WISCONSIN
UNIVERSITY OF WISCONSIN–MADISON

CS 839: Design the Next-Generation Database

Lecture 3: Analytics Basics

Xiangyao Yu

1/28/2020

Announcements

Course website

http://pages.cs.wisc.edu/~yxy/cs839-s20/index.html

Email me if you are not in HotCRP

https://wisc-cs839-ngdb20.hotcrp.com

Email me if you are not enrolled

Office hour: Tue 2:30pm - 3:30pm @ CS 4385

Discussion submission deadline: 11:59pm the day after the lecture

Discussion Highlights

2PL vs. OCC

- 2PL is better for high contention, but needs to handle deadlocks.
- May choose based on application behavior

SQL vs. NoSQL

- A tradeoff of highly-skilled system engineers vs. application developers
- Depends on the application
- Configurable isolation levels

Logging scalability

- I/O cost, context switching cost, hardware buffer bottleneck
- Potential solutions: SSD, asynchronous logging

Today’s Agenda

Relational database

Operations

Row store

Column store

C-Store

OLTP vs. OLAP (recap)

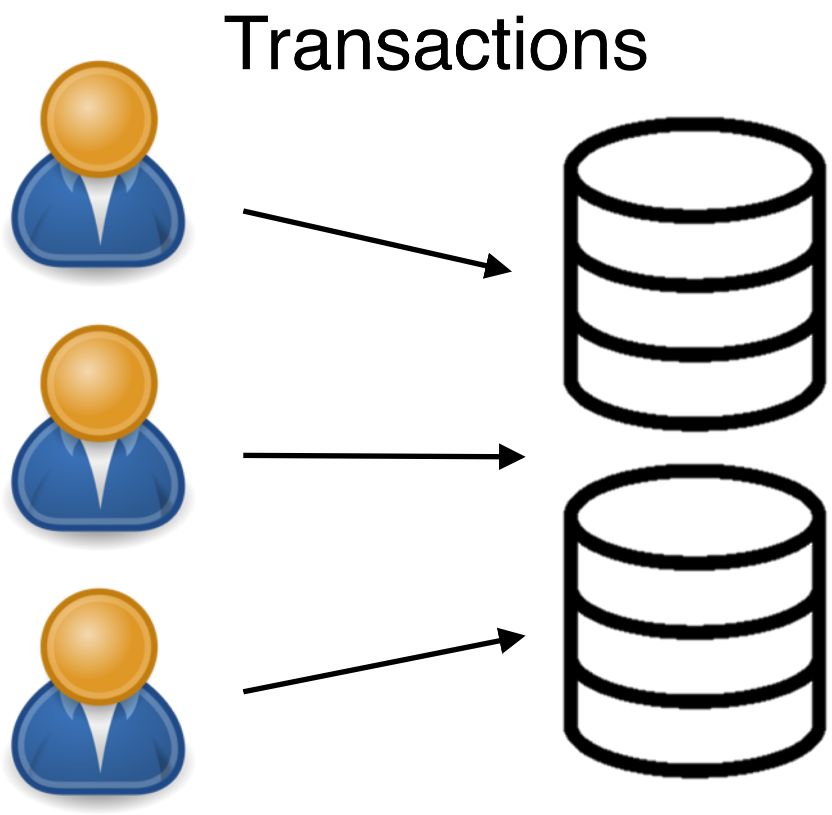
OLTP: On-Line Transaction Processing

- Users submit transactions that contain simple read/write operations
- Example: banking, online shopping, etc.

