# Running head: Sanders’ Sandwiches Database

# Sanders’ Sandwiches Database

# CIST 1307 Database and Design Management

# Drayson Sanders

# 30 April 2021

# **Table of Contents**

[Table of Contents 2](#_Toc69836583)

[Introduction 3](#_Toc69836584)

[Timeline 4](#_Toc69836590)

[Business Processes 5](#_Toc69836591)

[Logical and Conceptual Model 6](#_Toc69836592)

[Physical Data Model 7](#_Toc69836593)

[Analysis 10](#_Toc69836594)

[Appendix 11](#_Toc69836595)

[References 13](#_Toc69836596)

# 

# Introduction

# The Sanders’ Sandwiches restaurant is having a hard time keeping track of the tips and services being provided between the delivery drivers and the waitresses. The tips are not being divided fairly based on the number of tables the waitresses have tended to and the number of deliveries each delivery driver has made.

# In order to make sorting the tips easier and fairer, this database includes the customer entity to keep track of the table number and the address of the customer for delivery purposes. The database also aims to also keep track of the tips given to the delivery drivers by the customers (see “Business Rules” Appendix). The Customers entity holds the unique identifier of tableNo for waitresses to be able to uniquely identify the customer. The entity also holds the customers address broken into Address, City, State, and Zip for use by the delivery driver. The table relations state that any one customer can have one or many orders.

# The Order entity holds the unique identifier OrderNo to identify a customer’s order. This entity also has price, details, tip put on the order, wether or not the order is for dine in or delivery, and the foreign keys of empID and tableNo. This allows for the Customers order to be broken down more specifically as to what the order actually is and what type of employee will be handling the order.

# Both the waitress and delivery drivers are encompassed in the employee entity and have a unique identifier attribute named the empID (see the “Conceptual model” Appendix). This allows the database to present information on a specific employee. The data dictionary (see “Data Dictionary” Appendix) also displays more in-depth information about the entities and their attributes.

# The Employee entity holds the unique identifier of empID and is also referenced as a foreign key in the Order entity. The Employee entity also holds waitress and delivery attributes to differentiate between which employee handles which orders. The relations for this table state that any one employee can have zero or more orders that they handle.

# Timeline

**January 31st**: DELIVERABLE: Project Idea

**February 7th**: Business Rules and ER Model

**February 7th**: DELIVERABLE: Project Proposal

**February 21st:** DELIVERABLE: Business Rules

**March 14th:** Conceptual Model

**April 11th:** DELIVERABLE: Logical Data Model

**April 18th:** DELIVERABLE: Physical Database Model

**April 30th:** Project Due Date

# Business Processes

* An **Employee** is someone who works for the restaurant.
* A **Waitress** is an **Employee** who works at a restaurant and takes **orders**.
* An **Order** is a request from the customer.
* A **Customer** is someone who eats at a restaurant and makes an **order**.
* A **Delivery** **Driver** is an **Employee** who delivers food from the restaurant to the customer based on an order.
* A **Customer** has an **order**
* A **Customer** also has a table number.
* A **Customer** may also have an address.
* A **Customer** must place an **Order** with the **Waitress** or **Delivery Driver**.
* A **Customer** may leave a tip on their **order**.
* An **Employee** must bring the **order** to the **customer**.
* A **Waitress** may have zero or more **Order(s).**
* A **Delivery** driver may have zero or more **Order(s).**
* An **Order** has Price
* An **Order** has Order Details
* An **Order** Has a Delivery option.
* An **Order** has a Dine in option.

