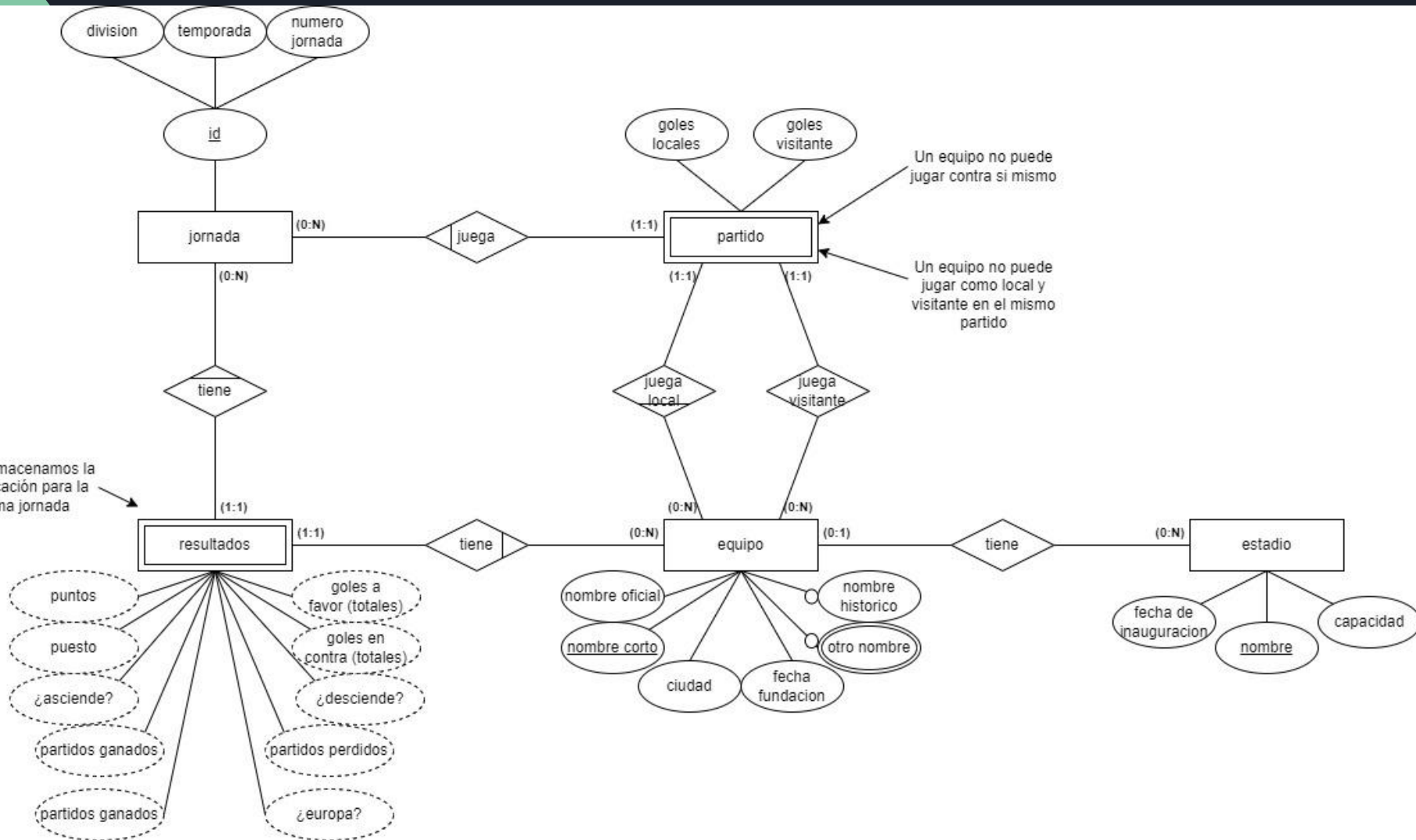




BBDD - Práctica 1

Héctor Toral
Francisco Javier Pizarro
Pablo López

03-04-2022





Alternativas a nuestro modelo:

- Planteamos representar jornada como varias entidades relacionadas entre sí
- Planteamos la posibilidad de representar la relación entre equipo y partido como una 2:N

jornadas	
division	VARCHAR2(30)
temporada	NUMBER
numJornada	NUMBER
division, temporada, numJornada	
division, temporada, numJornada	

