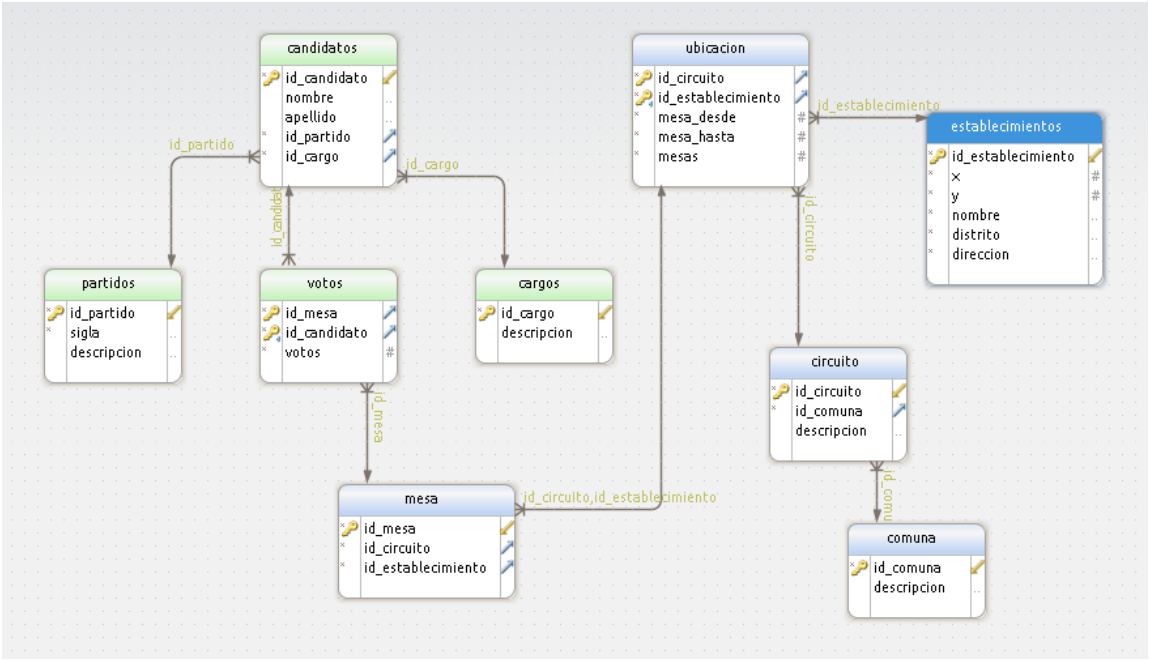
codigo: Código del partido político

votos: Cantidad de votos acumulados.

	A	B	C	D	E
1	COMUNA	CIRCUITO	MESA	CODIGO	VOTOS
2	1	1	1	5	2
3	1	1	2	5	2
4	1	1	3	5	0
5	1	1	4	5	0
6	1	1	5	5	1
7	1	1	6	5	0
8	1	1	7	5	0

2. Diagrama Entidad-Relación y transformación al Modelo Relacional (normalizado)





3. Modelo Relacional y Álgebra Relacional

Se procede a determinar los dominios, las relaciones y sus atributos para tener una imagen inicial y clara de los elementos de que constituirán la base de datos.

El modelo relacional nos ayuda proporcionando una estructura de los datos, a manera de una cantidad de relaciones con el fin de representar la información del mundo real que nos interesa. Además que resulta de fácil entendimiento para los usuarios.

Las relaciones están formadas por dominios (tipos de datos), esquemas de relación (cabecera de la tabla), extensión de la relación (datos de la tabla).

El esquema de la relación se la va a denotar con la nomenclatura R(A1, A2, … An), donde R es el nombre de la tabla y los Ai son los atributos, es decir, el esquema (la cabecera de la tabla), y el dominio vendría a ser el tipo de dato de cada atributo Ai.

