# Base de Datos (75.15 / 75.28 / 95.05)

Evaluación Integradora - 24 de julio de 2019

| **TEMA 20191C4** | | | | | |
|---|---|---|---|---|---|
| **DML** | | **Proc.** | | **CyT** | |
| **DR** | | **NoSQL** | | **NoSQL2** | |
| Corrigió: <br> **Nota:** | | | | | |

**Padrón:** _____

**Apellido:** _____

**Nombre:** _____

**Cantidad de hojas:** _____

☐ Aprobado ☐ Insuficiente

**Criterio de aprobación:** El examen está compuesto por 6 ítems, cada uno de los cuales se corrige como B/B-/Reg/Reg-/M. Se aprueba con nota mayor o igual a 4(cuatro), equivalente a desarrollar el 60 % del examen correctamente.

1. *(Lenguajes de manipulación de datos)* El operador de *antijoin* ($\triangleright$) es un operador del álgebra relacional cuyo resultado puede declararse en Cálculo Relacional de Tuplas de la siguiente manera: dadas dos relaciones $R(\bar{A})$ y $S(\bar{B})$ que poseen un conjunto de atributos en común $\bar{Y} = \bar{A} \cap \bar{B}$,

$$R \triangleright S = \{r | R(r) \wedge (\nexists s)(S(s) \wedge r[\bar{Y}] = s[\bar{Y}])\}$$

   *a*) Exprese el operador de *antijoin* en términos de los operadores $\pi, \sigma, \times, \cup, -, \cap, \bowtie, \div$.

   *b*) Considere ahora las siguientes relaciones que almacenan información sobre alumnos y materias de esta facultad, y las materias que cada alumno aprobó:

   - `Alumnos(padrón, apellido, nombre)`
   - `Aprobó(padrón, cod_dpto, cod_materia, fecha, calificación)`
   - `Materias(cod_dpto, cod_materia, nombre_materia)`

   A partir de la siguiente expresión del álgebra relacional que utiliza el operador de *antijoin*:

$$Alumnos \triangleright (Aprobó * \sigma_{cod\_dpto=62}(Materias))$$

   Se pide:

   *i)* Exprese en lenguaje coloquial el resultado de dicha consulta.

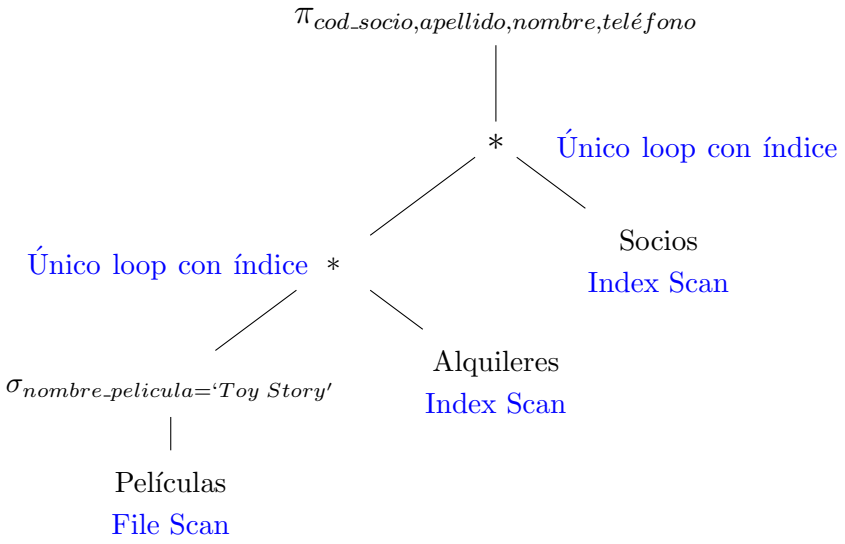
   *ii)* Traduzca la consulta al lenguaje SQL.

2. *(Procesamiento de Consultas)* Corre el año 1999 y Lucas es el dueño de un muy concurrido videoclub de barrio con un stock de 8000 cintas y 4000 alquileres por mes. Su sistema de administración del local utiliza una base de datos con las siguientes tablas:

   - `Películas(cod_película, nombre_película, año)`
   - `Cintas(cod_película, nro_ejemplar)`
   - `Socios(cod_socio, apellido, nombre, domicilio, teléfono)`
   - `Alquileres(fecha_alquiler, cod_película, nro_ejemplar, cod_socio, fecha_dev)`

   Adicionalmente, la tabla `Socios` posee un índice primario -y por ende está ordenada por su clave-, mientras que la tabla `Alquileres` posee un índice secundario de tipo árbol por el par de atributos {`cod_película, nro_ejemplar`}. Este es un índice compuesto que se puede acceder tanto por el par {`cod_película, nro_ejemplar`} como por el atributo `cod_película`.

   Lucas acaba de adquirir varias copias de la segunda parte de *Toy Story*, y le gustaría llamar por teléfono a todos los socios que habían alquilado la primera parte -que salió un año atrás- para avisarles que ya tiene disponible la nueva. Para ello, su SGBD ideó el siguiente plan de ejecución:




   Se pide:

   *a*) Estime el costo del plan de ejecución en términos de cantidad de accesos a disco, asumiendo que dispone de 100 bloques de memoria.

   *b*) Estime la cardinalidad del resultado en términos de cantidad de tuplas.

