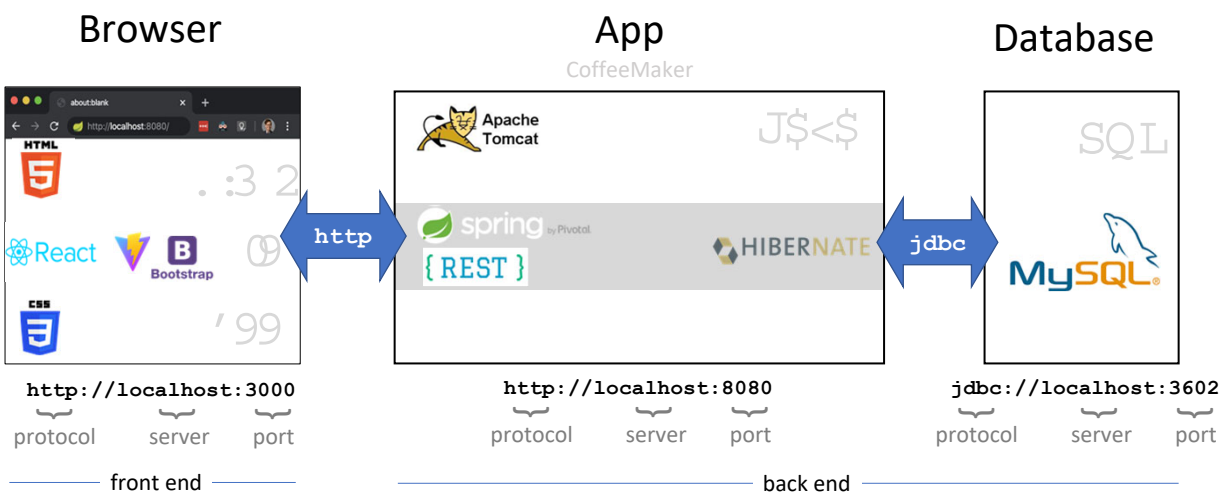
Backend + Persistence + Databases

CSC 326

1

High Level 326 App Architecture



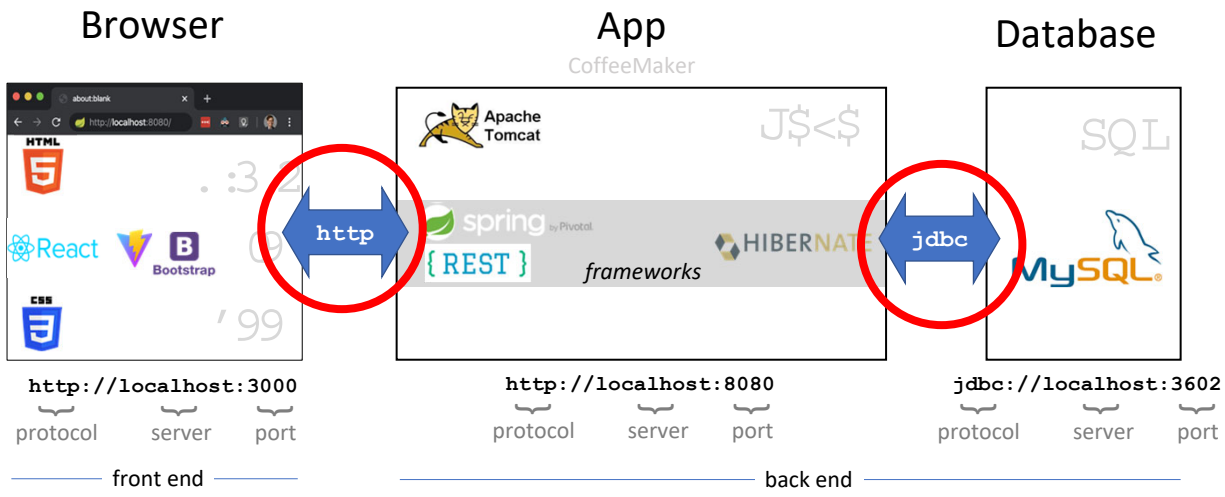
© NC State Software Engineering Faculty

2

2

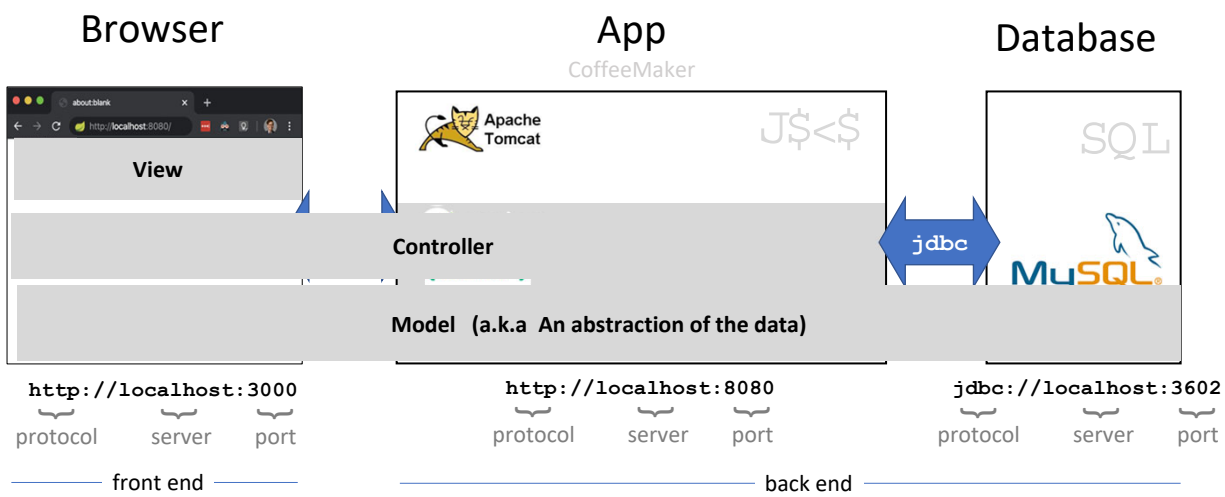
Key Pain Points

Transfer Representation of Data
1. Slow.
2. One change, three places.



© NC State Software Engineering Faculty

High Level 326 App Architecture



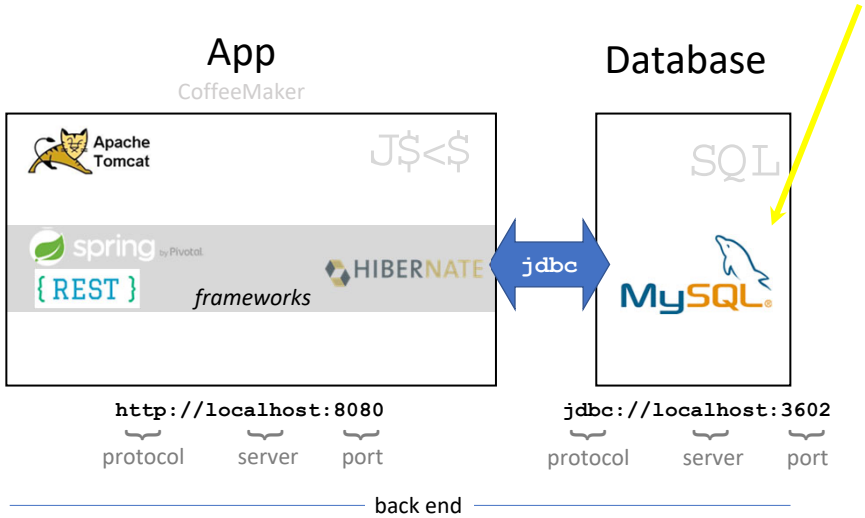
© NC State Software Engineering Faculty



Data Persistence

© NC State Software Engineering Faculty

CSC 326 App Architecture



© NC State Software Engineering Faculty

Exercise 05.01: Data Activity



- *What are different ways to store data?*
- *What are the advantages / disadvantages?*

think-pair-share （3mins）

© NC State Software Engineering Faculty

Storing Data



- Java objects just exist in memory
  - Close the program, everything disappears
- Often, we want to store data that persists when the program is closed
- In CSC 216, you used flat files (text, XML, CSV, markdown, or structured files)
  - Fine for simple programs
  - Doesn't scale well for larger programs
- Enter databases!
  - Versatile, scalable, high-performance way of storing data

