ECS 165B: Database System Implementation
Lecture 22

UC Davis

May 17, 2010

Class Agenda

- Last time:
  - DavisDB Part 4 Overview and Architectural Cookbook Session
- Today:
  - DavisDB Part 4, some clarifications/amendments
  - Data Warehousing and Decision Support
- Reading:
  - Chapter 25 of Ramakrishnan and Gehrke
    (Chapter 18 of Silberschatz et al)

Announcements

- Project Part 4 is out, due Friday 6/4 @ 11:59pm

DavisDB, Part 4 *encore une fois*

Updated Requirements for Insert, Delete, and Update

```
  insert into <relName> values (<value>, ..., <value>) ;
```

```
ReturnCode insert(const char* relName, int nValues,
                  const TypedValue values[]);
```

```
  delete from <relName>
[ where <attrName> <cmpOp> <attrOrValue> and ... and
        <attrName> <cmpOp> <attrOrValue>] ;
```

```
ReturnCode remove(const char* relName,
                  int nConditions,
                  const Condition conditions[]);
```

```
   update <relName>
   set <attrName> = <attrOrValue>
 [ where <attrName> <cmpOp> <attrOrValue> and ... and
         <attrName> <cmpOp> <attrOrValue> ] ;
```

```
ReturnCode update(const char* relName,
                  const RelationAttribute* left,
                  const AttributeOrValue* right,
                  int nConditions, const Condition conditions[]);
```

Each command should **give feedback to user by printing the inserted, deleted, or updated tuples** (using `SystemPrinter`)

Updated Requirements for Select, Delete, and Update

- **Select, delete, and update** (but not insert) **should all be implemented using *query execution plans***
- For these, plans should be **printed** iff **`queryplans = on`**
  - Generic new DavisDB shell command: `set <param> = <value> ;`
  - Retrieve value of a parameter via `SystemParser::getParam()`
  - Convenience method: `QueryEngine::isQueryPlansOn()`
  - You are free to introduce and use other sorts of parameters, e.g. for debugging
- **For grading, we must be able to understand your plans!**

~~Tip~~ Requirement: Make Operators for Delete and Update, Too!

- ~~Will again require a tweak to `IQueryOperator`, as `RecordIDs` need to be returning along with records: e.g., change~~

```
~~virtual ReturnCode getNextRecord(char* data) = 0;~~
```

~~to~~

```
virtual ReturnCode getNextRecord(Record* record) = 0;
```

- What should `DeleteOperator` or `UpdateOperator` return for `getNextRecord()`?
  - ~~Doesn't really matter... the results won't be printed~~
  - **Should return the deleted or updated tuples (final values, for update)**
- What is the `RecordID` of the result of a join?
  - Doesn't really matter... there won't be any operators above the join that care about the `RecordID` (update operations don't use joins)

Tip: Reduce, Reuse, Recycle

- For Part 2, many of you did this... (cf. Xcode)
- Instead of this... (cf. Xcode)