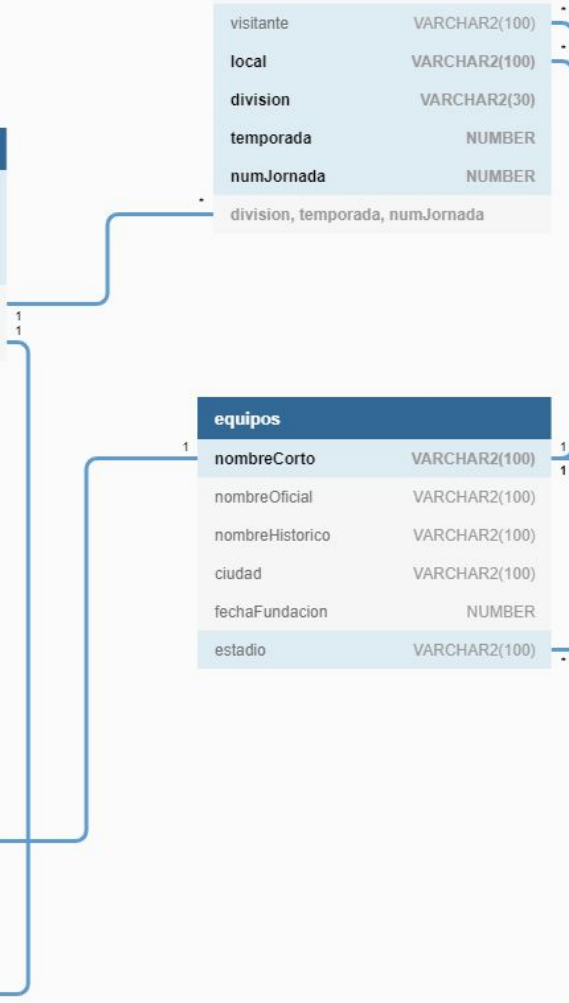
resultados	
puntos	NUMBER
puesto	NUMBER
golesAF	NUMBER
golesEC	NUMBER
partidosGanados	NUMBER
partidosEmpatados	NUMBER
partidosPerdidos	NUMBER
asciende	NUMBER
desciende	NUMBER
europa	NUMBER
equipo	VARCHAR2(100)
division	VARCHAR2(30)
temporada	NUMBER
numJornada	NUMBER
division, temporada, numJornada	

partidos	
golesLocal	NUMBER
golesVisitante	NUMBER
visitante	VARCHAR2(100)
local	VARCHAR2(100)
division	VARCHAR2(30)
temporada	NUMBER
numJornada	NUMBER
division, temporada, numJornada	

equipos	
nombreCorto	VARCHAR2(100)
nombreOficial	VARCHAR2(100)
nombreHistorico	VARCHAR2(100)
ciudad	VARCHAR2(100)
fechaFundacion	NUMBER
estadio	VARCHAR2(100)

otrosNombres	
nombre	VARCHAR2(100)
equipo	VARCHAR2(100)

estadios	
nombre	VARCHAR2(100)
capacidad	NUMBER
fechaInauguracion	VARCHAR2(100)





No ha habido dudas pasando del modelo E/R al modelo relacional.

Equipo(s) que han estado en primera división un mínimo de cinco temporadas y que no han ganado ninguna liga.

```
SELECT DISTINCT equipo
FROM resultados
WHERE equipo NOT IN (
    -- Equipos que han ganado al menos una liga
    SELECT DISTINCT equipo
    FROM resultados
    WHERE puesto = 1
) AND equipo IN (
    -- Equipos que han estado en primera al menos 5 temporadas
    SELECT equipo
    FROM (
        SELECT count(*) AS veces, equipo
        FROM resultados
        WHERE division = '1ª'
        GROUP BY equipo
    ) A
    WHERE A.veces >= 5
)
ORDER BY equipo;
```

Temporada(s) en las que el ganador de segunda división ha ganado más partidos que el ganador de primera división.

```
SELECT T1.temporada
FROM resultados T1
WHERE T1.division = '2ª' AND T1.puesto = 1
    AND EXISTS (
        -- Devuelve una "tupla"/"flag" cuando se cumple que el ganador
        -- de segunda
        -- ha ganado más partidos que el de primera / temporada
        SELECT 1
        FROM resultados T2
        WHERE T2.division = '1ª' AND T2.puesto = 1
        AND T1.partidosganados > T2.partidosganados
        AND T2.temporada = T1.temporada
    )
ORDER BY T1.temporada;-
```

