

Pizza Ordering System Database Design Report

Anasuya Sikdar

Introduction

In the fast-paced realm of food service, the ability to efficiently manage orders, maintain customer relationships, and streamline operations is crucial for success. Pizza dot Com, an emerging leader in the culinary industry, seeks to harness the power of information technology to achieve these goals. This report presents a comprehensive database design tailored for Pizza dot Com, aimed at facilitating their ordering system and providing a robust platform for data management.

The database design detailed herein has been meticulously crafted to capture the essence of Pizza dot Com's operational requirements. It encompasses a set of relational database structures that organize and store data pertaining to customers, pizza offerings, orders, and the detailed constituents of those orders. By adhering to principles of relational database design and normalization, the proposed schema not only accommodates the current needs of the business but is also scalable to meet future demands.

Through the implementation of this database, Pizza dot Com will be equipped with a powerful tool to monitor transactional data, analyze customer preferences, and optimize their service delivery. The subsequent sections of this report will delve into the specifics of the database schema, elucidating the SQL code utilized for table creation and data manipulation, and providing a rationale for each design choice. Visual aids in the form of Entity-Relationship Diagrams (ERD) using Chen's notation, and Crow's Foot physical schema diagrams will accompany the text to offer a clear and concise depiction of the database structure.

Formal Writing

ENTITIES:

CUSTOMER: A CUSTOMER is a person who has ordered pizzas from our company (or who may do so in the future). For each CUSTOMER, we record the Name and Address. A Name is the full name of a customer. For each CUSTOMER, there will be exactly one Name. An address is the full street address of a customer. For each CUSTOMER, there will be exactly one Address. For each CUSTOMER, the Name will uniquely identify the CUSTOMER.

ORDER: An ORDER is a transaction involving the purchase of one or more of each of one or more different types of pizzas from our company. For each ORDER, we record the Number. A Number is the identifying number assigned to an order. For each ORDER, there will be exactly one Number. For each ORDER, the Number will uniquely identify the ORDER.

LINE_ITEM: A LINE_ITEM is the part of a transaction that involves the purchase of one or more of one particular type of pizza from our company. For each LINE_ITEM, we record the Quantity. A Quantity is the number of pizzas of one particular kind ordered in one line item of an order. For each LINE_ITEM, there is exactly one Quantity. No attribute will be sufficient to uniquely identify the LINE_ITEM without referencing related entities.

PIZZA: A PIZZA is particular type of pizza that is made by our company. For each PIZZA, we record the Kind and Price. A Kind is the size and topping of a pizza. For each PIZZA, there is exactly one Kind. A Price is the cost of a single pizza of that kind. For each PIZZA, there is exactly one Price. For each PIZZA, the Kind will uniquely identify the PIZZA.

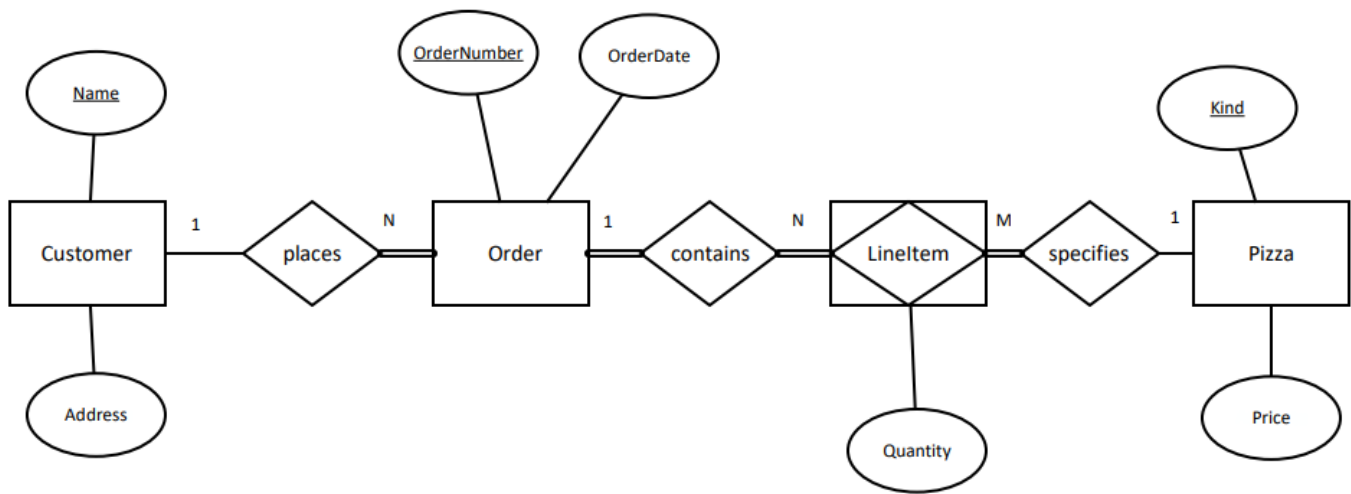
RELATIONSHIPS:

places: A CUSTOMER may place many ORDERS (but may not place any). Each ORDER must be placed by exactly one CUSTOMER.

contains: An ORDER must contain one or more LINEITEMs. Each LINEITEM must be contained in exactly one ORDER.

