FACULTAD DE CIENCIAS EXACTAS
UNIVERSIDAD NACIONAL DEL CENTRO DE LA PROVINCIA DE BUENOS AIRES

Tecnicatura Universitaria en Desarrollo de Aplicaciones Informáticas (TUDAI)

Base de Datos

Tema 7: Vistas

2025

Vistas: Concepto

Es una **tabla virtual y derivada**. Se define dándole un nombre a una consulta SQL. Los registros se generan al operar sobre ella.

EMPRESA_PRODUCTORA de unc_esq_peliculas

codigo_productora	nombre_productora	id_ciudad
381451	Productora Pantanetti	2246
803693	Productora Zoppi	2246
565978	Productora Reynoso	28229
458304	Productora Leale	28229
787883	Productora Ardisana	34806
121411	Productora Polimandi	34806
354343	Productora Picapietra	46402
63284	Productora Resa	46402

SQL de la vista: V1

```
SELECT codigo_productora, nombre_productora
FROM empresa_productora
WHERE id_ciudad IN (34806,28229);
```

Vistas: Concepto

El contenido de una vista está definido como una consulta sobre **una tabla, sobre varias tablas, sobre otras vistas o sobre otra/s vista/s o tablas**

CIUDAD

id_ciudad	nombre_ciudad	id_pais
2246	Tiourkuy	BR
28229	Industrial Park	JM
34806	Bharolian Kalan	ZM
46402	Kute	HT

codigo_productora	nombre_productora	id_ciudad
381451	Productora Pantanetti	2246
803693	Productora Zoppi	2246
565978	Productora Reynoso	28229
458304	Productora Leale	28229
787883	Productora Ardisana	34806
121411	Productora Polimandi	34806
354343	Productora Picapietra	46402
63284	Productora Resa	46402

SQL de la vista: V1

```
SELECT codigo_productora, nombre_productora, id_ciudad
FROM empresa_productora
WHERE id_ciudad IN (34806,28229);
```

SQL de la vista: V2

```
SELECT *
FROM ciudad
WHERE id_ciudad IN (34806, 28229);
```

SQL de la vista: V3

```
SELECT *
FROM V1 NATURAL JOIN ciudad;
```

SQL de la vista: V4

```
SELECT *
FROM V1 NTURAL JOIN V2;
```

Vistas: Creación

CREATE VIEW <nombre> [(n_col_1, …, n_col_n)]
AS <expresión_consulta>
[**WITH** [opción] **CHECK OPTION**];

n_col_1,..., $n\_col_{n)}$: nombres de columnas de la vista

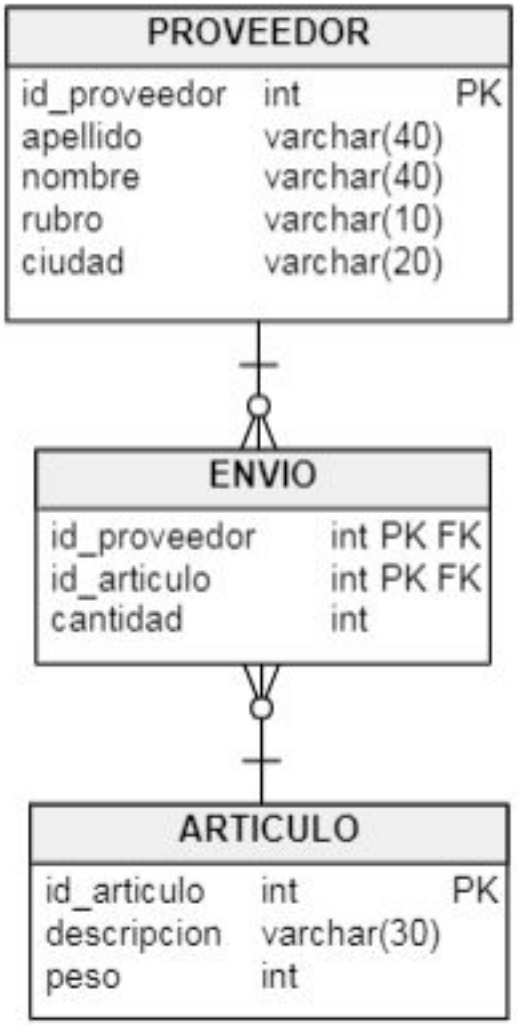
Los nombres de las columnas de la vista son iguales a los de las columnas de las tablas de la sentencia SELECT

Las columnas de la vista se pueden (re)nombrar colocando la lista ***completa*** de nombres entre paréntesis.

Hay que especificar con diferente nombre aquellas columnas provenientes de distintas tablas pero con igual nombre (ambigüas)

expresión_consulta: consulta sql que define los datos que integraran la vista

Vistas: ejemplos



Ejemplos

Vistas a partir de una tabla

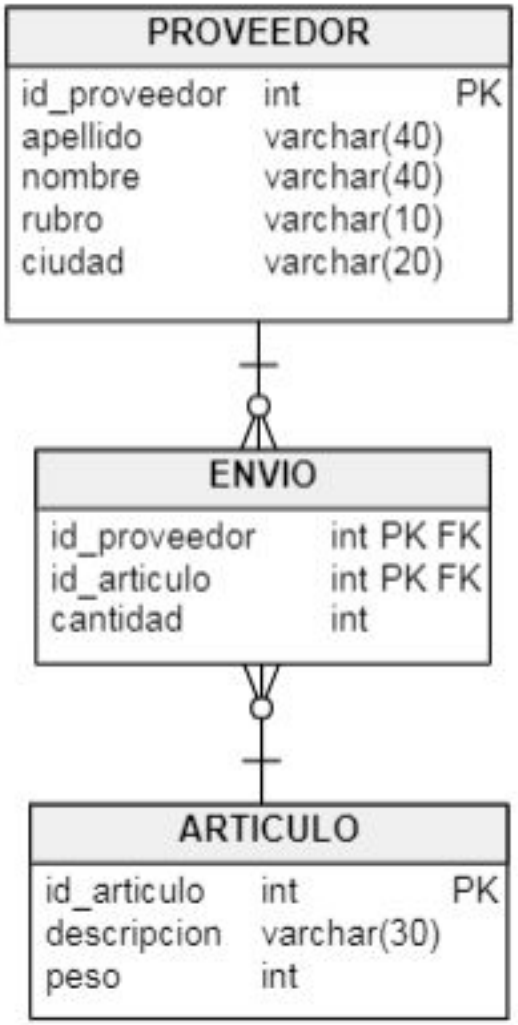
PROV_TANDIL que contenga el identificador y nombre de los proveedores de Tandil

```
CREATE VIEW PROV_TANDIL
AS SELECT id_provedor, nombre
      FROM PROVEEDOR
     WHERE ciudad = 'Tandil';
```

