

Exploratory Data Analysis with R

Introduction to SQL

Xuemao Zhang
East Stroudsburg University

November 21, 2022

Outline

- Database
- Introduction to SQL
- PostgreSQL and pgAdmin

Database

- What is a database?
 - ▶ A database is a collection of persistent data
 - ▶ A database models part of the real world
 - ▶ A database is, in general, a shared resource
- Examples of database include:
 - ▶ banking, shopping, . . .
 - ▶ scientific investigations, e.g. astronomical data, human genome
- DBMS (database-management system)
 - ▶ A DBMS is specialised software which is responsible for efficient storage and retrieval of large amounts of data in a database, allowing it to persist over long periods of time.
 - ▶ A DBMS is also referred to more simply as a database system.

Database

- Why do we study database management?
 - ▶ The database market is huge and there's a big demand for database skills
 - ▶ Managing data is a fundamental need for most applications
 - ★ It saves programmer time by providing a declarative query language, e.g. SQL.
 - ★ It saves programmer time by automatically checking constraints.
 - ★ It saves maintenance time by ensuring data independence.
 - ★ It provides concurrent access to the database for multiple, simultaneous users.
 - ★ It provides automatic recovery from failure.
 - ★ It provides security to ensure appropriate access to data.
- We'll use [PostgreSQL](#) in this course

Database

- PostgreSQL is a free and open-source **relational database management system** (RDBMS)
- A relational database is a database containing one or more tables of information.
 - ▶ The rows in a table are called **records** and
 - ▶ the columns in a table are called **fields** or **attributes**.
- A database that contains only one table is called a flat database.
- A database that contains two or more related tables is called a relational database.
- Let's use an example to understand the concept of relational database
 - ▶ The example is from [Relational Database Concepts for Beginners](https://webs.wofford.edu/whisnantdm/Courses/CS101/PDF/Database/Relational_database_concepts.pdf)
https://webs.wofford.edu/whisnantdm/Courses/CS101/PDF/Database/Relational_database_concepts.pdf

Database

- Imagine that you are responsible for keeping track of all the books being checked out of a library. You could use a single table (a flat database) to track all the critical information:

First Name	Last Name	Address	Phone	Book Title	Due Date
Bob	Smith	123 Main St.	555-1212	Don Quixote	7-14-09

- This table meets the basic need to keep track of who has checked out which book, but does have some serious flaws in terms of efficiency, space required, and maintenance time. For example, as voracious reader Bob checks out more books over time, you will have to re-enter all of his contact information for every book.

First Name	Last Name	Address	Phone	Book Title	Due Date
Bob	Smith	123 Main St.	555-1212	Don Quixote	7-14-09
Alicia	Petersohn	136 Oak St.	555-1234	Three Men in a Boat	7-16-09
Bob	Smith	123 Main St.	555-1212	Things Fall Apart	8-15-09
Bob	Smith	123 Main St.	555-1212	Anna Karenina	8-15-09
Zayn	Murray	248 Pine Dr.	555-1248	Heidi	8-17-09
Bob	Smith	123 Main St.	555-1212	The Old Man and the Sea	9-10-09

Database

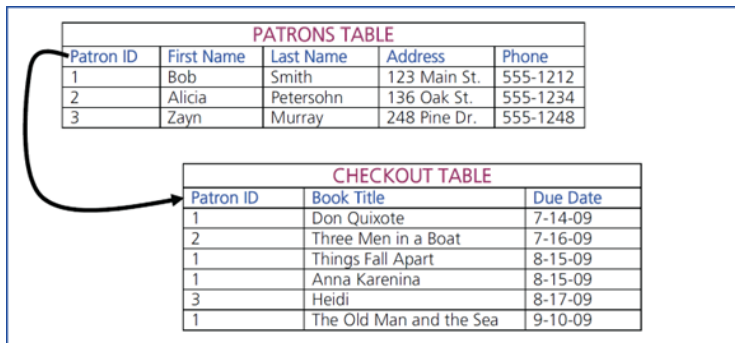
- Moreover, when an update is necessary (e.g. Bob's phone number changes), each of Bob's records must be located and corrected. If one of Bob's records has a different phone number from the rest, is it a correction, a record overlooked during the last update, or a data-entry mistake?
- These problems can be decreased by normalizing our data – in other words, dividing the information into multiple tables with the goal of having “a place for everything, and everything in its place.” Each piece of information should appear just once, simplifying data maintenance and decreasing the storage space required.

