Pizza Database

Reece Parks and Matt Fleschner

CS 333: Database Systems

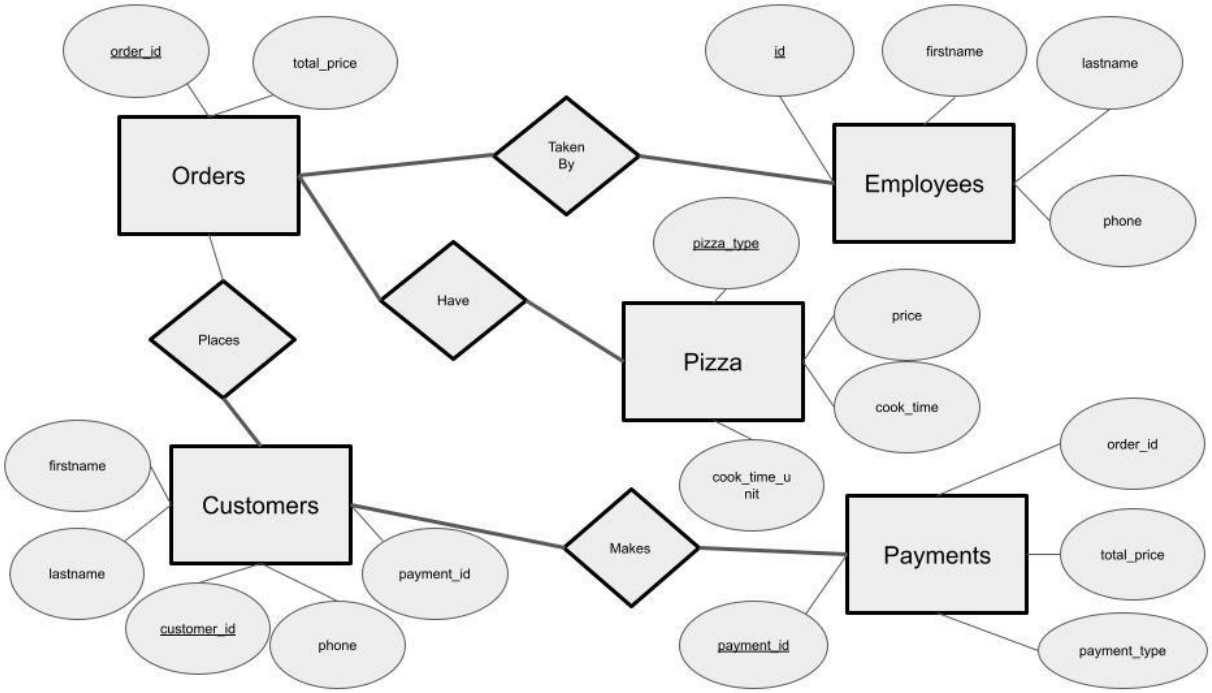
Dr. Sorenson

9 December 2022

Table of Contents/Overview

Entity-Relationship Diagram

Pictured below is the ER diagram used for the pizza restaurant's database:



This relational database includes five entity sets: Orders, Employees, Pizza, Payments, and Customers. Three of these entity sets are directly related to the Orders entity. Orders are taken by employees, orders have pizza, and customers place orders. The only entity set that is not directly related to Orders is the Payments entity; however, it still has a relationship through Customers as customers make payments. In addition, each entity consists of many attributes, and the primary keys, the underlined attributes shown above, include order_id for Orders, id for Employees, pizza_type for Pizza, payment_id for Payments, and customer_id for Customers.

Schema and Data

Schema

Below is the schema for the five tables created in the relational database for a pizza restaurant. The schema can also be found in “PizzaSchema.txt”.

```
create table employees(
    id int not null,
    firstname varchar(30) not null,
    lastname varchar(30) not null,
    phone char(12) not null,
    primary key(id) );

create table customers(
    customer_id int not null,
    firstname varchar(30) not null,
    lastname varchar(30) not null,
    payment_id int(5),
    phone char(12) not null,
    unique(payment_id),
    primary key(customer_id) );

create table pizza(
    pizza_type enum('cheese','pepperoni','sausage','meatlovers','hawaiian','supreme','margherita','bbqchicken','buffalo','mushroom'),
    pizza_price double,
    cook_time int,
    cook_time_unit varchar(3) not null,
    primary key(pizza_type) );

create table payments(
    order_id int,
    payment_id int,
    total_price double,
    payment_type char(1),
    primary key(payment_id),
    foreign key(payment_id) references customers(payment_id) );

create table orders(
    order_id int,
    customer_id int,
    total_price double,
    employee_id int,
    pizza_type enum('cheese','pepperoni','sausage','meatlovers','hawaiian','supreme','margherita','bbqchicken','buffalo','mushroom'),
    primary key(order_id),
    foreign key(customer_id) references customers(customer_id),
    foreign key(employee_id) references employees(id),
    foreign key(pizza_type) references pizza(pizza_type) );


load data local infile 'employees.txt' into table employees;
load data local infile 'customers.txt' into table customers;
load data local infile 'pizza.txt' into table pizza;
load data local infile 'payments.txt' into table payments;
load data local infile 'orders.txt' into table orders;
```