TOTAL_ARTICULO que contenga el identificador del articulo y la cantidad total de cada articulo enviado

```
CREATE VIEW TOTAL_ARTICULO
AS SELECT id_articulo, sum(cantidad)
      FROM ENVIO
      GROUP BY id_articulo;
```

Vistas: ejemplos



Ejemplos

Crear una Vista *PR_COMP* que contenga el identificador, nombre y ciudad de los proveedores de rubro computadoras

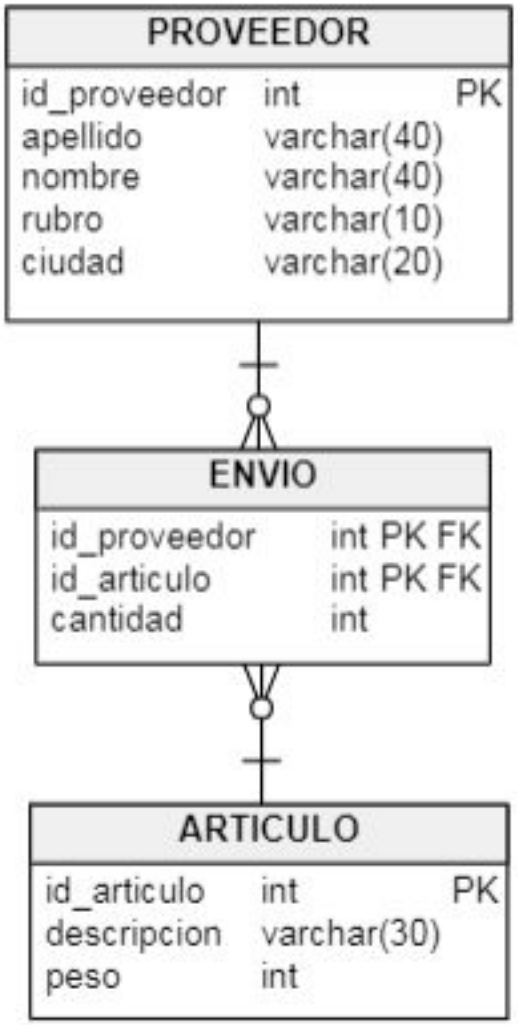
```
CREATE VIEW PR_COMP
AS SELECT id_provedor, nombre, ciudad
      FROM PROVEEDOR
     WHERE rubro = 'Computadoras';
```

Proveedores de computadoras de Tandil (usar vista previa)

```
CREATE VIEW PR_COMP_TANDIL
AS SELECT id_provedor, nombre
     FROM PR_COMP
     WHERE ciudad = 'Tandil';
```

Vista a partir de otra vista

Vistas: ejemplos



Ejemplos

Crear una *Vista PROV_ENVIOS_TANDIL* que contenga el identificador y nombre de los proveedores y el identificador de artículos enviados por cada uno

```
CREATE VIEW PROV_ENVIOS_TANDIL
AS SELECT P.id_provedor, P.nombre,
          E.id_articulo
     FROM PROVEEDOR P, ENVIO E
     WHERE P.id_proveedor = E.id_proveedor AND
           P.ciudad = 'Tandil';
```

Vista a partir de más de una tabla

Vistas: Consultas y Eliminación

Una vista puede ser consultada como cualquier tabla

Ejemplos

Listar alfabéticamente los proveedores de *PR_COMP_TANDIL*

```
SELECT nombre
FROM PR_COMP_TANDIL
ORDER BY nombre;
```

```
CREATE VIEW PR_COMP_TANDIL
AS SELECT id_proveedor, nombre
   FROM PR_COMP
   WHERE ciudad = 'Tandil';
```

Para eliminar una vista del esquema de la BD:

DROP VIEW <nom_vista> [<opción>];

Opción

RESTRICT → se rechaza si hay objetos del esquema que hacen referencia a esta vista (*opción por defecto*)

CASCADE → siempre tiene éxito y se eliminan también las vistas dependientes

Pausa

Vistas:Actualizaciones

Cuando se **actualizan los registros de una tabla** los cambios se reflejan automáticamente sobre la/s vista/s definida/s a partir de ella, dado que la vista está definida como una consulta sobre las tablas base o otra vista

codigo_productora	nombre_productora	id_ciudad
381451	Productora Pantanetti	2246
803693	Productora Zoppi	22
565978	Productora Reynoso	282
458304	Productora Leale	282
787883	Productora Ardisana	348
121411	Productora Polimandi	348
354343	Productora Picapietra	464
63284	Productora Resa	464

codigo_productora	nombre_productora	id_ciudad
381451	Productora Pantanetti	2246
803693	Productora Zoppi	2246
565978	Productora Reynoso	28229
458304	**Productora de los Leale**	28229
787883	Productora Ardisana	34806
121411	Productora Polimandi	34806
354343	Productora Picapietra	46402
63284	**Productora de Don Resa**	46402

UPDATE empresa_productora SET nombre_productora = …..;

Vistas:Actualizaciones

Las vistas que hemos visto hasta ahora NO mantienen copias de los datos

Si hay actualizaciones en los datos de las tablas que está utilizando alguna vista, el SGBD debe "actualizar" las vistas (los registros se generan al consultar la vista)

Sí existe un tipo de vista que mantienen copias de los datos, se las denomina VISTAS MATERIALIZADAS

El SGBD debe mantener automáticamente actualizados los datos de las vistas materializadas.

Vistas: Actualizaciones

Qué **DEBERÍA** pasar si la actualización se realiza sobre la vista y no sobre la/s tablas/s?

codigo_productora	nombre_productora	id_ciudad
381451	Productora Pantanetti	2246
903693	Productora Zoppi	2246
565978	Productora Reynoso	28229
458304	Productora Leale	28229
787883	Productora Ardisana	34806
121411	Productora Polimandi	34806
354343	Productora Picapietra	46402
63284	Productora Resa	46402

SQL de la vista: V1

```
SELECT codigo_productora, nombre_productora
FROM empresa_productora
WHERE id_ciudad IN (34806,8229);
```

UPDATE V1
SET nombre_productora = …..
WHERE codigo_productora = 121411;

Vistas: Actualizaciones

Puede siempre determinar el SGBD certeramente que registros actualizar?

