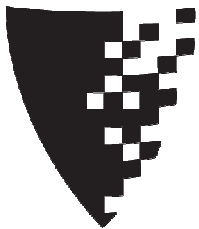


# SQL: Programming

Introduction to Databases

CompSci 316 Fall 2014



**DUKE**  
COMPUTER SCIENCE

# Announcements (Tue., Oct. 7)

- **Homework #2** due today midnight
  - Sample solution to be posted by tomorrow evening
- **Midterm** in class this Thursday
  - Open-book, open-notes
  - Same format as the sample midterm (posted on Sakai)
  - **Q&A session on sample midterm** conducted by Ben
    - Wednesday 6-8pm in Link
- **Project milestone #1** due next Thursday

# Motivation

- Pros and cons of SQL
  - Very high-level, possible to optimize
  - Not intended for general-purpose computation
- Solutions
  - Augment SQL with constructs from general-purpose programming languages
    - E.g.: SQL/PSM
  - Use SQL together with general-purpose programming languages
    - E.g.: Python DB API, JDBC, embedded SQL
  - Extend general-purpose programming languages with SQL-like constructs
    - E.g.: LINQ (Language Integrated Query for .NET)

# An “impedance mismatch”

- SQL operates on **a set of records at a time**
- Typical low-level general-purpose programming languages operate on **one record at a time**

☞ Solution: **cursor**

- **Open** (a result table): position the cursor before the first row
- **Get next**: move the cursor to the next row and return that row; raise a flag if there is no such row
- **Close**: clean up and release DBMS resources

☞ Found in virtually every database language/API

- With slightly different syntaxes

☞ Some support more positioning and movement options, modification at the current position, etc.

# Augmenting SQL: SQL/PSM

- **PSM** = **P**ersistent **S**tored **M**odules
- **CREATE PROCEDURE** *proc\_name* (*param\_decls*)  
*local\_decls*  
*proc\_body* ;
- **CREATE FUNCTION** *func\_name* (*param\_decls*)  
**RETURNS** *return\_type*  
*local\_decls*  
*func\_body* ;
- **CALL** *proc\_name* (*params*) ;
- Inside procedure body:  
**SET** *variable* = **CALL** *func\_name* (*params*) ;

[Click here to download full PDF material](#)