DEPARTAMENTO
DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico 1

9 de junio 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

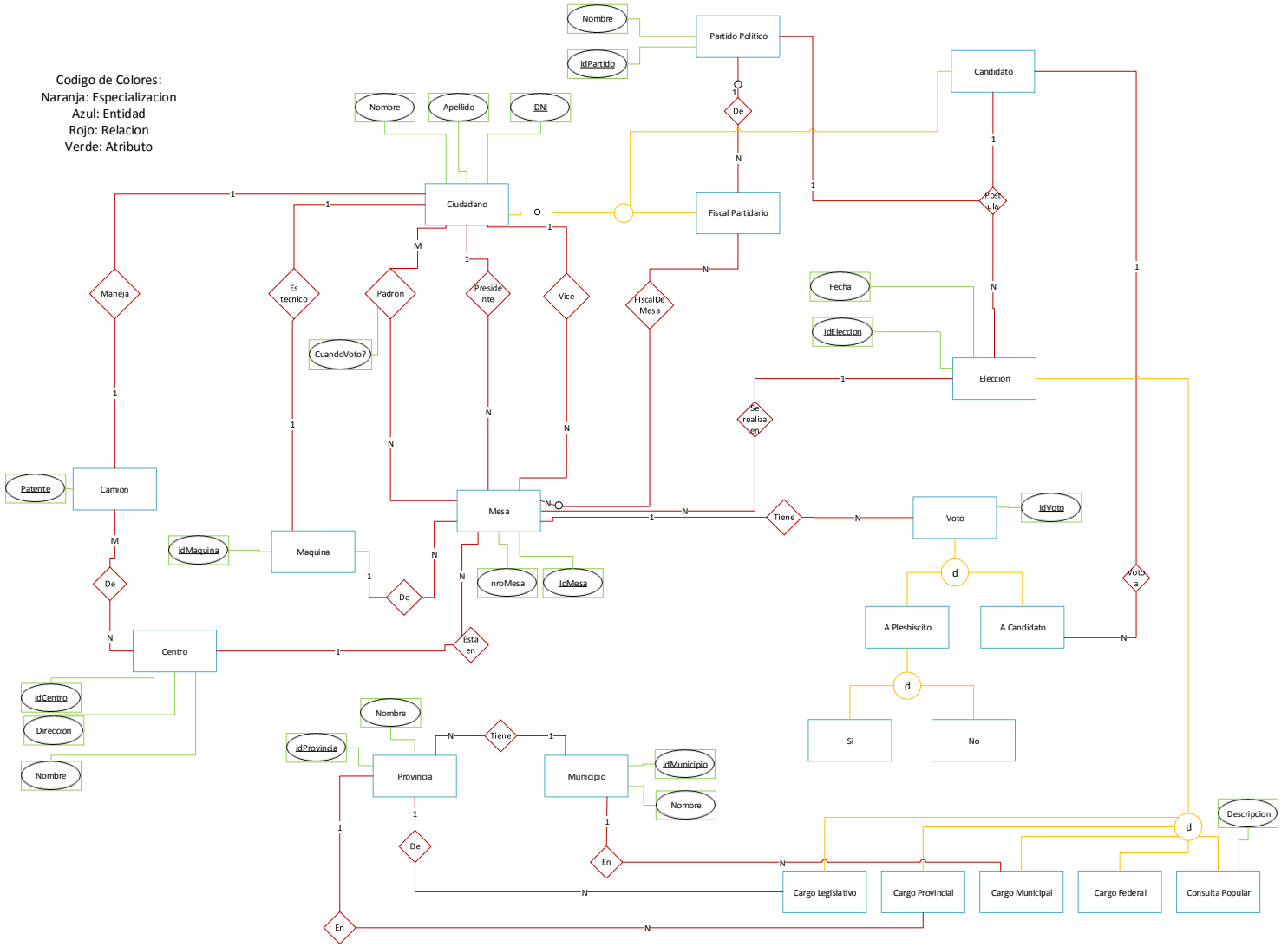

Figura 1: Diagrama Entidad Relacion.

