TÖL303G

Gagnasafnsfræði
Database Theory

Snorri Agnarsson

SQL forritun – SQL programming

ODBC, JDBC, ESQL, …

Yfirlit/Overview

- SQL inni í öðrum forritunarmálum
  Embedded SQL
  - Innfellt (embedded) SQL (ESQL), kvikt (dynamic) SQL, SQLJ
- Gagnagrunnsforritun gegnum staðlaðar einingar:
  Database programming using standardized libraries:
  SQL/CLI (t.d./e.g. ODBC) og/and JDBC
- **Geymd stef** (föll) í gagnagrunnum (**stored procedures**)
- Samanburður á aðferðum -- Comparing methods

Grunntækni -- Fundamentals

- Gagnagrunnsforritun – Database programming
  - Ytra forritunarmál – Outer programming language
    - Java, C, C++, C#, Python, Perl, COBOL, o.s.frv.
  - Innra gagnagrunnsmál – Embedded database language
    - SQL
- SQL staðlar – SQL standards
  - Breytast í tímanna rás – Mutable in time
  - Mismunandi útfærslur eftir gagnagrunnsframleiðendum Differs between different DBMS vendors

Gagnagrunnsforritun – Database programming

- Samtalsviðmót – Interactive
  - Notandi slær inn SQL skipanir
    User enters SQL commands
- Framkvæma skipanaskrár – Execute file of commands
  - @<filename> (í sqlite: .read <filename>)
- Sérforritun og vefviðmót – Custom programming and web access
  - Getum sérforritað aðgang að gagnagrunni
    Can custom program access to a database
  - Eða notað tilbúin vefviðmót
    Or use available web interfaces

Mismunandi aðferðir í gagnagrunnsforritun
Different methods for database programming

1. Innfelldar gagnagrunnsskipanir í almennu forritunarmáli
Embedded database commands in a general purpose programming language

- Gagnagrunnsskipanirnar eru afmarkaðar með sérstökum lykilorðum
  The database commands are delimited by special keywords
- Forþýðandi les forritstextann
  A precompiler/preprocessor reads the source code
  - Ber kennsl á gagnagrunnsskipanir og breytir þeim í eitthvað jafngilt sem virkar eins og til er ætlast (köll á gagnagrunnsföll með viðeigandi viðföngum)
    Recognizes database commands and changes them into something equivalent that works as intended (calls to database library functions with appropriate parameters)
- Kallast **innfellt** SQL eða **ESQL**
  This is called **embedded** SQL or **ESQL**

Mismunandi aðferðir í gagnagrunnsforritun
Different methods for database programming

2. Notum safn eða einingu af gagnagrunnsföllum
Use a library or module with database functions

- **Safn falla** sem kalla má á úr forritunarmálinu
  A **collection of functions** callable from the programming language
- **Application Programming Interface (API)**

3. Eða búum til glænýtt forritunarmál
Or create a brand new programming language

- Gagnagrunnsforritunarmál sem hannað er frá grunni
  A database programming language designed from scratch
- Sum slík forritunarmál voru kölluð fjórðu kynslóðar forritunarmál ekki alls fyrir löngu
  Some such programming languages were recently called 4GL, fourth generation languages

Tengivandamálið – Impedance mismatch)

- Gagnagrunnslíkanið og líkan forritunarmálsins eru mismunandi, þ.e. ekki sams konar gögn
  The data models in the database and in the programming language differ, i.e. not the same types of data
- Þurfum mismunandi **bindingar** fyrir hvert ytra forritunarmál
  Need different bindings for each outer programming language
  - Bindingin tilgreinir hvernig varpað er milli gagnanna í gagnagrunninum og gagnanna í ytra forritunarmálinu
    The binding specifies how data are transformed between the database and the programming language
- Mikilvægt atriði: Trítill (ítrari, cursor, iterator) er breyta...
  Important item: A cursor/iterator is a variable…
  - sem gerir kleift að ítra yfir n-dirnar í útkomu fyrirspurnar
    that makes it possible to iterate over the tuples in a query result
  - trítill er í ytra forritunarmálinu, ekki í SQL
    the cursor is in the outer programming language, not in SQL