En base a estos conceptos teóricos se procede a detallar estos elementos del caso propuesto:

PARTIDOS(id_partido, sigla, descripcion)
CARGOS(id_cargo, descripcion)
CANDIDATOS(id_candidato, nombre, apellido, id_partido, id_cargo)

COMUNA(id_comuna, descripcion)
CIRCUITO(id_circuito,id_comuna, descripcion)
ESTABLECIMIENTOS(id_establecimiento, **x,y,direccion,** nombre, distrito)

UBICACION(id_circuito, id_establecimiento, mesa-desde, mesa-hasta, mesas)
MESA(id_mesa, id_circuito, id_establecimiento)
VOTOS(id_mesa, id_candidato, voto)

Una vez descritos los esquemas de relación, procedemos con los dominios de los atributos, es decir, los tipos de datos.

PARTIDOS dominio(id_partido)=numero dominio(siglas)=texto dominio(descripcion)=descripcionComicios	CARGOS dominio(id_cargo)=codigoCargo dominio(descripcion)=numero
CANDIDATOS dominio(d_candidato)=codigoCandidato dominio(nombre)=texto dominio(apellido)=texto dominio(id_partido)=texto dominio(id_cargo)=codigoCargo	COMUNA dominio(id_comuna)=codigoComuna dominio(descripcion)=texto
CIRCUITO dominio(id_circuito)=codigoCircuito dominio(id_comuna)=codigoComuna dominio(descripcion)=texto	ESTABLECIMIENTOS dominio(id_establecimiento)=codigoEstablecimiento dominio(x)=coordenadasx dominio(y)=coordenadasy dominio(direccion)=direccionEstablecimiento dominio(nombre)=nombreEstablecimiento dominio(distrito)=nombreDistrito

UBICACION dominio(id_circuito)=codigoCircuito dominio(id_establecimiento)=codigoEstablecimiento dominio(mesa-desde)=numero dominio(mesa-hasta)=numero dominio(mesas)=numero	MESA dominio(id_mesa)=codigoMesa dominio(id_circuito)=codigoCircuito dominio(id_establecimiento)=codigoEstablecimiento
VOTOS dominio(id_mesa)=codigoMesa dominio(id_candidato)=codigoCandidato dominio(voto)=numero	

Claves primarias

PARTIDOS Clave primaria: {id_partido} Clave candidata: {id_partido}	CARGOS Clave primaria: {id_cargo} Clave candidata: {id_cargo}
CANDIDATOS Clave primaria: {id_candidato} Clave candidata: {id_candidato}	COMUNA Clave primaria: {id_comuna} Clave candidata: {id_comuna}
CIRCUITO Clave primaria: {id_circuito} Clave candidata: {id_circuito}	ESTABLECIMIENTOS Clave primaria: {id_establecimiento} Clave candidata: {id_establecimiento}
UBICACION Clave primaria: {id_circuito, id_establecimiento} Clave candidata: {id_circuito, id_establecimiento}	MESA Clave primaria: {id_mesa} Clave candidata: {id_mesa}
VOTOS Clave primaria: {id_mesa, id_candidatos} Clave candidata: {id_mesa, id_candidatos}	

Para concluir esta primera parte se terminará describiendo las claves foráneas de los esquemas relaciones:

CANDIDATOS
Su clave foránea son {id_partido}, que define su relación con el atributo {id_partido} de la relación PARTIDO, y {id_cargo} que define su relación con el atributo {id_cargo} de la relación CARGO.

CIRCUITO
Su clave foránea es {id_comuna} que define su relación con el atributo del mismo nombre de la relación COMUNA

UBICACION
Su claves foráneas son {id_circuito} que define su relación con el atributo {id_circuito} de la relación CIRCUITO, y {id_establecimiento} que define su relación con el atributo {id_establecimiento} de la relación ESTABLECIMIENTO.

MESA
Su claves foráneas son {id_circuito, id_establecimiento} que define su relación con los atributos {id_circuito, id_establecimiento} de la relación UBICACION.

VOTOS
Sus claves foráneas son {id_mesa}, que define su relación con el atributo {id_mesa} de la relación MESA, y {id_candidato} que define su relación con el atributo {id_candidato} de la relación CANDIDATO.

