DEPARTAMENTO
DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico 1

2 de julio de 2015 — Bases de Datos

Grupo 5

Integrante	LU	Correo electrónico
Vileriño, Silvio	106/12	svilerino@gmail.com
Chapresto, Matias Nahuel	201/12	matiaschapresto@gmail.com
Garassino, Agustín Javier	394/12	ajgarassino@gmail.com

Instancia	Docente	Nota
Primera entrega		
Segunda entrega		

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Ciudad Universitaria - (Pabellón I/Planta Baja)
Intendente Güiraldes 2160 - C1428EGA
Ciudad Autónoma de Buenos Aires - Rep. Argentina
Tel/Fax: (54 11) 4576-3359
http://www.fcen.uba.ar

Índice

1. Introducción

El objetivo de este trabajo practico es el diseño e implementacion de una solucion que satisfaga los requerimientos del cliente. En esta ocasion se nos requiere un sistema que permita implementar el `Voto Electronico` en Elecciones. Para ello, teniendo como base un relevamiento del problema -enunciado-, realizaremos un diagrama de entidad relacion y especificaremos restricciones adicionales que este modelo no capture. Como siguiente paso, derivaremos de un modelo relacional. Teniendo como base el modelo relacional, lo implementaremos en un motor de base de datos propietario, creando las tablas, claves y relaciones adecuadas. Adicionalmente, para satisfacer ciertas restricciones y proveer operaciones consistentes, necesitaremos implementar `triggers`, `stores procedures` y `constraint checks` sobre el motor fisico. Finalmente, proveeremos una suite adecuada de tests que verifiquen la robustez del sistema construido.

2. Diseño

2.1. Diagrama Entidad-Relacion

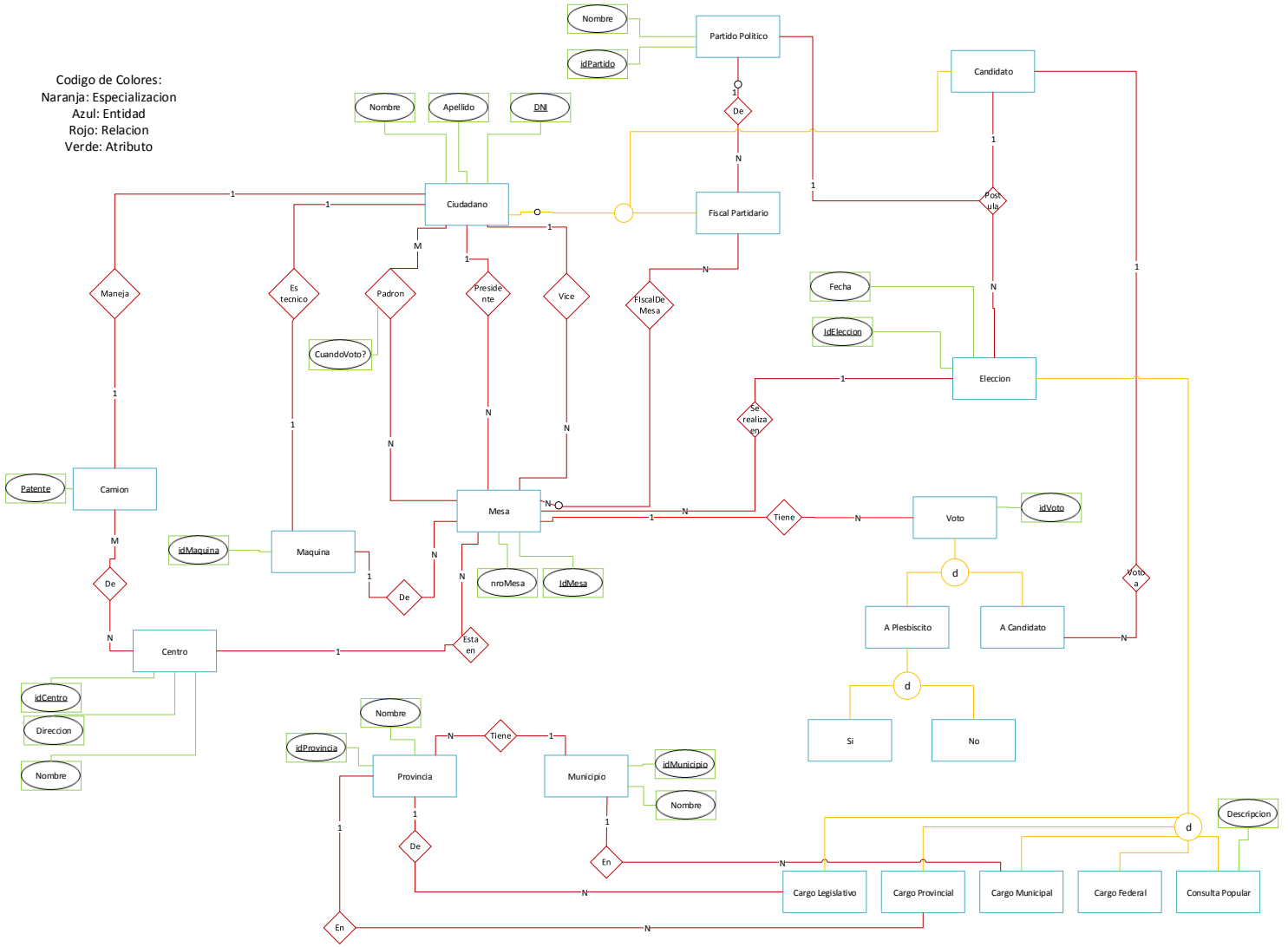

Figura 1: Diagrama Entidad Relacion.

2.2. Supuestos y Restricciones del DER

A continuacion se enumeran los supuestos y restricciones del diagrama entidad-relacion.

