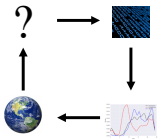
Review

DS 100, Spring 2017

Slides by:

Joe Hellerstein

hellerstein@berkeley.edu



Format of Exam

- When: Thursday 5/11 3PM
- Where: Genetics and Plant Bio 100
- How: Very similar to midterm in structure
- Cheat Sheets: 2!
- Topic areas: See syllabus.

Topics Covered in This Deck

- Relational Algebra and SQL
- Pandas
- Big Data: Data Warehousing, Map-Reduce, Spark
- Data Wrangling

Context and Connections

- We have seen a variety of data manipulation tasks
  - Both data "wrangling" and analysis
  - Diverse languages and frameworks
    - Pandas, SQL, MapReduce, Spark
- Check your fundamentals, study commonalities!
  - Matrix and set operations
  - Relational Algebra & Map/Reduce
    - Can you translated the operators from one into the other?
  - What operations appear frequently?
    - E.g. joins, groupby/sum/product, filtering on rows and columns
    - Relational algebra and linear algebra!

Relational Algebra & SQL

Relational Model & SQL

- Key Ideas:
  - Schemas, instances, queries, views, insert/delete/update
  - Relational algebra operators
  - SQL as a declarative language: everything referenced *by value*
    - *Not by "index" as in arrays or dataframes!*
  - Many big data features under the covers: parallel execution, user-defined functions and aggregates
- What you should know:
  - Relational/SQL model basics
  - Relational algebra and basic equivalences
  - Basic SQL structure and syntax. Recognize incorrect logic (as opposed to syntax errors)
- What you don't need to know:
  - You will not be asked to compose queries from scratch
- How to study?
  - Review HW4, SQL notebook files from lecture

Relational Model

- Key Idea:
  - Schema: describes attribute (column) names and types, keys
  - Instance: a *set* (unordered, duplicate-free) of tuples (rows)
- Review the Boat Club from class

```
CREATE TABLE sailors(sid integer PRIMARY KEY, sname text,
                     rating integer, age float);

CREATE TABLE boats(bid integer PRIMARY KEY, bname text, color text);

CREATE TABLE reserves(sid integer, bid integer, day date,
                      PRIMARY KEY (sid, bid, day),
                      FOREIGN KEY sid REFERENCES sailors,
                      FOREIGN KEY bid REFERENCES boats);
```

Boats

bid	bname	color
101	Interlake	blue
102	Interlake	red
104	Marine	red
103	Clipper	green

Reserves

sid	bid	day
28	101	10/10/16
58	103	11/12/96

Sailors

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

Note: primary keys underlined

Relational Algebra Operators

Unary Operators: *operate on **single** relation instance*

- **Projection (π):** *Retains only desired columns* (vertical)
- **Selection (σ):** *Selects a subset of rows (horizontal)*
- **Renaming (ρ):** *Rename attributes and relations.*

Binary Operators: *operate on **pairs** of relation instances*

- **Union ($\cup$):** *Tuples in r1 or in r2.*
- **Intersection ($\cap$):** *Tuples in r1 and in r2.*
- **Set-difference (—):** *Tuples in r1, but not in r2.*
- **Cross-product ($\times$):** *Allows us to combine two relations.*
- **Joins ($\bowtie_\theta$, $\bowtie$):** *Combine relations that satisfy predicates*

As you review this material, do you see how it relates to SQL, Pandas and Spark?

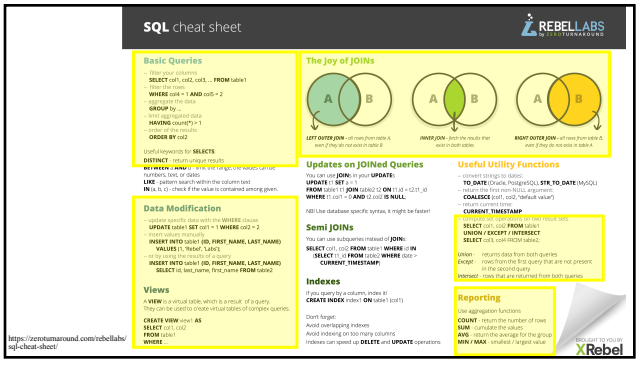
SQL Data Model

- Basically relational, but "multisets"
  - I.e. can have duplicate rows in a table, and they matter
    - E.g. for COUNT, AVG, etc.
  - I.e. some tables have *no* primary key!
- FOREIGN KEYs

SQL

A declarative language: say "what" you want in the output, not "how" to get it.

