Junio 2012

1. Crear Tabla

```
CREATE TABLE partido (
        dnimLocal varchar(9) primary key references pareja(dnim),
        dimVisit varchar(9) references pareja(dnimh),
        Set_Local number(1) check (Set_Local between 0 and 3),
        Set_Visitante number(1) check (Set_Visitante between 0 and 3),
        Fecha Date,
        check (dnimLocal != dnimVisit)
);
```

2. Crea un indice sobre el campo fecha de la tabla partido

```
CREATE INDEX fecha_partido on partido (Fecha);
```

3. Crea una vista que muestre el nombre y la edad (en años) de cada jugadora. Tengase en cuenta que la diferencia entre dos fechas devuelve el resultado en días

```
CREATE VIEW jugadora_edad AS
    SELECT nombreM, (SYSDATE - fecha_nacimientoM)/365.25
    FROM jugadora;
```

4. SQL "Mostrar el nombre de la jugadora más joven"

```
SELECT nombreM from jugadora Where Fecha_nacimientoM (
        SELECT max(Fec_nacimiento M) from jugadora
);

SELECT j1.nombreM from jugadora j1 where Not exists (
        Select * from jugadora j2
        Where j2.Fec_nacimiento M > j1.FechanacimientoM );
```

5. Muestra el dni de la componente femenina de la pareja que ha ganado más partidos como visitante.

```
SELECT dnimVisit From partido
Where Sets-Local < Sets-Visitante
GROUP BY dnimVisit HAVING count(*) =
    (SELECT max(count(*)) FROM partido
     Where Set_Local < Sets_Visitante
     GROUP BY dnimVisit);
```

Julio 2014

1. Cálculo Relacional

"Encontrar los clientes que han reservado en el hotel de nombre 'Estrella de mar' pero que no han solicitado ninguna actividad

$\{ R.dni \mid reserva(R) \text{ and } (\exists H)(Hotel(H) \text{ and } H.nombre = \text{'Estrella de Mar'} \text{ and } H.hotH = R.hotH \text{ and } \neg(\exists A)(Actividad(A) \text{ and } A.dni = R.dni)) \}$

2. Álgebra Relacional

"Encontrar los DNI de los clientes que han pedido todas las actividades de tipo infantil que se ofertan para al menos 10 personas.

$\Pi_{dni, actH}(\sigma_{tipo-act = infantil}(Actividad)) \div \Pi_{actH}(\sigma_{nmax >= 10}(oferta\text{-}actividad))$

3. SQL El hotel 'Estrella de mar' quiere saber la wifi de todas las habitaciones que estén disponibles

a) Crear una vista de usuario

```
CREATE VIEW Disponible AS
    SELECT ho.nombre_h, ha.hotH, num_hab, tipo, n_camas, precio,
    disponible
    FROM hoteles ho, habitaciones ha
    WHERE ho.nombre_h = 'Estrella de Mar' And ho.hotH = ha.hotH
    and ha.disponible = 'si'
```

b) Actualizar a 'disponibles' las habitaciones que hayan quedado libres

```
UPDATE Disponible set disponible = 'si'
where hotH, num_habitacion IN
    (SELECT r.hotel, r.num_hab
     FROM hoteles ho, reserva r
     WHERE ho.nombre_h = 'Estrella de Mar' And ho.hotH = r.hotH
    and r.fecha_llegada > sysdate OR r.fecha_salida < sysdate
    )
```

b) El hotel 'Estrella de Mar' quiere saber para cada día la cantidad de personas apuntadas a cada actividad durante el mes de Agosto del 2014. Dar la consulta

```
SELECT p.fecha, p.actH, a.descripcion, sum(p.num_personas)
FROM hoteles h, pde_actividad p, actividad a
WHERE h.nombre_h = 'I' and h.hotH = p.hotH and p.actH =
a.actH and GROUP BY fecha, p.actH, a.description.
```