1. Toda persona que vota en una mesa, debe tener nulo el campo selloVoto del padron para dicha mesa. ie. No permitir que la gente vote mas de una vez por eleccion.
2. Para voto que se inserta se debe actualizar correctamente la fecha y hora en la que voto y poniendo el "sello" virtual en el padron asignando un valor no nulo a cuandoVoto.
3. La suma de los votos de todas las mesas de todos los candidatos debe ser menor o igual(votos en blanco diferencia) a la cantidad de ciudadanos que tiene el timestamp cuandoVoto? No nulo en el padron de dicha eleccion. (notar que esto tambien lo acota por la cantidad de ciudadanos)
4. Cada partido politico presenta un solo candidato por eleccion.
5. Todos los votos para una eleccion son: o bien consulta popular o bien de tipo candidato según corresponda el tipo de eleccion
6. Si la eleccion es una consulta popular, los votos deben ser si/no. Sino, deben ser candidatos
7. Un voto a candidato tiene que ser a un candidato que este postulado en esa eleccion.
8. Un ciudadano es fiscal, presidente o vicepresidente de una sola mesa por eleccion.
9. Un ciudadano vota en una sola mesa por eleccion.
10. Un ciudadano no pueda ser tecnico, presidente, vicepresidente o fiscal en una misma eleccion.
11. Una maquina funciona en una sola mesa por eleccion.
12. No hay candidatos repetidos en una eleccion.
13. Un ciudadano no puede ser conductor y tecnico, presidente, vicepresidente o fiscal en una misma eleccion.

2.3. Modelo Relacional

Notacion:

- **Primary Key**
- **Not Nullable Foreign Key**
- **Nullable Foreign Key**
- Not Nullable Atribute
- **Nullable Atribute**

Entidades:

- **Eleccion** (idEleccion, Fecha, Tipo)
  - PK = CK = {idEleccion}
- **Eleccion_Consulta_Popular** (idEleccion, Descripcion)
  - PK = CK = FK = {idEleccion}

- **Provincia** (**idProvincia**, Nombre)
  - PK = CK = {idProvincia}
- **Municipio** (**idMunicipio**, Nombre, **idProvincia**)
  - PK = CK = {idMunicipio}
  - FK = {idProvincia}
- **Eleccion_Cargo_Municipal** ((**idEleccion**, **idMunicipio**)
  - PK = CK = {idEleccion}
  - FK = {idEleccion, idMunicipio}
- **Eleccion_Cargo_Provincial** ((**idEleccion**, **idProvincia**)
  - PK = CK = {idEleccion}
  - FK = {idEleccion, idProvincia}
- **Eleccion_Cargo_Legislativo** ((**idEleccion**, **idProvincia**)
  - PK = CK = {idEleccion}
  - FK = {idEleccion, idProvincia}
- **Voto** (**idVoto**, **idMesa**, Tipo)
  - PK = CK = {idVoto}
  - FK = {idMesa}
- **Voto_A_Candidato** (**idVoto**, **DNI**)
  - PK = CK = {idVoto}
  - FK = {idVoto, DNI (references Candidato on DNI)}
- **Ciudadano** (**DNI**, Nombre, Apellido)
  - PK = CK = {DNI}
- **Fiscal_Partidario** (**DNI**, **idPartido**)
  - PK = CK = {DNI}
  - FK = {idPartido}
- **Candidato** (**DNI**)
  - PK = CK = {DNI}
  - FK = {DNI references Ciudadano on DNI}
- **Partido_Politico** (**idPartido**, Nombre)
  - PK = CK = {idPartido}
- **Camion** (**Patente**, **idConductor**)

- PK = CK = {Patente}
- FK = {idConductor (references Ciudadano on DNI)}

- **Centro** (**idCentro**, Nombre_Establecimiento, Direccion)
  - PK = CK = {idCentro}

- **Maquina** (**idMaquina**, **idTecnico**)
  - PK = CK = {idMaquina}
  - FK = {idTecnico (references Ciudadano on DNI)}

- **Mesa** (**idMesa**, nroMesa, **idPresidente**, **idVicepresidente**, **idEleccion**, **idCentro**, **idMaquina**)
  - PK = CK = {idMesa}
  - FK = {idPresidente (references Ciudadano on DNI), idVicepresidente (references Ciudadano on DNI), idEleccion, idCentro, idMaquina}

Relaciones:

- **Camion_Centro** (**patente**, **idCentro**)
  - PK = CK = {(patente, idCentro)}
  - FK = {patente, idCentro}

- **Fiscales** (**DNI**, **idMesa**)
  - PK = CK = {(DNI, idMesa)}
  - FK = {idMesa, DNI (references Ciudadano on DNI)}
  - **Nota:** Puede haber mesas sin fiscal. Mesa participa parcialmente en esta relacion.

- **Postulaciones** (**idEleccion**, **DNI**, **idPartido**)
  - CK = {(idEleccion, DNI), (idEleccion, idPartido)}
  - PK = CK = {(idEleccion, DNI)}
  - FK = {idEleccion, idPartido, DNI (references Candidato on DNI)}

- **Padron** (**idMesa**, **DNI**, **selloVoto**)
  - PK = CK = {(idMesa, DNI)}
  - FK = {idMesa, DNI (references Ciudadano on DNI)}
  - **Nota:** el atributo selloVoto es NULL por defecto y es de tipo datetime.

Notas:

- Todos los atributos en negro son no nullables.
- Las **Primary Keys** del modelo fisico, no son autoincrement(identity) por motivos de testing.
- Todas las relaciones son de participacion total salvo que se explicite lo contrario.

