Acadia University

The Cat’s Meow Database

# Deliverable #4: Project Demo

COMP 3753 X1 Final Project

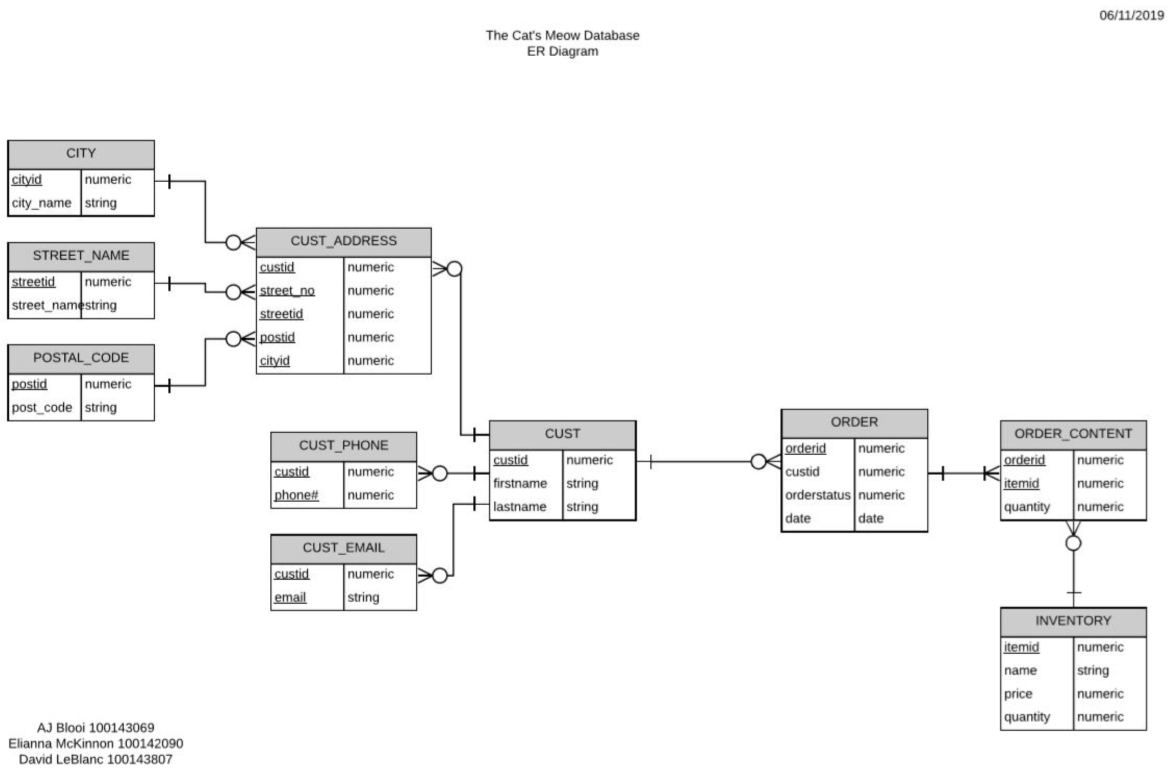
AJ Blooi | 100143069 | 143069b@acadiau.ca

David LeBlanc | 100143807 | 143807l@acadiau.ca

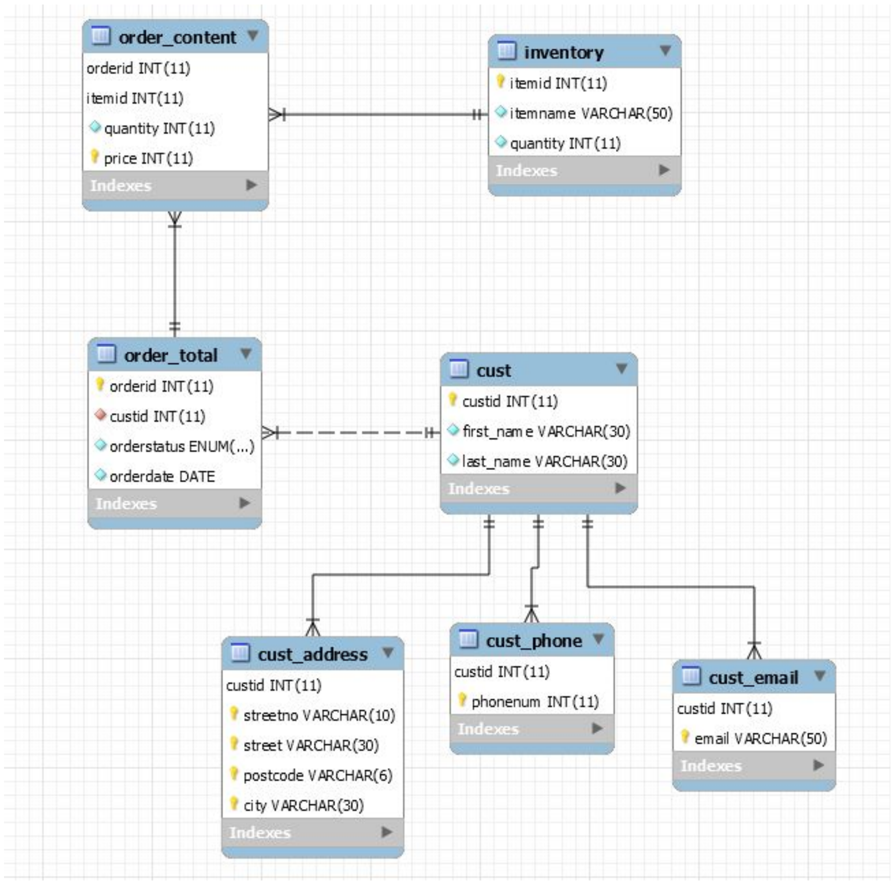
Elianna McKinnon | 100142090 | 142090m@acadiau.ca