# Diagram, schematic Description automatically generated**Diagram Description automatically generated**Logical and Conceptual Model

# Physical Data Model

use master;

drop database Sandwiches;

go

create database Sandwiches;

go

use Sandwiches;

Create table Employee(

empID int identity primary key not null,

waitress nvarchar(50) not null,

DeliveryDriver nvarchar(50) not null,

Constraint check\_employee\_id

check (empID between 1 and 10000)

)

Create table Customer(

TableNo int Identity Primary Key not null,

Addr nvarchar(150) null,

City nvarchar(150) null,

State nvarchar(150) null,

Zip nvarchar(150) null

)

create table Orders(

OrderNo int identity Primary Key,

Price nvarchar(150),

Details nvarchar(150),

Tip nvarchar(150),

Delivery nvarchar(150),

DineIn nvarchar(150),

empID int references Employee(empID),

TableNo int references Customer(TableNo)

)

insert into Employee(waitress,DeliveryDriver) values ('Yes','No')

insert into Employee(waitress,DeliveryDriver) values ('Yes','No')

insert into Employee(waitress,DeliveryDriver) values ('No','Yes')

insert into Customer(addr,City,State,Zip) values ('2620 Brick Chimney Rd','Bradford','PA','16701')

insert into Customer(addr,City,state,zip) values ('9545 Rose Lake Ct','Bradford','PA','16701')

insert into Customer(addr,City,State,Zip) values ('3345 blue bird Rd','Bradford','PA','16701')

insert into Orders (Price,Details,Tip,Delivery,DineIn) values ('$2','Grilled Cheese','$1','No','Yes')

insert into Orders (Price,Details,Tip,Delivery,DineIn) values ('$3','Hamburger','$4','No','Yes')

insert into Orders (Price,Details,Tip,Delivery,DineIn) values ('$3','CheeseBurger','$5','Yes','No')

insert into Orders (Price,Details,Tip,Delivery,DineIn) values ('$5','Bacon CheeseBurger','$3','No','Yes')

insert into Orders (Price,Details,Tip,Delivery,DineIn) values ('$4','Grilled Chicken','$3','Yes','No')

go

create procedure AddEmployee

@waitress nvarchar(5),

@Delivery nvarchar(5)

As

Begin

insert into Employee(waitress,DeliveryDriver)

values ('Yes','No')

End

go

exec AddEmployee 1,2;

go

create procedure getDetails

As

Begin

select O.Price,O.Details,O.Tip,O.Delivery,O.DineIn,C.TableNo

from Orders O

inner join Customer C on O.Details=str(C.TableNo)

where C.TableNo='2'

End

go

exec getDetails;

go

Create procedure detailsWithParameters

(@OrderNo int)

As

Begin

Select O.Details

from Orders O

where OrderNo=@OrderNo

End

go

exec detailsWithParameters 3;

go

create view[OrderPrice] AS

Select Details

from Orders

where Price ='$3';

go

Select\*from OrderPrice;

go

create view[Tip] AS

Select Tip,Details

from Orders

where Delivery='Yes';

go

Select\*from Tip;

go

create view[getAddress] AS

Select Addr,City,State,Zip

from Customer

where TableNo='3'

go

Select\*from getAddress;

# Analysis

In my original proposal I had three entities. These entities included the delivery driver, waitress, and customer. Delivery driver held attributes like tips, number of deliveries, and empID. The Waitress entity was very similar and held the tips, number of tables, and empID attributes. The Customer entity I made a weak entity at first, but later learned that because the customer entity does not rely on any other entity it cannot be identified as weak. The Customer entity also held the TableNumber (Spelled out), food, and drinks attributes. In the original relationships I created I had the customer tipping both a delivery driver and a waitress. I did not use correct cardinality to display the correct syntax of what I was trying to convey. I also had both the waitress and delivery driver taking an order from the same customer at the same time. My business rules also reflected the whacky relationships I had.

