Programming Foundations Databases

RELATIONAL DATABASES

- A relational database is made up of relations or tables, which are the set of columns.
- For example in a restaurant we have customers who come to eat. Therefore we record who is a customer in our first database. The customer is the entity. All instances of the entity share the same attribute. They all have a first name, last name, email, & phone.
- Then take for example the type of dishes that the restaurant offers. The types of dishes are the main entity of the second table. All instances of dishes share the same common attributes of name, price, and description.

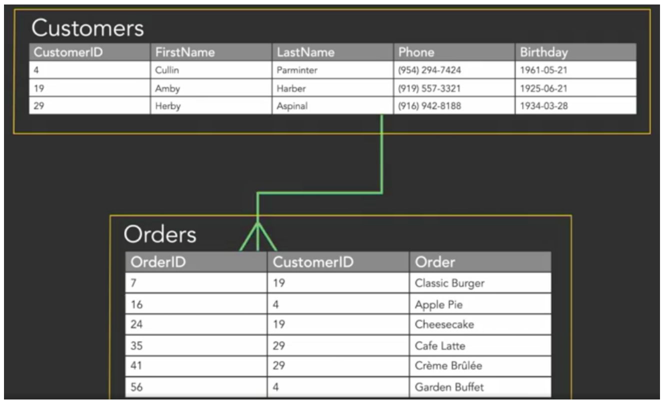
KEYS & UNIQUE VALUES

- Primary Key is the most important key in a table. A table does not require a PK, but helps to acquire records easily.
- Oftentimes an ID column is added to serve as a primary key. When this is done it is called a synthetic or surrogate key.
- Composite key is when two or more fields are taken together to act as a unique identifier.
- Foreign key is when a primary key from one table is referenced in another table.

RELATIONSHIPS

- We can query a piece of information from individual tables, but by forming relationships lets use data in more complex and realistic applications.

- One to many is by far the most common relationship. An example of this would be a primary key customer id of 4 is referenced as a foreign key multiple times in another table. It would be like if a certain customer has placed multiple orders.



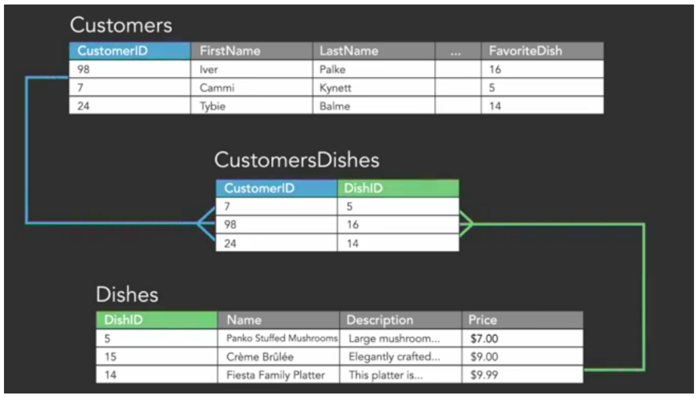
Customers

CustomerID	FirstName	LastName	Phone	Birthday
4	Cullin	Parminter	(954) 294-7424	1961-05-21
19	Amby	Harber	(919) 557-3321	1925-06-21
29	Herby	Aspinal	(916) 942-8188	1934-03-28

Orders

OrderID	CustomerID	Order
7	19	Classic Burger
16	4	Apple Pie
24	19	Cheesecake
35	29	Cafe Latte
41	29	Crème Brûlée
56	4	Garden Buffet

- Many to many relationships are joined using a linking table/associative. They are formed through foreign keys of the tables that we are bringing together. This can be used to associate many customers with many dishes. This can allow you to keep track of each dish every customer has ever ordered.



Customers

CustomerID	FirstName	LastName	...	FavoriteDish
98	Iver	Palke		16
7	Cammi	Kynett		5
24	Tybie	Balme		14

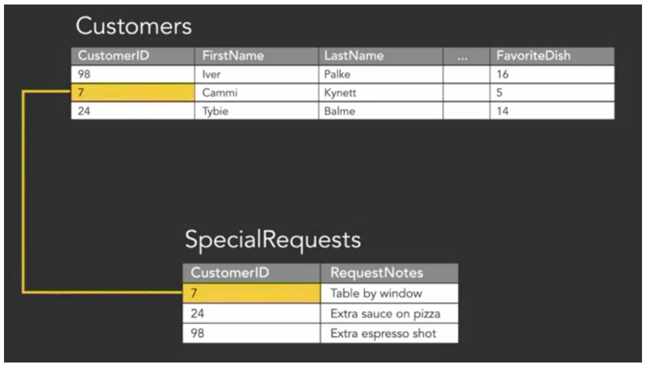
CustomersDishes

CustomerID	DishID
7	5
98	16
24	14

Dishes

DishID	Name	Description	Price
5	Panko Stuffed Mushrooms	Large mushroom...	$7.00
15	Crème Brûlée	Elegantly crafted...	$9.00
14	Fiesta Family Platter	This platter is...	$9.99

- A one to one relationship only associates one record on one table to one record on another. Such that it can not be associated with anything else.



