“Material Requirements Planning (MRP) Database Model”

Ahmet Yiğit Doğan

14.01.2023

Table of Contents

Introduction

In this study, a set of SQL calculations are programmed to implement the Material Resources Planning Algorithm. The queries include a sequence of statements to create a sample database, similar to the trumpet production example from Production and Operations Analysis (Nahmias, Olsen, 2015), function definitions, stored procedure definitions, and finally an SQL trigger that takes action after a new order is inserted into the database, all of which can be found in the folder as *".sql"* files. The key SQL concepts that were resorted to in the code are as follows:

- Functions (to easily reach to item specific information)
- Triggers (to make the database take action after an order insertion)
- Stored Procedures (to take a sequence of actions after the trigger)
- Recursive Common Table Expressions (to update the "required_items" table by inserting a breakdown structure)
- Cursors (to update the *"item_period"* table for each item and each period)

The E-R Diagram of the Database Model

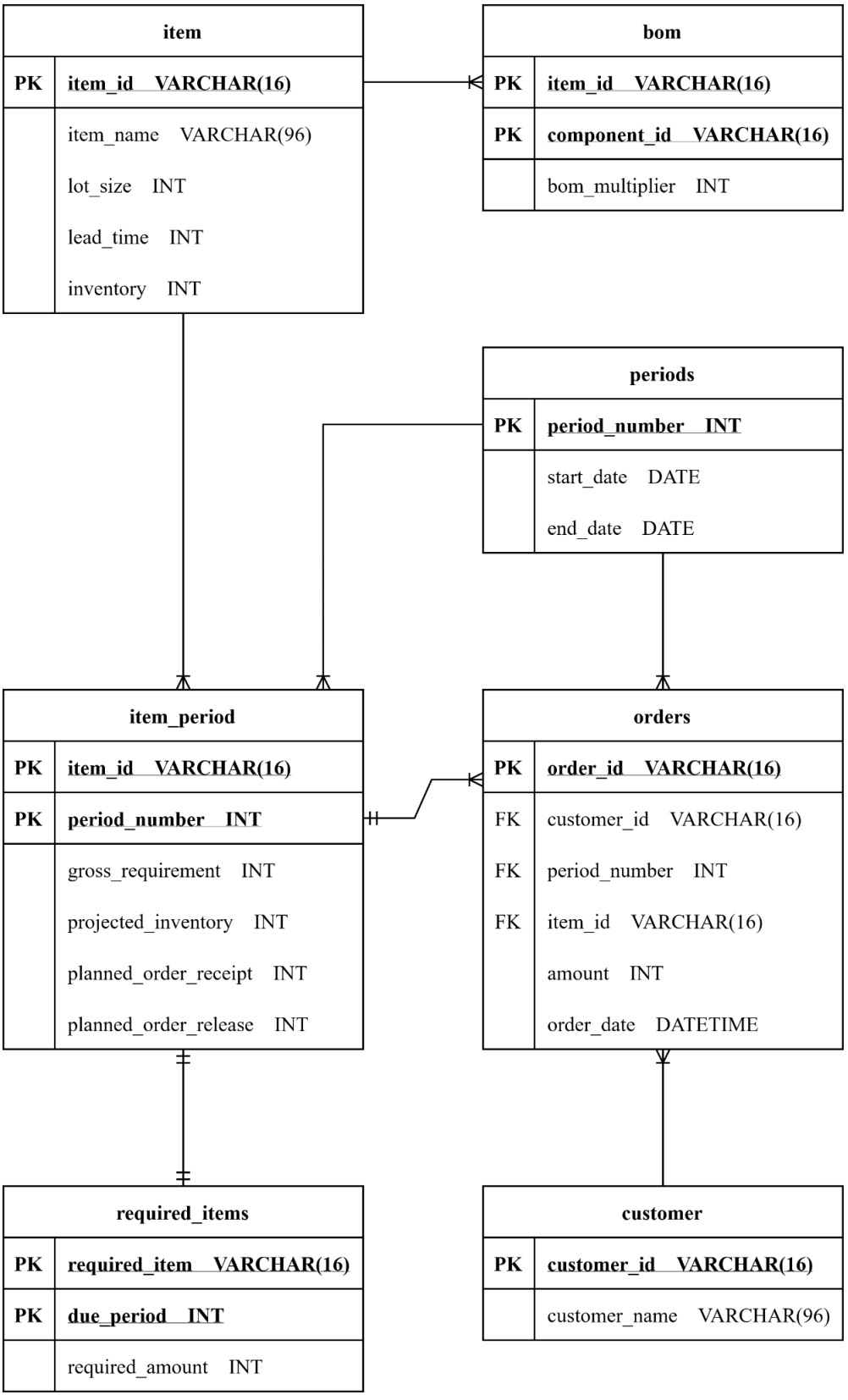

Figure 1. Entity-Relationship Diagram of the Database Model

Table 1: item

The table “item” includes the features of the materials, such as lot size and lead time. Its primary key is *“item_id”* which is unique for each material type. The field *“inventory”* refers to the initial inventory of the items.

