Brandon Litwin

Alan Labouseur

CMPT 308N-112

January 27, 2017

Agents

```
postgres on postgres@PostgreSQL 9.6
142         (1026, 'May', 'c002', 'a05', 'p03',  800,  744.00);
143
144
145 -- SQL statements for displaying the example data
146
147 select *
148 from Customers;
149
150 select *
151 from Agents;
152
```

Data Output | Explain | Messages | History

aid character	name text	city text	commissi... numeric ...
a01	Smith	New York	6.5
a02	Jones	Newark	6
a03	Perry	Tokyo	7
a04	Grey	New York	6
a05	Otasi	Duluth	5
a06	Smith	Dallas	5
a08	Bond	London	7.07

Customers

```
postgres on postgres@PostgreSQL 9.6
142         (1026, 'May', 'c002', 'a05', 'p03',  800,  744.00);
143
144
145 -- SQL statements for displaying the example data
146
147 select *
148 from Customers;
149
150 select *
151 from Agents;
152
```

Data Output | Explain | Messages | History

cid character	name text	city text	discount numeric ...
c001	Tiptop	Duluth	10
c002	Tyrell	Dallas	12
c003	Allied	Dallas	8
c004	ACME	Duluth	8.5
c005	Weyland	Risa	0
c006	ACME	Kyoto	0

Products

```
postgres on postgres@PostgreSQL 9.6
148  from Customers;
149
150  select *
151  from Agents;
152
153  select *
154  from Products;
155
156  select *
157  from Orders;
158
```

Data Output | Explain | Messages | History

pid character	name text	city text	quantity integer	priceusd numeric
p01	comb	Dallas	111400	0.5
p02	brush	Newark	203000	0.5
p03	razor	Duluth	150600	1
p04	pen	Duluth	125300	1
p05	pencil	Dallas	221400	1
p06	trapper	Dallas	123100	2
p07	case	Newark	100500	1
p08	eraser	Newark	200600	1.25

Orders

```
postgres on postgres@PostgreSQL 9.6
148  from Customers;
149
150  select *
151  from Agents;
152
153  select *
154  from Products;
155
156  select *
157  from Orders;
158
```

Data Output | Explain | Messages | History

ordnumb integer	month character	cid character	aid character	pid character	qty integer	totalusd numeric
1011	Jan	c001	a01	p01	1000	450
1012	Jan	c002	a03	p03	1000	880
1015	Jan	c003	a03	p05	1200	1104
1016	Jan	c006	a01	p01	1000	500
1017	Feb	c001	a06	p03	600	540
1018	Feb	c001	a03	p04	600	540
1019	Feb	c001	a02	p02	400	180
1020	Feb	c006	a03	p07	600	600
1021	Feb	c004	a06	p01	1000	460
1022	Mar	c001	a05	p06	400	720
1023	Mar	c001	a04	p05	500	450
1024	Mar	c006	a06	p01	800	400
1025	Apr	c001	a05	p07	800	720
1026	May	c002	a05	p03	800	744

Keys

In a database, a primary key refers to at least one column in a table that can be used to uniquely identify all the records in the table (https://www.techopedia.com). No two pieces of data within a primary key are the same. An example of a primary key used in everyday life is a social security number. In a database of every person's social security numbers, each one would be unique, meaning that social security numbers are a primary key in the database. There can only be one primary key in each database. A candidate key is the same as a primary key, in that it is a

column or set of columns that can uniquely identify a database. A table can have multiple candidate keys. A primary key is just a candidate key that is selected to be the one that best identifies the data table as a whole. Finally, a superkey is a combination of candidate keys that can also be used to identify a database. A superkey can be any size, while a candidate key is the minimun number of rows required to uniquely identify a database.

Data Types

Each piece of data in a table has a data type. There are many different data types that refer to what kind on information is represented in the data, but the most common ones are VARCHAR, BOOLEAN, INT, and TIMESTAMP. Data types can be nullable, meaning their value can be recorded as unknown in the data table. There are some data that are not nullable because they are required to be unique values in the table. Primary keys are not nullable because they must be used to uniquely identify the items in the table. Timestamps are also not nullable because the only valid data for a timestamp data type is a time. If I made a table that stored people's flight booking information, here are the fields and data types I would use:

fid	**firstName**	**lastName**	**destination**	**priceInUSD**	**timeOfReservation**
(VARCHAR)	(VARCHAR)	(VARCHAR)	(VARCHAR)	(INT)	(TIMESTAMP)
(Not Nullable)	(Nullable)	(Nullable)	(Nullable)	(Nullable)	(Not Nullable)

In this example, Fid is the primary key so it cannot be null. The timeOfReservation also cannot be null because it is a timestamp data type.

Relational Rules

The rule of "first normal form" describes the basic layout of a database. In first normal form, all data items must be defined, there must be no repeating groups of data, and there must be a primary key in the table. In general, each column can only have one data item per row. If,

for example, a table had a row called “Product 1” and a column called “color”, that item cannot have two colors in the same row and column. If “Product 1” comes in two colors, then there must be one row for “Product 1” and “Blue”, and a row below it for “Product 1” and “Green”. It would not follow first normal form if in the “Product 1” row there were colors “Blue, Green”.

The rule of “access rows by content only” means that rows and columns should not be referred to by their numerical positions in the table, e.g. “the fourth row”, but by their content, e.g. “numberOfPotatoes”. An example that goes against this rule is having the content be a pointer to another row with the same value instead of typing the repeated value. If two positions in the table have the same value, then creating another table may be necessary. This new table would have a foreign key that directly relates to the previous table. This rule is mainly in place to encourage users to think of relations between tables as “what” instead of “where”. This is one of the key differences between relational databases and older database models.
http://stackoverflow.com/questions/1546147/database-pointers-to-rows

The rule of “all rows must be unique” means that no rows in the database can have the exact same values in each column. If two rows were exactly the same, it would represent repeated data and would not make sense in a relational database. This is avoided by having a primary key such as rowID that has a different number for every row.