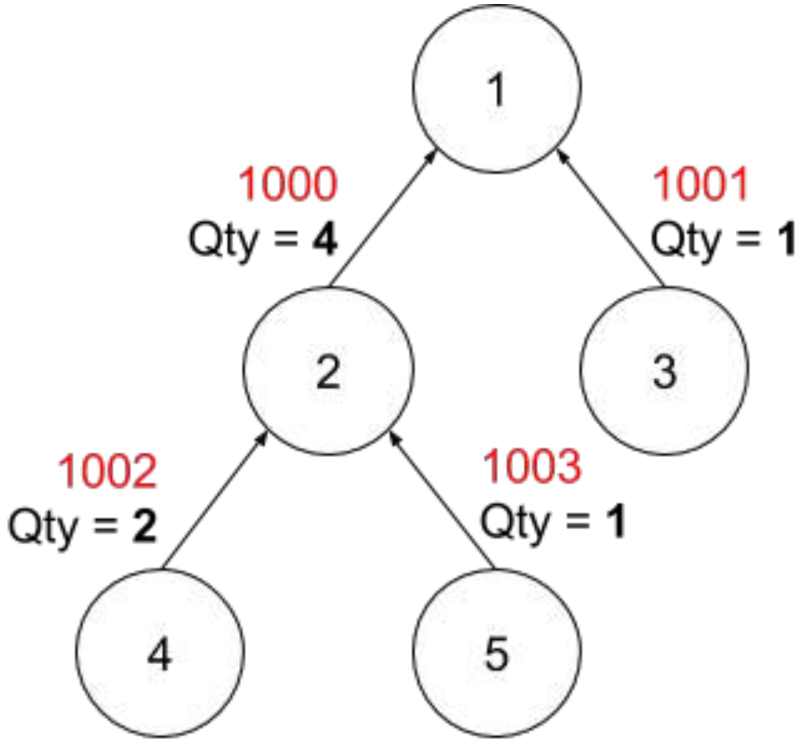
Orders (11 / 02 / 2016)

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.

sqliteonline: https://goo.gl/Mw4rYB



ProdProc Id	Obtained ProdId	Starting ProdId	Qty
1000	1	2	4
1001	1	3	1
1002	2	4	2
1003	2	5	1

sqliteonline: https://goo.gl/QiOS01

We have to define at least the following triggers:

- **T1 (NewOrder)** reacts to the insertion on ClientOrder and:
  - Adds a record in ProductionPlan if there is a process to build ProductId
  - Adds a record in PurchaseOrder if there is no process to build ProductId

- **T2 (UpdateSubItemsAfterPurchase)** reacts to insertion on PurchaseOrder
  - Sum the ordered Qty to the TotalSubItems of the order

- **T3 (UpdateSubItemsAfterProduction)** reacts to insertion on ProductionPlan
  - Sums the produced Qty to the TotalSubItems of the order

- **T4 (InsertSubProducts)** reacts to insertion on ProductionPlan
  - Adds a record in ProductionPlan if there is a process to build **StartingProdId**
  - Adds a record in PurchaseOrder if there is no process to build **StartingProdId**

- **T1 (NewOrder)** reacts to the insertion on ClientOrder

```
CREATE TRIGGER NewOrder
AFTER INSERT ON ClientOrder
FOR EACH ROW
BEGIN
        IF (EXISTS (SELECT * FROM ProductionProcess
                       WHERE ObtainedProdId = new.ProductId))

                INSERT INTO ProductionPlan
                SELECT GenerateId(), ProdProcId, Qty * new.Qty, new.OrderId
                FROM ProductionProcess
                WHERE ObtainedProdId = new.ProductId;

        ELSE
                INSERT INTO PurchaseOrder VALUES
                (GenerateId(), new.ProductId, new.Qty, new.OrderId);

        END;
END;
```

sqliteonline: https://goo.gl/9QGmtp

- **T1 considerations**:

  - When **new**.ProductId is the ObtainedProdId of a ProductionProcess, we need to insert the records in ProductionPlan to transform its starting products into the obtained product;

  - When **new**.ProductId **isn't** an ObtainedProdId of any ProductionProcess, we need to purchase the ProductId (we are actually re-selling);

  - The production quantity of each Starting Product is **new**.Qty (the number of **new**.ProductId items to produce for the order) * Qty (the number of Starting Products needed to produce one Obtained Product).

- **T2 (UpdateSubItemsAfterPurchase)** reacts to insertion on PurchaseOrder

```
CREATE TRIGGER UpdateSubItemsAfterPurchase
AFTER INSERT ON PurchaseOrder
FOR EACH ROW
BEGIN

        UPDATE ClientOrder
        SET TotalSubItems = TotalSubItems + new.Qty
        WHERE OrderId = new.OrderId;

END;
```

sqliteonline: https://goo.gl/JXiSXC

- **T3 (UpdateSubItemsAfterProduction)** reacts to insertion on ProductionPlan

```
CREATE TRIGGER UpdateSubItemsAfterProduction
AFTER INSERT ON ProductionPlan
FOR EACH ROW
BEGIN

        UPDATE ClientOrder
        SET TotalSubItems = TotalSubItems + new.Qty
        WHERE OrderId = new.OrderId;

END;
```

