Review

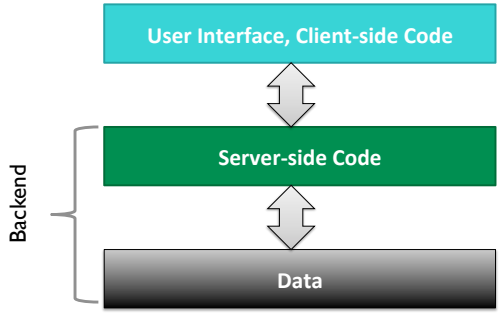
- What did you think of the talk?

More from Patrick

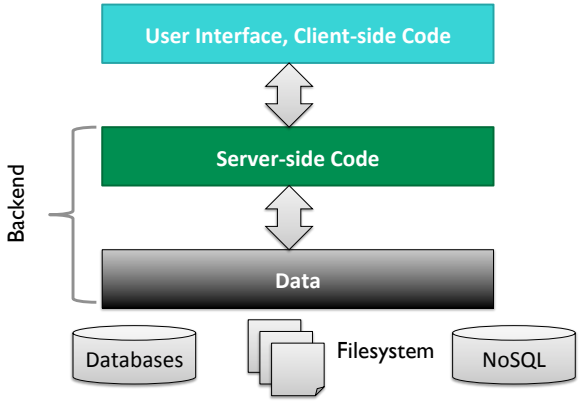
- Hubot is an open-source project
  - Can even tweet statuses
- 10s of thousands of tests
  - as many servers for CI as repositories
  - testing adds 20-30% to development
    - when impact will be smaller
    - 20 minute down -- really wish the CI would have caught it

FULL-STACK DEVELOPMENT: BACKENDS

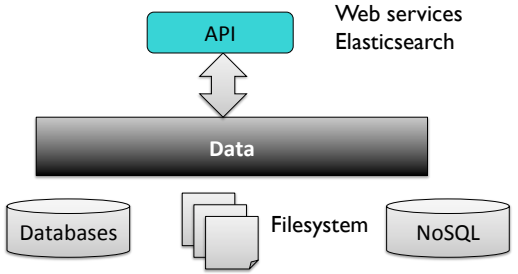
Web Software Architecture Overview



Web Software Architecture Overview



Data



RELATIONAL DATABASES

Database Overview

- Store data in such a way to allow *efficient* storage, search, and update
- **Relational Data Model** - currently most popular type of database
  - Different vendors: PostgreSQL, Oracle, MySQL, DB2, MSSQL
  - Data is stored in **tables**
  - ***Attributes***: column names (one word)
  - ***Entities***: rows in table
  - Often contain ***primary key***:
    a set of columns that uniquely identify a row

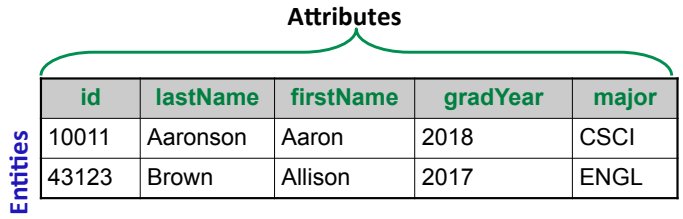
Example Students Table

- id is the primary key
- What are the attributes?
- What are the entities?

id	lastName	firstName	gradYear	major
10011	Aaronson	Aaron	2018	CSCI
43123	Brown	Allison	2017	ENGL

Example Students Table

- id is the primary key
- What are the attributes?
- What are the entities?



Courses Table

- Primary key is (Department, Number)
  - As a group, these uniquely identify a row

department	number	name	description
CSCI	101	Survey of Computer Science	A survey of …
CSCI	111	Fundamentals of Programming I	An introduction to …

SQL: STRUCTURED QUERY LANGUAGE

SQL: Structured Query Language

- Standardized language for manipulating and querying relational databases
  - May be slightly different depending on DB vendor
- Pronounced "S-Q-L" or "Sequel"

SQL: Structured Query Language

- Reserved words are not case-sensitive
  - I will tend to write them in all-caps and bold
  - Tables, column names - may be case sensitive
- Commands end in ;
  - Can have extra white space, new lines in commands
  - End when see ;
- Represent string literals with single quotes ''

SELECT Command

- Queries the database
- Returns a result—a ***virtual table***
- Syntax:

Optional

```
SELECT column_names
FROM table_names [WHERE condition];
```

  - Columns, tables separated by commas
  - Can select all columns with *
  - Where clause specifies constraints on what to select from the table