• Corredores

```
CREATE TABLE Tiempos (
        No-corre  INT  CHECK (No_corre between 0 and 9999)
        No-etapa  INT  CHECK (No_etapa between 0 and 30)
        Cod-pru   VARCHAR (30)
        año       INT  CHECK (año >= 1900)
        Tiempo    INT  Not-Null'

        FOREING KEY (Cod-pru, año, No-etapa) REFERENCE Etapa(——)
        PRIMARY KEY (No-corre, Cod-pru, año, No-etapa
)
```

REFERENCE / Corredores(No-corre)

Insert Into Tiempos Values (. . , , ,)

Muestra la etapa más corta

SQL// SELECT min(km) FROM Etapa

AR//

~~Π_{km} Etapa ($\mathcal{G}_{min(km)}$)~~

Π_{etapas} (Π_{E2} ($\mathcal{G}_{E2.km < Etapa.km}$ (Etapas × E2)))

```
Select * FROM Etapa WHERE
Km <= ALL (
Select E2.km
    FROM Etapas E2
)
```

CRT//

$\{ E \mid Etapas(E)$ and $E.km = min(Etapas.km) \}$

Muestra el nombre de los corredores que han realizado todas las etapas de la prueba 'Giro de Italia' del 2015

SQL//

```
SELECT nom-corre FROM Corredores WHERE NOT EXISTS (
    SELECT * FROM Etapa WHERE NOT EXISTS (
        SELECT * FROM Prueba WHERE
        Prueb. Cod-Prueba = Etapa. Cod-prueba and
        Prueba. año = Etapa. año 2015 and
        Corredor. pas
```

• Libros

```
CREATE TABLE Copia (
        Cop VARCHAR (10)
        L VARCHAR (10) REFERENCES Libro (L),
        PRIMARY KEY (Cop, L)

CREATE TABLE Prestado (
        Cop VARCHAR (10),
        L VARCHAR (10),
        fecha Date check ( fecha between to_date ('01/01/2012')
and to date ('31/02/2012')),
        DNI VARCHAR (8) NOT NULL REFERENCE Usuario (DNI),
        PRIMARY KEY (Cop, L, Fecha)
        UNIQUE ( Fecha, DNI)
        FOREING KEY ( Cop, L) REFERENCES Copia (Cop, L)
)
```

1. Encontrar los nombres de los usuarios que han tomado prestados todos los libros de la biblioteca

SQL// SELECT nombreU FROM Usuario WHERE NOT EXIST (
SELECT * FROM Prestados WHERE
Prestados.DNI = Usuario.DNI
)

CRT// { N.nombreU | Usuario (N) and Not (Exist P) (Prestados (P)
and P.DNI = U.DNI) }

AR// $\pi_{nombreU}((\pi_{DNI,Cop}(Prestado) \div \pi_{Cop}(Libro)) \bowtie Usuario)$

// igual que examen

Mostrar el título del 1º libro que se prestó P prestado = PR

SQL//

```
Select Titulo FROM Libro, Prestado WHERE (
        Libro.L = Prestado.L
        AND  Fecha IN SELECT MIN ( PR.Fecha
    FROM Prestado PR )
```

AR//

$$\Pi_{Titulo}\left(\left(Prestado - \Pi_{PR}\left(\sigma_{Prestado.Fecha < A.fec}\left(Prestado \times PR\right)\right) \bowtie LIBRO\right)\right)$$

CRT//

$$\{ T.titulo \mid Libro(T) \text{ and } Not\ (Exists\ PR)\ (Prestado(PR) \text{ and } TL = PR.L \text{ and } Not\ (Exists\ PR2)\ (Prestado(PR2) \text{ and } PR.Fecha < PR2.Fecha))\}$$

Crear Vista

```
CREATE VIEW AS (
    SELECT DNI, titulo, count(*) FROM Prestado, Libro
                                 WHERE Prestado.L = Libro.L
    GROUP BY (DNI, titulo)
    ORDER BY (DNI, titulo)
    HAVING ( count(*) > 1)
)
```