Reglas de Integridad

1. no se permitirán valores nulos
2. la integridad referencial está sostenida por las claves primarias y foráneas.
3. no se aplicarán borrados en cascada
4. no se aplicarán actualizaciones en cascada

Algebra relacionales
Inspirada en la teoría de conjuntos, permite especificar consultas en una base de datos relacional. Por lo tanto mediante operaciones de álgebra relacional vamos a especificar los pasos requeridos para realizar un conjunto determinado de consultas.

Como ejemplo se presentarán tres consultas entre todas las posibilidades:

1. Obtener los candidatos participantes
2. Obtener los candidatos votados en la comuna 5
3. Obtener los candidatos votados en el circuito 100

1. Obtener lo candidatos participantes de los comicios en curso.

CA(candidatoID, nombre, apellido):=CANDIDATOS(id_candidato, nombre, apellido)
R:=CA[candidatoID=id_candidato]VOTOS

2. Obtener los candidatos votados en la comuna 5

CI:=CIRCUITO(id_comuna=5)
ME:=MESA * CI
R:=CADIDATOS[id_candidato, nombre, apellido] * ME

3. Obtener los candidatos votados en el circuito 25

CI:=CIRCUITO(id_circuito=25)
ME:=MESA * CI
R:=CANDIDATOS[id_candidato, nombre, apellido] * ME

4. Lenguaje SQL

A continuación se presentan las sentencias en lenguaje SQL para la creación de los objetos en la base de datos del caso propuesto.

4.1 Sentencias de Definición

4.1.1 Creación de las bases de datos

```
CREATE DATABASE ELECCIONES
  WITH OWNER = francisco
       ENCODING = 'UTF8'
       TABLESPACE = pg_default
       LC_COLLATE = 'en_US.UTF-8'
       LC_CTYPE = 'en_US.UTF-8'
       CONNECTION LIMIT = -1;
```

4.1.2 Creación de tablas.

```
CREATE TABLE PARTIDOS (
     id_partido      CHARACTER VARYING(10) NOT NULL,
     sigla      CHARACTER VARYING(5) NOT NULL,
     descripcion     CHARACTER VARYING(2048),
     PRIMARY KEY(id_partido)
);

CREATE TABLE CARGOS (
     id_cargo   INTEGER NOT NULL,
     descripcion     CHARACTER VARYING(2048),
     PRIMARY KEY(id_cargo)
);

CREATE TABLE CANDIDATOS (
     id_candidato    CHARACTER VARYING(10) NOT NULL,
     nombre     CHARACTER VARYING(2048),
     apellido   CHARACTER VARYING(2048),
     id_partido CHARACTER VARYING(10) NOT NULL,
     id_cargo   INTEGER NOT NULL,
     PRIMARY KEY(id_candidato),
     FOREIGN KEY(id_partido) REFERENCES PARTIDOS(id_partido),
     FOREIGN KEY(id_cargo) REFERENCES CARGOS(id_cargo)
);
CREATE TABLE COMUNA (
     id_comuna  INTEGER NOT NULL,
     descripcion     CHARACTER VARYING(2048),
     PRIMARY KEY(id_comuna)
```

```
);

CREATE TABLE CIRCUITO (
      id_circuito     INTEGER NOT NULL,
      id_comuna  INTEGER NOT NULL,
      descripcion     CHARACTER VARYING(2048),
      PRIMARY KEY(id_circuito),
      FOREIGN KEY(id_comuna) REFERENCES COMUNA(id_comuna)
);

CREATE TABLE ESTABLECIMIENTOS (
      id_establecimiento    INTEGER NOT NULL,
      x               DECIMAL NOT NULL,
      y               DECIMAL NOT NULL,
      nombre                CHARACTER VARYING(2048) NOT NULL,
      distrito        CHARACTER VARYING(2048) NOT NULL,
      direccion       CHARACTER VARYING(2048) NOT NULL,
      PRIMARY KEY     (id_establecimiento)
);

CREATE TABLE UBICACION (
      id_circuito           INTEGER NOT NULL,
      id_establecimiento    INTEGER NOT NULL,
      mesa_desde      INTEGER NOT NULL,
      mesa_hasta      INTEGER NOT NULL,
      mesas           INTEGER NOT NULL,
      PRIMARY KEY(id_circuito, id_establecimiento),
      FOREIGN KEY(id_establecimiento) REFERENCES
ESTABLECIMIENTOS(id_establecimiento),
      FOREIGN KEY(id_circuito) REFERENCES CIRCUITO(id_circuito)
);

CREATE TABLE MESA (
      id_mesa         INTEGER NOT NULL,
      id_circuito     INTEGER NOT NULL,
      id_establecimiento    INTEGER NOT NULL,
      PRIMARY KEY(id_mesa),
      FOREIGN KEY(id_circuito, id_establecimiento)
            REFERENCES UBICACION(id_circuito, id_establecimiento)
);

CREATE TABLE VOTOS (
      id_mesa         INTEGER NOT NULL,
      id_candidato    CHARACTER(10) NOT NULL,
      votos      INTEGER NOT NULL,
      PRIMARY KEY (id_mesa, id_candidato),
      FOREIGN KEY(id_mesa) REFERENCES MESA(id_mesa),
      FOREIGN KEY(id_candidato) REFERENCES CANDIDATOS(id_candidato)
);
```

