

AdventureWorks Cycles' Database Design & Implementation

by Matthew Hatherly

Submission date: 08-Jan-2025 09:50PM (UTC+0000)

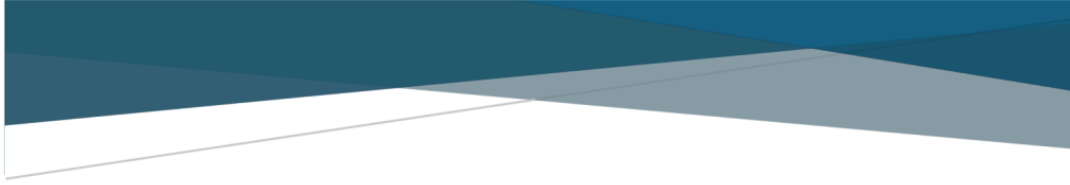
Submission ID: 247700288

File name:

49459_Matthew_Hatherly_AdventureWorks_Cycles_Database_Design_Implementation_621315_1669948103.docx
(865.21K)

Word count: 1652

Character count: 8357



DATABASES AND APPLICATION DEVELOPMENT

AdventureWorks Cycles' Database Design &
Implementation (ACCA5026)

Matthew Hatherly (2306287)
2306287@student.uwtsd.ac.uk

Abstract

AdventureWorks is a renowned bicycle manufacturer, and the purpose of this case study is to create an appropriate database solution to ensure peak productivity for their online marketplace. This will be achieved through innovation, modern operation techniques and discovering new methods to improve customer reach. With the successful application of this case study, AdventureWorks will be able to meet their ambitious goals.

Table of Contents

Entity Relationship Diagram	3
Database Design	3
Customer	4
Product	5
Manufacturer.....	6
Category	7
Product Order.....	8
Order Details.....	9
Rollback and Commit.....	9
Performance Tuning Advancement	10
Conclusion	10
References	11

Figure 1: AdventureWorks Entity Relationship Diagram	3
Figure 2: Database initial creation code	3
Figure 3: Code from creating customer table.....	4
Figure 4: Records to be inserted into customer table	4
Figure 5: Code from creating Product table.....	5
Figure 6: Records to be inserted in Products table	5
Figure 7: Code from creating Manufacturer table	6
Figure 8: Records to be inserted in Manufacturer table.....	6
Figure 9: Code from creating Category table.....	7
Figure 10: Records to be inserted in Category table	7
Figure 11: Code for creating the Product Order table	8
Figure 12: Records for Product Order table.....	8
Figure 13: Code for creating Order Details table	9

