BDA Examenes

GCED

October 2023

Contents

1 2019-2020

1. (25%) Given the following relation, generate two alternative equivalent schemas substantially different from this (still relational), but able to represent exactly the same information. Clearly indicate their integrity constraints.

PlaceCharacteristics(city, characteristic, value)

City	Characteristic	Value
Barcelona	Area	101Km^2
Barcelona	Elevation	12 m
Barcelona	AverageTemperature	21.2C
Barcelona	AveragePrecipitation	640 mm
Paris	Area	105Km^2
Paris	Elevation	70 m
Paris	AverageTemperature	11.4C
Paris	AveragePrecipitation	641 mm

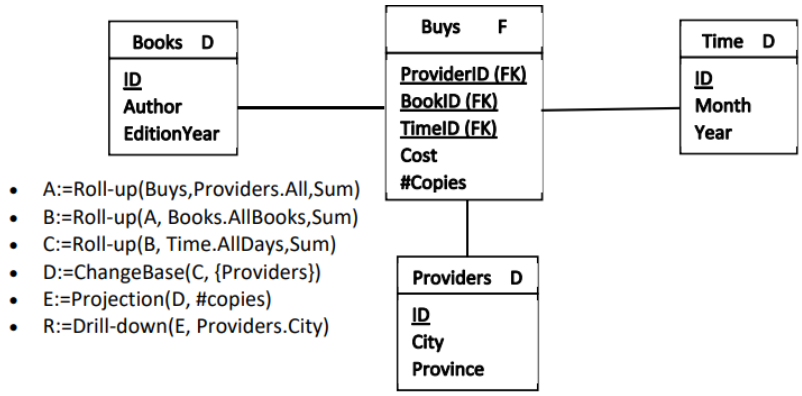
Schema 1

`PlaceCharacteristics(City, Area, Elevation, AverageTemperature, Value)`
Lo que eran datos ahora son metadatos.

Schema 2

```
PlaceCharacteristics(City,Charasteristic, Value)
City(cityID, ...)
Values(valueID, ...)
```

2. (25%) Given the multidimensional schema and the sequence of multidimensional operations, give the equivalent SQL query (simplify it as much as possible).



- A:=Roll-up(Buys,Providers.All,Sum)
- B:=Roll-up(A, Books.AllBooks,Sum)
- C:=Roll-up(B, Time.AllDays,Sum)
- D:=ChangeBase(C, {Providers})
- E:=Projection(D, #copies)
- R:=Drill-down(E, Providers.City)

```
SELECT p.ID, bo.ID, t.ID
FROM Buys b, Time t, Providers p, books bo
WHERE b.ProviderID = p.ID and b.BookID = bo.ID and b.TimeID = t.ID
GROUP BY p.ID, bo.ID, t.ID
ORDER BY p.ID, bo.ID, t.ID  SELECT 'AllProviders', bo.ID, t.ID, SUM(b.Cost),

SUM(b.Copies)
FROM Buys b, Time t, Providers p, books bo
WHERE b.ProviderID = p.ID and b.BookID = bo.ID and b.TimeID = t.ID
GROUP BY 'AllProviders', bo.ID, t.ID
ORDER BY 'AllProviders', bo.ID, t.ID  SELECT 'AllProviders', 'AllBooks',

t.ID, SUM(b.Cost), SUM(b.Copies)
FROM Buys b, Time t, Providers p, books bo
WHERE b.ProviderID = p.ID and b.BookID = bo.ID and b.TimeID = t.ID
GROUP BY 'AllProviders', 'AllBooks', t.ID
ORDER BY 'AllProviders', 'AllBooks', t.ID

SELECT 'AllProviders', 'AllBooks', 'AllDays', SUM(b.Cost), SUM(b.Copies)
FROM Buys b, Time t, Providers p, books bo
WHERE b.ProviderID = p.ID and b.BookID = bo.ID and b.TimeID = t.ID
GROUP BY 'AllProviders', 'AllBooks', 'AllDays'
ORDER BY 'AllProviders', 'AllBooks', 'AllDays'
```

```
SELECT 'AllProviders', SUM(b.Cost), SUM(b.Copies)
FROM Buys b, Time t, Providers p, books bo
WHERE b.ProviderID = p.ID and b.BookID = bo.ID and b.TimeID = t.ID
GROUP BY 'AllProviders'
ORDER BY 'AllProviders'

SELECT 'AllProviders', SUM(b.Copies)
FROM Buys b, Time t, Providers p, books bo
WHERE b.ProviderID = p.ID and b.BookID = bo.ID and b.TimeID = t.ID
GROUP BY 'AllProviders'
ORDER BY 'AllProviders'

SELECT p.City, SUM(b.Copies)
FROM Buys b, Providers p
WHERE b.ProviderID = p.ID
GROUP BY p.City
ORDER BY p.City
```

