IIC2413 ⋆ Bases de Datos

Proyecto: Entrega 2

Yuwen Cheng Hung *[1] and Brian Tapia Contreras†[2]

[1]Escuela de Ingenieria, Pontificia Universidad Catolica de Chile
[2]Instituto de Fisica, Pontificia Universidad Catolica de Chile

Esquema Inicial

- **Usuarios**(id, nombre, rut, edad, sexo, dirección)
- **ProductosV2**(id, nombre, precio, descripción, largo(cm), alto(cm), ancho(cm), peso(kg), fecha_de_caducidad, duracion_sin_refrigerar, tipo_de_conserva, tienda)
- **ComprasV2**(id, comprador, direccion, producto, cantidad, tienda)
- **trabajadores**(id, nombre, rut, edad, sexo, tienda)
- **plano_coberturaV2**(id, nombre, dirección, jefe, comuna_de_cobertura)
- **direcciones**(id, nombre_direccion, comuna)

Notamos que este esquema no está normalizado. A continuación se enlistan los cambios que se realizaron a este esquema, a fin de que quede normalizado.

- Dado que un usuario puede tener más de una dirección, encontramos conveniente separar esta información en dos tablas. Con esto, logramos guardar los usuarios de manera independiente, y en caso, por ejemplo, de actualización en la dirección de algún usuario, no perdemos información acerca de este, sólo modificamos la nueva tabla auxiliar.
  Esta tabla dependería del id del usuario solamente, y asociaría una dirección. Nombre, rut, edad y sexo se mantienen.
- En productos se tiene el problema de que los atributos no son comunes a todos los productos que se pueden guardar. Se prefirió entonces eliminar la posibilidad de guardar tantos nulos, y guardar, en su lugar, más tablas con productos más específicos.
  En pos de esto se crearon, por ejemplo, las tablas de **Comestibles** y **NoComestibles**, que tienen atributos en común. Cada uno de estos se dividió en más subgrupos, eliminando todos los atributos que almacenaran valores nulos.
- De manera análoga al primer punto, una compra puede (y lo hace) contemplar más de un producto distinto. Es por ello que se creó una tabla **Detalle**, que, mediante el id único de la compra, almacena tanto el producto como su cantidad. Por supuesto, un join entre estas tablas es la tabla original.
- A la tabla **trabajadores** se le agrefó un atributo *cargo*. Este puede ser o Empleado o Jefe. Lo hicimos así debido a la simplicidad que ofrece para realizar algunas consultas comunes.

*ycheng@uc.cl
†brian.tapia@uc.cl

- Al momento de guardar plan_coberturaV2 se realizó una reestructuración completa. Los despachos (comuna de cobertura) se almacenaron en una tabla única, a la cual se accede mediante el id de la tabla. Esta mostraría tabla_id, comuna_despacho.

  Debido al nuevo atributo de los trabajadores, el Jefe de cada tienda ya no tiene sentido definirlo según el esquema inicial.

  El id, nombre y dirección se guardan, esta vez bajo el nombre de **Tienda**.

- Se creó por último una tabla **comunas**, que almacena un id de dirección con el respectivo nombre y la comuna. Esta tabla sería equivalente a la última mostrada en el esquema anterior. No confundir con la nueva tabla a la que llamamos **direcciones**. Esta última almacena las distintas direcciones para un usuario.

Diagrama E/R

A continuación se adjunta el diagrama E/R resultante de lo mencionado anteriormente.