Customers

CustomerID	FirstName	LastName	...	FavoriteDish
98	Iver	Palke		16
7	Cammi	Kynett		5
24	Tybie	Balme		14

SpecialRequests

CustomerID	RequestNotes
7	Table by window
24	Extra sauce on pizza
98	Extra espresso shot

<u>ACID AND TRANSACTIONS</u>

- A set of operations which must all be completed. If any of these are not completed there will be no changes to the database. Transactions followed an acronym known as ACID. Atomic, Consistent, Isolated, Durable.
- Atomic means that the transaction is indivisible. Pieces of it can not be separated out.
- Consistency means whatever the transaction does it needs to leave the database in a valid and consistent state. The actions in the transaction cannot violate integrity rules that are defined for the database.
- Isolation means that while the activities in the transaction are being completed nothing else can make changes to the data involved. If we were in the middle of moving money from one account to another and the user submitted another transfer request, the user would have to wait until the first is done completely.
- Durability means that the info we change in the transaction actually gets written in the database. When the transaction gets written as complete the data is there.
- A database does not have to deal with financial info, for example we can use a transaction to book a reservation at a restaurant, or if we wanted to check the stock level of inventory, where we prevent another worker from altering it.

BASIC SQL

- SQL allows statements to be written for DBMS to interpret how to interact with data.
- In this role of interacting with the database, SQL is called a data manipulation language.
- SQL can be classified as a DDL -data definition language or DCL - data control language, when offering features to manage the database itself such as creating or modifying tables.

TABLES

MODELLING & PLANNING A DB

- What does your database need to store? Restaurant Example : Customers, Dishes, Events, Orders, Reservations, Favorite Dishes.
- Customer Table - FirstName, LastName, Email, Birthday, Phone, Address, City, State.
- Dishes Table - Name, Description, Price.
- Event Table - Name, Date, Description
- Reservations - Customer ID, Date, Partysize
- Orders - CustomerID, OrderDate

NAMING TABLES

- Plurals, Indented First Letter, Avoid Spaces. Short & concise.

COLUMNS & DATA TYPES

- Strings - alphanumeric characters and text
- CHAR - fixed number of characters, reserved for a field where we know the amount of characters for each recorded piece. CHAR(20) will be 20 spaces no matter what, even if the instance is only 5 characters. It still occupies that space.
- VARCHAR - variable number of characters up to a maximum length

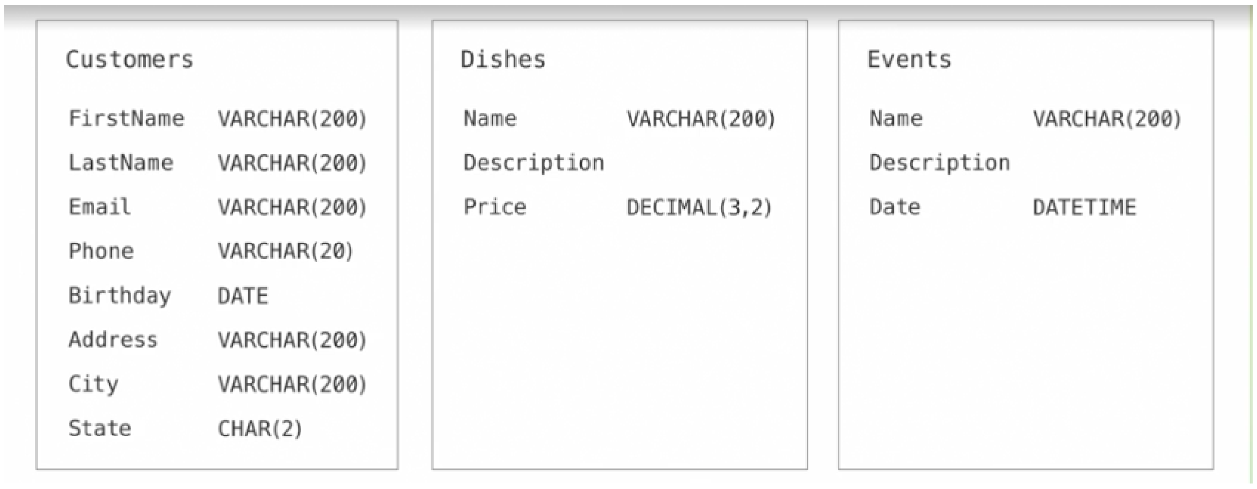
You want to use other types for longer text, because if you do VARCHAR (10) & the word is 11 characters long it will be cut off.

Will adapt and downsize to save memory. CHAR(20) will be 20 spaces no matter what.

- Dates & Times
- DATE - 2019-03-09 …….birthdays.
- DATETIME - 2013-03-09 16:51:00 …. Events & reservations
- TIMESTAMP - (saved when record is updated) when an order is placed

DATA TYPES FOR NUMBERS

- Integers, Double precision, Floating Point, Decimals
- It is important to be precise when choosing the right data type for numbers so there are no pitfalls when rounding.
- Null is not a data type, it is a condition.
- If a cell has a value, it is NOT NULL. Even if it contains the text null.. It is a string.



Customers

FirstName	VARCHAR(200)
LastName	VARCHAR(200)
Email	VARCHAR(200)
Phone	VARCHAR(20)
Birthday	DATE
Address	VARCHAR(200)
City	VARCHAR(200)
State	CHAR(2)