Figure 14: Added transaction through single commit and rollback..... 9



Entity Relationship Diagram

Practical Element

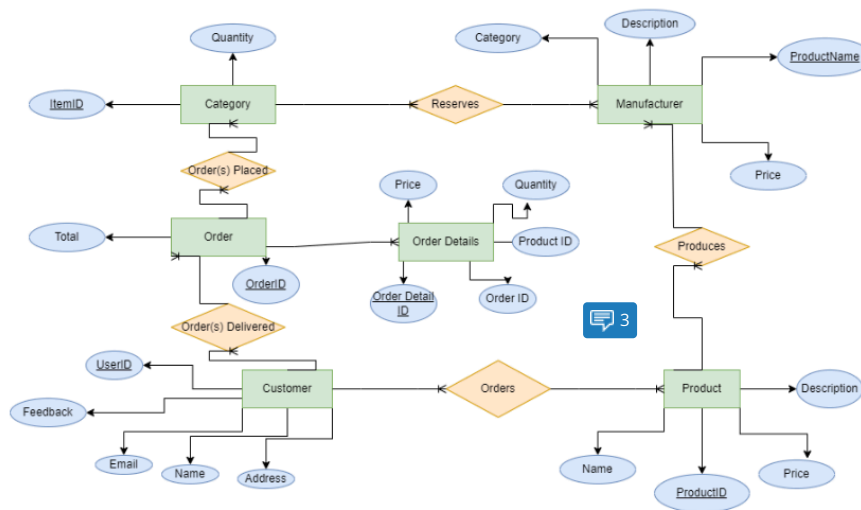


Figure 1: AdventureWorks Entity Relationship Diagram

The figure above shows an entity relationship diagram (ERD) which offers some insight into where the database will lead for AdventureWorks. Effectively, per HubGem [1], an ERD is “a type of flowchart that illustrates how “entities” such as people, objects or concepts relate to each other within a system.” In terms of the overall plan for the database, the customer will sign up and receive a User ID, which will be the primary key. When an order is placed, product information will be sorted, such as the Product ID, name and price. The manufacturer will then be aware of the order, and based on the category of the item ordered, whether it be a bicycle or an accessory, the manufacturer can reserve the order and place it for delivery to the address stated by the customer. When things such as the quantity, payment and order ID are confirmed, the order will be delivered to the customer, and they can offer feedback from their experience. Finally, there is an order details table which allows all prior order information to be viewed in a simpler way containing foreign keys for the product information.

```
1 • CREATE DATABASE awcycles;
2 • USE awcycles;
```

Figure 2: Database initial creation code

Database Design

The figure above shows the initial code for creating the AdventureWorks cycles database. From this point, tables can be created, and customer data can start to be implemented into the tables.

Customer

```
4 -- shows customer information
5 CREATE TABLE customer (
6   user_id INT AUTO_INCREMENT PRIMARY KEY,
7   first_name VARCHAR(50) NOT NULL,
8   last_name VARCHAR(50) NOT NULL,
9   customer_email VARCHAR(70) NOT NULL,
10  customer_address VARCHAR(70) NOT NULL,
11  customer_feedback VARCHAR(100) NOT NULL CHECK (customer_feedback IN ('perfect', 'fantastic', 'great', 'good', 'average', 'poor'))
12 );
```

Figure 3: Code from creating customer table

Above shows the code used for the customer table. Collecting customer information is important for the better running of AdventureWorks, because per OmniConvert [2], “with so many competitors fighting for your ideal customers’ attention, one of the few differentiators is represented by the customer experiences you build around your products and services.” In this table, the primary key is User ID, which is given to each user making a purchase with AdventureWorks cycles. Subsequently, some more user information will be stored, such as the first name, last name, customer email address, address, and feedback after purchase.

It is important to store this user information so AdventureWorks knows who made the purchase, and so they can send the item while having all the information necessary. In the customer feedback attribute, there is a check constraint, which allows the user to only select within a certain parameter of words, which avoids potential risks e.g. misspelling of words.

```
13
14 INSERT INTO customer (user_id, first_name, last_name, customer_email, customer_address, customer_feedback)
15 VALUES
16 ('1', 'John', 'Doe', 'johndoe@gmail.com', '7 Green Street', 'perfect'),
17 ('2', 'Jane', 'Smith', 'janesmith@gmail.com', '5 Red Street', 'fantastic'),
18 ('3', 'Michael', 'Brown', 'michaelbrown@gmail.com', '12 Blue Avenue', 'great'),
19 ('4', 'Bradley', 'Evans', 'bradevans@gmail.com', '1 Park Avenue', 'good');
20
```

Figure 4: Records to be inserted into customer table



The information above is a handful of realistic records for customers making purchases at AdventureWorks. As seen in the corresponding image, it gives each customer a User ID, as well as displaying their name, email, address and their review after purchasing.

Product

```
21 -- shows product information
22 • CREATE TABLE product (
23     product_id INT AUTO_INCREMENT PRIMARY KEY,
24     product_name VARCHAR(50) NOT NULL,
25     product_description VARCHAR(100) NOT NULL,
26     product_price INT CHECK(product_price BETWEEN 0 AND 1000),
27     is_available BOOLEAN DEFAULT TRUE,
28     category_id INT,
29     manufacturer_id INT,
30     FOREIGN KEY (category_id) REFERENCES category(item_id),
31     FOREIGN KEY (manufacturer_id) REFERENCES manufacturer(manufacturer_id)
32 );
```

Figure 5: Code from creating Product table

As seen above, the product table has the primary key of the product ID, which is the key identified to what product is being discussed. Each product having its own ID makes it easier to identify individual products. Other information stored in the product database is the product name, a brief description, the total price of the product, whether the product in question is available, and foreign keys related to both the category ID and manufacturer ID. These foreign keys will make it easier when it comes to linking to tables further on in the database.