PATRONS TABLE			
First Name	Last Name	Address	Phone
Bob	Smith	123 Main St.	555-1212
Alicia	Petersohn	136 Oak St.	555-1234
Zayn	Murray	248 Pine Dr.	555-1248

CHECKOUT TABLE	
Book Title	Due Date
Don Quixote	7-14-09
Three Men in a Boat	7-16-09
Things Fall Apart	8-15-09
Anna Karenina	8-15-09
Heidi	8-17-09
The Old Man and the Sea	9-10-09

Database

- We need a way to show which records in the PATRONS table correspond to which records in the CHECKOUT table – in other words, who checked out which book.
 - ▶ Instead of repeating everything we know about a patron whenever he checks out a book, we will instead give each library patron an ID, and repeat only the ID whenever we want to associate that person with a record in a different table.



Database


- Now the PATRONS and CHECKOUT tables can be related (how relationships are formally declared in various database software is beyond the scope of this paper). At this point, we need some new terms to talk about our related tables.
- The **primary key** is a field whose values are **unique** in this table, and so can be used as **identifiers** for the records (multi-field or composite primary keys are beyond the scope of this paper, and are unlikely in an ArcGIS geodatabase).
 - ▶ In table PATRONS, the Patron ID field is the primary key and so its values must remain unique. For example, the value "2" can appear only on one record - Alicia's - and Alicia can have only one Patron ID - "2."
- Is the Patron ID field in table CHECKOUT the primary key?
 - ▶ We can see that it contains duplicate values, so the answer is No.
 - ▶ If Patron ID were the primary key for CHECKOUT, each person would only be permitted to check out one book, and afterward would be forbidden to check out any more books, ever.

Database

- So if Patron ID is not the primary key for table CHECKOUT, which field is?
 - ▶ We can't make Book Title the primary key, or we'd have a similar problem – each book could only be checked out once, and afterward no one would be permitted to check it out ever again.
 - ▶ We can't make Due Date the primary key, or else only one book could be due each day.
- Since none of the existing fields works as a primary key, we will add a new field to hold an identifier for each record. We could name this field Checkout ID, or we could follow ESRI's convention of giving all primary key fields exactly the same name: ObjectID.

PATRONS TABLE				
ObjectID	First Name	Last Name	Address	Phone
1	Bob	Smith	123 Main St.	555-1212
2	Alicia	Petersohn	136 Oak St.	555-1234
3	Zayn	Murray	248 Pine Dr.	555-1248

CHECKOUT TABLE			
ObjectID	Patron ObjectID	Book Title	Due Date
1	1	Don Quixote	7-14-09
2	2	Three Men in a Boat	7-16-09
3	1	Things Fall Apart	8-15-09
4	1	Anna Karenina	8-15-09
5	3	Heidi	8-17-09
6	1	The Old Man and the Sea	9-10-09



Database

- Naming every primary key field **ObjectID** does make it easy to tell at a glance which field uniquely identifies the records in this table.
 - ▶ We can also use this naming convention to provide hints about which fields are related. For example, Patron ObjectID in CHECKOUT is related to ObjectID in PATRONS.
- To further increase efficiency, decrease required space, and improve ease of maintenance, we can separate the book information into its own table.

Database

- Now ObjectID in BOOKS is related to Book ObjectID in CHECKOUT.

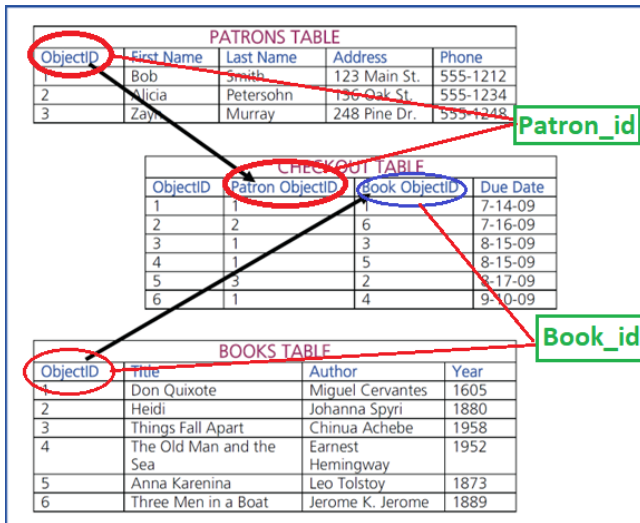
PATRONS TABLE				
ObjectID	First Name	Last Name	Address	Phone
1	Bob	Smith	123 Main St.	555-1212
2	Alicia	Petersohn	136 Oak St.	555-1234
3	Zayne	Murray	248 Pine Dr.	555-1248