4.1.3 Creación de Vistas.

```
CREATE VIEW votos_comuna_candidato AS (
SELECT
     ci.id_comuna, co.descripcion, vo.id_candidato, ca.nombre,
     sum(votos) as total
FROM
     votos vo, mesa me, circuito ci, candidatos ca, comuna co
WHERE
     vo.id_mesa          = me.id_mesa        AND
     me.id_circuito      = ci.id_circuito    AND
     vo.id_candidato     = ca.id_candidato   AND
     ci.id_comuna        = co.id_comuna
GROUP BY
     ci.id_comuna, co.descripcion, vo.id_candidato, nombre

ORDER BY  sum(votos) DESC
);
```

4.2 Sentencias DML

4.2.4 Inserción manual de datos

Nuevo candidato
INSERT INTO CANDIDATO VALUES('Francisco','Silva',1,2)

Nuevo circuito
INSERT INTO CIRCUITO VALUES(25,2,'circuito 25')

Obtener los 3 candidatos más votados por circuito en una comuna determinada.

```
SELECT
        ca.id_candidato, ca.nombre, ca.apellido, ci.descripcion, sum(votos)
FROM
        candidatos ca, circuito ci, votos vo, mesa me, comuna co
WHERE
        vo.id_candidato     = ca.id_candidato     AND
        vo.id_mesa          = me.id_mesa          AND
        me.id_circuito      = ci.id_circuito      AND
        ci.id_comuna        = co.id_comuna        AND
        co.id_comuna        = 15
GROUP BY
        ca.id_candidato, ca.nombre, ca.apellido, ci.descripcion
ORDER BY sum(votos) DESC
LIMIT 3
```

4.3. Importación de datos.

En esta etapa se procede a subir todos los archivos descargados del sitio web de Gobierno de la Ciudad proporcionados en formato csv.

Una vez subidos se procederá en base a sentencias SQL realizar la carga de datos al esquema normalizado.

4.3.1 Carga de archivos no-normalizados.

Se procede a crear cada tabla en la misma base de datos, y a continuación la carga de los datos. Este proceso de itera por cada archivo.

```
CREATE TABLE codigo_elecciones_caba_2013
(
      codigo integer,
      descripcion varchar(1024)
);

copy codigo_elecciones_caba_2013
from '/home/francisco/Documents/OTROS ESTUDIOS/DIPLOMATURA-SL/3.
Basededatos/Examen/codigo-elecciones-caba-2013.csv'
with  csv
      delimiter ';'
      header
      encoding 'latin1';


CREATE TABLE establecimientos_caba_2013 (
      x     decimal,
      y     decimal,
      distrito    character(2048),
      comuna            integer,
      circuito    integer,
      establecimiento character(2048),
      mesa_desde integer,
      mesa_hasta integer,
      cantidad_mesa    integer,
      direccion   character(2048)
);

copy establecimientos_caba_2013
from '/home/francisco/Documents/OTROS ESTUDIOS/DIPLOMATURA-SL/3.
Basededatos/Examen/establecimientos-caba-2013.csv'
with  csv
      delimiter ';'
      header
      encoding 'latin1';

CREATE TABLE resultado_caba_elecciones_2013 (
```

```
      comuna          integer,
      circuito   integer,
      mesa       integer,
      codigo          character(10),
      votos      integer
);

copy resultado_caba_elecciones_2013
from '/home/francisco/Documents/OTROS ESTUDIOS/DIPLOMATURA-SL/3.
Basededatos/Examen/resultado-caba-elecciones-2013.csv'
with  csv
      delimiter ';'
      header
      encoding 'latin1';
```

