DEPARTAMENTO
DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico 1

24 de Abril de 2013 — Bases de Datos

Integrante	LU	Correo electrónico
Agustina Ciraco	630/06	agusciraco@gmail.com
Nadia Heredia	589/08	heredianadia@gmail.com
Pablo Antonio	290/08	pabloa@gmail.com
Vanesa Stricker	159/09	vanesastricker@gmail.com

Índice

1. Introducción **3**

2. Diagrama Entidad-Relación **4**
2.1. Aclaraciones y Restricciones 4

3. Modelo Relacional **6**

4. Consultas **10**
4.1. Otras Aclaraciones 16

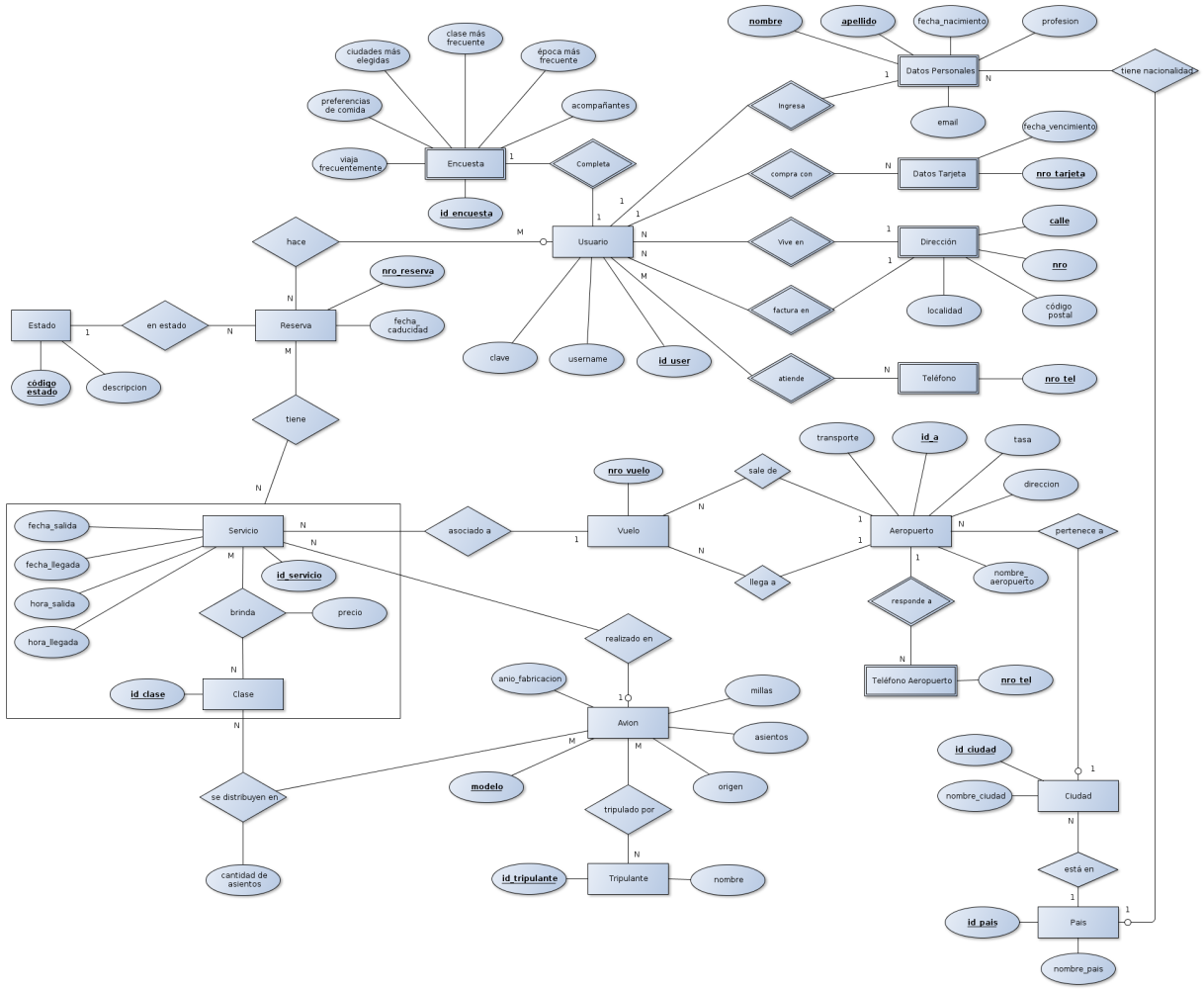
1. Introducción

En este trabajo práctico se implementa una solución a un problema de la vida real, implementando una solución usando las herramientas de un motor de bases de datos.

En este caso el problema es el de manejo de un sistema de información online de una aerolínea. Se diseña e implementa una base de datos que brinde soporte para lograrlo.

Los servicios que debe brindar el sistema de la aerolínea incluyen la apertura de cuentas personales para clientes, el manejo de reservas para vuelos, y el confeccionamiento de diferentes planes de viaje.

2. Diagrama Entidad-Relación



(a) der

2.1. Aclaraciones y Restricciones

- Consideramos que una reserva puede tener varios estados, estos son
  - Realizada
  - Cancelada
  - Confirmada

  Las reservas realizadas pueden cancelarse o confirmarse. Es necesario confirmar una reserva para poder viajar. Para simplificar, suponemos que todos los usuarios con reservas confirmadas realizan efectivamente el viaje.

- La aerolínea ofrece servicios de viaje, y cada usuario realiza una reserva sobre pares de servicio-clase. De esta forma vemos la cantidad de asientos que está reservando por cada clase para un servicio. Cada servicio especifica la fecha y el horario del viaje. Además tiene asociado un avión, que es el avión con el que se realizará el viaje. A