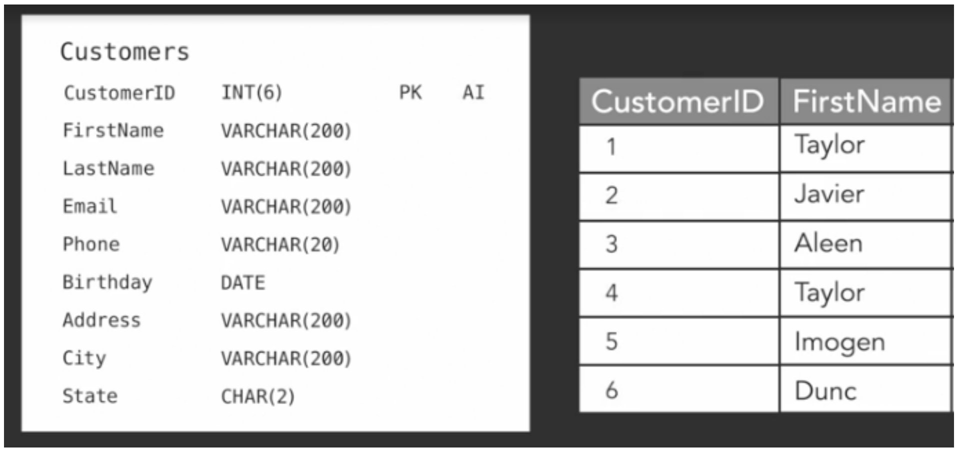
Dishes

Name	VARCHAR(200)
Description	
Price	DECIMAL(3,2)

Events

Name	VARCHAR(200)
Description	
Date	DATETIME

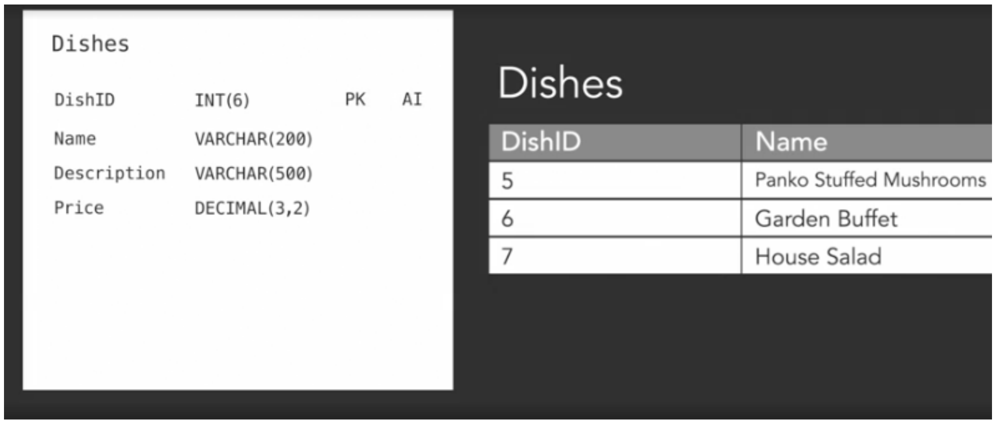
PRIMARY & FOREIGN KEYS



Customers

```
CustomerID   INT(6)          PK    AI
FirstName    VARCHAR(200)
LastName     VARCHAR(200)
Email        VARCHAR(200)
Phone        VARCHAR(20)
Birthday     DATE
Address      VARCHAR(200)
City         VARCHAR(200)
State        CHAR(2)
```

CustomerID	FirstName
1	Taylor
2	Javier
3	Aleen
4	Taylor
5	Imogen
6	Dunc

- None of the information in the customers table was reliably unique. Therefore we created a CustomerID column as a numeric type INT(6), PK, and did an AI (auto increment)
- An integer key is great for simple examples, but many people use a UUID which is much longer & therefore more difficult for a hacker to guess.
- 

Dishes

```
DishID       INT(6)          PK    AI
Name         VARCHAR(200)
Description  VARCHAR(500)
Price        DECIMAL(3,2)
```

Dishes

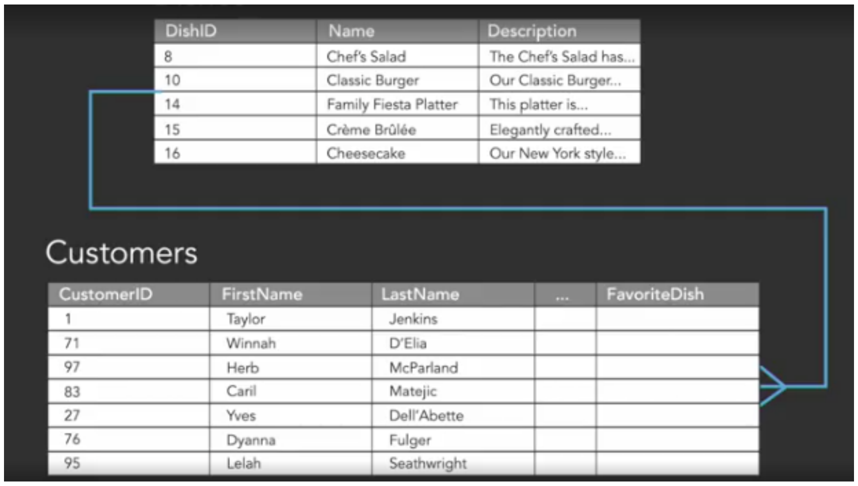
DishID	Name
5	Panko Stuffed Mushrooms
6	Garden Buffet
7	House Salad

- The same is done for the dish table, as there is nothing to uniquely identify, and changing the name of the dishes if it was a primary key would cause problems later down the line.

RELATIONSHIPS

ONE TO MANY RELATIONSHIP

- It connects one piece of data to one or more other pieces of data.
- For example, we want to store a customer's favorite type of dish in the customer table. By writing in the favorite dish in the dish column it can cause big problems when doing maintenance. For instance having a spelling error we would be changing it in two tables.