4. (25%)) Give the relational representation (i.e., tables and their corresponding integrity constraints) of the geographical dimension for a data mart in United Nations,so that it allows to keep track of changes in territories. Assume you only need to keep track of countries and their first administrative level (e.g., Autonomous communities in Spain). We would like to consider the split of countries (like in the case Catalonia would become independent), as well as annexations (like the case of Crimea peninsula by Russian Federation, assuming that peninsula was already a pre-existing component of Ukraine at the first administrative level). Briefly justify your answer and explicit any assumption you make.

Una posible solución es añadir un atributo temporal para saber la validez de los cambios en los paises.

`TerritorialChanges(adminID, yearID, countryID)`

Y por lo tanto la tabla de dimensión tendrá un registro de las administraciones de cada país acompañada de una fecha para marcar la validez/comprobación de actualidad de los datos. En el momento que se produzca un cambio de administración a país, se puede añadir una nueva tupla indicando el cambio con la fecha de suceso/actualización.

Por ejemplo habría dos filas con el registro de Cataluña, una antes y otra despues de su cambio como país independiente.

(España, Cataluña, 2017), (Cataluña, Cataluña, 9999)

Para el caso de una anexión sucedería un caso parecido, p.e. en el caso de Crimea habría una tupla en la que aparecería que pertenece a Ucrania hasta una cierta fecha de actualización y posteriormente un fecha de actualización más reciente con la información de que Crimea pasa a ser de URSS.

(Ucrania, Crimea, 2019), (URSS, Crimea, 2023)

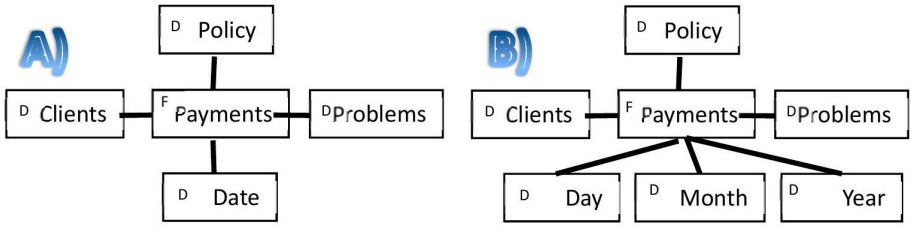
La otra opción sería añadir una columna con `OldCountries`, esto sería una mejor aproximacón si sabemos el número de cambios que se realizaran.

`TerritorialChanges(adminID, OldCountryID, CountryID)`

En la primera podemos añadir el número de cambios que queramos, pero las consultas son más complicadas. En este caso como los países no suelen cambiar mucho podríamos considerar que la mejor opción es la segunda, pero todo depende la situación que tengamos.

2 2020-2021

1. (25%) Which schema would you choose and why:



Si lo que consultamos son datos concretas mejor la opción A, si queremos hacer consultas de por ejemplo todos los días 1 de cada mes mejor la opción B.

Es razonable la opción A porque esperamos que tenga muchas tuplas pero no demasiadas para no se puede hacer join por Payments. Si hubieran segundos esto sería inviable.

Una razón para escoger la B esque las tablas sean densas, es decir, que hayan Payments cada día. Sino habrían diás que no habrían Payments pero seguirian en la tabla de Day, Month y Year.

Ambos son esquemas **Star**, pero el esquema B presenta dos dimensiones adicionales al esquema A. Este aumento de dimensiones presenta un problema mayor que el esquema A a la hora de hacer `join` con la tabla de `payments`. Mientras que el esquema A incorpora las tres dimensiones *Day, Month, Year* de B en una sola dimenión *Date*, haciendo esta más extensa pero aún así se le da prioridad a la reducción de `join` que conlleva.

Efectivamente, todo y que en el esquema A en la tabla de dimensiones de Date habrán muchos términos repetidos (sobretodo si se alarga en el tiempo) y puede sobrecargar la dimensión, el número de tuplas siempre viene dominado por la tabla de hechos. Lo que hace que tener una tabla de dimensión Date, aunque tenga muchas tuplas, sea mejor que tener que hacer demasiados *joins* que sabemos que es la operación más costosa en DBMS.

