INTRODUCTION TO DATABASES

BACKEND-WORKSHOP

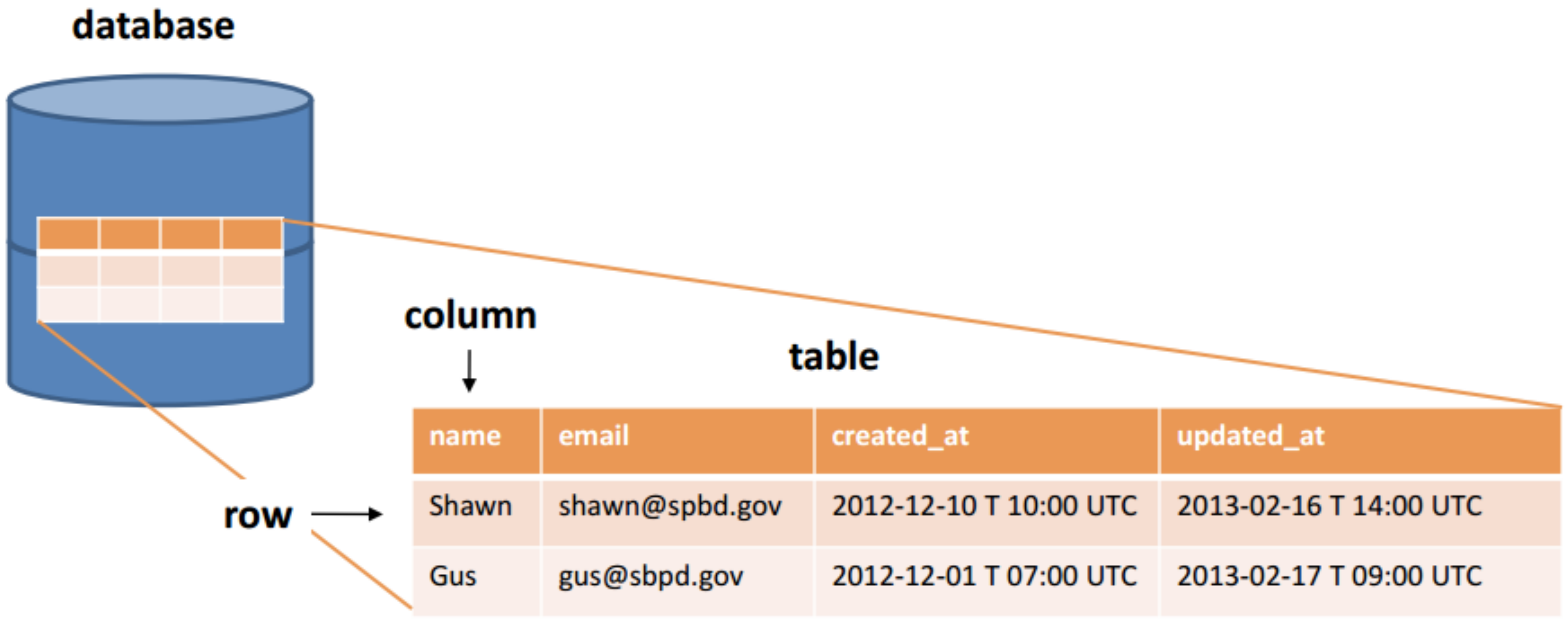
Michael Fröhlich - michael-froehlich@cdtm.de

Tobias Dümmling - tobias.duemmling@cdtm.de

WHAT IS A DATABASE?

WHAT IS A DATABASE?

- A database stores data in an organized way so that it can be searched and retrieved later. It should contain one or more tables.
- Tables consists of columns (attributes) and rows (records).