DDL	DML
`CREATE TABLE (<col> <type>, ...);`	`SELECT ...`
Understand PRIMARY KEY, FOREIGN KEY clauses.	`FROM ...`
	`[WHERE ...]`
`INSERT INTO TABLE values (…), (…), …;`	`[GROUP BY ...]`
`INSERT INTO TABLE SELECT …;`	`[HAVING ...]`
	`[ORDER BY ...]`
`UPDATE TABLE SET … WHERE …;`	`[LIMIT ...];`
note: value-based references in UPDATE!	



Also

- TABLESAMPLE BERNOULLI(…)
- Scalar functions (like "map") in SELECT and WHERE
  - Arithmetic, Math functions, string functions…
- Aggregate functions (like "reduce") in SELECT and HAVING
  - count(), sum(), average(), stddev()
  - Note: DISTINCT clause inside aggregates
- Table functions: generate_series()

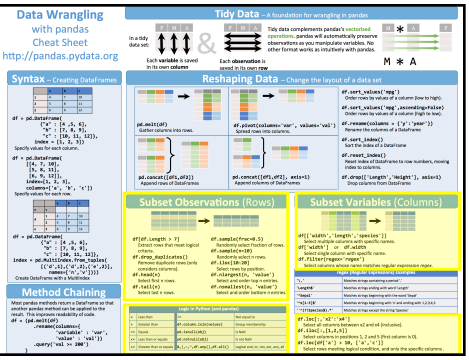
Pandas and Numpy

Arrays and DataFrames

- Key Ideas:
  - Array is a multi-d array; single type for all cells
  - Pandas DataFrame is like a relation (table); single type per column
    - Unlike relational model, fixed order to rows (allowing array-style indexing)
- What you should know:
  - Basic numpy array handling
  - Key pandas operations (see next slides)
  - Recognize incorrect logic
- What you don't need to know:
  - You will not be asked to write code from scratch
- How to study?
  - Review HW1-3, notebook files from lecture
  - Be able to translate basic SQL block above into Pandas

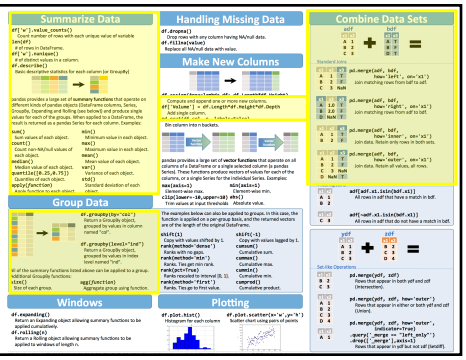
Topics

- Data Frame & Numpy Arrays
  - Subsets
  - Groupby/Agg
  - Joining



Topics

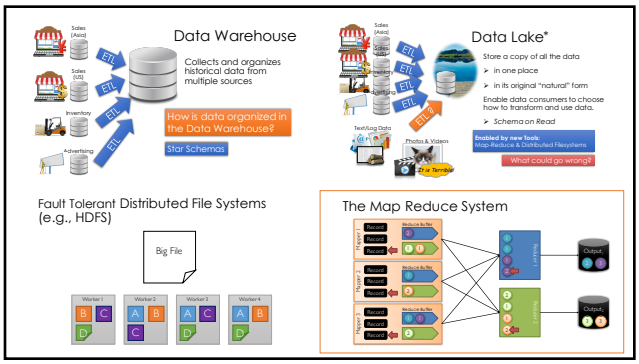
- Data Frame & Numpy Arrays
  - Subsets
  - Indexing
  - Groupby/Agg
  - Joining