2. (25%) Name three extraction mechanisms using exclusively the DBMS and briefly explain the requirements each of these pose on it.

- **Triggers**: usando triggers para cuando los datos se carguen en la base de datos original nos los manden directamente a nosotrs.
- **Timestamp-based**: si la BD es temporal podemos pedir los cambios que se han hecho desde la última extracción que hicimos.
- **File comparison**: extraemos todos los datos de la BD y los comparamos con la extracción previa que hicimos para ver las modificaciones.

3. (25%) Consider the R-Swoosh algorithm given for record matching and merging. **Fill the gap and briefly explain** what the missing part does

```
O:= ∅;
while (I<> ∅ ) do {
  pick up r∈I; I:=I-r;
  if (∃s∈O so that s ≈ r) {
     O:=O-s; [          ];
  } else {
     O:=O∪{r};
  }
}
```

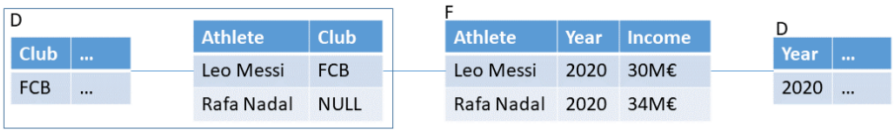
Lo que falta en el algoritmo es

$$I := I \cup \{s \wedge r\}$$

Lo que hace esta parte es meter al input la fusión entre las dos tuplas que tenian similitud usando la función de fusión $\wedge$ que tenemos definida. De esta manera el algoritmo volverá a pasar esta fusión de las tuplas para ver si se parece (similitud) a otras tuplas (se refieren al mismo objeto del mundo real).

3 2021-2022

25% 1. Identify the main problem in the multidimensional schema below, and propose a solution to solve it.



Hay un problema de summarizabilidad, en este caso tenemos un problema que no hay completeza. Podemos añadir un fantasma para todos los que son individuales. Esto sería útil para cuando hacemos por ejemplo dinero por club.

También se debería bajar la granularidad por año a por mes, en el caso que un jugador pueda cambiar de club en un mismo año. En este caso se trata de la *slow changing dimensions*.

***Problema principal en términos de esquema relacional**: En la dimensión de Athlete, los valores de la columna Club pueden see null (en los casos donde el deportista no pertenezca a un equipo, como en el caso de deportes individuales como el tennis), como se muestra en la segunda fila de la tabla de dimensión. Esto es un problema pues el atributo Club es una clave foránea de la tabla de Clubs a la que hace referencia y por lo tanto no puede adoptar valores nulos.*

Una de las posibles soluciones para solucionar este problema sería añadir una tupla en la tabla de *Clubs* donde el identificador sea algo tipo "Individual", donde los demás atributos se pondrían en nulo. Entonces cambiaríamos el Club de estos deportistas en la tabla a "Individual". De esta manera no se violaría la restricción de integridad de la clave foránea.

Otro problema es tener una slowchanging dimension.

2. Explain the difference between "Historic" and "Non-volatile" in the DW definition of W. Inmon. Illustrate it with an example.

En la definición de DW, histórico hace referencia a que la DW registra los cambios que han sufrido los datos a lo largo del tiempo, es decir, tiene datos tanto antiguos como nuevos.

Mientras que No-volatil hace enfasis en la robusteza del DW. No se pueden eliminar no modifcar datos, solo inserir. De manera que no se pierden los datos hasta que el usuario haga petición explicita de ello.

25% 3. Which is the worse-case cost of the R-Swoosh algorithm and when would it happen?

25% 4. Give an example of schematic discrepancy (different from that in the slides), where data in one source is represented as metadata in the other. Draw and briefly explain it.

Haremos un ejemplo con tablas relacionadas con la comida. En el primera esquema las comidas serán los nombres de las tablas y por lo tanto representarán metadatos.

```
Carnes(tipo, precio, ...)
Legumbres(tipo, precio, ...)
Lacteos(tipo, precio, ...)
```

En el segundo esquema, estos metadatos actuaran como instancias (datos) de una tabla principal.

