USC Viterbi
School of Engineering

Introduction & an Overview of Relational Database Systems

CSCI 550: Advanced Data Stores
Spring 2024

Ibrahim Sabek
sabek@usc.edu

Course Staff

Ibrahim Sabek (Instructor)
sabek@usc.edu

Meet Raval (TA)
mkraval@usc.edu

Vash Singh (CP)
singhvas@usc.edu

Administrivia

Course materials

- Instructor topics, CMU (Advanced DB Systems), DB Redbook (www.redbook.io/)
  - No textbook! Mostly research papers
- Lectures will be posted on Blackboard

Ask questions on Piazza! Use signup link

Gradescope for uploading reports and grading! Use entry code

Instructor office hours

- TuTh 4:00–5:00 pm
- Office: PHE 420

TA office hours: TBD

Administrivia

No lecture recording, unless stated otherwise!

In-person attendance, unless permission is granted

Interactive class! Raise your hand if you have questions

2-hours class

1 hour for the instructor's materials

1 hour for the students' presentations (3 20-min presentations, individual)

Starting on Jan 16 (Tuesday)

15% of total grade

Guest lectures (maybe!)

Administrivia

Review reports (25% of total grade)
- Conference review style!
- 8 reports (almost weekly, topic-based, individual)
  - 1st report due is Jan 23 (Tuesday)
- Late submission policy (max of 3 days late per report, reduce max. credit by 20% each day)

Checkpoint exams (30% of total grade)
- 2 exams (middle of the semester, end of the semester)

Course project (30% of total grade)
- Groups of 2-3
- Extend an open-source DBMS with a feature!
- Project proposal, implementation, presentation, and conference paper-style report
- 1 status update meeting

Academic honesty
- Refer to the USC Academic Integrity Policy
- Plagiarism will **not** be tolerated (writing and implementation must be your own)

Why Should I Take CSCI 550?

Designing and building database management systems (DBMSs), not how to use them!

- We will cover state-of-the-art and modern DBMS techniques (not classical DBMSs)
- Mostly in-memory DBMSs (components + overall system design)

DBMS developers are needed everywhere!

What Is A Database?

Structured Data Collection (Tuples + Relationships)

Database Management Systems (DBMSs)

Software systems for storing, querying, and processing data in databases

Source: mattturck.com

Main Modern (In-memory) DBMS Concepts

Indexing (OLAP and OLTP)

