# What is a database?

- → Storing large amounts of data in a structured way
- → Allows for dynamic queries for data

- → It used to be about storing:
  - → Corporate data
    - → Payrolls, inventory, sales, customers, accounting, documents, …
    - → Banking Systems
    - → Stock Exchanges
    - → Airline Systems
    - → Etc.

- → Today, databases are used in all fields:
  - → Web backends:
    - → Web search (google, bing, etc.)
    - → Social Networks (Facebook, Instagram, etc.)
    - → Blogs, Forums, etc.
  - → Mobile applications
  - → Data Warehouses
  - → Big Data
  - → AI
  - → Etc.



# Why use a database?

→ Easy to use

→ Flexible search

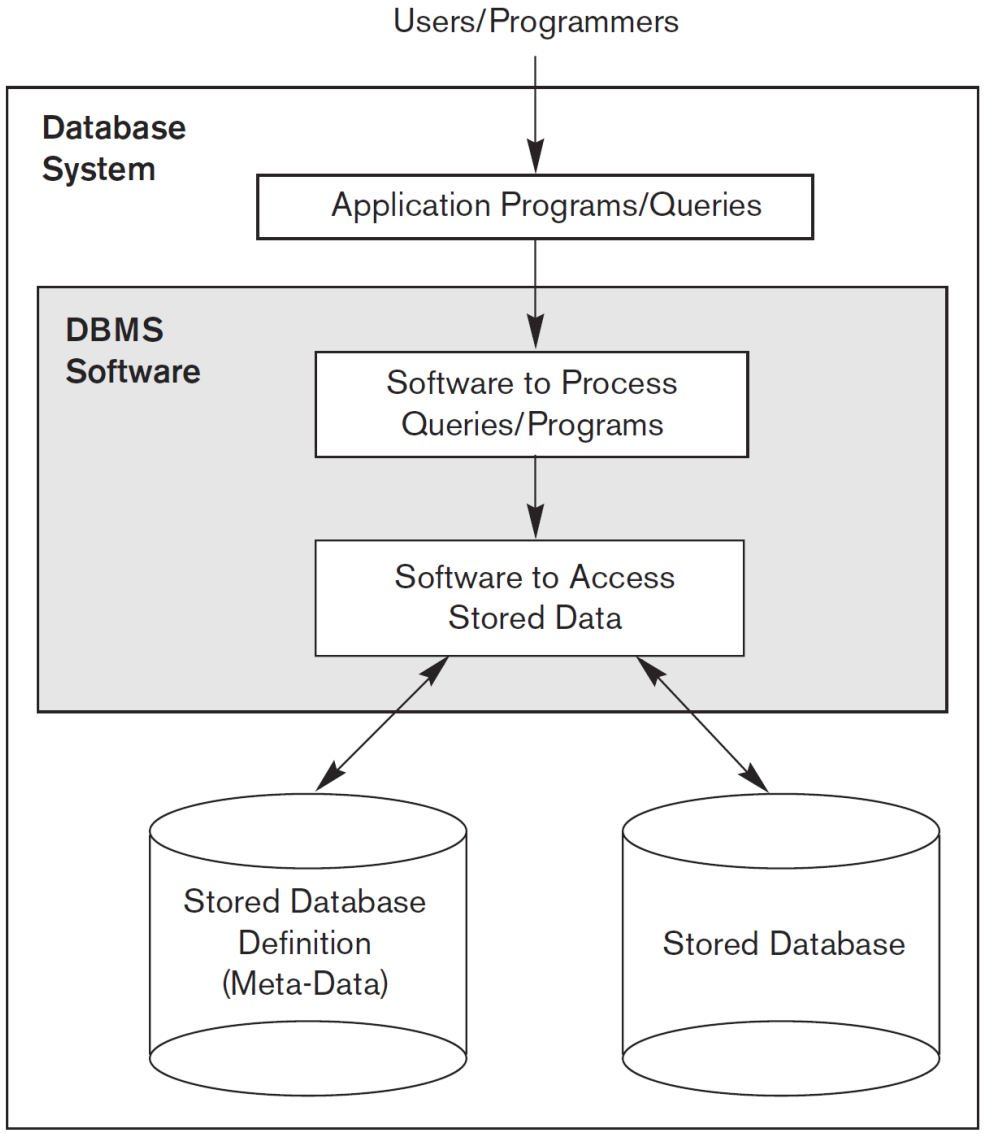
→ Efficient

→ Centralized storage

→ Multi-user access

→ Scalability

→ Security and consistency

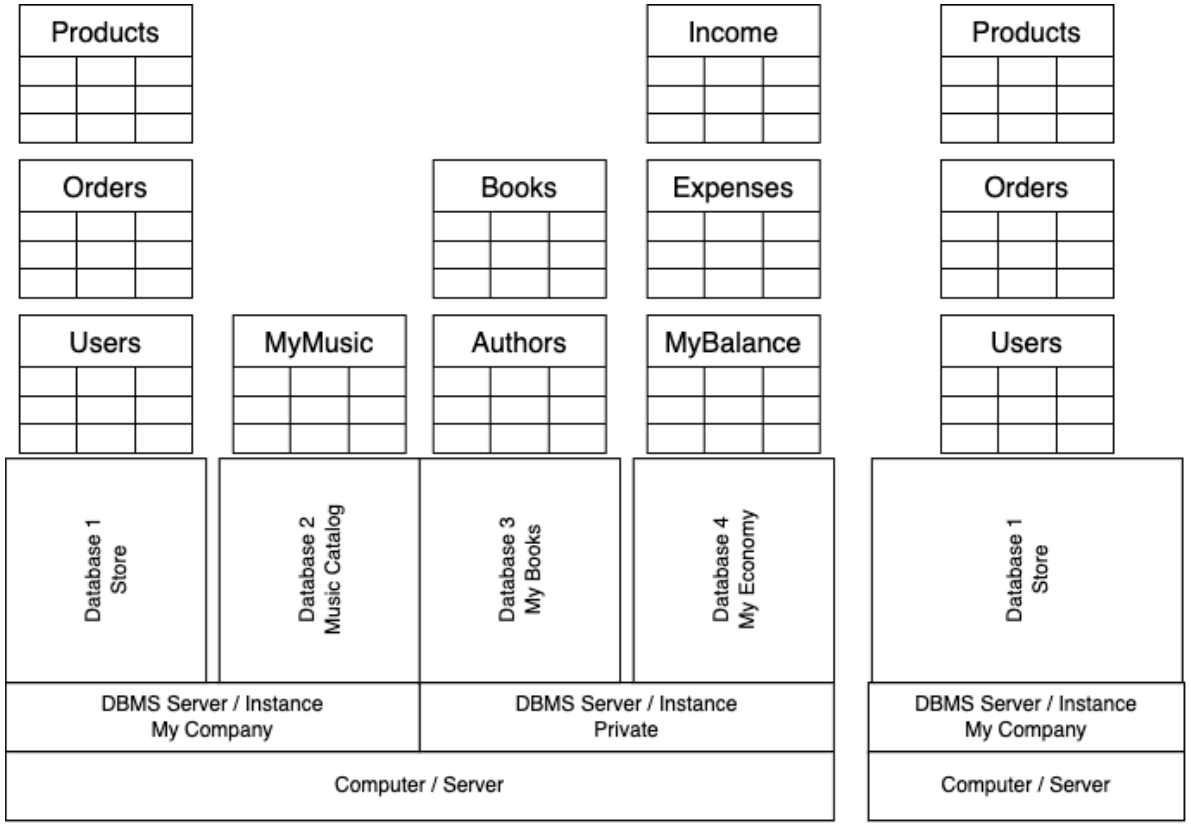



# Definitions 1/4

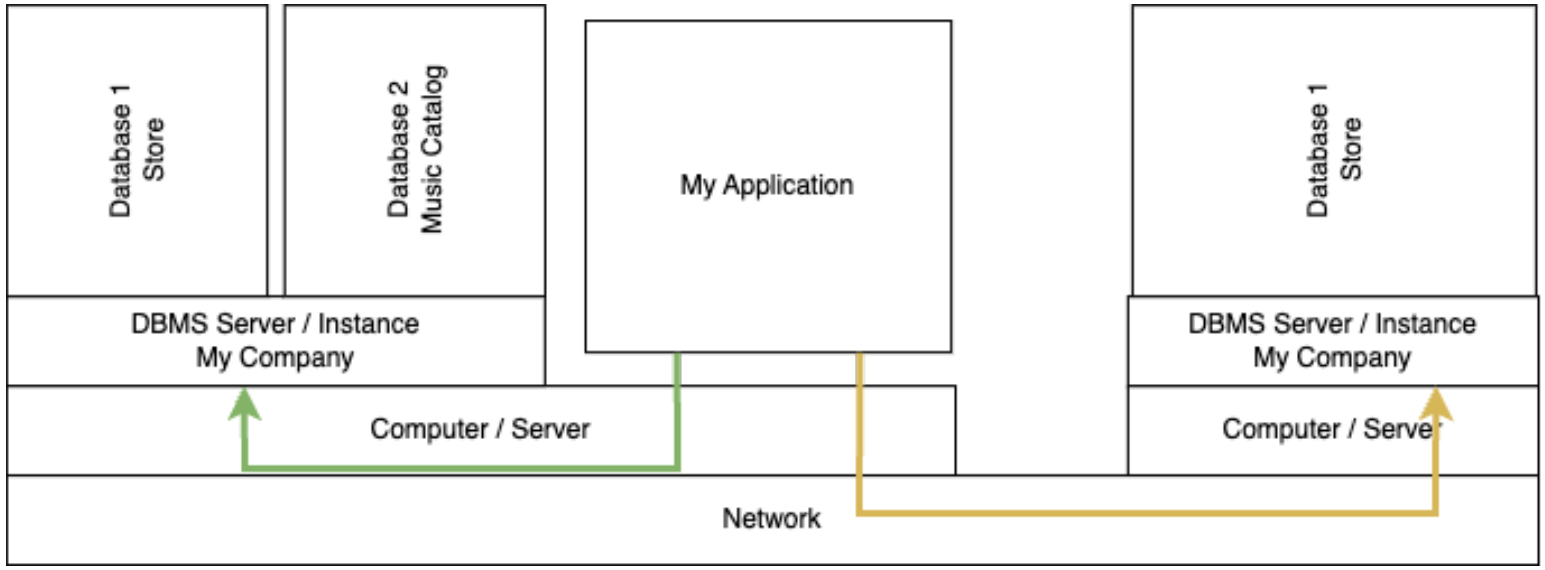
- → Database Management System (DBMS) – A system that enables users to create and maintain a database.
- → Database – A collection of related data. A collection of random data is not a database. Often the word database also, incorrectly, refers to a DBMS.
- → Data – Known facts that can be recorded that has implicit meaning.
- → Mini-World / Universe of Discourse – a database that represents some aspect of the real world. (Mostly here as the book refers to it)
- → Meta data – Data about data, or data that gives context to other data.
- → Schema – A definition of table structures and their relationships.
- → SQL – Structured Query Language – A language to extract data from a database.