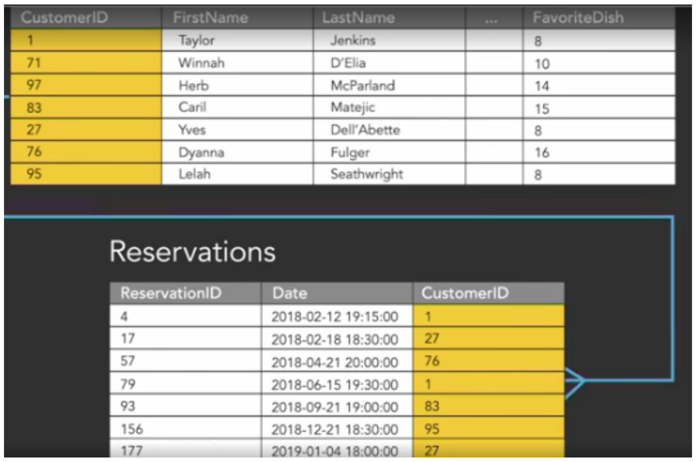


DishID	Name	Description
8	Chef's Salad	The Chef's Salad has...
10	Classic Burger	Our Classic Burger...
14	Family Fiesta Platter	This platter is...
15	Crème Brûlée	Elegantly crafted...
16	Cheesecake	Our New York style...

Customers

CustomerID	FirstName	LastName	...	FavoriteDish
1	Taylor	Jenkins		
71	Winnah	D'Elia		
97	Herb	McParland		
83	Caril	Matejic		
27	Yves	Dell'Abette		
76	Dyanna	Fulger		
95	Lelah	Seathwright		

- We use a one to many relationship here because one dish may be the favorite of many different customers. The foreign key is on the many side.
- There is no such thing that has a many to one relationship, it is just a matter of how you look at it.



CustomerID	FirstName	LastName	...	FavoriteDish
1	Taylor	Jenkins		8
71	Winnah	D'Elia		10
97	Herb	McParland		14
83	Caril	Matejic		15
27	Yves	Dell'Abette		8
76	Dyanna	Fulger		16
95	Lelah	Seathwright		8

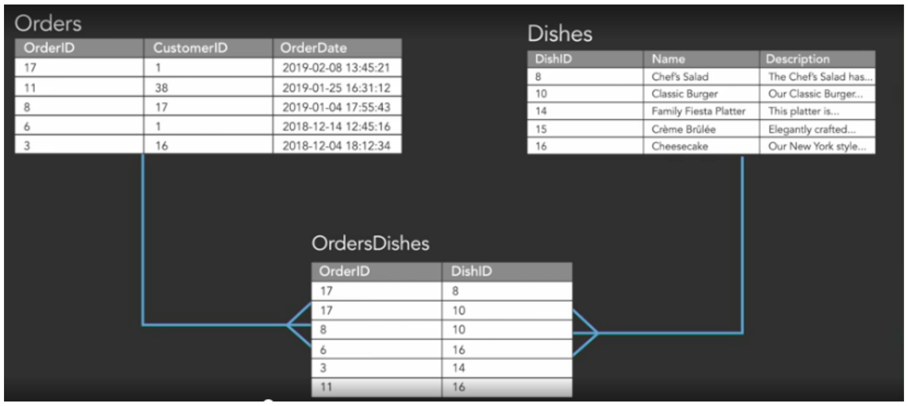
Reservations

ReservationID	Date	CustomerID
4	2018-02-12 19:15:00	1
17	2018-02-18 18:30:00	27
57	2018-04-21 20:00:00	76
79	2018-06-15 19:30:00	1
93	2018-09-21 19:00:00	83
156	2018-12-21 18:30:00	95
177	2019-01-04 18:00:00	27

- One record is being associated with many records. One customer can have many reservations. PK in Customer table, and FK in reservations.

MANY TO MANY RELATIONSHIPS

- This model is useful when we want to associate more than one thing with more than one other thing. For example, down below we utilize a linking table to show what multiple dishes compose an order. When we want to call up information about a customer's order, we will ask the database for rows matching the order numbers, and we'll get back info matching dish ids. It could 1, 5, 10, or 50 depending on how many orders from that specific CustomerID. This keeps our orders table nice & clean while allowing us to record the details of what each order included. It lets us easily see how many orders a particular dish was included in.



Orders

OrderID	CustomerID	OrderDate
17	1	2019-02-08 13:45:21
11	38	2019-01-25 16:31:12
8	17	2019-01-04 17:55:43
6	1	2018-12-14 12:45:16
3	16	2018-12-04 18:12:34

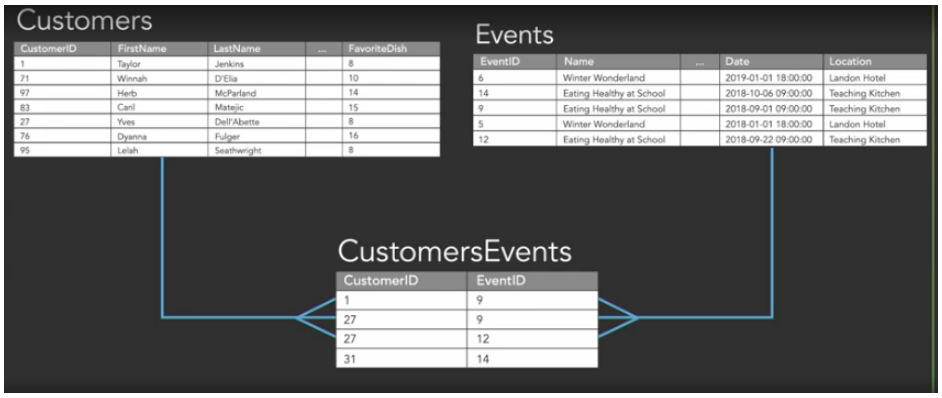
Dishes

DishID	Name	Description
8	Chef's Salad	The Chef's Salad has...
10	Classic Burger	Our Classic Burger...
14	Family Fiesta Platter	This platter is...
15	Crème Brûlée	Elegantly crafted...
16	Cheesecake	Our New York style...

