DEPARTAMENTO
DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico 1

17 de Julio 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

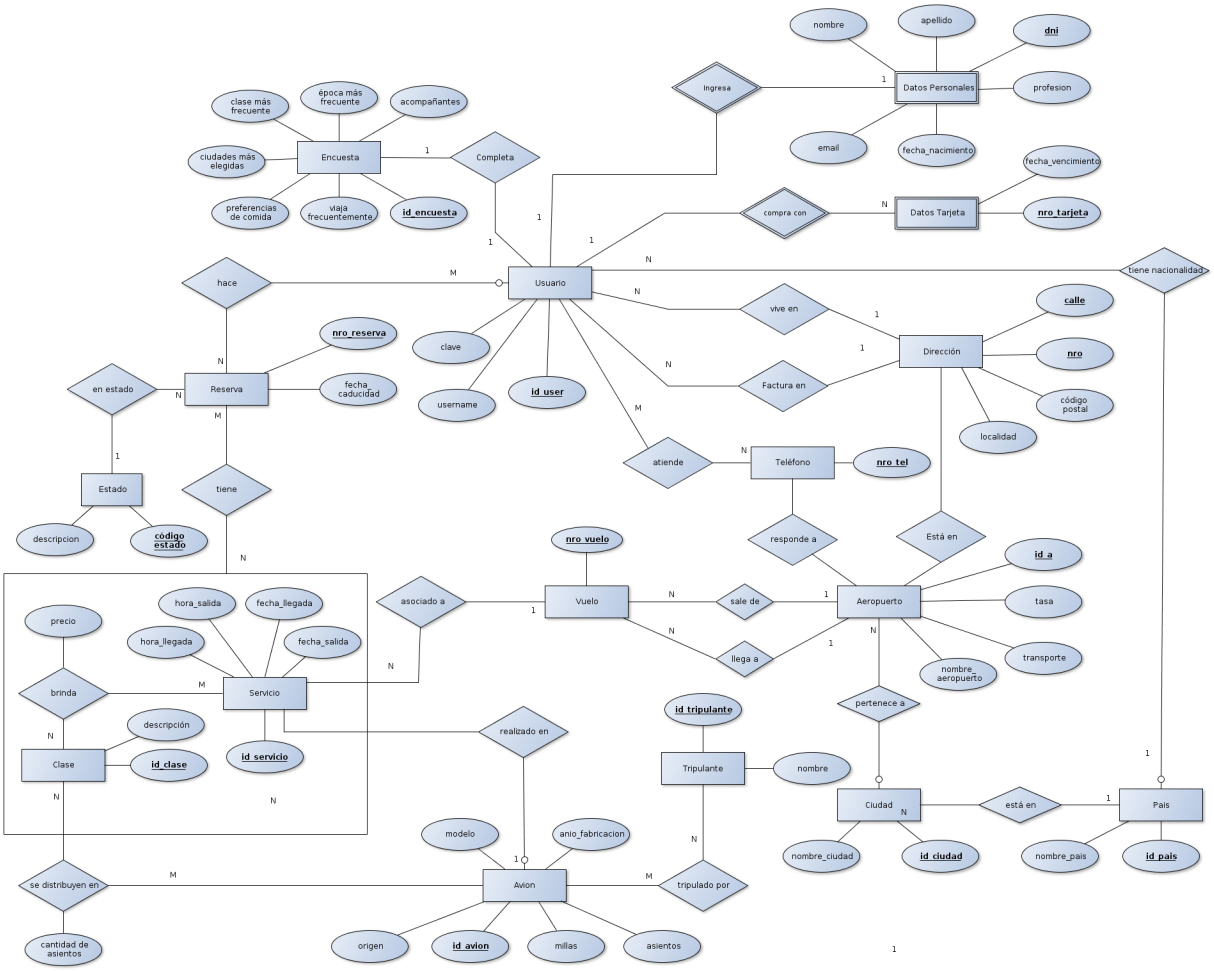
1. Introducción

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 Jueves".
- 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 servicio. Con lo cual todo vuelo existente es efectuado.
- Para todos los vuelos, el aeropuerto de salida y el de llegada son diferentes.
- Todos los servicios tienen la misma duración.