Source: Fundamentals of DB Systems, by Elmasri and Navathe

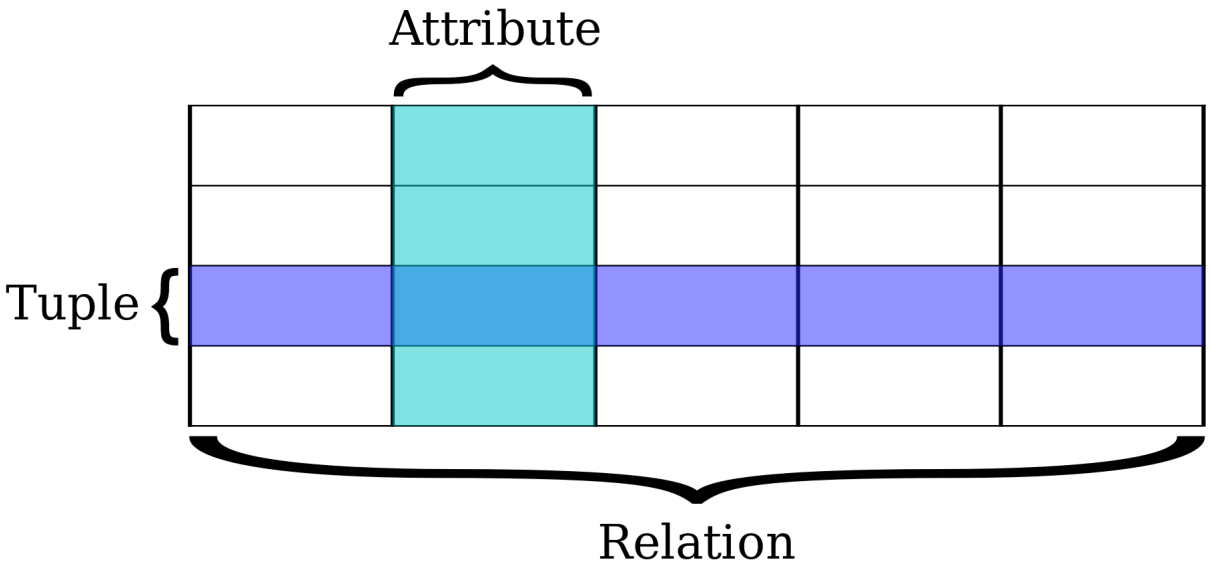
# Definitions 2/4



# Definitions 3/4



# Definitions 3/4




- → Table – A collection of rows and columns that contains Cells.
- → Row, Tuple, or Record – A collection of cells that together form a set of data that has a concrete Relation to each other. Shown as the horizontal line to the left.
- → Column, Attribute, or Field – A collection of Cells that that are all the same type. They are a part of multiple Rows.
- → Cell – A specific Rows Column. Ergo the intersection.
- → Relation – The relationship between data in a Row.
- → Value – The information saved in the specific Cell.
- → View / Result Set – The table that is created in memory and sent to the client as a result of a query.

# Relational Databases

- → Consists of multiple tables
- → Tables relate to each other
- → Uses primary and foreign keys to enforce the relationship constraints

| Rank | | | DBMS | Database Model | Score | | |
|---|---|---|---|---|---|---|---|
| Feb 2020 | Jan 2020 | Feb 2019 | | | Feb 2020 | Jan 2020 | Feb 2019 |
| 1. | 1. | 1. | Oracle | Relational, Multi-model | 1344.75 | -1.93 | +80.73 |
| 2. | 2. | 2. | MySQL | Relational, Multi-model | 1267.65 | -7.00 | +100.36 |
| 3. | 3. | 3. | Microsoft SQL Server | Relational, Multi-model | 1093.75 | -4.80 | +53.69 |
| 4. | 4. | 4. | PostgreSQL | Relational, Multi-model | 506.94 | -0.25 | +33.38 |
| 5. | 5. | 5. | IBM Db2 | Relational, Multi-model | 165.55 | -3.15 | -13.87 |
| 6. | 6. | 6. | Microsoft Access | Relational | 128.06 | -0.52 | -15.96 |
| 7. | 7. | 7. | SQLite | Relational | 123.36 | +1.22 | -2.81 |
| 8. | 8. | 8. | MariaDB | Relational, Multi-model | 87.34 | -0.11 | +3.91 |
| 9. | 9. | ↑ 10. | Hive | Relational | 83.53 | -0.71 | +11.25 |
| 10. | 10. | ↓ 9. | Teradata | Relational, Multi-model | 76.81 | -1.48 | +0.84 |
| 11. | ↑ 12. | ↑ 12. | SAP HANA | Relational, Multi-model | 54.97 | +0.28 | -1.58 |
| 12. | ↓ 11. | ↓ 11. | FileMaker | Relational | 54.88 | -0.23 | -2.91 |
| 13. | 13. | 13. | SAP Adaptive Server | Relational | 52.73 | -1.86 | -3.02 |
| 14. | 14. | 14. | Microsoft Azure SQL Database | Relational, Multi-model | 31.41 | +3.20 | +4.28 |
| 15. | 15. | ↑ 20. | Google BigQuery | Relational | 27.56 | +0.81 | +8.81 |