OrdersDishes

OrderID	DishID
17	8
17	10
8	10
6	16
3	14
11	16

- A customers events linking table would allow us to be able to get an insight on which events particular customers are interested in. Keep in mind you may need to add in some linking tables to create some relationships.



Customers

CustomerID	FirstName	LastName	...	FavoriteDish
1	Taylor	Jenkins		8
71	Winnah	D'Elia		10
97	Herb	McParland		14
83	Caril	Matejic		15
27	Yves	Dell'Abette		8
76	Dyanna	Fulger		16
95	Lelah	Seathwright		8

Events

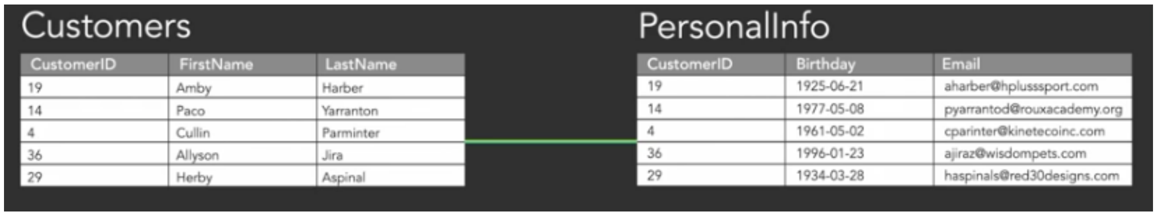
EventID	Name	...	Date	Location
6	Winter Wonderland		2019-01-01 18:00:00	Landon Hotel
14	Eating Healthy at School		2018-10-06 09:00:00	Teaching Kitchen
9	Eating Healthy at School		2018-09-01 09:00:00	Teaching Kitchen
5	Winter Wonderland		2018-01-01 18:00:00	Landon Hotel
12	Eating Healthy at School		2018-09-22 09:00:00	Teaching Kitchen

CustomersEvents

CustomerID	EventID
1	9
27	9
27	12
31	14

ONE TO ONE RELATIONSHIP

- This is not frequently used. Because if one row is only linked to one row it suggests that the two should just be in one table.



Customers

CustomerID	FirstName	LastName
19	Amby	Harber
14	Paco	Yarranton
4	Cullin	Parminter
36	Allyson	Jira
29	Herby	Aspinal

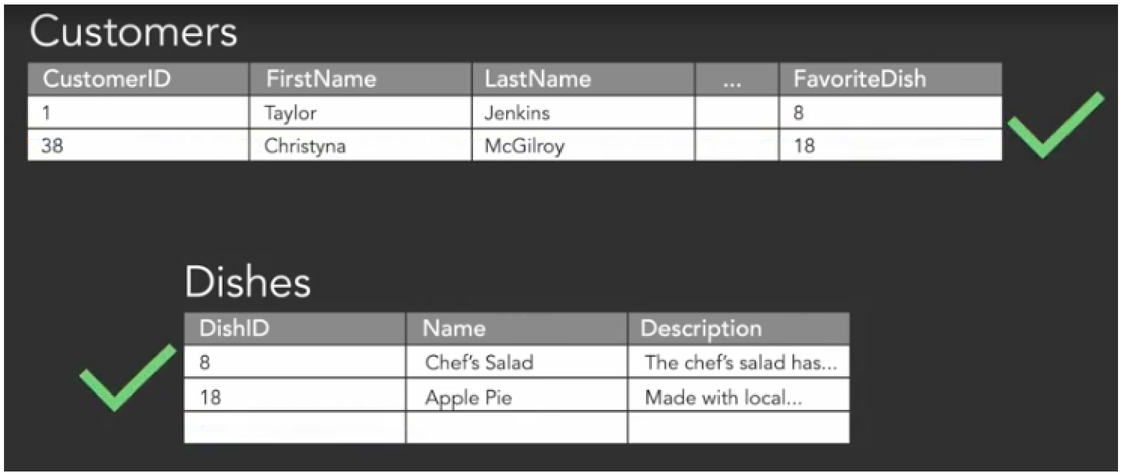
PersonalInfo

CustomerID	Birthday	Email
19	1925-06-21	aharber@hplussport.com
14	1977-05-08	pyarrantod@rouxacademy.org
4	1961-05-02	cparinter@kinetecoinc.com
36	1996-01-23	ajiraz@wisdompets.com
29	1934-03-28	haspinals@red30designs.com

- In this case a one to one relationship is necessary because the database keeps customer info private in another table. While leaving their name & key available to use in other relationships.

RELATIONSHIP RULES AND REFERENTIAL INTEGRITY

- Referential integrity is when databases are aware of relationships and won't allow a user to modify data in a way that violates those relationships.



