

KETAN MANUBHAI PANCHAL - 2509903

by Ketan Manubhai Panchal

Submission date: 09-Jan-2026 12:19PM (UTC+0000)

Submission ID: 272664138

File name: 66642_Ketan_Manubhai_Panchal_KETAN_MANUBHAI_PANCHAL_-
_2509903_709999_1731366726.docx (132.08K)

Word count: 2324

Character count: 14821

DWR Database Creation Report

Introduction

DragonWave Renewables Ltd (DWR) is a renewable energy supplier that operates fully within the UK, offering services in solar panel installation in England, Scotland, Wales, and Northern Ireland. They employ a diverse clientele, and supply homeowners, businesses, and large scale corporations with customized solar energy systems. With the rapid expansion of the renewable energy sector, DWR has also seen a significant surge in the demand for services. This has compounded the operational data DWR has to handle, which includes, but is not limited to, customer details, installation requests, installer time tables, system specifications, maintenance data, and customer feedback.

Using poorly constructed data management systems, or managing large quantities of data in an unstructured system, could lead to data duplication, disorganization, and overall operational inefficiency. Because of unstructured and poorly designed systems, DWR could risk record loss which could lead to incomplete information being available which could result in delayed installations, inaccurate billing, and a poor customer experience. DWR has operational data management needs a solid, designed, and extensible system. Flexible systems may be scaled to meet changing needs and are much more suitable to support DWR core operational needs, data integrity and system performance. In contemporary systems, an espoused function of a record is that the data be stored securely, be accessible reliably, and be updated efficiently.

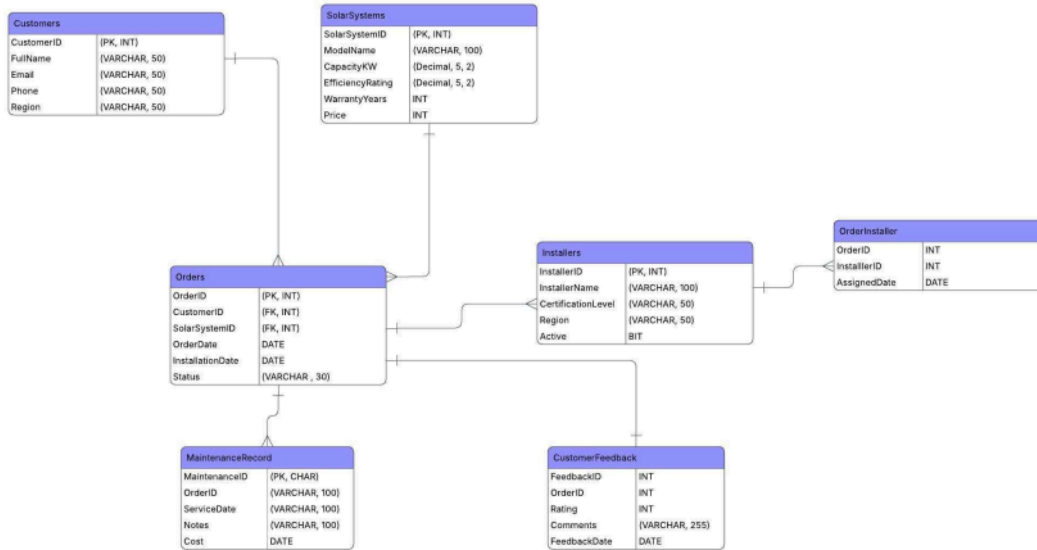
This project focuses on the development of a relational database using Microsoft SQL Server that serves DWR's operational requirements. The database will facilitate customer sign up, order processing, assigning installers, tracking maintenance, and collecting customer feedback. The system also needs to address transactional integrity, auditing, and performance optimization. This case study is relevant as it illustrates the difficulties of a growing company, and showcases how the correct application of database design principles can support the company's objectives [1].