Dæmigerð framkvæmdaröð í gagnagrunnsforritun

1. Opnum tengingu við gagnagrunnsþjón og opnum tengingu við gagnagrunn á þjóninum
2. Höfum samskipti við gagnagrunninn með því að senda honum fyrirspurnir, uppfærslur og aðrar skipanir (og fá svör)
3. Lokum tengingum við gagnagrunninn og við þjóninn

Typical sequence of operations in database programming

1. Open a connection to the database server and open a connection to a database on the server
2. Communcate with the database by sending it queries, updates, and other commands (and receive answers)
3. Close the connections to the database and server

Innfellt SQL (ESQL), kvikt SQL, SQLJ

- Innfellt (embedded) SQL
  - Til dæmis í C og COBOL
- SQLJ er dæmi um innfellt SQL í Java
- Notum ytra forritunarmálið (host programming language) og einnig SQL sem innra (innfellt, embedded) forritunarmál

Embedded SQL (ESQL), dynamic SQL, SQLJ

- Embedded SQL
  - For example in C and COBOL
- SQLJ is an example of embedded SQL in Java
- Use the outer programming language (the host programming language) and also SQL as an embedded programming language

Innfellt SQL

- EXEC SQL
  - Forskeyti notað í forritstexta til merkja hluta af forritstextanum svo **forþýðandinn** (preprocessor) viti hvaða hluta af forritstextanum hann eigi að fikta í
  - Forþýðandinn breytir innri SQL skipununum í samsvarandi löglegan forritstexta í ytra forritunarmálinu
  - Innri SQL skipanarununni lýkur með einhverju endatákni svo sem END-EXEC (í COBOL) eða semíkommu (; - í C)
- Sameiginlegar breytur
  - Sumar breytur verða bæði notaðar í innra (SQL) og ytra forritunarmálinu
  - Merktar með forskeyti (:) í SQL skipununum

Embedded SQL

- EXEC SQL
  - A prefix used in source code to mark a part of the code so that the preprocessor knows what part of the code it should process
  - The preprocessor changes the embedded SQL commands into corresponding valid source code in the host language
  - The embedded SQL sequence of commands is ended with some token such as END-EXEC (in COBOL) or a semicolon (; in C)
- Common variables
  - Some variables are used both in the (embedded) SQL and the host programming language
  - Marked with a prefix (:) in the SQL commands

Innfellt SQL – Embedded SQL

```
0)   int loop ;
1)   EXEC SQL BEGIN DECLARE SECTION ;
2)   varchar dname [16], fname [16], lname [16], address [31] ;
3)   char ssn [10], bdate [11], sex [2], minit [2] ;
4)   float salary, raise ;
5)   int dno, dnumber ;
6)   int SQLCODE ; char SQLSTATE [6] ;
7)   EXEC SQL END DECLARE SECTION ;
```

Figure 13.1
C program variables used in the embedded SQL examples E1 and E2.

Innfellt SQL

- Tengjumst þjóni og gagnagrunni

  CONNECT TO <server name> AS <connection name>
  AUTHORIZATION <user account name and password> ;

- Skiptum um tengingu

  SET CONNECTION <connection name> ;

- Lokum tengingu

  DISCONNECT <connection name> ;

Innfellt SQL

- Breyturnar SQLCODE og SQLSTATE
  - Fá ný gildi við SQL skipanir til að gefa vísbendingar um villur og aðrar upplýsingar
- SQLCODE breytan
  - 0 = allt í fína, skipunin tókst
  - 100 = ekki meiri gögn til staðar
  - <0 = einhver villa

Embedded SQL

- The variables SQLCODE and SQLSTATE
  - Get new values during execution of SQL commands to indicate errors and other information
- The SQLCODE variable
  - 0 = All is fine, the command succeeded
  - 100 = No more data
  - <0 = Some error

Innfellt SQL

- SQLSTATE
  - Fimm stafa strengur
  - ‘00000‘ þýðir allt í fína
  - Önnur gildi gefa til kynna villur og afbrigði (exceptions)
  - T.d. þýðir ‘02000‘ að gögnin séu búin (sama og SQLCODE=100)

Embedded SQL

- SQLSTATE
  - Five character string
  - ‘00000‘ means all is fine
  - Other values indicate errors and exceptions
  - For example ‘02000‘ means that there is no more data (same as SQLCODE=100)