la vez un servicio está vinculado con un vuelo. Un vuelo es un conjunto de servicios sin especificar. Es decir, un vuelo indica que dias de la semana se brindarán servicios de un aeropuerto origen a uno destino. Así, por ejemplo, un servicio podría ser: "Un vuelo de Aeropuerto de origen Ezeiza- Buenos Aires, Aeropuerto destino Newark-New Jersey el día 26/04/2013, 15:00hs, Primera Clase". En cambio, un ejemplo de vuelo es: "Vuelos de Aeropuerto origen Barajas-Madrid, Aeropuerto destino JFK-New York, los días martes y jueve".
- Consideramos que los mensajes que el sistema le envía a los usuarios no son lo suficientemente relevantes para el problema que queremos modelar.
- Una reserva que se encuentra en estado Realizada, está asociada con un servicio cuya fecha y hora de salida es anterior a la fecha actual.
- La suma de la cantidad de asientos por clase en un avión (en la interrelación se distribuyen en) es igual a lo que indica el atributo asientos de la entidad avión.
- Este modelo no contempla la posibilidad de que la aerolinea cancele un vuelo. Con lo cual todo vuelo existente es efectuado.

3. Modelo Relacional

primary key	foreign key

Usuario(id_user, username, clave)

- $FK = \{\}$
- $PK = CK = \{id\_user\}$

Encuesta(id_user, id_encuesta, viaja_frecuentemente, preferencias_de_comida, ciudades_elegidas, clase_mas_frecuente, epoca_mas_frecuente, acompaniantes)

- $FK = \{id\_user\}$
- $PK = CK = \{(id\_user, id\_encuesta)\}$
- $Encuesta.id\_user$ debe estar en $Usuario.id\_user$

DatosPersonales(id_user, nombre, apellido, nacionalidad, email, profesión, fecha_nacimiento)

- $FK = \{id\_user, id\_pais\}$
- $PK = CK = \{(id\_user, nombre, apellido\}$
- $DatosPersonales.id\_user$ debe estar en $Usuario.id\_user$
- $DatosPersonales.id\_pais$ debe estar en $Pais.id\_pais$

DatosTarjeta(id_user, nro_tarjeta, fecha_vencimiento)