EMPRESA_PRODUTORA		CIUDAD		
codigo_productora	nombre_productora	id_ciudad	nombre_ciudad	id_pais
381451	Productora Pantanetti	2246	Tiourkuy	BR
803693	Productora Zoppi	2246	Tiourkuy	BR
565978	Productora Reynoso	28229	Industrial Park	JM
458304	Productora Leale	28229	Industrial Park	JM
787883	Productora Ardisana	34806	Bharolian Kalan	ZM
121411	Productora Polimandi	34806	Bharolian Kalan	ZM
354343	Productora Picapietra	46402	Kute	HT
63284	Productora Resa	46402	Kute	HT

SQL de la vista: V1

```
SELECT *
FROM empresa_productora NATURAL JOIN
ciudad
WHERE id_ciudad IN (34806,28229);
```

UPDATE V1
SET nombre_ciudad = …..
WHERE id_pais = 'JM';

DELETE V1
WHERE id_pais = 'JM' ?????

Vistas: Actualizaciones

Las operaciones de **actualización sobre una vista DEBERÍA propagarse automáticamente** a operaciones sobre las tablas base

Vistas: Actualizaciones

En algunos casos puede hacerse, en otros, la actualización de vistas puede generar ambigüedades

Suprimir una tupla en una vista => borrarla de la tabla base ?
(o modificar la tupla existente para que "desaparezca" de la vista?)

Insertar una fila en una vista => insertar una tupla en la tabla base? (o actualizar alguna tupla existente para que ahora pueda ser seleccionada en la vista?)

Cómo se propaga una actualización sobre un **campo derivado**?
sobre el valor resultante de una **función de agregación**?
sobre una consulta a una tabla que **no conserva la clave**?

Vistas Actualizables

Vistas Definidas sobre una Tabla

De acuerdo a lo que plantea SQL, una vista definida **sobre una (1) sola tabla base** es actualizable si:

1- conserva **todas** las columnas de la clave primaria (que forme parte del SELECT)

2- NO contiene **funciones de agregación** o **información derivada**

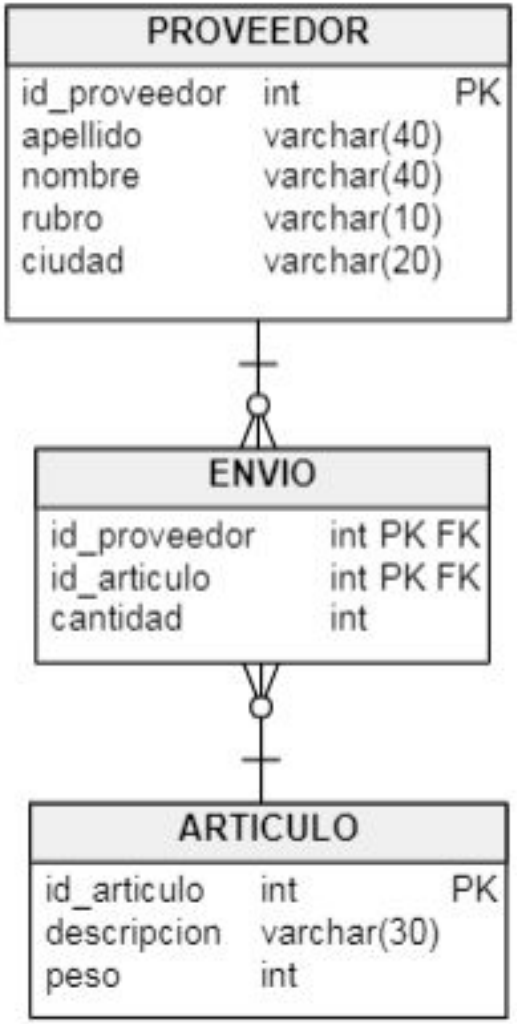
3- NO incluye la claúsula **DISTINCT**

4- NO incluye subconsultas **EN EL SELECT**

Ejemplos

Vistas Actualizables

Vistas Definidas sobre una Tabla



Cuales de las siguientes vistas son actualizables?

PROV_TANDIL1 con id y nombre de los proveedores de Tandil

```
CREATE VIEW PROV_TANDIL1
AS SELECT id_provedor, nombre
FROM PROVEEDOR WHERE ciudad = 'Tandil';
```

PROV_COMP con nombre y ciudad de proveedores de computadoras

```
CREATE VIEW PROV_COMP
AS SELECT nombre, ciudad
FROM PROVEEDOR WHERE rubro = 'Computadoras';
```

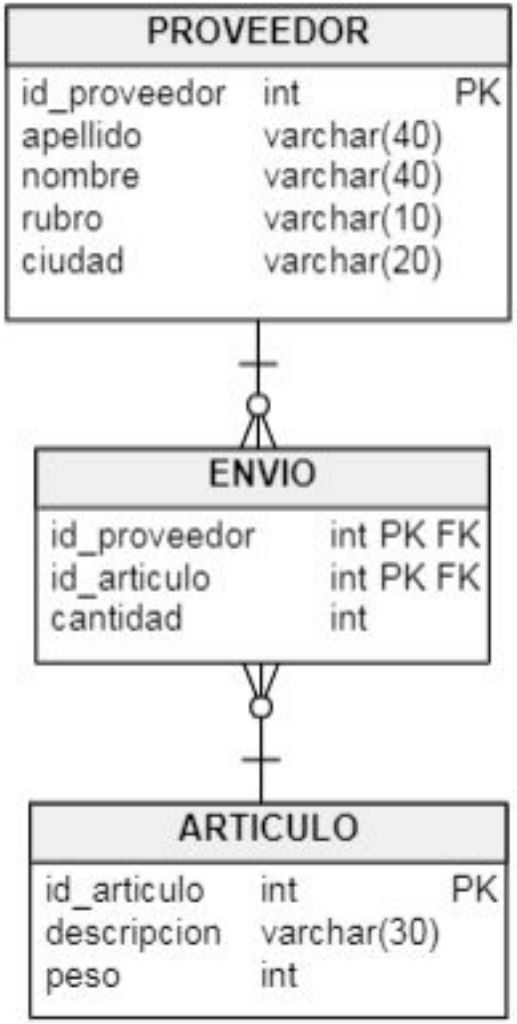
TOTAL_ARTICULO con la cantidad total enviada de cada artículo

```
CREATE VIEW TOTAL_ARTICULO
AS SELECT id_articulo, sum(cantidad)
      FROM  ENVIO
      GROUP BY id_articulo;
```

Ejemplos

Vistas Actualizables

Vistas Definidas sobre una Tabla



Las siguientes vistas son actualizables?