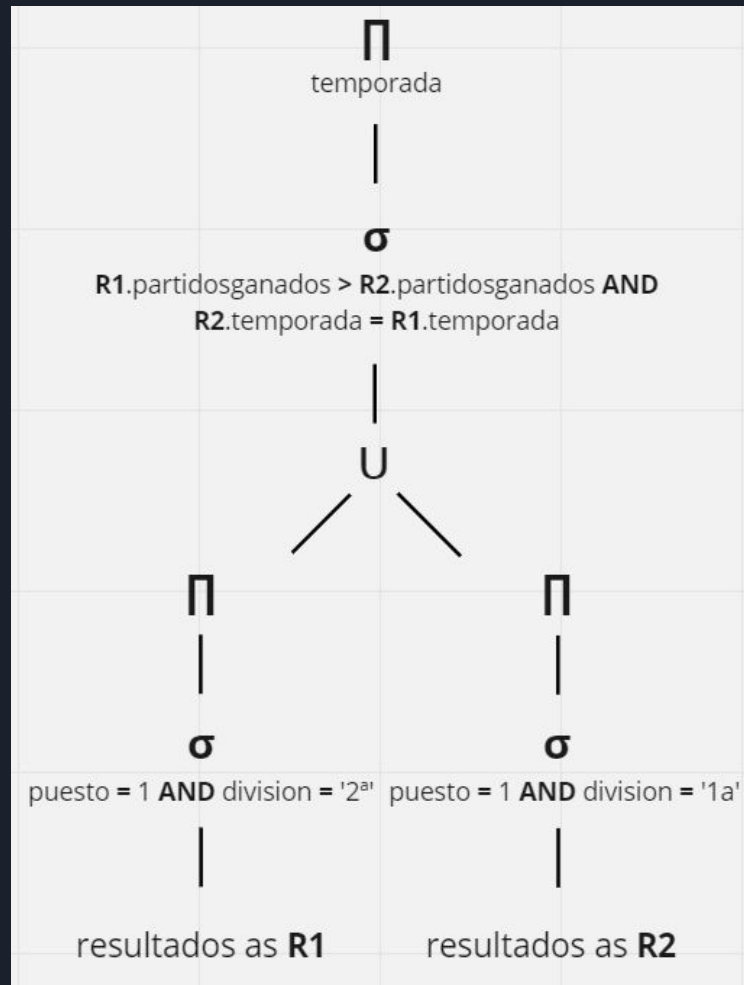
Equipo(s) y temporada(s) donde dicho equipo ha ganado dicha temporada, habiendo perdido todos los partidos contra el equipo que quedó en segunda posición.

```
SELECT C.equipo, C.temporada
FROM (
    SELECT COUNT(*) AS veces, R.temporada, R.division
    FROM partidos P, (
        -- Devuelve pares de ganador, subcampeon
        SELECT DISTINCT A.puesto AS puesto_1, A.equipo AS campeon, B.puesto AS puesto_2, B.equipo AS subcampeon,
                        A.temporada, A.division, A.numJornada
        FROM resultados A, resultados B
        WHERE A.puesto = 1 AND B.puesto = 2 AND A.division = B.division AND A.temporada = B.temporada
    ) R
    WHERE ((P.local = R.campeon AND P.visitante = R.subcampeon) OR (P.local = R.subcampeon AND
        P.visitante = R.campeon))
        AND P.division = R.division AND P.temporada = R.temporada
    GROUP BY R.temporada, R.division
) A, (
    SELECT COUNT(*) AS veces, R.temporada, R.division
    FROM partidos P, (
        -- Devuelve pares de ganador, subcampeon
        SELECT DISTINCT A.puesto AS puesto_1, A.equipo AS campeon, B.puesto AS puesto_2, B.equipo AS subcampeon,
                        A.temporada, A.division, A.numJornada
        FROM resultados A, resultados B
        WHERE A.puesto = 1 AND B.puesto = 2 AND A.division = B.division AND A.temporada = B.temporada
    ) R
    WHERE ((P.local = R.campeon AND P.visitante = R.subcampeon AND P.goleslocal < P.golesvisitante) OR
        (P.local = R.subcampeon AND P.visitante = R.campeon AND P.goleslocal > P.golesvisitante)) AND
        P.division = R.division AND P.temporada = R.temporada
    GROUP BY R.temporada, R.division
) B, resultados C
WHERE A.veces = B.veces AND A.temporada = B.temporada AND A.division = B.division AND A.temporada = C.temporada AND
    A.division = C.division AND puesto = 1
ORDER BY A.temporada;
```