- En la doble especializacion de voto por la rama de consulta popular, consideramos innecesario crear la tabla A_Plesbicito con el campo tipo que indica si el voto es por Si o por No, pues es mas sencillo directamente especializar en el nivel superior, que un voto puede ser de 3 tipos:
    - A Candidato
    - A Consulta Popular, A favor
    - A Consulta Popular, En contra

3. Implementacion

3.1. Modelo Fisico

3.1.1. Motor elegido

El motor elegido para implementar la solucion es **Microsoft Sql Server 2008** junto a sus herramientas graficas.

3.1.2. Diagrama fisico

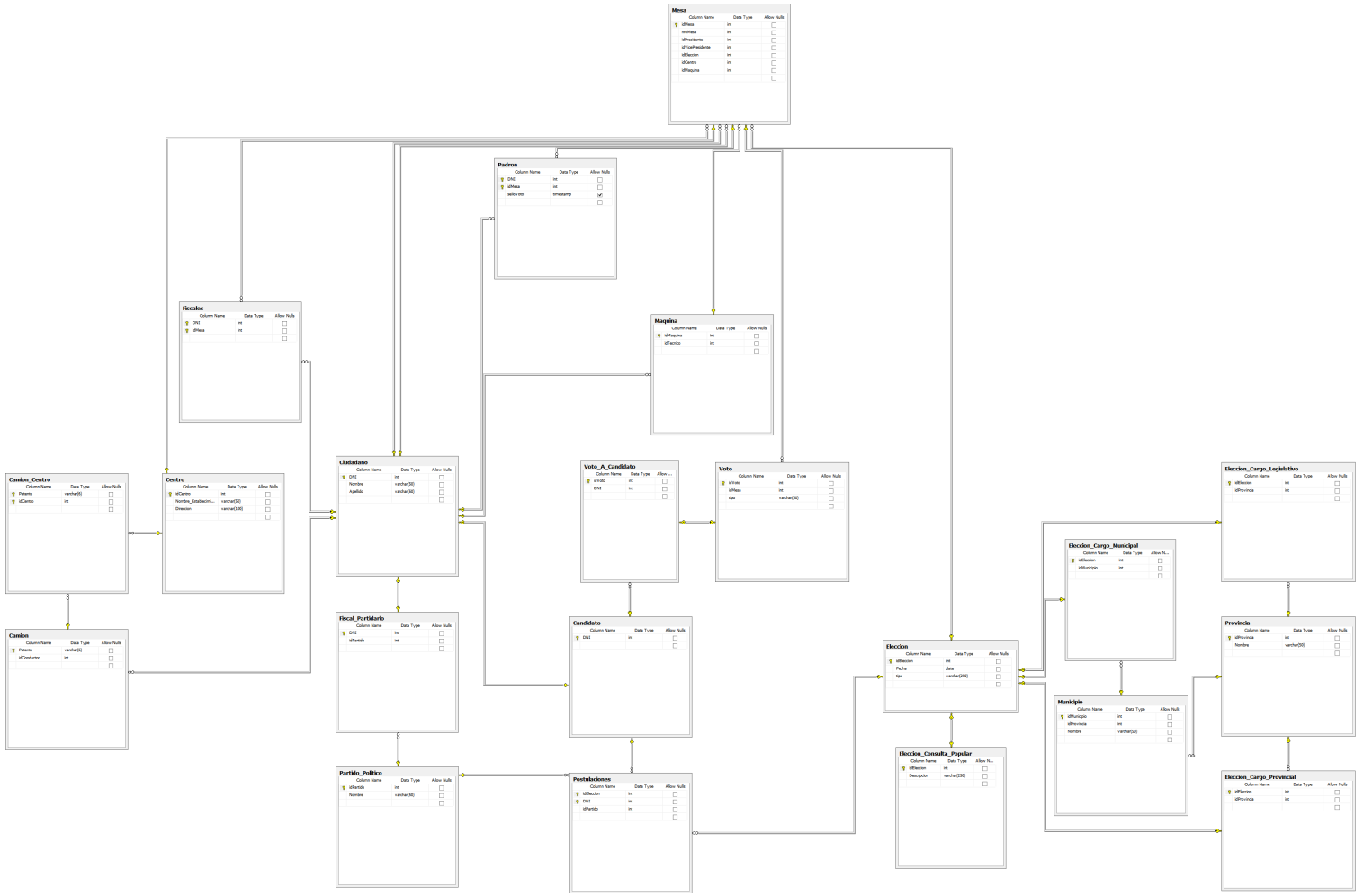

Figura 2: Diagrama fisico.

3.1.3. Modelado fisico de las restricciones con triggers

Modelaremos las restricciones al modelo utilizando funcionalidades provistas por el motor de la base de datos, como ser **Stored Procedures**, **Triggers**, **Checks**, etc. segun nos parezca conveniente. A continuacion se presenta el modelado de las restricciones. La notacion utilizada es: Restricciones resueltas, soluciones itemizadas de forma anidada. **Nota:** Algunas restricciones quedaran expresadas en lenguaje natural en la seccion `Supuestos y Restricciones del DER`.

★ Los campos tipo de las tablas con jerarquia disjunta deben contener los valores correctos.

- ✓ Se crearon **Check Constraints** para validar los campos `tipo` de las tablas que tienen jerarquias disjuntas.

★ Todos los votos para una eleccion son: o bien consulta popular o bien de tipo candidato según corresponda el tipo de eleccion.

★ Si la eleccion es una consulta popular, los votos deben ser si/no. Sino, deben ser candidatos.

- ✓ **After Insert trigger** sobre tabla voto. navegamos a mesa y de mesa a eleccion y chequeamos que el campo tipo de la eleccion corresponda con el campo tipo del voto. Sino, rollbackeamos.

★ Un voto a candidato, debe referenciar a un candidato postulado para dicha eleccion.

- ✓ **After Insert trigger** sobre tabla voto a candidato. Navegamos a Voto, luego a Mesa y obtenemos el idEleccion. Luego buscamos los postulados a dicha eleccion y verificamos que el voto sea valido. Sino, rollbackeamos.

