Syllabus & DB Background

CIS 6500

Ryan Marcus

About Me

- Extensive bio: https://ryanmarc.us

Prereqs

- Database course
- Discrete math, basic algorithms and data structures knowledge

Topics

- OLAP databases (as opposed to OLTP)
- Data layouts (row stores, column stores)
- Query execution in modern systems
- Query optimization in modern systems
- Index structures and access methods
- ML techniques for data management

Learning Goals

- Mastery of the course topics
- Ability to pick up and read an academic paper about databases
- Practice preparing and giving a lecture on advanced technical topics

Seminar-Style Course

- First few weeks: lectures from me
  - Background, gap-filling
- After that, each course led by a student
  - Assigned paper
  - Non-presenting students: write short summary
  - Presenting students: 3-page summary + lecture + discussion

Short Summary

- Read each paper before class
- 100 – 300 word summary of each paper, due at start of class sharp

Long Summary

- 2-4 pages (target 3 pages), “walk-through” of the technique
- With a worked example that was not in the paper
- When will the technique work well / not work well?

Assigning students to papers

- You can view a list of all papers on the syllabus
- On Sept 10th, you can rank your most preferred papers.
- Students presenting earlier will get extra credit (up to 20 points)
- Students will rank papers in order of preference, assignments given on Sept 18th

Participation

- Attendance is important.
- Ask questions, engage in discussion.
- Remember, it’ll be your turn soon.
- Expectation: miss at most 4 class periods
  - Preferably 0 to 1.

Final Projects

- You'll conduct a *research grade* final project for this course
  - Implement and analyze an idea from a paper.
- Proposals: Oct 15th (<1 page)
- Reports: Dec 8th (2-5 pages)
- PhD students: we can align this with your thesis

Final Projects

- If you can’t think of one, we’ll work together to find one.
- Example projects:
  - Implement and measure the performance of “column sketches” over different data sizes
  - Analyze cardinality estimation errors in PostgreSQL

Grades

Component	**Weight**
Short paper summaries	5%
Long paper summaries	20%
Lecture and seminar	30%
Final project	25%
Attendance and participation	20%

Survey

- Go to: https://rm.cab/survey

Databases

What percentage of Fortune 500 companies use databases as part of their core business?

What percentage of Fortune 500 companies use databases as part of their core business?

100%

Database vs. the text file

- Databases were designed to maintain ACID properties:
- A – atomicity. Complex transactions execute entirely or not at all.
- C – consistency. Transactions bring the database from one correct state to another.
- I – isolation. Each transaction sees a complete and correct version the database.
- D – durability. Once a transaction is committed, it stays committed, even if the system crashes.

Relational model

Name	**Dept**	**Admin**	**Admin email**
Ryan	CIS	Cheryl	che@...
Zack	CIS	Cheryl	che@...
Boon	CIS	Cheryl	che@...
Susan	CIS	Cheryl	che@...
Martha	Psych	Martin	mar@...

Relational model

Name	Dept
Ryan	CIS
Zack	CIS
Boon	CIS
Susan	CIS
Martha	Psych

Dept	Admin	Admin email
CIS	Cheryl	che@...
Psych	Martin	mar@...

Relational model

Name	Dept
Ryan	CIS
Zack	CIS
Boon	CIS
Susan	CIS
Martha	Psych

⋈

Dept	Admin	Admin email
CIS	Cheryl	che@...
Psych	Martin	mar@...

=

Name	Dept	Admin	Admin email
Ryan	CIS	Cheryl	che@...
Zack	CIS	Cheryl	che@...
Boon	CIS	Cheryl	che@...
Susan	CIS	Cheryl	che@...
Martha	Psych	Martin	mar@...

Relational model

Professor

Name	**Dept**
Ryan	CIS
Zack	CIS
Boon	CIS
Susan	CIS
Martha	Psych

⋈

Admin

Dept	**Admin**	**Admin email**
CIS	Cheryl	che@...
Psych	Martin	mar@...