In order to fix the entities and their relationships I absorbed the waitress and delivery driver entity into one. I named this entity Employee, which held the empID, waitress, and Delivery Driver attributes. This allowed for a more streamlined and easier process to determine the type of employee taking the order. I also create the entity Order(s). This also helped immensely with nailing down attributes to entities. I moved the tips attribute from the waitress and delivery driver entities into the orders entity. I also created an OrderNo as a primary key, the price, details, delivery, and dine in attributes. I also added empID and tableNo as foreign keys to the Order(s) entity. Adding a delivery and dine in attribute allowed to solve the confusion between whether or not the tips and order from the customer entity were going to be handled by a delivery driver or waitress. Overall, the structure of my models, and subsequently my business rules, changed quite a bit the more I learned and the feedback I got from my peers. The changes I made streamlined the process and ended up making my physical database more efficient.

# Appendix

**SQL Database:**

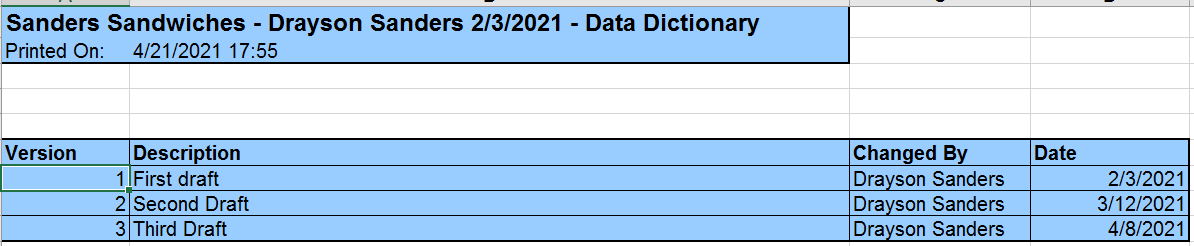
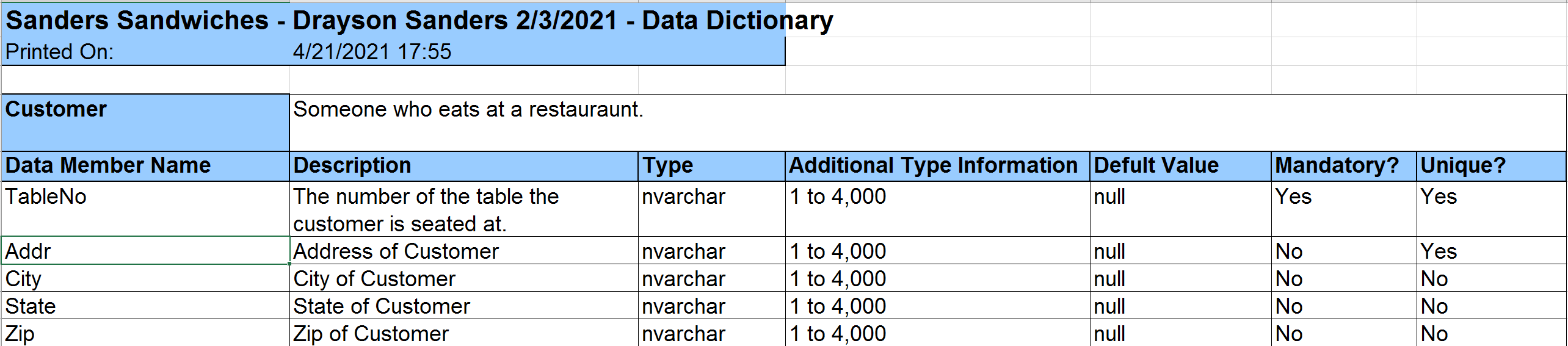