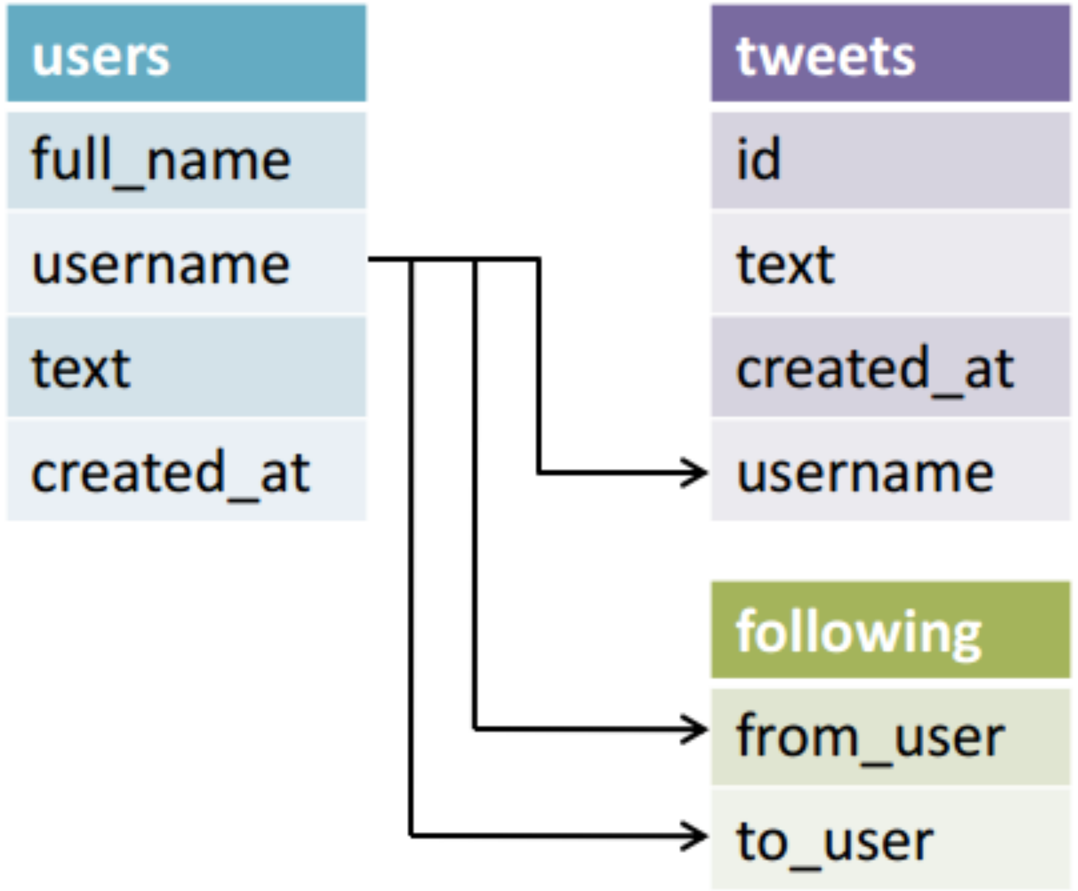


name	email	created_at	updated_at
Shawn	shawn@spbd.gov	2012-12-10 T 10:00 UTC	2013-02-16 T 14:00 UTC
Gus	gus@sbpd.gov	2012-12-01 T 07:00 UTC	2013-02-17 T 09:00 UTC

Relational Databases for Dummies: https://code.tutsplus.com/tutorials/relational-databases-for-dummies--net-30244

WHY NOT USE EXCEL?

- Easy manipulation of data
- **Relational databases** allow to define 'relations' between tables
  - System of Tables
  - Minimum Redundancy
  - Referential Integrity
  - Database
  - Keys
- Relational Database Management Systems (RDMBS) comply to the **ACID** model



users
full_name
username
text
created_at

tweets
id
text
created_at
username

following
from_user
to_user

ACID COMPLIANCE

- **Atomicity:** The database transaction must completely succeed or completely fail. Partial success is not allowed.

- **Consistency:** During the database transaction, the RDBMS progresses from one valid state to another. The state is never invalid.

- **Isolation:** The client’s database transaction must occur in isolation from other clients attempting to transact with the RDBMS.

- **Durability:** The data operation that was part of the transaction must be reflected in *nonvolatile storage* (computer memory that can retrieve stored information even when not powered – like a hard disk) and persist after the transaction successfully completes. Transaction failures cannot leave the data in a partially committed state.