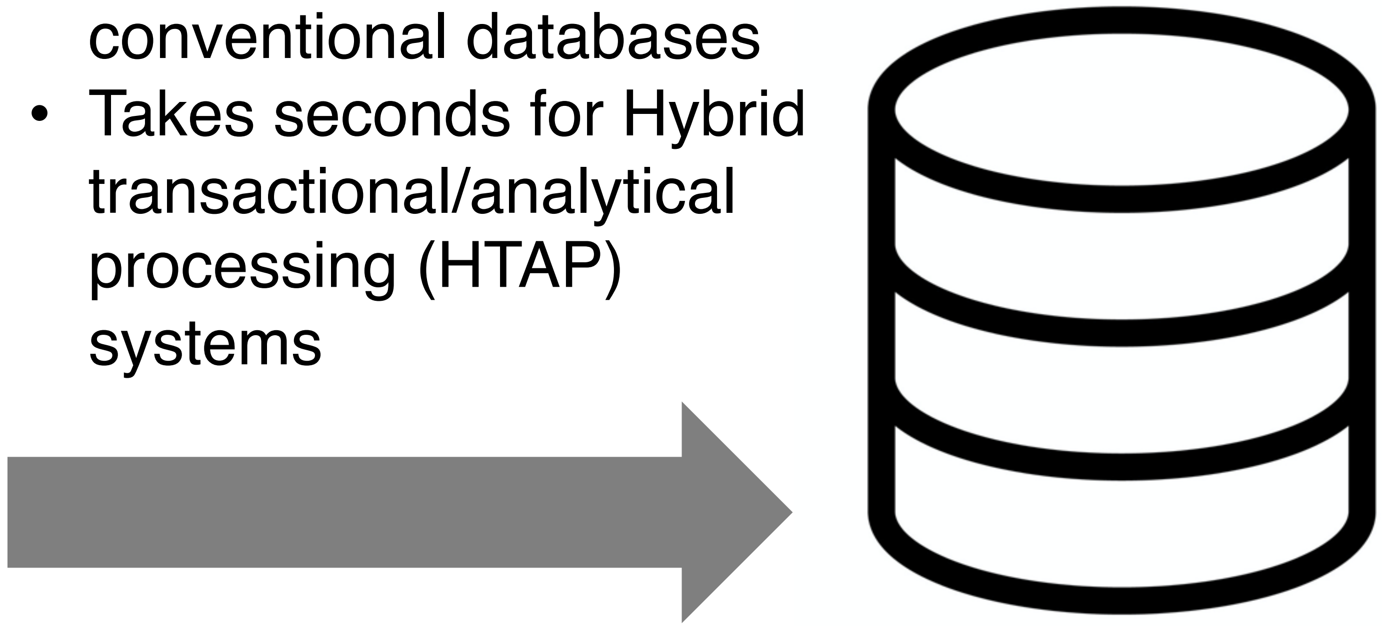
OLAP: On-Line Analytical Processing

- Complex analytics queries that reveal insights behind data
- Example: business report, marketing, forecasting, etc.

OLTP vs. OLAP (recap)



- Takes hours for conventional databases
- Takes seconds for Hybrid transactional/analytical processing (HTAP) systems