PROV_TANDIL1 con id y nombre de los proveedores de Tandil

```
CREATE VIEW PROV_TANDIL1
AS SELECT id_provedor, nombre
FROM PROVEEDOR WHERE ciudad = 'Tandil';
```

√

PROV_COMP con nombre y ciudad de proveedores de computadoras

```
CREATE VIEW PROV_COMP
AS SELECT nombre, ciudad
FROM PROVEEDOR WHERE rubro = 'Computadoras';
```

X

TOTAL_ARTICULO con la cantidad total enviada de cada artículo

```
CREATE VIEW TOTAL_ARTICULO
AS SELECT id_articulo, sum(cantidad)
      FROM  ENVIO
      GROUP BY id_articulo;
```

X

Vistas Actualizables
Vistas Definidas sobre una Tabla

El SGBD traduce una actualización de una vista definida a partir de una tabla base en una operación de actualización sobre la misma:

- la actualización de un registro en una vista debe propagarse en una única actualización **(del mismo tipo)** en la tabla subyacente

- esto se logra si y sólo si la vista conserva la/s columna/s que conforman la clave de la tabla subyacente

Vistas Actualizables

La operación de insert, update o delete sobre una vista procederá siempre que no se viole ninguna restricción de integridad definida sobre la tabla base

Lo anterior NO significa que la vista no sea actualizable

Vistas Actualizables
Vistas Definidas sobre más de una Tabla

- En una **vista definida sobre más de una tabla**:
  - La actualización sólo puede modificar **UNA** de las tablas o vistas subyacentes: la que cumpla la propiedad de **preservación de la clave (vista *Key preserved*)**, es decir la que tiene la misma clave de la vista
  - NO debe estar definida en base a operaciones de Unión, Intersección o Diferencia *(algunas versiones posteriores del estándar permiten algunas operaciones, por ej. Intersección)*

→ Se denominan **vistas de ensamble**, o en general Selección-Proyección-Ensamble**.** Se obtienen mediante condiciones de ensamble sobre los pares FK , PK especificados en las RIRs

Vistas

Propiedad de **preservación de la clave**

vistas 'key-preserved'

- Establece la condición básica que hace que una vista sea actualizable: cada actualización de una tupla en una vista debe propagarse a una única actualización (de igual tipo) en una tabla base
- Se satisface si una fila dada en una tabla aparece como máximo una vez en la vista; esto es:
  - la clave de la vista es la clave de la tabla o vista de la cual procede (en vistas de una única tabla/vista), o
  - la clave de la vista es la de alguna de las tablas o vistas de las cuales procede (vista de ensamble).
  - En una jerarquía de vistas, si una vista conserva la clave de una vista que no es *key preserved*, entonces no será *key preserved*.

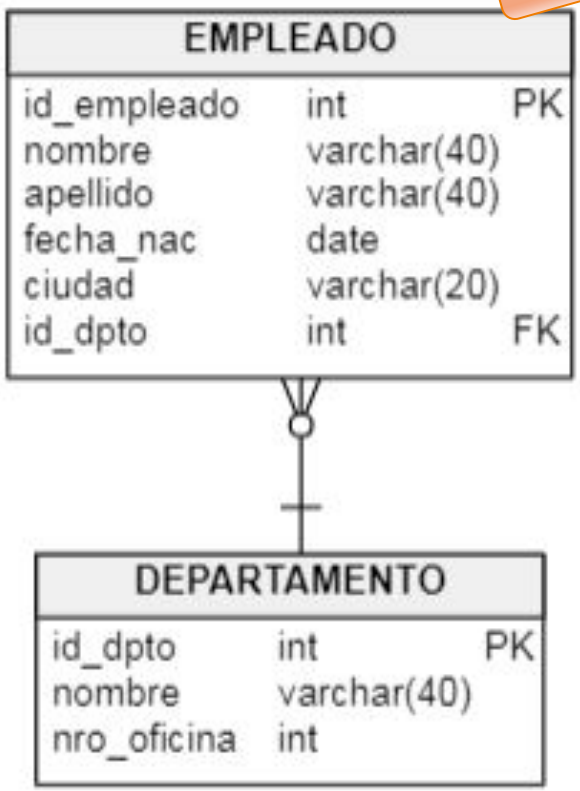
Vistas: Preservación de la Clave

- Este concepto ha sido explicado por diversos autores (*ej: Date*), aplicado por el estándar SQL como forma de operar, y 'popularizado' por Oracle que lo ha implementado y denominado 'propiedad key-preserved'.
- Es una propiedad necesaria para que cualquier sentencia SQL haga **actualizaciones correctas y verificables**
- **NOTA**: La propiedad de preservación de la clave en una vista NO depende de los datos actuales en las tablas de la base de datos, sino que es una propiedad estructural de su esquema.

Vistas Actualizables

Vistas Definidas sobre más de una Tabla

Ejemplos

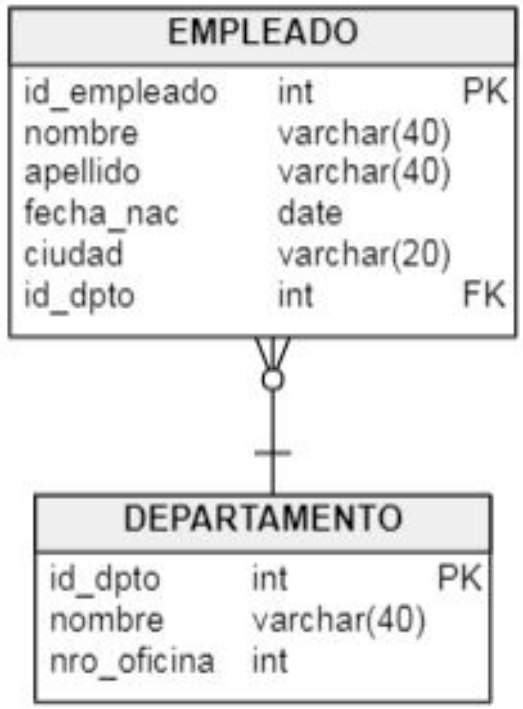


Vista ensamble mediante una RIR – ubicación 2 (proveniente de relación N:1) preserva la clave de la tabla del lado N

```
CREATE VIEW EMPL_SISTEMAS  AS
SELECT E.id_empleado, nombre, apellido
    FROM EMPLEADO E JOIN
         DEPARTAMENTO D
         ON (E.id_dpto = D.id_dpto)
    WHERE D.nombre = 'Tandil' ;
```