4.3.2 Carga de datos a las tablas normalizadas.

```
INSERT INTO partidos
      SELECT     codigo, 'XXXXX' as abrev, descripcion
      FROM codigo_elecciones_caba_2013;

-----------------------

INSERT INTO cargos VALUES(100,'Legislador Ciudad de Buenos Aires');

-----------------------

INSERT INTO candidatos
      SELECT codigo, descripcion, 'XXXXX', codigo, 100 FROM
codigo_elecciones_caba_2013;

-----------------------

INSERT INTO comuna
      SELECT DISTINCT comuna, 'Comuna No ' || comuna
      FROM resultado_caba_elecciones_2013
      WHERE comuna is not null;

-----------------------

INSERT INTO circuito
      SELECT DISTINCT ON (circuito)
      circuito, comuna, 'Circuito No. ' || circuito as desc
      FROM establecimientos_caba_2013
      WHERE comuna is not null;

-----------------------

CREATE SEQUENCE establecimientos_id;
```

```
ALTER TABLE establecimientos ALTER COLUMN id_establecimiento SET DEFAULT
nextval('establecimientos_id');
ALTER SEQUENCE establecimientos_id OWNED BY
establecimientos.id_establecimiento INCREMENT BY 1 MINVALUE 0 START WITH 1;

INSERT INTO establecimientos (x,y,nombre, distrito, direccion)
     SELECT DISTINCT ON (x,y,establecimiento)
     x,y,establecimiento, distrito, direccion
     FROM establecimientos_caba_2013 ;
INSERT INTO establecimientos VALUES (0,0,0,'no definido','no definido','no
definido');
select min(id_establecimiento), max(id_establecimiento) from
establecimientos;

-----------------------

INSERT INTO UBICACION
     SELECT DISTINCT ON (circuito, ide)
          circuito,
          (select id_establecimiento from establecimientos where x = e.x
and y = e.y and nombre = establecimiento) as ide,
          mesa_desde, mesa_hasta, cantidad_mesa
     FROM establecimientos_caba_2013 as e; --where mesa_desde = 86;
INSERT INTO UBICACION VALUES(21,0,0,0,0);
INSERT INTO UBICACION VALUES(92,0,0,0,0);

-----------------------

INSERT INTO MESA
     SELECT DISTINCT ON (mesa, circuito)
          mesa,
          circuito,
          COALESCE((select id_establecimiento FROM ubicacion WHERE mesa
between mesa_desde and mesa_hasta),0) as id_e
     FROM resultado_caba_elecciones_2013
     WHERE mesa > 0 and mesa < 8000 and mesa is not null;

-----------------------

INSERT INTO VOTOS
     SELECT mesa, codigo, sum(coalesce(votos,0))
     FROM resultado_caba_elecciones_2013
     WHERE mesa <> 0 and codigo is not null and codigo <> 'A' and mesa <
8000
     GROUP BY mesa, codigo;
```

5. Análisis comparativo

PostgreSQL vs Oracle
Una de las bases de datos comerciales mayormente difundida por su enorme inversión publicitaria y tecnológica es Oracle, por lo tanto se la ha seleccionado como referente para un análisis comparativo respecto a las características técnicas empresariales.