Database Design



The first step taken to develop the database was to look at the DWR business and operational workflows. This helped capture the essential elements of the business which were Customers, Orders, Solar Systems, Installers, Maintenance, Feedback, and supporting tables OrderInstallers and OrderAudit. Each of these elements is a business component analogous to DWR's operation. The Customers table includes the records pertaining to the customer, specifically the customer's name, contact details, and address. A single customer may place multiple orders, creating a one-to-many relation between Customers and Orders. This design allows a customer's details to be stored once and assigned to different orders, avoiding redundancy. The orders table is indicative of the installations the customers want. Each order is linked to one and only one customer and one solar system. Orders also contain operational details such as installation date and order status. Keeping orders separated from customers allows DWR to maintain a transaction history without recreating customer records.

DWR maintains a record of solar panels in the SolarSystems table, which describes the models, details, capacities, efficiencies, warranties, and prices. DWR also avoids the need to record certain repetitive order specifics by keeping solar system data in a separate table. Additionally, this approach allows DWR to modify system specs without impacting the order history. Including the Installers table, the database encapsulates the order-assigning process. Since an order can have multiple installers, and an installer can work on multiple orders, there is a many-to-many relationship between Orders and Installers. This is accomplished with the OrderInstallers table, which contains a composite primary key of OrderID and InstallerID. This is a straightforward approach, which also helps maintain the database in Third Normal Form (3NF) by removing repeating groups and partial dependencies [1]. The Maintenance and Feedback tables provide support for activities that occur after an order is complete. Maintenance enables DWR to complete an inventory of the solar systems and capture expenses and performance of each system. Feedback enables customers to provide post-installation ratings and comments. Both tables link to Orders, ensuring the validity of their records. The database design optimally exhibits the principles of normalization by minimizing redundancy and demonstrating a strong level of referential integrity.



Implementation Documentation

Tables in the database are created and documented in SQL Server Management Studio. Received screenshots explain successful schema deployment. Each table, primary key, and the relationships of foreign keys were defined in the implementation of the system scoped SQL DDL statements.

In the implementation of the system, business rules were enforced at the database level, for example, CHECK constraints limit the region to valid UK locations and the ratings to a specific range. There are UNIQUE constraints on the customer's email address to avoid duplicate records. It is SQL Server that is shown to automatically restrict the entry of bad data, demonstrating the enforcement of constraints.

CREATE TABLE Customers (

CustomerID INT IDENTITY PRIMARY KEY,

FullName NVARCHAR(100) NOT NULL,

Email NVARCHAR(100) UNIQUE NOT NULL,

Phone NVARCHAR(20) NOT NULL,

Region NVARCHAR(50) NOT NULL

CHECK (Region IN ('England', 'Scotland', 'Wales', 'Northern Ireland')),

CreatedAt DATETIME DEFAULT GETDATE()

);

CREATE TABLE SolarSystems (

SolarSystemID INT IDENTITY PRIMARY KEY,

ModelName NVARCHAR(100) NOT NULL,

CapacityKW DECIMAL(5,2) NOT NULL CHECK (CapacityKW > 0),

EfficiencyRating DECIMAL(4,2) CHECK (EfficiencyRating BETWEEN 10 AND 30),

WarrantyYears INT CHECK (WarrantyYears BETWEEN 5 AND 30),

Price DECIMAL(10,2) NOT NULL CHECK (Price > 0)

);

CREATE TABLE Installers (

InstallerID INT IDENTITY PRIMARY KEY,

InstallerName NVARCHAR(100) NOT NULL,

CertificationLevel NVARCHAR(50) NOT NULL,

```
    Region NVARCHAR(50) NOT NULL,  
  
    Active BIT DEFAULT 1  
);
```



```
CREATE TABLE Orders (  
  
    OrderID INT IDENTITY PRIMARY KEY,  
  
    CustomerID INT NOT NULL,  
  
    SolarSystemID INT NOT NULL,  
  
    OrderDate DATE DEFAULT GETDATE(),  
  
    InstallationDate DATE NOT NULL,  
  
    Status NVARCHAR(30) NOT NULL  
  
    CHECK (Status IN ('Pending', 'Installed', 'Cancelled')),  
  
    CONSTRAINT FK_Order_Customer FOREIGN KEY (CustomerID)  
  
        REFERENCES Customers(CustomerID),  
  
    CONSTRAINT FK_Order_Solar FOREIGN KEY (SolarSystemID)  
  
        REFERENCES SolarSystems(SolarSystemID)  
);
```

```
CREATE TABLE OrderInstallers (  