Innfellt SQL – Embedded SQL

```
    //Program Segment E1:
 0) loop = 1 ;
 1) while (loop) {
 2)   prompt("Enter a Social Security Number: ", ssn) ;
 3)   EXEC SQL
 4)     select Fname, Minit, Lname, Address, Salary
 5)     into :fname, :minit, :lname, :address, :salary
 6)     from EMPLOYEE where Ssn = :ssn ;
 7)   if (SQLCODE == 0) printf(fname, minit, lname, address, salary)
 8)     else printf("Social Security Number does not exist: ", ssn) ;
 9)   prompt("More Social Security Numbers (enter 1 for Yes, 0 for No): ", loop) ;
10)   }
```

Figure 13.2
Program segment E1, a C program segment with embedded SQL.

Takið eftir línu 5, þetta er **ekki SQL** en er löglegt **ESQL**

Notice line 5, this is **not SQL** but is valid **ESQL**

Innfellt SQL með trítlum og mörgum niðurstöðum

- Trítill (cursor)
  - Vísar á eina röð (n-d) í útkomu fyrirspurnar
- OPEN CURSOR skipun
  - Setur af stað fyrirspurn og staðsetur trítil fyrir framan fyrstu röð niðurstöðu
- FETCH skipanir
  - Færa trítil yfir á næstu röð niðurstöðu

Embedded SQL with cursors and multiple results

- A cursor
  - Refers to one row (tuple) in the result of a query
- OPEN CURSOR command
  - Starts a query and locates the cursor in front of the first row of the result set
- FETCH commands
  - Move the cursor to the next row of results

Figure 13.3
Program segment E2, a C program segment that uses cursors with embedded SQL for update purposes.

```
     //Program Segment E2:
 0)  prompt("Enter the Department Name: ", dname) ;
 1)  EXEC SQL
 2)    select Dnumber into :dnumber
 3)    from DEPARTMENT where Dname = :dname ;
 4)  EXEC SQL DECLARE EMP CURSOR FOR
 5)    select Ssn, Fname, Minit, Lname, Salary
 6)    from EMPLOYEE where Dno = :dnumber
 7)    FOR UPDATE OF Salary ;
 8)  EXEC SQL OPEN EMP ;
 9)  EXEC SQL FETCH from EMP into :ssn, :fname, :minit, :lname, :salary ;
10)  while (SQLCODE == 0) {
11)    printf("Employee name is:", Fname, Minit, Lname) ;
12)    prompt("Enter the raise amount: ", raise) ;
13)    EXEC SQL
14)      update EMPLOYEE
15)      set Salary = Salary + :raise
16)      where CURRENT OF EMP ;
17)    EXEC SQL FETCH from EMP into :ssn, :fname, :minit, :lname, :salary ;
18)    }
19)  EXEC SQL CLOSE EMP ;
```

Kvikt SQL

- Kvikt SQL (dynamic SQL)
  - Getum búið til SQL skipanir á keyrslutíma (eða fyrr, auðvitað) og keyrt þær
- Kvikar uppfærslur
  - Öflugt en stórhættulegt!
- Kvikar fyrirspurnir (skárra)

```
    //Program Segment E3:
0)  EXEC SQL BEGIN DECLARE SECTION ;
1)  varchar sqlupdatestring [256] ;
2)  EXEC SQL END DECLARE SECTION ;
    ...
3)  prompt("Enter the Update Command: ", sqlupdatestring) ;
4)  EXEC SQL PREPARE sqlcommand FROM :sqlupdatestring ;
5)  EXEC SQL EXECUTE sqlcommand ;
    ...
```

Figure 13.4
Program segment E3, a C program segment that uses dynamic SQL for updating a table.