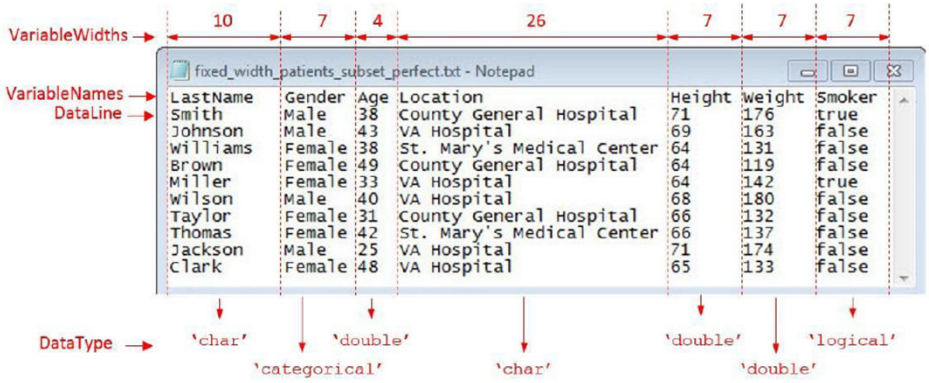
© NC State Software Engineering Faculty

"Flat Flies" – example: *Comma Separated (CSV)*

```
policyID,statecode,county,eq_site_limit,hu_site_limit,fl_site_limit,fr_site_limit,tiv_2011,tiv_2012,eq_site_deductible,hu_site_deductible,point_latitude,point_longitude,line,construction,point_granularity
119736,FL,CLAY COUNTY,498960,498960,498960,498960,498960,792148.9,0,9979.2,0,0,30.102261,-81.711777,Residential,Masonry,1
448094,FL,CLAY COUNTY,1322376.3,1322376.3,1322376.3,1322376.3,1322376.3,1438163.57,0,0,0,0,30.063936,-81.707664,Residential,Masonry,3
206893,FL,CLAY COUNTY,190724.4,190724.4,190724.4,190724.4,190724.4,192476.78,0,0,0,0,30.089579,-81.700455,Residential,Wood,1
333743,FL,CLAY COUNTY,0,79520.76,0,0,79520.76,86854.48,0,0,0,0,30.063236,-81.707703,Residential,Wood,3
172534,FL,CLAY COUNTY,0,254281.5,0,254281.5,254281.5,246144.49,0,0,0,0,30.060614,-81.702675,Residential,Wood,1
785275,FL,CLAY COUNTY,0,515035.62,0,0,515035.62,884419.17,0,0,0,0,30.063236,-81.707703,Residential,Masonry,3
995932,FL,CLAY COUNTY,0,0,19260000,0,0,19260000,20610000,0,0,0,0,30.102226,-81.713882,Commercial,Reinforced Concrete,1
223488,FL,CLAY COUNTY,328500,328500,328500,328500,328500,348374.25,0,16425,0,0,30.102217,-81.707146,Residential,Wood,1
433512,FL,CLAY COUNTY,315000,315000,315000,315000,315000,265821.57,0,15750,0,0,30.118774,-81.704613,Residential,Wood,1
142071,FL,CLAY COUNTY,705600,705600,705600,705600,705600,1010842.56,14112,35280,0,0,30.100628,-81.703751,Residential,Masonry,1
253816,FL,CLAY COUNTY,831498.3,831498.3,831498.3,831498.3,831498.3,1117791.48,0,0,0,0,30.10216,-81.719444,Residential,Masonry,1
894922,FL,CLAY COUNTY,0,24059.09,0,0,24059.09,33952.19,0,0,0,0,30.095957,-81.695099,Residential,Wood,1
422834,FL,CLAY COUNTY,0,48115.94,0,0,48115.94,66755.39,0,0,0,0,30.100073,-81.739822,Residential,Wood,1
582721,FL,CLAY COUNTY,0,28869.12,0,0,28869.12,42826.99,0,0,0,0,30.09248,-81.725167,Residential,Wood,1
842700,FL,CLAY COUNTY,0,56135.64,0,0,56135.64,50656.8,0,0,0,0,30.101356,-81.726248,Residential,Wood,1
874333,FL,CLAY COUNTY,0,48115.94,0,0,48115.94,67905.07,0,0,0,0,30.113743,-81.727463,Residential,Wood,1
580146,FL,CLAY COUNTY,0,48115.94,0,0,48115.94,66938.9,0,0,0,0,30.121655,-81.732391,Residential,Wood,3
456149,FL,CLAY COUNTY,0,80192.49,0,0,80192.49,86421.04,0,0,0,0,30.109537,-81.741661,Residential,Wood,1
767862,FL,CLAY COUNTY,0,48115.94,0,0,48115.94,73798.5,0,0,0,0,30.11824,-81.745335,Residential,Wood,3
353022,FL,CLAY COUNTY,0,60946.79,0,0,60946.79,62467.29,0,0,0,0,30.065799,-81.717416,Residential,Wood,1
367814,FL,CLAY COUNTY,0,28869.12,0,0,28869.12,42727.74,0,0,0,0,30.082993,-81.710581,Residential,Wood,1
671392,FL,CLAY COUNTY,0,13410000,0,0,13410000,11700000,0,0,0,0,30.091921,-81.711929,Commercial,Reinforced Concrete,3
772887,FL,CLAY COUNTY,0,1669113.93,0,0,1669113.93,2099127.76,0,0,0,0,30.117352,-81.711884,Residential,Masonry,1
983122,FL,CLAY COUNTY,0,179562.23,0,0,179562.23,211372.57,0,0,0,0,30.095783,-81.713181,Residential,Wood,3
934215,FL,CLAY COUNTY,0,177744.16,0,0,177744.16,157171.16,0,0,0,0,30.110518,-81.727478,Residential,Wood,1
385951,FL,CLAY COUNTY,0,17757.58,0,0,17757.58,16948.72,0,0,0,0,30.10288,-81.705719,Residential,Wood,1
716332,FL,CLAY COUNTY,0,130129.87,0,0,130129.87,101758.43,0,0,0,0,30.068468,-81.71624,Residential,Wood,1
751262,FL,CLAY COUNTY,0,42854.77,0,0,42854.77,63592.88,0,0,0,0,30.068468,-81.71624,Residential,Wood,1
633663,FL,CLAY COUNTY,0,785.58,0,0,785.58,662.18,0,0,0,0,30.068468,-81.71624,Residential,Wood,1
105851,FL,CLAY COUNTY,0,170361.91,0,0,170361.91,177176.38,0,0,0,0,30.068468,-81.71624,Residential,Wood,1
710400,FL,CLAY COUNTY,0,1430.89,0,0,1430.89,1861.41,0,0,0,0,30.068468,-81.71624,Residential,Wood,1
```

