This Lecture

How does the database turn your query into a sequence of operations?

How do the choices the database makes affect query performance?

High Level

- Parsing
- Optimization
- Execution

SQL → Relational algebra → Plan → Execution → Query results

Parsing

Parsing is the process of interpreting a string of characters as a SQL query.

Steps:
- Scanning - find SQL keywords, operators, identifiers, and literals
- Parsing - test validity of query structure, recognize identifiers as database objects or columns, etc.
- Validation - ensure query request is actually correct w.r.t. database schema
- Translation - turn the SQL into a relational algebra / query tree representation

These steps are more or less equivalent to what you would do in a programming language compiler. We won’t study them in this class.

Optimization

- Reorganize query tree for best performance (next lecture)
- Determine best techniques to enact query on the database (this lecture)

Execution

Basically, this step just implements the plan.

We won’t really be talking about this step.
Query Blocks

- Queries can get fairly complex:
  - Subqueries
  - Set operation (e.g., UNION) queries
- Optimization and planning are performed at the level of a query block: a SELECT-FROM-WHERE expression.
- Each block (e.g., each subquery) gets planned separately
  - Some blocks can be combined (unnested) into a more complex operation
  - Some operations (e.g., semi-join) not directly part of SQL

Algorithms for Selection – Simple
(Selection in the relational algebra sense)

- Simple selection: one condition
  - Linear search
  - Index lookup (B-tree)
    - Various implications if condition is equality or inequality
    - Various implications if attribute is key (unique) or non-key

Algorithms for Selection – Conjunctive
(Selection in the relational algebra sense)

- Conjunctive selection: multiple conditions combined with AND operations
  - Linear search
  - Index lookup on one condition, linear search of result set on other condition(s)
  - Multiple index lookup followed by intersection of record pointers
    - Assumes no condition is equality on a key attribute (only one record possible)
  - Composite index lookup
    - Conditions apply to attributes in composite index
    - Order matters - we can only lookup attributes as ordered in index specification

Algorithms for Selection – Disjunctive
(Selection in the relational algebra sense)

- Disjunctive selection: multiple conditions combined with OR operations
  - If any condition is on a non-indexed attribute, must use linear search
  - Otherwise (all conditions indexed):
    - Use best method to get records selected by each condition
    - Union results

Algorithms for Joins (A \( \bowtie \) B)

- Assuming two-way equijoins – other cases can be extrapolated.
  - Nested loops
    - Loop on one table
    - Loop on other table to find rows matching current row
  - Indexed based nested loop
    - Loop on one table
    - Use index scan to find matching rows in other table
  - Sort merge join:
    - Especially useful if already sorted by join attributes
    - Can apply sort (internal or external) first
  - Hash join

Projection or Set Operations
(in algebraic sense: e.g., applying DISTINCT or UNION)

- If key included, then nothing needs to be done (already unique)
- Otherwise relatively expensive – must remove duplicates
  - Sorting
  - Hashing
Next Time

- Query optimization
  - How the database internally optimizes your query
  - How you can manually "tune" your database queries