Data

Below are some sample data used to fill each table from the schema:

The full Employees table data can also be found in “employees.txt”.

```
12234   Tami    Harrison    410-304-3201
41675   Katrina Bennett 317-230-0239
69561   Lucian  Payton  317-093-3043
66956   Julia   Mann    317-765-9320
27909   Janet   Daniell 765-320-0439
76485   Brande  Silver  312-900-4399
37740   Micheal Jeffries    123-456-7890
85095   Ella    Causer  301-203-1055
21550   Carissa Lynwood 443-020-3933
39918   Jillie  Hobson  317-203-3211
11182   Cole    Bische  317-531-9427
83336   Elwood  Ellsworth   317-883-5832
85387   Ann Rowntree    312-029-3240
43174   Cory    Simons  410-262-3191
83546   Bret    Ridley  765-322-2300
85918   Micah   Westcott    314-222-2299
15459   Johnny  Payne   314-249-3005
44481   Jon Sorenson    317-239-9213
77959   Ankur   Gupta   317-110-2943
28567   Miles   Mann    765-524-7932
59647   Ryan    Rybarczyk   314-555-6840
64195   Anthony Signorelli  513-238-5569
```

The full Customers table data can also be found in “customers.txt”.

```
4381395 Toby    Mills   15792   188-701-0175
4094819 Shyann  Boone   78764   350-007-5650
6997178 Kadyn   Robertson   59083   790-545-5211
1930944 Aliyah  Dunn    96530   241-665-5854
9583086 Giselle Donaldson   72730   784-986-4990
5569598 Weston  Hunt    82947   942-171-6308
6371068 Skyler  Howard  53951   532-485-4658
3257273 Matthias    Mcfarland   94837   212-078-972(
7693764 Mekhi   Alvarez 45409   967-971-7197
2578508 Abdullah    Bowen   62312   941-514-3919
8811413 Arjun   Mccall  22461   916-737-7331
6726086 Marquise    Bates   77087   818-557-0989
2299182 Yasmine Navarro 47211   850-543-6640
7249812 Cassie  Acevedo 27992   627-071-6814
6782284 Jacob   Underwood   13865   404-693-2680
3578280 Gillian Strickland  45880   340-144-3776
2421032 Dominique   Mata    69200   695-768-7475
5349229 Ayla    Allen   49427   615-088-3842
7402250 Simeon  Frost   62583   526-867-6179
8292398 Desirae Williamson  89722   777-879-4848
6524373 Nora    Cross   37209   225-035-3945
5864648 Valentino   Bolton  3550    269-807-6816
8091182 Eden    Sutton  90584   757-744-9871
8836828 Raiden  Barrera 58380   997-954-6056
2107718 Lilly   Hanna   89723   669-114-6252
697947  Tatiana Stevens 84411   903-066-1088
5724185 Penelope    Boone   56257   992-106-4176
3525181 Gabriel Hayden  96934   468-007-2974
8718302 Gia Hardin  83888   973-001-7743
9059256 Esmeralda   Vargas  18225   777-346-8190
1030413 Scarlet Kent    2201    945-899-1073
1315309 Logan   Andrade 35199   853-059-7286
1538882 Aliana  Cordova 96403   894-719-7227
4710655 Lesly   Bradford    13942   420-721-5446
386870  Callum  George  33321   359-753-2951
1396785 Carley  Best    10037   170-432-6273
6666325 Alonzo  Newton  44378   210-381-2816
7153648 Abigail Maddox  23742   304-800-1149
677668  Savanah Ryan    43284   576-184-8835
4626571 Mayra   Fischer 10224   909-362-4117
706820  Casey   Lane    65395   676-292-1398
596146  Enrique Mccoy   17765   312-746-1392
4620076 Lindsey Espinoza    74640   304-988-9214
7925817 Leticia Beard   40903   683-779-8954
2123159 Zain    Hunt    23009   507-471-0559
```