★ No hay candidatos repetidos en una eleccion.

- ✓ Queda determinado por la **PK** de la tabla Postulaciones.

★ Cada partido politico presenta un solo candidato por eleccion.

- ✓ **Check Constraint Unique** (idEleccion, idPartidoPolitico) en postulaciones.

★ La suma de los votos de todas las mesas de todos los candidatos debe ser menor o igual(votos en blanco diferencia) a la cantidad de ciudadanos que tiene el datetime cuandoVoto? No nulo en el padron de dicha eleccion. (notar que esto tambien lo acota por la cantidad de ciudadanos).

- Queda implicada por el stored procedure de voto, que chequea sello null y marca no null el sello luego. No hay forma que de se inserten mas votos que personas.

★ Un ciudadano vota en una sola mesa por eleccion.

✓ **Instead of Insert trigger** en Padron que verifique que para ese DNI, no exista otra mesa, de la misma eleccion, que ya lo tenga asociado. Si no existe la relacion entre dni, mesa y eleccion, se inserta el registro correspondiente.

★ Una maquina funciona en una sola mesa por eleccion.

✓ **Instead of Insert Trigger** en Mesa para ver que la maquina asignada no este asignada a otra mesa en esa eleccion. Si no existe una mesa de esta eleccion con esa maquina, se inserta la relacion.

★ Toda persona que vota en una mesa, debe tener nulo el campo selloVoto del padron para dicha mesa. ie. No permitir que la gente vote mas de una vez por eleccion.

★ Para todo voto que se inserta se debe actualizar correctamente la fecha y hora en la que voto y poniendo el "sello" virtual en el padron asignando un valor no nulo a cuandoVoto.

★ Para todo voto que se inserta se debe actualizar correctamente la tabla hija de ser necesario. ie Voto y Voto A Candidato.

★ Validar que el voto puede hacerse en la fecha de la eleccion de 8am a 6pm.

✓ Con respecto a las restricciones referidas al voto y el sello en el padron, no podemos modelarlas con before y after triggers dado que no tenemos forma de navegar desde la tabla voto hacia el padron(no se guarda info de la persona en la tabla voto.). Con lo cual crearemos un stored procedure, que realizará la operacion de voto como una transacción, verificando que el sello sea NULO antes de insertar el voto y que el sello contenga el datetime una vez insertado el voto. Ademas de realizar la consistencia de insercion del voto si es a candidato en varias tablas. Las validaciones adicionales al voto mencionadas anteriormente, estan contempladas dentro de este procedimiento pues las tablas involucradas ya contienen los triggers necesarios.

★ Inserciones consistentes en tablas con jerarquia: ie. Entidades: Ciudadano, Eleccion, Voto y sus hijos.

✓ Dado que la PK de la tabla hijo es FK de la PK de la tabla padre, la insercion en tablas hijos fallara si no existe el registro correspondiente en la tabla padre. Analogamente, debemos proveer mecanismos para poder realizar inserciones consistentes de entidades padres que requieran insercion en tablas hijas. Para la tabla Voto, el punto anterior resuelve esto. Ciudadano no tiene herencia disjunta, con lo cual, no es necesario proveer este mecanismo. Se proveeran mecanismos (**Stored Procedure**) para crear elecciones de diferentes tipos.

3.1.4. Stored Procedures

Estos procedimientos permiten realizar operaciones que involucran mas de una operacion en la base de datos de manera transaccional.

- Voto a candidato(dniVotante, dniCandidato, idEleccion)
- Voto a consulta popular(dniVotante, idEleccion, respuestaPlesbicito)
- Crear Consulta Popular(Fecha, Descripcion)
- Crear Eleccion Municipal(Fecha, idMunicipio)
- Crear Eleccion Provincial(Fecha, idProvincia)
- Crear Eleccion Legislativa(Fecha, idProvincia)

3.2. Codigo de resolucion de consignas

3.2.1. Script de creacion de base de datos fisica

Por motivos de latex y caracteres, no se pueden incluir los scripts. Los scripts .sql se encuentran en la carpeta fuentes.

3.2.2. Consultas pedidas por el enunciado

1. Poder obtener los ganadores de las elecciones transcurridas en el último año. Para resolver este problema nos creamos una vista auxiliar que nos devuelve la lista de elcciones del ultimo año, con sus resultados, es decir, el ranking (Candidato, CantVotos) para cada eleccion del último año.

```
                CREATE VIEW [dbo].[Ranking_Elecciones_Cargo_Ultimo_Anio] AS
                --Vista auxiliar para ranking de las ultimas elecciones del
                    anio.
                SELECT
        elec.Fecha as FechaEleccion,
        elec.tipo as TipoEleccion,
        (ciu.Nombre + ciu.Apellido) as Candidato,
        COUNT(vc.idVoto) AS CantVotos
FROM         dbo.Voto AS v
INNER JOIN dbo.Voto_A_Candidato AS vc ON v.idVoto = vc.idVoto -- Quiero saber
    a que candidato fue ese voto
INNER JOIN dbo.Ciudadano AS ciu ON vc.DNI= ciu.DNI -- Join para saber el
    nombre del candidato(es un ciudadano)
INNER JOIN dbo.Mesa AS m ON m.idMesa = v.idMesa -- Join para saber la eleccion
     de la mesa y agrupar
INNER JOIN dbo.Eleccion AS elec ON elec.idEleccion = m.idEleccion -- Join para
     saber la fecha de la eleccion
WHERE   (v.idMesa IN
        (SELECT     idMesa
        FROM          dbo.Mesa AS m
        WHERE      (idEleccion IN
                (SELECT     idEleccion
                FROM          dbo.Eleccion AS e
                WHERE      (YEAR(Fecha) = YEAR(GETDATE()))))))
GROUP BY vc.DNI, ciu.Nombre, ciu.Apellido, elec.Fecha, elec.tipo
ORDER BY FechaEleccion ASC, CantVotos DESC
```