SELECT Examples

- `SELECT * FROM Students;`

id	lastName	firstName	gradYear	major
10011	Aaronson	Aaron	2018	CSCI
43123	Brown	Allison	2017	ENGL

- `SELECT lastName, major FROM Students;`

Virtual Tables

lastName	major
Aaronson	CSCI
Brown	ENGL

WHERE Conditions

- Limits which rows you get back
- Comparison operators: `>, >=, <, <=, <>`
- Can contain **AND** for compound conditions
- **LIKE** matches a string against a pattern
  - Wildcard: `%`, matches any sequence of 0 or more characters
- **IN** : match any
- **BETWEEN**: Like comparison using **AND**, inclusive

SELECT Examples

- What do these select statements mean?
  - `SELECT * FROM students WHERE major='CSCI';`
  - `SELECT firstName, lastName FROM students WHERE major='CSCI' AND gradYear=2017;`
  - `SELECT lastName FROM students WHERE firstName LIKE 'Eli%';`

SELECT Examples

- What do these select statements mean?
  - `SELECT lastName FROM students WHERE major IN ('CSCI', 'PHYS', 'MATH');`
  - `SELECT lastName FROM students WHERE major NOT IN ('CSCI', 'PHYS', 'MATH');`
  - `SELECT firstName FROM students WHERE gradYear BETWEEN 2017 AND 2019;`

Set vs Bag Semantics

Set vs Bag Semantics

- Bag
  - Duplicates allowed
  - Number of duplicates is significant
  - Used by SQL by default
- Set
  - No duplicates
  - Use keyword **DISTINCT**

Set vs Bag

```
SELECT lastName
FROM Students;
```

lastName
Smith
…
Smith
Jones
Jones

```
SELECT DISTINCT lastName
FROM Students;
```

lastName
Smith
Jones

Aggregates

- Standard SQL aggregate functions: **COUNT, SUM, AVG, MIN, MAX**
- Can only used in the **SELECT** part of query
- Example
  - `SELECT COUNT(*), AVG(GPA) FROM students WHERE gradYear=2017;`

ORDER BY

- Last operation performed, last in query
- Orders:
  - **ASC** = ascending
  - **DESC** = descending
- Example
  - `SELECT firstName, lastName FROM Students WHERE gradYear=2017 ORDER BY GPA DESC;`

Majors Table

- Another table to keep track of majors
- Primary Key: id

id	name	department
1	ART-BA	ART
2	ARTH-BA	ART

Changes Students Table

- Use an id to identify major (primary key)

Majors:

id	name	department
1	ART-BA	ART
2	ARTH-BA	ART

Foreign Key

Students:

id	last Name	first Name	gradYear	majorID
10011	Aaronson	Aaron	2018	123
43123	Brown	Allison	2017	157

JOIN Queries

- Join two tables on an attribute

Majors:

id	name	department
1	ART-BA	ART
2	ARTH-BA	ART

Students:

id	last Name	first Name	gradYear	majorID
10011	Aaronson	Aaron	2018	123
43123	Brown	Allison	2017	157

```
SELECT lastName, name
FROM Students, Majors
WHERE Students.majorID=Majors.id;
```


JOIN Queries: Creates a Cross-Product

- Join two tables on an attribute

Majors:

id	name	department
1	ART-BA	ART
2	ARTH-BA	ART

Students:

id	last Name	first Name	gradYear	majorID
10011	Aaronson	Aaron	2018	123
43123	Brown	Allison	2017	157

```
every entry in Majors
    x
        every entry in Studens
```

JOIN Queries

- Join two tables on an attribute

```
SELECT lastName, name
FROM Students, Majors
WHERE Students.majorID=Majors.id;
```

From **Students** From **Majors**

lastName	name
Aaronson	CSCI
Brown	ENGL

JOIN Queries

- What if two tables have the same column name?
  - Add the table name and a . to the beginning of the column, i.e., **TableName.columnName**

```
SELECT Students.lastName, Majors.name
FROM Students, Majors
WHERE Students.majorID=Majors.id;
```


What if Students Have Multiple Majors?

- We don't necessarily want to add another column to Students table
  - What if student has 3 majors?
- Example of Many to Many Relationship
- Solution: Create **StudentsToMajors** table:

studentID	majorID
435	243
435	232

Primary Key: (StudentID, MajorID)
Foreign Keys from Students, Majors Tables

Looking Ahead

- More on data