Introducción a las Bases de Datos y Bases de Datos

VISTAS

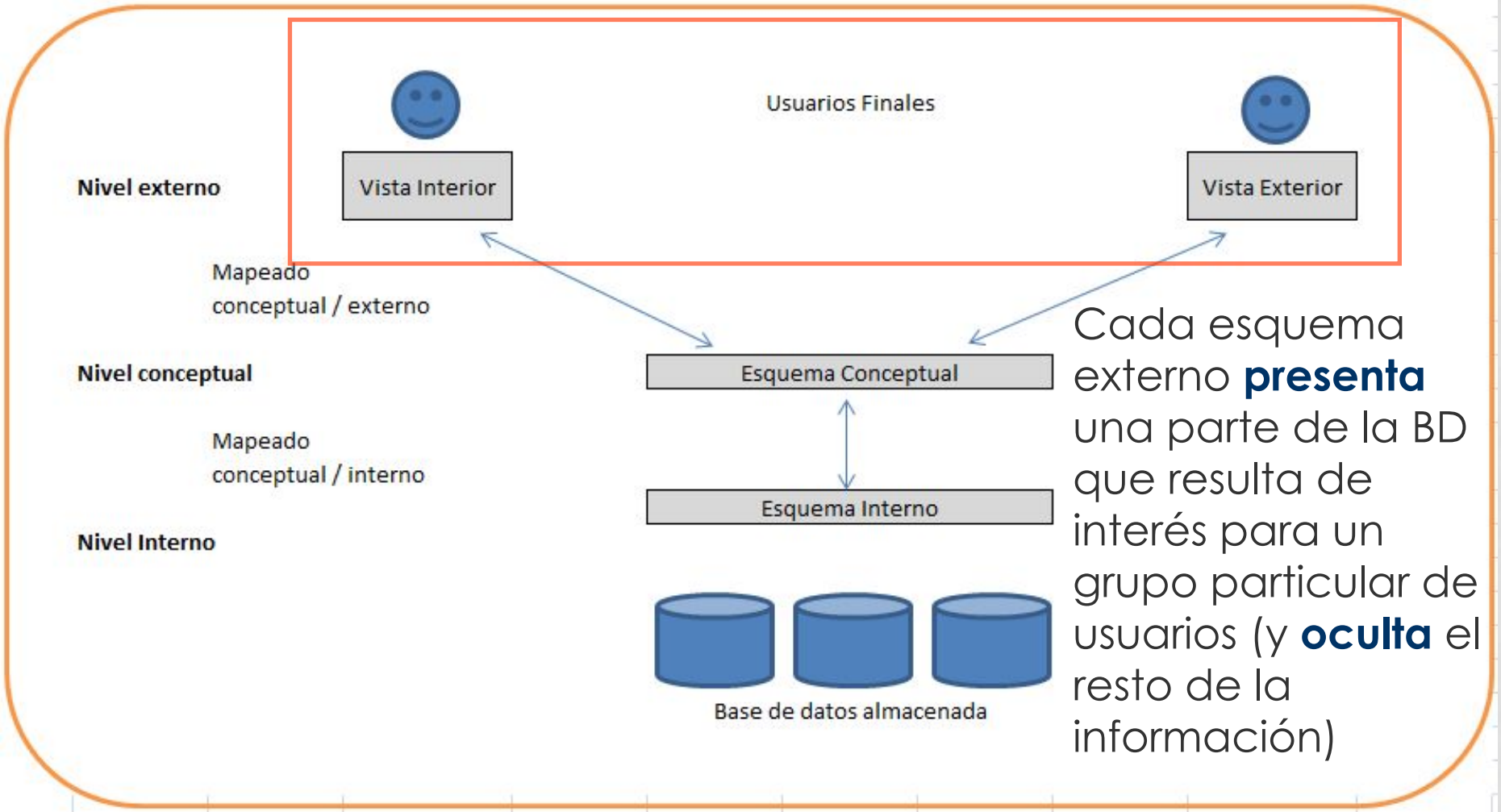
TEORÍA 8

2017

Tecnicaturas TUPAR y TUDAI

VISTAS: ESQUEMA EXTERNO

Las VISTAS forman parte del **esquema externo** de la BD - elementos del catálogo



Cada esquema externo **presenta** una parte de la BD que resulta de interés para un grupo particular de usuarios (y **oculta** el resto de la información)