9

"Flat Files" – example: *Fixed-Width Data*



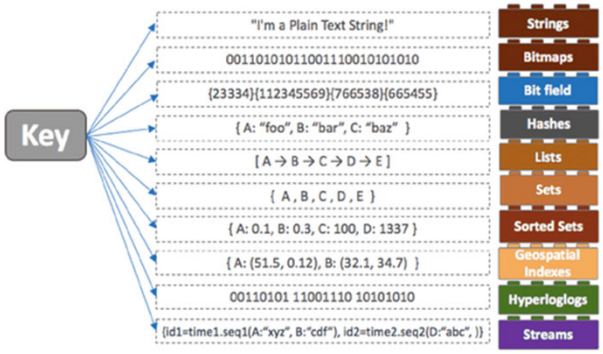
Simple. Inflexible. Imagine not being able to modify the width of columns. Still in use!

© NC State Software Engineering Faculty

10

10

Key Value - example




```
> set mykey somevalue
OK
> get mykey
"somevalue"
```

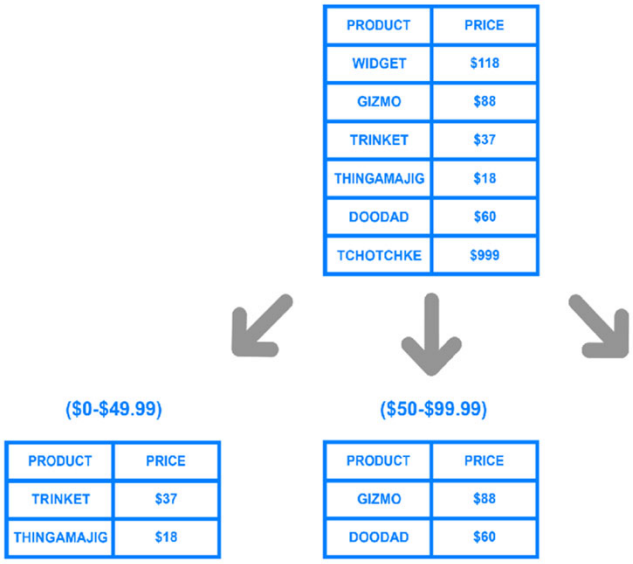
```
> mset a 10 b 20 c 30
OK
> mget a b c
1) "10"
2) "20"
3) "30"
```

+: More organized than flat files.
+: Can hold different data in each value store.
+: Might use much less memory since there are no placeholders for optional data.
+: Fast and flexible.

-: Lack of ability to perform ad hoc joins across tables.
-: Bad at sharding and concurrency.
-: No enforced data structure or model.

© NC State Software Engineering Faculty

Img: redis.com



SHARDING

For when your data is too big for a single physical datastore.

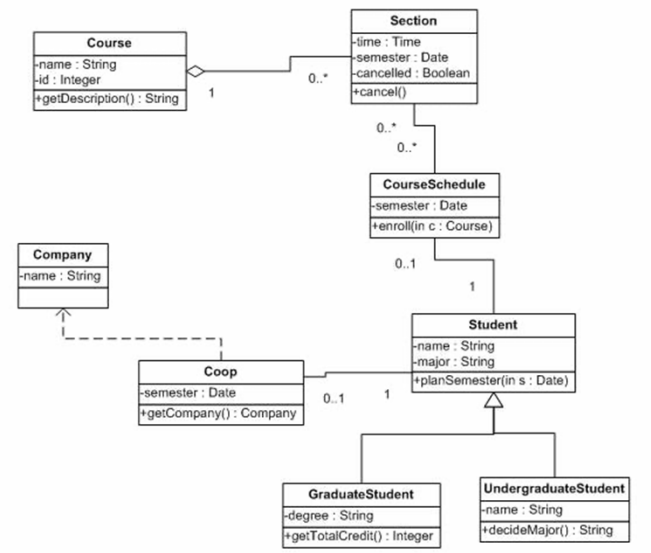
Requires structured data. Complicates management and maintenance.
Used in datacenters. The backbone of many world class data companies.