Luego nos quedamos con los que tienen mas votos de los agrupamientos por eleccion.

```
WITH WINNERS AS (
        SELECT FechaEleccion, Candidato, CantVotos,
         ROW_NUMBER() over (
                PARTITION BY FechaEleccion
                ORDER BY CantVotos DESC
```

```
            ) AS NumFila
        FROM Ranking_Elecciones_Cargo_Ultimo_Anio

)
SELECT FechaEleccion, Candidato, CantVotos FROM WINNERS WHERE NumFila <= 1--
    cantidad de items por grupo
```

2. Poder consultar las cinco personas que más tarde fueron a votar antes de terminar la votación por cada centro electoral en una elección. Usamos una tabla intermedia que obtiene los votos de la eleccion, particionado por centro electoral, a su vez, utilizamos la funcion `ROW_NUMBER` para numerar los resultados de cada grupo, ordenados por tiempo de votacion y quedarnos con los 5 mayores tiempos de votacion de cada centro.

```
                WITH TOPFIVE AS (
            SELECT cen.Nombre_Establecimiento, (ciu.Nombre + ciu.Apellido) as
                Votante, p.selloVoto as HoraVoto,
            ROW_NUMBER() over (
                PARTITION BY cen.idCentro
                ORDER BY p.selloVoto DESC
            ) AS NumFila
            FROM Padron p
            INNER JOIN Mesa m ON m.idMesa = p.idMesa
            INNER JOIN Centro cen ON cen.idCentro = m.idCentro
            INNER JOIN Ciudadano ciu ON ciu.DNI = p.DNI)

        SELECT Nombre_Establecimiento, Votante, HoraVoto FROM TOPFIVE WHERE
            NumFila <= 5--cantidad de items por grupo
```

3. Poder consultar quienes fueron los partidos políticos que obtuvieron más del 20 % en las últimas cinco elecciones provinciales a gobernador. **Nota:** En las elecciones a cargo provincial se elije gobernador(legislativas y municipales estan separadas de las provinciales).

   Creamos una vista que nos devuelve el ranking de las ultimas 5 elecciones a gobernador, y ademas la cantidad de votos por cada eleccion.

```
        SELECT elec.idEleccion, elec.Fecha AS FechaEleccion, ciu.Nombre + ciu.
            Apellido AS Candidato, partPolitico.Nombre as PartidoPolitico,
COUNT(vc.idVoto) AS CantVotos,

SUM(COUNT(vc.idVoto)) over (
                PARTITION BY elec.idEleccion
            ) AS Cant_Total_Votos_Por_Eleccion

FROM          dbo.Voto AS v
                        INNER JOIN dbo.Voto_A_Candidato AS vc ON v.idVoto = vc.
                            idVoto
                        INNER JOIN dbo.Ciudadano AS ciu ON vc.DNI = ciu.DNI
                        INNER JOIN dbo.Mesa AS m ON m.idMesa = v.idMesa
                        INNER JOIN dbo.Eleccion AS elec ON elec.idEleccion = m.
                            idEleccion
                        INNER JOIN dbo.Postulaciones post ON (post.idEleccion =
                            elec.idEleccion AND post.DNI = ciu.DNI)
                        INNER JOIN dbo.Partido_Politico partPolitico ON post.
                            idPartido = partPolitico.idPartido
WHERE      (v.idMesa IN
               (SELECT      idMesa
                 FROM           dbo.Mesa AS m
                 WHERE       (idEleccion IN
                              (SELECT      TOP (5) e.idEleccion
                                FROM           dbo.Eleccion AS e INNER JOIN
                                                        dbo.
                                                            Eleccion_Cargo_Provincial
                                                             AS ecp ON ecp.
                                                            idEleccion = e.
                                                            idEleccion
```

```
                                WHERE       (e.tipo = 'Cargo Provincial') AND (
                                    YEAR(e.Fecha) = YEAR(GETDATE()))
                                ORDER BY e.Fecha DESC))))
GROUP BY vc.DNI, partPolitico.Nombre, ciu.Nombre, ciu.Apellido, elec.Fecha,
    elec.idEleccion
```

Creamos otra vista que sobre la vista anterior, calcula el porcentaje de votos de cada partido y lo filtra por quienes tienen mas de 20 %.

```
SELECT     TOP (100) PERCENT FechaEleccion, PartidoPolitico, CantVotos * 100 /
     Cant_Total_Votos_Por_Eleccion AS Porcentaje
FROM         dbo.Ranking_Candidatos_Ultimas_5_Elecciones_A_Gobernador
WHERE     (CantVotos * 100 / Cant_Total_Votos_Por_Eleccion >= 20)
ORDER BY FechaEleccion, CantVotos DESC
```

Finalmente, consultamos la vista.

```
        SELECT * FROM Partidos_Con_Mas_Del_20_Porciento_Ultimas_5_Gobernador
```

4. Testing

4.1. Codigo de testing de la solucion provista

En primera instancia, se realizaran pruebas de solidez y correctitud de las restricciones implementadas. Por último, tests de correctitud de las consultas requeridas por el enunciado. El codigo de los tests de restricciones se encuentra en 'tests.sql'

4.1.1. Votos de una eleccion

Verificaremos la conjuncion de las restricciones "Todos los votos para una eleccion son: o bien consulta popular o bien de tipo candidato según corresponda el tipo de eleccion", "Si la eleccion es una consulta popular, los votos deben ser si/no. sino, deben ser candidatos" y por ultimo, "Un voto a candidato, debe referenciar a un candidato postulado para dicha eleccion"

1. **Consulta Popular:** Si la eleccion es una consulta popular, los votos cuya clave foranea referencia esa eleccion deben ser de tipo ' A Plesbiscito Si ' o ' A Plesbiscito No'. Creamos una elección de tipo 'Consulta Popular' utilizando la Stored Procedure 'Crear_ Consulta_ Popular' con los valores Descripcion: TEST y Fecha: '2015-06-09'.

