Query Planning & Optimization – Part I
ADMINISTRIVIA

Mid-Term Exam is Wed Oct 16\textsuperscript{th} @ 12:00pm
\rightarrow See \textcolor{red}{mid-term exam guide} for more info.

Project \#2 is due Sun Oct 20\textsuperscript{th} @ 11:59pm
QUERY OPTIMIZATION

Remember that SQL is declarative.
→ User tells the DBMS what answer they want, not how to get the answer.

There can be a big difference in performance based on plan is used:
→ See last week: 1.3 hours vs. 0.45 seconds
IBM SYSTEM R

First implementation of a query optimizer from the 1970s.
→ People argued that the DBMS could never choose a query plan better than what a human could write.

Many concepts and design decisions from the System R optimizer are still used today.
QUERY OPTIMIZATION

Heuristics / Rules
→ Rewrite the query to remove stupid / inefficient things.
→ These techniques may need to examine catalog, but they do not need to examine data.

Cost-based Search
→ Use a model to estimate the cost of executing a plan.
→ Evaluate multiple equivalent plans for a query and pick the one with the lowest cost.
ARCHITECTURE OVERVIEW

1. **Application**
   - SQL Query

2. **SQL Query**
   - **Parser**
   - **SQL Rewriter** (Optional)

3. **Abstract Syntax Tree**

4. **Logical Plan**
   - **Binder**
   - **Tree Rewriter** (Optional)

5. **Logical Plan**
   - **Optimizer**

6. **Physical Plan**
   - **Cost Model**
   - Estimates

**Diagram Details**:
- The diagram shows the flow of processing from SQL queries to physical plans.
- Each step is connected with arrows, indicating the direction of data flow.
- Key components include the system catalog, binder, and optimizer.
- Optional steps are indicated within boxes with dashed borders.
- The process starts with an SQL query and involves multiple stages of rewriting and optimization before reaching the physical plan.
LOGICAL VS. PHYSICAL PLANS

The optimizer generates a mapping of a logical algebra expression to the optimal equivalent physical algebra expression.

Physical operators define a specific execution strategy using an access path.
→ They can depend on the physical format of the data that they process (i.e., sorting, compression).
→ Not always a 1:1 mapping from logical to physical.
QUERY OPTIMIZATION IS NP-HARD

This is the hardest part of building a DBMS. If you are good at this, you will get paid $$$.

People are starting to look at employing ML to improve the accuracy and efficacy of optimizers.

I am expanding the Advanced DB Systems class to cover this topic in greater detail.
TODAY'S AGENDA

Relational Algebra Equivalences
Static Rules
RELATIONAL ALGEBRA EQUIVALENCES

Two relational algebra expressions are equivalent if they generate the same set of tuples.

The DBMS can identify better query plans without a cost model.

This is often called query rewriting.
SELECT s.name, e.cid
    FROM student AS s, enrolled AS e
    WHERE s.sid = e.sid
    AND e.grade = 'A'

π name, cid(σ grade='A' (student⋈enrolled))
SELECT s.name, e.cid
FROM student AS s, enrolled AS e
WHERE s.sid = e.sid
AND e.grade = 'A'
RELATIONAL ALGEBRA EQUIVALENCES

\[ \pi_{\text{name}, \text{cid}} (\sigma_{\text{grade} = 'A'}(\text{student} \bowtie \text{enrolled})) \]

\[ = \]

\[ \pi_{\text{name}, \text{cid}} (\text{student} \bowtie (\sigma_{\text{grade} = 'A'}(\text{enrolled}))) \]
RELATIONAL ALGEBRA EQUIVALENCES

**Selections:**
- Perform filters as early as possible.
- Reorder predicates so that the DBMS applies the most selective one first.
- Break a complex predicate, and push down
  \[ \sigma_{p_1 \land p_2 \land \ldots \land p_n}(R) = \sigma_{p_1}(\sigma_{p_2}(\ldots \sigma_{p_n}(R))) \]

Simplify a complex predicate
- \((X=Y \ AND \ Y=3) \rightarrow X=3 \ AND \ Y=3\)
RELATIONAL ALGEBRA EQUIVALENCES

Projections:

→ Perform them early to create smaller tuples and reduce intermediate results (if duplicates are eliminated)
→ Project out all attributes except the ones requested or required (e.g., joining keys)

This is not important for a column store...
SELECT s.name, e.cid 
FROM student AS s, enrolled AS e 
WHERE s.sid = e.sid 
AND e.grade = 'A'

PROJECTION PUSHDOWN
MORE EXAMPLES

Impossible / Unnecessary Predicates

```
SELECT * FROM A WHERE 1 = 0;
```

Source: Lukas Eder
Impossible / Unnecessary Predicates

\[
\text{SELECT } * \text{ FROM } A \text{ WHERE } 1 = 0;
\]

\[
\text{SELECT } * \text{ FROM } A \text{ WHERE } 1 = 1;
\]
Impossible / Unnecessary Predicates

```sql
CREATE TABLE A ( id INT PRIMARY KEY, val INT NOT NULL );
```

```sql
SELECT * FROM A WHERE 1 = 0; \[X\]
SELECT * FROM A;
```
MORE EXAMPLES

Impossible / Unnecessary Predicates

```
SELECT * FROM A WHERE 1 = 0;
```

```
SELECT * FROM A;
```

Join Elimination

```
SELECT A1.*
FROM A AS A1 JOIN A AS A2
ON A1.id = A2.id;
```
MORE EXAMPLES

Impossible / Unnecessary Predicates

```
SELECT * FROM A WHERE 1 = 0;  \[X]\n```

```
SELECT * FROM A;
```

Join Elimination

```
SELECT * FROM A;
```
MORE EXAMPLES

Ignoring Projections

```
CREATE TABLE A (
    id INT PRIMARY KEY,
    val INT NOT NULL
);

SELECT * FROM A AS A1
WHERE EXISTS(SELECT val FROM A AS A2
             WHERE A1.id = A2.id);
```
MORE EXAMPLES

Ignoring Projections

```
SELECT * FROM A;
```

Source: Lukas Eder
MORE EXAMPLES

Ignoring Projections

```sql
SELECT * FROM A;
```

Merging Predicates

```sql
SELECT * FROM A
WHERE val BETWEEN 1 AND 100
OR val BETWEEN 50 AND 150;
```
Ignoring Projections

```
SELECT * FROM A;
```

Merging Predicates

```
SELECT * FROM A
WHERE val BETWEEN 1 AND 100
  OR val BETWEEN 50 AND 150;
```
MORE EXAMPLES

Ignoring Projections

```
SELECT * FROM A;
```

Merging Predicates

```
SELECT * FROM A
WHERE val BETWEEN 1 AND 150;
```
Joins:
→ Commutative, associative
\[ R \bowtie S = S \bowtie R \]
\[ (R \bowtie S) \bowtie T = R \bowtie (S \bowtie T) \]

How many different orderings are there for an \( n \)-way join?
How many different orderings are there for an $n$-way join?

**Catalan number** $\approx 4^n$

$\rightarrow$ Exhaustive enumeration will be too slow.

We’ll see in a second how an optimizer limits the search space...
CONCLUSION

We can use static rules and heuristics to optimize a query plan without needing to understand the contents of the database.
NEXT CLASS

MID-TERM EXAM!
→ Seriously, this is not a joke.