The full Pizza table data can also be found in “pizza.txt”.

```
cheese  13.99   6   min
pepperoni   13.99   7   min
sausage 13.99   7   min
meatlovers  17.99   8   min
hawaiian    17.99   7   min
supreme 18.99   9   min
margherita  15.49   6   min
bbqchicken  16.99   8   min
buffalo 17.99   8   min
mushroom    13.99   7   min
```

The full Payments table data can also be found in “payments.txt”.

```
5066029 15792   14.7    c
958308  78764   19.37   c
6515641 59083   17.43   c
9298627 96530   14.71   c
4565387 72730   19.05   c
3753496 82947   14.45   c
4528492 53951   18.2    c
4683596 94837   19.91   c
6964782 45409   17.75   c
2928106 62312   15.34   c
1956857 22461   16.18   c
273143  77087   14.78   c
2628999 47211   21.46   c
1089057 27992   15.83   c
2564849 13865   21.42   c
2917385 45880   17.7    c
9732534 69200   14.17   c
6503968 49427   19.23   c
2332130 62583   16.81   c
4011255 89722   19.59   c
8760430 37209   16.13   c
8674972 3550    16.21   c
740423  90584   20.74   c
1631353 58380   17.52   c
3661487 89723   21.63   c
208992  84411   19.46   c
1485813 56257   18.27   c
767087  96934   17.03   c
5650133 83888   17.87   c
7327457 18225   18.3    c
7771302 2201    21.36   c
8508910 35199   19.19   c
8972794 96403   16.09   c
5257773 13942   21.56   c
8756115 33321   19.69   c
1799663 10037   14.94   c
2594937 44378   18.7    c
2550998 23742   21.19   c
6481259 43284   20.01   c
6272290 10224   19.86   c
2374277 65395   21.61   c
2400076 17765   18.07   c
2272111 74640   15.03   c
5914964 40903   16.8    c
5470974 23009   20.18   c
```

The full Orders table data can also be found in “orders.txt”.

```
5066029 4381395 14.7    85095   pepperoni
958308  4094819 19.37   76485   bbqchicken
6515641 6997178 17.43   28567   cheese
9298627 1930944 14.71   41675   supreme
4565387 9583086 19.05   83336   meatlovers
3753496 5569598 14.45   83546   meatlovers
4528492 6371068 18.2    59647   meatlovers
4683596 3257273 19.91   76485   hawaiian
6964782 7693764 17.75   83336   margherita
2928106 2578508 15.34   28567   margherita
1956857 8811413 16.18   11182   buffalo
273143  6726086 14.78   15459   cheese
2628999 2299182 21.46   85095   hawaiian
1089057 7249812 15.83   15459   supreme
2564849 6782284 21.42   39918   supreme
2917385 3578280 17.7    83546   pepperoni
9732534 2421032 14.17   12234   meatlovers
6503968 5349229 19.23   43174   cheese
2332130 7402250 16.81   85095   margherita
4011255 8292398 19.59   83336   cheese
8760430 6524373 16.13   85387   hawaiian
8674972 5864648 16.21   76485   supreme
740423  8091182 20.74   83336   bbqchicken
1631353 8836828 17.52   83546   hawaiian
3661487 2107718 21.63   66956   cheese
208992  697947  19.46   21550   supreme
1485813 5724185 18.27   21550   supreme
767087  3525181 17.03   44481   cheese
5650133 8718302 17.87   43174   bbqchicken
7327457 9059256 18.3    76485   buffalo
7771302 1030413 21.36   85387   hawaiian
8508910 1315309 19.19   39918   meatlovers
8972794 1538882 16.09   39918   buffalo
5257773 4710655 21.56   66956   meatlovers
8756115 386870  19.69   69561   meatlovers
1799663 1396785 14.94   44481   supreme
2594937 6666325 18.7    85918   bbqchicken
2550998 7153648 21.19   44481   meatlovers
6481259 677668  20.01   41675   pepperoni
6272290 4626571 19.86   69561   pepperoni
2374277 706820  21.61   39918   sausage
2400076 596146  18.07   37740   meatlovers
2272111 4620076 15.03   77959   margherita
5914964 7925817 16.8    43174   meatlovers
5470974 2123159 20.18   69561   supreme
```

As seen above, this pizza database accounts for all the important pieces of information and relational data that are needed for a pizza restaurant. The database schema allows for relationships and connections between the unique order IDs, customer IDs, payment IDs, employee IDs, and so much more. With this structure, accessing information and data about orders, pizza, or customers can be completed very easily and efficiently with a low amount of effort. Information is divided into separate tables that maintain accuracy properly and eliminate redundancy whenever possible.