3. Modelo Relacional

primary key	foreign key

Usuario(id_user, username, clave, nacionalidad)

- $FK = \{nacionalidad\}$
- $PK = CK = \{id\_user\}$
- $Usuario.nacionalidad$ debe estar en $Pais.id\_pais$

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

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

DatosPersonales(id_user, nombre, apellido, 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, vencimiento)

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

Telefono(nro_tel)

- $FK = \{\}$
- $PK = CK = \{nro\_tel\}$

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 $Telefono.nro\_tel$

Responde(id_a, nro_tel)

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

Direccion(calle, nro, localidad, código postal)

- $FK = \{\}$
- $PK = CK = \{(calle, numero)\}$

Vive(id_user, calle, nro)

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

Factura(id_user, calle, nro)

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

Está(id_a, calle, nro)

- $FK = \{id\_a, calle, nro\}$
- $PK = CK = \{(id\_a, calle, nro)\}$
- $Vive.id\_a$ debe estar en $Aeropuerto.id\_a$
- $Vive.(calle, nro)$ debe estar en $Direccion.(calle, nro)$

Reserva(id_reserva, caducidad, id_estado)

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

Hace(id_reserva, id_user)

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

Estado(id_estado, descripcion)

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

Tiene(id_res, id_serv, id_clase)

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

Servicio(id_servicio, salida, llegada, id_vuelo, id_av)

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

Brinda(id_serv, id_clase, precio)

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

Clase(id_clase, descripcion)

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

Vuelo(id_vuelo, a_salida, a_llegada)

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

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

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

Distribuido(id_av, id_clase, asientos)

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

Tripulante(id_trip, nombre)

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

Tripulado(id_trip, id_av)

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

Pais(id_pais, nombre)

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

Ciudad(id_ciudad, nombre, id_pais)

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

Aeropuerto(id_a, tasa, transporte, nombre, id_ciudad)

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

4. Consultas

A continuación se presentan las consultas implementadas en SQL.

Consulta 1

Mediante SQL escribir una consulta para obtener el nombre, apellido y número identificatorio de aquellos pasajeros que han viajado a todos los países cubiertos por la línea aérea en los últimos 5 años.

```
SELECT dp.nombre, dp.apellido, u.id_user
FROM Usuario u, DatosPersonales dp
WHERE NOT EXISTS
(
    SELECT p.id_pais, p.nombre
    FROM Vuelo v1, Aeropuerto a1, Ciudad c1, Pais p
    WHERE (v1.a_salida = a1.id_a or v1.a_llegada = a1.id_a)
    AND a1.id_ciudad = c1.id_ciudad
    AND c1.id_pais = p.id_pais
    AND NOT EXISTS
    (
        SELECT *
        FROM Hace h, Reserva r, Tiene t, Servicio s, Vuelo v,
          Aeropuerto a, Ciudad c, Estado e
        WHERE u.id_user = h.id_user
        AND h.id_reserva = r.id_reserva
        AND r.id_reserva = t.id_res
        AND t.id_serv = s.id_servicio
        AND s.id_vuelo = v.id_vuelo
        AND (v.a_llegada = a.id_a or v.a_salida = a.id_a)
        AND a.id_ciudad = c.id_ciudad
        AND c.id_pais = p.id_pais
        AND r.id_estado = 3
        AND ((s.salida >= date_sub(now(), interval 5 year))
          OR (s.llegada >= date_sub(now(), interval 5 year)))
    )
)
AND u.id_user = dp.id_user
```

Consulta 2

Obtener un reporte que como mínimo contenga: Todos los códigos identificatorios de los aeropuertos, un periodo de tiempo de la forma año/mes, la cantidad de pasajeros que ascendieron y descendieron en ese aeropuerto durante el periodo. El reporte debe estar ordenado por la cantidad total de personas que viajaron. Y debe ser ejecutado para un rango de fechas. Aclaración: dependiendo del rango de fechas, para un mismo aeropuerto pueden aparecer varios periodos distintos.