SQL

Structured Query Language

- Programming language to manage data in RDBMS
- Originally based on **Relational Algebra**

- **Scope of SQL**: Insert, Query, Update & Delete Data
  - Data Definition Language
  - Data Manipulation Language
  - Data Control Language

SQLite Tutorial: https://www.tutorialspoint.com/sqlite/

DID ANYONE MAKE FIRST EXPERIENCES WITH SQL?

CREATE A TABLE – THE CREATE COMMAND

- Creates a new table within the database
  - <AttrName>   Attribute name
  - <Type>   Type of Data which is stored
  - <Flags>   Zero to * Flags which define additional rules

```
CREATE TABLE USER(
   <AttrName> <Type> <Flags>,
     … ,
   <AttrName> <Type> <Flags>
);
```

```
CREATE TABLE Users (
   id        INTEGER    PRIMARY KEY AUTOINCREMENT,
   email     TEXT        NOT NULL UNIQUE.
   pw        TEXT       NOT NULL,
   active    INTEGER    NOT NULL DEFAULT 0
);
```

IMPORTANT DATA TYPES

INTEGER	Integer numeric value
TEXT	Storing text data
TIMESTAMP	Stores year, month, day, hour, minute, and second values
FLOAT	Numerical rounded after 16 digits

IMPORTANT FLAGS

NOT NULL	The value has to exist once you store something in the table
DEFAULT <VAL>	Sets the default value <VAL> if the value is not set
CURRENT_TIMESTAMP	Returns the current time
PRIMARY KEY	Sets the value to be the unique identifier of the entry
AUTOINCREMENT	Automatically increments this value once you insert new data
ON DELETE CASCADE	Automatically deletes the data in the referred table once the data in this table is deleted
PRIMARY KEY()	Sets a value to be primary key. The primary key can consist of more than one value.
FOREIGN KEY TABLE(PRIMARY KEY)	Refers to a primary key within another table

INSERT VALUES INTO AN EXISTING TABLE

- Inserts a new record into an existing table

<TableName>	Table name
<AttrName>	Attribute name
<AttrValue>	Concret value to be inserted

```
INSERT INTO TABLE <TableName>
    (<AttrName1>, <AttrName2>)
VALUES
    (<AttrValue1>, <AttrValue2>);
```

```
INSERT INTO TABLE Users
    (email, pw)
VALUES
    (“Tobi“, “secretPassword“);
```

ACCESS THE VALUES IN A TABLE

- Returns a list of records (rows) with the specified attributes
  - <AttrName> Attribute name (use * to get all attributed of a table)
  - <TableName> Table name
  - <Conditions> Specify which conditions have to be met
  - <Direction> ASC (ascending) or DESC (descending)

```
SELECT  <AttrName1>, <AttrName2>
FROM <TableName>
WHERE <Conditions>
ORDER BY <AttrName> <Direction>
```

```
SELECT  id, email, pw
FROM Users
WHERE email = “tobias.duemmling@cdtm.de“
     OR email = “michael.froehlich@cdtm.de“
ORDER BY id DESC
```

UPDATE AN EXISTING ENTRY

- Updates an existing record
  - <AttrName> Attribute name (use * to get all attributed of a table)
  - <TableName> Table name
  - <Conditions> Specify which conditions have to be met
    IMPORTANT: If you do not specify a condition all records are updated

```
UPDATE <TableName>
SET <AttrName1> = <AttrValue1> ,
    <AttrName2>  = <AttrValue2>
WHERE <Conditions>
```

```
UPDATE Users
SET pw= “ILOVEBACKENDS“
WHERE email = “tobias.duemmling@cdtm.de“
```

DELETE AN ENTRY FROM YOUR DATABASE

- Deletes an existing record
  - <TableName> Table name
  - <Conditions> Specify which conditions have to be met
    IMPORTANT: If you do not specify a condition all records are updated

```
DELETE FROM <TableName>
WHERE <Conditions>
```

```
DELETE FROM Users
WHERE email = “tobias.duemmling@cdtm.de“
```

CODING CHALLENGE

- Create an sql script which creates our task database with the following tables
    - Lists
    - Tasks
- Write functions to perform CRUD operations on the Tasks table in Python and adapt your existing routes so that all tasks are saved in the database.

NOW THAT YOU HAVE THAT FEATURE DONE
I WANT IT TO DO SOMETHING ELSE INSTEAD