	Oracle	**PostgreSQL**
Modelo de base de datos	Relacional	Objeto-Relacional
Documentación	Manuales disponible. Documentación completa de parámetros de tuning dispobible solo para técnicos de oracle.	Todo tipo de información disponible y completa.
Fabricante	Oracle	PostgreSQL Global Development Group
Versiones iniciales	1980	1989
Versión actual	12 Release 1 (12.1.0.2), July 2014	9.4.4, June 2015
Licencia	Commercial	Open Source - BSD
Database as a Service (DBaaS)	no	no
Lenguaje de implementación	C and C++	C
Sistemas Operativos	AIX HP-UX Linux OS X Solaris Windows z/OS	FreeBSD HP-UX Linux NetBSD OpenBSD OS X Solaris Unix Windows
Data scheme	yes	yes
Secondary indexes	yes	yes
SQL	yes	yes
NoSQL like feature	no	Desde versión 9.5
APIs métodos de acceso	ODP.NET Oracle Call Interface (OCI) JDBC ODBC	native C library streaming API for large objects ADO.NET JDBC ODBC
Lenguajes de programación soportados.	C, C#, C++, Clojure, Cobol Eiffel, Erlang, Fortran, Groovy, Haskell, Java, JavaScript, Lisp,	.Net, C, C++, Java, Perl, Python, Tcl, PHP

	Objective C, Ocaml, Perl, PHP, Python, R, Ruby, Scala, Tcl, Visual Basic	
Programación de lado del servidor	PL/SQL	PL/pgSQL, perl, python, tcl
Método de particionamiento	horizontal partitioning	no, pero puede realizarse usando herencia de tablas.
Método de replicación	Master-master replication Master-slave replication	Master-slave replication
MapReduce	no	no
Concepto transaccional	ACID	ACID
Concurrencia	MVCC	MVCC
In-memory capabilities	yes	no
Soporte	Empresas, foros, online	Foros, comunidad, online, listas de discusión, empresas.
Tamaños limite(max) **tablas :** **base de datos :** **registro :** **columna por registro :** **blob/clob :**	 4GB ilimitado 8K 1000 128 TB	 32 TB ilimitado 1.6TB 250-1600 1 GB (en linea), 4TB (pg_largeobjetct)

Fuente: http://db-engines.com/en/system/Oracle%3BPostgreSQL
http://www.sql-workbench.net/dbms_comparison.html
http://database-management.softwareinsider.com/compare/36-43/Oracle-vs-PostgreSQL
http://www.enterprisedb.com/hp-compare-features-advanced-server-oracle

6. Consideraciones a tener en cuenta.

Respaldo y Restauración
Es importante definir un plan de respaldo, en función de lo que se quiere proteger como prioridad. Esto podría determinar si los mecanismos de respaldo del gestor de base de datos son los idóneos para cumplir con el objetivo, o si son necesarios otros mecanismos como por ejemplo replicación de datos. Eso en relación al nivel de downtime tolerable, y la cantidad tolerable de información que se pudiera perder. Para los efectos del caso propuesto, no es tolerable perder ni un bit de información, por lo cual se podrá sugerir una combinación de ambos mecanismos.

En todo caso los mecanismos de respaldo y restauración son contemplados para casos de desastre, en las que se considera siempre un downtime. PostgreSQL permite realizar un respaldo de los WAL, y una recuperación en un punto en el tiempo.

Una consideración a tomar en cuenta es que mientras se corre pg_dump, no se podrá deshacer cambios DDL en el sistema de producción, por lo tanto habrá que tener cuidado en considerar esta ventana de mantenimiento.

Para el caso propuesto, la tabla con mayor prioridad para respaldo es la de votos, la cual en caso de algún incidente si no se tiene respaldos sería imposible reconstruirla, el resto de tablas no tienen mayor

movimiento lo cual podrían generarse con el primer respaldo realizado. En cambio la tabla de votos al ser muy dinámica se recomiendo mantener un respaldo periódico, realizando snapshot de la misma.

Tipos de Datos
El gestor de base de datos tiene una amplia gama de tipos de datos, que van desde los tipos estándares entre los distintos gestores de bases de datos del mercado, hasta tipos específicos. Además de permitir la posibilidad de definir tipos de datos propios.