| Data Model | Example Databases |
|---|---|
| Key-Value ("Key-Value Databases," p. 81) | BerkeleyDB |
| | LevelDB |
| | Memcached |
| | Project Voldemort |
| | Redis |
| | *Riak* |
| Document ("Document Databases," p. 89) | CouchDB |
| | *MongoDB* |
| | OrientDB |
| | RavenDB |
| | Terrastore |
| Column-Family ("Column-Family Stores," p. 99) | Amazon SimpleDB |
| | *Cassandra* |
| | HBase |
| | Hypertable |
| Graph ("Graph Databases," p. 111) | FlockDB |
| | HyperGraphDB |
| | Infinite Graph |
| | *Neo4J* |
| | OrientDB |

Source: NoSQL Distilled, by P. Sadalage and M. Fowler





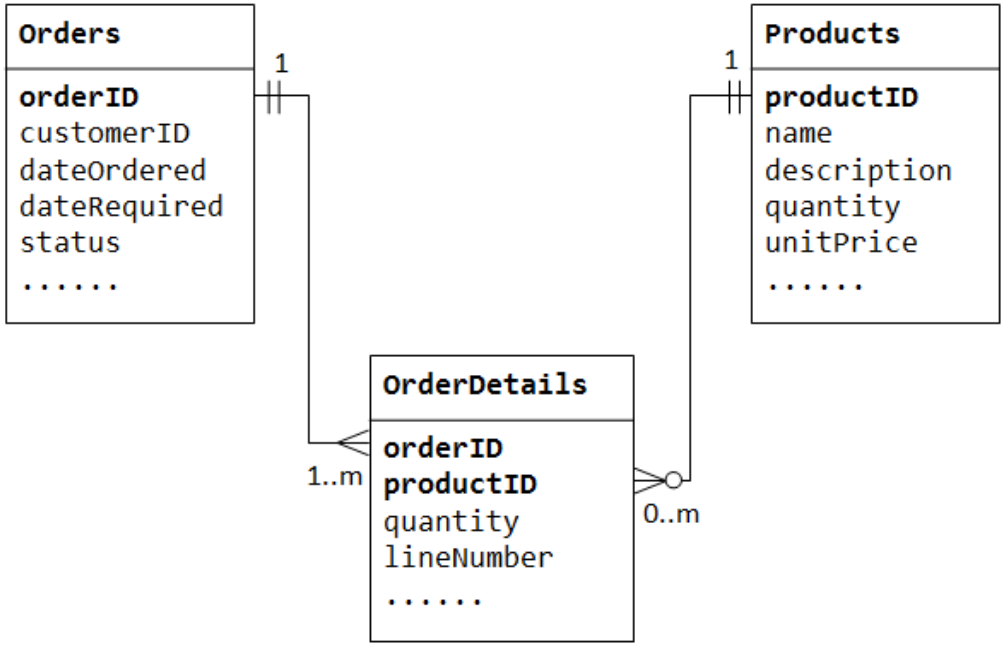
# NoSQL Databases



→ NoSQL means **N**ot **o**nly **SQL**

→ NoSQL covers multiple types of databases

# What is a relational database?




- → Has a Schema that defines the Tables
- → Uses multiple tables to store data
- → Uses the notion of Primary Keys and Foreign keys to relate table content to each other.
- → Uses SQL which makes CRUD (Create, Read, Update, Delete) operations on Schemas, Tables, and Rows.


# PostgreSQL

- → This course will use PostgreSQL (AKA Postgres)
- → However, this is **NOT** a PostgreSQL course!