- $FK = \{id\_user\}$
- $PK = CK = \{(id\_user, nro\_tarjeta\}$
- $DatosTarjeta.id\_user$ debe estar en $Usuario.id\_user$

Direccion(id_user, calle, nro, localidad, código postal)

- $FK = \{id\_user\}$
- $PK = CK = \{(id\_user, calle, numero)\}$
- $Direccion.id\_user$ debe estar en $Usuario.id\_user$

Atiende(id_user, nro_tel)

- $FK = \{id\_user, nro\_tel\}$
- $PK = CK = \{(id\_user, nro\_tel)\}$
- $Atiende.id\_user$ debe estar en $Usuario.id\_user$
- $Atiende.nro\_tel$ debe estar en $Telefono.nro\_tel$

Telefono(id_user, nro_tel)

- $FK = \{id\_user\}$
- $PK = CK = \{(id\_user, nro\_tel)\}$
- $Telefono.id\_user$ debe estar en $Usuario.id\_user$

Reserva(nro_reserva, fecha_caducidad, codigo_estado)

- $FK = \{estado\}$
- $PK = CK = \{nro\_reserva\}$
- $Reserva.codigo\_estado$ debe estar en $Estado.codigo\_estado$

Hace(nro_reserva, id_user)

- $FK = \{nro\_reserva, id\_user\}$
- $PK = CK = \{(nro\_reserva, id\_user)\}$
- $Hace.nro\_reserva$ debe estar en $Reserva.nro\_reserva$
- $Hace.id\_user$ debe estar en $Usuario.id\_user$

Estado(codigo_estado, descripcion)

- $FK = \emptyset$
- $PK = CK = \{codigo\_estado\}$

Tiene(nro_reserva, id_servicio, id_clase)

- $FK = \{nro\_reserva, id\_servicio, id\_clase\}$
- $PK = CK = \{(nro\_reserva, id\_servicio, id\_clase)\}$
- $Tiene.id\_servicio$ debe estar en $Servicio.id\_servicio$
- $Tiene.id\_clase$ debe estar en $Clase.id\_clase$
- $Tiene.nro\_reserva$ debe estar en $Reserva.nro\_reserva$

Servicio(id_servicio, fecha_salida, fecha_llegada, hora_salida, hora_llegada, nro_vuelo, modelo)

- $FK = \{nro\_vuelo, modelo\}$
- $PK = CK = \{id\_servicio\}$
- $Servicio.nro\_vuelo$ debe estar en $Vuelo.nro\_vuelo$
- $Servicio.modelo$ debe estar en $Avion.modelo$

Brinda(id_servicio, id_clase, precio)

- $FK = \{id\_servicio, id\_clase\}$
- $PK = CK = \{(id\_servicio, id\_clase)\}$
- $Brinda.id\_servicio$ debe estar en $Servicio.id\_servicio$
- $Brinda.id\_clase$ debe estar en $Clase.id\_clase$

Clase(nro_reserva, id_user)

- $FK = \{nro\_reserva, id\_user\}$
- $PK = CK = \{(nro\_reserva, id\_user)\}$
- $Hace.nro\_reserva$ debe estar en $Reserva.nro\_reserva$
- $Hace.id\_user$ debe estar en $Usuario.id\_user$

Vuelo(nro_vuelo, aeropuerto_salida, aeropuerto_llegada)

- $FK = \{aeropuerto\_salida, aeropuerto\_llegada\}$
- $PK = CK = \{nro\_vuelo\}$
- $Vuelo.aeropuerto\_salida$ debe estar en $Aeropuerto.id\_a$
- $Vuelo.aeropuerto\_llegada$ debe estar en $Aeropuerto.id\_a$

Avion(modelo, anio_fabricacion, millas, asientos, origen)

- $FK = \emptyset$
- $PK = CK = \{modelo\}$

SeDistribuyenEn(modelo, id_clase, cantidad_asientos)