Table 2: bom

This table stores the parent-child relationships between the items. For each row of the table, *“item_id”* and *“component_id”* pairs (parenthood cases) are unique. The table *“item”* has one-to-many relationship with this table since an item can have more than one child item.

Table 3: item_period

This table is actually the MRP table. It contains period-based information about the items. Thus, it has *“item_id”* and *“period_number”* as composite primary keys. Period number refers to a specific time period (like 6th week, 10th month etc. It depends on the basis that the database is decided to be used. In the test case of this study, it will be used on a weekly basis, covering one quarter of the year.). The start and end dates of the periods are stated in the *“periods”* table which has one-to-many relationship with this table.

Table 4: orders

The *“orders”* table is the table to be edited when a new order is inserted in the database. An insert on this table triggers updates on “item_period” and “required_items” tables. It has many-to-one relationship with “item_period” since more than one order may request a certain product to be delivered in the same period.

Table 5: required_items

This table plays the role of intermediary step for the calculations. Since MySQL does not allow temporary table creation inside triggers, I inserted “required_items” as a dummy table to the database to store the list of required

items after each order insertion. Before each update on this table, all the data present in the table is wiped, and then a new list of items are inserted.

The data inserted to this table is actually the breakdown structure of an item, which lists all the required materials to produce that material along with their due periods. For example, assume that we have the following Bill of Materials:

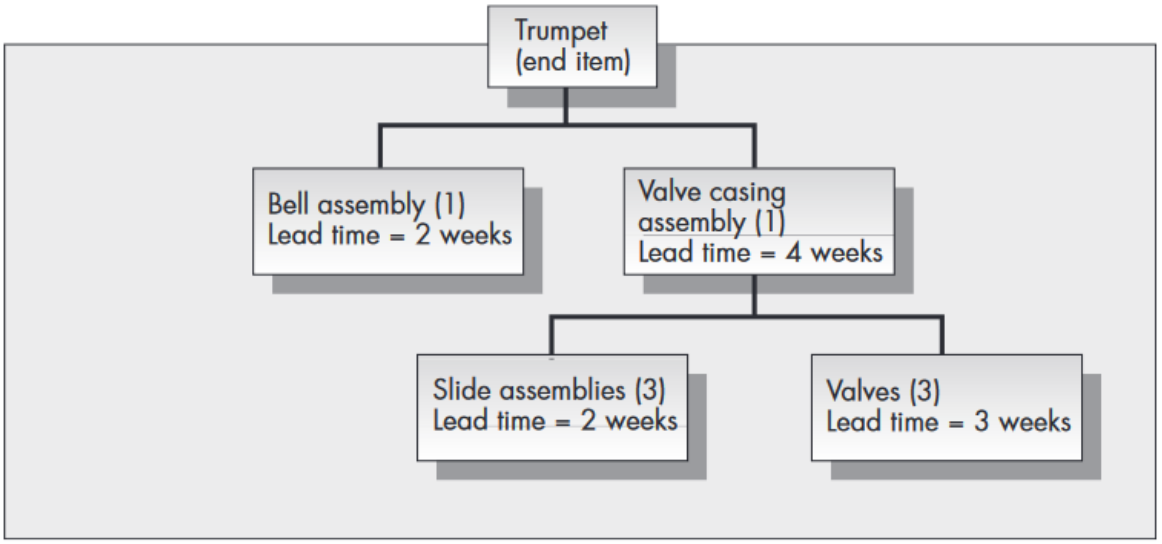

Figure 2. BOM of the Trumpet Production Example

Where Trumpet has the lead time 2 weeks. If an order of 5 trumpets given to be delivered in the 10 weeks from now, *"required_items"* table would be updated as follows:

item_name	required_amount	due_period
Trumpet	15	10
Valve Casing Assembly	15	8
Bell Assembly	15	8
Slide Assemblies	45	4
Valves	45	4

Table 1. "required_items" After Order

Afterwards, the *"gross_requirement"* field of the *"item_period"* table is updated according to these values. Since the required amounts are calculated

according to an algorithm inside a procedure called “GetRequiredItemsCount”, it is not required to include BOM level as a separate field in the database.