   A continuacion, debemos crear una mesa. Para eso, debemos tener un presidente, un vicepresidente, un centro y una maquina(y para ella, un tecnico). Las creamos ejecutando:

```
INSERT INTO Ciudadano VALUES
 (1,'JUAN','VOTANTE'),
 (2,'PEPE','PRESIDENTE'),
 (3,'VICTOR','VICEPRESIDENTE'),
 (4,'TOMAS','TECNICO'),

 INSERT INTO Centro VALUES
 (1,'Escuela', '9 de julio 1');

 INSERT INTO Maquina VALUES
 (1,4);

 INSERT INTO Mesa VALUES
 (1,1,2,3,1,1,1);
```

   - 'A Plesbiscito Si': Insertar un voto 'A Plesbiscito Si' deberia realizarse correctamente. (En todos los casos, al terminar el test borramos el voto) Ejecutamos la Stored Procedure 'Votar_ Consulta_ Popular' con los parametros DNIdelVotante=1, idEleccion=1 y OpcionVoto=1. Acto seguido, ejecutamos:

```
        SELECT * FROM Voto
```

     Y efectivamente esto nos da como resultado:

	idVoto	idMesa	Tipo
1	1	1	A Plesbiscito Si

   - 'A Plesbiscito No': Insertamos un voto de tipo 'A Plesbiscito No', lo cual deberia ejecutarse correctamente, usando la Stored Procedure 'Votar_ Consulta_ Popular' con los parametros DNIdelVotante=1, idEleccion=1 y OpcionVoto=0. Resultado:

	idVoto	idMesa	Tipo
1	1	1	A Plesbiscito No

   - 'Voto de tipo invalido': Insertamos un voto de tipo invalido, lo cual NO deberia ejecutarse correctamente, usando la Stored Procedure 'Votar_ Consulta_ Popular' con los parametros DNIdelVotante=1, idEleccion=1 y OpcionVoto=3 (Esto representa un tipo de voto invalido). Resultado:

	idVoto	idMesa	Tipo

- 'Voto a un candidato': Insertamos un voto 'A Candidato' lo cual NO deberia ejecutarse correctamente, usando la Stored Procedure 'Votar_ Candidato' sobre la Consulta Popular, con los parametros DNIdelVotante=1, idEleccion=1 y DniCandidato=5. Inmediatamente recibimos el aviso: "La eleccion es para Consulta Popular pero el voto no es a plesbicito". y como resultado:

	idVoto	idMesa	Tipo

2. **Eleccion a candidato:** Si la eleccion es de tipo 'Cargo Federal', 'Cargo Municipal', Cargo Provincial' o tipo 'Cargo Legislativo', los votos cuya clave foranea referencia esa eleccion deben ser de tipo 'A Candidato', y deben referenciar a un candidato que este postulado a dicha eleccion. Vamos a crear una eleccion provincial, debemos crear una provincia, un partido, y postular a 'CARLOS CANDIDATO' a la eleccion.

```
 INSERT INTO Ciudadano VALUES
(5,'CARLOS', 'CANDIDATO');

INSERT INTO Candidato VALUES
(5);

INSERT INTO Provincia VALUES
(1,'BUENOS AIRES');

INSERT INTO Partido_Politico VALUES
(1,'FRENTE POR LA APROBACION');

INSERT INTO Postulaciones VALUES
(1,5,1)
```

Eliminamos la anterior eleccion, de forma de poder usar la misma mesa y sus encargados, y usamos la Stored Procedure 'Crear_ Eleccion_ Provincial', con los parametros idProvincia=1 y fecha='2015-06-10'.

- 'Candidato Valido': Insertamos un voto de tipo 'A candidato', cuya clave foranea a candidato referencia a un candidato cuyo DNI se encuentra en la tabla postulaciones para esa eleccion, lo cual deberia ejecutarse sin problemas. Usamos la Stored Procedure: 'Votar_ Candidato' con los parametros: DNIVotante=1, idEleccion=1, DNICandidato=5.Si ejecutamos:

```
SELECT * FROM Voto
```

Obtenemos:

	idVoto	idMesa	Tipo
1	1	1	A Candidato

Y si ejecutamos:

```
SELECT * FROM Voto_A_Candidato
```

Obtenemos:

idVoto	DNI
1	5

- 'Candidato Invalido': e insertamos un voto de tipo 'A candidato', cuya clave foranea a candidato NO referencia a un candidato cuyo DNI se encuentra en la tabla postulaciones para esa eleccion. Esto no deberia ser aceptable. Para esto creamos otro ciudadano:

```
 INSERT INTO Ciudadano VALUES
(6,'IGNACIO','IMPOSTOR');
```

Usamos la Stored Procedure: 'Votar_ Candidato' con los parametros: DNIVotante=1, idEleccion=1, DNICandidato=6. Obtenemos un aviso: 'Instrucción INSERT en conflicto con la restricción FOREIGN KEY "FK_Voto_ A_ Candidato_ Candidato'. Y los resultados de consultar la tabla Voto y la tabla Voto_ A_ Candidato son:

	idVoto	idMesa	Tipo

idVoto	DNI

- 'Voto a plesbiscito': Insertamos un voto de tipo 'A plesbiscito Si', lo cual no deberia ejecutarse. Ejecutamos la Stored Procedure 'Votar_ Consulta_ Popular' con los parametros DNIdelVotante=1, idEleccion=1 y OpcionVoto=1. Obtenemos el aviso: "La eleccion es para Cargo pero el voto no es a candidato", y como resultados:

	idVoto	idMesa	Tipo

idVoto	DNI

4.1.2. Multiples Candidatos:

Un partido politico puede presentar a un solo candidato por eleccion. Para chequear esto, vamos a intentar postular a ambos 'CARLOS CANDIDATO' e 'IGNACIO IMPOSTOR' al partido 'Frente por la Aprobacion'. No deberia ejecutarse. Al ejecutar esto, obtenemos el aviso: "Infracción de la restricción UNIQUE KEY 'Unique_ Key_ Eleccion_ Partido". Si consultamos la tabla postulaciones, nos devuelve:

idEleccion	DNI	idPartido
1	5	1

Con lo que efectivamente, no se inserto el segundo candidato.