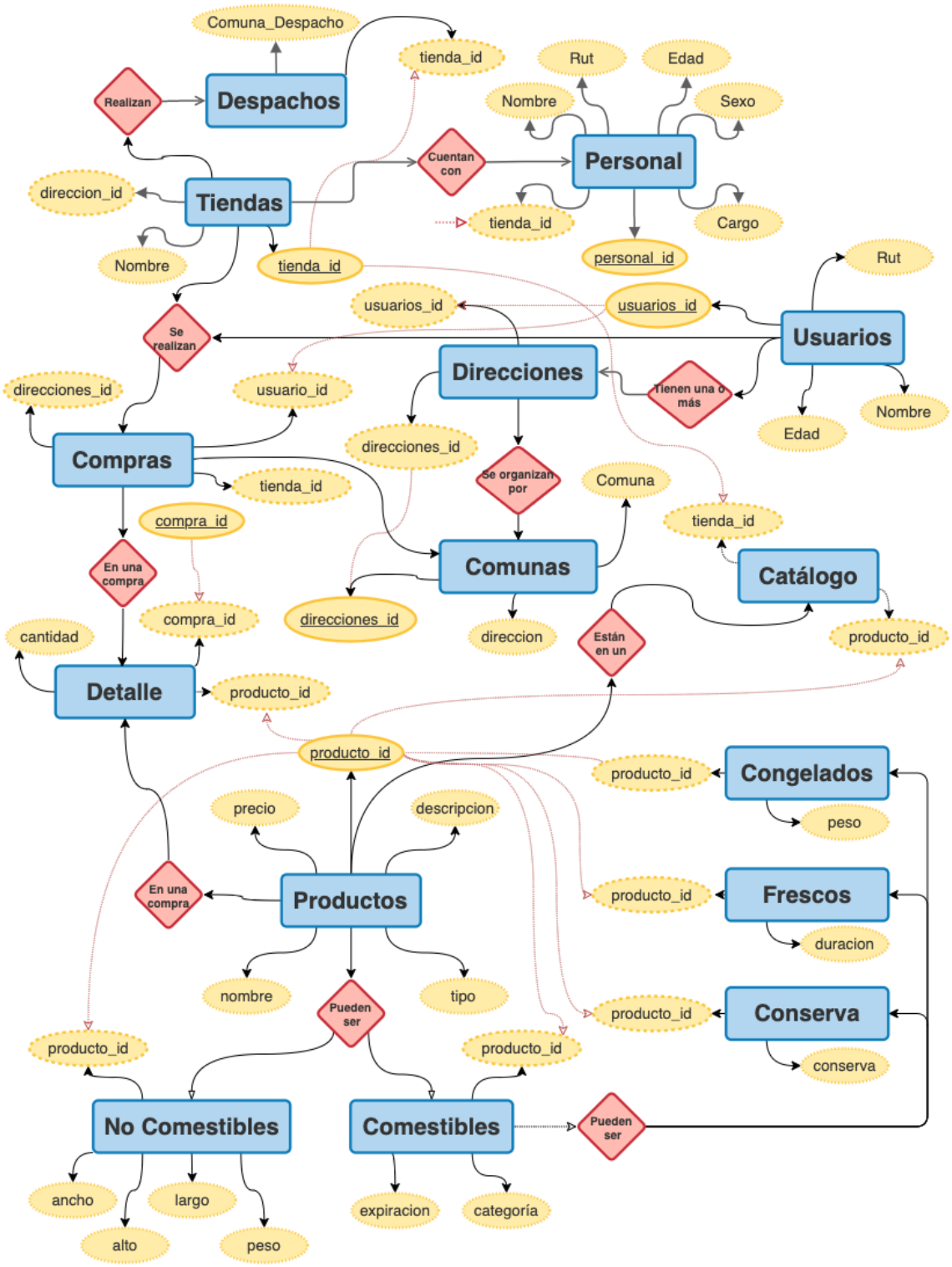

Figura 1: Diagrama E/R final. Se indican en línea continua las llaves primarias, en línea segmentada las llaves foráneas, y en línea punteada los atributos.