Consulta 3

Controlar mediante alguna restricción que un usuario no pueda realizar reservas que se superpongan en el tiempo. La única excepción que se permite consiste en que un usuario puede realizar a lo sumo 2 reservas para la misma fecha de viaje entre los mismos aeropuertos de origen y destino, siempre que la fecha de partida no se encuentre dentro de los próximos 7 días.

```
CREATE TRIGGER chequear_reservas_superpuestas
BEFORE INSERT ON Tiene
FOR EACH ROW
begin
  IF EXISTS (
    SELECT h1.id_reserva, h1.id_user
    FROM Hace h1, Servicio s1, Usuario u1, Reserva r1
    WHERE h1.id_user = u1.id_user AND
      h1.id_reserva = r1.id_reserva AND
      NEW.id_res = r1.id_reserva AND
      NEW.id_serv = s1.id_servicio
      AND EXISTS(
          SELECT s2.salida
          FROM Servicio s2, Tiene t2, Reserva r2, Hace h2
          WHERE h2.id_user = u1.id_user
        AND h2.id_reserva <> h1.id_reserva
        AND h2.id_reserva = r2.id_reserva
        AND t2.id_res = r2.id_reserva
        AND t2.id_serv = s2.id_servicio
        AND s2.salida = s1.salida
        AND s1.id_servicio <> s2.id_servicio
        AND s2.salida NOT IN (
              (SELECT s3.salida
              FROM Reserva r3, Hace h3, Tiene t3, Servicio s3, Usuario u3, Vuelo v3
              WHERE r3.id_reserva = h3.id_reserva
                AND r3.id_reserva = t3.id_res
                AND u3.id_user = h3.id_user
                AND t3.id_serv = s3.id_servicio
                AND s3.id_vuelo = v3.id_vuelo
                AND s3.salida IN (
                    SELECT s4.salida
                    FROM Servicio s4, Hace h4, Tiene t4, Reserva r4, Vuelo v4
                    WHERE r4.id_reserva <> r3.id_reserva
                      AND h4.id_reserva = r4.id_reserva
                      AND h4.id_user = h3.id_user
                      AND r4.id_reserva = t4.id_res
                      AND t4.id_serv = s4.id_servicio
                      AND s4.id_vuelo = v4.id_vuelo
```

```
                    AND v4.a_salida = v3.a_salida
                    AND v4.a_llegada = v4.a_llegada
                    AND datediff(s4.salida, now()) > 7)
                  )
                )
  )) THEN
    SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = 'No se puede insertar la reserva';
  END IF;
end;
```

Para poder imponer esta restricción, usamos un trigger, que cuando ocurre el evento: insertar en la tabla Tiene (tabla que relaciona las reservas con los servicios), se fija que la reserva a la cual está asociada esta tabla no tenga como fecha de salida de vuelo la misma fecha de salida de otras reservas del mismo usuario, salvo que la fecha de partida del vuelo no sea dentro de la próxima semana. En caso de que tengta la misma fecha de salida se dispara una excepción inexistente de manera tal que no se concreta la operación.

Consulta 4

Para las reservas superpuestas que se permiten en el punto anterior, se debe contar con una funcionalidad que cancele una de las reservas duplicadas (la más económica) cuando la fecha de partida se encuentre dentro de los próximos 7 días.

```
delete
from Reserva r1
where r1.id_reserva IN (
  select r2.id_reserva
  from Reserva r2, Tiene t, Servicio s, Hace h, Usuario u, Vuelo v, Brinda b
  where t.id_res = r2.id_reserva
  and t.id_serv = s.id_servicio
        and h.id_reserva = r2.id_reserva
        and h.id_user = u.id_user
        and s.id_vuelo = v.id_vuelo
        and s.salida IN (
                select s2.salida
                from Servicio s2, Tiene t2
                where s2.id_servicio = t2.id_serv
                and t2.id_res = r1.id_reserva)

        and s.llegada IN (
                select s3.llegada
                from Servicio s3, Tiene t3
                where s3.id_servicio = t3.id_serv
                and t3.id_res = r1.id_reserva)
        and v.a_salida IN (
                select v4.a_salida
                from Servicio s4, Tiene t4, Vuelo v4
                where s4.id_servicio = t4.id_serv
                and t4.id_res = r1.id_reserva
                and s4.id_vuelo = v4.id_vuelo)

  and v.a_llegada IN (
    select v5.a_llegada
    from Servicio s5, Tiene t5, Vuelo v5
    where s5.id_servicio = t5.id_serv
    and t5.id_res = r1.id_reserva
    and s5.id_vuelo = v5.id_vuelo)
  group by b.precio
  having min(b.precio)
)
```

Consulta 5

Restricción adicional: Restricción que no permite modificar el estado de una reserva a realizada si la fecha de la reserva es posterior a la fecha actual

```
create trigger chequear_fechas_correctas
BEFORE UPDATE ON Reserva
FOR EACH ROW
begin
    IF NEW.id_estado = 3 THEN
        IF EXISTS (
            SELECT 1
            FROM Servicio s, Tiene t
            WHERE
                s.id_servicio = t.id_serv AND
                t.id_res = r.id_reserva AND
                (datediff(now(), s.salida) < 0) AND
                (datediff(now(), s.llegada) < 0)
        ) THEN
            SIGNAL SQLSTATE '45000'
                SET MESSAGE_TEXT =
                    'No se puede cambiar el  estado de la reserva';
        END IF;
    END IF;
end;
```

El estado

5. Tablas Existentes

1. Usuario
2. Encuesta
3. DatosPersonales
4. DatosTarjeta
5. Telefono
6. Atiende
7. Responde
8. Direccion
9. Vive
10. Factura
11. Está
12. Reserva
13. Hace
14. Estado
15. Tiene
16. Servicio
17. Brinda
18. Clase
19. Vuelo
20. Avion
21. Distribuido
22. Tripulante
23. Tripulado
24. Pais
25. Ciudad
26. Aeropuerto