Databases

For projects with databases it is important to provide a db/migrations folder including all the steps used to setup the database require for the program to work. Instructions for the database manager should also be included in the installation part of the readme.

PostgreSQL

* createdb - on cmd to create base database in username
* psql - to enter sql shell
* \q - to quit shell
* \l - to list all databases
* \c <database-name> - to connect to databse

Once connecting to database, to insert data:

* INSERT INTO <table-name>(<column1>, <column2>, …)

VALUES

(<value1>, <value2>,..),

(<value1>, <value2>,..);

Delete data:

* DELETE FROM <table> WHERE <column> = <val>;

Update data:

* UPDATE <table> SET <column> = <val> WHERE <column> = <val>;

Selecting Data: SELECT <columns> FROM "<table>" WHERE …;

* WHERE <column> IN (‘<value>’); - Select data which matches value in column
* WHERE <column> BETWEEN <value> AND <value>; - data between values
* WHERE <column> > <value>; - mathematical operators can be used in conditions
* WHERE <column> LIKE(‘value%’); - Looks for values inc value at beginning
* WHERE <column> NOT LIKE(‘%a%’); - Looks for values which don’t include ‘a’

Conditions can be linked with AND and OR. XOR allows for exclusive or, so value must be one or other but not both. XOR will separate conditions so multiple conditions can be used eitherside without brackets.

Notes:

* NOT can be put before LIKE and IN.
* Quotes can be escaped using standard notation \’

Editing results:

* columns can be modied by mathematical operators to change their output on selection, for example dividing one column by another:
  + SELECT name, gdp/population FROM world WHERE population > 200000000;
* Results can be ordered using the ORDER BY <column>, <column> command
  + SELECT winner FROM nobel WHERE winner LIKE(‘Sir%’) ORDER BY yr DESC, winner ASC; This will order year descending first then each year by winner name ascending.
* Since IN(<column>) returns 0 or 1 as a result for each row, it can be used to order items. Putting as an ORDER BY ASC will order by results not matching the IN clause first then results which match.

SQL Operations:

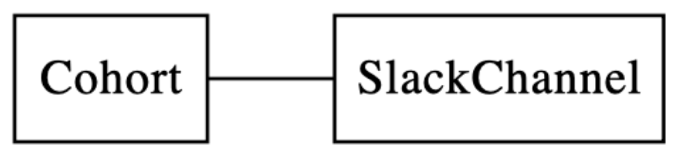
* = 🡸 equal too
* <> 🡸 not equal too
* General operations such as \* & /
* ROUND(‘<column>’, decimal place) 🡸 Decimal place can be negative
* LENGTH(‘<column>’) 🡸 returns length, can be used to compare or output
* LEFT(‘<column>’, num) 🡸 returns number of letters from left of column
* COUNT(<column>) 🡸 returns number of rows matched
* DISTINCT(<column) 🡸 returns only the unique values selected

<https://www.postgresqltutorial.com/>

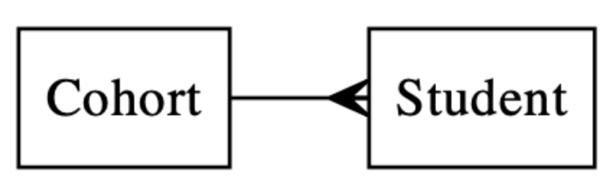
Entity Relationship Diagrams

To keep databases consistent it is important to diagram the relationships each entity will have with eachother. There is one item per row of a table each of which has a unique identifier key. When refering to items from one table to another, the item should be identified using the unique key, since any other value is liable to change.

One to one relationships - One cohort has one slack channel. One to one relationships mean one record in a table is associated with only one record in another table. Can be tabulated simply using the unique key of one in another table. Insert the forgein key of one into another.



One to many relationships - Cohort has many students. Can be tabulated by adding the unique key of the ‘one’ into the table for the ‘many’, this will be called the forgein key.



Many to many - Students are members of many slack channels and slack channels have many students. An issue here is if a student adds a slack channel a database would have to do two operations to add both the student to the slack channel and the channel to the student. To avoid this join tables are used where each unique key is related to eachother:



Connecting to PSQL Databse using Ruby

Using the ‘pg’ gem it is possible to conenct the a psql database directly from ruby code. Create a connection using the gem:

‘conn = PG::Connection.open(:dbname => ‘test’, :user => ENV['USER'], <:login, :password, etc…>)’

conn.copy\_data

How Databases Work

Each database has a Databse Management System (DBMS), like psql, which interacts and manages all stored data. There are four ways to interact with data:

CRUD - Create, Read, Update, Delete

There are two types of database:

* Relational - Tables made of columns and rows, with strict datatypes. Use the Structured Query Language (SQL). Patterns of data arrangment are called schema, and everything must be defined before any data can be added.
* Non-relational - NoSQL database which covers databases which aren’t strictly relational. Good for getting a database running quickly as no schema is required.

Most popular DBMS:

1. Oracle
2. MySQL
3. Microsoft SQL
4. PostgreSQL
5. mongoDB (NoSQL)