Dynamic SQL

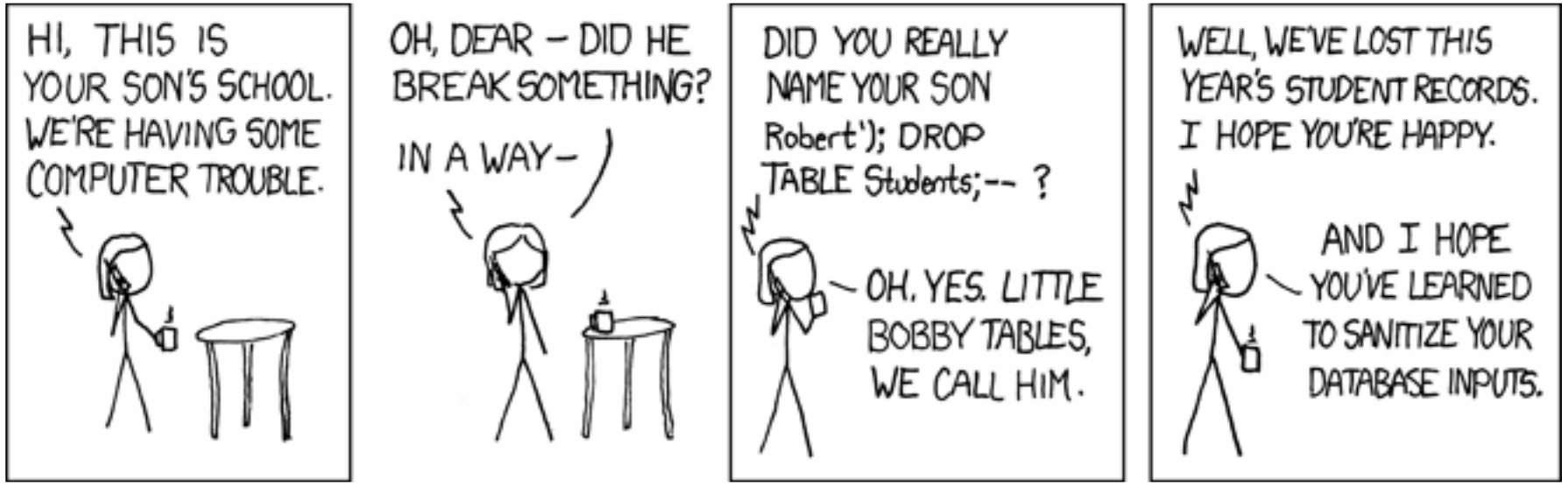
- Dynamic SQL
  - We can construct SQL commands during run time (or earlier, of course) and execute them
- Dynamic updates
  - Powerful but extremely dangerous!

```
    //Program Segment E3:
0)  EXEC SQL BEGIN DECLARE SECTION ;
1)  varchar sqlupdatestring [256] ;
2)  EXEC SQL END DECLARE SECTION ;
    ...
3)  prompt("Enter the Update Command: ", sqlupdatestring) ;
4)  EXEC SQL PREPARE sqlcommand FROM :sqlupdatestring ;
5)  EXEC SQL EXECUTE sqlcommand ;
    ...
```

Figure 13.4
Program segment E3, a C program segment that uses dynamic SQL for updating a table.

Breytingar á gagnagrunni með kviku SQL

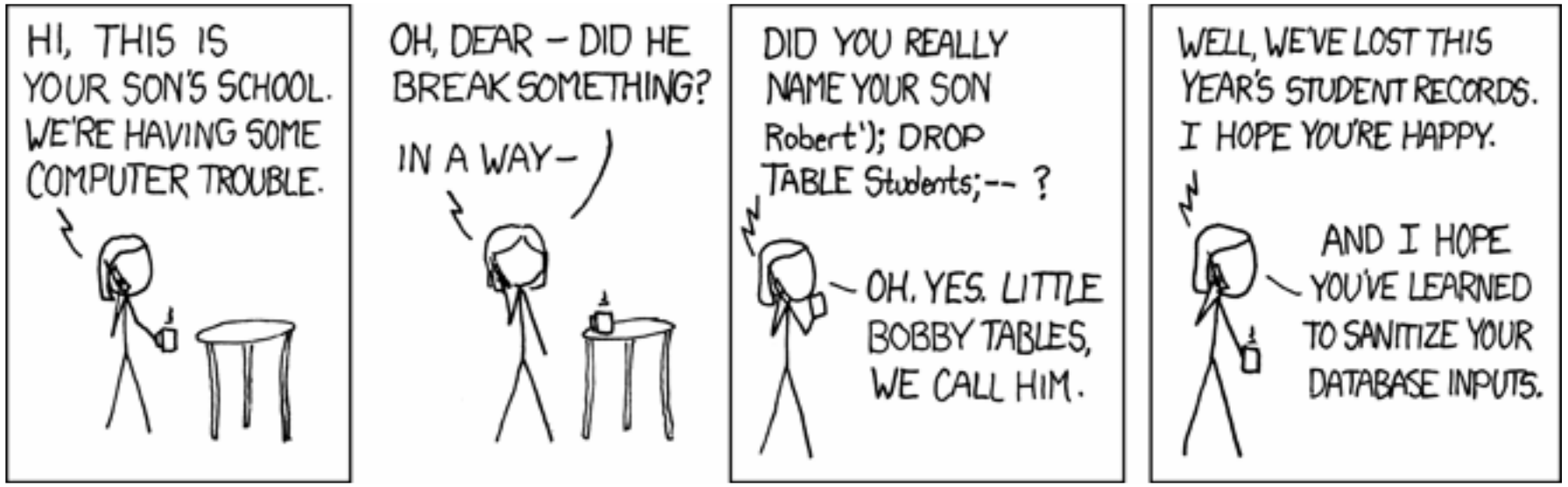
- Hendum burt töflu og öllum gögnum hennar með DROP TABLE