=

Name	**Dept**	**Admin**	**Admin email**
Ryan	CIS	Cheryl	che@...
Zack	CIS	Cheryl	che@...
Boon	CIS	Cheryl	che@...
Susan	CIS	Cheryl	che@...
Martha	Psych	Martin	mar@...

```
SELECT *
FROM Professor p, Admin a
WHERE p.Dept = A.Dept
```

Join Algorithms

- Essentially, three options:
- Loop join
- Merge join
- Hash join

OLTP vs OLAP

- OLTP: online transaction processing
- PostgreSQL, MySQL, Oracle
- Focus on inserts, updates, deletes, small reads

- OLAP: online analytics processing
- DuckDB, Redshift, Vertica, BigQuery
- Focus on aggregations, bulk loads, large reads

OLTP vs OLAP

OLTP

```
SELECT *
FROM User u
WHERE u.id = 15;
```

Reads a single row, can take advantage of an index

OLAP

```
SELECT AVG(s.total)
FROM Sale s
WHERE s.date BETWEEN
   ‘010122’ AND
   ‘123022’
```

Reads a lot of data, aggregates

Query Optimization

- Essentially: translate complex requests for information into fast programs.
- Ex: find each professor's admin contact email

PROFESSOR

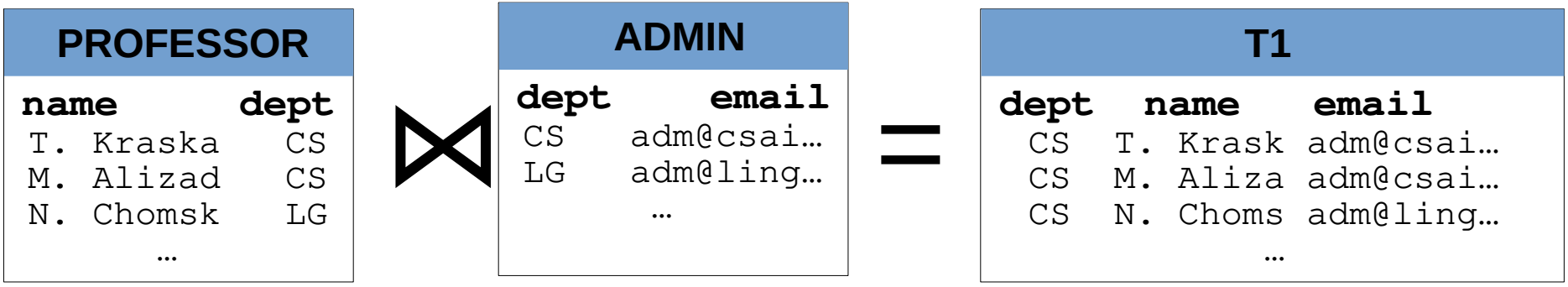
name	dept
T. Kraska	CS
M. Alizad	CS
N. Chomsk	LG
…	

ADMIN

dept	email
CS	adm@csai…
LG	adm@ling…
	…

Query Optimization

- Fundamental operator: join (⋈)



"For every row in this table, emit all the matches from another table."

Many, many ways to perform this operation: loop, hash, sort.

Query Optimization

- Essentially: translate complex requests for information into fast programs.
- Ex: find all movies with Scarlett Johansson produced by Sony