Functional Dependencies

Taking note of functional dependencies is always important for good database design. A functional dependency is a relationship between two attributes in a table with one usually, but not always, being a primary key and the other a non-key attribute. The right-hand side attribute is functionally dependent on the left-hand side in a functional dependency. Below is a list of all the function dependencies for each of the tables in the pizza database:

Orders

order_id -> total_price, employee_id, customer_id, pizza_type

Customers

customer_id -> firstname, lastname, payment_id, phone

payment _id -> firstname, lastname, customer_id, phone

phone -> firstname, lastname, customer_id, payment_id

Payments

order_id -> payment_id, total_price, payment_type

payment_id -> payment_type, total_price, order_id

Pizza

pizza_type -> cook_time, cooktime_unit, pizza_price

Employees

id -> firstname, lastname, phone

phone -> id, firstname, lastname

BCNF Verification

For a database to be in Boyce-Codd Normal Form all attributes must be dependent on the primary key with no partial dependencies. In addition, the left-hand side of the dependency must also be a superkey of the database, or an attribute that uniquely determines all other attributes of the database. Lastly, all non-trivial attributes must be removed. Below is verification that the pizza database is in BCNF going table by table. For simplicity, letters such as A,B,C, or D will represent the attributes of each table.

Orders

order_id(A) -> total_price(B), employee_id(C), customer_id(D), pizza_type(E)

Since order_id gets every other attribute of the table(A -> BCDE), it is the superkey and the chosen candidate/primary key of the Orders table. As this is the only functional dependency, BCNF holds true.

Customers

customer_id(A) -> firstname(B), lastname(C), payment_id(D), phone(E)

payment _id(D) -> customer_id(A), firstname(B), lastname(C), phone(E)

phone(E) -> customer_id(A), firstname(B), lastname(C), payment_id(D)

For the Customers table, the primary key and chosen super key is customer_id with the functional dependency A -> BCDE. As every attribute can already be accounted for and determined by A, the table is in BCNF. Non-primary dependencies such as D -> ABCE or E -> ABCE stay true to the BCNF principles as well.

Payments

payment_id (A) -> order_id(B), total_price(C), payment_type(D)

order_id(B) -> payment_id(A), total_price(C), payment_type(D)

Similarly to the Customers table, the Payments table's candidate and primary key is the unique ID of the table known as payment_id. Therefore, every other attribute is determined by payment_id(A -> BCD) and the table is in BCNF. The other functional dependency present, B -> ACD, also is a superkey that does not violate the BCNF rules.

Pizza

pizza_type(A) -> cook_time(B), cooktime_unit(C), pizza_price(D)

Since pizza_type determines every other attribute of the table(A -> BCD), it is the candidate and chosen super key of the Pizza table which verifies itself into BCNF.

Employees

id(A) -> firstname(B), lastname(C), phone(D)

phone(D) -> id(A), firstname(B), lastname(C)

For the Employees table, id is the chosen primary key and superkey of the table as it determines everything else(A -> BCD). The table is therefore in BCNF as that along with the other functional dependency on phone(D -> ABC) allows the rules of BCNF to hold true.

Sample Queries

Listed below are some questions along with some sample SQL queries to show how effective the pizza restaurant is at finding needed information for the business. Being able to efficiently create queries allows for a strong database that can be thoroughly applied to a real life business.

1.List all the names of customers who ordered sausage pizza.

select firstname,lastname from customers natural join orders where customers.customer_id = orders.customer_id and orders.pizza_type = 'sausage';

```
+-----------+----------+
| firstname | lastname |
+-----------+----------+
| Lewis     | Woodward |
| Rishi     | Cooley   |
| Rafael    | Jimenez  |
| Madeline  | Morrow   |
| Noah      | Duke     |
| Anahi     | Long     |
| Ally      | Skinner  |
| Casey     | Lane     |
| Esteban   | Novak    |
| Ruben     | Lamb     |
| Haylee    | Price    |
| Callie    | Suarez   |
| Alexis    | Sparks   |
| Maci      | Jennings |
| Skye      | Delacruz |
| Mariana   | Bartlett |
| Ezekiel   | Bray     |
| Luciana   | Moreno   |
| Cannon    | Cochran  |
| Brock     | Dillon   |
+-----------+----------+
```

2. List the amount of debit transactions.

select count(payment_type) as debit_payments from payments where payment_type = 'd';

```
+----------------+
| debit_payments |
+----------------+
|             30 |
+----------------+
```

3.How many of each pizza were ordered?.

select pizza_type, count(*) as NumberOfOrders from orders group by pizza_type;

pizza_type	NumberOfOrders
cheese	20
pepperoni	19
sausage	20
meatlovers	25
hawaiian	24
supreme	23
margherita	24
bbqchicken	23
buffalo	22

4.List the first name of customers who share the same first name with someone else.

select firstname from customers group by firstname having count(firstname)>1;

firstname
Carley
Dominique
Gillian
Jaidyn
Lilly
Nora
Yasmine

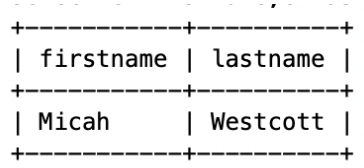
5. Give the maximum number of one type of pizza ordered.

select max(mycount) from (select pizza_type,count(pizza_type) mycount from orders group by pizza_type) as T;

max(mycount)
25

6.Which employee took the most orders from customers?

