Exercise session 4

Data bases 2

marco.abbadini@unibg.it

XQuery - Real Estate 🏘️ (1/2)

```
<!ELEMENT Catalogue ( Ad*, VisitRequest* )>

<!ELEMENT Ad ( Apartment, PublishedPrice, Owners,
MinimumAcceptablePrice?, MortgageLoan?, … )>

<!ATTLIST Ad code ID #REQUIRED PublicationDate CDATA
#REQUIRED >

<!ELEMENT Owners ( Person+ )>

<!ELEMENT Person ( FirstName, LastName, Email, Telephone )>

<!ELEMENT VisitRequest ( Person, DateOfRequest,
ScheduledDateForTheVisit?, OfferedPriceAfterVisit?, … )>

<!ATTLIST VisitRequest AdRef IDREF #REQUIRED >
```

XQuery - Real Estate 🏘️ (2/2)

The DTD above describes apartments offered online by a real estate agency on behalf of the owners, and the visits to the apartments booked by potential buyers. Owners specify an initial price and possibly a (confidential) minimum acceptable price to set the bargain range for the agents. Unspecified elements only contain Pcdata. Extract in XQuery:

1. the Apartments that received offers by at least 5 different potential buyers (Email is an identifier for people).
2. the Apartment that received its first visit request after the longest wait after publication.
3. the potential buyers who always and only offered prices below the minimum threshold fixed by the owners.

XQuery - Medical Center 🏥 (1/2)

In the following DTD, unspecified elements contain only PCDATA

```
<!ELEMENT MedicalCenter (Patient+, Exam+)>

<!ELEMENT Patient (Name, Age, Email, HighRisk)>

<!ATTLIST PatientId ID # REQUIRED>

<!ELEMENT Exam (Date, Time, Cost, Outcome +, Doctor)>

<!ATTLIST Exam PatientId IDREF # REQUIRED>

<!ELEMENT Outcome (Parameter, Value, MinVal, MaxVal)>
```

XQuery - Medical Center 🏥 (2/2)

1. Extract in XQuery the parameter that is regular (between the reference values) with the highest frequency (for the query, consider for each parameter the percentage of "normal" outcomes)
2. Extract in XQuery the doctors who have only prescribed exams to patients who came out as perfectly healthy
3. Extract in XQuery the patient with the largest number of values outside of the healthy range in a single exam.

Additional exercises (if we have time)

Trigger - Bollette 💸 (1/2)

Un database gestisce le bollette telefoniche di una compagnia di telefonia mobile.

CLIENTE (CodiceFiscale, nome, cognome, numTelefonico, PianoTariffario)
PIANOTARIFFARIO (Codice, costoScattoAllaRisposta, costoAlSecondo)
TELEFONATA (CodiceFiscale, Data, Ora, numeroDestinatario, durata)
BOLLETTA (CodiceFiscale, Mese, Anno, importo)

1. Scrivere un trigger che a seguito di ogni telefonata aggiorna la bolletta del cliente che ha chiamato.
2. Facciamo l'ipotesi che le bollette da aggiornare siano sempre già presenti nella base di dati, demandando a un altro trigger la creazione di una bolletta di importo 0 per ogni cliente registrato all'inizio di ogni mese.

(si supponga che la fine del mese sia segnalata dall evento END_MONTH)

Trigger - Bollette 💸 (2/2)

3. Scrivere un trigger che alla fine di ogni mese (si supponga che la fine del mese sia segnalata dall evento END_MONTH) sconti dalle bollette 5 centesimi per ogni telefonata diretta a utenti della compagnia (cioè verso numeri di utenti registrati nella tabella CLIENTE) se l importo complessivo della bolletta mensile supera i 100 euro.