4.1.3. Multiples Mesas para un ciudadano:

Un ciudadano solo vota en una mesa por eleccion. Para comprobar eso creamos otro presidente, vicepresidente, tecnico y maquina para poder crear otra mesa, e intentamos asignar en el Padron a 'PEPE VOTANTE' ambas mesas.

```
INSERT INTO Ciudadano VALUES
(7,'TITO','TECNICODOS'),
(8,'PEDRO','PRESIDENTEDOS'),
(9,'VALENTIN','VICEPRESIDENTEDOS');

INSERT INTO Maquina VALUES
(2,7);

 INSERT INTO Mesa VALUES
(2,2,8,9,1,1,2)
```

Y finalmente, asignamos al ciudadano a ambas mesas

```
Insert INTO Padron VALUES
(1,1,NULL)
(1,2,NULL)
```

A lo que recibimos el aviso: "Ya existe una mesa en la cual esta persona vota en esta eleccion". Y si consultamos el padron, obtenemos:

DNI	idMesa	selloVoto
1	1	NULL

Efectivamente solo se asigno la primera vez.

4.1.4. Multiples Mesas para una maquina:

Una maquina solo funciona en una mesa por eleccion. Para esto eliminamos la segunda mesa que creamos en el test anterior, y la vamos a crear de nuevo pero asignandole la misma maquina 1.

```
        DELETE FROM Mesa
        WHERE idMesa=2


INSERT INTO Mesa VALUES
         (2,2,8,9,1,1,1);
```

Obtenemos el aviso: "Ya existe una mesa en esta eleccion con esta maquina", y si consultamos la tabla Mesa vemos que efectivamente no se agrego la segunda:

idMesa	nroMesa	idPresidente	idVicePresidente	idEleccion	idCentro	idMaquina
1	1	2	3	1	1	1

4.1.5. Validaciones de los votos

: Validaremos las restricciones: "No permitir que la gente vote mas de una vez por eleccion", "El voto puede hacerse en la fecha de la eleccion de 8am a 6pm".

- 'Unico voto': Creamos un partido politico para el Candidato 'IGNACIO IMPOSTOR' y lo postulamos

```
        INSERT INTO Partido_Politico VALUES
        (2,'UNIDOS POR LA UNION');


        INSERT INTO Postulaciones VALUES
        (1,5,1);
        (1,6,2);
```

  Usamos dos veces consecutivas la Stored Procedure: 'Votar_ Candidato' con los parametros: DNIVotante=1, idEleccion=1, DNICandidato=5 en la primera vez, y en la segunda vez: DNIVotante=1, idEleccion=1, DNICandidato=6. Obtenemos el aviso: 'Esta persona ya voto en esta eleccion', y como resultado:

	idVoto	idMesa	Tipo
	1	1	aCandidato

idVoto	DNI
1	5

  Donde vemos que solo se inserto el primer voto.

- 'Hora valida': Creamos una elección de tipo 'Consulta Popular', y hacemos que un ciudadano intente votar en un horario invalido (cambiamos la hora de la computadora a las 19:00hs). Recibimos el aviso:Estamos fuera de horario de votacion, y no se modificaron las tablas de votos.

4.1.6. Consultas del enunciado

:

Insertamos los siguientes datos en la base para poder realizar los tests: 15 ciudadanos, que seran los votantes, dos presidentes, vicepresidentes y tecnicos de mesa, dos candidatos, dos conductores(a estos ultimos, no los hacemos votar, solo hacemos votar a los ciudadanos para que sea mas entendible), dos centros, dos camiones, una provincia y dos partidos. Este codigo se encuentra en el archivo 'testQueries.sql'

```
INSERT INTO Ciudadano VALUES
(1,'CIUDADANO','UNO'),
(2,'CIUDADANO','DOS'),
(3,'CIUDADANO','TRES'),
(4,'CIUDADANO','CUATRO'),
(5,'CIUDADANO','CINCO'),
(6,'CIUDADANO','SEIS'),
(7,'CIUDADANO','SIETE'),
(8,'CIUDADANO','OCHO'),
(9,'CIUDADANO','NUEVE'),
(10,'CIUDADANO','DIEZ'),
(11,'CIUDADANO','ONCE'),
(12,'CIUDADANO','DOCE'),
(13,'CIUDADANO','TRECE'),
(14,'CIUDADANO','CATORCE'),
(15,'CIUDADANO','QUINCE'),
(16,'PRESIDENTEMESA','UNO'),
(17,'PRESIDENTEMESA','DOS'),
(18,'VICEPRESIDENTEMESA','UNO'),
(19,'VICEPRESIDENTEMESA','DOS'),
(20,'TECNICO','UNO'),
(21,'TECNICO','DOS'),
(22,'CANDIDATO','UNO'),
(23,'CANDIDATO','DOS'),
(24,'CANDIDATO','TRES'),
(25,'CANDIDATO','CUATRO'),
(26,'CONDUCTOR','UNO'),
(27,'CONDUCTOR','DOS');

INSERT INTO Centro VALUES
(1,'ESCUELA','UNO'),
(2,'ESCUELA2','DOS');

INSERT INTO Camion VALUES
(1,26),
(2,27);

INSERT INTO Camion_Centro VALUES
(1,1),
(2,2);

INSERT INTO Maquina VALUES
(1,20),
(2,21);

INSERT INTO Candidato VALUES
(22), (23), (24), (25);

INSERT INTO Provincia VALUES
(1,'BUENOS AIRES');

INSERT INTO Partido_Politico VALUES
(1,'PARTIDO DE IZQUIERDA'),
(2,'PARTIDO DE DERECHA');
```