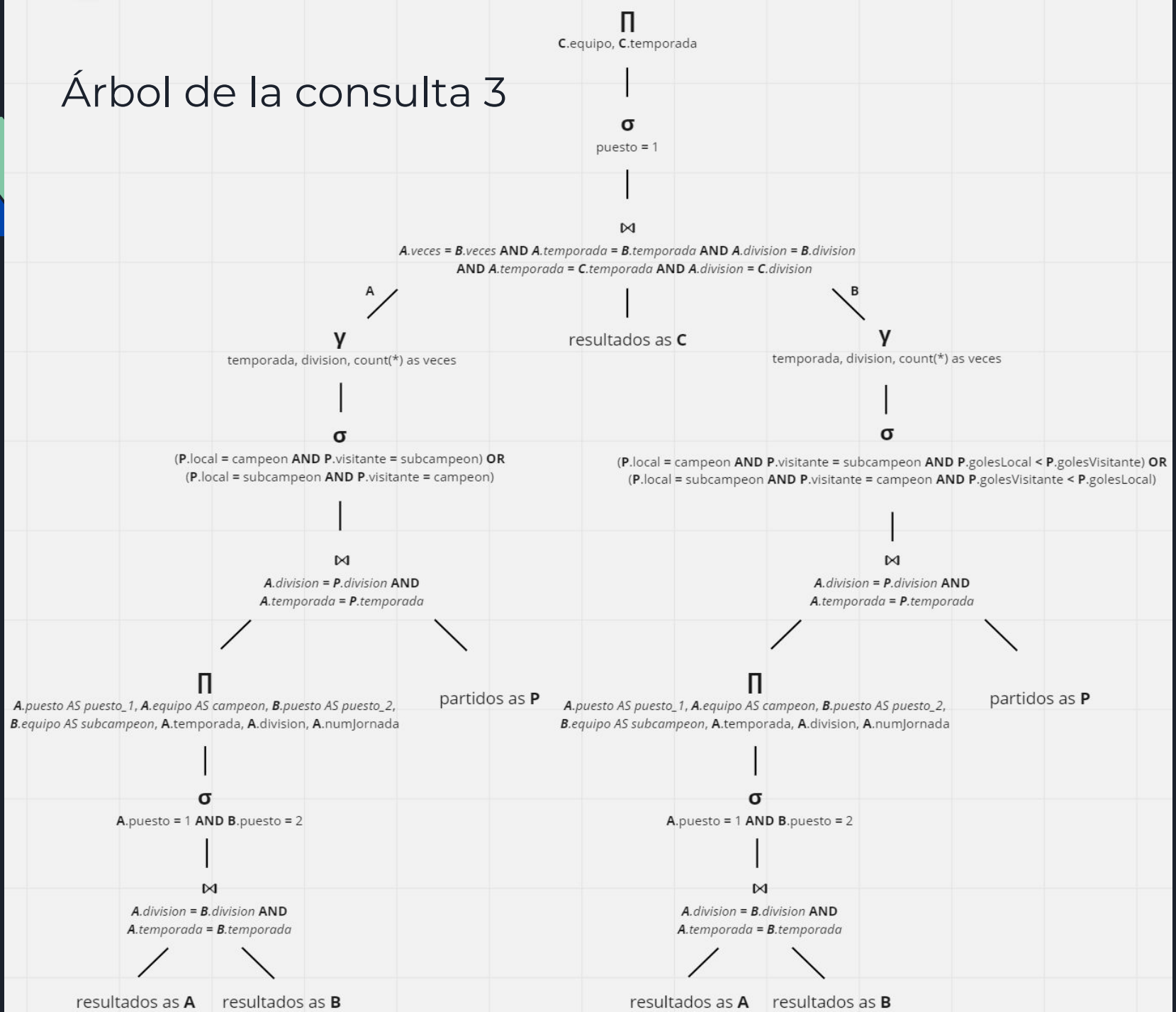

Árbol de la consulta 1



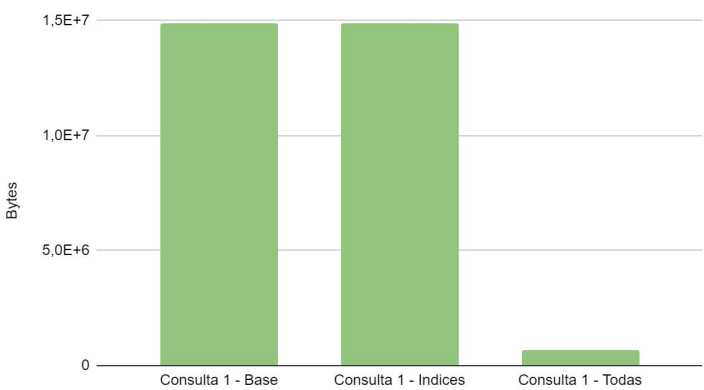
Árbol de la consulta 2



Árbol de la consulta 3



Bytes de memoria Consulta 1



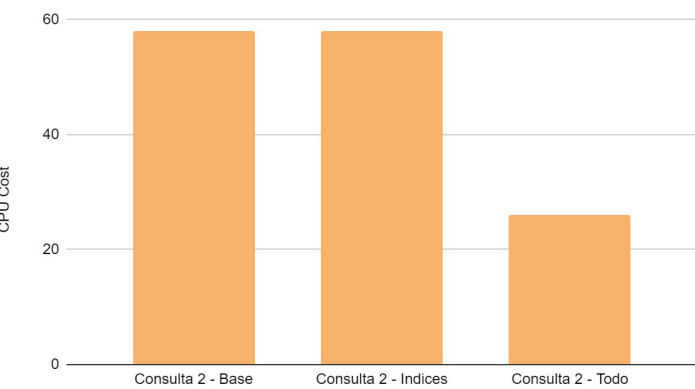
CPU Cost Consulta 1



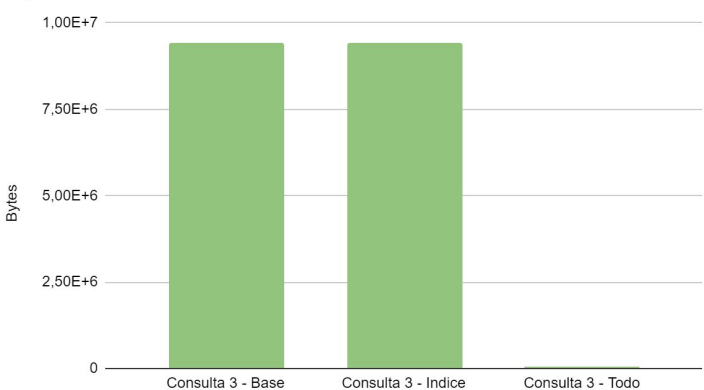
Bytes de Memoria Consulta 2



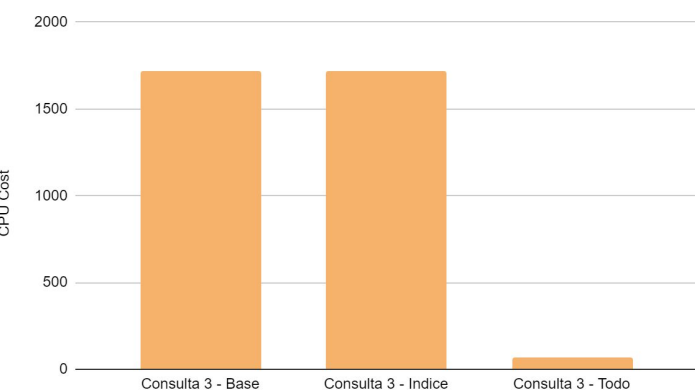
CPU Cost Consulta 2




Bytes de memoria Consulta 3



CPU Cost Consulta 3



1 Nada
2 Indices
3 Vistas+ partició



Restricción: Que la división introducida esté dentro de las permitidas.


```
-- TRIGGER 1:
CREATE OR REPLACE TRIGGER CHECK_INTEGRIDAD
BEFORE INSERT ON partidos
FOR EACH ROW
DECLARE
    flag NUMBER;
BEGIN
    SELECT COUNT(*) INTO flag
    FROM partidos
    WHERE temporada = :NEW.temporada AND division = :NEW.division AND
        numJornada = :NEW.numJornada AND
        (local = :NEW.visitante OR visitante = :NEW.visitante);

    IF flag >= 1 THEN
        RAISE_APPLICATION_ERROR (-20000,
            'Un equipo no puede jugar como local y visitante en la misma jornada.');
```

END IF;

END;