- http://xkcd.com/327/
- Sérstök SQLite skipun (oft þægileg, en óþægileg ef hún kemur utan frá í kviku SQL): DROP TABLE t IF EXISTS;

Changing a database with dynamic SQL

- Throw away a table and all its contents with DROP TABLE



- http://xkcd.com/327/
- A special SQLite command (often convenient, but dangerous if it arrives from external data in dynamic SQL): DROP TABLE t IF EXISTS;

SQLJ: Innfellt SQL í Java

- Sameiginlegur staðall nokkurra framleiðenda fyrir innfellt SQL í Java

```
    //Program Segment J1:
 1) ssn = readEntry("Enter a Social Security Number: ") ;
 2) try {
 3)    #sql { select Fname, Minit, Lname, Address, Salary
 4)       into :fname, :minit, :lname, :address, :salary
 5)       from EMPLOYEE where Ssn = :ssn} ;
 6) } catch (SQLException se) {
 7)       System.out.println("Social Security Number does not exist: " + ssn) ;
 8)       Return ;
 9)    }
10) System.out.println(fname + " " + minit + " " + lname + " " + address
      + " " + salary)
```

Figure 13.7
Program segment J1, a Java program segment with SQLJ.

SQLJ: Embedded SQL in Java

- A common standard from multiple vendors for embedded SQL in Java

```
    //Program Segment J1:
 1) ssn = readEntry("Enter a Social Security Number: ") ;
 2) try {
 3)    #sql { select Fname, Minit, Lname, Address, Salary
 4)       into :fname, :minit, :lname, :address, :salary
 5)       from EMPLOYEE where Ssn = :ssn} ;
 6) } catch (SQLException se) {
 7)       System.out.println("Social Security Number does not exist: " + ssn) ;
 8)       Return ;
 9)    }
10) System.out.println(fname + " " + minit + " " + lname + " " + address
      + " " + salary)
```

Figure 13.7
Program segment J1, a Java program segment with SQLJ.

Trítlar í SQLJ – Cursors in SQLJ

Figure 13.9
Program segment J2B, a Java program segment that uses a positional iterator to print employee information in a particular department.

```
    //Program Segment J2B:
 0) dname = readEntry("Enter the Department Name: ") ;
 1) try {
 2)   #sql { select Dnumber into :dnumber
 3)     from DEPARTMENT where Dname = :dname} ;
 4) } catch (SQLException se) {
 5)   System.out.println("Department does not exist: " + dname) ;
 6)   Return ;
 7)   }
 8) System.out.printline("Employee information for Department: " + dname) ;
 9) #sql iterator Emppos(String, String, String, String, double) ;
10) Emppos e = null ;
11) #sql e = { select ssn, fname, minit, lname, salary
12)   from EMPLOYEE where Dno = :dnumber} ;
13) #sql { fetch :e into :ssn, :fn, :mi, :ln, :sal} ;
14) while (!e.endFetch()) {
15)   System.out.printline(ssn + " " + fn + " " + mi + " " + ln + " " + sal) ;
16)   #sql { fetch :e into :ssn, :fn, :mi, :ln, :sal} ;
17) } ;
18) e.close() ;
```