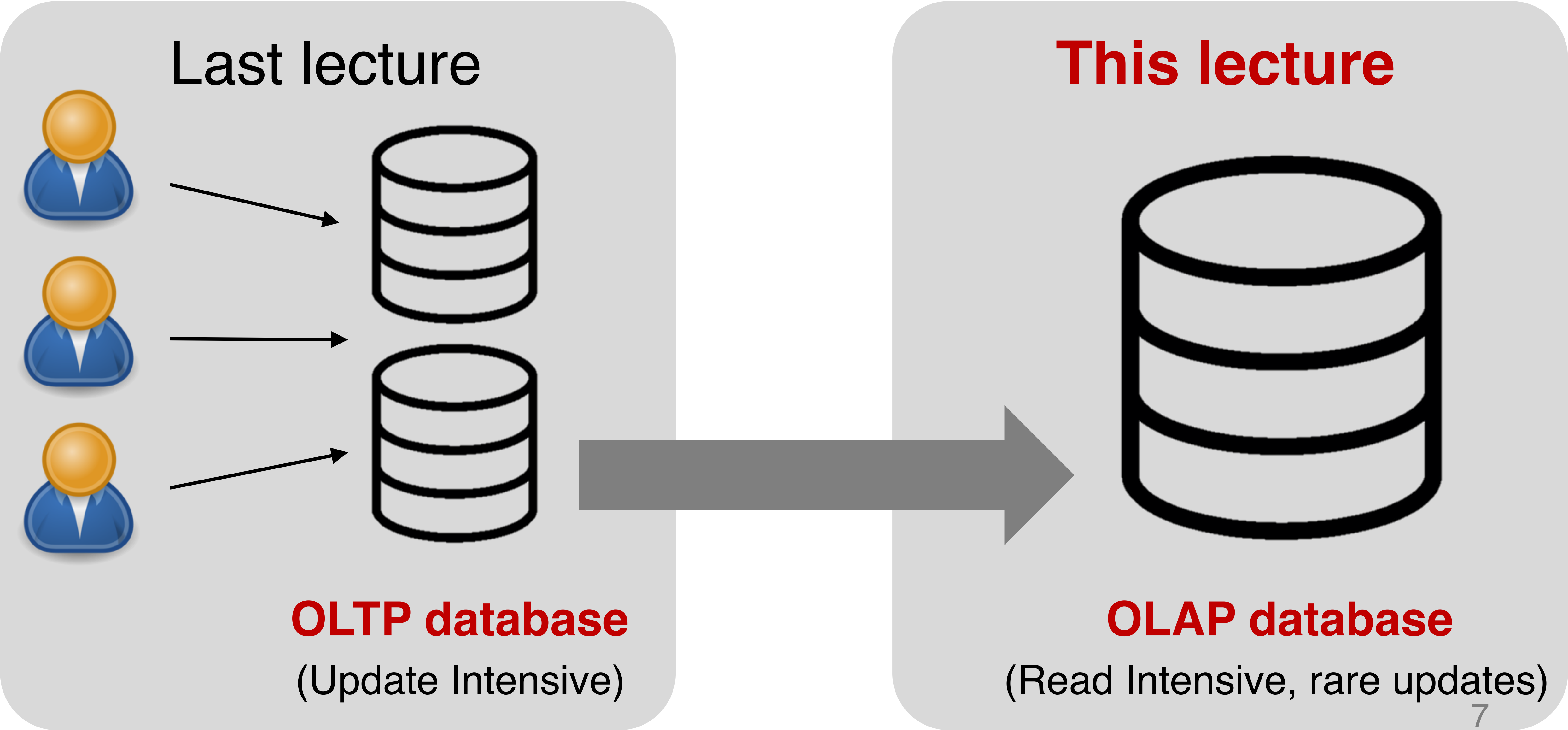
OLTP database

(Update Intensive)

OLAP database

(Read Intensive, rare updates)

OLTP vs. OLAP


Last lecture
This lecture
OLTP database
(Update Intensive)
OLAP database
(Read Intensive, rare updates)

Relation/Table

Table 1: Students

Student_ID	Name	Department_ID	Age
1	Smith	1	21
2	Bob	2	25
3	Alex	1	26

Row/Tuple

Relation/Table

Table 1: Students

Student_ID	Name	Department_ID	Age
1	Smith	1	21
2	Bob	2	25
3	Alex	1	26

Row/Tuple

Column/Attribute

Relation/Table

Table 1: Students

Student_ID	Name	Department_ID	Age
1	Smith	1	21
2	Bob	2	25
3	Alex	1	26

Primary key

Table 1: Department

Department_ID	D_name	Address
1	Computer Sciences	1210 W Dayton St
2	Math	480 Lincoln Dr

Relation/Table

Table 1: Students

Student_ID	Name	Department_ID	Age
1	Smith	1	21
2	Bob	2	25
3	Alex	1	26

Foreign key

Primary key

Table 1: Department

Department_ID	D_name	Address
1	Computer Sciences	1210 W Dayton St
2	Math	480 Lincoln Dr

Relation/Table

Table 1: Students

Student_ID	Name	Department_ID	Age
1	Smith	1	21
2	Bob	2	25
3	Alex	1	26

Foreign key

Relationship

Primary key

Table 1: Department

Department_ID	D_name	Address
1	Computer Sciences	1210 W Dayton St
2	Math	480 Lincoln Dr