ACTOR

a_id	name	YOB
1	Scarl	84
2	BradP	63
3	JonTr	54
…		

FILM

f_id	name	RAT
1	Her	86
2	Aveng	81
3	PulpF	93
…		

COMPANY

c_id	name
1	Sony
2	Fox
3	MGM
…	

APPEARS_IN

a_id	f_id
1	1
1	2
3	3
…	

PRODUCED

c_id	f_id
1	1
2	2
3	2
…	

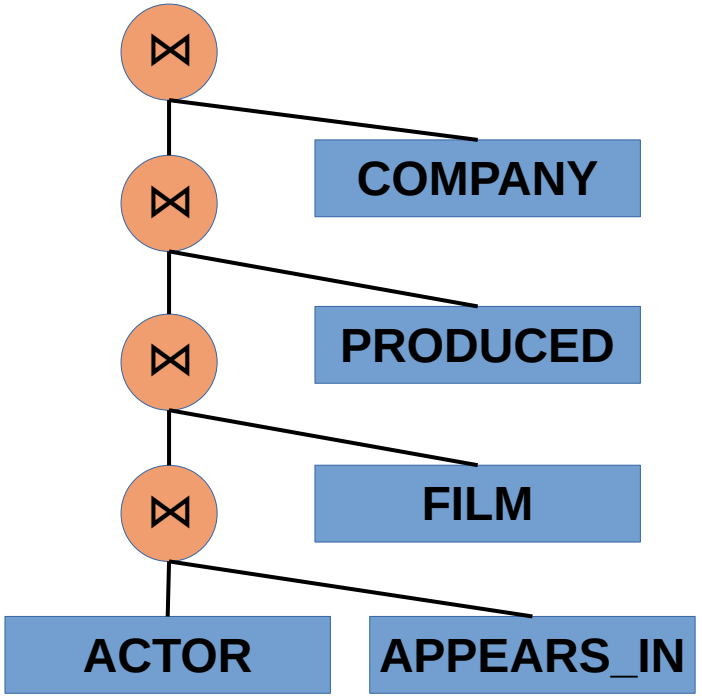
Query Optimization

- Find all movies with Scarlett Johansson produced by Sony.
- *Logically,* what we want is to filter:

$$\text{ACTOR} \bowtie \text{APPEARS\_IN} \bowtie \text{FILM} \bowtie \text{PRODUCED} \bowtie \text{COMPANY}$$

- Physically, I have a lot of options...

Query Optimization



ACTOR

a_id	name	YOB
1	Scarl	84
2	BradP	63
3	JonTr	54
…		

FILM

f_id	name	RAT
1	Her	86
2	Aveng	81
3	PulpF	93
…		

COMPANY

c_id	name
1	Sony
2	Fox
3	MGM
…	

APPEARS_IN

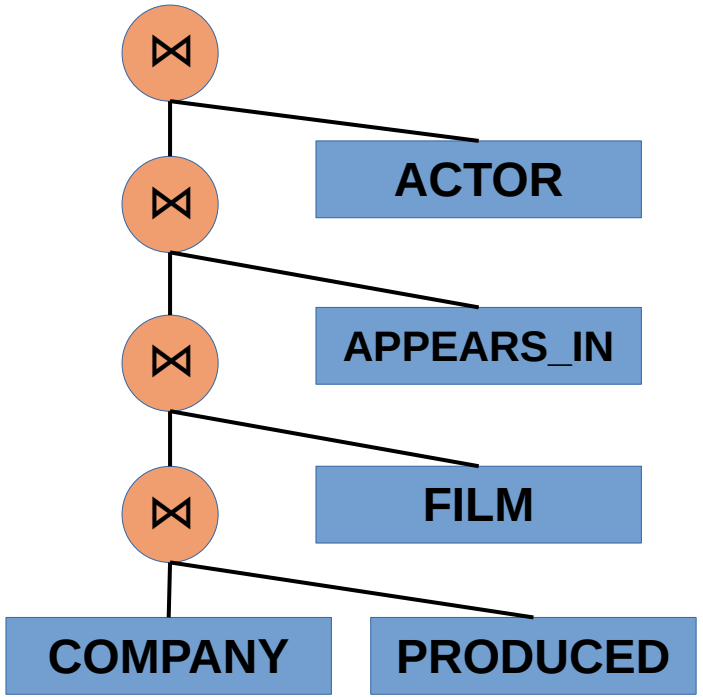
a_id	f_id
1	1
1	2
3	3
…	

PRODUCED

c_id	f_id
1	1
2	2
3	2
…	

Find all movies with SJ. Then, filter those by movies produced by Sony.