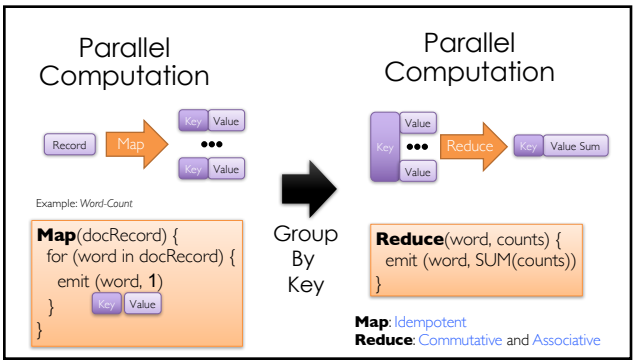


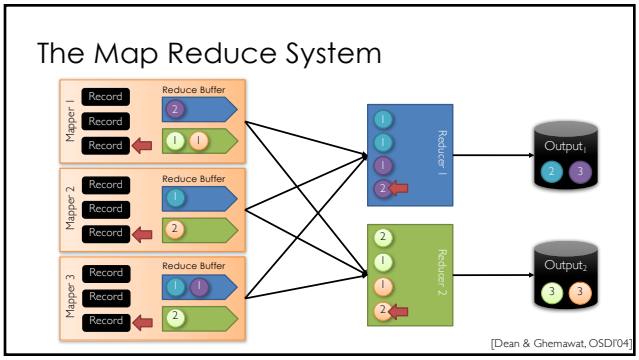
Big Data

Big Data

- Key Ideas:
  - Schema on load (data warehousing) vs. on use (data lakes)
  - Fault-tolerant file systems
  - Data-parallel computing (e.g. MapReduce)
  - Statistical Query Pattern
- What you should know:
  - Star schema design
  - File system data layouts and failure handling
  - Map vs. Reduce
  - Spark Transformations vs. Actions
  - How to compute basic linear algebra operations in statistical query pattern
- What you don't need to know:
  - You will not be asked to write code from scratch
  - Fault tolerance in MapReduce
- How to study?
  - Review HW 7 and lecture notebooks

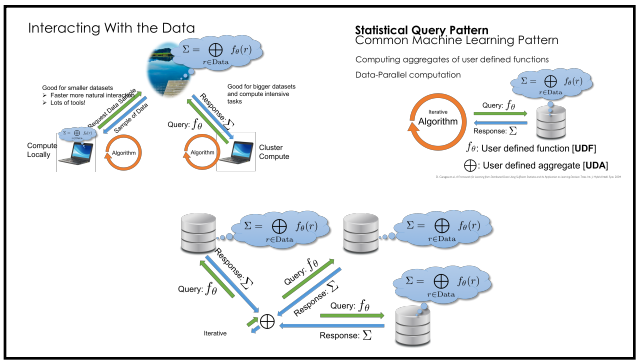






Spark

- A newer MapReduce framework
  - With higher-level programming APIs: **Dataframes & SQL**
- Data representations:
  - *Resilient Distributed Dataset (RDD):* A collection of **objects**
    - Objects could be text, images, python dictionaries …
  - *DataFrame:* A collection of **rows** with common schema
- Computation
  - **Transformations:** users apply functions to the distributed collections which return new distributed collections
    - E.g., map, flatmap, filter, reduceByKey
  - **Actions**: users apply functions which return values:
    - E.g., count, sum, collect, take, first, reduce



Data Wrangling

Data Wrangling

- Key Ideas:
  - Wrangling output determines your downstream analytics
  - Assessing data quality like querying + vis
  - Data transformation also like querying!
  - A petri dish for many of the topics in class
- What you should know:
  - Structural issues: arrays, relations, and converting between
  - Granularity: assessing it (keys) and changing it (grouping)
  - Faithfulness: finding and handling outliers
  - Basic UNIX commands
- What you don't need to know
  - UNIX command flags
  - Trifacta Wrangler
  - Memorizing "Structure, Granularity, Faithfulness, Temporality, Scope" (that's just to organize the ideas)
  - Live-coding Pandas/Python details like "read_csv" or strip() or head()
- How to study?
  - Lecture notes
  - HW3

Rough Guide to Topics

- **Structure: the "shape" of a data file**
- **Granularity: how fine/coarse is each datum**
- **Faithfulness: how well does the data capture "reality"**
- Temporality: how is the data situated in time
- Scope: how (in)complete is the data

We didn't really discuss temporality/scope much

Structure

- Relations and Relational languages (algebra, SQL)
- Arrays and Linear Algebra
- Coarse structure
  - Record and field delimiters
  - Common schemas
  - Nested?
- Value encoding: nominal vs. ordinal vs. quantitative
- See connections to SQL and DataFrames, n-d Arrays

Granularity

- Keys as an indicator of granularity
- Granularity can be coarsened via groupby
  - Grouping columns become a primary key of result
- Granularity can be refined via join
  - With more detailed tables
  - Foreign keys
  - Understand the # of matches for various joins

Faithfulness

- Data *in context*
  - Application context
    - E.g. human beings are unlikely to be 1000 years old
    - Constraints: relation to "types" or "domains"
      - E.g. "123456789" is not a valid zip code
      - E.g. the word "spunow" is not in the english dictionary
  - Cross-record context
    - Detecting Outliers
      - "Center": e.g. average and median
      - "Spread": e.g. stdev, inter-quartile range
    - Resolving Outliers
      - Trimming: setting outlier values to N/A or NULL
      - Winsorizing: setting outlier values to the nearest non-outlier value