Relational Algebra

Select

Project

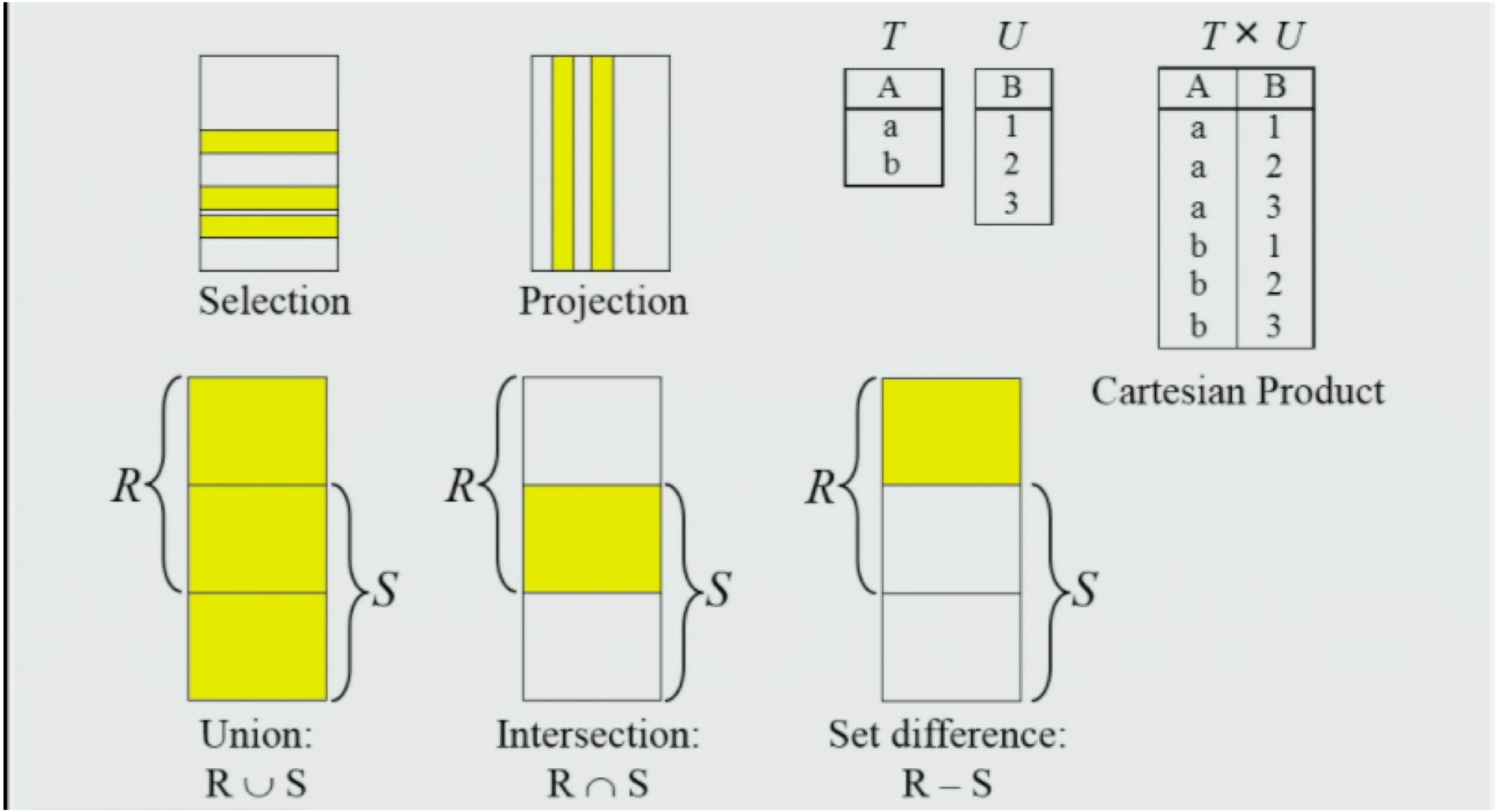
Cartesian product

Union

Set different

Rename

Relational Algebra Operations



Selection and Production Examples

Table 1: Students

Student_ID	Name	Department_ID	Age
1	Smith	1	21
2	Bob	2	25
3	Alex	1	26

1. [Selection] All information of students under 24

Selection and Production Examples

Table 1: Students

Student_ID	Name	Department_ID	Age
1	Smith	1	21
2	Bob	2	25
3	Alex	1	26

1. [Selection] All information of students under 24
2. [Projection] Names of all students in the department with Department_ID = 1

Cartesian product

Table 1: Students

Student_ID	Name	Department_ID	Age
1	Smith	1	21
2	Bob	2	25
3	Alex	1	26

x

Table 1: Department

Department_ID	D_name	Address
1	Computer Sciences	1210 W Dayton St
2	Math	480 Lincoln Dr

=

Student_ID	Name	Department_ID	Age	Department_ID	D_name	Address
1	Smith	1	21	1	Computer Sciences	1210 W Dayton St
2	Bob	2	25	1	Computer Sciences	1210 W Dayton St
3	Alex	1	26	1	Computer Sciences	1210 W Dayton St
1	Smith	1	21	2	Math	480 Lincoln Dr
2	Bob	2	25	2	Math	480 Lincoln Dr
3	Alex	1	26	2	Math	480 Lincoln Dr

Why Cartesian product?