© NC State Software Engineering Faculty

Img: digitalocean.com

Relational Data

UML Class Diagram *(recall from 216)*



img: slideshare.net

© NC State Software Engineering Faculty

Relational Database

Simplest Form: *Kind of like a spreadsheet*

Table: "*recipes*"

		column					
	id	name	price	coffee	milk	sugar	chocolate
row	1	Coffee	50	2	1	1	0
	2	Mocha	75	2	1	1	1

© NC State Software Engineering Faculty

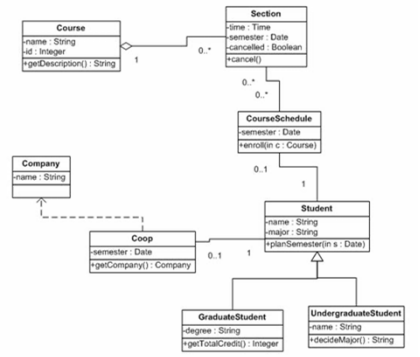


Remember this acronym! It's going to show up again.

recipe

id	name	price	coffee	milk	sugar	chocolate
1	Coffee	50	2	1	1	0
2	Mocha	75	2	1	1	1

- Relational database: manages data and [illegible] relations between data
- Relational Database Management System (RDBMS): system optimized for inserting, updating, selecting, and deleting data in relational database
- Components
  - Database: organizes information for a system
  - Table: organizes information about an "object"
  - Attribute/Column: a piece of information in a table – "fields" of an "object"
  - Tuple/Row: an entry in a table with values for each attribute – "instance" of an "object"
- Use Structured Query Language (SQL) to interact with data
- Want to learn more? CSC 440 / CSC 540 – Database Management Systems



© NC State Software Engineering Faculty

15

Considerations when choosing persistence

- How general is the representation?
- What is the maintenance cost?
- What performance considerations do I have?
- Does it scale?
- Is my data distributed?
- How well does it work with my existing development & testing toolchains?
- Is it compatible with my current environment?
- What is the speed to read? to write?

© NC State Software Engineering Faculty

16

Structured Query Language (SQL)

Based on slides developed by Dr. Kai Presler-Marshall

© NC State Software Engineering Faculty

17

Structured Query Language Overview

- A *declarative* language for working with databases
  - Declarative – describes logic without describing control flow
  - You say what you want, not how to get what you want
- SQL works with tables of information
  - Columns/fields describe what is stored in the tables
  - Rows/records/entities are specific instances of information in the table
- SQL is a standardized language, but there may be differences depending on the RDBMS that you're using – read the documentation!

© NC State Software Engineering Faculty

18

Syntax

- SQL statements end in semicolons
- Most RDBMS SQL defaults to *case insensitive*
  - But setting can be updated for case sensitivity
  - May vary by OS - be careful on Linux!
- Often SQL commands/keywords are written in ALL CAPS to separate them from table, column, and data information.
- Snake-case is often used for column names




```
SELECT my_field FROM my_table WHERE my_field = 42;
```

© NC State Software Engineering Faculty

Creating and Working with Databases

Create a Database

```
CREATE SCHEMA coffeemaker;
```

SCHEMA is the keyword for a database in SQL

Use the Database

```
USE coffeemaker;
```

When you start a RDBMS, you don't have a database selected for use. You have to switch to the database you want to use.
Think of it like changing directories to the correct database.

© NC State Software Engineering Faculty

Creating Tables

- Defines the columns and their types
- Name comes before the type
- There may be additional information about the columns
  - Auto-increment
  - Not null
- Can include information about a primary key
  - One or more columns that uniquely identify a row

```
1  CREATE TABLE recipes (
2    id BIGINT NOT NULL AUTO_INCREMENT,
3    name VARCHAR(20) NOT NULL,
4    price INT,
5    coffee INT,
6    milk INT,
7    chocolate INT,
8    PRIMARY KEY (id)
9  );
```

Result Grid | Filter Rows: | Edit:

	id	name	price	coffee	milk	sugar	chocolate
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

© NC State Software Engineering Faculty

21

Datatypes

- SQL supports a (relatively) wide range of datatypes
  - `INT`
  - `DOUBLE`
  - `BOOLEAN`
  - `CHAR/VARCHAR($size)`
    - This is equivalent to a String
    - String values wrapped in single quotes `'like this'`
    - Size provides a limit on the number of characters that a string can store
  - Lots more we won't see today
    - But if you're curious: https://www.w3schools.com/sql/sql_datatypes.asp

© NC State Software Engineering Faculty

22

Altering Tables

- Tables can be altered (updated), but this may lead to missing data or losing data. Be careful with updates!
  - Alter columns – add, delete or modify
  - Alter constraints – primary keys, null, size of things, etc.