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

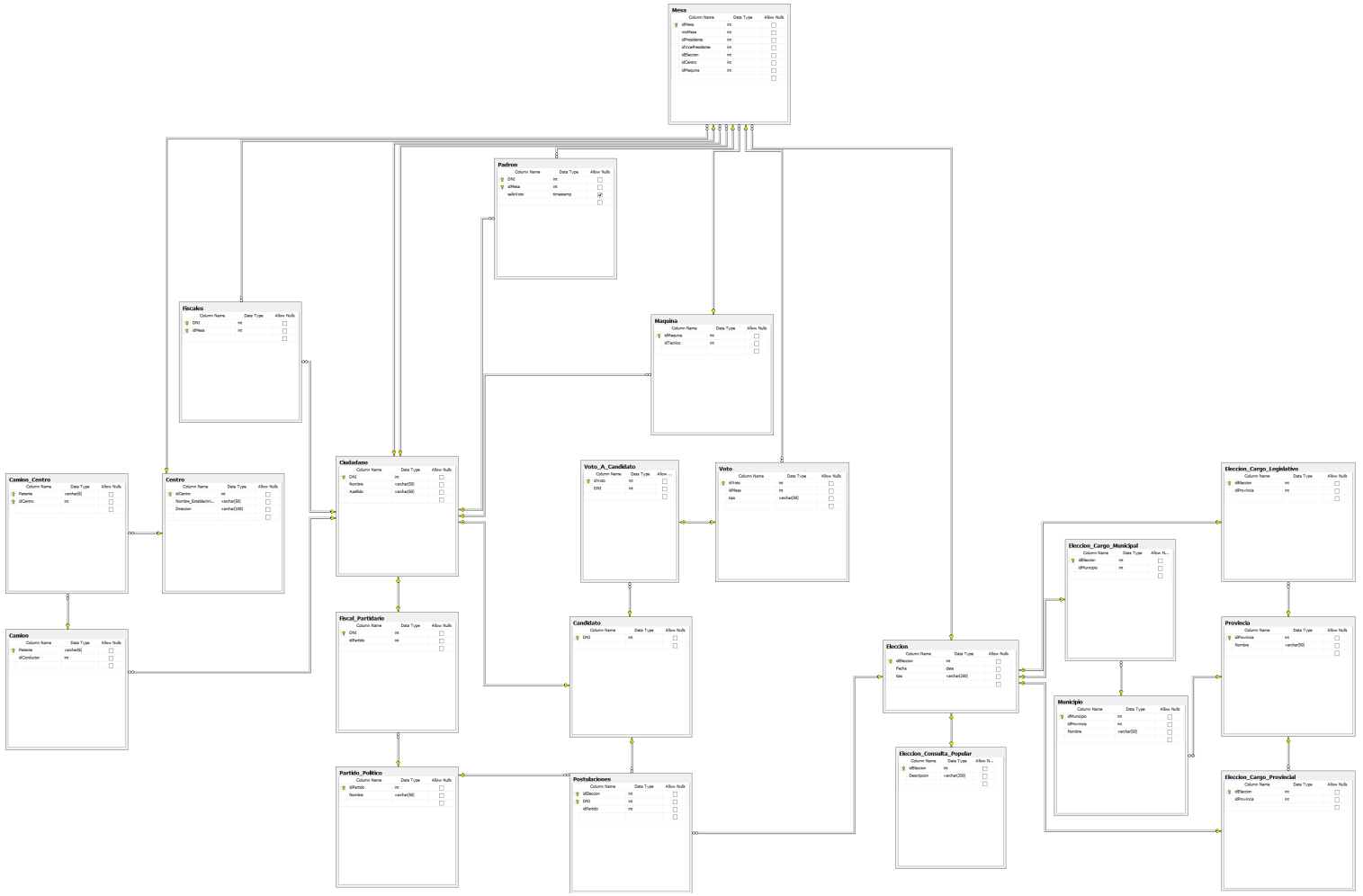

Figura 2: Diagrama fisico.

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, idMesa)
- Voto a consulta popular(dniVotante, idMesa, 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 aplicamos una consulta sobre esta vista, que nos devuelve para cada eleccion de la vista, el (o los en caso de empate) ganador(es). Creamos otra vista para acceder directamente a esta consulta pedida.

```
CREATE VIEW [dbo].[
    Ganadores_Elecciones_Cargo_Ultimo_Anio] AS
SELECT FechaEleccion, Candidato, MAX(CantVotos) as
    ganador_con_max_votos
FROM Ranking_Elecciones_Cargo_Ultimo_Anio
GROUP BY FechaEleccion, Candidato
```

Finalmente, se accede a los datos pedidos ejecutando:

```
SELECT * FROM dbo.[
    Ganadores_Elecciones_Cargo_Ultimo_Anio]
```

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
    WHERE p.idMesa IN (SELECT m.idMesa FROM Mesa m WHERE m.idEleccion
        = 2)--<eleccionTarget>
        )
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.Fecha AS FechaEleccion, ciu.Nombre + ciu.Apellido AS
            Candidato, partPolitico.Nombre as PartidoPolitico,
COUNT(vc.idVoto) AS CantVotos,

COUNT(*) over (
            PARTITION BY elec.fecha
        ) 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
```

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

Testear triggers para validaciones –Utiles para testear triggers rapido –ALTER TABLE Voto NOCHECK CONSTRAINT ALL –ALTER TABLE Voto CHECK CONSTRAINT ALL

5. Conclusiones