

Final Project Submission – Shopfloor Inventory Management

MIS 659

Relational Database Management

Mr. David Sanford

Shopfloor Inventory Management System

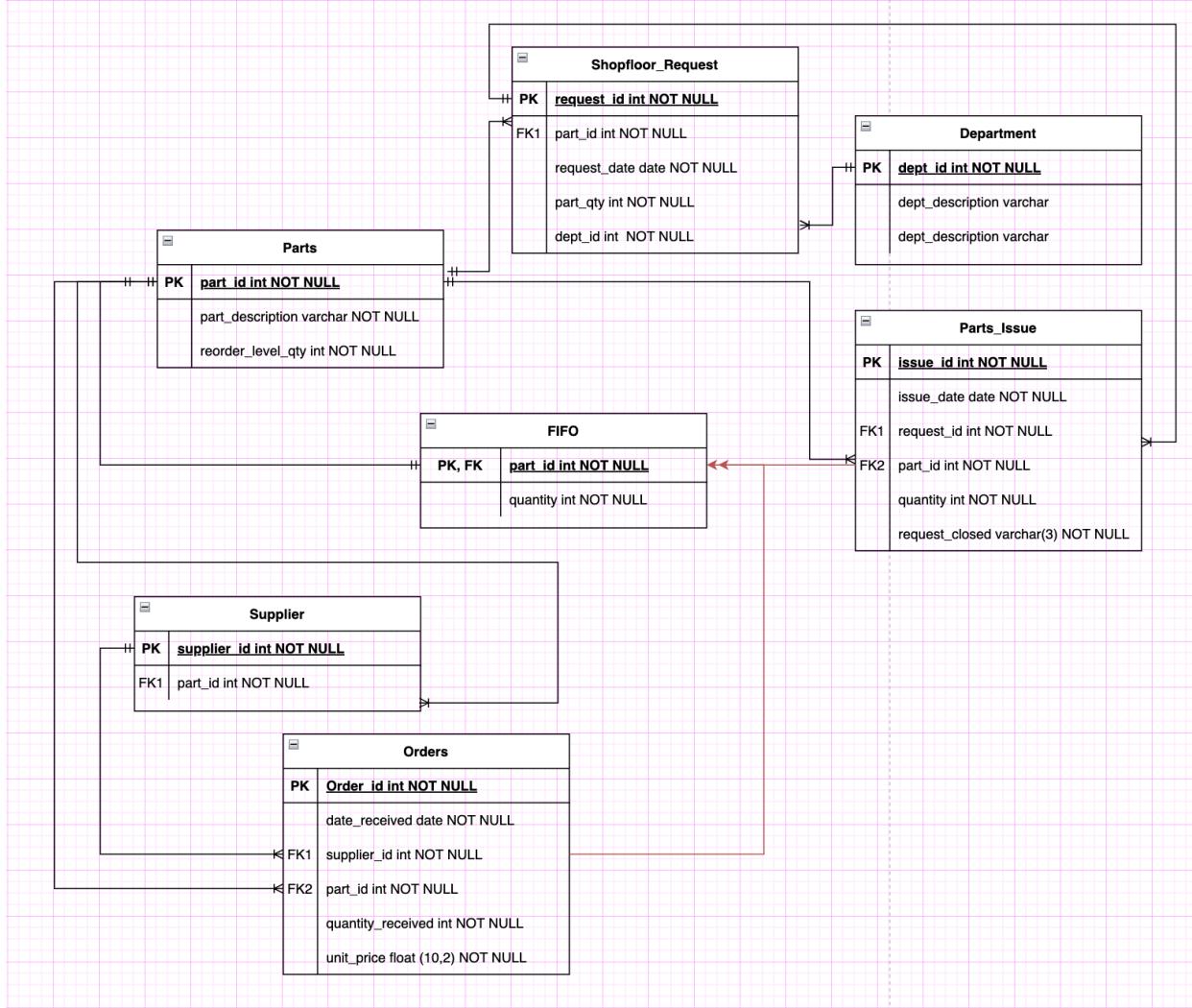
This document details the process of designing a database structure to manage inventory of parts for a shopfloor that is engaged in assembling home appliances such as refrigerators and air conditioners. This hypothetical shopfloor consists of few Assembly lines (please note that initially I have inserted data for four assembly lines and to test user interface created with Python code using Qt5 Python Widgets I have input more Assembly lines). A sample of ten types of parts are taken for this assignment. The iterative process of the system involves obtaining request for material parts from assembly lines, issuing parts to respective assembling departments, ordering the parts with pre-approved vendors as and when reorder levels are reached, automatically updating inventory table with available parts of each variety upon issue of parts as well as orders received from vendors.