Trigger - Orders 🏭

ClientOrder (OrderId, ProductId, Qty, ClientId, TotalSubItems)

ProductionProcess (ProdProcId, ObtainedProdId, StartingProdId, Qty, ProcessDuration, ProductionCost)

ProductionPlan (BatchId, ProdProcId, Qty, OrderId)

PurchaseOrder (PurchaseId, ProdId, Qty, OrderId)

The relational database above supports the production systems of a factory. Table ProductionProcess describes how a product can be obtained by (possibly several) other products, which can be themselves obtained from other products or bought from outside. Build a trigger system that reacts to the insertion of orders from clients and creates new items in ProductionPlan or in PurchaseOrder, depending on the ordered product, so as to manage the client's order (for the generation of the identifiers, use a function GenerateId()). The triggers should also update the value of TotalSubItems (initially always set to 0) to describe the number of sub-products (internally produced or outsourced) that are used overall in the production plan deriving from the order. Also briefly discuss the termination of the trigger system

Trigger - Parcel Delivery 🛬 (1/2)

A parcel delivery company has distribution centers around the world, connected by flights. Delivery is requested by inserting tuples into table PARCEL, annotated with two timestamps: the request time and the delivery deadline (time by which the parcel should be in the destination center). Scheduled flights have fixed routes and timing (departure and arrival Ids and timestamps), and a capacity that is expressed in number of transportable parcels. A table DISTANCE stores the distances between all centers.

FLIGHT (<u>FlightId</u>, OriginId, DestinId, DepTime, ArrivalTime, TotalCapacity, AvailableCapacity)

PARCEL (<u>ParcelId</u>, OriginId, DestinId, TimeReceived, DeliveryDeadline, FlagOnTime)

PARCELROUTESTEP (<u>ParcelId, FlightId</u>, FlagFinalStep)

DISTANCE (<u>OriginId, DestinId</u>, Km)

Trigger - Parcel Delivery 🛬 (2/2)

Write a trigger system that manages the creation of the route for the parcels, possibly split into several PARCELROUTESTEPs, according to the following “greedy” strategy. The triggers react to the insertion of a new parcel, and assign it to the first flight with available capacity that either

(i) directly reaches the final destination f of the parcel, or (ii) moves it to a center c that is not only closer to f than the current location cl, but is the closest to f among those directly reachable from cl.

The triggers must also verify that the TimeReceived of the parcel precedes the DepTime of the chosen flight and, when the delivery is not direct but split over multiple steps, the ArrivalTime of each step must precede the DepTime of the next step.

FlagFinalStep is set to true when the step reaches the final parcel destination, while FlagOnTime, initially always set to true, must state (at the end of the computation for each parcel) whether it will reach the final destination before the deadline or not.

Also, briefly discuss the termination of the trigger system.

Optimization - Student 😱

A table STUDENT(RegNo, Name, City) has 100K tuples in 7K blocks with an entry-sequenced organization. There are B +-tree indexes on Name and City, both of depth 3, with the ability to reach 5 tuples in average for each value of Name, and 40 tuples in average for each value of City. Consider the following SQL query

```
select * from Student

where Name in (Name1, Name2, ..., Namen) and

City in (City1, City2, ..., Cityc)
```

Determine the optimal strategy for query execution based on the number of values (n,c) of Name and City listed in the where clause query, considering these four cases:

1. (n,c) = (10, 10)
2. (n,c) = (1000, 10)
3. (n,c) = (10, 1000)
4. (n,c) = (1000, 1000)

Optimization - Actors and Roles 🎬 (1/2)

A table Role(Actor, Movie, Character) records 400K roles played by Hollywood actors in over many decades. Estimate the execution cost (under reasonable assumptions) of the following query in the scenarios listed below. Please briefly describe the considered query plan in each scenario.

```
select Actor, Movie, count(*) as NumberOfCharacters

from Role

group by Actor, Movie

having count(*) > 2
```

Optimization - Actors and Roles 🎬 (2/2)

1. The table is primarily stored in 16K blocks, with tuples in no particular order. There is also a hash based secondary index with Movie as key, with 5K buckets of 1 block each (val(Movie)=20K, val(Actor)=25K).
2. The table is primarily stored in 16K blocks, with tuples sequentially ordered according to the Movie attribute (as they are sequentially appended as soon as new movies are released), and there are no secondary access structures.
3. The table is primarily stored as in case 1, but the secondary structure, instead of being a hash, is a B+ tree with two attributes as key (Actor, Movie) – i.e., the key is composed of the two attributes, in this order. The tree has depth 3 (a root, an intermediate level, and 3.5K leaf nodes).