/



Restricción: Al introducir el resultado de un equipo para una temporada, dicho equipo debía existir en dicha temporada

```
-- TRIGGER 2:
CREATE OR REPLACE TRIGGER DATES_OK
BEFORE INSERT ON resultados
FOR EACH ROW
DECLARE
    fechaFUN NUMBER;
BEGIN
    SELECT fechaFundacion INTO fechaFUN
    FROM equipos
    WHERE :NEW.equipo = nombreCorto;

    IF :NEW.temporada < fechaFUN
    THEN
        RAISE_APPLICATION_ERROR (-20003, 'El equipo no existe aún en esta temporada');
    END IF;
END;
/
```

```

CREATE OR REPLACE TRIGGER UPT_RESULTADOS FOR INSERT ON partidos COMPOUND TRIGGER
    filaLocal resultados%ROWTYPE; filaVisitante resultados%ROWTYPE; puntosLocal NUMBER; puntosVisitante NUMBER; ganaLocal NUMBER; ganaVisitante NUMBER;
    empate NUMBER; existeParLocA NUMBER; existeParVisA NUMBER; existeResLoc NUMBER; existeResVis NUMBER;

    BEFORE EACH ROW IS BEGIN -- Garantiza una inserción de los partidos en orden -- El partido que juega local en la jornada anterior no existe
        SELECT COUNT(*) INTO existeParLocA FROM partidos
        WHERE (local = :NEW.local OR visitante = :NEW.local) AND temporada = :NEW.temporada AND division = :NEW.division AND numJornada = :NEW.numJornada - 1;
        SELECT COUNT(*) INTO existeParVisA FROM partidos
        WHERE (local = :NEW.visitante OR visitante = :NEW.visitante) AND temporada = :NEW.temporada AND division = :NEW.division AND
            numJornada = :NEW.numJornada - 1; -- El partido que juega visitante en la jornada anterior no existe
        IF :NEW.numJornada <> 1 AND NOT (1 <= existeParLocA) OR NOT (1 <= existeParVisA) THEN RAISE_APPLICATION_ERROR (-20003,
            'Esta tupla esta siendo insertada antes de lo que debería'); END IF; END BEFORE EACH ROW;

    -- Actualiza la tabla de resultados AFTER EACH ROW IS BEGIN -- captura los resultados antiguos de los 2 equipos insertados
        SELECT * INTO filaLocal FROM resultados WHERE equipo = :NEW.local AND temporada = :NEW.temporada AND division = :NEW.division;
        SELECT * INTO filaVisitante FROM resultados WHERE equipo = :NEW.visitante AND temporada = :NEW.temporada AND division = :NEW.division;
        -- Establece los puntos nuevos: Gana el local | visitante | empatan
        IF :NEW.golesLocal > :NEW.golesVisitante THEN puntosLocal := 3; puntosVisitante := 0; ganaLocal := 1; ganaVisitante := 0; empate := 0;
        ELSIF :NEW.golesLocal < :NEW.golesVisitante THEN puntosLocal := 0; puntosVisitante := 3; ganaLocal := 0; ganaVisitante := 1; empate := 0;
        ELSE puntosLocal := 1; puntosVisitante := 1; ganaLocal := 0; ganaVisitante := 0; empate := 1; END IF;
        -- si existe un resultado para el equipo local
        SELECT COUNT(*) INTO existeResLoc FROM resultados WHERE equipo = :NEW.local AND temporada = :NEW.temporada AND division = :NEW.division;
        -- si existe un resultado para el equipo local
        SELECT COUNT(*) INTO existeResVis FROM resultados WHERE equipo = :NEW.visitante AND temporada = :NEW.temporada AND division = :NEW.division;
        IF (1 >= existeResLoc AND 1 >= existeResVis) THEN -- Actualiza los resultados del equipo local
            UPDATE RESULTADOS