Database Structure

The database structure consists of seven interrelated tables to maintain data integrity and consistency viz. Department (Assembly lines), Parts, Shopfloor_Request, Supplier, Orders, Parts_Issue, and FIFO (First-in-First-Out inventory). Each of the rows of these database tables store unique records. The database tables viz. Parts and Department are independent tables with no foreign keys. Null values are restricted for all the tables. Relationships between these database tables along with an ER diagram are described in next section.

Database Design

Figure 1: ER Diagram



In this section the relationships between the tables and key constraints are discussed.

- There is a one-to-many relationship between Department and Shopfloor_Request, a department table can have unique departments but could create multiple requests for material parts.
- FIFO table has a one-to-one relationship with Parts table and as the relationship line indicates a minimum of one part_id is necessitated for both tables.

Final Project Submission – Shopfloor Inventory Management

- part_id is primary as well as foreign key for FIFO table. As denoted with colored lines in the above ER Diagram, FIFO table is automatically updated upon issue of parts and receipt of orders (using triggers).
- Another trigger is created to update the field “request_closed” as and when entire quantity of parts of a request_id are issued.
- Parts table is one of the crucial link between tables with five other tables depending on it.
- Please note that as indicated in the cardinality and connectivity lines, a minimum of one entry for primary and foreign key fields is maintained.
- Database Views are created to query the most sought parts by assembly lines.
- Other relationships in the above ER diagram are self-explanatory.

Final Project Submission – Shopfloor Inventory Management

Table Creation and Data types

Here are the screenshots of the creation of database tables (DB file and sql files are attached/submitted along with this document).

The screenshot shows the SQLiteOnline IDE interface. On the left, there's a sidebar with a tree view of databases and their tables. The current database is 'sqlite-8.db' which contains tables like Department, FIFO, Orders, Parts, etc. The main area displays SQL code for creating tables:

```
1 CREATE TABLE Parts (
2     part_id INTEGER PRIMARY KEY AUTOINCREMENT,
3     part_description VARCHAR(255),
4     reorder_level_qty INT
5 );
6
7 CREATE TABLE Department (
8     dept_id INTEGER PRIMARY KEY AUTOINCREMENT,
9     dept_description VARCHAR(255)
10 );
11
12 CREATE TABLE Supplier (
13     supplier_id INTEGER,
14     part_id,
15     PRIMARY KEY (supplier_id, part_id),
16     FOREIGN KEY (part_id) REFERENCES Parts(part_id)
17 );
18
19 CREATE TABLE Shopfloor_Request (
20     request_id INTEGER,
21     part_id INTEGER,
22     request_date DATE,
23     part_quantity INT,
24     dept_id INTEGER,
25     PRIMARY KEY (request_id, part_id),
26     FOREIGN KEY (part_id) REFERENCES Parts(part_id),
27     FOREIGN KEY (dept_id) REFERENCES Department(dept_id)
28 );
```

On the right, there's a 'History' panel showing recent queries:

- SQLite.10.sql: `SELECT * FROM high_demand_parts` at 14:43:13
- SQLite.10.sql: `CREATE VIEW high_demand_parts AS SELECT sum(part_quantity) ...` at 13:36:38

A preview table is shown at the bottom:

part_id	part_description	sum(part_quantity)
3	Evaporator Fan Motor	23

Final Project Submission – Shopfloor Inventory Management

The screenshot shows a Google Chrome window with multiple tabs open. The active tab is "sqliteonline.com". The interface includes a sidebar with database navigation, a central code editor with SQL scripts, and a bottom pane showing a table preview.