Esquema Relacional Resultante

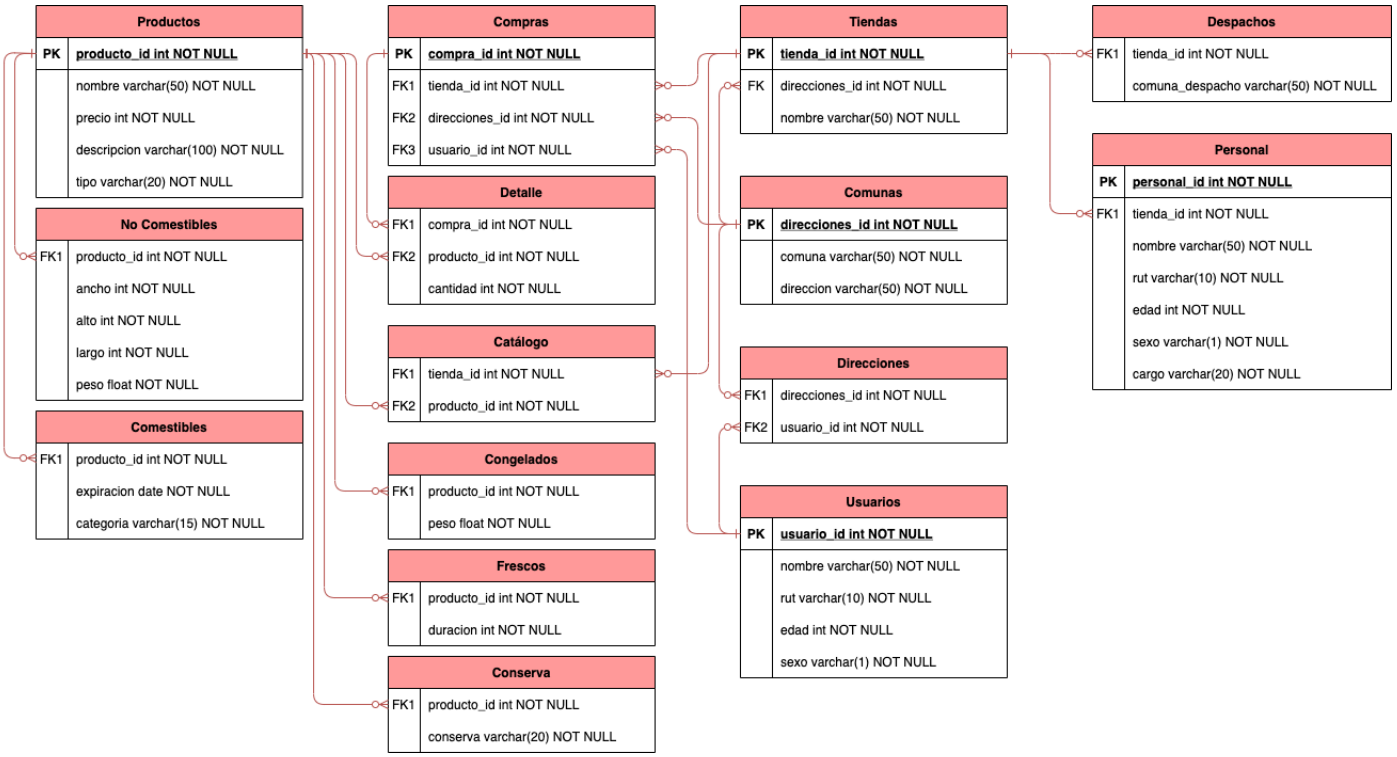

Figura 2: Esquema Relacional.

Se prefirió escribir en forma gráfica, a diferencia de como se presentó el esquema inicial, pero en escencia ambos esquemas muestran lo mismo. Se puede notar en cada atributo el tipo de dato y se indican las llaves primarias y foráneas.

Consultas

Consulta 1

```
SELECT DISTINCT t.nombre, d.Comuna_Despacho
FROM Tiendas as t
  JOIN Despachos as d ON t.tienda_id = d.tienda_id
ORDER BY t.nombre
```

Consulta 2

```
SELECT DISTINCT p.Nombre
FROM
  Personal as p,
  Tiendas  as t,
  Comunas  as c
WHERE
  p.Tienda_id    = t.Tienda_id    AND
  t.direccion_id = c.Direccion_id AND
  p.Cargo        LIKE '%Jefe%'    AND
  c.Comuna       LIKE '%$comuna%'
```

Consulta 3

```
SELECT DISTINCT t.Nombre
FROM
  Tiendas   as t,
  Catalogo  as c,
  Productos as p
WHERE
  t.tienda_id   = c.tienda_id   AND
  c.producto_id = p.producto_id AND
  p.tipo        = '$tipo'
```

Consulta 4

```
SELECT u.nombre, p.nombre, p.descripcion
FROM
  Usuarios  as u,
  Productos as p,
  Compras   as c,
  Detalle   as d
WHERE
  p.producto_id = d.producto_id AND
  d.compra_id   = c.compra_id   AND
  c.usuario_id  = u.usuario_id  AND
  p.descripcion LIKE '%$descripcion%'
```

Consulta 5

```
SELECT DISTINCT ROUND(AVG(p.edad),0)
FROM
  Tiendas  as t,
  Personal as p,
  Comunas  as c
WHERE
  t.tienda_id    = p.tienda_id    AND
  c.direccion_id = t.direccion_id AND
  c.comuna       = '$comuna'
```

Consulta 6

```
SELECT t.nombre, SUM(d.cantidad) as suma
FROM
  Tiendas   as t,
  Productos as p,
  Detalle   as d,
  Compras   as c
WHERE
  t.tienda_id   = c.tienda_id  AND
  c.compra_id   = d.compra_id  AND
  d.producto_id = p.producto_id AND
  p.tipo        = '$tipo'

GROUP BY t.nombre
ORDER BY suma DESC
```

Consideraciones

El supuesto más importante que se realizó fue que, para lograr que la búsqueda sea *case-insensitive*, se asumió que los datos en las tablas (por lo menos aquellos en las columnas de comuna y descripción, que son los pedidos en las consultas) estaban escritos en minúscula.