Data Bases II
Trigger

Michele Beretta

michele.beretta@unibg.it

Exercise T.1

```
ClientOrder(OrderId, ProductId, Qty, ClientId, TotalSubItems)
ProductionProcess(ProdProcId, ObtainedProdId, StartingProdId, Qty, ProcessDuration, ProductionCost)
ProductionPlan(BatchId, ProdProcId, Qty, OrderId)
PurchaseOrder(PurchaseId, ProdId, 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 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.

Exercise T.2

Si consideri il seguente schema relativo a un sistema di noleggio di sale prove per gruppi musicali

```
Cliente(CodiceFiscale, Nome, Cognome, Tipo)
Prenotazione(CodFisCliente, CodiceSala, Giorno, OraInizio, OraFine)
UsoEffettivo(CodFisCliente, CodiceSala, Giorno, OraInizio, OraFine, Costo)
Sala(Codice, CostoOrario)
```

L'uso effettivo può avvenire solo in corrispondenza di una prenotazione, e al limite iniziare dopo o terminare prima degli orari della prenotazione

1. Si scriva un trigger che impedisca di prenotare una sala già prenotata
2. Si supponga che i dati sull'uso effettivo siano inseriti solo al termine dell'uso della sala. Si propongano un arricchimento dello schema per regole che assegni il valore `"Inaffidabile"` al campo `tipo` dei clienti all'accumulo di 50 ore prenotate e non usate.

Exercise T.3

```
Possiede(Societa1, Societa2, Percentuale)
```

Si consideri, per semplicità, che le tuple della tabella `Possiede` siano inserite a partire da una tabella vuota, che poi non viene più modificata.

Si costruisca tramite trigger la relazione `Controlla` (una società A controlla una società B se A possiede direttamente o indirettamente più del 50% di B).

Si assuma che le percentuali di controllo siano numeri decimali (tra 0 e 1), e si tenga presente che la situazione di controllo avviene in modo diretto quando una società possiede più del 50% della “controllata” o indiretto quando una società controlla altre società che possiedono percentuali di B e la somma delle percentuali possedute e controllate da A supera il 50%.

Exercise T.4 – Transazioni

Dato il seguente schema:

```
Titolo(CodTitolo, Nome, Tipologia)
Transazione(CodTrans, CodVenditore, CodAcquirente, CodTitolo, Qta, Valore, Data, Istante)
Operatore(Codice, Nome, Indirizzo, Disponibilità)
```

Costruire un sistema di trigger che mantenga aggiornato il valore di `Disponibilità` di `Operatore` in seguito all'inserimento di tuple in `Transazione`, tenendo conto che per ogni transazione in cui l'operatore vende l'ammontare della transazione deve far crescere la disponibilità e per ogni acquisto deve invece diminuire.

Inserire inoltre gli operatori la cui disponibilità scende sotto lo zero in una tabella che elenca gli operatori "scoperti".

```
Scoperto(Codice, Nome, Indirizzo)
```

Exercise T.5 – Viaggi

Facendo riferimento alla base dati:

```
Dipendente(Codice, Nome, Qualifica, Settore)
Viaggio(Codice, Dip, Destinazione, Data, Km, TargaAuto)
Auto(Targa, Modello, CostoKm)
Destinazione(Nome, Stato)
```

Si consideri la vista:

```
Viaggi(Dipendente, KmTot, CostoTotale)
```

Scrivere due regole attive che calcolano il valore della vista a seguito di inserzioni di nuovi viaggi, una in modo incrementale e una ricalcolando l'intera vista.

Exercise T.6 – Campus

Dato il seguente schema relazionale:

```
Studenti(Matr, Nome, Residenza, Telefono, CorsoDiLaurea, AnnoCorso, Sede, TotCrediti)
Iscrizioni(MatrStud, Corso, Anno, Data)
CorsiAnni(CodCorso, Anno, Docente, Semestre, NroStudenti, NroFuoriSede)
Abbinamenti(CodCorso, CorsoLaurea)
Corsi(CodCorso, Titolo, Crediti, Sede)
```

1. Scrivere una regola attiva che controlla ogni inserimento di una tupla nella tabella `Iscrizioni` e nel caso in cui non esistano elementi corrispondenti in `Studenti` o `CorsiAnni` (cioè l'iscrizione non sia una iscrizione significativa, perché lo studente è sconosciuto o il corso non attivo) annulli l'inserimento.
2. Supponendo che l'attributo `NroFuoriSede` di `CorsiAnni` rappresenti il numero di studenti iscritti al corso che fanno riferimento a una sede *diversa* da quella del corso, scrivere una regola attiva che reagisce alle modifiche dell'attributo `Sede` di `Studente`, aggiornando se necessario il valore dell'attributo `NroFuoriSede`.

Exercise T.7 – Prodotti

Lo schema seguente descrive un insieme gerarchico di prodotti, dove per i prodotti non contenuti in altri prodotti assumiamo `Level` (che descrive il livello di profondità a cui il prodotto è situato nella gerarchia) pari a zero e `SuperProdotto` pari a `NULL`:

```
Product(Code, Name, Description, SuperProduct, Level)
```

1. Scrivere una regola attiva che alla cancellazione di un prodotto cancelli tutti i sottoprodotti corrispondenti
2. Scrivere una regola attiva che, alla creazione di un nuovo prodotto (*eventualmente* figlio di un prodotto esistente), calcoli il valore dell'attributo `Level`
3. Implementare mediante regole attive un vincolo di pseudo-integrità referenziale sull'attributo `SuperProduct`, per cui gli unici valori ammessi sono `NULL` oppure il codice di un altro prodotto (non se stesso)