sqliteonline: https://goo.gl/PKyDlJ

- **T4 (InsertSubProducts)** reacts to insertion on ProductionPlan

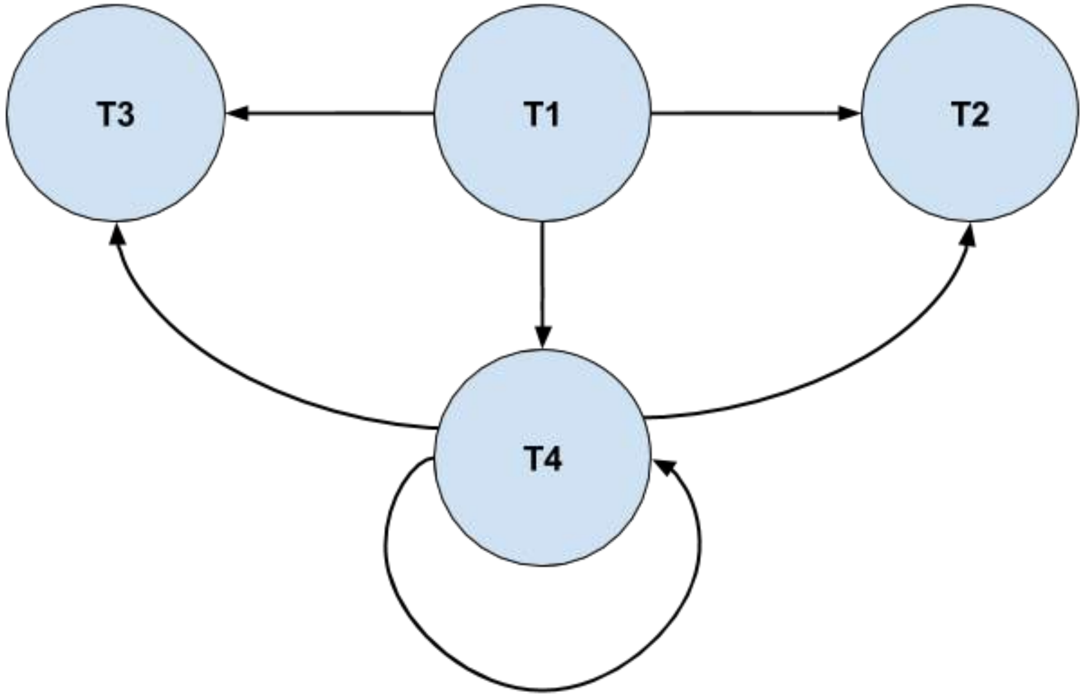
```
CREATE TRIGGER InsertSubProducts
AFTER INSERT ON ProductionPlan
FOR EACH ROW
BEGIN
        DEFINE S;
        SELECT StartingProdId INTO S
        FROM ProductionProcess WHERE ProdProcId = new.ProdProcId;

        IF (EXISTS (SELECT * FROM ProductionProcess
                    WHERE ObtainedProdId = S))

                INSERT INTO ProductionPlan
                SELECT GenerateId(), ProdProcId, new.Qty * Qty, new.OrderId
                FROM ProductionProcess WHERE ObtainedProdId = S;
        ELSE
                INSERT INTO PurchaseOrder VALUES
                (GenerateId(), S, new.Qty, new.OrderId);
        END;
END;
```

sqliteonline: https://goo.gl/ifrAJO

Termination of the trigger system



- T4 is the only trigger that could be non-terminating
- Nevertheless, if the product hierarchy is well-formed (no cycles), T4 will eventually terminate reaching the leaves.

We can define other (optional and not required) triggers to improve the system:

- **T5 (Validate Order)**
  - Validates TotalSubItems = 0
  - Validates Qty > 0

- **T6 (Delete Order)**
  - Delete all associated PurchaseOrders
  - Delete all associated ProductionPlans

- **T7 (Disable Order Updates)**
  - Permit updates on TotalSubItems
  - Disable updates on other fields

sqliteonline: https://goo.gl/JwhKT2

- **T5 (Validate Order)**

```
CREATE TRIGGER NewOrder_validate
BEFORE INSERT ON ClientOrder
FOR EACH ROW
WHEN ((new.TotalSubItems <> 0) OR (new.Qty <= 0))
BEGIN

        SELECT RAISE(ABORT, "Invalid Order");

END
```

sqliteonline: https://goo.gl/JwhKT2

- **T6 (Delete Order)**

```
CREATE TRIGGER DeleteOrder
AFTER DELETE ON ClientOrder
FOR EACH ROW
BEGIN

        DELETE FROM ProductionPlan
        WHERE OrderId = old.OrderId;

        DELETE FROM PurchaseOrder
        WHERE OrderId = old.OrderId;

END;
```

sqliteonline: https://goo.gl/JwhKT2

- **T7 (Disable Order Updates)**

```
CREATE TRIGGER DisableOrderUpdates
BEFORE UPDATE OF OrderId, ProductId, Qty, ClientId ON ClientOrder
FOR EACH ROW
BEGIN

        SELECT RAISE(ABORT, "Updates on ClientOrder are disabled");

END;
```

sqliteonline: https://goo.gl/JwhKT2