La clave de EMPL_SISTEMAS es E.id_empleado

Vistas Actualizables
Vistas Definidas sobre más de una Tabla



Ejemplos

Vista ensamble mediante una RIR – ubicación 2 (proveniente de relación N:1) preserva la clave de la tabla del lado N

```
CREATE VIEW EMPL_SISTEMAS  AS
SELECT E.id_empleado, nombre, apellido
FROM    EMPLEADO  E
WHERE id_dpto IN
      (SELECT id_dpto
        FROM DEPARTAMENTO
        WHERE nombre = 'Tandil') ;
```

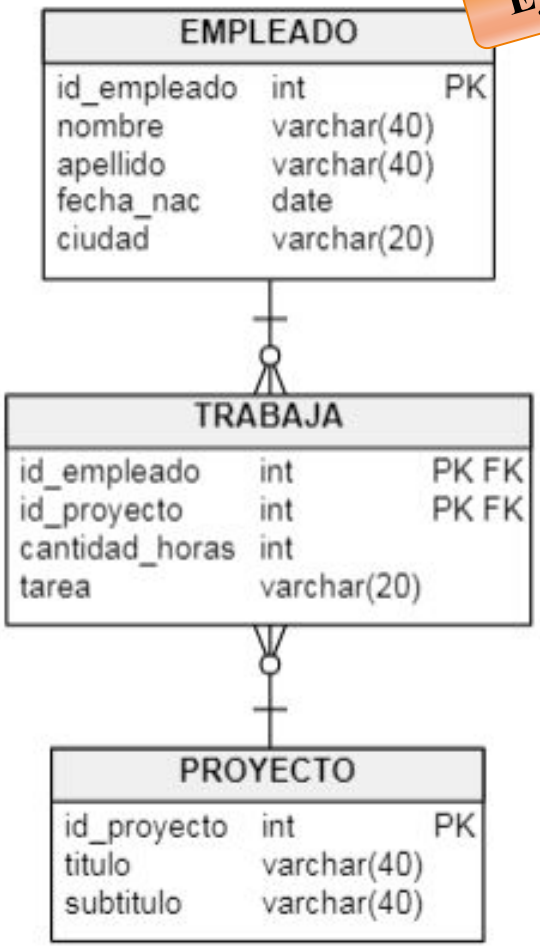
PostgreSQL impone que para que una vista sea actualizable en el FROM debe haber *una sola tabla*. La clave de EMPL_SISTEMAS es E.id_empleado

Vistas Actualizables

Vistas Definidas sobre más de una Tabla

ansi sql

Ejemplos

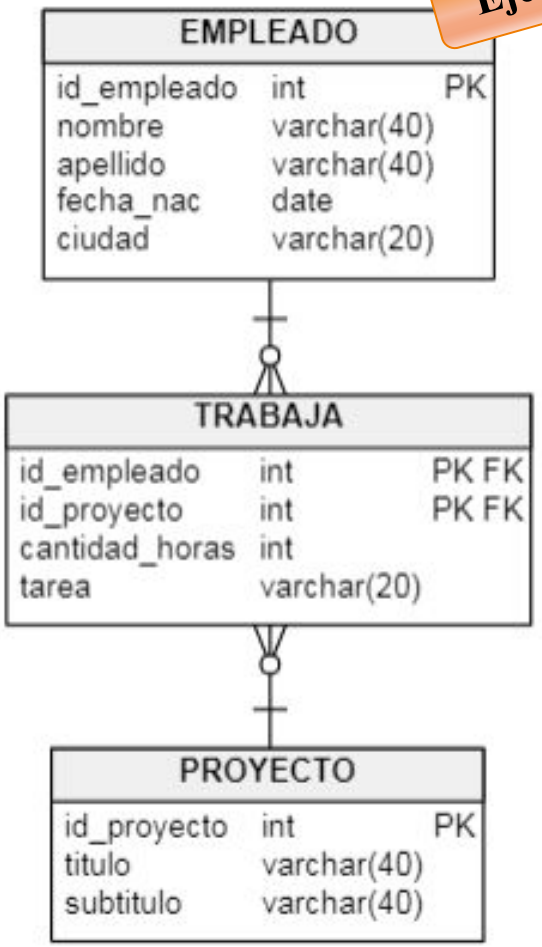


Vista ensamble mediante una RIR – relaciones N:N

```
CREATE VIEW EMPL_PROY  AS
  SELECT id_empleado, id_proyecto,
         cantidad_horas, tarea
  FROM TRABAJA T JOIN EMPLEADO E ON
                (T.id_empleado = E.id_empleado)
  WHERE E.ciudad = 'TANDIL');
```

La clave de EMPL_PROY es **id_empleado, id_proyecto**

Vistas Actualizables
Vistas Definidas sobre más de una Tabla



Vista ensamble mediante una RIR – relaciones N:N

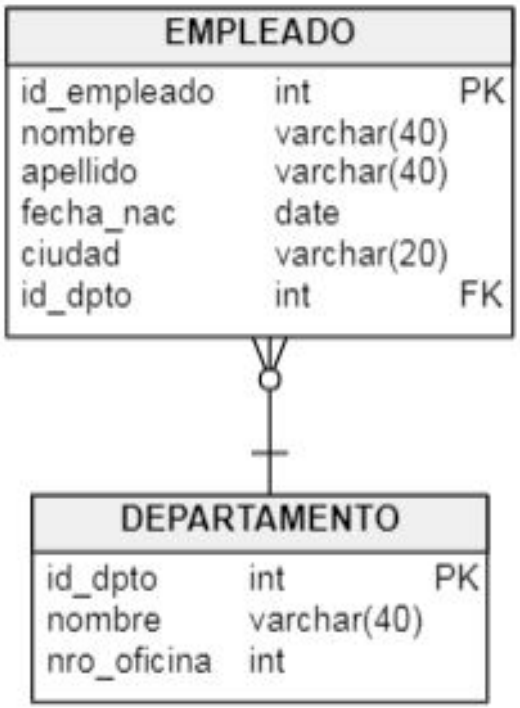
```
CREATE VIEW EMPL_PROY  AS
  SELECT id_empleado, id_proyecto,
         cantidad_horas, tarea
  FROM TRABAJA T
  WHERE id_empleado IN
              (SELECT id_empleado
                FROM EMPLEADO E
                WHERE E.ciudad = 'TANDIL');
```