This table is an important one for AdventureWorks as it stores their entire product information, subsequently, creating a more optimised system for their day-to-day operations. Per Salsify [3], “without a product information database, businesses run the risk of distributing inaccurate or outdated product information, which can cause customers to move on from the competition. A product information database is a tool for optimizing product information to build customer loyalty and drive sales.”

```
34 • INSERT INTO product (product_id, product_name, product_description, product_price)
35 VALUES
36 ('1', 'Bike', 'Road Bike', '£100'),
37 ('2', 'Bike', 'Mountain Bike', '£350'),
38 ('3', 'Accessory', 'Bike Lock', '£20'),
39 ('4', 'Bike', 'Road Bike', '£100');
```

Figure 6: Records to be inserted in Products table

The records inserted into the products table as seen above, are purchases of 4 items, including a unique ID, the name of each product, a short description to distinguish the product, and the total price of each product in question.

Manufacturer

```
41  -- gives manufacturer info
42  • CREATE TABLE manufacturer (
43      product_name INT AUTO_INCREMENT PRIMARY KEY,
44      product_price INT CHECK(product_price >=0),
45      product_description VARCHAR(100) NOT NULL,
46      product_category VARCHAR(50) NOT NULL
47  );
```

Figure 7: Code from creating Manufacturer table

As seen above, the primary key for the manufacturer table is the product name. It also includes information of the price of the product, as well as a brief description and the category of the product. The purpose of this table is to show the products that are created by the manufacturer, displaying a brief description of each product that is for sale by AdventureWorks.

```
49  • INSERT INTO manufacturer (product_name, product_price, product_description, product_category)
50  VALUES
51  ('Bike', '£100', 'Road Bike', 'Product for sale'),
52  ('Bike', '£350', 'Mountain Bike', 'Product for sale'),
53  ('Accessory', '£20', 'Bike Lock', 'Product for sale'),
54  ('Bike', '£100', 'Road Bike', 'Product for sale');
```

Figure 8: Records to be inserted in Manufacturer table

The above table highlights some realistic records displaying the primary key, which is the product name, as well as the product price, a brief description of the product and the sale category of the product.

Category

```
56  -- stores product categories and quantities
57  • CREATE TABLE category (
58      item_id INT AUTO_INCREMENT PRIMARY KEY,
59      item_quantity INT CHECK(item_quantity >=0)
60  );
```

Figure 9: Code from creating Category table

The figure above shows the code to create the category table, with the item ID being the primary key and the table also displays the item quantity. Being able to see the ID allows whoever is viewing the database to identify the item in question, and the quantity option allows the user to see the quantity of products purchased.

```
62  • INSERT INTO category (item_id, item_quantity)
63  VALUES
64  ('1', '1'),
65  ('2', '1'),
66  ('3', '2'),
67  ('4', '1');
```

Figure 10: Records to be inserted in Category table

The image above shows 4 unique item ID's, which are the primary keys, followed by the item quantity.

Product Order

```
69  -- gives product order details
70  • CREATE TABLE productorder (
71      order_id INT AUTO_INCREMENT PRIMARY KEY,
72      order_total INT CHECK(order_total >=0),
73      user_id INT,
74      FOREIGN KEY (user_id) REFERENCES customer(user_id)
75  );
```

Figure 11: Code for creating the Product Order table

The figure above shows the order ID, which will be the primary key, followed by the order total, which is the total financial cost for the order in question, as well as the user ID to link who bought the product in line with the order ID. A foreign key is used to link the User ID with the customer table.

```
77  • INSERT INTO productorder (order_id, order_total)
78      VALUES
79      ('1', '100'),
80      ('2', '350'),
81      ('3', '40'),
82      ('4', '100');
```

Figure 12: Records for Product Order table

The records above show the order ID as well as the total cost of the order.

Order Details

```
84  -- final order details after purchase
85  • CREATE TABLE order_details (
86      order_detail_id INT AUTO_INCREMENT PRIMARY KEY,
87      order_id INT NOT NULL,
88      product_id INT NOT NULL,
89      quantity INT CHECK(quantity > 0),
90      price INT CHECK(price >= 0),
91      FOREIGN KEY (order_id) REFERENCES productorder(order_id),
92      FOREIGN KEY (product_id) REFERENCES product(product_id)
93  );
```

Figure 13: Code for creating Order Details table

The table above shows the final order details after the purchase is complete. The primary key is a unique order detail ID, and this table will also show the order ID, the product ID, the quantity of the product, and the total price of the order. The foreign keys for this table will be the order ID from the product order table and the product ID from the product table