   A continuación se muestra la información de catálogo disponible sobre estas tablas:

| **CINTAS** | **PELÍCULAS** |
|---|---|
| n(Cintas) = 8.000 | n(Películas) = 4.000 |
| B(Cintas) = 800 | B(Películas) = 400 |
| V(cod_película, Cintas) = 4.000 | V(nombre_película, Películas) = 4.000 |

| **ALQUILERES** | **SOCIOS** |
|---|---|
| n(Alquileres) = 200.000 | n(Socios) = 3.000 |
| B(Alquileres) = 40.000 | B(Socios) = 300 |
| V(cod_película, Alquileres) = 4.000 | H(I(cod_socio, Socios)) = 2 |
| H(I({cod_película, nro_ejemplar}, Alquileres)) = 4 | |

3. *(Concurrencia y transacciones)* Dadas las siguientes tres transacciones:

$$T_1 : \mathtt{b}_{\mathtt{T}_1}; \mathtt{R}_{\mathtt{T}_1}(\mathtt{X}); \mathtt{R}_{\mathtt{T}_1}(\mathtt{Y}); \mathtt{W}_{\mathtt{T}_1}(\mathtt{Z}); \mathtt{c}_{\mathtt{T}_1};$$
$$T_2 : \mathtt{b}_{\mathtt{T}_2}; \mathtt{R}_{\mathtt{T}_2}(\mathtt{Y}); \mathtt{W}_{\mathtt{T}_2}(\mathtt{Y}); \mathtt{c}_{\mathtt{T}_2};$$
$$T_3 : \mathtt{b}_{\mathtt{T}_3}; \mathtt{R}_{\mathtt{T}_3}(\mathtt{Z}); \mathtt{R}_{\mathtt{T}_3}(\mathtt{Y}); \mathtt{W}_{\mathtt{T}_3}(\mathtt{Z}); \mathtt{c}_{\mathtt{T}_3};$$

   Escriba un solapamiento *serializable* de las mismas tal que si en algún momento de ese solapamiento una transacción abortara, se debiera abortar otra transacción en cascada. Indique puntualmente esta potencial situación en el solapamiento, y muestre el grafo de precedencias del mismo para justificar su serializabilidad.

4. *(Diseño relacional)* La teoría del diseño relacional brinda herramientas para llevar un esquema de base de datos relacional a una forma normal superior, a través de un proceso conocido como *normalización*. Mencione dos posibles desventajas de mantener una base de datos desnormalizada, y una posible desventaja que surja de normalizarla.

5. *(NoSQL)* El lenguaje CQL utilizado por Cassandra no permite realizar consultas de junta. Por ejemplo, una consulta como la siguiente no está permitida por la sintaxis del lenguaje:

```
SELECT *
FROM R, S
WHERE R.t = S.t;
```

   Explique por qué esta limitación no debiera ser un problema para una aplicación del mundo real que use un SGBD como este.

6. *(NoSQL II)* Una cadena de supermercados almacena información sobre las ventas que realiza en una base de datos MongoDB. El siguiente documento JSON ejemplifica la estructura almacenada por cada venta, que incluye los siguientes campos:

   - El número de ticket, que se utiliza como *_id*.
   - La fecha y hora de la venta.
   - El DNI del comprador.
   - El nombre del comprador.
   - Un vector de documentos embebidos conteniendo cada uno un código de producto, nombre de producto, cantidad de unidades y precio por dicha cantidad de unidades.
   - El monto total de la venta.

```
    {
      "_id": "0043-19354002",
      "fecha_venta": Date("2019-07-15 18:20:58"),
      "DNI_comprador": 37818512,
      "nombre_comprador": "GALVEZ, JUAN ROMAN",
      "productos": [
          { "codigo_producto": 39715,
            "nombre_producto": "arroz la selva 1/2kg",
            "cantidad_producto": 2,
            "precio_por_cantidad": 158.00
          },
          { "codigo_producto": 21820,
            "nombre_producto": "aceite buen olio 1l",
            "cantidad_producto": 1,
            "precio_por_cantidad": 69.00
          }],
      "monto_total": 227.00
    }
```

La gerencia quisiera saber cuál es el código y nombre del producto en que cada persona ha gastado *en términos históricos* la mayor parte de su dinero en el supermercado, con el fin de diseñar alguna nueva oferta tentadora. Escriba una consulta que permita obtener este resultado, con un formato como el mostrado en el siguiente documento de ejemplo, en que la persona se identifica con su DNI:

```
    {
      "_id": 37818512,
      "cod_prod_maximo": 35218,
      "nombre_prod_maximo": "papas emoticones mccoin 1/2kg",
      "total_gastado_prod_maximo": 8918.30
    }
    ...
```

Para recordar parte de la sintaxis de *MongoDB* puede ayudarse con la siguiente consulta que encuentra para cada compra el/los producto/s en que se gastó más dinero (asumiendo que cada producto aparece una única vez en el listado de la compra), y devuelve para cada persona –identificada con su DNI– un listado con dichos productos de mayor gasto indicando: el código y nombre de el/los producto/s en que más gastó en cada compra y el monto gastado en dicho/s producto/s en esa compra:

```
[{ $project: { dni: "$DNI_comprador",
               productos: "$productos",
               mayor_gasto: { $max: "$productos.precio_por_cantidad" }}},
 { $unwind:  "$productos" },
 { $match:   { $expr: {$eq: ["$productos.precio_por_cantidad", "$mayor_gasto"]} } },
 { $group:   { _id: "$dni",
               productos_maximos: { $addToSet:
               { cod_prod_maximo: "$productos.codigo_producto",
                 nombre_prod_maximo: "$productos.nombre_producto",
                 total_gastado_prod_maximo: "$productos.precio_por_cantidad" } } } } ]
```

Nota: Puede asumir que el nombre de producto asociado a cada código de producto es siempre el mismo.