ALTER TABLE recipes

ADD sugar INT AFTER milk;

© NC State Software Engineering Faculty

Populating Tables

Inserting rows into a table

```
INSERT INTO $tablename ($columns) VALUES ($values);
```

- If **`$columns`** is provided, number of **`$values`** MUST match
  - Example: don't include the auto-increment column
- If **`$columns`** is not provided, provide all **`$values`**

```
1  INSERT INTO recipes (name, price, coffee, milk, sugar, chocolate)
2  VALUES ('Coffee', 50, 1, 1, 1, 0);
```

Result Grid | Filter Rows: | Edit:

id	name	price	coffee	milk	sugar	chocolate
1	Coffee	50	1	1	1	0
NULL	NULL	NULL	NULL	NULL	NULL	NULL

© NC State Software Engineering Faculty

Retrieving Data

Retrieve rows from tables

```
SELECT $columns
FROM $tables
WHERE $conditions
```

- General syntax is to select some set of columns from a table that meets specific checks

© NC State Software Engineering Faculty

SELECT Examples

- Retrieve all the data from a table

```
SELECT * FROM recipes;
```

- Select the recipe names

```
SELECT name FROM recipes;
```

- Select the recipe name with a particular id

```
SELECT name FROM recipes WHERE id = 1;
```

- Select recipes with at least one unit of coffee and one unit of milk

```
SELECT name FROM recipes
WHERE coffee > 1 AND milk > 1;
```

© NC State Software Engineering Faculty

Additional WHERE Syntax

- Wildcard (*) is everything
- Comparison Operators
  - >, >=, <, <=, =, !=
  - Note that equality comparison is only *one equals!*
- Checks for null
  - `WHERE $column IS NULL`

© NC State Software Engineering Faculty

Additional SELECT Syntax

- Get only unique values
  - `SELECT DISTINCT $columns FROM $table`
- Sorting
  - `SELECT $columns FROM $table WHERE $conditions ORDER BY $columns`
- Aggregation
  - `SELECT $columns, COUNT($column) FROM $table WHERE $conditions GROUP BY $column`

© NC State Software Engineering Faculty

Exercise 05.02: SQL Practice

id	name	grade
1	Emerson Weaver	78
2	Fiona Moss	96
3	Hayden Sweeney	93
4	Mariam Cobb	81
5	Irma Harmon	88

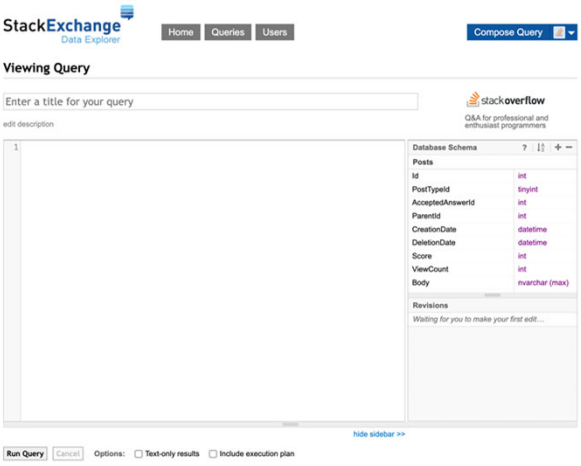
Given the **grades** table, create queries to do the following:

1. Select names of students who earned grades of at least 85
2. Order the information by grade.
3. Count the number of grades between 80 and 95

3 mins!

© NC State Software Engineering Faculty

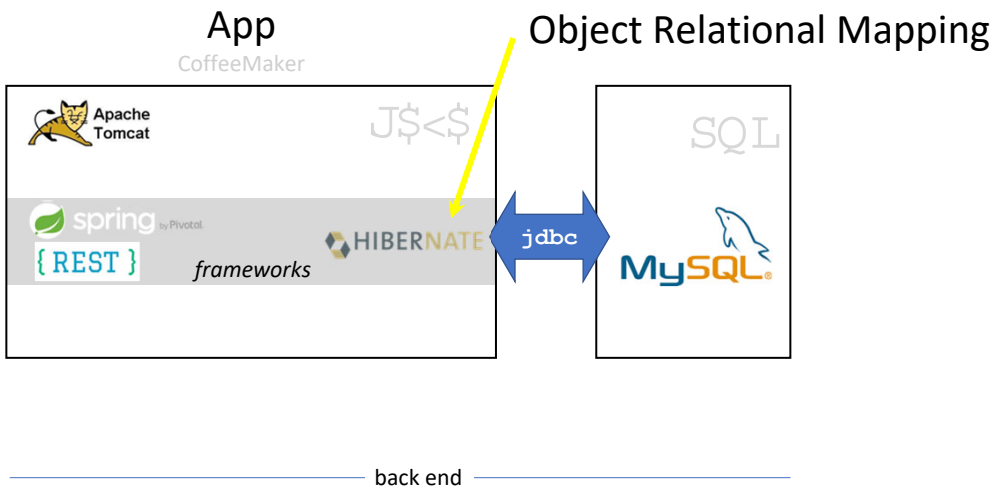
Extra food - Analyze Stack Overflow data!