- → PostgreSQL is:
  - → A Relational Database
  - → Cross-Platform (Windows, Mac OS, Linux, BSD, Solaris)
  - → Licensed under the PostgreSQL License, a liberal Open Source license, similar to the BSD or MIT licenses.
  - → Open Source (https://github.com/postgres/postgres)
- → Documentation and usage sites:
  - → https://www.postgresql.org/docs/12/index.html
  - → https://www.postgresqltutorial.com/

```sql
-- creating the initial accounts table
CREATE TABLE account(
   user_id serial PRIMARY KEY,
   username VARCHAR (50) UNIQUE NOT NULL,
   password VARCHAR (50) NOT NULL,
   email VARCHAR (355) UNIQUE NOT NULL,
   created_on TIMESTAMP NOT NULL,
   last_login TIMESTAMP
);

-- inserting two users
INSERT INTO account (username, password, email, created_on)
VALUES ('John', 'myPassW0rd', 'john@acme.com', NOW());

INSERT INTO account (username, password, email, created_on)
VALUES ('Anne', 'myPassW0rd', 'anne@acme.com', NOW());

-- querying for all rows in the account table
SELECT * FROM account;

-- updating the password of the user Anne
UPDATE account SET password = 'newPassW0rd' WHERE username = 'Anne';
```

# What is SQL?

→ Acronym for: **S**tructured **Q**uery **L**anguage.

→ Pronounced either Sequel, or SQL.

→ Allows for retrieval of data from a database.

→ A query results in a result set, which is structured as a table with rows and columns.

# Tables and relationships

**STUDENT**

| Name | Student_number | Class | Major |
|---|---|---|---|
| Smith | 17 | 1 | CS |
| Brown | 8 | 2 | CS |

**COURSE**

| Course_name | Course_number | Credit_hours | Department |
|---|---|---|---|
| Intro to Computer Science | CS1310 | 4 | CS |
| Data Structures | CS3320 | 4 | CS |
| Discrete Mathematics | MATH2410 | 3 | MATH |
| Database | CS3380 | 3 | CS |

**PREREQUISITE**

| Course_number | Prerequisite_number |
|---|---|
| CS3380 | CS3320 |
| CS3380 | MATH2410 |
| CS3320 | CS1310 |

**SECTION**

| Section_identifier | Course_number | Semester | Year | Instructor |
|---|---|---|---|---|
| 85 | MATH2410 | Fall | 07 | King |
| 92 | CS1310 | Fall | 07 | Anderson |
| 102 | CS3320 | Spring | 08 | Knuth |
| 112 | MATH2410 | Fall | 08 | Chang |
| 119 | CS1310 | Fall | 08 | Anderson |
| 135 | CS3380 | Fall | 08 | Stone |

**GRADE_REPORT**

| Student_number | Section_identifier | Grade |
|---|---|---|
| 17 | 112 | B |
| 17 | 119 | C |
| 8 | 85 | A |
| 8 | 92 | A |
| 8 | 102 | B |
| 8 | 135 | A |

# Creating a Table

```
CREATE TABLE account(
    user_id serial PRIMARY KEY,
    username VARCHAR (50) UNIQUE NOT NULL,
    password VARCHAR (50) NOT NULL,
    email VARCHAR (355) UNIQUE NOT NULL,
    created_on TIMESTAMP NOT NULL,
    last_login TIMESTAMP
);
```

| user_id<br>[PK] integer | username<br>character varying (50) | password<br>character varying (50) | email<br>character varying (355) | created_on<br>timestamp without time zone | last_login<br>timestamp without time zone |
|---|---|---|---|---|---|
| | | | | | |

# Inserts (CRUD)

```
INSERT INTO account (username, password, email, created_on)
VALUES ('John', 'myPassW0rd', 'john@acme.com', NOW());

INSERT INTO account (username, password, email, created_on)
VALUES ('Anne', 'myPassW0rd', 'anne@acme.com', NOW());
```

| | user_id [PK] integer | username character varying (50) | password character varying (50) | email character varying (355) | created_on timestamp without time zone | last_login timestamp without time zone |
|---|---|---|---|---|---|---|
| 1 | 4 | John | myPassW0rd | john@acme.com | 2020-02-06 11:43:44.158522 | [null] |
| 2 | 5 | Anne | myPassW0rd | anne@acme.com | 2020-02-06 11:43:44.158522 | [null] |