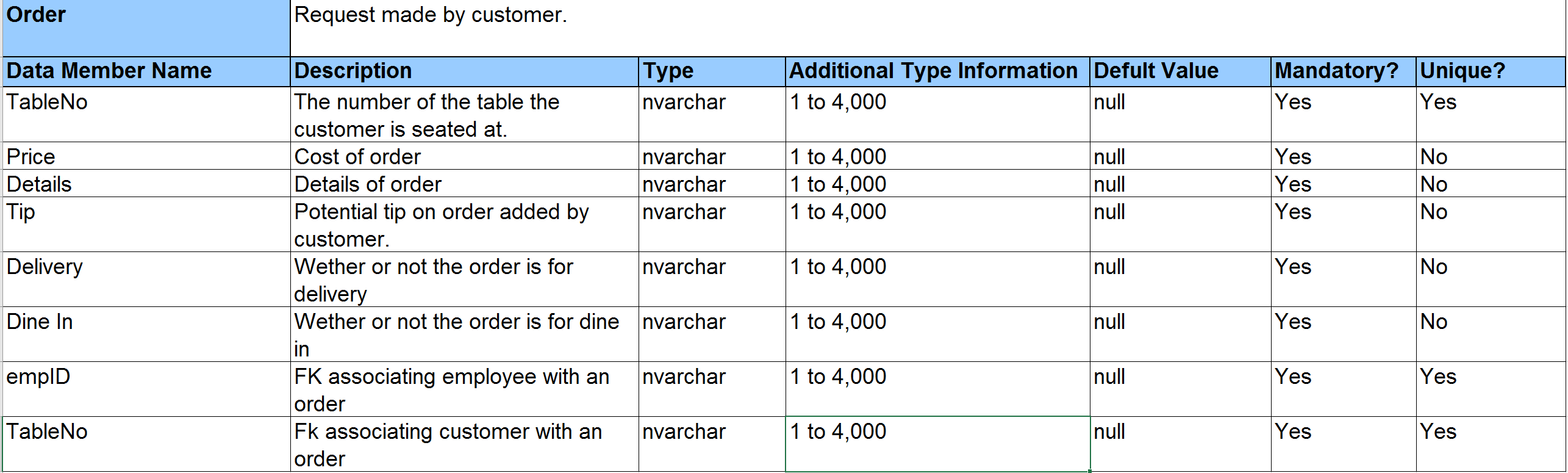
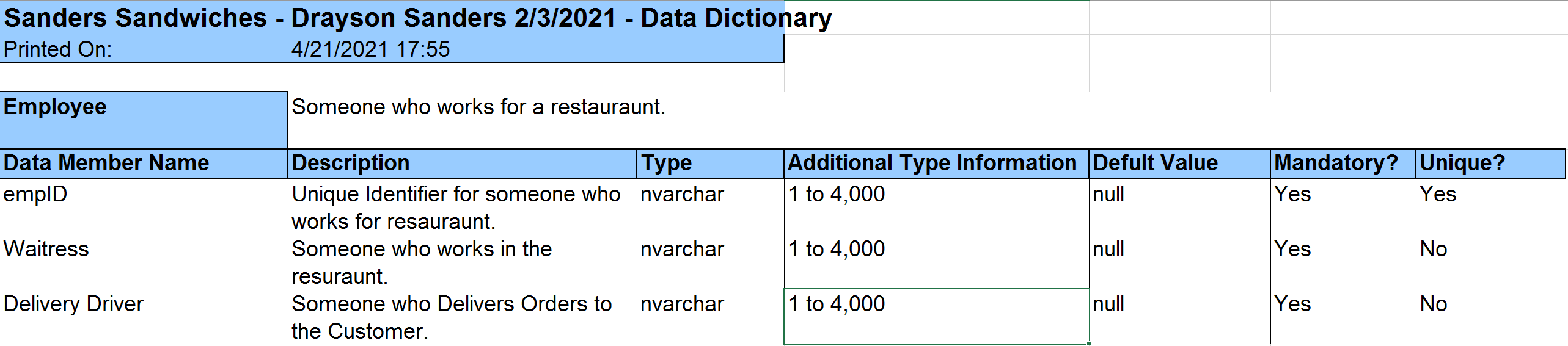
**DIA Conceptual Model:**



**DIA Logical Model:**



**Data Dictionary:**



# References

Ricardo, Catherine M., and Susan D. Urban. *Databases Illuminated*. Jones\*Bartlett Learning LLC, 2015.