VISTAS: CONCEPTO

- Es una **tabla virtual** (habitualmente no materializada)
  → las tuplas se generan al operar sobre la vista, según su definición (no necesariamente existen en forma física)
- Una vista es una relación **derivada** que se define dando un nombre a una expresión de consulta
- El contenido de una vista está definido como una consulta sobre **una o varias tablas (u otras vistas)**

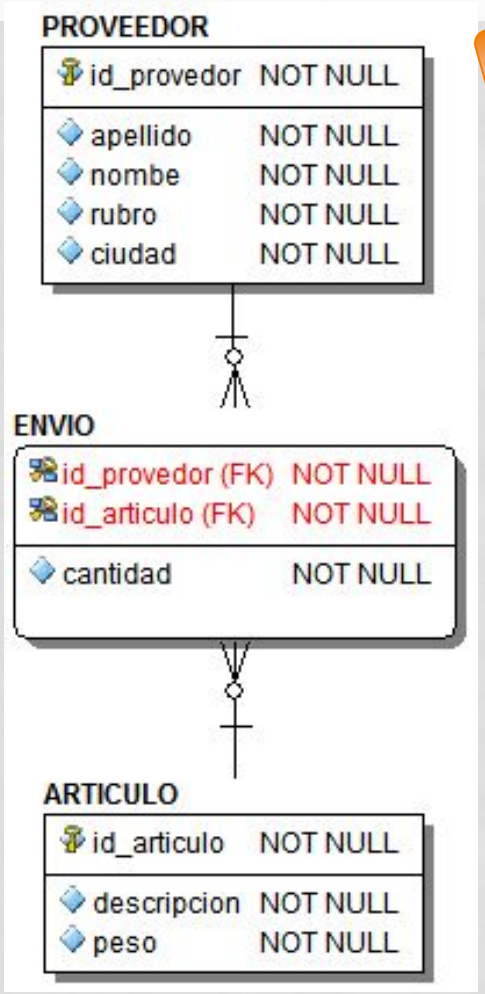
VISTAS: CREACIÓN

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
  - Las columnas de la vista se pueden nombrar especificando la lista completa de atributos de la vista entre paréntesis.
  - Si no se especifican nuevos nombres, los nombres son los mismos que los de las columnas de las tablas especificadas en la sentencia SELECT
  - es necesario especificar con diferente nombre aquellas columnas provenientes de distintas tablas pero con igual nombre (ambigüas)
- expresión_consulta: consulta que define la relación derivada

VISTAS: CREACIÓN

Vistas a partir de una tabla



Ejemplos

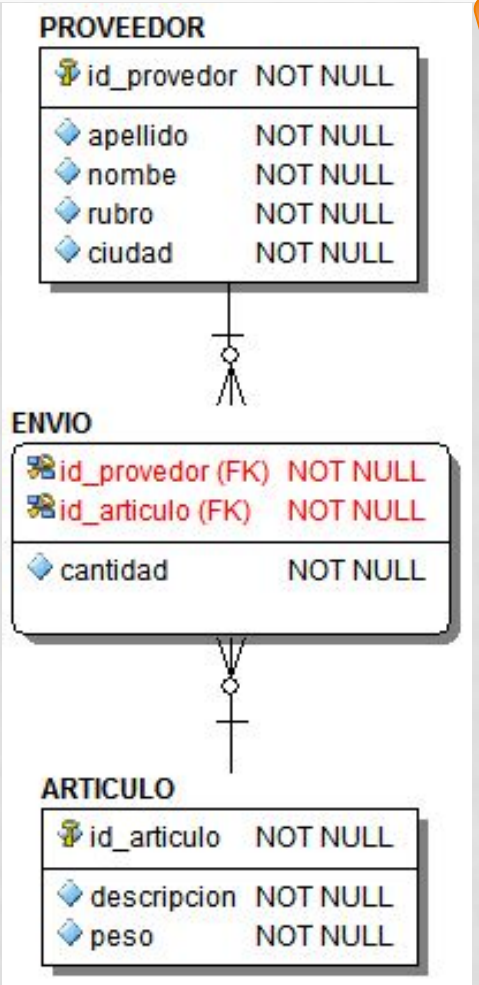
- *PROV_TANDIL* que contenga los id_ y nombre de los proveedores de Tandil

```
CREATE VIEW PROV_TANDIL
AS SELECT id_proveedor, nombre
    FROM PROVEEDOR
   WHERE ciudad = ´Tandil´;
```

- *TOTAL_ARTICULO* que liste la cantidad total enviada de cada artículo

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

VISTAS: CREACIÓN



Ejemplos

- Crear una Vista *PR_COMP* que contenga los id_ y nombre de los proveedores de rubro computadoras