Table 1: Students

Student_ID	Name	Department_ID	Age
1	Smith	1	21
2	Bob	2	25
3	Alex	1	26

x

Table 1: Department

Department_ID	D_name	Address
1	Computer Sciences	1210 W Dayton St
2	Math	480 Lincoln Dr

=

Student_ID	Name	Department_ID	Age	Department_ID	D_name	Address
1	Smith	1	21	1	Computer Sciences	1210 W Dayton St
2	Bob	2	25	1	Computer Sciences	1210 W Dayton St
3	Alex	1	26	1	Computer Sciences	1210 W Dayton St
1	Smith	1	21	2	Math	480 Lincoln Dr
2	Bob	2	25	2	Math	480 Lincoln Dr
3	Alex	1	26	2	Math	480 Lincoln Dr

Names of departments that contain students under 24?

Why Cartesian product?

Table 1: Students

Student_ID	Name	Department_ID	Age
1	Smith	1	21
2	Bob	2	25
3	Alex	1	26

x

Table 1: Department

Department_ID	D_name	Address
1	Computer Sciences	1210 W Dayton St
2	Math	480 Lincoln Dr

=

Student_ID	Name	Department_ID	Age	Department_ID	D_name	Address
1	Smith	1	21	1	Computer Sciences	1210 W Dayton St
2	Bob	2	25	1	Computer Sciences	1210 W Dayton St
3	Alex	1	26	1	Computer Sciences	1210 W Dayton St
1	Smith	1	21	2	Math	480 Lincoln Dr
2	Bob	2	25	2	Math	480 Lincoln Dr
3	Alex	1	26	2	Math	480 Lincoln Dr

Names of departments that contain students under 24?

Why Cartesian product?

Table 1: Students

Student_ID	Name	Department_ID	Age
1	Smith	1	21
2	Bob	2	25
3	Alex	1	26

x

Table 1: Department

Department_ID	D_name	Address
1	Computer Sciences	1210 W Dayton St
2	Math	480 Lincoln Dr

=

Student_ID	Name	Department_ID	Age	Department_ID	D_name	Address
1	Smith	1	21	1	Computer Sciences	1210 W Dayton St
2	Bob	2	25	1	Computer Sciences	1210 W Dayton St
3	Alex	1	26	1	Computer Sciences	1210 W Dayton St
1	Smith	1	21	2	Math	480 Lincoln Dr
2	Bob	2	25	2	Math	480 Lincoln Dr
3	Alex	1	26	2	Math	480 Lincoln Dr

Names of departments that contain students under 24?

Why Cartesian product?

Table 1: Students

Student_ID	Name	Department_ID	Age
1	Smith	1	21
2	Bob	2	25
3	Alex	1	26

x

Table 1: Department

Department_ID	D_name	Address
1	Computer Sciences	1210 W Dayton St
2	Math	480 Lincoln Dr

=

Student_ID	Name	Department_ID	Age	Department_ID	D_name	Address
1	Smith	1	21	1	Computer Sciences	1210 W Dayton St
2	Bob	2	25	1	Computer Sciences	1210 W Dayton St
3	Alex	1	26	1	Computer Sciences	1210 W Dayton St
1	Smith	1	21	2	Math	480 Lincoln Dr
2	Bob	2	25	2	Math	480 Lincoln Dr
3	Alex	1	26	2	Math	480 Lincoln Dr

Names of departments that contain students under 24?