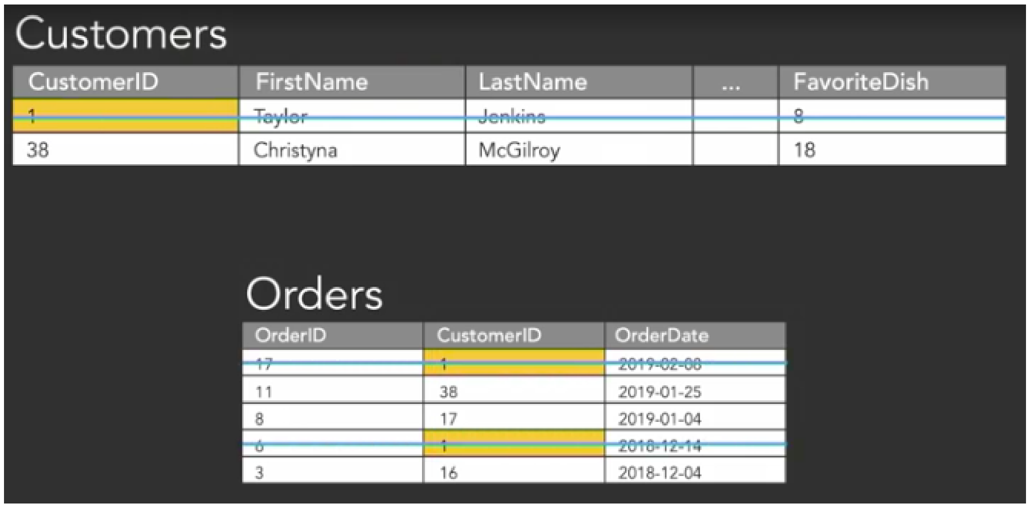
Customers

CustomerID	FirstName	LastName	...	FavoriteDish
1	Taylor	Jenkins		8
38	Christyna	McGilroy		18

Dishes

DishID	Name	Description
8	Chef's Salad	The chef's salad has...
18	Apple Pie	Made with local...

- For example for CustomerID 38 if we attempted to input a favorite dish of 999, the table would reject the row. This helps keep the DB consistent and accurate. We do not want to worry about having bad data in the table, and lets the table do the work of preventing.



Customers

CustomerID	FirstName	LastName	...	FavoriteDish
~~1~~	~~Taylor~~	~~Jenkins~~		~~8~~
38	Christyna	McGilroy		18

Orders

OrderID	CustomerID	OrderDate
~~17~~	~~1~~	~~2019-02-08~~
11	38	2019-01-25
8	17	2019-01-04
~~6~~	~~1~~	~~2018-12-14~~
3	16	2018-12-04

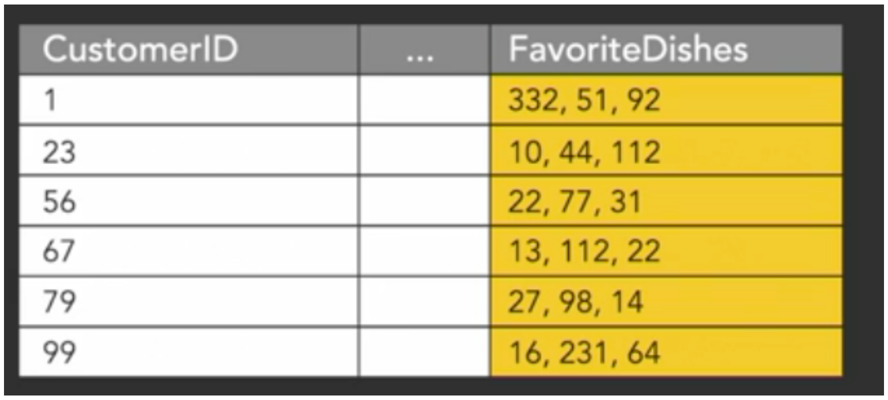
- When a CustomerID is deleted it performs a cascading delete where all orders are also deleted that correspond with that CustomerID.
- However, we do not want to do that with everything. For example we would want to retain the dishes.

<u>DATABASE OPTIMIZATION</u>

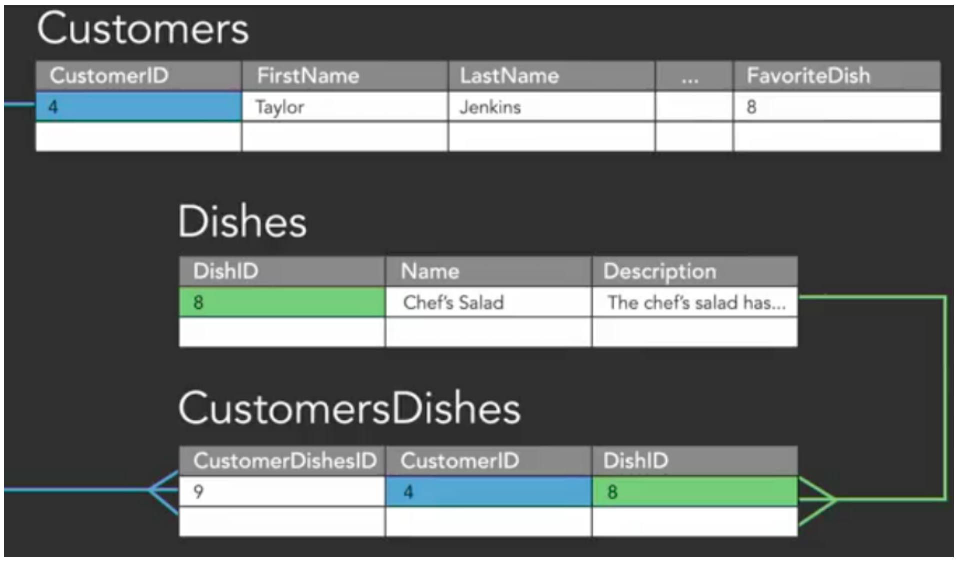
<u>NORMALIZATION</u>

<u>1NF</u>

- First Normal Form requires that values in each cell should be atomic and tables should have no repeating groups.
- Violations of 1NF are usually pretty easy to spot. Such as putting multiple values in a cell or creating columns that end in numbers such as favorite dish1, favorite dish2…
- Below is an example of a table violating 1NF, and in order to fix it we create a linking table with its own ID. We join many dishes, with many customers.