Query Pipelining and Materialization

Query Scheduling

Vectorized Execution

Query Compilation and Optimization

Consistency and Concurrency Control

Scalable System Architectures

A Database from A User Perspective

How to model and query data as a user

Zoo website features

Admin interface

Add and edit an animal

Public interface

Pictures and maps

Zookeeper interface

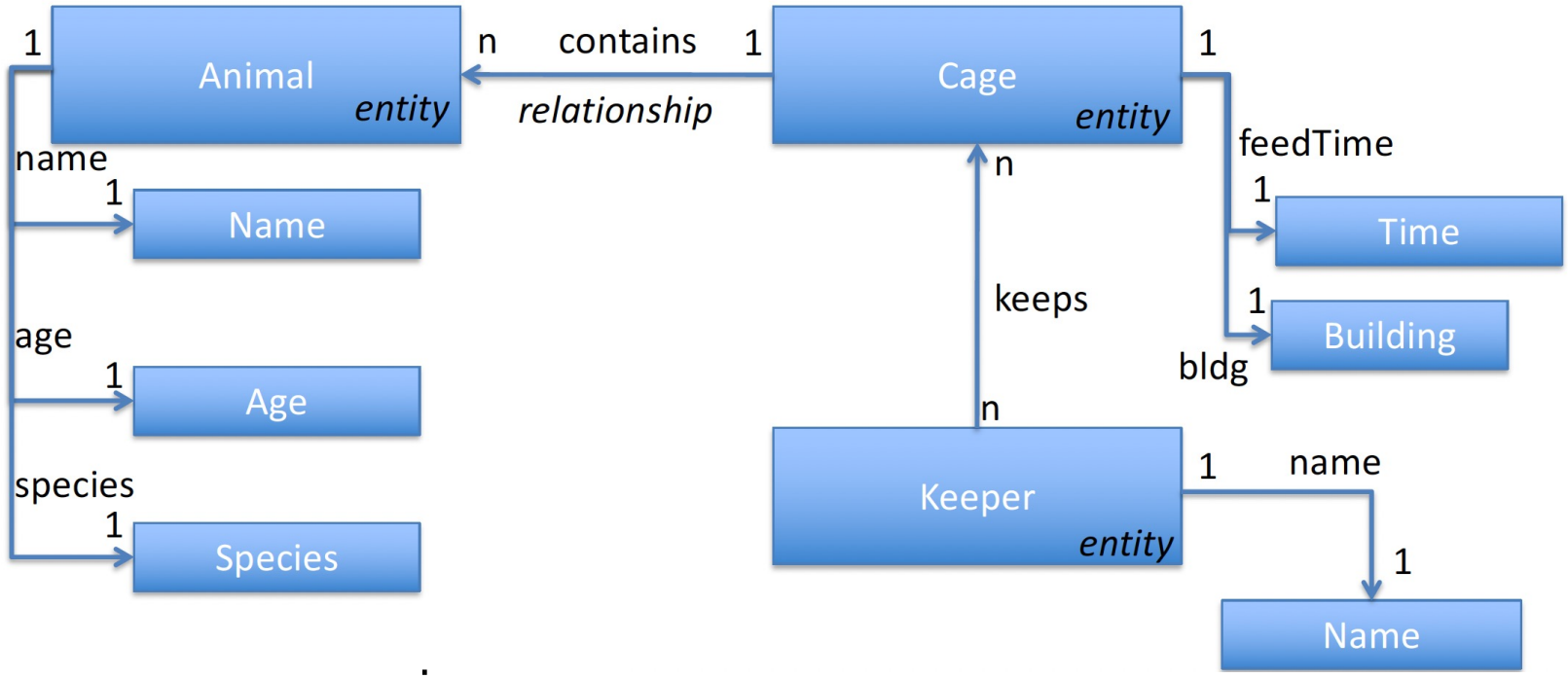
Feed times

1K animals, 5K URLs, 10 admins, 200 keepers

Source: MIT 6.5830 (Readings in Databases)

Zoo Data Model

Entity Relationship Diagram (ERD)



Animals have names, ages, species
Keepers have names
Cages have cleaning times, and buildings
Animals are in 1 cage; cages have multiple animals
Keepers keep multiple cages, cages kept by multiple keepers

Structured Query Language (SQL)

```
SELECT  field1, …, fieldM
FROM table1, …
WHERE condition1, …

INSERT INTO table VALUES (field1, …)

UPDATE table SET field1 = X, …
WHERE condition1,…
```

Structured Query Language (SQL)

Names of giraffes

Imperative

```
for each row r in animals
  if r.species = 'giraffe'
    output r.name
```

Declarative

```
SELECT r.name FROM animals
WHERE r.species = 'giraffe'
```

Structured Query Language (SQL)

Cages in Building 32

Imperative

```
for each row a in animals
   for each row c in cages
      if a.cageno = c.no and c.bldg = 32
         output a
```

Nested Loop

Declarative

```
SELECT a.name FROM animals AS a, cages AS c
WHERE a.cageno = c.no AND c.bldg = 32
```

Join

Structured Query Language (SQL)

Average age of bears

Declarative

```
SELECT AVG(age) FROM animals
WHERE species = 'bear'
```

Declarative Queries: What Not How!

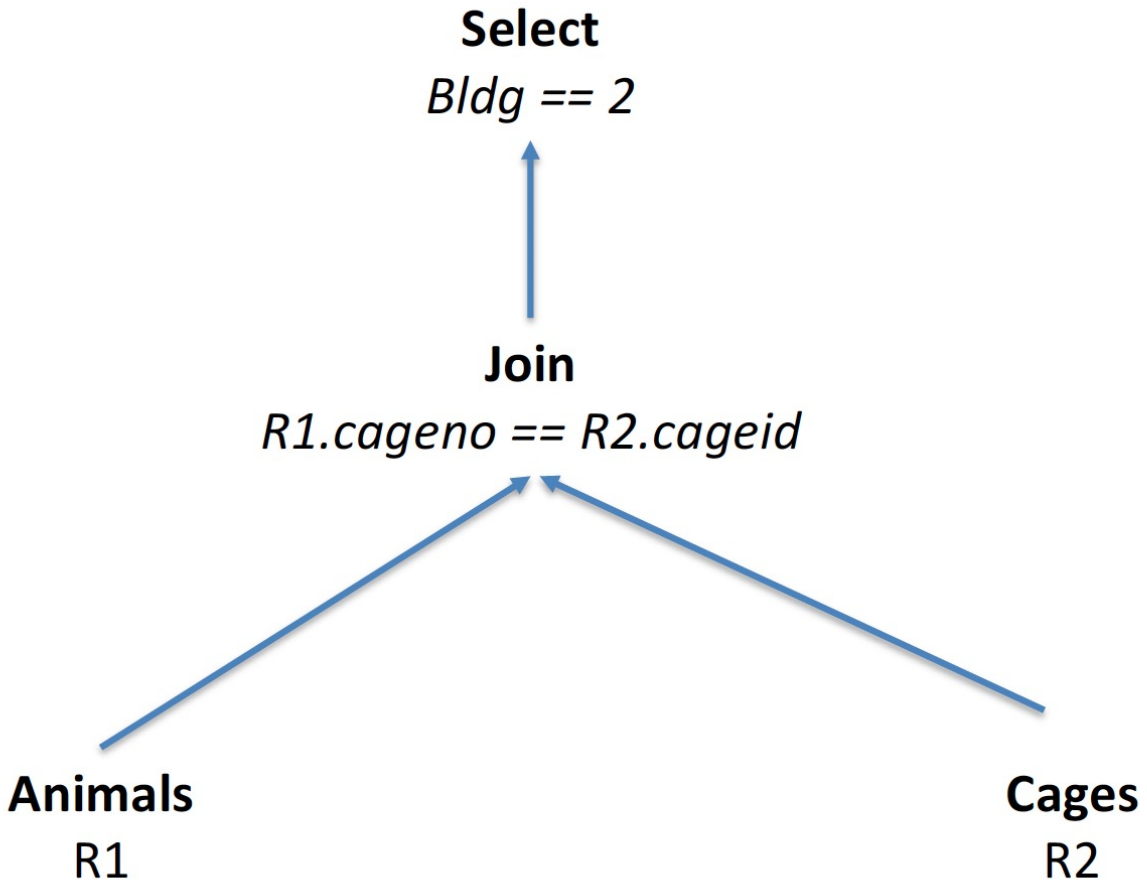
Many possible procedural plans for a given SQL query