```

```
OrderID INT NOT NULL,  
  
InstallerID INT NOT NULL,  
  
AssignedDate DATE DEFAULT GETDATE(),  
  
PRIMARY KEY (OrderID, InstallerID),  
  
FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),  
  
FOREIGN KEY (InstallerID) REFERENCES Installers(InstallerID)  
);
```

```
CREATE TABLE Maintenance (  
  
MaintenanceID INT IDENTITY PRIMARY KEY,  
  
OrderID INT NOT NULL,  
  
ServiceDate DATE NOT NULL,  
  
Notes NVARCHAR(255),  
  
Cost DECIMAL(10,2) CHECK (Cost >= 0),  
  
FOREIGN KEY (OrderID) REFERENCES Orders(OrderID)  
);
```

```
CREATE TABLE Feedback (  
  
FeedbackID INT IDENTITY PRIMARY KEY,
```

```

OrderID INT UNIQUE NOT NULL,

Rating INT CHECK (Rating BETWEEN 1 AND 5),

Comments NVARCHAR(255),

FeedbackDate DATE DEFAULT GETDATE(),

FOREIGN KEY (OrderID) REFERENCES Orders(OrderID)

);

```

Each table was populated with sample data to verify the schema and constraint validity. The records are created to reflect plausible business operations that include customers from all areas, different models of solar systems, multiple orders of installations, and orders of maintenance. Successful INSERT statements and SELECT queries capture that the database retains and retrieves data while respecting the rules of integrity.

```

INSERT INTO Customers (FullName, Email, Phone, Region) VALUES

```

```

('Alice Brown', 'alice@dwr.co.uk', '0711111111', 'England'),

```

```

('John Smith', 'john@dwr.co.uk', '0722222222', 'Scotland'),

```

```

('Sara Khan', 'sara@dwr.co.uk', '0733333333', 'Wales'),

```

```

('Mark Lee', 'mark@dwr.co.uk', '0744444444', 'England'),

```

```

('Emma Jones', 'emma@dwr.co.uk', '0755555555', 'Northern Ireland');

```

```

INSERT INTO SolarSystems (ModelName, CapacityKW, EfficiencyRating, WarrantyYears, Price)
VALUES

```

```

('SolarMax A1', 4.5, 18.5, 15, 6500),

```

('SolarMax B2', 6.0, 20.0, 20, 8200),

('EcoSun X', 3.2, 17.0, 10, 5200),

('PowerGrid Pro', 8.0, 22.5, 25, 11500),

('HomeLite S', 5.0, 19.0, 15, 7000);

INSERT INTO Installers (InstallerName, CertificationLevel, Region) VALUES

('Tom Installer', 'Level A', 'England'),

('Jane Green', 'Level B', 'Scotland'),

('Ali Hassan', 'Level A', 'Wales'),

('Peter Long', 'Level C', 'England'),

('Nina Patel', 'Level B', 'Northern Ireland');

INSERT INTO Orders (CustomerID, SolarSystemID, InstallationDate, Status) VALUES

(1, 1, '2026-02-01', 'Pending'),

(2, 2, '2026-02-05', 'Installed'),

(3, 3, '2026-02-10', 'Pending'),

(4, 4, '2026-02-15', 'Installed'),

(5, 5, '2026-02-20', 'Pending');

INSERT INTO OrderInstallers (OrderID, InstallerID) VALUES

(1,1),(2,2),(3,3),(4,4),(5,5);

INSERT INTO Maintenance (OrderID, ServiceDate, Notes, Cost) VALUES

(2,'2026-06-01','Routine check',120),

(4,'2026-07-01','Inverter replacement',450),

(2,'2026-12-01','Panel cleaning',80),

(4,'2026-12-10','System diagnostics',150),

(2,'2027-01-15','Firmware update',60);