- $FK = \{modelo, id\_clase\}$
- $PK = CK = \{(modelo, id\_clase)\}$
- $SeDistribuyenEn.modelo$ debe estar en $Avion.modelo$
- $SeDistribuyenEn.id\_clase$ debe estar en $Clase.id\_clase$

Tripulante(id_tripulante, nombre)

- $FK = \emptyset$
- $PK = CK = \{id\_tripulante\}$

TripuladoPor(id_tripulante, modelo)

- $FK = \{id\_tripulante, modelo\}$
- $PK = CK = \{(id\_tripulante, modelo)\}$
- $TripuladoPor.id\_tripulante$ debe estar en $Tripulante.id\_tripulante$
- $TripuladoPor.modelo$ debe estar en $Avion.modelo$

Pais(id_pais, nombre_pais)

- $FK = \emptyset$
- $PK = CK = \{id\_pais\}$

Ciudad(id_ciudad, nombre_ciudad, id_pais)

- $FK = \{id\_pais\}$
- $PK = CK = \{id\_ciudad\}$
- $Ciudad.id\_pais$ debe estar en $Pais.id\_pais$

TelefonoAeropuerto(nro_tel, id_a)

- $FK = \{id\_a\}$
- $PK = CK = \{(nro\_tel, id\_a\}$
- $TelefonoAeropuerto.id\_a$ debe estar en $Aeropuerto.id\_a$

Aeropuerto(id_a, tasa, direccion, transporte, nombre_aeropuerto, id_ciudad)

- $FK = \{id\_ciudad\}$
- $PK = CK = \{id\_a\}$
- $Aeropuerto.id\_ciudad$ debe estar en $Ciudad.id\_ciudad$

4. Consultas

Consulta 1

```
select dp.nombre, dp.apellido, id_user
from usuario u, datospersonales dp
where not exists
(
    select *
    from pais p
    where not exists
    (
        select *
        from hace h, reserva r, tiene t, servicio s, vuelo v,
            aeropuerto a, ciudad c
        where u.id_user = h.id_user
        and h.nro_reserva = r.nro_reserva
        and r.nro_reserva = t.nro_reserva
        and t.id_servicio = s.id_servicio
        and s.nro_vuelo = v.nro_vuelo
        and (v.llega_a = a.id_a or v.sale_de = a.id_a)
        and a.id_ciudad = c.id_ciudad
        and c.id_pais = p.id_pais
        and ((s.fecha >= date_sub(now(), interval 5 year))
            or (s.fecha >= date_sub(now(), interval 5 year)))
    )
)
and u.id_user = dp.id_user
```

Consulta 2

```
create function viajes (@fecha_1 date, @fecha_2 date) as
begin
        select * from aeropuerto a
        inner join vuelo v
        on v.sale_de = a.id_a
        inner join servicio s
        on s.nro_vuelo = v.nro_vuelo
        inner join reserva r
        on r.id_servicio = s.id_servicio
        inner join usuario u
        on u.id_usuario = r.id_usuario
        as usuarios_que_salieron_de_aeropuerto

        select count( usuario ) as cant,
                date_format(fecha, '%m %y'), aeropuerto
        from usuarios_que_salieron_de_aeropuerto
        where fecha between @fecha_1 and @fecha_2
        group by fecha as personas_que_salieron

        select * from aeropuerto a
        inner join vuelo v
        on v.llega_a = a.id_a
        inner join servicio s
        on s.nro_vuelo = v.nro_vuelo
        inner join reserva r
        on r.id_servicio = s.id_servicio
        and r.estado = realizado
        inner join usuario u
        on u.id_usuario = r.id_usuario
        as usuarios_que_llegaron_a_aeropuerto

        select count( usuario ) as cant,
                date_format(fecha, '%m %y'), aeropuerto
        from usuarios_que_llegaron_a_aeropuerto
        where fecha between @fecha_1 and @fecha_2
        group by fecha as personas_que_llegaron

        select (s.cant + l.cant) as cant_total,
                s.cant, l.cant, fecha, aerop
        from personas_que_salieron as s,
                personas_que_llegaron as l
```

```
        order by cant_total
end
```

