Practicum in Database Systems

Introduction to Project 2

Project 2 Introduction

- Like Project 1, but for SQL instead of arithmetic expressions
- Overall architecture:
  - takes as input DB and SQL queries
  - processes each query and outputs answers
- Samples provided
- Essential to follow our input and output format for grading purposes

The life cycle of a query

- Parsing
  - JSqlParser
- Translation to a (bag) relational algebra query plan
- Evaluation
  - Using relations in the DB
  - Writing result to a file

About the project overall

- Start from empty directory (no skeleton code)
  - except that a .jar with JSqlParser is provided
- Relatively few hard architectural requirements, see instructions
- Will be building on your codebase for Projects 3-5 (no solution code)
- Reference implementation is ~1100 LOC
- Extensive instructions are provided

SQL

- Won't be supporting all of SQL
- Only a limited subset, see Section 2.1 in instructions
- Basically SELECT-FROM-WHERE with optional DISTINCT and/or ORDER BY
- All fields are integers

Relevant info that is/isn't in 4320

- How to evaluate operators
  - We'll come to that soon in 4320, but get started on this project now – don't wait for 4320 to catch up
  - Project instructions and textbook should be more than enough
- Iterator model for evaluating operators
  - Every operator extends an abstract Operator class
  - Provides getNextTuple() and reset() methods
  - Discussion on how these work and why we use them

How to do the project

- Suggested step-by-step process in written instructions
- Will only discuss highlights now

How to do the project

- Become familiar with JSqlParser
- Implement the scan operator
  - Will need some other classes
    - DB catalog, Tuple, etc.
- Selection
  - Will require evaluating an expression on a tuple
  - JSqlParser provides an ExpressionVisitor interface
- Projection

How to do the project

- Join
  - How to translate a query into a RA plan
  - Don't compute cross products!!
  - Consider pushing selections
- Aliases (Sailors S1, Sailors S2 etc)
- ORDER BY
- DISTINCT (using sorting)

Must-have requirements

- Use Operator model
- Build query plan and evaluate query by calling getNextTuple() on root repeatedly
- Have a method to extract join conditions from the WHERE clause