La clave de EMPL_PROY es **id_empleado, id_proyecto**

Pausa

Vistas Actualizables

MIGRACIÓN DE TUPLAS DE LA VISTA



Ejemplos

Al actualizar tuplas en una vista, podría ocurrir que éstas dejen de pertenecer a la vista

```
CREATE VIEW EMPL_TANDIL AS
        SELECT * FROM EMPLEADO
        WHERE ciudad = 'Tandil';
```

Si se hace:

```
UPDATE EMPL_TANDIL SET ciudad= 'Rauch';
```

- Todos los registros de la vista (propagados a la tabla base) serían actualizados con un valor diferente de ciudad y entonces dejarían de pertenecer a la vista
- El efecto puede propagar errores inadvertidos por el usuario

Vistas Actualizables

SOLUCIÓN

CLÁUSULA WITH CHECK OPTION

Incluir la cláusula **WITH CHECK OPTION (WCO):**

```
CREATE VIEW EMPL_TANDIL AS
   SELECT * FROM EMPLEADO
   WHERE ciudad = 'Tandil'
    WITH [opción] CHECK OPTION;
```

Es opcional (y sólo se aplica para vistas actualizables)

- **CASCADED**:chequea la integridad sobre la vista y cualquiera dependiente de ella (*x defecto*)
- **LOCAL**: chequea la integridad sólo sobre la vista *(se ha propuesto eliminar esta opción)*

Si se especifica WCO: la condición del WHERE debe evaluar verdadero para que la tupla sea insertada/modificada
Se rechaza cualquier inserción o actualización que haga *migrar* una tupla de la vista (*porque la tupla ya no satisfaría la condición del query que define la vista*)

Vistas Actualizables

CLÁUSULA WITH CHECK OPTION

Ejemplos

Considerar la vista:

```
CREATE VIEW Envios500 AS
SELECT * FROM ENVIO
WHERE cantidad>=500;
```

qúe pasará si la vista tiene definido WCO? … y si no lo tiene?

INSERT INTO Envios500 **VALUES** (P1, A1, 500);

INSERT INTO Envios500 **VALUES** (P2, A2, 300);

UPDATE Envios500 **SET** cantidad=100

WHERE id_proveedor= P1;

Vistas Actualizables

CLÁUSULA WITH CHECK OPTION

Ejemplos

```
CREATE VIEW Envios500_1000
AS
SELECT * FROM Envios500
WHERE cantidad<=1000
WITH LOCAL CHECK OPTION;
```

```
CREATE VIEW Envios500_1000
AS
SELECT * FROM Envios500
WHERE cantidad<=1000
WITH CASCADE CHECK OPTION;
```

	LOCAL	*CASCADE*
INSERT INTO Envios500_1000 VALUES (P3, A3, 300);	√	×
INSERT INTO Envios500_1000 VALUES (P4, A4, 700)	√	√
INSERT INTO Envios500_1000 VALUES (P5, A5, 1300);	×	×

Pausa

Actualización de Vistas

TRIGGERS INSTEAD OF

Se pueden "interceptar" las operaciones de actualización sobre una vista mediante triggers INSTEAD OF (opción especial para vistas)

- Recurso que **permite la actualización de vistas** que no pueden ser actualizadas vía sentencias del DML (UPDATE, INSERT, o DELETE)

- Definir triggers instead of para las distintas operaciones
- El trigger se dispara **en lugar de** ejecutar la sentencia disparadora, en forma invisible para el usuario
- Por defecto, los triggers INSTEAD OF son 'for each row'

Ejemplos

Actualización de Vistas

TRIGGERS INSTEAD OF

Dos ejemplos para utilizar esta funcionalidad:

- ★ Mantener consistentes las tablas que representan las jerarquías del modelo de datos

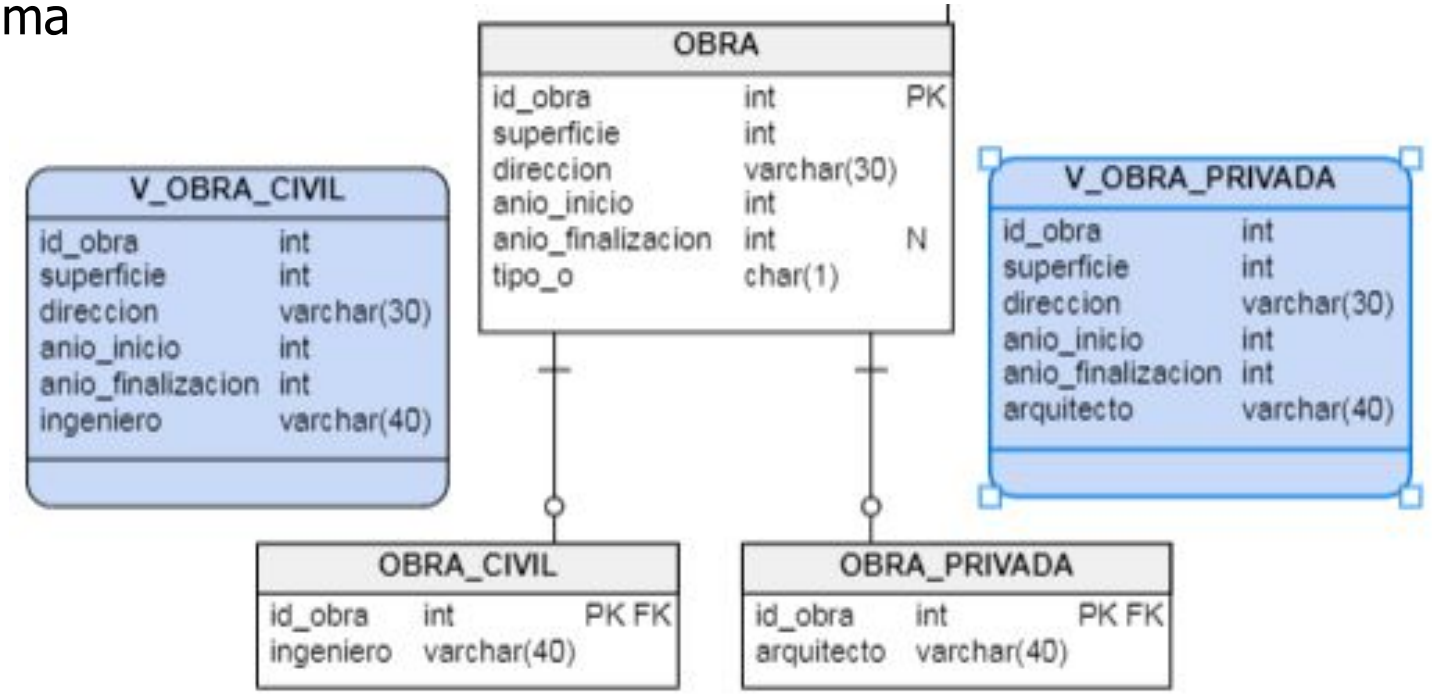
- ★ Generar registros de auditoría cuando se actualizan registros de otra tabla