Join (Natural Join)

Table 1: Students

Student_ID	Name	Department_ID	Age
1	Smith	1	21
2	Bob	2	25
3	Alex	1	26

$\bowtie$

Table 1: Department

Department_ID	D_name	Address
1	Computer Sciences	1210 W Dayton St
2	Math	480 Lincoln Dr

=

Student_ID	Name	Department_ID	Age	D_name	Address
1	Smith	1	21	Computer Sciences	1210 W Dayton St
2	Bob	2	25	Math	480 Lincoln Dr
3	Alex	1	26	Computer Sciences	1210 W Dayton St

Implementation

Storage Formats:

- Row-store
- Column-store

Operators:

- Select
- Project
- Join

Tables on Storage

Row store

1
Smith
1
21
2
Bob
2
25
3
Alex
1
26

Student_ID	Name	Department_ID	Age
1	Smith	1	21
2	Bob	2	25
3	Alex	1	26

Tables on Storage

Student_ID	Name	Department_ID	Age
1	Smith	1	21
2	Bob	2	25
3	Alex	1	26

Row store

1
Smith
1
21
2
Bob
2
25
3
Alex
1
26

Column store

1
2
3
Smith
Bob
Alex
1
2
1
21
25
26

Select (Row-Store) - Scan

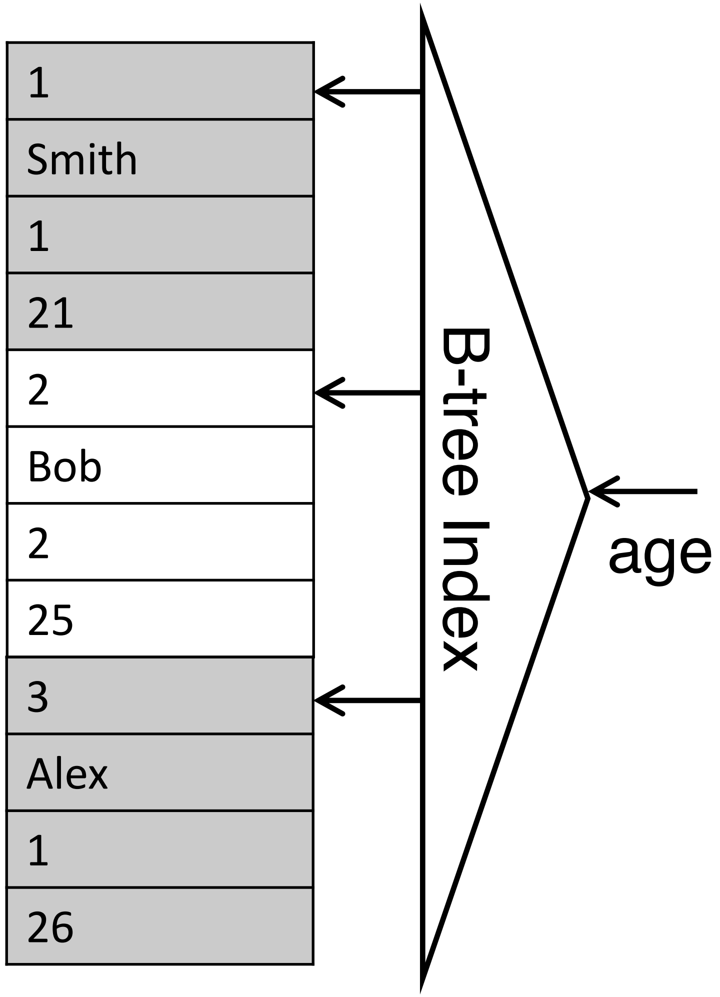
1
Smith
1
21
2
Bob
2
25
3
Alex
1
26

Student_ID	Name	Department_ID	Age
1	Smith	1	21
2	Bob	2	25
3	Alex	1	26

```
SELECT * FROM Student WHERE age < 24;
```

- Sequentially read all rows from the table
- Send the row to output if age < 24

Select (Row-Store) - Index



Student_ID	Name	Department_ID	Age
1	Smith	1	21
2	Bob	2	25
3	Alex	1	26

```
SELECT * FROM Student WHERE age < 24;
```

Indexing vs. Scan

- **Runtime**: O(output size) vs. O(input size)
- **Access pattern**: Potentially Random vs. Sequential

Project (Row-Store)

1
Smith
1
21
2
Bob
2
25
3
Alex
1
26

Student_ID	Name	Department_ID	Age
1	Smith	1	21
2	Bob	2	25
3	Alex	1	26

```
SELECT name FROM Student WHERE age < 24;
```

