Swaddle Swap Database Design

What type of Database?

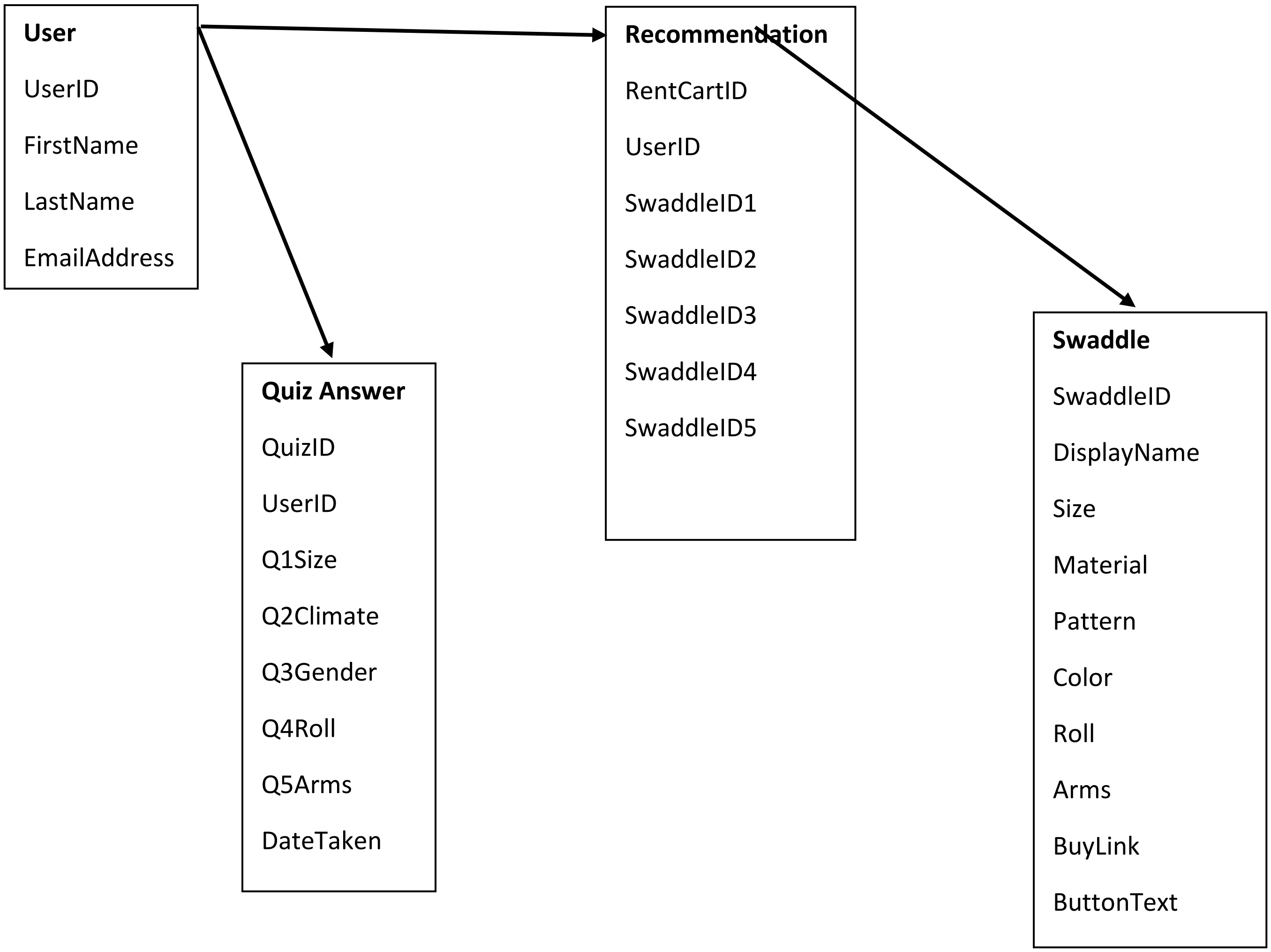
Given that Swaddle Swap is an ecommerce site that will have many visitors and many SKUs on the site, it is best suited for a relational database. There will be several one to many and many to many relationships so this type of database can scale more easily. I am going to use AWS' DynamoDB for the website. DynamoDB is serverless and was meant to scale with large datasets.

For the MVP, DynamoDB may be too "big" but as the site scales and if we decide to build in more trial type items for young children, we will need something that will easily scale with the site.

Relational Databases

See below for the different tables and the relationships between them. The first table, User is the main table and serves to connect most of the other tables in the database. The second table, quiz answer will store the answers to the swaddle quiz and will have UserID so that we can easily link together tables. The third table, Swaddle is for the swaddles on the site and will have data in it that can make it easily filtered and sorted, not only for the quiz but for later when we let users shop the site without the quiz. The last table for the MVP will be the recommendation table. This table will have UserID and RecommendationID and will be populated with the SwaddleIDs that match the quiz answers.