Query Optimization



ACTOR

a_id	name	YOB
1	Scarl	84
2	BradP	63
3	JonTr	54
…		

FILM

f_id	name	RAT
1	Her	86
2	Aveng	81
3	PulpF	93
…		

COMPANY

c_id	name
1	Sony
2	Fox
3	MGM
…	

APPEARS_IN

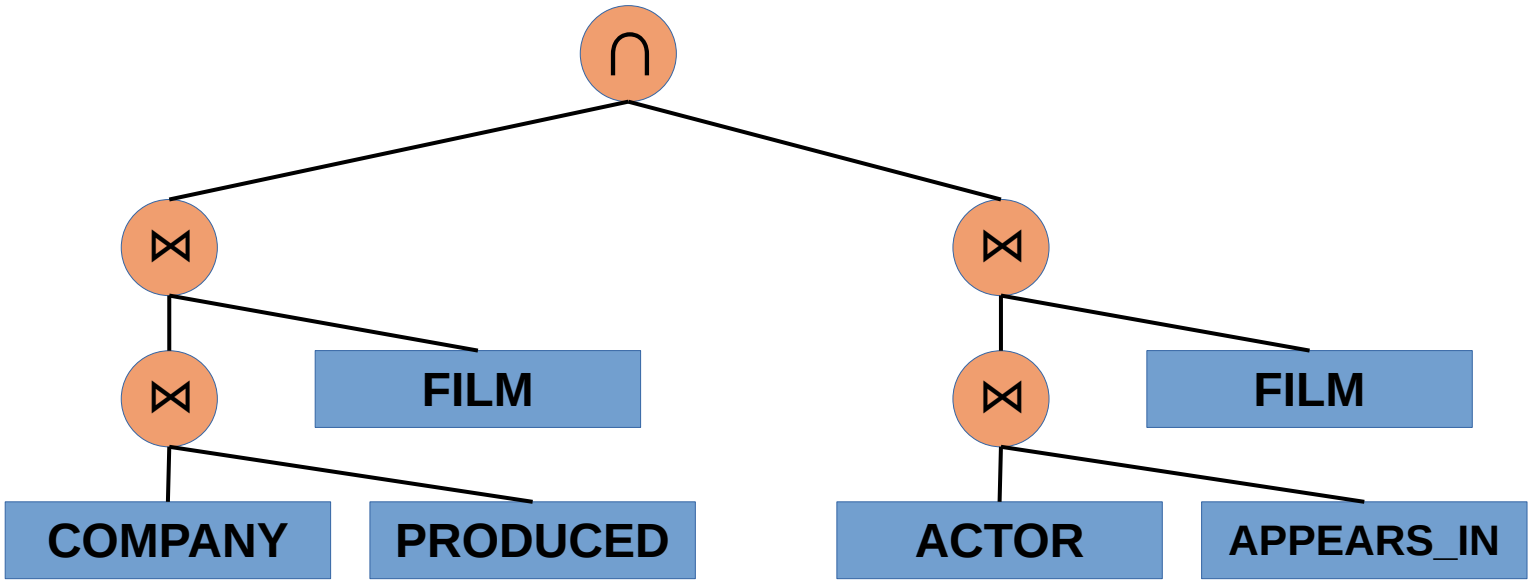
a_id	f_id
1	1
1	2
3	3
…	

PRODUCED

c_id	f_id
1	1
2	2
3	2
…	

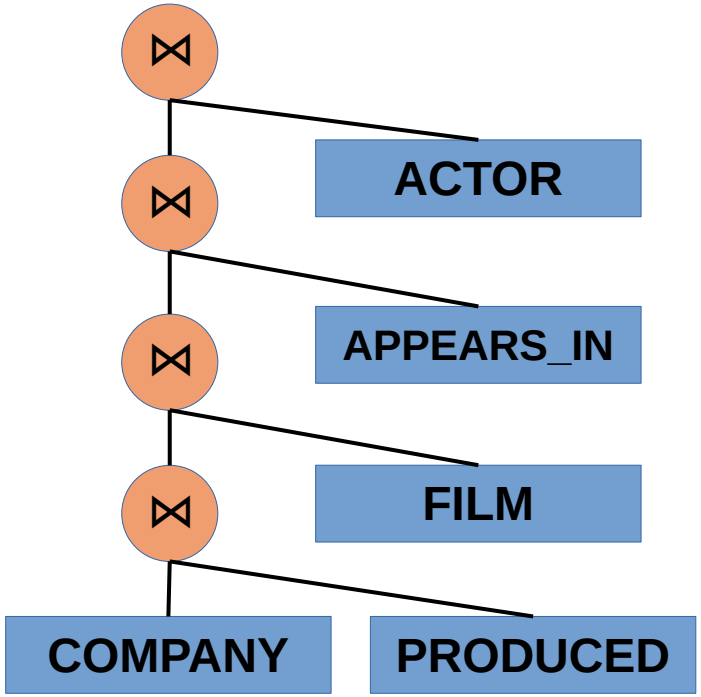
Find all Sony movies. Then, filter by those movies with SJ.