SQLite Online IDE Features:

- File:** Options for "Owner DB", "Run", "Export", "Import", and "Client".
- Sign In:** Available in the top right corner.
- Toolbar:** Includes icons for "Star", "Share", "Help", and "More".
- Right Panel:** Settings for "Summary", "Left Menu", "Editor", and "History".
- History:** Shows recent queries and their execution times.
- Bottom Panel:** Preview of the "high_demand_parts" table.

Table Structure:

```
CREATE TABLE Parts_Issue (
    issue_id INTEGER PRIMARY KEY AUTOINCREMENT,
    issue_date DATE,
    request_id INT,
    part_id INTEGER,
    quantity INT,
    request_closed VARCHAR(3),
    FOREIGN KEY (request_id) REFERENCES Shopfloor_Request(request_id),
    FOREIGN KEY (part_id) REFERENCES Parts(part_id)
);

CREATE TABLE Orders (
    order_Id INTEGER PRIMARY KEY AUTOINCREMENT,
    date_received DATE,
    part_id INTEGER,
    supplier_id INTEGER,
    quantity_received INT,
    unit_price FLOAT(10, 2),
    FOREIGN KEY (part_id) REFERENCES Parts(part_id),
    FOREIGN KEY (supplier_id) REFERENCES Supplier(supplier_id)
);

CREATE TABLE FIFO (
    part_id INTEGER PRIMARY KEY,
    quantity INT,
    FOREIGN KEY (part_id) REFERENCES Parts(part_id)
);
```

Table Preview:

part_id	part_description	sum(part_quantity)
3	Evaporator Fan Motor	23

Final Project Submission – Shopfloor Inventory Management

Data Population

Screenshots for data input are attached below.

The screenshot shows a Google Chrome browser window with the SQLite Online IDE extension open. The URL in the address bar is `sqliteonline.com`. The interface includes a sidebar with database connections (Owner DB, SQLite, MariaDB, PostgreSQL, MS SQL) and a main area for managing tables and executing SQL scripts.

In the main area, the `Parts` table is displayed with the following data:

part_id	part_description	reorder_level_qty
1	Compressor	50
2	Thermostat	50
3	Evaporator Fan Motor	50
4	Condenser	50
5	Water Inlet Valve	50
6	Water Filter	50
7	Evaporator Coil	50
8	Capacitor	50
9	Accumulator	50
10	Supply vent	50

The SQL code used to create and populate the table is:

```
1 INSERT INTO Parts (part_description, reorder_level_qty) VALUES ('Compressor', 50),
2 ('Thermostat', 50),
3 ('Evaporator Fan Motor', 50),
4 ('Condenser', 50),
5 ('Water Inlet Valve', 50),
6 ('Water Filter', 50),
7 ('Evaporator Coil', 50),
8 ('Capacitor', 50),
9 ('Accumulator', 50),
10 ('Supply vent', 50);
11
12 INSERT INTO Department (dept_description) VALUES ('Assembly_1'),
13 ('Assembly_2'),
14 ('Assembly_3');
15 INSERT INTO Supplier VALUES (1,1),(1,2),(1,3),(2,4),(2,5),(2,6),(3,7),(3,8),(4,9),(4,10);
16 SELECT * FROM Parts
```

The right side of the interface shows configuration settings for the browser extension, including options for disabling BIGINT, enabling a left menu, and selecting skins for the editor.