Ejemplos

Actualización de Vistas

TRIGGERS INSTEAD OF

Esquema




```
CREATE VIEW V_OBRA_CIVIL AS
SELECT o.id_obra, superficie,
       direccion, anio_inicio,
       anio_finalizacion, ingeniero
FROM obra o NATURAL JOIN obra_civil c;
```

```
CREATE VIEW V_OBRA_PRIVADA AS
SELECT o.id_obra, superficie,
       direccion, anio_inicio,
       anio_finalizacion, arquitecto
FROM obra o NATURAL JOIN obra_privada p;
```

Ejemplos

Actualización de Vistas

TRIGGERS INSTEAD OF

```
CREATE OR REPLACE FUNCTION fn_obra_civil()
RETURNS TRIGGER AS $$
BEGIN
   IF (TG_OP = 'UPDATE') THEN
     UPDATE obra SET superficie = new.superficie,
            direccion = new.direccion,
            anio_inicio = new.anio_inicio,
            anio_finalizacion = new.anio_finalizacion
     WHERE id_obra = new.id_obra;
     UPDATE obra_civil SET ingeniero = new.ingeniero
     WHERE id_obra = new.id_obra;
   ELSE
     INSERT INTO obra (id_obra, superficie, anio_inicio, anio_finalizacion)
        VALUES (new.id_obra, new.superficie, new.anio_inicio,
    new.anio_finalizacion);
     INSERT INTO obra_civil (id_obra, ingeniero) VALUES (new.id_obra,
new.ingeniero);
    END IF;
RETURN NULL;
END;
$$ LANGUAGE plpgsql;
```

Ejemplos

Actualización de Vistas

TRIGGERS INSTEAD OF

```
CREATE TRIGGER tr_obra_civil
    INSTEAD OF INSERT OR UPDATE
    ON V_OBRA_CIVIL
    FOR EACH ROW
    EXECUTE PROCEDURE fn_obra_civil();
```

NOTA: la operación de DELETE puede ser manejada por medio de la acción referencial CASCADE en ambas tablas

Otra opción es realizar una única función para los 2 triggers y utilizar la variable TG_NAME para detectar cual es el trigger que ha llamado a la función

Ejemplos

Actualización de Vistas

TRIGGERS INSTEAD OF

```
CREATE OR REPLACE FUNCTION fn_audit_constructor()
RETURNS TRIGGER AS $$
BEGIN
    IF (TG_OP = 'DELETE') THEN
         INSERT INTO constructor_audit
            SELECT 'D', now(), user, o.*
            FROM old_table o;
         DELETE FROM CONSTRUCTOR
         WHERE tipo_doc, nro_doc IN (
               SELECT o.tipo_doc, o.nro_doc
               FROM old_table o);
    ELSIF (TG_OP = 'UPDATE') THEN
         INSERT INTO constructor_audit
            SELECT 'U', now(), user, n.*
            FROM new_table n;
         UPDATE constructor c
         SET (nombre, apellido, direccion) =
               (SELECT n.nombre, n.apellido, n.direccion
               FROM new_table n
               WHERE n.tipo_doc = c.tipo_doc
                 AND n.nro_doc = c.nro_doc);
    ……(continua)
```

CONSTRUCTOR_AUDIT	
operacion	char(1)
fecha_hora	timestamp
usuario	text
tipo_doc	char(3)
nro_doc	int
nombre	varchar(30)
apellido	varchar(30)
direccion	varchar(30)

CONSTRUCTOR		
tipo_doc	char(3)	PK
nro_doc	int	PK
nombre	varchar(30)	
apellido	varchar(30)	
direccion	varchar(30)	

Ejemplos

Actualización de Vistas

TRIGGERS INSTEAD OF

```
    ELSIF (TG_OP = 'INSERT') THEN
         INSERT INTO constructor_audit
             SELECT 'I', now(), user, n.*
             FROM new_table n;
         INSERT INTO constructor
             SELECT n.*
             FROM new_table n;
     END IF;
    -- se retorna NULL porque es un trigger AFTER
    RETURN NULL;
END;
$$ LANGUAGE plpgsql;
```

CONSTRUCTOR_AUDIT	
operacion	char(1)
fecha_hora	timestamp
usuario	text
tipo_doc	char(3)
nro_doc	int
nombre	varchar(30)
apellido	varchar(30)
direccion	varchar(30)

CONSTRUCTOR		
tipo_doc	char(3)	PK
nro_doc	int	PK
nombre	varchar(30)	
apellido	varchar(30)	
direccion	varchar(30)	

Ejemplos

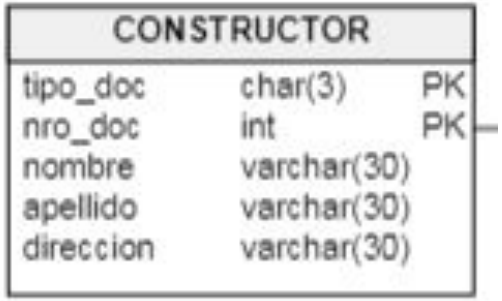
Actualización de Vistas

TRIGGERS INSTEAD OF

```
CREATE TRIGGER tr_audit_ins_constr
    INSTEAD OF INSERT ON constructor
    REFERENCING NEW TABLE AS new_table
    FOR EACH STATEMENT
    EXECUTE PROCEDURE fn_audit_constructor();

CREATE TRIGGER tr_audit_upd_constr
    INSTEAD OF UPDATE ON constructor
    REFERENCING OLD TABLE AS old_table
                NEW TABLE AS new_table
    FOR EACH STATEMENT
    EXECUTE PROCEDURE fn_audit_constructor();

CREATE TRIGGER tr_audit_del_constr
    INSTEAD OF DELETE ON constructor
    REFERENCING OLD TABLE AS old_table
    FOR EACH STATEMENT
    EXECUTE PROCEDURE fn_audit_constructor();
```



Vistas Materializadas

Una vista puede ser **materializada** → el SGBD precomputa y almacena su contenido

- Cuestiones de performance en la elaboración de la consulta → cuáles vistas conviene materializar? Qué índices definir?
- Cuestiones asociadas al mantenimiento de vistas materializadas:
  - Cómo mantener consistencia entre las tablas de la BD y los resultados de vistas materializadas?
  - Aplicar actualización por regeneración o incremental?