Data explorer: https://data.stackexchange.com/stackoverflow/query/new

Database schema documentation: https://meta.stackexchange.com/questions/2677/database-schema-documentation-for-the-public-data-dump-and-sede

326 App Architecture



© NC State Software Engineering Faculty

HIBERNATE



- Hibernate is an Object Relational Mapping persistence framework that supports the connection between Java Objects and persistent storage in a Relational Database Management System (RDBMS)
- Annotations are used to connect classes/fields to database tables/columns

- coffee_maker Specifics
  - Information to connect with the actual running relational database application is in **`application.properties`**
  - All persistence classes use the **`@Entity`** annotation
  - Represent DAO (Data Access Object)
  - Standard CRUD operations handled by **`*Repository`**
    - Extends **`JpaRepository`**

Create
Read
Update
Delete

© NC State Software Engineering Faculty

HIBERNATE



- Hibernate provides several annotations to describe relationships between classes
  - @OneToOne
  - @OneToMany & @ManyToOne
  - @ManyToMany
  - @ElementCollection
- The annotations are used on collections and references to create unidirectional or bidirectional associations as appropriate for the objects in your system.

© NC State Software Engineering Faculty

Classes → Tables

- When using Hibernate, we create **classes** that correspond to *tables*
- **Fields** in the *class* are **columns** in the *table*
- **Rows** in the *table* are **instances** of the *class*

- So far, we've created simple tables that use standard types

- What if we want something more complex?

© NC State Software Engineering Faculty

Creating Tables

- Defines the columns and their types
- Name comes before the type
- There may be additional information about the columns
  - Auto-increment
  - Not null
- Can include information about a ***primary key***
  - One or more columns that uniquely identify a row

```
1  CREATE TABLE recipes (
2    id BIGINT NOT NULL AUTO_INCREMENT,
3    name VARCHAR(20) NOT NULL,
4    price INT,
5    coffee INT,
6    milk INT,
7    chocolate INT,
8    PRIMARY KEY (id)
9  );
```

Result Grid | Filter Rows: | Edit:

	id	name	price	coffee	milk	sugar	chocolate
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

© NC State Software Engineering Faculty

Primary Keys

- A way to uniquely identify a row in a database table
- With Hibernate, we typically use an id column to create the primary key
  - But you can have two, three, or more columns that together uniquely identify a row in a table
- You can select information about a row in a table using the primary key

```
SELECT * FROM recipes WHERE id = 3;
```

© NC State Software Engineering Faculty

Composition in Persistent Objects

- Suppose that we wanted to keep track of coffee purchases

- Transaction
  - id
  - Recipe purchased
  - Amount paid

```
CREATE TABLE transactions (
  id BIGINT NOT NULL AUTO_INCREMENT,
  recipe_id BIGINT,
  paid INT,
  PRIMARY KEY (id)
);
```

- How do we tie the Recipe to a Transaction in a Database?

© NC State Software Engineering Faculty

Connecting Tables - Join

- One table can refer to a row in another table using the id

transaction Table

id	recipe_id	paid
1	1	75
2	1	50
3	2	100
4	3	150
5	1	55

recipe Table

id	name	price	coffee	milk	sugar	chocolate
1	Coffee	50	1	1	1	0
2	Coffee	75	2	1	1	0
3	Latte	75	2	2	1	0
4	Mocha	75	2	2	0	1
5	Hot Chocolate	100	0	2	1	2

```
SELECT *
FROM transaction, recipe
WHERE transactions.recipe_id = recipe.id;
```

© NC State Software Engineering Faculty

Connecting Tables (2)

- Does this cause an error?

```
INSERT INTO transactions (recipe_id, paid)
VALUES (45, 5000);
```

- There is no recipe with an id of 45!
- There's no error. Should we be able to do this?