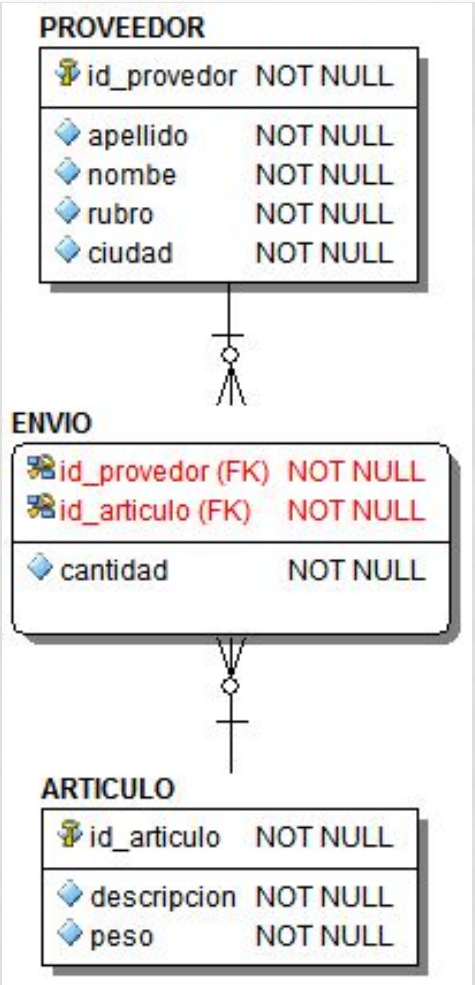
```
CREATE VIEW PR_COMP
AS SELECT id_proveedor, nombre, ciudad
     FROM PROVEEDOR
    WHERE rubro = 'Computadoras´;
```

- *Proveedores de computadoras de Tandil (usar vista previa):*

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

Vista a partir de otra vista

VISTAS: CREACIÓN



Ejemplos

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

```
CREATE VIEW PROV_ENVIOS_TANDIL
AS SELECT P.id_proveedor, P.nombre, E.id_articulo
    FROM PROVEEDOR P, ENVIO E
    WHERE P.id_proveedor = E.id_proveedor;
```

Vista a partir de más de una tabla

VISTAS: CONSULTAS Y ELIMINACIÓN

Ejemplos

- Una vista puede ser consultada como cualquier tabla

  Listar alfabéticamente los proveedores de *PR_COMP_TANDIL*

  SELECT nombre
  FROM *PR_COMP_TANDIL*
  ORDER BY nombre;

- 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

VISTAS: ACTUALIZACIONES

- Cuando se **actualizan las tuplas 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
- Las vistas no mantienen copias de los datos (salvo las vistas materializadas)
- Si hay actualizaciones de los datos de la BD (en las tablas)
  → el SGBD debe actualizar las vistas que los utilizan:
  - por recálculo (sus tuplas se generan al acceder a la vista)
  - por mantenimiento incremental (vistas materializadas)

VISTAS: ACTUALIZACIONES

- Las operaciones de **actualización sobre una vista DEBERÍA propagarse automáticamente** a operaciones sobre las tablas base
- 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**?
    o sobre el valor resultante de una **función de agregación**?
    o 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:
  - conserva todas las columnas de la clave (primaria o alternativa)
  - no contiene funciones de agregación o información derivada
  - no incluye la claúsula DISTINCT
  - no incluye subconsultas en el SELECT
  - se denominan **vistas de proyección-selección** (permiten sólo **selección** sobre las tuplas y **proyección** de columnas)
- Algunos SGBD soportan otras posibilidades que las especificadas por el estándar SQL