CHECKOUT TABLE			
ObjectID	Patron ObjectID	Book ObjectID	Due Date
1	1	1	7-14-09
2	2	6	7-16-09
3	1	3	8-15-09
4	1	5	8-15-09
5	3	2	8-17-09
6	1	4	9-10-09

BOOKS TABLE			
ObjectID	Title	Author	Year
1	Don Quixote	Miguel Cervantes	1605
2	Heidi	Johanna Spyri	1880
3	Things Fall Apart	Chinua Achebe	1958
4	The Old Man and the Sea	Ernest Hemingway	1952
5	Anna Karenina	Leo Tolstoy	1873
6	Three Men in a Boat	Jerome K. Jerome	1889

Database

- It is better to name same variables the same names. See below.



Database

- When two tables have an unequal relationship, we call the independent table the **parent** and the dependent table the **child**.
 - ▶ You can identify the parent table by determining which table could contain a record without needing a corresponding record in the table on the other side of the relationship.
 - ▶ Another way to identify the child table is to find the field which refers to the other table's ObjectID. BOOKS does not contain an ObjectID field for the CHECKOUTS, but CHECKOUTS does contain a field to store Book ObjectIDs. Therefore, CHECKOUTS is the child table in this relationship.
- In the above database
 - ▶ PATRONS is the parent, and CHECKOUT is the child.
 - ▶ BOOKS is the parent, and CHECKOUT is the child.
- If somehow the child table contains a record that does not have a corresponding record in the parent table, that record is called an Orphan. **Orphaned records** are a problem that generally requires attention from the database administrator.

Database

- **Attribute:** In relational databases, attributes are the describing characteristics or properties that define all items pertaining to a certain category applied to all cells of a column.
 - ▶ The rows, instead, are called **tuples**, and represent data sets applied to a single entity to uniquely identify each item.
 - ▶ Attributes are, therefore, the characteristics of every individual tuple that help describe its unique properties.
 - ▶ Think of a table in a relational database as being analogous to an electronic spreadsheet. An attribute is simply one non-null cell in the spreadsheet, or the conjunction of a column and row.
 - ▶ An attribute can be composite composed of several other simple attributes. For example, the Address attribute of an Employee entity could consist of the Street, City, Postal code and Country attributes.
 - ▶ There are several other types of attributes.

Database

Foreign Key is the attribute of a table that is used in identifying a **Primary Key** of another table.

- Each Patron in the table CHECKOUT is from the table PATRONS. That is, the Primary Key ObjectID of the table PATRONS is added to the table CHECKOUT that helps in relating these two tables.
 - ▶ That is, **Patron ObjectID** is a **Foreign Key**.
- Similarly, **Book ObjectID** in the table CHECKOUT is a **Foreign Key** to relate the table BOOKS with Primary Key **ObjectID**.

Database

- **Candidate Key:** Candidate Key is a set of columns or a single column which helps in uniquely recognizing any database record without referring to any other information.
 - ▶ It is as strong as the Primary Key.
- **Super Key** is a combination of columns that helps in uniquely recognizing any row in a table.
 - ▶ Candidate Key is related to Super Key wherein the latter is a superset of the candidate key. That is, a Candidate Key must be a Super Key.

Database

- The last new concept to consider is **cardinality**, which describes how many records in one table can be related to records in another table. Two tables might have a cardinality of
 - ▶ 1 - 1 (one to one),
 - ▶ 1 - ∞ (one to many),
 - ▶ 1 - 3 (one to three),
 - ▶ ∞ - ∞ (many to many), etc.
- The PATRONS – CHECKOUT relationship has a 1 - ∞ cardinality,
 - ▶ because 1 patron may have any number of checkouts, from zero to infinity.
 - ▶ Put another way, the CHECKOUT – PATRONS relationship has a cardinality of ∞ - 1.
 - ▶ If the cardinality of PATRONS – CHECKOUT were 1 - 1, then each patron could check out only one book.
 - ▶ If the cardinality of PATRONS – CHECKOUT were ∞ - ∞ , then several patrons together might share joint responsibility for one or more checkouts.
- The BOOKS – CHECKOUT relationship is also 1 - ∞ , since one book may be checked out multiple times.