# Selects (CRUD)

```sql
SELECT * FROM account;

SELECT username, created_on FROM account WHERE email = 'john@acme.com';

SELECT username, created_on FROM account WHERE email LIKE '%anne%';
```

| | user_id [PK] integer | username character varying (50) | password character varying (50) | email character varying (355) | created_on timestamp without time zone | last_login timestamp without time zone |
|---|---|---|---|---|---|---|
| 1 | 1 | John | myPassW0rd | john@acme.com | 2020-02-06 11:41:11.729203 | [null] |

# **Updates** (CR**U**D)

```
UPDATE account SET password = 'newPassW0rd' WHERE username = 'Anne';
```

| | user_id [PK] integer | username character varying (50) | password character varying (50) | email character varying (355) | created_on timestamp without time zone | last_login timestamp without time zone |
|---|---|---|---|---|---|---|
| 1 | 1 | John | myPassW0rd | john@acme.com | 2020-02-06 11:41:11.729203 | [null] |
| 2 | 2 | Anne | newPassW0rd | anne@acme.com | 2020-02-06 11:42:13.27506 | [null] |

# **Delete** (CRU**D**)

```
DELETE FROM account WHERE email = 'john@acme.com';
```

| | user_id [PK] integer | username character varying (50) | password character varying (50) | email character varying (355) | created_on timestamp without time zone | last_login timestamp without time zone |
|---|---|---|---|---|---|---|
| 1 | 5 | Anne | myPassW0rd | anne@acme.com | 2020-02-06 11:43:44.158522 | [null] |

BEWARE of doing this: `DELETE FROM account;`

# Constraints and Data types in SQL

```sql
CREATE TABLE account(
   user_id serial PRIMARY KEY,
   username VARCHAR (50) UNIQUE NOT NULL,
   password VARCHAR (50) NOT NULL,
   email VARCHAR (355) UNIQUE NOT NULL,
   created_on TIMESTAMP NOT NULL,
   last_login TIMESTAMP
);
```

# Constraints – Abbreviated, more later.

→ PRIMARY KEY – The unique identifier for the current row, which is always NOT NULL

→ FOREIGN KEY – A key that refers to a primary key in a different table, signifying their relationship to each other

→ UNIQUE – Two cells in this Column cannot be the same

→ NOT NULL – This attribute HAS to be specified

# PostgreSQL Data Types 1/3

- → Null – Value Missing
- → Boolean – 1 bit
  - → Converts Boolean values e.g., 1, yes, y, t, true are converted to true, and 0, no, n false, f are converted to false.
- → Character types
  - → CHAR(n) – Fixed length text. Unused space is padded with space characters.
  - → VARCHAR(n) – Variable length string, as in “store up to”.
  - → TEXT – Large size text lengths such as book texts.

# PostgreSQL Data Types 2/3

- → Numeric types
  - → Integer
    - → SMALLINT – 2 byte, ranges from -32.768 to 32.767 (2 byte = 2 x 8 bit = 256 x 256 = 65.536)
    - → INT – 4 byte integer, rage from -2,147,483,648 to 2,147,483,647
    - → SERIAL – Same as INT, but used for auto incrementing.
  - → Floating-point
    - → FLOAT(n) – floating point number with at the precision of n, and a maximum of 8 bytes. (n as in numbers after the “,”)
    - → REAL – A floating point number. 0.001, 0.00000001, etc.
- → Temporal types
  - → DATE – dates only
  - → TIME – time of day values
  - → TIMESTAMP – both date and time values
  - → INTERVAL – periods of time

# PostgreSQL Data Types 3/3

→ UUID for storing Universally Unique Identifiers

→ Array for storing array strings, numbers, etc.

→ JSON stores JSON data

→ hstore stores key-value pair

→ Special types

  → box, line, point, lseg, polygon, inet, macaddr.

→ See more here: https://www.postgresql.org/docs/12/datatype.html