VISTAS ACTUALIZABLES

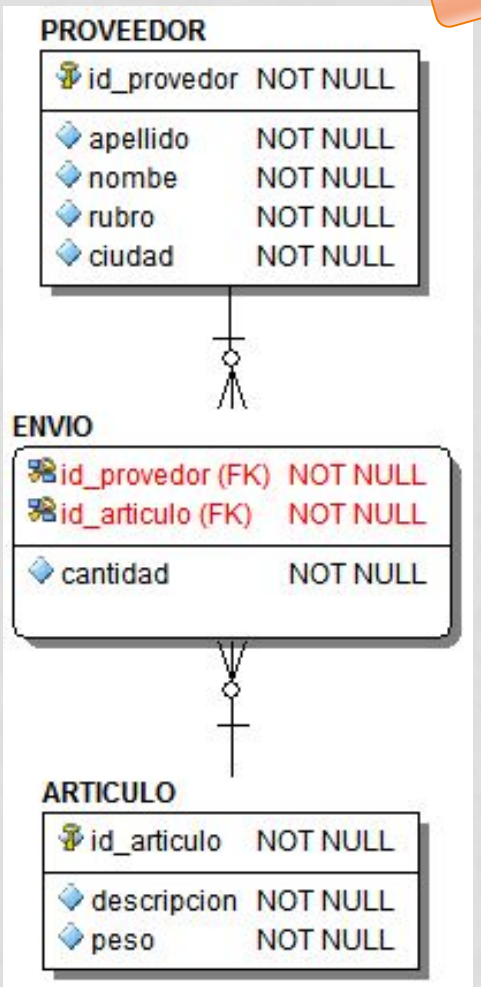
VISTAS DEFINIDAS SOBRE UNA TABLA

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

  siempre que no se viole ninguna restricción de integridad definida sobre dicha relación *(ej: afectando campos que no aceptan nulos o no tienen valores por defecto definidos)*

VISTAS ACTUALIZABLES

VISTAS DEFINIDAS SOBRE UNA TABLA



Ejemplos

Las siguientes vistas son actualizables?

PROV_TANDIL1 con id y nombre de los proveedores de Tandil

```
CREATE VIEW PROV_TANDIL1
AS SELECT id_proveedor, 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 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

- La actualización no se realizará si viola alguna restricción definida sobre la relación base a actualizar. Esto no significa que la vista no sea actualizable, sólo que se viola alguna RI.

VISTAS

Propiedad de **preservación de la clave**

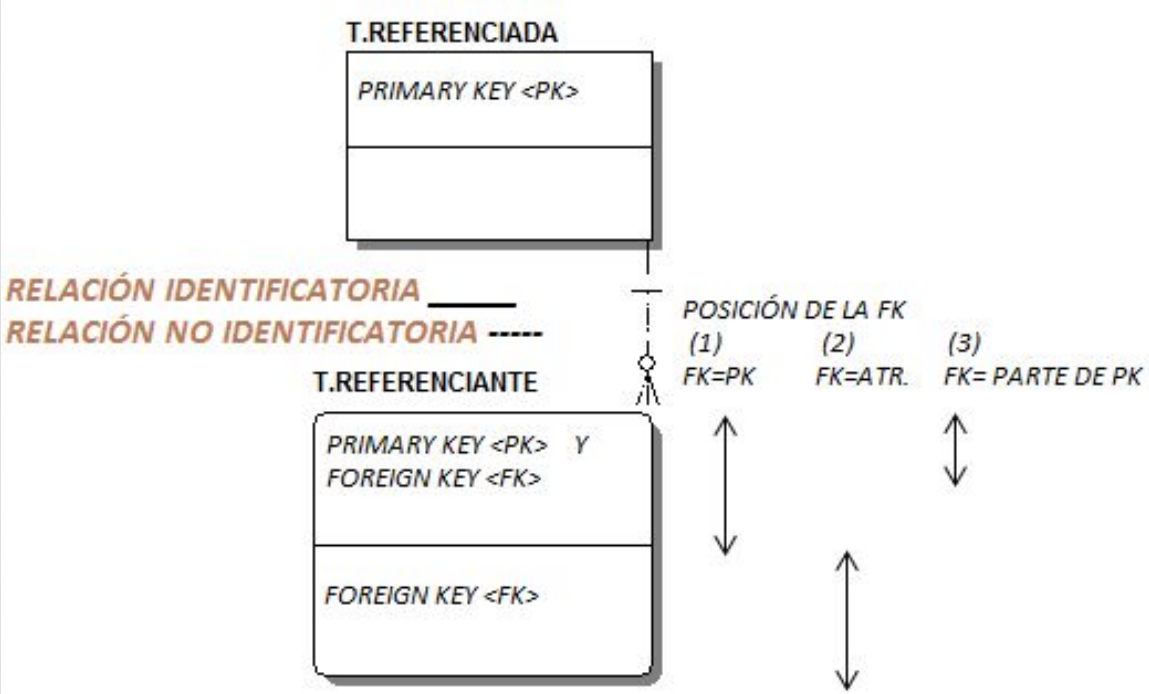
vistas 'key-preserved'

- 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: PRESERVACIÓN DE LA CLAVE



En las **Restricciones de Integridad Referencial** (RIRs) el conjunto de atributos que conforma la Foreign Key **(FK)** en la tabla referenciante se corresponde con el conjunto de atributos que conforman la clave primaria **(PK)** de la tabla referenciada.

Existen varias **ubicaciones de la FK**, en la tabla referenciante con respecto a **su PK**

1. FK ≡ → procede de la representación de relaciones de tipo-subtipo del DERExt.
2. FK ≡ A_1,..., A_r (siendo FK ∩ PK = ∅) → proviene de relaciones 1:1, N:1 o *n*-arias con al menos una cardinalidad 1 del DERExt.
3. FK ≡ PK_1,..., PK_q (o sea FK ⊂ K)→ resulta de las relaciones N:N, *n*-arias en general y las correspondientes a los vínculos entidad fuerte-entidad débil

VISTAS ACTUALIZABLES

VISTAS DEFINIDAS SOBRE MÁS DE UNA TABLA

Ejemplos

Vista ensamble mediante una RIR – ubicación 1 (relación tipo-subtipo), considerando las siguientes tablas:

EMPL(id_empl, nombre, fecha_nac, ciudad)
ING (id_empl, fecha_grad, título, univ)

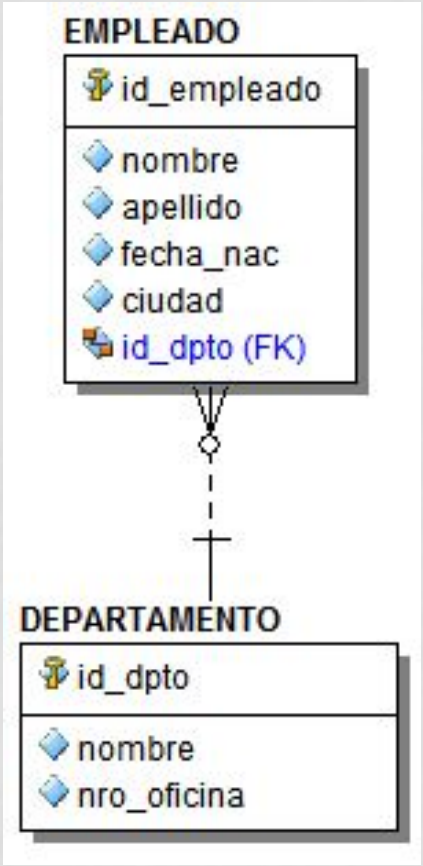
```
CREATE VIEW ING_TANDIL AS
SELECT I.id_empl, fecha_grad, título, univ
FROM ING I, EMPL E
WHERE I.id_empl = E.id_empl;
```

La clave de ING_TANDIL es I.id_empl (subtipo)
Es esperable que que el dominio de definición de E.id_empl sea más extenso

→ Operaciones de actualización sobre la vista?

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    DEPARTAMENTO D,
        EMPLEADO  E
WHERE D.id_depto = E.id_depto;
```