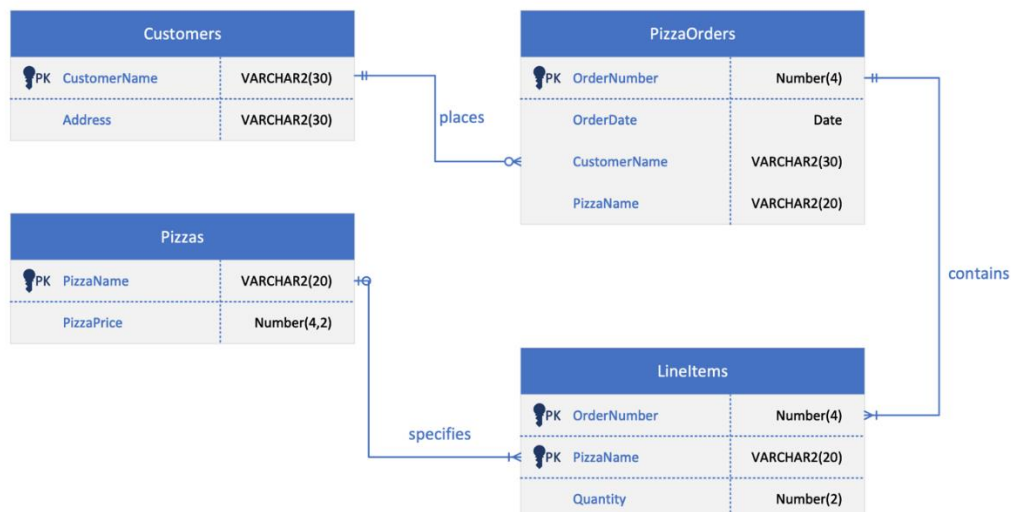
specifies: A LINEITEM must specify exactly one PIZZA. Each PIZZA may be specified by many LINEITEMS (but may not be specified by any).

Entity-Relationship Diagram (ERD)



The ERD provides a visual representation of the entities (**CUSTOMERS**, **PIZZAS**, **PIZZAORDERS**, **LINEITEMS**) and their relationships. In Chen's notation, entities are represented by rectangles, and relationships by diamonds. The **LINEITEMS** associative entity is depicted as a diamond connecting **PIZZAORDERS** and **PIZZAS**, with **Quantity** being an attribute of this relationship.

Physical Schema Diagram (Crow's Foot Notation)



The physical schema diagram illustrates how the tables are structured in the database, including primary keys, foreign keys, and other attributes. Crow's Foot notation is employed to detail the cardinality and nature of the relationships between tables, with crow's feet indicating one-to-many relationships and lines indicating one-to-one relationships.