            SET PUNTOS = filaLocal.puntos + puntosLocal, GOLESF = filaLocal.golesF + :NEW.golesLocal, GOLESEC = filaLocal.golesEC + :NEW.golesVisitante,
                PARTIDOSGANADOS = filaLocal.partidosGanados + ganaLocal, PARTIDOSEMPATADOS = filaLocal.partidosEmpatados + empate,
                PARTIDOSPERDIDOS = filaLocal.partidosPerdidos + ganaVisitante, NUMJORNADA = :NEW.numJornada
            WHERE EQUIPO = :NEW.local AND DIVISION = :NEW.division AND TEMPORADA = :NEW.temporada;
            -- Actualiza los resultados del equipo visitante
            UPDATE RESULTADOS SET PUNTOS = filaVisitante.puntos + puntosVisitante, GOLESF = filaVisitante.golesF + :NEW.golesVisitante,
                GOLESEC = filaVisitante.golesEC + :NEW.golesLocal, PARTIDOSGANADOS = filaVisitante.partidosGanados + ganaVisitante,
                PARTIDOSEMPATADOS = filaVisitante.partidosEmpatados + empate, PARTIDOSPERDIDOS = filaVisitante.partidosPerdidos + ganaLocal,
                NUMJORNADA = :NEW.numJornada WHERE EQUIPO = :NEW.visitante AND DIVISION = :NEW.division AND TEMPORADA = :NEW.temporada;
        ELSE -- Inserta el primer resultado del equipo local
            INSERT INTO RESULTADOS (PUNTOS, GOLESF, GOLESEC, PARTIDOSGANADOS, PARTIDOSEMPATADOS, PARTIDOSPERDIDOS, EQUIPO, DIVISION, TEMPORADA, NUMJORNADA)
            VALUES (puntosLocal, :NEW.golesLocal, :NEW.golesVisitante, ganaLocal, empate, ganaVisitante, :NEW.local, :NEW.division, :NEW.temporada, 1);
            -- Inserta el primer resultado del equipo visitante
            INSERT INTO RESULTADOS (PUNTOS, GOLESF, GOLESEC, PARTIDOSGANADOS, PARTIDOSEMPATADOS, PARTIDOSPERDIDOS, EQUIPO, DIVISION, TEMPORADA, NUMJORNADA)
            VALUES (puntosVisitante, :NEW.golesVisitante, :NEW.golesLocal, ganaVisitante, empate, ganaLocal, :NEW.visitante, :NEW.division, :NEW.temporada, 1);
        END IF; -- Zona para recalcular los puestos y ascensos -- Consulta que calcula los datos buenos a usar con un update para actualizar la tabla
        MERGE INTO resultados
        USING ( SELECT ROW_NUMBER() OVER (ORDER BY puntos DESC) AS puesto, R1.partidosGanados, R1.partidosEmpatados, R1.partidosPerdidos, R1.golesF, R1.golesEC,
            R1.puntos, R1.equipo, R1.division, R1.temporada, R1.numJornada, CASE WHEN puesto <= 3 AND division = '2ª' THEN 1 END AS asciende,
            CASE WHEN puesto >= 2.numEquipos - 1 AND division = '1ª' THEN 1 END AS desciende, CASE WHEN puesto <= 5 AND division = '1ª' THEN 1 END AS europa FROM resultados
            R1, ( SELECT COUNT(*) AS numEquipos FROM resultados WHERE temporada = :NEW.temporada AND division = :NEW.division ) R2 WHERE R1.temporada = :NEW.temporada AND
            R1.division = :NEW.division ORDER BY R1.division, R1.puesto ) N ON (resultados.equipo = N.equipo AND resultados.temporada = :NEW.temporada AND
            resultados.numJornada = :NEW.numJornada AND resultados.division = :NEW.division) WHEN MATCHED THEN UPDATE SET resultados.puesto = N.puesto, resultados.asciende =
            N.asciende, resultados.desciende = N.desciende, resultados.europa = N.europa;
    END AFTER EACH ROW; END UPT_RESULTADOS;

```