La clave de EMPL_SISTEMAS es E.id_empleado

VISTAS ACTUALIZABLES

VISTAS DEFINIDAS SOBRE MÁS DE UNA TABLA

Ejemplos

Vista ensamble mediante una RIR – relaciones N:N

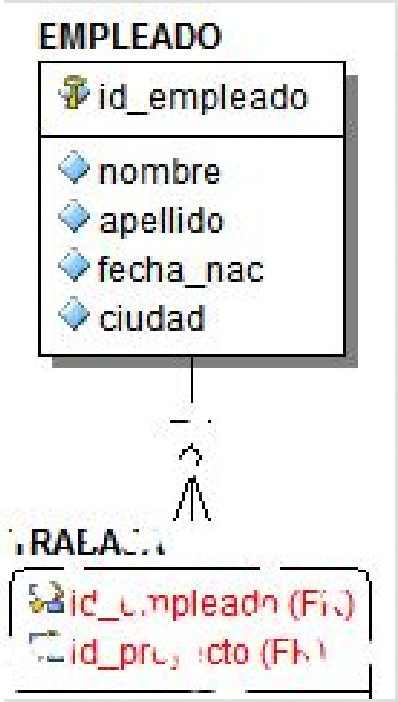
```
CREATE VIEW EMPL_PROY  AS
  SELECT id_empleado, id_proyecto,
cantidad_horas, tarea
  FROM TRABAJA T,  EMPLEADO E
  WHERE T.id_empleado = E.id_empleado
  AND E.ciudad = 'TANDIL';
```

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

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

CLÁUSULA WITH CHECK OPTION

Solución → 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;
```

determinar el efecto de las siguientes operaciones, considerando si
a) se define con WCO,
b) se define sin WCO:

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

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

UPDATE Envios500 SET cantidad=100  WHERE id_proveedor= P1;
```

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'

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)