Final Project Submission – Shopfloor Inventory Management

The screenshot shows a Google Chrome browser window with the SQLite Online IDE extension open. The left sidebar lists databases: sqlite-8.db (selected), SQLite, MariaDB, PostgreSQL, and MS SQL. The sqlite-8.db schema includes tables: Department, FIFO, Column, Orders, Parts, Parts_Issue, Shopfloor_Request, sqlite_sequence, Supplier, View, and high_demand_parts. The Orders table has 10 rows. The Shopfloor_Request table has 5 rows. The main area shows the following SQL code and its execution results:

```
1 INSERT INTO Orders (date_received,part_id,supplier_id,quantity_received,unit_price)
2 ('2/3/24', 2, 1, 100,35),
3 ('2/3/24', 3, 1, 100,40),
4 ('2/3/24', 4, 2, 100,20),
5 ('2/3/24', 5, 2, 100,21),
6 ('2/3/24', 6, 2, 100,24),
7 ('2/3/24', 7, 3, 100,35),
8 ('2/3/24', 8, 3, 100,54),
9 ('2/3/24', 9, 4, 100,32),
10 ('2/3/24', 10, 4, 100,23);
11
12 INSERT INTO Shopfloor_Request (request_id, part_id,request_date,part_quantity,dept_
13 (8,9, '3/2/24', 3, 1),
14 (9,9, '3/2/24', 4, 2),
15 (10,8, '3/2/24', 5, 3),
16 (11,1, '3/2/24', 6, 1),
17 (12,3, '3/2/24', 7, 2),
18 (13,3, '3/2/24', 8, 3),
19 (14,3, '3/2/24', 2, 1);
20
21 SELECT * FROM Shopfloor_Request
```

request_id	part_id	request_date	part_quantity	dept_id
1	1	3/2/24	3	1
2	1	3/2/24	4	2
3	2	3/2/24	5	3
4	3	3/2/24	6	1
5	2	3/2/24	7	2

Final Project Submission – Shopfloor Inventory Management

Triggers and Complex Queries

As described earlier, these triggers are run immediately after insertion into Orders table and Parts_Issue table and automatically update quantities of parts in FIFO table.

The screenshot shows a Google Chrome browser window with the SQLite Online IDE extension open. The URL is `sqliteonline.com`. The interface includes a sidebar with database connections (Owner DB, SQLite, MariaDB, PostgreSQL, MS SQL) and a main area for viewing and editing SQL scripts. The current script is `SQLite.7.sql`, which contains two triggers:

```
1 -- Trigger to update FIFO table after inserting new orders
2 CREATE TRIGGER update_fifo_after_orders_insert
3 AFTER INSERT ON Orders
4 FOR EACH ROW
5 BEGIN
6     -- Add new row to FIFO table
7     INSERT INTO FIFO (part_id, quantity)
8     VALUES (NEW.part_id, NEW.quantity_received)
9     ON CONFLICT(part_id) DO UPDATE SET
10         quantity = quantity + NEW.quantity_received;
11 END;
12
13 -- Trigger to update FIFO table after inserting new parts issue
14 CREATE TRIGGER update_fifo_after_parts_issue_insert
15 AFTER INSERT ON Parts_Issue
16 FOR EACH ROW
17 BEGIN
18     -- Subtract issued quantity from FIFO table
19     UPDATE FIFO
20     SET quantity = quantity - NEW.quantity
21     WHERE part_id = NEW.part_id;
22 END;
23
24 SELECT * FROM FIFO
```

Below the code editor, there is a table view showing the contents of the `FIFO` table:

part_id	quantity
1	187
2	175
3	192

The right side of the interface features a sidebar with various configuration options and a history section showing previous queries and their results.

Final Project Submission – Shopfloor Inventory Management

Complex Query – Creation of View

Running this query gives information about high demand parts from Assembly Lines

The screenshot shows a Google Chrome browser window with the SQLite Online IDE extension open. The URL is `sqliteonline.com`. The interface includes a sidebar with database connections (sqlite-8.db, SQLite, MariaDB, PostgreSQL, MS SQL) and a main area for writing and running SQL queries.