Rollback and Commit

```
95  • START TRANSACTION;
96
97  • INSERT INTO customer (first_name, last_name, customer_email, customer_address, customer_feedback)
98  VALUES ('Rob', 'Davies', 'rdavies@gmail.com', '52 Garden Terrace', 'great');
99
100 • SAVEPOINT before_update;
101
102 • INSERT INTO customer (first_name, last_name, customer_email, customer_address, customer_feedback)
103 VALUES ('Ashley', 'Hughes', 'ashhughes@gmail.com', '30 Purple Avenue', 'good');
104
105 • ROLLBACK TO before_update;
106
107 • COMMIT;
```

Figure 14: Added transaction through single commit and rollback

As seen in the above instance, a transaction was started, and records were added to the customer table. From then, a save point was added before another record was added. The commit then allows the rollback to take place to the point where the save point is set.

Performance Tuning Advancement

In terms of the AdventureWorks Cycles database, some of the most common queries could be product searches in terms of reviewing availability as well as the reviewing stock levels. There is also the order details table which could be used to review order previous order information, potentially if there is an issue with the order in the future, and there can also be order summaries linked to each individual user id, allowing whoever is reviewing the database to see an individual's entire history of purchases, as well as reviewing specific customer information, potentially if an amendment has to be made, such as a change of address.

Some secondary indexes that could prove useful for this database could be for:

- Customer email.
- Product orders.
- Category.



These columns should be indexed as it offers an opportunity to improve efficiency when reviewing the database. For instance, an indexed customer email column could allow whoever is reviewing the database to quickly see customer information and verify that they're looking at the right customers information. An indexed product orders tab would allow those reviewing the database to see the entire order history of each individual customer in a much quicker way. Finally, an indexed customer column would allow the user of the database to quickly review stock levels and make any decision a lot quicker. All of these circumstances could allow for the better running of the database.

Indexes can help improve general efficiency of the database, however, it is also important that they are well maintained and not overused so they don't prove to be a detriment, which can be caused by having too many indexes, or indexes that aren't completely necessary, as a specific search may not be performed as much as another. Per OrangeMatter [4], any column that should ideally be indexed are "any columns referenced in the where and order by clauses of commonly executed queries. Those columns will benefit the most, as that's where the initiation work in data retrieval is done.

Conclusion

Considering the entity relationship diagram being the backbone of the SQL server and its corresponding tables, this gives AdventureWorks a good backbone to manage their operations effectively. The foreign keys implemented into table queries should allow for smoother movement between tables, and properly placed indexes can improve operations even further.

References

- [1] M. Sullivan, "The Importance of Entity Relationship Diagrams," HubGem.
- [2] A. Panaitescu, "8 Benefits of Collecting Customer Data: Why Customer Data Is Important," OmniConvert, 2024.
- [3] "Product Information Database," Salsify.  9
- [4] J. D'Antoni, "Why Does My Database Need Indexes?," 2021.

AdventureWorks Cycles' Database Design & Implementation

GRADEMARK REPORT

FINAL GRADE

GENERAL COMMENTS

41 /100

PAGE 1

PAGE 2



Comment 1 | Transferable Skills

ToC should provide descriptive titles for each section to ensure the reader understands what each part covers.

PAGE 3



Comment 2 | Transferable Skills

could benefit from a more engaging introduction as opening

PAGE 4



Comment 3 | Knowledge&Understanding Skills

entities have clear and well-defined primary keys

PAGE 5



Comment 4 | Practical Application Skills

well implemented however evidence of the values inserted into table could be shown

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10



Comment 5 | Practical Application Skills

testing constraints missing



Comment 6 | Cognitive & Intellectual Skills

"In addition to explaining the code, it's important to highlight the significance of using rollback and commit

PAGE 11



Comment 7 | Practical Application Skills

implementation of indexing and performance testing to be included



Comment 8 | Transferable Skills

Conclusion briefly summarizes some of the key points of the report

PAGE 12



Comment 9 | Literature Skills

literature review lacks in-depth analysis

GRADING FORM: DAD_2024

MATTHEW HATHERLY

41

LITERATURE SKILLS



2

KNOWLEDGE&UNDERSTANDING SKILLS



9

COGNITIVE &INTELLECTUAL SKILLS



9

PRACTICAL APPLICATION SKILLS



10

TRANSFERABLE SKILLS



11