CS 1555

Lecture 19

**Database Programming at Large (continued)**

Accessing a database

- Open connection: Connection dbcon;, then constructor

- Close connection: dbcon.close()

Executing an SQL statement

- Execute statements without parameters

- Create statement object

- Directly execute

Cursor navigation types

- NAV\_TYPE

- TYPE\_FORWARD\_ONLY: can only be navigated forward

- SCROLL\_INSENSITIVE: can be navigated forward, backwards, and jump; concurrent database changes are not visible

- SCROLL\_SENSITIVE: can be navigated forward, backwards, and jump; concurrent database changes are visible

Cursor concurrency types

- CONC\_TYPE

- CONCUR\_READ\_ONLY: can only be read

- CONCUR\_UPDATABLE: can only be updated

PreparedStatement class

- Create and pre-compile parameterized queries using parameter markers, indicated by question marks

- Specify the values of parameters using setXXX(i, v) where XXX: SQL type including NULL; i: argument\_index; v: value

Error handling

- JDBC provides SQLException class to deal with errors

Executing transactions

- Each JDBC statement is treated as a separate transaction that is auto-committed by default

- A new transaction automatically is set after either dbcon.commit() or dbcon.rollback()

- No global transactions, transactions across many databases

- No atomicity or “all or nothing” property

Not deferred constraints

- Transaction atomicity is enforced in a flexible way by the developer

SQL injection vulnerabilities

- Allow an attacker to inject (or execute) SQL commands within an application

- Problem: accepting user input without performing adequate input validation or escaping meta-characters

Avoiding SQL injection

- In the same way, attackers can inject other SQL commands

- Solution: good programming practice; use prepareStatement()

- All queries should be parameterized

- All dynamic data should be explicitly bound to parameterized queries

- String concatenation should never be used to create dynamic SQL