The query being run is:

```
1 --complex queries
2 CREATE VIEW high_demand_parts AS SELECT Shopfloor_Request.part_id, part_description,
3 sum(part_quantity) FROM Shopfloor_Request
4 JOIN Parts ON
5 Parts.part_id=Shopfloor_Request.part_id
6 GROUP BY Shopfloor_Request.part_id
7 HAVING part_quantity>1
8 ORDER BY sum(part_quantity) DESC
9
10 SELECT * FROM high_demand_parts
```

The results of the query are displayed in a table:

part_id	part_description	sum(part_quantity)
3	Evaporator Fan Motor	23
1	Compressor	13
2	Thermostat	12
5	Water Inlet Valve	8
9	Accumulator	7
8	Capacitor	5
7	Evaporator Coil	2

On the right side of the interface, there are several configuration sections:

- Sign in**: Buttons for GitHub, Bitbucket, and Local sign-in.
- browser, select version (wasm or asm.js):** A dropdown set to "default".
- Summary**: Buttons for "Disable BIGInt" (checked) and "Left Menu".
- Left Menu**: A checkbox for "DBClick - SELECT Table".
- Editor**: A checkbox for "Disable autocomplete".
- History**: A section showing recent queries and their execution times (e.g., "SELECT * FROM high_demand_parts" at 21:13:17).
- Skins**: A dropdown menu showing "Night" (selected), "Color settings", and "TableFO".
- Format number**: A dropdown menu showing "None" (selected), "Hover cell", and "21:08:02".

Final Project Submission – Shopfloor Inventory Management

Trigger to update the field 'request_closed'

The screenshot shows the SQLite Online IDE interface. On the left, the database schema for 'sqlite-8.db' is displayed, including tables like Department, FIFO, Orders, Parts, and Parts_Issue, along with their columns and triggers. The main area shows the SQL code for the 'update_request_closed' trigger:

```
1 CREATE TRIGGER update_request_closed
2 AFTER INSERT ON Parts_Issue
3 FOR EACH ROW
4 BEGIN
5     UPDATE Parts_Issue
6     SET request_closed = CASE
7         WHEN (SELECT SUM(pi.quantity)
8             FROM Parts_Issue pi
9                 JOIN Shopfloor_Request sr ON pi.request_id = sr.request_id
10                WHERE sr.request_id = NEW.request_id
11                  AND sr.part_id = NEW.part_id)
12                      >= (SELECT sr.part_quantity
13                          FROM Shopfloor_Request sr
14                            WHERE sr.request_id = NEW.request_id
15                              AND sr.part_id = NEW.part_id)
16          THEN 'Yes'
17          ELSE 'No'
18      END
19      WHERE issue_id = NEW.issue_id;
20 END;
21 SELECT * FROM Shopfloor_Request
22 SELECT * FROM Parts_Issue
```

On the right, there are three tabs for history: 'sqlite-8.db.2', 'sqlite-8.db.2', and 'sqlite-8.db.2'. The first tab shows a query to select all from Parts_Issue at 21:44:30. The second tab shows an insert into Parts_Issue at 21:44:12. The third tab shows an insert into Shopfloor_Request.

issue_id	issue_date	request_id	part_id	quantity	request_closed
10	3/2/24	10	8	5	Yes
11	3/2/24	11	1	6	Yes
12	3/2/24	12	2	7	Yes
13	3/2/24	13	2	6	No

Final Project Submission – Shopfloor Inventory Management

Google Chrome File Edit View History Bookmarks Profiles Tab Window Help

Untitled Diagram.drawio - draw.io Announcements - 112334-MI MIS 659 NW1 Relational Data

Wed Mar 6 21:46

sqliteononline.com

File Owner DB Run Export Import Client Sign in

sqlite-8.db
0.1.3 beta

Table
Department
FIFO
Orders
Parts
Parts_Issue
Column
issue_id INTEGER
issue_date DATE
request_id INT
part_id INTEGER
quantity INT
request_closed VARCHAR(3)
Trigger
update_fifo_after_parts_issue...
update_request_closed
Shopfloor_Request
sqlite_sequence
Supplier
View
high_demand_parts