- Send certain columns (rather than the entire rows) to output

Join (Row-Store)

Student_ID	Name	Department_ID	Age
1	Smith	1	21
2	Bob	2	25
3	Alex	1	26

⋈

Department_ID	D_name	Address
1	Computer Sciences	1210 W Dayton St
2	Math	480 Lincoln Dr

=

Student_ID	Name	Department_ID	Age	D_name	Address
1	Smith	1	21	Computer Sciences	1210 W Dayton St
2	Bob	2	25	Math	480 Lincoln Dr
3	Alex	1	26	Computer Sciences	1210 W Dayton St

Join (Row-Store) – Nested Loop

Relation R

Student_ID	Name	Department_ID	Age
1	Smith	1	21
2	Bob	2	25
3	Alex	1	26

$\bowtie$

Relation S

Department_ID	D_name	Address
1	Computer Sciences	1210 W Dayton St
2	Math	480 Lincoln Dr

```
foreach tuple r in R
    foreach tuple s in S
        if r and s satisfy the join condition
            yield tuple <r,s>
```

runtime = |R| * |S|

Join (Row-Store) – Index Join

Relation R

Student_ID	Name	Department_ID	Age
1	Smith	1	21
2	Bob	2	25
3	Alex	1	26

$\bowtie$

Relation S

Department_ID	D_name	Address
1	Computer Sciences	1210 W Dayton St
2	Math	480 Lincoln Dr

```
foreach tuple r in R
        S' = Lookup r.joinKey in index of S
        foreach s in S'
                yield tuple <r,s>
```

The inner relation must have the index

Join (Row-Store) – Merge Sort

Relation R

Student_ID	Name	Department_ID	Age
1	Smith	1	21
2	Bob	2	25
3	Alex	1	26

$\bowtie$

Relation S

Department_ID	D_name	Address
1	Computer Sciences	1210 W Dayton St
2	Math	480 Lincoln Dr

Sort R using joinKey

Sort S using joinKey

Make one pass of R and S to join

Relations must be sorted on the join key

```
sort time = |R|log(|R|) + |S|log(|S|)
runtime = |R| + |S|
```

Join (Row-Store) – Hash Join

Relation R

Student_ID	Name	Department_ID	Age
1	Smith	1	21
2	Bob	2	25
3	Alex	1	26

$\bowtie$

Relation S

Department_ID	D_name	Address
1	Computer Sciences	1210 W Dayton St
2	Math	480 Lincoln Dr

```
Foreach r in R
      lookup the hash table of S
```

1	
NULL	NULL
2	

Hash table of S

runtime = |R| + |S|

Column-Store

Row store

1
Smith
1
21
2
Bob
2
25
3
Alex
1
26

Student_ID	Name	Department_ID	Age
1	Smith	1	21
2	Bob	2	25
3	Alex	1	26

Column store

1
2
3
Smith
Bob
Alex
1
2
1
21
25
26

Column-Store

Row store

1
Smith
1
21
2
Bob
2
25
3
Alex
1
26

Student_ID	Name	Department_ID	Age
1	Smith	1	21
2	Bob	2	25
3	Alex	1	26

Column store

1
2
3
Smith
Bob
Alex
1
2
1
21
25
26

Pros of column store:

- Great when accessing a subset of columns
- Easy to compress data

Cons of column store:

- Updates are expensive

Column-Store – Compression

Product

ID	Value
1	Beer
2	Beer
3	Vodka
4	Whiskey
5	Whiskey
6	Vodka
7	Vodka

Run length Encode

Product'

ID	Value
1-2	Beer
3	Vodka
4-5	Whiskey
6-7	Vodka

Gender
Male
Male
Female
Female
Male
Female
Male

1100101

Column-Store – Selection, Projection

Projection: Straight-forward

Selection:

SELECT name, age **WHERE** age < 25;

age < 25 => bitstring

use the bitstring as a mask to access column “name”

C-Store

Aggressive compression

Overlapping projections of tables

- **SELECT** * **WHERE** age < 24;
- **SELECT** * **WHERE** gender = ’Male’