Gagnagrunnsforritun með köllum: SQL/CLI (call level interface) og JDBC

- Köllum á gagnagrunnsföll
  - Kvik aðferð til gagnagrunnsforritunar
- Safn af gagnagrunnsföllum
  - Á ensku kallast almennt skilgreining slíks safns **application programming interface (API)**
    - Oft er API einnig notað (frekar óformlega) til að merkja safnið sjálft, ekki aðeins **skilgreiningu** þess
  - Köllin gera aðgerðir á gagnagrunninn
- SQL Call Level Interface (SQL/CLI)
  - Hluti af SQL staðlinum

Database programming with library calls: SQL/CLI (call level interface) and JDBC

- Call library database functions
  - A dynamic method for database programming
- A library of database functions
  - The definition of such a library is called an **application programming interface (API)**
    - Often the term API is also used (rather informally) to denote the library itself, not just its **definition**
  - The calls execute operations on the database
- SQL Call Level Interface (SQL/CLI)
  - Part of the SQL standard

SQL/CLI í C

- Gagnahlutir í SQL/CLI
  - Environment færsla (record)
    - Inniheldur sameiginlegar upplýsingar og gögn um safn af gagnagrunnstengingum
  - Connection færsla
    - Inniheldur upplýsingar um eina tiltekna gagnagrunnstengingu
  - Statement færsla
    - Inniheldur upplýsingar um eina tiltekna SQL skipun
  - Description færsla
    - Inniheldur upplýsingar um n-dir eða viðföng
  - Handföng (Handle) vísa á færslur
    - C bendar (pointer) eru (yfirleitt) notaðir sem handföng til að vísa á færslur

SQL/CLI in C

- Data types in SQL/CLI
  - Environment record
    - Contains common information and data about a collection of database connections
  - Connection record
    - Contains information about a single database connection
  - Statement record
    - Contains information about a single SQL command
  - Description record
    - Contains information about tuples or arguments
  - A Handle is used to refer to these records
    - C pointers are (usually) used as handles to refer to records

Figure 13.11
Program segment CLI2, a C program segment that uses SQL/CLI for a query with a collection of tuples in its result.

```
    //Program Segment CLI2:
 0) #include sqlcli.h ;
 1) void printDepartmentEmps() {
 2) SQLHSTMT stmt1 ;
 3) SQLHDBC con1 ;
 4) SQLHENV env1 ;
 5) SQLRETURN ret1, ret2, ret3, ret4 ;
 6) ret1 = SQLAllocHandle(SQL_HANDLE_ENV, SQL_NULL_HANDLE, &env1) ;
 7) if (!ret1) ret2 = SQLAllocHandle(SQL_HANDLE_DBC, env1, &con1) else exit ;
 8) if (!ret2) ret3 = SQLConnect(con1, "dbs", SQL_NTS, "js", SQL_NTS, "xyz",
      SQL_NTS) else exit ;
 9) if (!ret3) ret4 = SQLAllocHandle(SQL_HANDLE_STMT, con1, &stmt1) else exit ;
10) SQLPrepare(stmt1, "select Lname, Salary from EMPLOYEE where Dno = ?",
      SQL_NTS) ;
11) prompt("Enter the Department Number: ", dno) ;
12) SQLBindParameter(stmt1, 1, SQL_INTEGER, &dno, 4, &fetchlen1) ;
13) ret1 = SQLExecute(stmt1) ;
14) if (!ret1) {
15)   SQLBindCol(stmt1, 1, SQL_CHAR, &lname, 15, &fetchlen1) ;
16)   SQLBindCol(stmt1, 2, SQL_FLOAT, &salary, 4, &fetchlen2) ;
17)   ret2 = SQLFetch(stmt1) ;
18)   while (!ret2) {
19)     printf(lname, salary) ;
20)     ret2 = SQLFetch(stmt1) ;
21)     }
22)   }
23) }
```

JDBC: SQL föll í Java

- JDBC
  - Java klasasafn
- Sama Java forritið getur tengst mörgum mismunandi gagnagrunnum
  - Stundum kallað gagnalindir (data source)
- Hlaða þarf JDBC rekli (driver)

```
Class.forName("oracle.jdbc.driver.OracleDriver");
```

JDBC: SQL functions in Java

- JDBC
  - Java class library
- The same Java program can connect to many different databases
  - Sometimes called data sources
- A JDBC driver needs to be loaded

```
Class.forName("oracle.jdbc.driver.OracleDriver");
```