(Stretch)The rental cart is the table that holds all the swaddles that are to be rented. This way we can store the swaddles by SwaddleID in this cart. In the swaddle table, the "Status" field will be updated based on if the swaddle is rented, purchased or available. For the Rent Cart, we need the Rental Date so that we know the return date which will automatically calculate to 14 days after the rental date. Lastly, the payment table will be connected by User ID and will serve to collect the payment information of the user. This table will be populated from the checkout screen. With our stretch goal of having a return flow, the database will use the rental cart and update the swaddle status based on what the user chooses on checkout. For example, if the swaddle is rented and they decide to return, the swaddle status will update to available.



User

- UserID
- FirstName
- LastName
- EmailAddress

Recommendation

- RentCartID
- UserID
- SwaddleID1
- SwaddleID2
- SwaddleID3
- SwaddleID4
- SwaddleID5

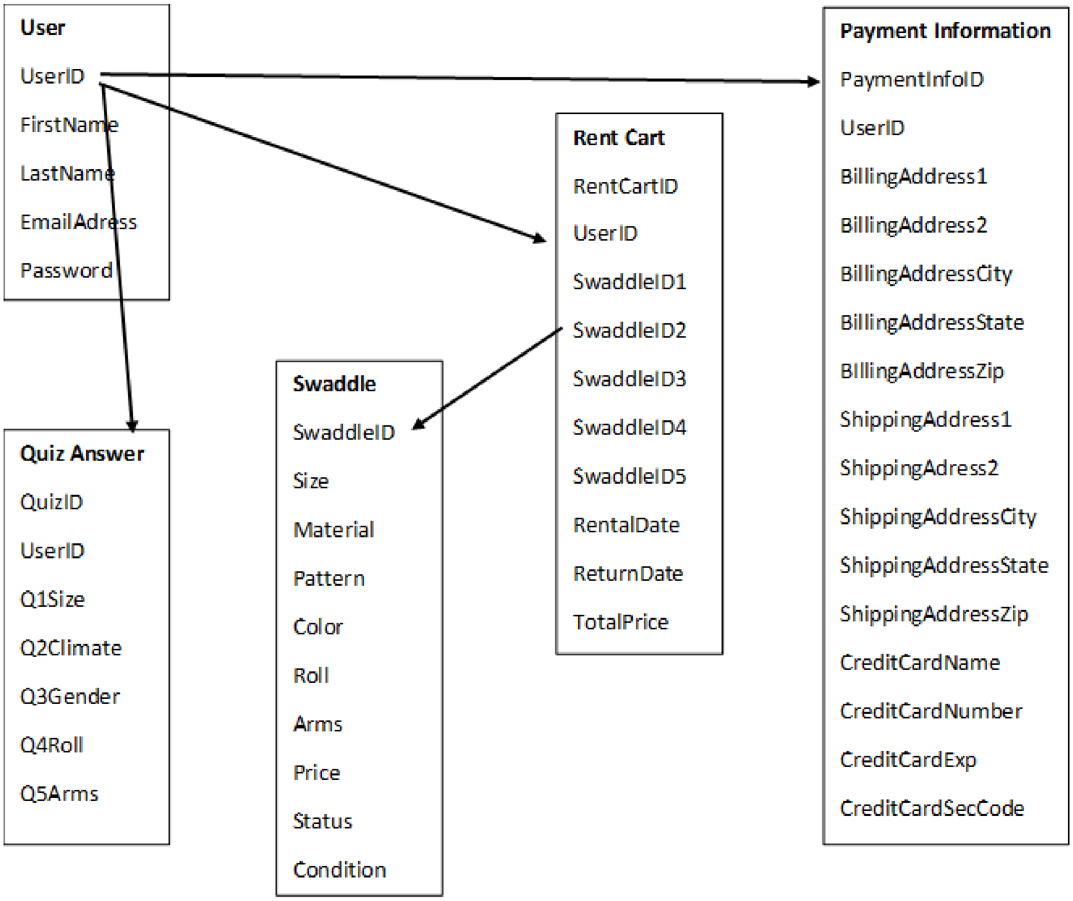
Quiz Answer

- QuizID
- UserID
- Q1Size
- Q2Climate
- Q3Gender
- Q4Roll
- Q5Arms
- DateTaken

Swaddle

- SwaddleID
- DisplayName
- Size
- Material
- Pattern
- Color
- Roll
- Arms
- BuyLink
- ButtonText

Stretch



Data Implementation

User

Purpose

User will be the most important table for the database for Swaddle Swap. This table will serve as the authenticator for logins (stretch) and will set the UserID which will persist as a key in other tables.

Implementation

User information will be collected on the quiz screen for the MVP and the sign up screen for the stretch. For the MVP, they will enter their first and last name and their email address on the quiz screen. (stretch) The sign up screen will either display when they click the sign up button or in the flow of the rental user flow. First they will fill out the quiz, then choose their swaddles and then they will sign up or login if they aren't already logged in.

Interaction

This data will follow the user throughout the site. Ideally, they would be able to update the information in the user settings screen but this is not in scope for the MVP.

```
{
        “UserID”: “String”,
        “Email Address”: “String”
        “FirstName”: “String”,
        “LastName”: “String”,
        “Password”: “String”,
}
```

Swaddle

Purpose

The swaddle table is going to be used in a few different instances. First, it will house information that will be displayed on the page to the end user such as the Name, Size and Price. The rest of the data will be used to power the recommendations for the recommendations screen for the user. For the stretch goal, it will power the rental cart and the state will be updated.