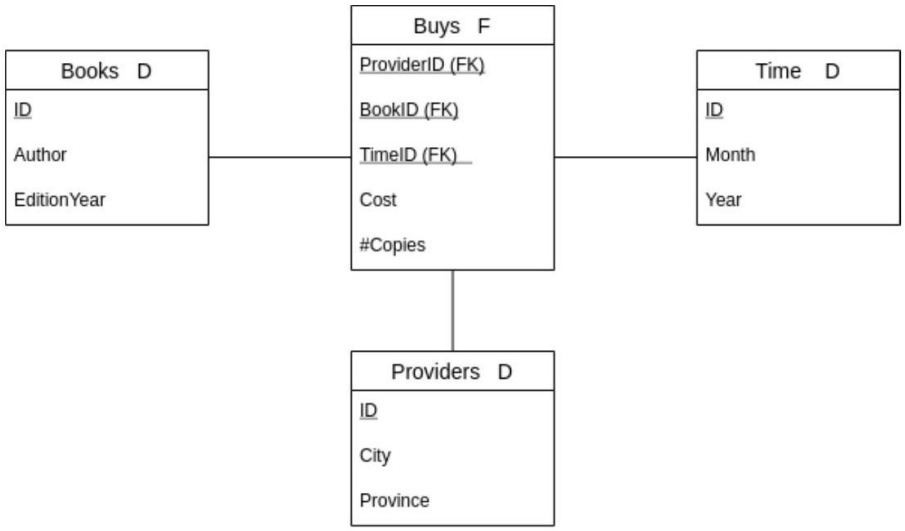
```
Comida(grupo, ...)
```

Con tuplas del tipo:

```
(Carnes, ...)
(Legumbres, ...)
(Lacteos, ...)
```

4 2022-2023

20% 1. Given the multidimensional schema, the sequence of multidimensional operations below, and the equivalent SQL query, justify if they correspond to each other. If they don't, briefly explain why and how the SQL query should be fixed.



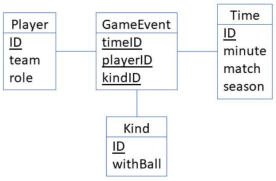
- A:=Roll-up(Buys,Providers.city,Sum)
- B := Roll-up(A, Books. AllBooks, Sum)
- C := Roll-up(B, Time.AllDays,Sum)
- D := ChangeBase (C, { Providers })
- E := Selection(D, city="Barcelona")
- F := Projection(E, #copies)
- R := Drill-down(F, Providers.ID)

```
SELECT p. ID, SUM(bu . cost)
FROM Providers p, Buys bu
WHERE p.ID = bu.providerID AND p. city='Barcelona'
GROUP BY p. ID
ORDER BY p.ID;
```

```
SELECT p.ID, bo.ID, t.ID
FROM Buys bu, Time t, Providers p, books bo
WHERE bu.ProviderID = p.ID and bu.BookID = bo.ID and bu.TimeID = t.ID
GROUP BY p.ID, bo.ID, t.ID
ORDER BY p.ID, bo.ID, t.ID

SELECT p.City, 'AllBooks', t.ID, SUM(bu.cost), SUM(bu.copies)
FROM Buys bu, Time t, Providers p, books bo
WHERE bu.ProviderID = p.ID and bu.BookID = bo.ID and bu.TimeID = t.ID
GROUP BY p.City, 'AllBooks', t.ID
ORDER BY p.City, 'AllBooks', t.ID

SELECT p.City, 'AllBooks', 'AllDays', SUM(bu.cost), SUM(bu.copies)
FROM Buys bu, Time t, Providers p, books bo
WHERE bu.ProviderID = p.ID and bu.BookID = bo.ID and bu.TimeID = t.ID
GROUP BY p.City, 'AllBooks', 'AllDays'
ORDER BY p.City, 'AllBooks', 'AllDays'

SELECT p.City, SUM(bu.cost), SUM(bu.copies)
FROM Buys bu, Providers p
WHERE bu.ProviderID = p.ID
GROUP BY p.City
ORDER BY p.City

SELECT p.City, SUM(bu.cost), SUM(bu.copies)
FROM Buys bu, Providers p
WHERE bu.ProviderID = p.ID and p.City = "Barcelona"
GROUP BY p.City
ORDER BY p.City

SELECT p.ID, SUM(bu.copies)
FROM Buys bu, Providers p
WHERE bu.ProviderID = p.ID and p.City = "Barcelona"
GROUP BY p.ID
ORDER BY p.iD
```

La secuencia de operaciones multidimensionales se corresponde con el esquema multidimensional. El error se encuentra en la equivalencia SQL. Cuando se realiza el select, el atributo de la agregación deberia corresponder a `#copies` y no a `cost`, pues en el resultado de la `Proyection(E,...)`, se selecciona solamente `#copies`.