December 5th, 2019

## Schema Design



For the Cat's Meow Database we started off with an ER diagram to demonstrate the relationships between these entities in the database: customer, order (items ordered), and inventory (total items). The final schema design went through some minor changes following review of project deliverable #1, which is visible from reverse engineering the current state of the database:

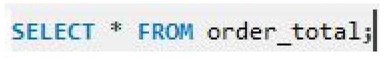


For one, city, street name (now street), and postal code (postcode) were migrated to cust_address. This was done in an effort to reduce the difficulty of maintenance of the database for the client, as having them separate would potentially bring the need to keep a current record of all current streets/cities/postal codes in Canada. This way, the problem of determining a valid location can be offloaded to the client side and be handled by some external organisation, like Canada Post.

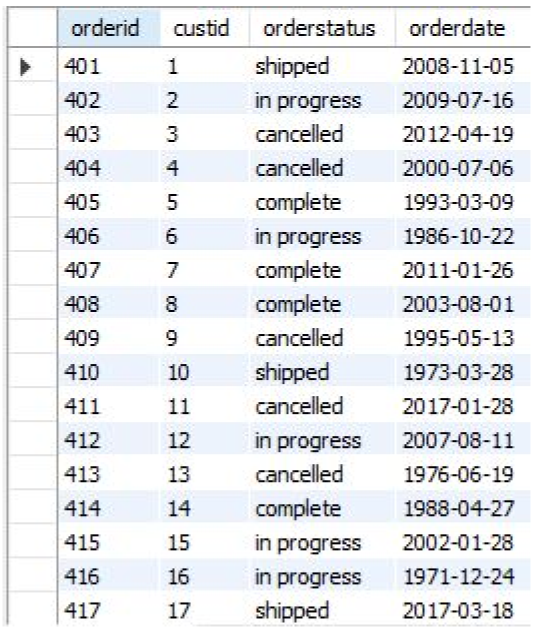
Price was moved from inventory to order content, in order to support the tracking of altered prices at the time of sale (eg. due to a special offer or re-pricing). This enables more accurate tracking of the history of sales / orders for the client.

As for other choices made in general about the schema, the overall goal was to only store minimal amounts of data to keep the database operation simple and to avoid storing non mission critical data. For example, customer phone numbers and email addresses are stored in a separate table to avoid NULL values in cust for the case that the customer does not want / need to give out both pieces of information, as well as to stop the redundant storage of phone numbers. Order statuses are stored as an enumerated type so that queries about the statuses are easily read and understood by staff while still regulating valid inputs without the need for an additional table (see image below).

With this underlying structure which is mostly normalised, it should be relatively straightforward to make changes or additions to the database schema. For instance, if the client wishes to store further optional information on customers, tables could be created for things like store preferences, names of pets, etc.



```
SELECT * FROM order_total;
```



| orderid | custid | orderstatus | orderdate |
|---|---|---|---|
| 401 | 1 | shipped | 2008-11-05 |
| 402 | 2 | in progress | 2009-07-16 |
| 403 | 3 | cancelled | 2012-04-19 |
| 404 | 4 | cancelled | 2000-07-06 |
| 405 | 5 | complete | 1993-03-09 |
| 406 | 6 | in progress | 1986-10-22 |
| 407 | 7 | complete | 2011-01-26 |
| 408 | 8 | complete | 2003-08-01 |
| 409 | 9 | cancelled | 1995-05-13 |
| 410 | 10 | shipped | 1973-03-28 |
| 411 | 11 | cancelled | 2017-01-28 |
| 412 | 12 | in progress | 2007-08-11 |
| 413 | 13 | cancelled | 1976-06-19 |
| 414 | 14 | complete | 1988-04-27 |
| 415 | 15 | in progress | 2002-01-28 |
| 416 | 16 | in progress | 1971-12-24 |
| 417 | 17 | shipped | 2017-03-18 |

## API

For the construction of our API we put together two PHP files that both query data from the database. One file is for querying information about customers (customers.php), and the other file for grabbing details about orders in terms of their status and order dates (orders.php).