SQL Developer Oracle Code for the Pizza Ordering

```
Worksheet  Query Builder

/*
Pizza dot Com Case
Anasuya Sikdar
*/

-- Drop tables if they exist
DROP TABLE LINEITEMS;
DROP TABLE PIZZAORDERS;
DROP TABLE CUSTOMERS;
DROP TABLE PIZZAS;

-- Create CUSTOMERS table
CREATE TABLE CUSTOMERS
(
  CustomerName VARCHAR2(30),
  Address VARCHAR2(30),
  CONSTRAINT CUSTOMERS_PK PRIMARY KEY (CustomerName)
);

-- Insert customers
INSERT INTO CUSTOMERS (CustomerName, Address) VALUES ('John Smith', '123 Lakeview, Chicago');
INSERT INTO CUSTOMERS (CustomerName, Address) VALUES ('Emily Clark', '433 Westend, Chicago');
INSERT INTO CUSTOMERS (CustomerName, Address) VALUES ('Michael Brown', '867 Beuna, Chicago');
INSERT INTO CUSTOMERS (CustomerName, Address) VALUES ('Alex Johnson', '888 ChinaTown, Chicago');

-- Fetch customers table
SELECT * FROM CUSTOMERS;

-- Create PIZZAS table
CREATE TABLE PIZZAS
(
  PizzaName VARCHAR2(20),
  PizzaPrice number(4,2),
  CONSTRAINT PIZZAS_PK PRIMARY KEY (PizzaName)
);

-- Insert pizzas
INSERT INTO PIZZAS (PizzaName, PizzaPrice) VALUES ('Large Cheese', 14.99);
INSERT INTO PIZZAS (PizzaName, PizzaPrice) VALUES ('Small Cheese', 11.99);
INSERT INTO PIZZAS (PizzaName, PizzaPrice) VALUES ('Large Pepperroni', 16.99);
INSERT INTO PIZZAS (PizzaName, PizzaPrice) VALUES ('Small Pepperroni', 13.99);

-- Fetch pizzas table
SELECT * FROM PIZZAS;

-- Create PIZZAORDERS table without Quantity
CREATE TABLE PIZZAORDERS
(
  OrderNumber NUMBER(4),
  CustomerName VARCHAR2(20),
  PizzaName VARCHAR2(20),
  OrderDate DATE,
  CONSTRAINT PIZZAORDERS_PK PRIMARY KEY (OrderNumber),
  CONSTRAINT PIZZAORDERS_FK1 FOREIGN KEY (CustomerName) REFERENCES CUSTOMERS (CustomerName),
  CONSTRAINT PIZZAORDERS_FK2 FOREIGN KEY (PizzaName) REFERENCES PIZZAS (PizzaName)
);

-- Insert orders without Quantity
INSERT INTO PIZZAORDERS (OrderNumber, CustomerName, PizzaName, OrderDate)
VALUES (1001, 'John Smith', 'Large Cheese', TO_DATE('21-FEB-2024', 'DD-MON-YYYY'));
INSERT INTO PIZZAORDERS (OrderNumber, CustomerName, PizzaName, OrderDate)
VALUES (1002, 'Emily Clark', 'Small Pepperroni', TO_DATE('22-FEB-2024', 'DD-MON-YYYY'));
INSERT INTO PIZZAORDERS (OrderNumber, CustomerName, PizzaName, OrderDate)
VALUES (1003, 'Michael Brown', 'Large Pepperroni', TO_DATE('23-FEB-2024', 'DD-MON-YYYY'));
INSERT INTO PIZZAORDERS (OrderNumber, CustomerName, PizzaName, OrderDate)
VALUES (1004, 'Alex Johnson', 'Small Cheese', TO_DATE('24-FEB-2024', 'DD-MON-YYYY'));

-- Fetch pizzaorders table
SELECT * FROM PIZZAORDERS;

-- Create LINEITEMS table with Quantity
CREATE TABLE LINEITEMS
(
  OrderNumber NUMBER(4),
  PizzaName VARCHAR2(20),
  Quantity NUMBER(2),
  CONSTRAINT LINEITEMS_PK PRIMARY KEY (OrderNumber, PizzaName),
  CONSTRAINT LINEITEMS_FK1 FOREIGN KEY (OrderNumber) REFERENCES PIZZAORDERS (OrderNumber),
  CONSTRAINT LINEITEMS_FK2 FOREIGN KEY (PizzaName) REFERENCES PIZZAS (PizzaName)
);

-- Insert line items with Quantity
INSERT INTO LINEITEMS (OrderNumber, PizzaName, Quantity)
VALUES (1001, 'Large Cheese', 2);
INSERT INTO LINEITEMS (OrderNumber, PizzaName, Quantity)
VALUES (1002, 'Small Pepperroni', 1);
INSERT INTO LINEITEMS (OrderNumber, PizzaName, Quantity)
VALUES (1003, 'Large Pepperroni', 3);
INSERT INTO LINEITEMS (OrderNumber, PizzaName, Quantity)
VALUES (1004, 'Small Cheese', 2);

-- Fetch lineitems table
SELECT * FROM LINEITEMS;
```

Outputs:

Script Output x

Task completed in 0.305 seconds

Table LINEITEMS dropped.

Table PIZZAORDERS dropped.

Table CUSTOMERS dropped.

Table PIZZAS dropped.

Table CUSTOMERS created.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

Script Output x Query Result x

SQL | All Rows Fetched: 4 in 0.266 seconds

	CUSTOMERNAME	ADDRESS
1	John Smith	123 Lakeview, Chicago
2	Emily Clark	433 Westend, Chicago
3	Michael Brown	867 Beuna, Chicago
4	Alex Johnson	888 ChinaTown, Chicago

Table PIZZAS created.

1 row inserted.

1 row inserted.





1 row inserted.

1 row inserted.

Script Output x		Query Result x	
		SQL All Rows Fetched: 4 in 0.095 seconds	
	PIZZANAME	PIZZAPRICE	
1	Large Cheese	14.99	
2	Small Cheese	11.99	
3	Large Pepperroni	16.99	
4	Small Pepperroni	13.99	

Script Output x

Query Result x



SQL | All Rows Fetched: 4 in 0.09 seconds

	ORDERNUMBER	CUSTOMERNAME	PIZZANAME	ORDERDATE
1	1001	John Smith	Large Cheese	21-FEB-24
2	1002	Emily Clark	Small Pepperroni	22-FEB-24
3	1003	Michael Brown	Large Pepperroni	23-FEB-24
4	1004	Alex Johnson	Small Cheese	24-FEB-24

Script Output x		Query Result x	
		SQL All Rows Fetched: 4 in 0.065 seconds	
	ORDERNUMBER	PIZZANAME	QUANTITY
1	1001	Large Cheese	2
2	1002	Small Pepperroni	1
3	1003	Large Pepperroni	3
4	1004	Small Cheese	2

Table PIZZAORDERS created.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

Table LINEITEMS created.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.