Query Optimization



Find all Sony movies, find all SJ movies, take the intersection.

Query Optimization

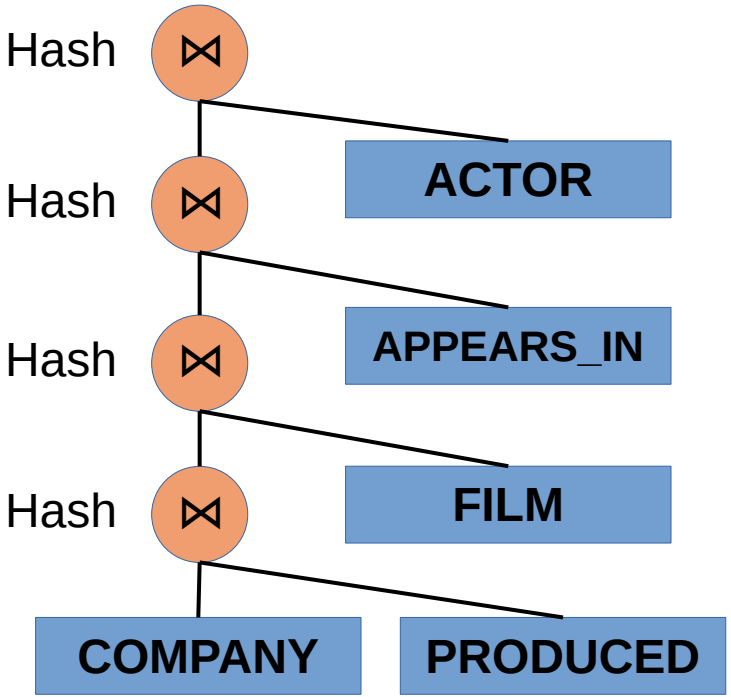


Not quite done…

Still have to choose “physical operators” for each join.

- Ex: hash join. Build a hash table from the LHS, step through with the RHS.
- Ex: sort both relations. Merge them together (merge join style)
- Ex: Nested for loop

Query Optimization

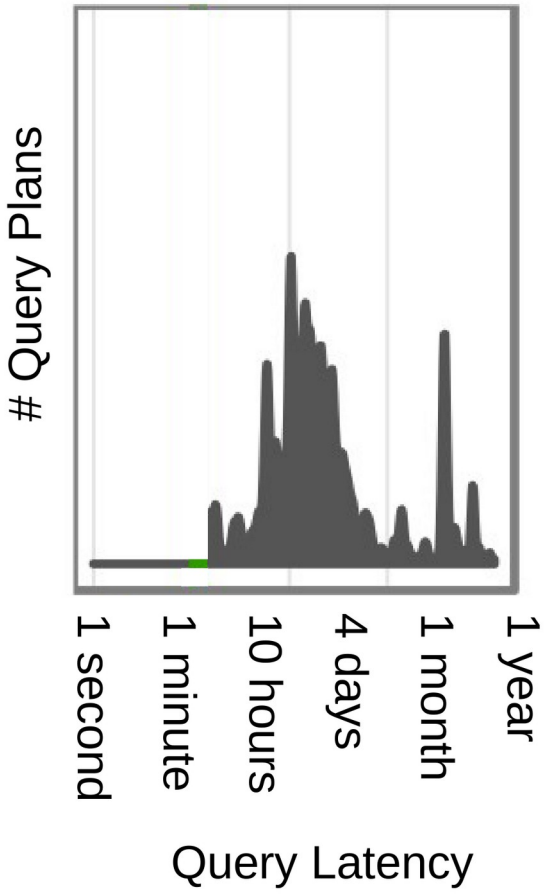


Not quite done…

Still have to choose “physical operators” for each join.

- Ex: hash join. Build a hash table from the LHS, step through with the RHS.
- Ex: sort both relations. Merge them together (merge join style)
- Ex: Nested for loop

Query Optimization



- # plans follows Catalan numbers
- At n = 19, more than 2^32 plans

TODOs for You

- Read the syllabus, make sure the course requirements are clear.
- Check out the papers posted on the website, you'll need to rank them soon.
- Class on 9/3 is CANCELLED
  - Next class is 9/8