Final Project: Query Rewriter

Final Presentation

15-721: Advanced Database Systems
Newton Xie, William Zhang, Erik Sargent

Goals & Progress

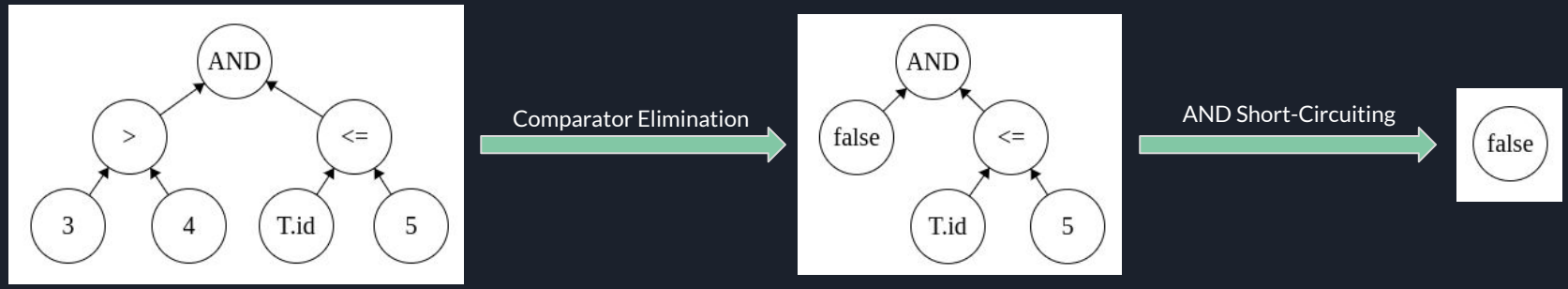
- **75% Goal:** Extendible framework and simple transformations **(DONE)**
- **100% Goal:** *Multi-level* syntax-based transformations **(DONE)**
- **125% Goal:** Catalog-based transformations **(35% DONE)**
  - **35% done:** Checking non-null columns in tables (e.g. “T.X IS NULL” ⇒ “FALSE” if X is a non-null attribute in table T)
  - **65% left:** Other catalog-based constraints (e.g. CHECK constraints -- looking up “tomato” in a column whose only allowed values are “broccoli”, “cucumber”, and “lettuce” should evaluate to FALSE)

Rules Implemented

- SPECIAL RULE: Swapping order of arguments in symmetric operators (AND, OR, =)
- Comparing two constants (=, !=, <, >, ≤, ≥) ⇒ TRUE or FALSE (depending on whether constants agree with the comparison)
- Predicate short-circuiting:
  - FALSE AND X ⇒ FALSE
  - TRUE OR X ⇒ TRUE
- Comparing an attribute against two distinct constants
  - A=3 AND A=4 ⇒ FALSE
- Transitivity of attribute-constant, attribute-attribute comparisons
  - A=3 AND B=A ⇒ A=3 AND B=3
- Checking IS NULL and IS NOT NULL on non-null columns (*Catalog-based rule!*)
  - Assume T.X is non-null, but T.Y is possibly null
  - T.X IS NULL ⇒ FALSE, T.X IS NOT NULL ⇒ TRUE
  - T.Y IS NULL and T.Y IS NOT NULL stay the same

Testing & Code Quality

- Manually-written test cases for each of our rewrite rules
- Mix tests for some combinations of rules



- No performance benchmarks at the moment

Problems Encountered & Lessons Learned

- Working in Peloton was a lot different than working in terrier for Project 1
  - A lot more dependencies with existing code, how things affect each other wasn't always super intuitive
- Leveraging the Cascades framework already used for the optimizer required touching a lot of files, and even more so to switch from templates to abstract classes
  - Also used a lot of levels of indirection and a lot of different types (Lesson: don't cover a file with *auto* declarations when your teammates need to read your code)

Future Improvements

- Migration to terrier and possible fixes
  - Pointer management
  - AbstractExpression tree equality
- Associative Transform Rule / Passing “context” through rewrite stages
- Improved group collapse / rule application logic
- Other catalog based transformations (as mentioned previously)
- Adding some more purely syntax-based functionality:
  - Reducing operator nodes on constants (e.g. 2 + 2 ⇒ 4)
  - Recursively applying rewriter to subquery expressions

Peloton Demo...

Thank you