- Actualizar una vista (si estuviera materializada) debe resultar en la misma relación que si se modificaran las tablas base aplicando una o más actualizaciones y luego aplicando la definición de la vista

Vistas

COMO MECANISMO DE SEGURIDAD

- Definiendo diferentes vistas y otorgando privilegios selectivamente sobre ellas, puede restringirse el acceso de los usuarios a ciertos subconjuntos de datos:
  - Vistas sobre determinadas columnas, ocultando otras reservadas para usuarios específicos → ej. *antecedentes penales*
  - Vistas sobre determinadas filas, ocultando otras reservadas para usuarios específicos → ej. *películas no aptas para el público infantil*
  - Vistas sobre determinadas columnas y filas
- Los usuarios no deberían notar la diferencia entre el uso de tablas o vistas

Vistas

VENTAJAS

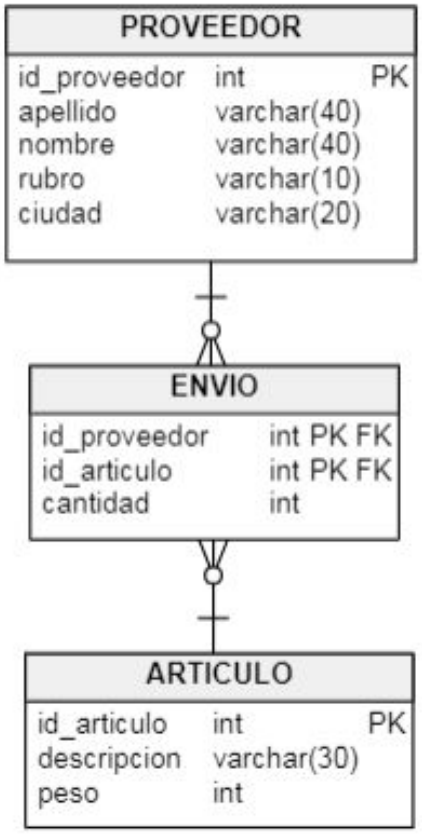
- **Simplifican la percepción** que los usuarios tienen de la BD, presentando la información necesaria y ocultando el resto
- **Presentan diferentes datos** a distinto tipo de usuarios, aún cuando los estén compartiendo (misma BD)
- **Permiten definir consultas complejas/frecuentes** para no tener que especificarlas cada vez que se utilizan
- **Facilitan la independencia de los datos** (ocultando a los usuarios cambios en la estructura en las tablas base)
- **Permiten aplicar políticas de seguridad** (privacidad) de los datos (acceso restringido)

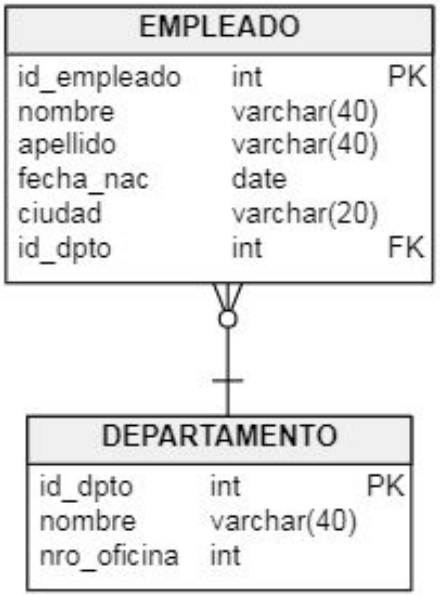
Vistas

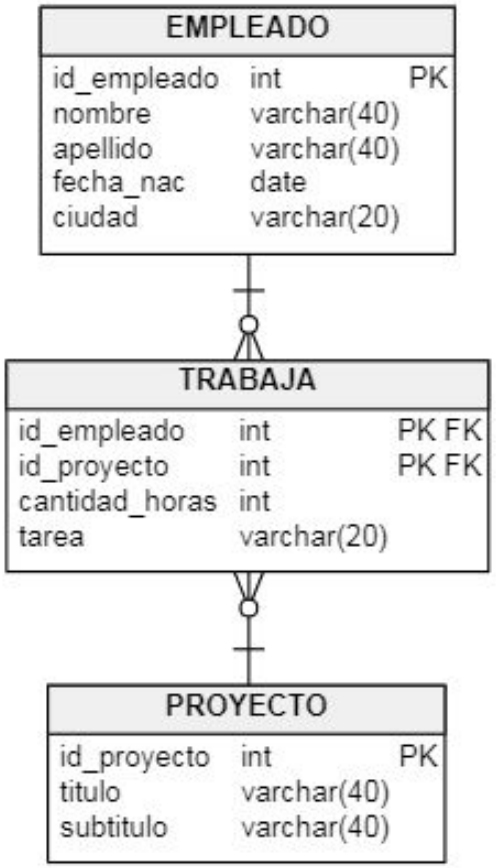
DESVENTAJAS

- Las actualizaciones sobre las vistas son restringidas (hay varias limitaciones sobre la estructura de las vistas)
- Restricciones estructurales: la estructura de una vista se determina en el momento de su creación, entonces si los componentes cambian, no son considerados.
- Rendimiento: el proceso de resolución de la vista puede exigir el acceso a múltiples tablas cada vez que se accede a ella, entonces implica un procesamiento adicional (puede justificarse su materialización - técnicas alternativas de mantenimiento de vistas)

Scripts utilizados







Script de Esquema 1

Script de Esquema 2

Script de Esquema 3

Script de Creación de Vistas