INSERT INTO Feedback (OrderID, Rating, Comments) VALUES

(2,5,'Excellent service'),

(4,4,'Very professional'),

(1,5,'Smooth process'),

(3,4,'Good communication'),

(5,3,'Installation delayed');

Triggers and stored procedures were also tested during implementation. The OrderAudit trigger automatically records changes to order status, creating a history of updates for auditing purposes. Screenshots showing trigger execution confirm that audit records are generated

whenever an order's status is modified. Stored procedures were executed with both valid and invalid input values to demonstrate successful transactions and rollback behaviour.

```
CREATE TRIGGER trg_OrderStatusAudit  
  
ON Orders  
  
AFTER UPDATE  
  
AS  
  
BEGIN  
  
    INSERT INTO OrderAudit (OrderID, OldStatus, NewStatus)  
  
    SELECT i.OrderID, d.Status, i.Status  
  
    FROM inserted i  
  
    JOIN deleted d ON i.OrderID = d.OrderID  
  
    WHERE i.Status <> d.Status;  
  
END;
```

Stored Procedures and Transactional Scenario

Stored procedures were developed to encapsulate business logic and ensure consistent data manipulation. Procedures such as sp_CreateOrder and sp_UpdateOrderStatus handle common operations in a controlled manner. By using stored procedures, DWR can reduce the risk of inconsistent data updates and improve security by limiting direct table access.

```
CREATE PROCEDURE sp_CreateOrder  
  
    @CustomerID INT,
```

```

    @SolarSystemID INT,
    @InstallationDate DATE
AS
BEGIN
    BEGIN TRY
        BEGIN TRANSACTION;

        INSERT INTO Orders (CustomerID, SolarSystemID, InstallationDate, Status)
        VALUES (@CustomerID, @SolarSystemID, @InstallationDate, 'Pending');

        COMMIT TRANSACTION;
    END TRY
    BEGIN CATCH
        ROLLBACK TRANSACTION;

        THROW;
    END CATCH;
END;

CREATE PROCEDURE sp_UpdateOrderStatus

```

```
@OrderID INT,  
  
@NewStatus NVARCHAR(30)  
  
AS  
  
BEGIN  
  
BEGIN TRY  
  
BEGIN TRANSACTION;  
  
  
UPDATE Orders  
  
SET Status = @NewStatus  
  
WHERE OrderID = @OrderID;  
  
  
COMMIT TRANSACTION;  
  
END TRY  
  
BEGIN CATCH  
  
ROLLBACK TRANSACTION;  
  
THROW;  
  
END CATCH;  
  
END;
```

A key part of this project is the transactional scenario implemented in the `sp_ProcessInstallationOrder` stored procedure. This scenario reflects a real business process in which a customer places an order, an installer is assigned, and the order status is updated. These operations are interdependent and must be treated as a single unit of work.

```
CREATE PROCEDURE sp_ProcessInstallationOrder
```

```
@CustomerID INT,
```

```
@SolarSystemID INT,
```

```
@InstallerID INT,
```

```
@InstallationDate DATE
```

```
AS
```

```
BEGIN
```

```
SET NOCOUNT ON;
```

```
BEGIN TRY
```

```
BEGIN TRANSACTION;
```

```
INSERT INTO Orders (CustomerID, SolarSystemID, InstallationDate, Status)
```

```
VALUES (@CustomerID, @SolarSystemID, @InstallationDate, 'Pending');
```

```
DECLARE @OrderID INT = SCOPE_IDENTITY();
```

INSERT INTO OrderInstallers (OrderID, InstallerID)

VALUES (@OrderID, @InstallerID);

UPDATE Orders

SET Status = 'Installed'

WHERE OrderID = @OrderID;

COMMIT TRANSACTION;

END TRY

BEGIN CATCH

-- Any error → rollback everything

ROLLBACK TRANSACTION;

THROW;

END CATCH

END;