© NC State Software Engineering Faculty

Foreign Keys

- A ***foreign key*** is a key that refers to the primary id of another table!
  - Could be more than one column/field
  - Explicitly set when a table is created!
  - Helps with error checking.

transaction Table

id	recipe_id	paid
1	1	75
2	1	50
3	2	100
4	3	150
5	1	55

recipe Table

id	name	price	coffee	milk	sugar	chocolate
1	Coffee	50	1	1	1	0
2	Coffee	75	2	1	1	0
3	Latte	75	2	2	1	0
4	Mocha	75	2	2	0	1
5	Hot Chocolate	100	0	2	1	2

© NC State Software Engineering Faculty

Transactions with Foreign Key

- By including a foreign key, the database will check to ensure that the row matching the foreign key value is there before doing an insertion!
- There's also some nice things you can do with cascading deletes
  - But we don't want to do that here, we want to keep our recipes

```
CREATE TABLE transactions (
   id BIGINT NOT NULL AUTO_INCREMENT,
   recipe_id BIGINT,
   paid INT,
   PRIMARY KEY (id),
   FOREIGN KEY (recipe_id)
       REFERENCES recipes(id)
);
```

© NC State Software Engineering Faculty

Transactions with Many Beverages

- What if a transaction could include multiple beverages?
- We can't include multiple items in a cell – it can only hold one thing
- Instead, we can create an additional table, called a *join table*, to help us

transactions Table

id	paid
1	175
2	50
3	100
4	150
5	55

transaction_recipe Table

transaction_id	recipe_id
1	1
1	2
2	1
3	3
4	4
5	2

recipe Table

id	name	price	coffee	milk	sugar	chocolate
1	Coffee	50	1	1	1	0
2	Coffee	75	2	1	1	0
3	Latte	75	2	2	1	0
4	Mocha	75	2	2	0	1
5	Hot Chocolate	100	0	2	1	2

© NC State Sc

Transactions with Many Beverages (2)

transactions Table

id	paid
1	175
2	50
3	100
4	150
5	55

transactions_recipes Table

transactions_id	recipe_id
1	1
1	2
2	1
3	3
4	4
5	2

recipe Table

id	name	price	coffee	milk	sugar	chocolate
1	Coffee	50	1	1	1	0
2	Coffee	75	2	1	1	0
3	Latte	75	2	2	1	0
4	Mocha	75	2	2	0	1
5	Hot Chocolate	100	0	2	1	2

```
create table transactions (
   id BIGINT NOT NULL AUTO_INCREMENT,
   paid INT,
   PRIMARY KEY (id)
);
create table transactions_recipes (
   transactions_id BIGINT,
   recipe_id BIGINT,
   FOREIGN KEY (transactions_id)
      REFERENCES transactions(id),
   FOREIGN KEY (recipe_id)
      REFERENCES recipes(id)
);
```

43

43

HIBERNATE



Spring manages Hibernate for us

- Controls the connection to the database
  - Using credential in the **`application.properties`** file
- Spring Data JPA (Java-Persistence API) will write the SQL needed for common operations
  - See docs for examples

© NC State Software Engineering Faculty

44

44

HIBERNATE



- **`entity` Package**
  - POJOs with annotations to connect to DB
  - DAOs - Data Access Objects
- **`repositories` Package**
  - Extends **`JpaRepository`**
  - Has standard CRUD operations
  - Can add methods for additional operations (see **`RecipeRepository`**)
- Hibernate builds and sends the Standard Query Language (SQL) statements for the DB to execute (SQL Cheat Sheets)
  - SELECT → get*()
  - INSERT/UPDATE → save()
  - DELETE → delete()
- The results are returned from the database and transformed into Java objects you can work with.

© NC State Software Engineering Faculty

HIBERNATE



Annotations *start with @*

recipe Table

1	Coffee	50	2	1	1	0
2	Mocha	75	2	1	1	1

Annotations

```
@Entity
@Table (name = "recipes")
public class Recipe {
    private Long id;
    private String name;
    …
    public Recipe() { //needed to construct empty object
    }
```

Specifying the table name is optional. If not specified, table name defaults to name of class.

© NC State Software Engineering Faculty

HIBERNATE Annotations



database entity

`@Table` ties **Recipe** object to the *recipes* table

Annotations

```
@Entity
@Table (name = "recipes")
public class Recipe {
    private Long id;
    private String name;
    …
    public Recipe() { //needed to construct empty object
    }
```


47

HIBERNATE



Fields are the columns in the table.
Annotations describe column characteristics

```
public class Recipe {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @NotNull
    private String name;

    private Integer price;

    private Integer coffee;

    private Integer milk;
    …
```

recipe Table

id	name	price	coffee	milk	sugar	chocolate
1	Coffee	50	2	1	1	0
2	Mocha	75	2	1	1	1

```
CREATE TABLE recipe (
  id BIGINT NOT NULL AUTO_INCREMENT,
  name VARCHAR(20) NOT NULL,
  price INT,
  coffee INT,
  milk INT,
  sugar INT,
  chocolate INT,
  PRIMARY KEY (id)
);
```


48

HIBERNATE



- Testing with databases adds extra complications
  - Best practice: separate test database from production database
  - Best practice: clear out database between each test and populate with the data needed for your test
    - This can be time consuming
- Mocking
  - The database may also be mocked, or faked, by creating an object that acts like a database but is seeded with test data.
  - See Mockito (site.mockito.org) for a mocking library

© NC State Software Engineering Faculty