Implementation

Most of the data in this table will be powered by the website owner. This will be based on the different attributes of the swaddles. The only data fields that will be updated by the site is the Status field (stretch) which will be updated if the swaddle is out for rent or purchased. The Condition field (stretch) will also be updated if the swaddle is rented and returned. If it is rented once, it will be updated to “Used” but the default is “New”.

Interaction

The swaddles are displayed in the Recommendations screen based on the answers from the quiz. The information comes from the quiz page and then the recommended swaddles are displayed on the Checkout page with affiliate links from the database out to specific sites to purchase. The swaddles that display are based on how closely the attributes match the answers on the quiz. For the stretch goals, the rental cart will be displayed and the user will choose rent, buy or pass and that will update the “status”.

```
{
        “SwaddleID”: “Integer”,
        “UserID”: “Integer”,
        “Size”: “String”,
        “Material”: “String”,
        “Color”: “String”,
        “Pattern”: “String”,
        “Roll”: “Boolean”,
        “Arms”: “String”,

        “SwaddleName”: “String”,

        “BuyLink”:”String”,

        “ButtonText”,

        “Price”: “String”,

        “Status”: “String”,

        “Condition”: “Boolean”

}
```

Quiz Answer

Purpose

The Swaddle Quiz answer come from the 5 quiz questions we are going to ask prior to recommending swaddles to try.

Implementation

This quiz will be filled out by the user when they click on the “Take Quiz” button or click “Get Started” from the How it Works page. The data will be collected from the answers.

Interaction

The answers to the quiz will be used to power the recommendations screen based on the attributes of the 5 swaddles with the most similar attributes to the quiz answers. These answers will be stored as well and if swaddles are returned, this should work as an AI engine (post MVP) and get smarter with user feedback.

```
{

        “QuizAnswerID”: “Integer”

        “UserID”: “Integer”,

        “Q1Size”: “String”,

        “Q2Climate”: “String”,

        “Q3Gender”: “String”,

        “Q4Roll”: “String”,

        “Q5Arms”: “String”,

        “DateTaken”: “String”

}
```

Recommendation

Purpose

The recommendation will be used during the recommendations as well as the checkout screen (stretch). This table will hold all the Swaddle IDs of the 5 recommended swaddles.

Implementation

This table will render based on the quiz answers in the quiz table and will get the swaddles that most closely align to those answers in the quiz table.

Interaction

It will also have the User ID as the primary key so that it can connect to the other tables. (Stretch) The table also has the rental date which will automatically set the return date to 14 days after the rental date. Lastly, it will have the total prices of the cart which it will calculate from the price of each swaddle using the Swaddle IDs.

```
{
        “RecommendationID”: “String”,
        “UserID”: “String”,
        “Swaddle1”: “String”,
        “Swaddle2”: “String”,
        “Swaddle3”: “String”,
        “Swaddle4”: “String”,
        “Swaddle5”: “String”,
        “RentalDate”: “String”,
        “ReturnDate”: “String”,
        “TotalPrice”: “String”


}
```

(Stretch) Rent Cart

Purpose

The rent cart will be used during the recommendations as well as the checkout screen. This table will hold all the Swaddle IDs of the 5 recommended swaddles.

Implementation

This table will render based on the quiz answers in the quiz table and will get the swaddles that most closely align to those answers in the quiz table.

Interaction

The table also has the rental date which will automatically set the return date to 14 days after the rental date. It will also have the User ID as the primary key so that it can connect to the other tables. Lastly, it will have the total prices of the cart which it will calculate from the price of each swaddle using the Swaddle IDs.

```
{

        “RentCartID”: “Integer”,

        “UserID”: “Integer”,

        “Swaddle1”: “String”,

        “Swaddle2”: “String”,

        “Swaddle3”: “String”,

        “Swaddle4”: “String”,

        “Swaddle5”: “String”,

        “RentalDate”: “String”,

        “ReturnDate”: “String”,

        “TotalPrice”: “String”


}
```

(Stretch) Payment Information

Purpose

The payment information is important as it will be how we collect a deposit for the rental on the MVP. If I have time and build out the buy experience, it will also be stored so that we can charge the card for the items that they decide to buy

Implementation

This data will be collected in the check out process within the checkout screens. It will be stored in the database.

Interaction

If the user does not return the swaddles within 14 days, their card will be charged the full amount. If they decide to buy some swaddles and return some, they will only be charged for those swaddles that they buy.

```
{
        “PaymentInfoID”: ”Integer”,
        “UserID”: “Integer”,
        “ShippingAddress”: “String”,
        “ShippingCity”: “String”,
        “ShippingState”: “String”,
        “ShippingZip”: “String”,
        “CreditCardNum”: “String”,
        “CCExpDate”: “String”,
        “SecurityCode”: “String”,
        “BillingAddress”: “String”,
        “BillingCity”: “String”,
        “BillingState”: “String”,
        “BillingZip”: “String”
}
```