EXEC sp_ProcessInstallationOrder

```

@CustomerID = 1,

@SolarSystemID = 2,

@InstallerID = 1,

@InstallationDate = '2026-03-01';

```

EXEC sp_ProcessInstallationOrder

```

@CustomerID = 1,

@SolarSystemID = 2,

@InstallerID = 999, --Invalid

@InstallationDate = '2026-03-01';

```



Control of transactions takes place using `transaction_control` as well as using the commands `BEGIN TRANSACTION`, `COMMIT`, and `ROLLBACK` within a `TRY-CATCH` block. If all operations succeed then the transaction is committed and the changes become permanent. An example of a failing operation is assigning an invalid installer. In such a case, the system will rollback the transaction to ensure that partial changes to the data will not be saved. This prevents an order from existing without an installer being assigned to it.

This is an example of the principle of atomicity, which is one of the main ACID properties of database transactions. The reason being is that the database will ensure that all of the operations fully succeed, or none are performed at all, which is the attribute of the principle of atomicity. This characteristic will maintain the database's consistency and reliability even when there are errors or simultaneous access to the database [2].

Performance Tuning Enhancement

In the case of DWR, as the company continues to expand, it has to ensure the database continues to perform optimally. Critical to daily operations are high-frequency queries like checking orders for a specific customer and filtering orders by customer. As data volume increases, such queries, without optimization, would become exceedingly slow and highly resource-consuming.

In order to achieve improved performance, secondary indexes were established on the frequently accessed columns, specifically the customer IDs in the orders, the order statuses, and the maintenance order IDs. With these indexes, SQL Server is able to perform index seeks as opposed to the more time-consuming and resource-consuming full table scans, which in turn reduces the disk I/O and the time taken to execute queries.

Based on the anticipated usage patterns, the specific customer service indexed columns were quite effective. Staff members order histories are frequently retrieved by customer, while operational members breakdown orders based on status for scheduling, Maintenance records. Accessing OrderID is almost always completed by order identifier, proving the indexing quite effective, even in the case of Microsoft documentation which states that appropriate indexing directs performance improvement, especially in read-heavy systems [3].

The negative aspects of JDW having developed indexes are that it takes more time to data modify operations. However, there are more positives than negative for DWR operational workload being data modification operational indexes developed as operational workload does and index gaps enables maintaining the database responsiveness.

```
CREATE INDEX idx_Orders_CustomerID ON Orders(CustomerID);  
CREATE INDEX idx_Orders_Status ON Orders(Status);  
CREATE INDEX idx_Maintenance_OrderID ON Maintenance(OrderID);
```

Conclusion and Reflection

The customized database solution for DragonWave Renewables Ltd fully satisfies the operational needs of the company. Systems of this nature, as a result of careful construction, maintain real-world business relationship modeling, robust referential integrity, and redundancy

removal via normalization. Constraints, triggers, stored procedures, and transaction control are employed for data accuracy, consistency, and reliability.

I can say this project improved my grasp of some of the concepts entwined in the creation and the structure of the implementation of the database. One of the main challenges was the alignment of the schema and the business needs so that I was able to create a logically usable and fully normalized schema. This was solved via some intensive schema detail breakdown and then some associative table construction. Designing transaction fully demonstrated the business processes that needed to be understood when coding the system. I hope to include the design of reporting views, the construction of more complex systems of role-based access control, and the design of more complex systems of proactive performance monitoring. The worth of the database can also be enhanced through the construction of systems that allow the database to interact with outer frameworks and outer systems of analytical tools. This project demonstrates the construction of a database system and the tools that help enhance the efficiency, scalability, and sustainability of a business.

References

- [1] Elmasri, R. (2008). Fundamentals of database systems. Pearson Education India.
- [2] Murnawana, S. L., & Mardianac, S. (2019). Design of the Business Intelligence System at the Co&Co Training Center. system, 6(5).
- [3] Microsoft, "SQL Server Index Architecture and Design Guide," Microsoft Learn, 2023.