SQLite.8-2.sql
SQLlite.10.sql
sqlite-8.db.1
sqlite-8.db.2

```
1 INSERT INTO Shopfloor_Request (request_id,part_id,request_date,part_quantity,dept_id)
2 VALUES (15,4,'06/03/2024',20,1),
3 (16,5,'06/03/2024',25,2);
4 INSERT INTO Parts_Issue (issue_date,request_id,part_id,quantity) VALUES
5 ('06/03/2024',15,4,12),
6 ('06/03/2024',16,5,25);
7
8 SELECT * FROM Shopfloor_Request
9 SELECT * FROM Parts_Issue
```

issue_id	issue_date	request_id	part_id	quantity	request_closed
12	3/2/24	12	2	7	Yes
13	3/2/24	13	2	6	No
14	3/2/24	14	3	2	Yes
15	06/03/2024	15	4	12	No
16	06/03/2024	16	5	25	Yes

History
Syntax | History

sqlite-8.db.2
SELECT * FROM Parts_Issue 21:44:30

sqlite-8.db.2
INSERT INTO Parts_Issue (issue_date, ('06/03/2024',15,4,12), ('06 21:44:12

sqlite-8.db.2
INSERT INTO Shopfloor_Request (requ

Final Project Submission – Shopfloor Inventory Management

User Interface using Python QWidgets (Python Code document is also submitted)

The screenshot shows the Spyder IDE interface with the following components:

- Left Panel:** A file browser showing three files: `databaseproject.py`, `untitled0.py`, and `DatabaseProjectCode_final.py`. The `DatabaseProjectCode_final.py` file is open, displaying Python code for a PyQt5 application.
- Middle Panel:** A preview window showing the user interface. It has two text input fields: "Dept Description" and "Dept:", and a button labeled "Insert".
- Bottom Panel:** A console window titled "Console 1/A" showing the output of running the script. It includes Python and IPython version information, and the command runfile('...').
- System Tray:** Shows system icons for battery, signal, and volume.
- Status Bar:** Displays "Spyder: Checking for updates", "internal (Python 3.9.14)", "Completions: internal", "LSP: Python", "Line 45, Col 1", "UTF-8", "LF", "RW", and "Mem 74%".

```
from PyQt5.QtWidgets import QApplication, QMainWindow, QLabel, QLineEdit, QPushButton
import sys
import sqlite3

class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()

        # Create labels and input fields for fields
        self.dept_label = QLabel('Dept Description:', self)
        self.dept_label.move(50, 80)
        self.dept_entry = QLineEdit(self)
        self.dept_entry.move(150, 80)

        self.id_label = QLabel('Dept:', self)
        self.id_label.move(50, 110)
        self.id_entry = QLineEdit(self)
        self.id_entry.move(150, 110)

        # Create a button to insert the values
        self.insert_button = QPushButton('Insert', self)
        self.insert_button.move(150, 150)
        self.insert_button.clicked.connect(self.insert_values)

    def insert_values(self):
        # Get the values from the input fields
        dept = self.dept_entry.text()
        id = self.id_entry.text()

        # Connect to the SQLite database
        conn = sqlite3.connect('/Users/indiramallela/sqlite-8.db')
        cursor = conn.cursor()

        # Insert the values into the table
        cursor.execute("INSERT INTO Department (dept_description, dept_id) VALUES (?, ?)", (dept, id))

        # Commit the changes and close the connection
        conn.commit()
        conn.close()

        # Clear the input fields
        self.dept_entry.clear()
        self.id_entry.clear()

if __name__ == '__main__':
    app = QApplication(sys.argv)

    # Create the main window instance
    window = MainWindow()
    window.setGeometry(100, 100, 300, 200) # Set the window size and position
    window.show()
```

Conclusion

This project has helped me gain insights into how database systems are designed, and constraints are built to maintain robustness of data. My understanding on triggers, views, and complex queries has enhanced significantly. This exercise gave me hands on experience to work on real world inspired project and explore ways to automate the processes in database management.