What else could we do for the ”bear” query?

- Sort animals according to type
  - good for the “bear” query
  - inserts are slower
- Store animals table in a hash table or tree (“index”)

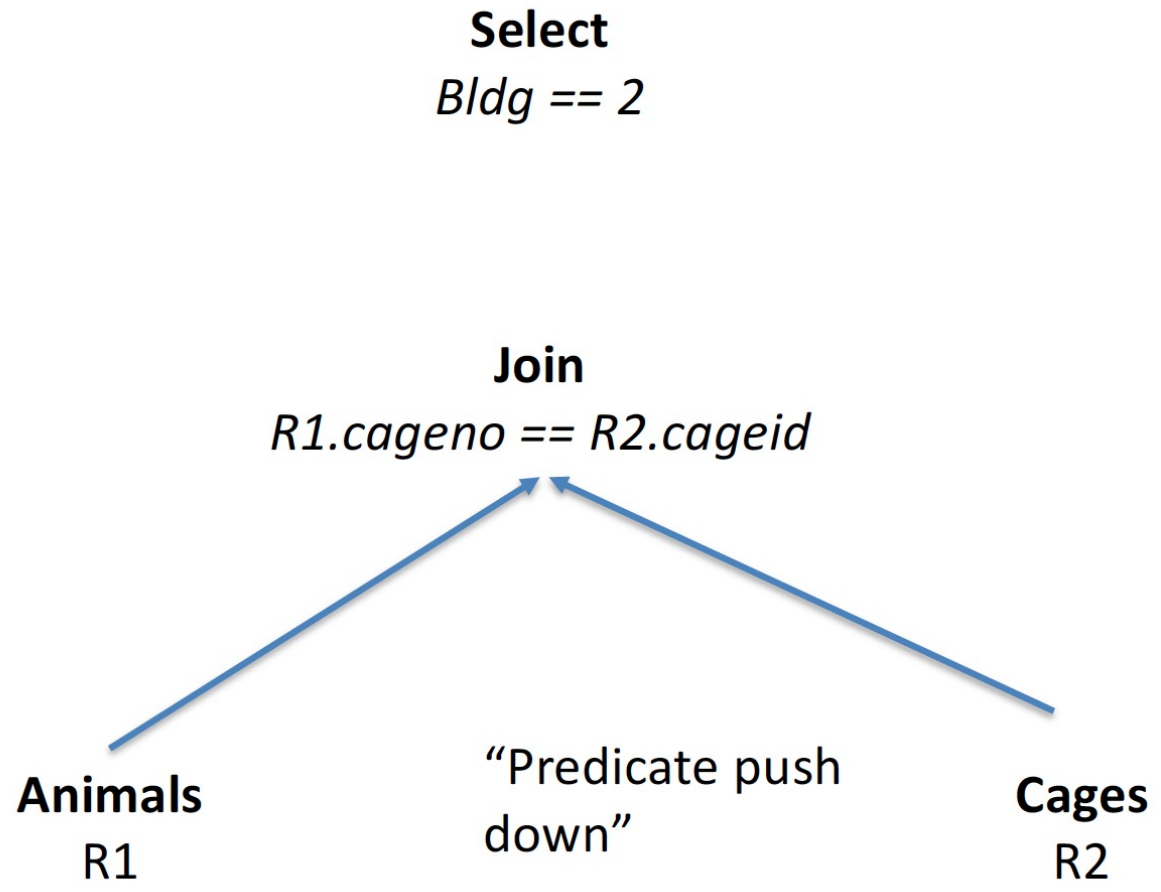
From Logical Plan to An Optimized Plan

Cages in Building 2



From Logical Plan to An Optimized Plan

Cages in Building 2



Summary

DBMSs are the key success to many revolutionary applications we witness

Different components interact together

- Data Modeling
- Declarative Querying
- Query Optimization
- Efficient access and updates to data
  - Recoverability
  - Consistency

Administrivia: be active and ask if you have questions