Introduction to SQL

- SQL stands for Structured Query Language
 - ▶ can be pronounced 'sequel' as well
- It is a very-high-level (declarative) language
 - ▶ user specifies what is wanted
- History of standards
 - ▶ original ANSI SQL in 1986
 - ▶ major revision in 1992 to SQL-92 or SQL2
 - ▶ most recent is SQL:1999 (also called SQL3)
 - ▶ other extensions SQL:2003, SQL:2006, SQL:2008, SQL:2011, SQL:2016, and SQL:2019
- We learn RDBMS(Relational Database Management System)
- DBMS vendors
 - ▶ there are different versions of the SQL language
 - ▶ all support ANSI SQL and largely SQL-92
 - ▶ some of SQL:1999 and SQL:2003
 - ▶ with variations and own extensions

Introduction to SQL

- SQL is a data definition and manipulation language
- What Can SQL do?
 - ▶ SQL can execute queries against a database
 - ▶ SQL can retrieve data from a database
 - ▶ SQL can insert records in a database
 - ▶ SQL can update records in a database
 - ▶ SQL can delete records from a database
 - ▶ SQL can create new databases
 - ▶ SQL can create new tables in a database
 - ▶ SQL can create stored procedures in a database
 - ▶ SQL can create views in a database
 - ▶ SQL can set permissions on tables, procedures, and views

Introduction to SQL

Some SQL Data Types

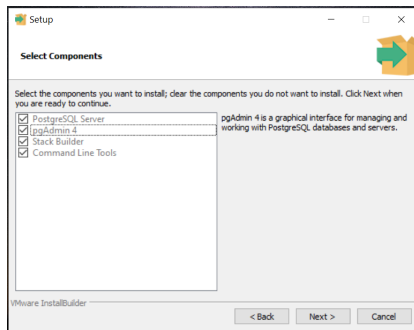
Data type	Description
char [(n)]	Character string with fixed-length n
character varying [(n)] or varchar[(n)]	Character string; with maximum length n
smallint or int2	Signed two-byte integer; Small integer. Precision 5
integer or int	Signed four-byte integer. Precision 10
decimal [(p, s)] or numeric [(p, s)]	Exact numerical, precision p , scale s . Example: decimal(5,2) is a number that has 3 digits before the decimal and 2 digits after the decimal
real or float4	single precision floating-point number (4 bytes)
double precision or float8	double precision floating-point number (8 bytes)
date	Stores year, month, and day values
time	Stores hour, minute, and second values
timetz	time of day, including time zone
timestamp [(p)] [without time zone]	date and time (no time zone)
timestampz	date and time, including time zone

PostgreSQL and pgAdmin

- **PostgreSQL** is a powerful, open source object-relational database system with over 30 years of active development that has earned it a strong reputation for reliability, feature robustness, and performance.
- **A Brief History of PostgreSQL**
<https://www.postgresql.org/docs/current/history.html>
- **pgAdmin** <https://www.pgadmin.org/>
 - ▶ pgAdmin is designed to monitor and manage multiple PostgreSQL and EDB Advanced Server database servers, both local and remote, through a single graphical interface that allows the easy creation and management of database objects, as well as a number of other tools for managing your databases.

PostgreSQL and pgAdmin

- Installation of PostgreSQL and pgAdmin
 - ▶ **Download PostgreSQL 14.3** <https://www.postgresql.org/download/> (Windows x86-64 only; Windows x86-32 is not supported)
 - ▶ **Installation** <https://www.youtube.com/watch?v=fZQI7nBu32M&t=9s>
 - ★ Select all components in Setup so we do not need to install pgAdmin separately
 - ★ We deselect the Stack Builder option following the video



- Do not forget your passwords for PostgreSQL and pgAdmin!

PostgreSQL and pgAdmin

- Create a database following the above video
 - ▶ It is an empty database
 - ▶ You name it

License



This work is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-nc-sa/4.0/).