Se recomienda utilizar los tipos estándares que permitan una interoperabilidad menos conflictiva con otros sistemas y gestores de bases de datos.

Seguridad
Los privilegios deben ser definidos con cuidado, puesto que por facilidad se suelen otorgar privilegios abiertos para evitar problemas, sin embargo esto deja brechas de seguridad. Los privilegios deben ser minuciosamente definidos y configurados, para lo cual debe llevarse registro de ello.

Los permisos a nivel de sistema de archivos también son importantes tenerlos en cuenta.

Es importante además habilitar la opción de SSL para las comunicaciones del gestor de base de datos. Además los accesos pueden restringirse por dirección IP o nombre de usuario, característica que puede utilizarse para los casos que amerite reforzar seguridades.

Procedimientos almacenados
PosrgreSQL por defecto utiliza su propio lenguaje procedural (PL/pgSQL) que permite la creación de funciones y procedimientos almacenados en la base de datos. Sin embargo, se pueden utilizar otros lenguajes para la elaboración de código que den ciertas funcionalidades desde el servidor.

Mantenimiento
Toda gestor de base de datos requiere ciertas tareas de mantenimiento mínimas, para su buen funcionamiento y salud operacional. Entre las tareas básicas de mantenimiento podemos considerar:

1. limpieza de indentificadores internos.
2. mantenimiento de las estadísticas del planificador de consultas
3. re-indexación periódicas de tablas.
4. mantenimiento los archivos de registro (logs)

Las tres primeras tareas se realizan mediante un proceso denominado vacuum. La re-indexación aunque no es una tarea muy habitual, puede mejorar significantemente el rendimiento de las consultas complejas sobre todo en tablas con mayor actividad.

Lo archivos de registro o logs, acumulan información de lo que sucede con el gestor de base de datos, y permite determinar un diagnóstico en caso de algún incidente. Estos archivos se acumulan con el tiempo y crecen. Por esto es importante mantener mecanismos de rotación de los archivos de logs para que estos cada cierto tiempo se hagan copias comprimidas para un registro histórico y poderlos vaciar.

7. Desarrollo de aplicación en Python con conexión a la base de datos (importación,procesamiento, búsqueda, reportes, visualización, exportación, o similar)

Programa para reportar los 3 candidatos más votados por comuna.

```
#!/usr/bin/
# -*- coding: latin1 -*-

# carga de librería para conexión con postgresql
import psycopg2.extras

# obtener lista comunas para iterar
def obtenerComunas( cur ):
  cur.execute("SELECT id_comuna FROM comuna")
  regs={}
  for row in cur:
    regs[ row[0] ] = 0

  return regs

# creación de objeto para conexión con la base de datos
conn = psycopg2.connect(database='examen', user='francisco',
password='pancho',host='localhost')

cur = conn.cursor()
comunas = obtenerComunas( cur )

# iterar por cada comuna el query (vista) para obtención de candidato
más votado de una comuna determinada

for i in comunas:
  dato = (i,)
  query = "SELECT * FROM votos_comuna_candidato WHERE id_comuna = %s
LIMIT 3;"
  cur.execute(query,dato)

  print "Comuna No. " + str(i)
  for row in cur:
      print " " * ( 5 ),
      print "%5s" % row[2],
      print "%-30s" % row[3],
      print '{:10,}'.format(row[4])
```