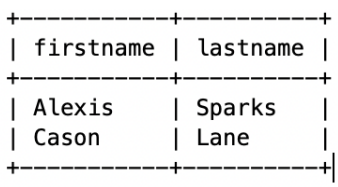
select e.firstname,e.lastname from employees e where e.id = (select max(employee_id) from orders);



```
+-----------+----------+
| firstname | lastname |
+-----------+----------+
| Micah     | Westcott |
+-----------+----------+
```

7.Which customer spent the most amount of money on orders(there will be two with equal amounts)?

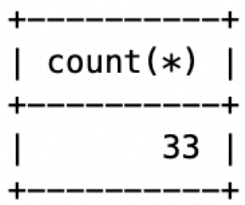
select c.firstname,c.lastname from customers c inner join orders o on o.customer_id = c.customer_id where o.total_price = (select max(o.total_price) from orders o);



```
+-----------+----------+
| firstname | lastname |
+-----------+----------+
| Alexis    | Sparks   |
| Cason     | Lane     |
+-----------+----------+
```

8.How many customers tipped more than $4 on their order?

select count(*) from orders join pizza on orders.pizza_type = pizza.pizza_type where total_price - pizza_price > 4;



```
+----------+
| count(*) |
+----------+
|       33 |
+----------+
```

9.List the names and phone numbers for all customers who ordered bbq chicken pizza.

select firstname,lastname,phone from customers natural join orders where pizza_type = 'bbqchicken';

firstname	lastname	phone
Lacey	Trevino	268-238-9047
Kira	Bridges	241-813-5579
Eden	Sutton	757-744-9871
Shyann	Boone	350-007-5650
Samara	Santos	599-872-6018
Alonzo	Newton	210-381-2816
Jaidyn	Marquez	953-644-4257
Carson	Dickson	826-900-6808
Phillip	Case	371-002-2855
Finn	Aguirre	340-916-8888
Jordan	Holden	158-175-3093
Ralph	Benson	639-842-8564
Max	Mcmahon	496-032-9277
Julia	Zamora	810-564-1265
Armani	Logan	878-729-4236
Gia	Hardin	973-001-7743
Kamora	Reeves	152-932-1424
Melissa	Waters	933-504-1494
Audrina	Daugherty	884-595-1728
Jessica	Medina	222-195-2162
Mathew	Vazquez	982-584-4702
Bronson	Lozano	280-349-0088
Kamden	Holt	739-264-4616

10.Organize all employees by last name.

select firstname,lastname from employees order by lastname;

firstname	lastname
Katrina	Bennett
Cole	Bische
Ella	Causer
Janet	Daniell
Elwood	Ellsworth
Ankur	Gupta
Tami	Harrison
Jillie	Hobson
Micheal	Jeffries
Carissa	Lynwood
Julia	Mann
Miles	Mann
Johnny	Payne
Lucian	Payton
Bret	Ridley
Ann	Rowntree
Ryan	Rybarczyk
Anthony	Signorelli
Brande	Silver
Cory	Simons
Jon	Sorenson
Micah	Westcott

Conclusion

In conclusion, the pizza database succeeds in being an effective database for a real life pizza restaurant. With a few minor adjustments, it could fit and be applied to any modern day pizza delivery system that can be found. Not only does this database have a clear focus on accessing data about orders taken, but it also provides clear connections and relationships to customer, employee, and payment information as seen by the entity-relationship diagram. With 5 entity sets: Orders, Customers, Employees, Payments, and Pizza, along with simple but effective relationships between them, any necessary information can be located through simple SQL queries. Customer phone numbers, payment methods for pizza, and employee tips received along with orders taken are just a few of the many details that can be accessed efficiently with little to no effort. This database succeeds at dividing information into many important tables and minimizing redundancy while also staying true to Boyce-Codd Normal Form. In addition, it ensures accuracy of information while accommodating the specific needs of the business at hand. The Pizza Database is a very refined design in which its application into any pizza restaurant would drastically improve quality of life overall.