SQL: Augmented and Embedded SQL (Optional)

CS348: Introduction to Database Management

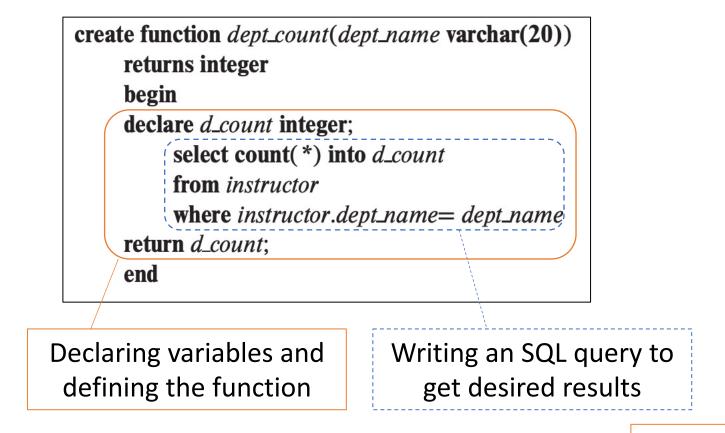
SQL programming

- Dynamic SQL (in class)
- Augmented SQL
- Embedded SQL

Augmenting SQL: functions & procedures

- Procedures and functions allow business logic to be stored in db and executed from SQL statements
- 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);

Creating function in SQL

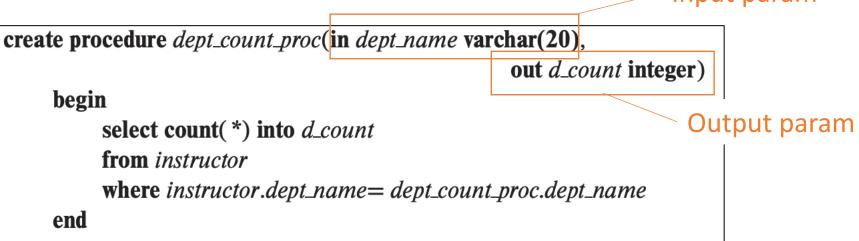


select dept_name, budget
from department
where dept_count(dept_name) > 12;

Invoking the function: returns dept. names & budgets for all depts with > 12 instructors

Creating a procedure in SQL

 Functions used to calculate something based on inputs; procedure are precompiled statements to perform some tasks in a specified order



Invoking the procedure (either from another procedure or embedded SQL)

declare d_count integer; call dept_count_proc('Physics', d_count);

Other SQL features

- Conditional constructs
 - IF, IF ELSIF ELSE
- Loop constructs
 - FOR, REPEAT UNTIL, LOOP
- Flow control
 - GOTO
- Exceptions
 - SIGNAL, RESIGNAL

• • •

Read DMBS manual for more details!

Augmenting SQL vs. API

- Pros of augmenting SQL:
 - More processing features for DBMS
 - More application logic can be pushed closer to data
- Cons of augmenting SQL:
 - SQL is already too big
 - Complicate optimization and make it impossible to guarantee safety
- Augmented SQL is not commonly used

Embedded SQL

- "Embed" SQL in a general-purpose programming language
- A language in which SQI queries are embedded is referred to as a host language
- The SQI structures permitted in the host language constitute embedded SQL
- To identify embedded SQL requests to the preporcessor, we use the "exec SQL" statements.

Embedding SQL in a language

Example in C

Declare variables to be "shared" **EXEC SQL BEGIN DECLARE SECTION;** between the application and DBMS int thisUid; float thisPop; **EXEC SQL END DECLARE SECTION;** EXEC SQL DECLARE ABCMember CURSOR FOR SELECT uid, pop FROM User WHERE uid IN (SELECT uid FROM Member WHERE gid = 'abc') **EXEC SQL OPEN ABCMember;** Specify a handler for **EXEC SQL** WHENEVER NOT FOUND DO break; **NOT FOUND exception** while (1) { **EXEC SQL** FETCH ABCMember INTO :thisUid, :thisPop; printf("uid %d: current pop is %f\n", thisUid, thisPop); printf("Enter new popularity: "); scanf("%f", &thisPop); EXEC SQL UPDATE User SET pop = :thisPop WHERE CURRENT OF ABCMember; **EXEC SQL CLOSE ABCMember;**

Embedded SQL v.s. API

- Pros of embedded SQL:
 - Be processed by a preprocessor prior to compilation → may catch SQL-related errors at preprocessing time
 - API: SQL statements are interpreted at runtime
- Cons of embedded SQL:
 - New host language code \rightarrow complicates debugging
 - Need a preprocessor s/w to parse/compile SQL related code