salida de ejecución

```
francisco@pcfsg:~/Documents/OTROS ESTUDIOS/DIPLOMATURA-SL/3. Basededatos/Examen$ python programa1.py
Comuna No. 1
      503       UNIONPRO                        43,999
      502       UNEN                            27,645
      501       FRENTE_PARA_LA_VICTORIA         23,045
Comuna No. 2
      503       UNIONPRO                        43,326
      502       UNEN                            29,897
      501       FRENTE_PARA_LA_VICTORIA         10,665
Comuna No. 3
      503       UNIONPRO                        36,234
      502       UNEN                            26,957
      501       FRENTE_PARA_LA_VICTORIA         22,053
Comuna No. 4
      503       UNIONPRO                        44,299
      501       FRENTE_PARA_LA_VICTORIA         28,040
      502       UNEN                            24,818
Comuna No. 5
      503       UNIONPRO                        33,866
      502       UNEN                            29,160
      501       FRENTE_PARA_LA_VICTORIA         21,902
Comuna No. 6
      503       UNIONPRO                        36,211
      502       UNEN                            33,475
      501       FRENTE_PARA_LA_VICTORIA         19,140
Comuna No. 7
      503       UNIONPRO                        39,293
      502       UNEN                            31,358
      501       FRENTE_PARA_LA_VICTORIA         24,122
Comuna No. 8
      503       UNIONPRO                        35,022
      501       FRENTE_PARA_LA_VICTORIA         24,616
      502       UNEN                            16,931
Comuna No. 9
      503       UNIONPRO                        37,048
      502       UNEN                            26,172
      501       FRENTE_PARA_LA_VICTORIA         22,416
Comuna No. 10
      503       UNIONPRO                        35,321
      502       UNEN                            29,152
      501       FRENTE_PARA_LA_VICTORIA         20,104
Comuna No. 11
      503       UNIONPRO                        41,352
      502       UNEN                            34,752
      501       FRENTE_PARA_LA_VICTORIA         21,396
Comuna No. 12
      503       UNIONPRO                        48,294
      502       UNEN                            35,426
      501       FRENTE_PARA_LA_VICTORIA         22,534
Comuna No. 13
      503       UNIONPRO                        61,476
      502       UNEN                            43,887
      501       FRENTE_PARA_LA_VICTORIA         18,811
Comuna No. 14
      503       UNIONPRO                        57,214
      502       UNEN                            41,736
      501       FRENTE_PARA_LA_VICTORIA         19,354
Comuna No. 15
      503       UNIONPRO                        35,600
      502       UNEN                            29,202
      501       FRENTE_PARA_LA_VICTORIA         22,400
```

8. Análisis DAFO y conclusiones finales

8.1 DAFO

Fortalezas

- Motor de base de datos con extensiones de mucha utilidad
- Variedad de tipos de datos embebidos.
- Cumple con los estándares.
- Trayectoria de ingeniería al a par de los más grandes gestores de bases de datos comerciales.
- Implementa replicación de datos.
- Software libre

Debilidades

- No implementa paralelismo completamente como lo hacen otros motores de base de datos comerciales.
- Escasa documentación en español
- Escasa difusión y pobre estrategia de marketing.
- Casos de éxito poco difundidos.

Amenazas

- Escasa capacidad técnica en países de latinoamérica.
- Implementación completa de replicación de datos y características de alta disponibilidad en bases de datos comerciales.
- Variedad de productos opensource robustos con apoyo comercial tal como MySQL

Oportunidades

- Reconocida entre las bases de datos más robustas del mundo opensource
- Compite con los motores de base de datos comerciales de mayor rango.
- Comunidad extendida de rápida respuesta para corrección de bugs.

8.2 Conclusiones finales

Considero que este motor de base de datos, en su más reciente versión ha evolucionado lo suficiente para competir fuertemente con las productos comerciales mejor ranqueados en el mercado. Un nicho muy importante sería el mundo de las pymes en el mercado Latinoamericano. Sin embargo, este motor adolece del mismo mal del cual sufría el gestor de base de datos Informix, es decir, a pesar de su avanzada tecnología esta era superada comercialmente por otros productos que tenía inversiones millonarias en estrategias de marketing.

Lamentablemente hasta nuestros días, aún se sigue formando a los ingenieros en sistemas o a los desarrolladores con productos comerciales, que es lo que demanda el mercado. Es importante que en las universidades se utilice postgresql como parte de la educación básica en las materias de base de datos, desarrollo de software o afines.

PostgreSQL como motor de base de datos, entra con las versiones 9.x a las grandes ligas, la tecnología ha llegado a su punto de madurez, ahora toca formar especialistas y mejora las estrategias para difundir la tecnología.