FINAL GRADE

51 / 100

GENERAL COMMENTS

The SQL code meets many core requirements: a working schema with constraints, a workable ERD, stored procedures with transaction handling, and performance indexes. The main shortcomings are in the written report and academic presentation. Referencing is poor and does not support the work. The report structure is basic and lacks details, with missing visual evidence from the implementation phase. The student shows they can build the system but does not fully articulate the reasoning behind their choices.

PAGE 1

PAGE 2



Comment 1 | Knowledge & Understanding(20)

The database design is logical. Tables, primary keys, foreign keys, and a composite key are correctly implemented. The purpose of constraints, triggers, and stored procedures is understood and demonstrated in the SQL code.

PAGE 3



Comment 2 | Knowledge & Understanding(20)

The explanation of design decisions is brief. The report states the design is in 3NF but does not demonstrate the normalization process or explain how transitive dependencies were removed. The understanding is shown through implementation but not elaborated in writing.

PAGE 4

PAGE 5



Comment 3 | Transferable Skills(20)

The assessment specification explicitly requires "Implementation Documentation" to "Include screenshots of..." the various build steps and

PAGE 6



Comment 4 | Cognitive & Intellectual (20)

The code does not show the process of normalizing to 3NF. It presents a final schema but does not document the steps taken to eliminate redundancy and transitive dependencies.

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11



Comment 5 | Cognitive & Intellectual (20)

Justifications lack specificity . For example, the choice of indexes is listed but not critically analysed—there is no discussion of trade-offs impact on INSERT/UPDATE speed. The decision to make `Feedback.OrderID` UNIQUE is implemented but not logically justified in the business context. Detailed critical evaluation of alternative designs or solutions is absent.

PAGE 12

PAGE 13

PAGE 14

PAGE 15



Comment 6 | Practical Application(30)

While screenshots are mentioned, they are not included in the submitted document, only the ERD image. There is no visual evidence of constraint testing, sample data insertion, or trigger execution as required by the "Implementation Documentation" section.

PAGE 16

PAGE 17

PAGE 18



Comment 7 | Engagement with Literature(10)

Source [1] is cited in the text but is used only once to support a basic point about normalization.

Source [2] is irrelevant (discusses Business Intelligence, not core database concepts).

Source [3] is appropriate for indexing but is only mentioned in passing, not used to justify choices.

GRADING FORM: DC-DBS

KETAN MANUBHAI PANCHAL

51

ENGAGEMENT WITH LITERATURE(10)

References to the literature to discuss the importance of the concepts of rollback, commit, functions, triggers, procedures, transactions and indexes.



A reference list is provided but is insufficient and applied incorrectly. There is no systematic use of literature to explain or justify the application of key concepts such as transactions (ACID), triggers, stored procedures, or normalization decisions. The requirement to use sources to discuss concepts like rollback and commit is not met.

5

KNOWLEDGE & UNDERSTANDING(20)

ERD



12

COGNITIVE & INTELLECTUAL (20)

3NF, Creating tables with keys and populating database

-  Theory is applied to a practical case. A transactional stored procedure is created with a clear rollback condition. There is an attempt to justify indexing choices based on expected use.

11

PRACTICAL APPLICATION(30)

Implementation of Database design

-  A complete schema is provided with all required SQL: table creation, constraints (CHECK, UNIQUE, FOREIGN KEY), sample data, a trigger for auditing, and multiple stored procedures with explicit transaction control (BEGIN, COMMIT, ROLLBACK). The transaction scenario correctly includes a rollback test with an invalid installer.

17

TRANSFERABLE SKILLS(20)

• Readability of the report. • Written quality, including spelling and grammar and the use of appropriate formal language. • Structure and presentation of the report. • Critical evaluation of the selected database methods, tools and techniques and their deployment in a multi-user database environment.

-  The document is a text file with an embedded image; it does not follow professional report formatting (no title page, table of contents, consistent heading styles, or page numbers). The

6

ERD is included but is of low quality and lacks clear labels for cardinalities (e.g., 1:M, M:N).
Communication of technical decisions is present but brief.