CustomerID	...	FavoriteDishes
1		332, 51, 92
23		10, 44, 112
56		22, 77, 31
67		13, 112, 22
79		27, 98, 14
99		16, 231, 64



Customers

CustomerID	FirstName	LastName	...	FavoriteDish
4	Taylor	Jenkins		8

Dishes

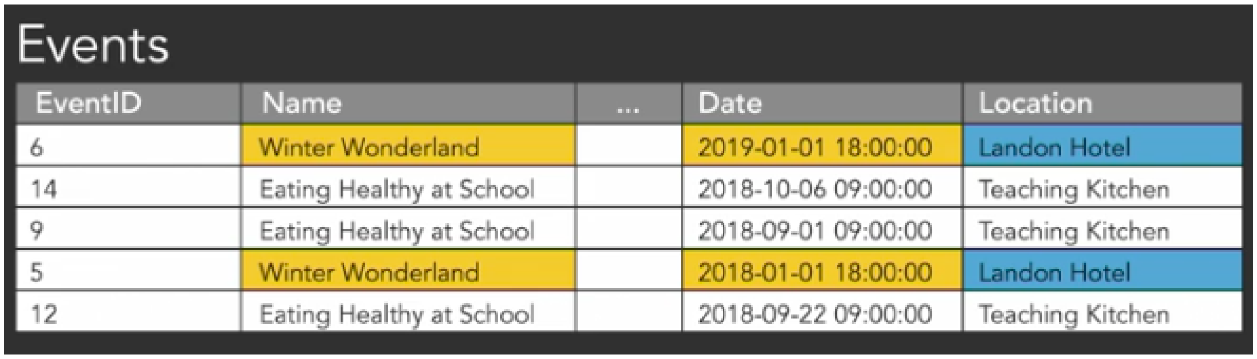
DishID	Name	Description
8	Chef's Salad	The chef's salad has...

CustomersDishes

CustomerDishesID	CustomerID	DishID
9	4	8

<u>2NF</u>

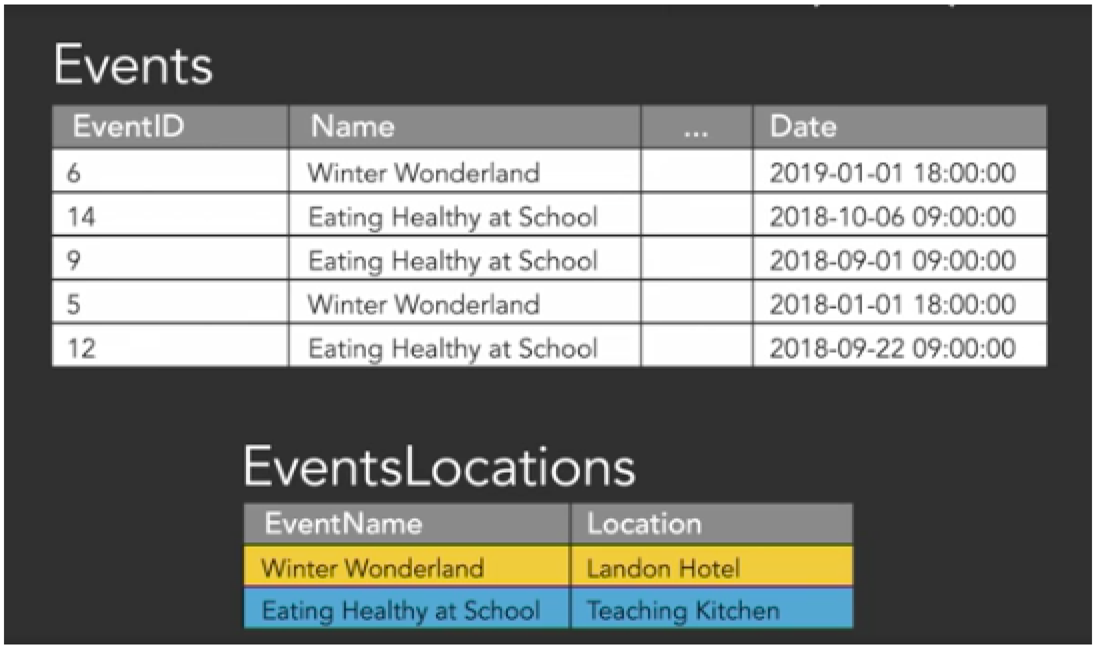
- No value in a table should depend on only part of a key that can be used to uniquely identify a row.
- Every column in the table that isn't a key; each of the values must rely on only the whole key. The values must describe something about that row that we cant determine from just part of a key. This problem comes up in the context of composite keys. Down below name & date form a composite key. Location is only dependent on the name key.



Events

EventID	Name	...	Date	Location
6	Winter Wonderland		2019-01-01 18:00:00	Landon Hotel
14	Eating Healthy at School		2018-10-06 09:00:00	Teaching Kitchen
9	Eating Healthy at School		2018-09-01 09:00:00	Teaching Kitchen
5	Winter Wonderland		2018-01-01 18:00:00	Landon Hotel
12	Eating Healthy at School		2018-09-22 09:00:00	Teaching Kitchen

The location is not dependent on the full composite key. We know where the location of the event will be held just based on the name of the event. Now the new table below reflects the fact that each event is just held in one place.