- **Ganadores de elecciones del ultimo año** A continuacion creamos dos elecciones, una federal y una provincial. Cambiamos el reloj de la computadora al 1 de enero de 2015 para crear la federal, para poder estar la correctitud del calculo del año corriente, y la provincial se efectua el 12/06/2015. Usamos la Stored Procedure para crear la provincial, y el siguiente codigo para crear la federal:

```
INSERT INTO Eleccion VALUES
(1,GETDATE(),'Cargo Federal');
```

A continuacion creamos dos mesas, una en cada centro y para cada eleccion. Acto seguido, postulamos a los primeros dos candidatos a la eleccion del 1ero de enero, uno por el partido de izquierda, el otro por el de derecha. Hacemos lo mismo con el tercer y cuarto candidato, en la eleccion del 12 de junio.

```
INSERT INTO Mesa VALUES
(1,1,16,18,1,1,1);

INSERT INTO Mesa VALUES
(2,2,17,19,2,2,2);

INSERT INTO Postulaciones VALUES
(1,22,1),
(1,23,2),
(2,24,1),
(2,25,2);
```

Finalmente, armamos el padron y asignamos a los 15 ciudadanos, la mesa 1 para la primera eleccion y la mesa 2 para la segunda eleccion.

```
INSERT INTO Padron VALUES
(1,1,NULL),
(2,1,NULL),
(3,1,NULL),
(4,1,NULL),
(5,1,NULL),
(6,1,NULL),
(7,1,NULL),
(8,1,NULL),
(9,1,NULL),
(10,1,NULL),
(11,1,NULL),
(12,1,NULL),
(13,1,NULL),
(14,1,NULL),
(15,1,NULL),
(1,2,NULL),
(2,2,NULL),
(3,2,NULL),
(4,2,NULL),
(5,2,NULL),
(6,2,NULL),
(7,2,NULL),
(8,2,NULL),
(9,2,NULL),
(10,2,NULL),
(11,2,NULL),
(12,2,NULL),
(13,2,NULL),
(14,2,NULL),
(15,2,NULL);
```

Finalmente, hacemos votar a los ciudadanos. En la primera eleccion, haremos votar 10 al 'CANDIDATO UNO' y los restantes 5 al 'CANDIDATO DOS'. En la segunda eleccion, haremos votar 8 al 'CANDIDATO TRES' y 7 al 'CANDIDATO CUATRO. Hacemos uso de las debidas Stored Procedures. Si consultamos la tabla 'Voto_ A _ Candidato':

idVoto	DNI
1	22
2	22
3	22
4	22
5	22
6	22
7	22
8	22
9	22
10	22
11	23
12	23
13	23
14	23
15	23
16	24
17	24
18	24
19	24
20	24
21	24
22	24
23	24
24	25
25	25
26	25
27	25
28	25
29	25
30	25

Donde podemos ver que se refleja lo que queriamos. A continuacion ejecutamos la query, y esperamos que nos de como resultado 'CANDIDATO UNO' y 'CANDIDATO TRES'.

- **Ultimas personas que fueron a votar por Centro**

  Por la forma en las que ejecutamos las Stored Procedures(votaron del 1 al 15 en orden en la primera, y luego del 1 al 15 en orden en la segunda), la query deberia devolvernos los ciudadanos desde el 11 al 15 para cada eleccion. Ejecutamos el codigo de la consulta:

```
\begin{lstlisting}
                WITH TOPFIVE AS (
            SELECT cen.Nombre_Establecimiento, (ciu.Nombre + ciu.Apellido) as
                Votante, p.selloVoto as HoraVoto,
            ROW_NUMBER() over (
                PARTITION BY cen.idCentro
                ORDER BY p.selloVoto DESC
            ) AS NumFila
            FROM Padron p
            INNER JOIN Mesa m ON m.idMesa = p.idMesa
            INNER JOIN Centro cen ON cen.idCentro = m.idCentro
            INNER JOIN Ciudadano ciu ON ciu.DNI = p.DNI

        SELECT Nombre_Establecimiento, Votante, HoraVoto FROM TOPFIVE WHERE
            NumFila <= 5--cantidad de items por grupo
```

  Y la consulta nos da como resultado:

Nombre_ Establecimiento	Votante	HoraVoto
ESCUELA	CIUDADANOQUINCE	2015-01-01 15:31:45.777
ESCUELA	CIUDADANOCATORCE	2015-01-01 15:31:39.827
ESCUELA	CIUDADANOTRECE	2015-01-01 15:31:33.407
ESCUELA	CIUDADANODOCE	2015-01-01 15:31:19.793
ESCUELA	CIUDADANOONCE	2015-01-01 15:30:54.793
ESCUELA2	CIUDADANOQUINCE	2015-06-12 15:34:20.013
ESCUELA2	CIUDADANOCATORCE	2015-06-12 15:34:15.037
ESCUELA2	CIUDADANOTRECE	2015-06-12 15:34:11.063
ESCUELA2	CIUDADANODOCE	2015-06-12 15:34:05.987
ESCUELA2	CIUDADANOONCE	2015-06-12 15:34:02.407

5. Conclusiones