With how our APIs function they both connect to the database through its server hosted by AMPPS, then builds the query by integrating a SQL SELECT statement inside the PHP code. Once retrieved from the database, the data is then sent along to the client from the API as a JSON array.

In orders.php, between the connection and disconnections from our database, our SQL SELECT statement queries all data from the *order_total* column according to the conditions given for *orderstatus* and *orderdate*. It is as shown below:

```
$sql = "
SELECT * FROM order_total"
. " WHERE orderstatus = '" . $status . "'"
. " AND orderdate BETWEEN '" . $startDate . "' AND '" . $endDate .
"'";
```

For customers.php our SELECT statement is split into different if statements according to how the client searches for a customer's name. If the client wants to view customers by only first or last name, then the query will locate all customers that have a matching first or last name. If the client attempts to search for a customer with both a matching first and last name, the database will find all customers that have both those names. Our conditional query is structured in this way below:

```
// If client types in only first name
if (($firstName != '') && ($lastName == '')) {
      $sql = "
      SELECT * FROM cust"
      . " WHERE first_name = '" . $firstName . "'";
}

// If client types in only last name
if (($firstName == '') && ($lastName != '')) {
      $sql = "
      SELECT * FROM cust"
      . " WHERE last_name = '" . $lastName . "'";
}

// If client types in first name & last name
if (($firstName != '') && ($lastName != '')) {
      $sql = "
      SELECT * FROM cust"
      . " WHERE first_name = '" . $firstName . "'"
      . " AND last_name = '" . $lastName . "'";
}
```

## **Demo**

Our database can be queried with links similar to the ones below using AMPPS along the structure of*:*

