Order Up Database

Jack MacCallum: jmaccallum16@winona.edu

Matthew Hickey: mhickey17@winona.edu

Roles

Jack: Milestone I report, Milestone II report, ERD, finished database

Matthew: Initial database design, relation schemas, Final report

Summary

The goal of this project was to create a solution for a restaurant to manage orders that were placed by customers. It includes features such as managing tables, taking orders, having those orders assigned to the server responsible and keeping track of the cost for the final payment. We had previously consulted with someone who had been working in the restaurant industry a few years ago and worked with him to develop a solution that could fix the failing of the previous system that he had used. The system we developed only focuses on the orders placed in the restaurant from start to finish. In the future the system could be expanded to not only take care of the orders but be an all in one solution for the whole restaurant. This could include things like managing the employees schedule and payroll information. We decided when making this project our goal was to take one small but complete area of a restaurant's operation and do it well. After that is accomplished it

leaves us free to pursue adding more feature to reach the goal of being a total restaurant solution.

Data

The data we used from this project came from a Buffalo Wild Wings menu and random generation. Using real world data like this is important because it can help us catch something we might have left out and shows that our system could be used in a restaurant that is already well established.

Relational Schema

Orderables(ID, name, type, price)

Contains(Orders_ID, Orderables_ID)

Orders(ID, Tables_ID)

Tables(ID, TableArea_ID)

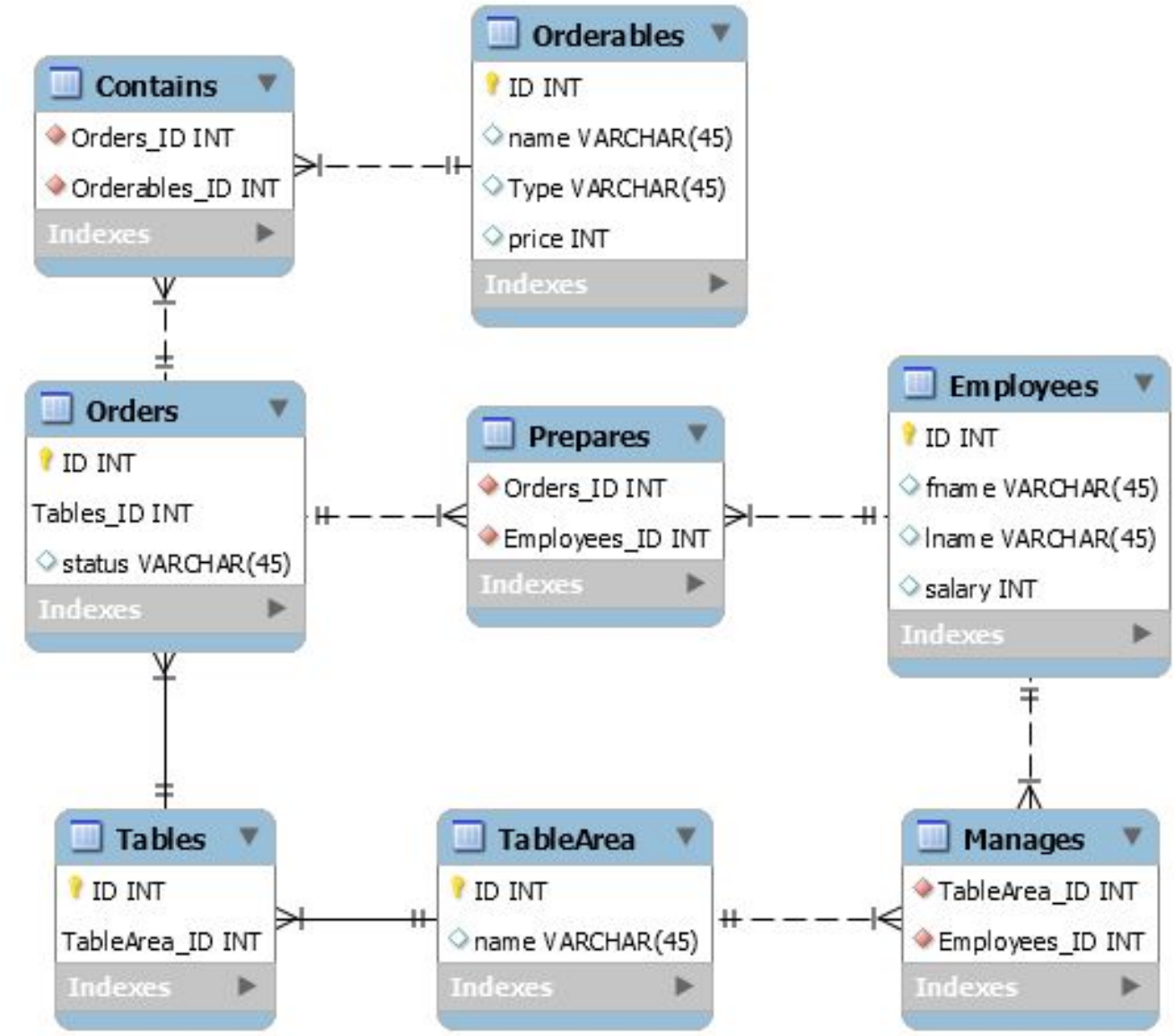
TableArea(ID, name)

Manages(TableArea_ID, Employees_ID)

Employees(ID, fname, lname, salary)

Prepares(Orders_ID, Employees_ID)

Entity Relationship Diagram



Use Cases

- Placing and Adding Items to an Order
    - When a group of customers initially comes in, an order needs to be created for their table.
    - Adding individual food or drink items to the order.

```
lock table orders WRITE;
insert into orders
values (51, 1, "Placed");

lock table contains WRITE;
insert into contains
values  (51, 3),
        (51, 21),
        (51, 39);
unlock tables;
```

- Getting the total cost of the order
    - When it's time to pay the total cost of the order needs to be retrieved.

```
select contains.orders_ID as "Order #", sum(price) as "Total Price"
from contains join orderables
where contains.orders_ID = "1" and contains.orderables_ID = orderables.ID;
```

- Who Manages the Table Areas
    - Control which Employees are assigned to each Table Area.

```
select tablearea.name as `Table Area`, fname as `First Name`, lname as `Last Name`
from employees natural join manages join tablearea
where employees.ID = manages.Employees_ID and tablearea.ID=manages.TableArea_ID;
```

- Manage the Order statuses for each Table
    - See the status for each order.
    - Advance the status of an order.

```
select tables.ID as `Table ID`, orders.ID as `Order ID`, orders.status as `Order Status`
from tables join orders
where tables.ID = orders.Tables_ID;
```

- Which Orders and Items a Table has
    - Itemized list of items and prices for receipt.

```
select orders.Tables_ID as "Table", contains.Orders_ID as "Order #", orderables.name as "Item Name", price as "Item Price"
from contains join orders join orderables
where orders.Tables_ID = "1" and orders.ID = contains.Orders_ID and orderables.ID = contains.Orderables_ID;
```