Figure 13.12
Program segment JDBC1, a Java program segment with JDBC.

```
    //Program JDBC1:
 0) import java.io.* ;
 1) import java.sql.*
    ...
 2) class getEmpInfo {
 3)   public static void main (String args []) throws SQLException, IOException {
 4)     try { Class.forName("oracle.jdbc.driver.OracleDriver")
 5)     } catch (ClassNotFoundException x) {
 6)       System.out.println ("Driver could not be loaded") ;
 7)     }
 8)     String dbacct, passwrd, ssn, lname ;
 9)     Double salary ;
10)     dbacct = readentry("Enter database account:") ;
11)     passwrd = readentry("Enter password:") ;
12)     Connection conn = DriverManager.getConnection
13)       ("jdbc:oracle:oci8:" + dbacct + "/" + passwrd) ;
14)     String stmt1 = "select Lname, Salary from EMPLOYEE where Ssn = ?" ;
15)     PreparedStatement p = conn.prepareStatement(stmt1) ;
16)     ssn = readentry("Enter a Social Security Number: ") ;
17)     p.clearParameters() ;
18)     p.setString(1, ssn) ;
19)     ResultSet r = p.executeQuery() ;
20)     while (r.next()) {
21)       lname = r.getString(1) ;
22)       salary = r.getDouble(2) ;
23)       system.out.printline(lname + salary) ;
24)   } }
25) }
```

JDBC – SQL klasasafn í Java

- **Driver** klasi
- **Connection** klasi
- **Statement** klasi hefur tvo undirklasa
  - **PreparedStatement** og **CallableStatement**
- Spurningarmerki (?)
  - Stendur fyrir viðfang (eða frjálsa breytu) í SQL setningu/skipun
  - Gildi viðfangs verður tilgreint á keyrslutíma
- **ResultSet** klasi
  - ResultSet hlutur inniheldur niðurstöðu fyrirspurnar
  - Sambærilegt við trítil (cursor) í ESQL

JDBC – SQL class library in Java

- **Driver** class
- **Connection** class
- **Statement** class has two subclasses
  - **PreparedStatement** og **CallableStatement**
- A question mark (?)
  - Stands for an argument (or free variable) in an SQL command
  - The value of the argument will be specified at run time
- **ResultSet** class
  - A ResultSet object contains the result of a query
  - Comparable to a cursor in ESQL

Geymd gagnagrunnsstef (Stored Procedures)

- Stef geymd í gagnagrunni á gagnagrunnsþjóni
- PL/pgSQL í PostgreSQL, SQL/PSM í Oracle, ...
- Geta skilað gildi eða ekki
- Viðbætur við SQL
- Hægt að forrita á mjög almennan hátt

https://en.wikipedia.org/wiki/PL/SQL

Stored Procedures

- Procedures stored in a database on a database server
- PL/pgSQL in PostgreSQL, SQL/PSM í Oracle, ...
- May or may not return values
- Additions to SQL
- Can be programmed in a very general fashion

https://en.wikipedia.org/wiki/PL/SQL

Samanburður aðferða

- Innfellt SQL (ESQL)
  - Þægilegt
  - Möguleiki (stundum) á að sannreyna SQL skipanirnar gagnvart gagnagrunni á þýðingartíma
  - En ef smíða þarf flóknar fyrirspurnir á keyrslutíma þá getur vel verið þægilegra að nota SQL/CLI
- SQL/CLI (þ.m.t. ODBC, JDBC)
  - Sveigjanlegra
  - Yfirleitt alltaf til staðar í öllum algengum forritunarmálum
  - Flóknara í notkun
  - Ekki sannreynt í þýðingu
- Gagnagrunnsforritunarmál
  - Ekki vandamál með tög í gagnagrunni og ytra forritunarmáli
  - Forritarar þurfa að læra nýtt forritunarmál

Comparing methods

- Embedded SQL (ESQL)
  - Convenient
  - Possible (sometimes) to validate the SQL commands against a database during compilation
  - But if complicated queries need to be constructed at run time it may well be more convenient to use SQL/CLI
- SQL/CLI (including ODBC, JDBC)
  - More flexible
  - Almost always available in all common programming languages
  - More complicated in use
  - Not verified during compilation
- Stored procedures
  - No mismatch problem with the database types and host programming language types
  - Programmers need to learn a new programming language