*localhost/folder_containing_PHP_file/file.php?myCondition=this*

customers.php

http://localhost/db%20proj/customers.php?firstName=eunice&lastName=kub

http://localhost/db%20proj/customers.php?firstName=eunice

http://localhost/db%20proj/customers.php?lastName=kub

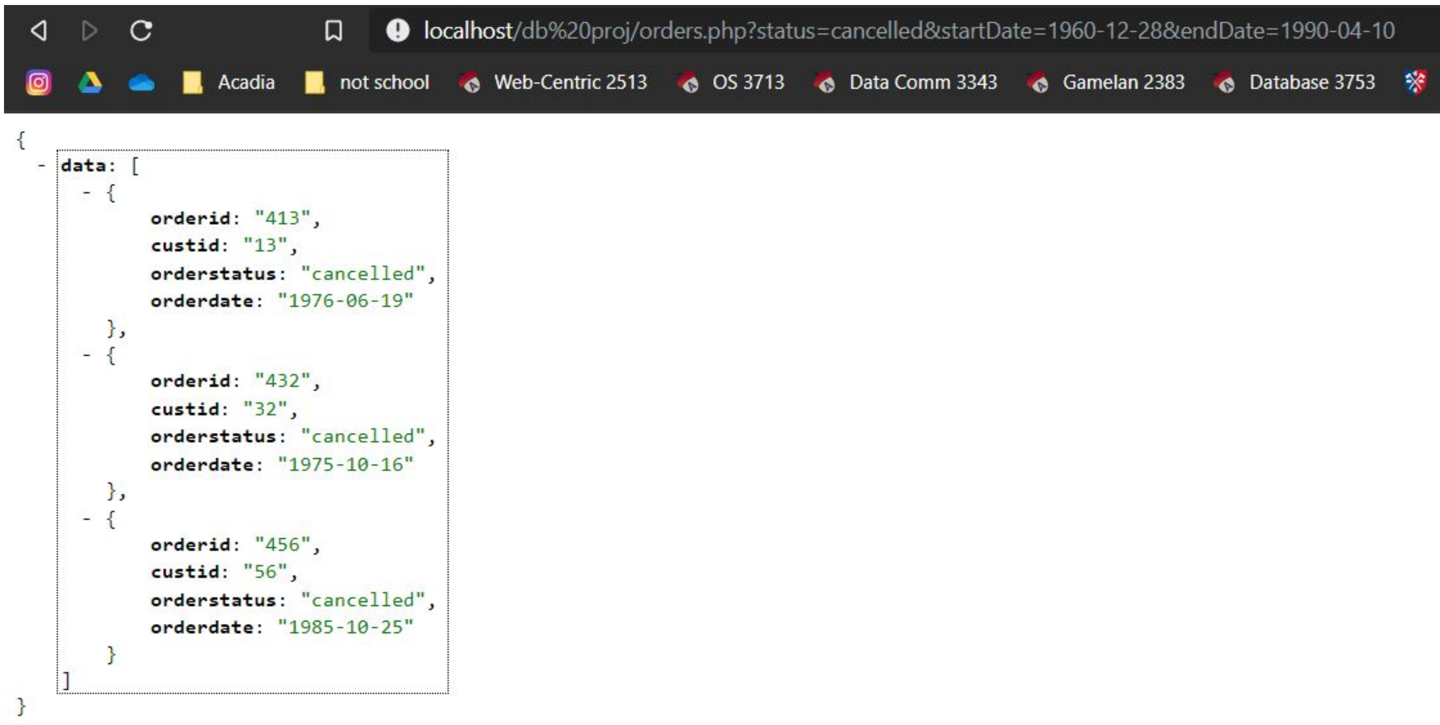
orders.php

http://localhost/db%20proj/orders.php?status=complete&startDate=2000-01-01&endDate=2019-11-27

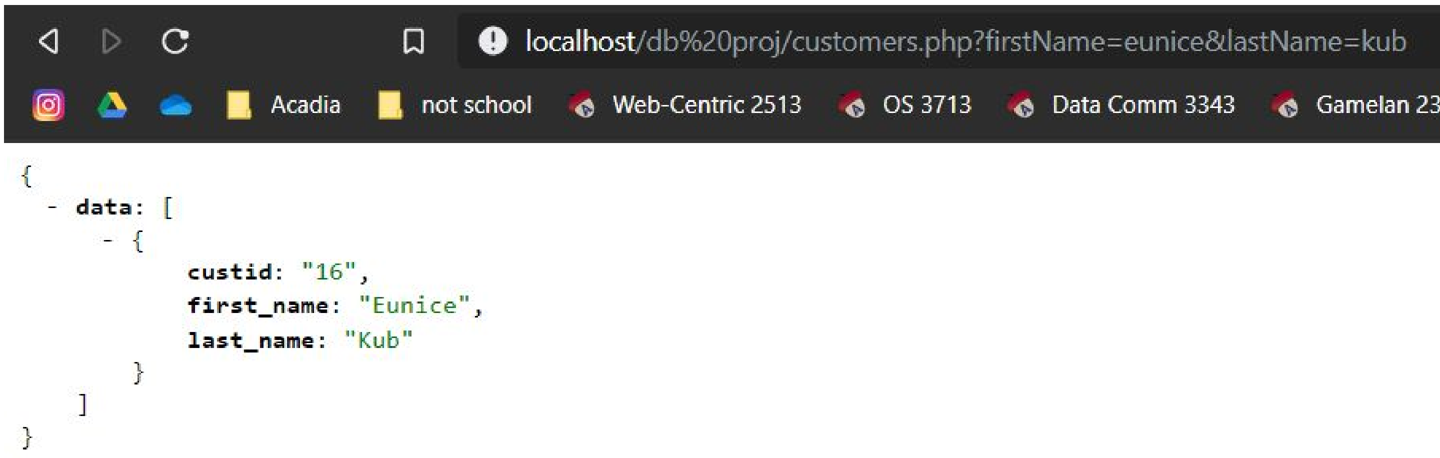
http://localhost/db%20proj/orders.php?status=shipped&startDate=1800-01-01&endDate=9000-01-01

http://localhost/db%20proj/orders.php?status=cancelled&startDate=1960-12-28&endDate=1990-04-10

Examples of the links functioning with JSON data being displayed:


localhost/db%20proj/orders.php?status=cancelled&startDate=1960-12-28&endDate=1990-04-10
Acadia
not school
Web-Centric 2513
OS 3713
Data Comm 3343
Gamelan 2383
Database 3753

```
{
  - data: [
      - {
            orderid: "413",
            custid: "13",
            orderstatus: "cancelled",
            orderdate: "1976-06-19"
        },
      - {
            orderid: "432",
            custid: "32",
            orderstatus: "cancelled",
            orderdate: "1975-10-16"
        },
      - {
            orderid: "456",
            custid: "56",
            orderstatus: "cancelled",
            orderdate: "1985-10-25"
        }
    ]
}
```


localhost/db%20proj/customers.php?firstName=eunice&lastName=kub
Acadia
not school
Web-Centric 2513
OS 3713
Data Comm 3343
Gamelan 23

```
{
  - data: [
      - {
            custid: "16",
            first_name: "Eunice",
            last_name: "Kub"
        }
    ]
}
```