Gender	Age
Male	21
Male	24
Male	23
Male	22
Female	23
Female	24
Female	21

Sort by gender

Gender	Age
Male	21
Male	21
Male	22
Male	23
Female	23
Female	24
Female	24

Sort by age

C-Store – Evaluation

Disk Space

C-Store	Row Store	Column Store
1.987 GB	4.480 GB	2.650 GB

Query Execution Time

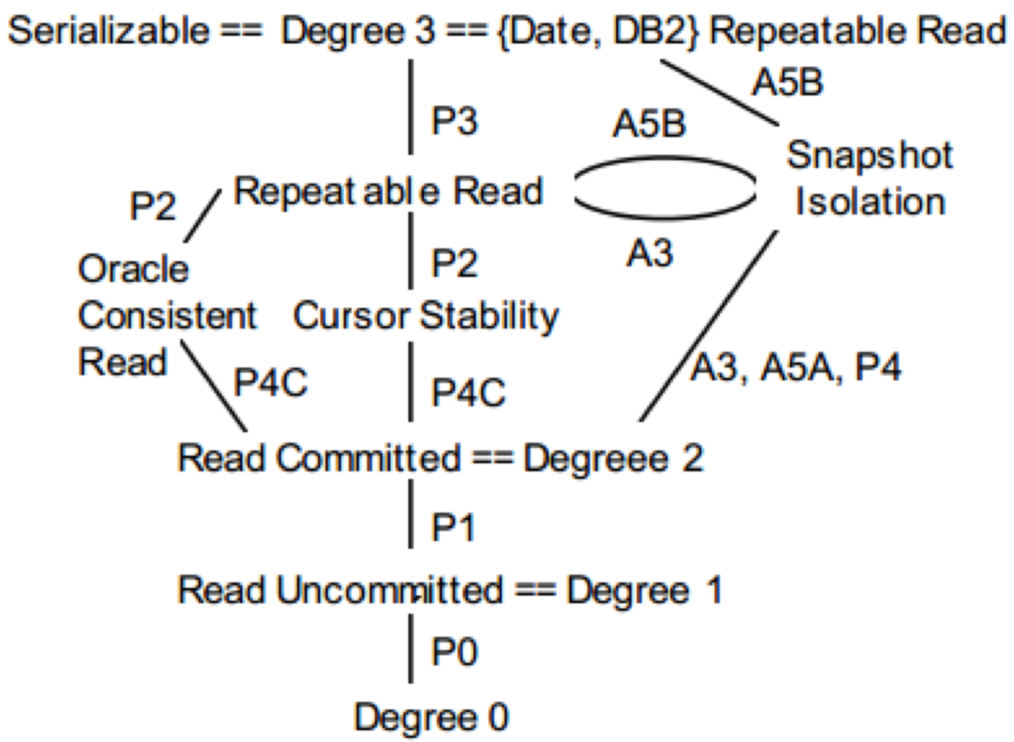
Query	C-Store	Row Store	Column Store
Q1	0.03	6.80	2.24
Q2	0.36	1.09	0.83
Q3	4.90	93.26	29.54
Q4	2.09	722.90	22.23
Q5	0.31	116.56	0.93
Q6	8.50	652.90	32.83
Q7	2.54	265.80	33.24

C-Store – Q/A

Is C-store commercially available today?

- Yes. It is called Vertica https://www.vertica.com

How does snapshot-isolation work? Isn't this a weak-isolation model?



Summary

Relation/table

Common operators: **selection, projection, join**

Implementations in row store

Column store

C-store

Group Discussion

What are the advantages and disadvantages of running transactions on a column store?

What is the right data layout for **HTAP** (Hybrid transactional/analytical processing)? Can you think of a way to combine the benefits of row-store and column-store?

If there is a small processor (weak CPU and small DRAM) sitting right next to disk, what would you use it for?

Before Next Lecture

Submit discussion summary to https://wisc-cs839-ngdb20.hotcrp.com

- One summary per group
- Authors: group members
- Any format is ok (e.g., pdf, doc, txt)
- **Deadline: Wednesday 11:59pm**

Submit review for

Staring into the Abyss: An Evaluation of Concurrency Control with One Thousand Cores

[optional] Concurrency Control Performance Modeling: Alternatives and Implications

[optional] OLTP Through the Looking Glass, and What We Found There