Consulta 3

```
create trigger chequear_reservas_superpuestas
before insertion on Reserva
for each row
begin
        fecha_salida := select s.fecha_salida
        from servicio s, tiene t
        where s.id_servicio = t.id_servicio
        and t.nro_reserva = :new.nro_reserva

        fecha_llegada := select s.fecha_salida
        from servicio s, tiene t
        where s.id_servicio = t.id_servicio
        and t.nro_reserva = :new.nro_reserva

        aeropuerto_llegada := select a.id_a
    from servicio s, tiene t, vuelvo v, aeropuerto a
    where s.id_servicio = t.id_servicio
    and s.nro_vuelo = v.nro_vuelo
    and v.llega_a = a.id_a

        user := select u.id_user
        from usuario u, hace h
        where h.id_user = u.id_user
        and h.nro_reserva = .new.nro_reserva

        if data_sub(fecha_salida, now()) < 7
        and exists(
          select *
          from reserva r, tiene t, servicio s,
                       hace h, usuario u, estado e
          where t.nro_reserva = r.nro_reserva
          and t.id_servicio = s.id_servicio
          and h.nro_reserva = r.nro_reserva
          and h.id_user = u.id_user
          and r.codigo_estado = e.codigo_estado
          and e.codigo_estado = 2 /* Confirmada */
          /* Hasta aca tenemos las reservas del usuario */
          and (not
                (s.fecha_salida >= fecha_llegada
                        or s.fecha_llegada <= fecha_salida))
        )
```

```
        begin
          raiserror ('Se superpone la reserva con
                                        una ya realizada.', 5, 1);
          rollback transaction;
          return
        end;
end;
```

Consulta 4

```
delete from reserva r1
where
exists
(
        select *
        from reserva r2, tiene t, servicio s, hace h,
                        usuario u, vuelvo v, brinda b
        where t.nro_reserva = r2.nro_reserva
        and t.id_servicio = s.id_servicio
        and h.nro_reserva = r2.nro_reserva
        and h.id_user = u.id_user
        and s.asociadoA = v.nro_vuel
        and b.id_servicio = s.id_servicio
        /* Hasta aca tenemos las reservas del usuario */

        /* Iguales fechas */
        and s.fecha_salida =
                (select s2.fecha_salida
                from servicio s2, tiene t2
                where s2.id_servicio = t2.id_servicio
                and t2.nro_reserva = r1.nro_reserva)

        and s.fecha_llega =
                (select s2.fecha_llegada
                from servicio s2, tiene t2
                where s2.id_servicio = t2.id_servicio
                and t2.nro_reserva = r1.nro_reserva)

        /* Iguales aeropuerto */
        and v.saleDe =
                (select v2.saleDe
                from servicio s2, tiene t2, vuelvo v2
                where s2.id_servicio = t2.id_servicio
                and t2.nro_reserva = r1.nro_reserva
                and s2.nro_vuelo = v2.nro_vuelo)

        and v.llegaA =
                (select v2.llegaA
                from servicio s2, tiene t2, vuelvo v2
                where s2.id_servicio = t2.id_servicio
                and t2.nro_reserva = r1.nro_reserva
```

```
                and s2.nro_vuelo = v2.nro_vuelo)

        /* Fecha partida dentro de los proximos 7 dias */
        and data_sub(s.fecha_salida, now()) < 7

        /* Menor precio */
        and sum(b.precio) < sum(
                select b2.precio
                from servicio s2, brinda b2, tiene t2
                where s2.id_servicio = t2.id_servicio
                and t2.nro_reserva = r1.nro_reserva
                and b2.id_servicio = s2.id_servicio)
)
```

4.1. Otras Aclaraciones

- Cuando se pide la cantidad de personas que pasaron por un aeropuerto, si una persona realiza más de un viaje a un mismo aeropuerto en el periodo especificado, se cuenta como personas distintas. Es decir, que por cada vez que una misma persona pasa por un aeropuerto incrementa en uno la cantidad total de personas que pasan por el aeropuerto.