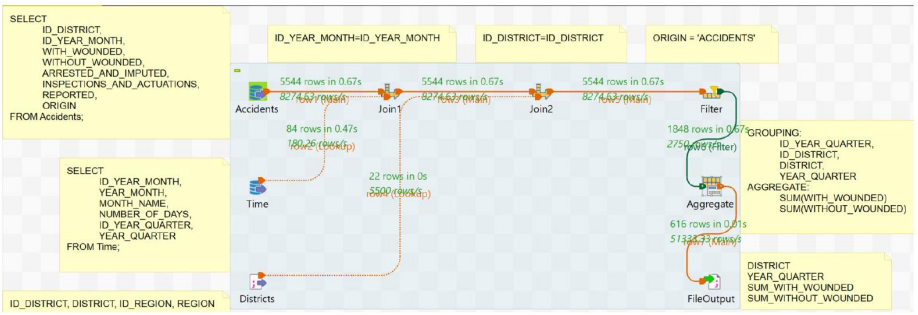
25% 2. The multidimensional schema below corresponds to events in football matches with and without ball (e.g., pass, tackling) involving players with any role in the team (e.g., goalkeeper, defender). Write a cube-query (SQL) over it. If you cannot do it, briefly explain why.



No hay medidas/atributos en la fact table, por lo que se considera una *Factless table*. Aun así se podría hacer una cube-query para contar el número de GameEvents que han ocurrido.

```
SELECT t.ID, p.ID, k.ID, COUNT(*)
FROM GameEvent g, Player p, Kind k, Time t
WHERE g.timeID = t.ID AND g.playerID = p.ID AND g.kindID = k.ID
GROUP BY t.ID, p.ID, k.ID
ORDER BY t.ID, p.ID, k.ID;
```

3. Given the ETL below, where the stickers indicate the attributes and conditions of the nearest node, briefly explain how you would optimize it or justify why you think it is already optimal. Explicit any assumption you need to make and annotate any change in the same diagram to facilitate the explanation.



En la extracción de los datos, se extraen atributos (ARRESTED, INSPECTIONS, REPORTED, YEAR_MONTH, MONTH_NAME, NUMBER_OF_DAYS, ID_YEAR_QUARTER) que no se utilizan ni se cargan al final del proceso de transformación, por lo que desde un principio se pudo haber realizado una selección solo de los atributos necesarios. Además, el paso de filtrado por origen se puede hacer antes del Join1 (de hecho se podria hacer dentro del propio select pero se haría una sobrecarga en el servidor) para reducir el número de tuplas

cuando se hace join (significativamente como se ve en el output de filas después del filter). La idea general es reducir el número de filas y lo antes posible.

También podemos asumir que ID_District es un identificador y por lo tanto no puede pasar que haya 2 identificadores con el mismo distrito. Por lo tanto el `GROUP BY` se podría hacer antes de la segunda join ya que el `DISTRICT` del `GROUP BY` viene completamente identificado por el `ID_DISTRICT` (el resultado será el mismo aunque saquemos `DISTRICT` de la agrupación. .

25% 4. Determine the liveliness of the next three tables. Briefly justify your answer.

create table Teams (
name **varchar (25)**,
budget **int**,
city **varchar (15) not NULL**,
primary key (name),
check (name = 'Futbol Club Barcelona')
);

create table Players (
name **varchar (30) not NULL**,
age **int check (age** > 0 **and age** < 50**),**
team **varchar (25) not NULL,**
number int check (number $>= 0$ **and number** < 100 **),**
primary key (name),
foreign key (team) references teams(name)
) ;

create table Matches
local **varchar** (25)**,**
visiting **varchar(25),**
primary key (local, visiting),
foreign key (local) references teams(name),
foreign key (visiting) references teams(name),
check (local <> visiting)
) ;

1. **Teams (yes/no):** La tabla está viva ya que se pueden inserir tablas mientras se cumplan las restricciones
2. **Players (yes/no):** La tabla está viva ya que se pueden inserir tablas mientras se cumplan las restricciones
3. **Matches (yes/no):** La tabla está muerta porque el check obliga a que los equipos de un partido sean diferentes, sin embargo, tanto los atributos `local` como `visitante` son FK de la tabla `Teams`, donde hay una restricción de que los equipos deben ser del `'Futbol Club Barcelona'`