The table has one-to-one relationship with “item_period” since each row can be paired with one *“item_id”*, *“period_id”* pair.

Table 6: periods

This table just stores the period start and finish dates, and its primary key is *“period_number”* as expected.

Table 7: customer

The “customer” table stores the customer specific information, which is not that crucial for the workflow of the database.

Workflow of the Program

The database updates itself when a new row is inserted into the “orders” table. Thus, it is the only table to be updated manually, when a new order is received. After insertion, the following actions are taken by database:

- The *“after_order_insertion”* trigger gets alerted. This trigger consists of two parts. The first part calls the procedure *“GetRequiredItemsCount”* to obtain the breakdown structure mentioned in the previous parts.
- By using the breakdown structure, the gross requirements in the table *“item_period”* are updated.
- The second part of the trigger calls the procedure “UpdateMRPTables”, which includes the core MRP calculations.
- Inside this procedure, cursors loop through the items and periods to update projected inventories, planned order receipts, and planned order releases “according to the gross requirements” which have been updated in the first part of the trigger.

Test Case Results

To test the calculations, I created a sample database that includes the item information of trumpet production example. I set the initial inventories as follows:

- Trumpet: 3
- Valve Casing Assembly: 10
- Bell Assembly: 10
- Slide Assemblies: 20
- Valves: 30

I used a one quarter time horizon (12 weeks, excluding the initial state) and inserted a 15 trumpet order to be delivered in the 10th week. The MRP calculations yielded the following results:

Period	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Gross Requirement	0	0	0	0	0	0	0	0	0	0	15	0	0
Projected Inventory	3	3	3	3	3	3	3	3	3	3	0	0	0
Planned Order Receipt	0	0	0	0	0	0	0	0	0	0	12	0	0
Planned Order Release	0	0	0	0	0	0	0	0	12	0	0	0	0

Table 2. Test Case Results of Trumpet

Period	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Gross Requirement	0	0	0	0	0	0	0	0	15	0	0	0	0
Projected Inventory	10	10	10	10	10	10	10	10	5	5	5	5	5
Planned Order Receipt	0	0	0	0	0	0	0	0	10	0	0	0	0
Planned Order Release	0	0	0	0	10	0	0	0	0	0	0	0	0

Table 3. Test Case Results of Valve Casing Assembly

Period	0	1	2	3	4	5	6	7	8	9	10	11	12
Gross Requirement	0	0	0	0	0	0	0	0	15	0	0	0	0
Projected Inventory	10	10	10	10	10	10	10	10	5	5	5	5	5
Planned Order Receipt	0	0	0	0	0	0	0	0	10	0	0	0	0
Planned Order Release	0	0	0	0	0	0	10	0	0	0	0	0	0

Table 3. Test Case Results of Bell Assembly

Period	0	1	2	3	4	5	6	7	8	9	10	11	12
Gross Requirement	0	0	0	0	45	0	0	0	0	0	0	0	0
Projected Inventory	20	20	20	20	15	15	15	15	15	15	15	15	15
Planned Order Receipt	0	0	0	0	40	0	0	0	0	0	0	0	0
Planned Order Release	0	0	40	0	0	0	0	0	0	0	0	0	0

Table 4. Test Case Results of Slide Assemblies

Period	0	1	2	3	4	5	6	7	8	9	10	11	12
Gross Requirement	0	0	0	0	45	0	0	0	0	0	0	0	0
Projected Inventory	30	30	30	30	15	15	15	15	15	15	15	15	15
Planned Order Receipt	0	0	0	0	30	0	0	0	0	0	0	0	0
Planned Order Release	0	30	0	0	0	0	0	0	0	0	0	0	0

Table 5. Test Case Results of Valves

Future Improvements

Although this model does not cover all details of a database that will be used for MRP calculations, it provides a good basis to work on. It can be enhanced by the introduction of the following functionalities, which lack currently:

- Backlog handling
- Data validation (for example, only accepting feasible orders)
- Advanced inventory management
- Continuously-measured items (by height, weight etc.)
- New triggers in addition to order insertion (for example, scheduled receipts)

References

Nahmias, S., Olsen, T. (2015). *Production and Operations Analysis.* (pp. 444-445). Waveland Press. Seventh edition.