Events

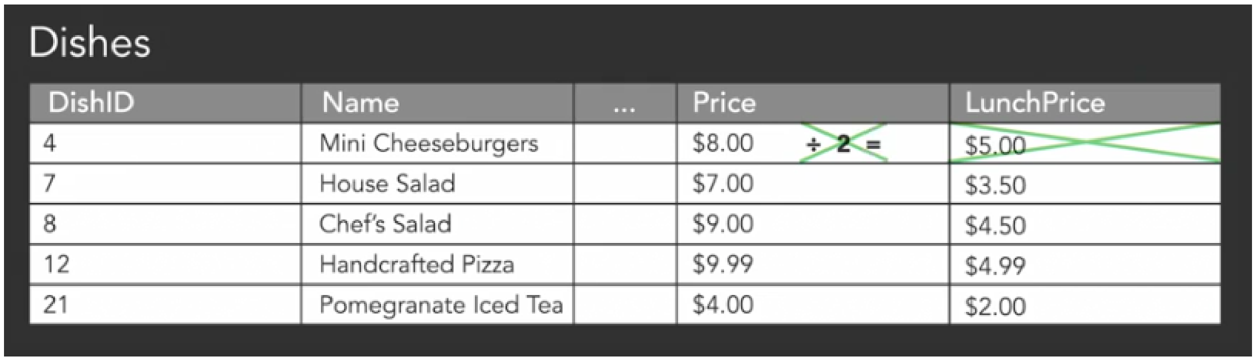
EventID	Name	...	Date
6	Winter Wonderland		2019-01-01 18:00:00
14	Eating Healthy at School		2018-10-06 09:00:00
9	Eating Healthy at School		2018-09-01 09:00:00
5	Winter Wonderland		2018-01-01 18:00:00
12	Eating Healthy at School		2018-09-22 09:00:00

EventsLocations

EventName	Location
Winter Wonderland	Landon Hotel
Eating Healthy at School	Teaching Kitchen

<u>3NF</u>

- Values should not be stored if they can be calculated from another non-key field.
- In the example below, lunch price depends on the price column. Therefore calculated from another field. In other words, lunch price must be dependent and unique.

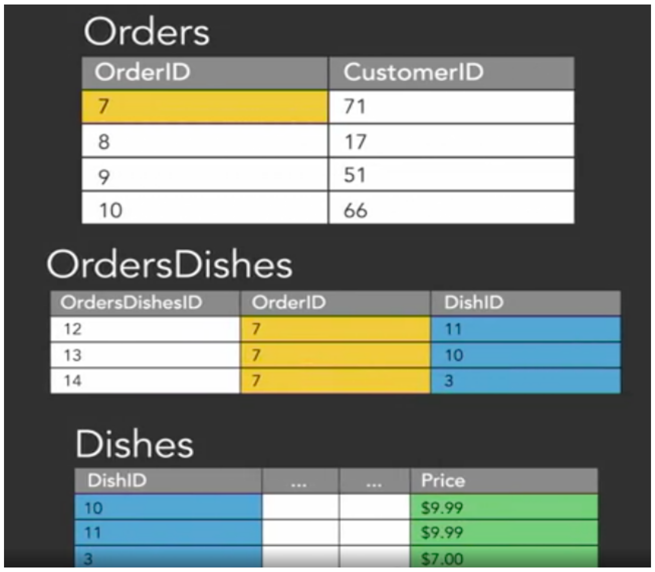


Dishes

DishID	Name	...	Price	LunchPrice
4	Mini Cheeseburgers		$8.00 ~~÷ 2 =~~	~~$5.00~~
7	House Salad		$7.00	$3.50
8	Chef's Salad		$9.00	$4.50
12	Handcrafted Pizza		$9.99	$4.99
21	Pomegranate Iced Tea		$4.00	$2.00

DENORMALIZATION

- The process of intentionally duplicating information in a table, in violation of normalization rules. Denormalization is done to a previously normalized database.
- Denormalization is a trade-off. Gaining speed may reduce consistency.



Orders

OrderID	CustomerID
7	71
8	17
9	51
10	66

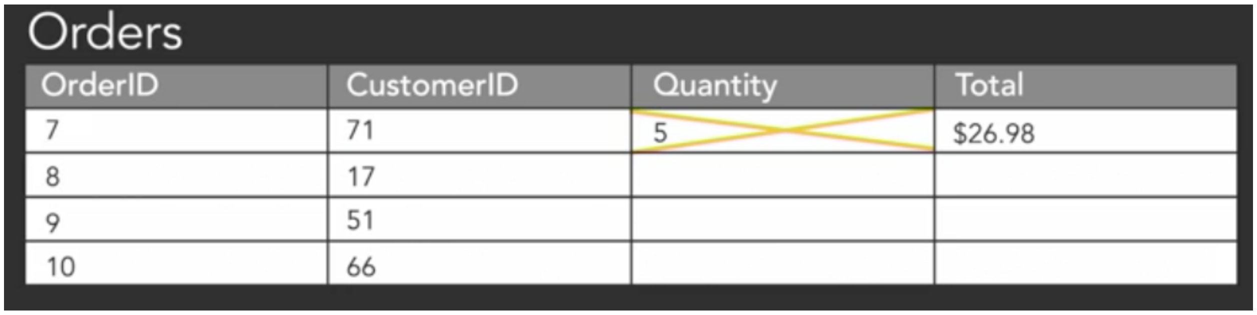
OrdersDishes

OrdersDishesID	OrderID	DishID
12	7	11
13	7	10
14	7	3

Dishes

DishID	...	...	Price
10			$9.99
11			$9.99
3			$7.00

- By using the order ID, we can get the associated items in the orders dishes table, count them up, and pull information from the dishes table to get the price of each item. Example: Order ID, and DishID are passed to a linking table to accurately total up the price of an order.



Orders

OrderID	CustomerID	Quantity	Total
7	71	~~5~~	$26.98
8	17		
9	51		
10	66		

- If someone changed the quantity within the orders table, it would cause the linking table to be inaccurate, and damage referential integrity.