

Zugriff auf die Datenbank



Ulf Leser
Wissensmanagement in der
Bioinformatik



Wie komme ich an die Datenbank?

- Alternativen
 - Alles in Java machen
 - JDBC Treiber von Oracle herunterladen
 - Java programmieren
 - Java-basierte GUI verwenden (z.B. Aquafold)
 - JDBC Treiber von Oracle herunterladen
 - Anwendung installieren
 - Tool benutzen, die den Oracle Client verwenden (z.B. SQL*PLUS, Toad, TORA, ...)
 - Oracle Client herunterladen (Oracle Technology Network)
 - Ca. 650 MB
 - Installieren
 - Dann Tool installieren

Verbindungsdaten

- Server: MANGOLD.INFORMATIK.HU-BERLIN.DE
- Port: 1521
- Datenbankinstanz: DWH07
- Typischer „Connectstring“
 - MANGOLD.INFORMATIK.HU-BERLIN.DE:1521:DWH07
- Wer den lokalen Oracle Client installiert
 - Datei \oracle\product...\network\admin\tnsnames.ora
 - Eintrag dazu (Klammern beachten!):
 - DWH =
(DESCRIPTION =
 (ADDRESS = (PROTOCOL = TCP)(HOST =
mangold.informatik.hu-berlin.de)(PORT = 1521))
 (CONNECT_DATA =
 (SERVER = DEDICATED)
 (SERVICE_NAME = dwh07)
)
)
- Danach kann man die Verbindung als DWH ansprechen

JDBC

-Java Database Connectivity-

Ulf Leser

Wissensmanagement in der
Bioinformatik

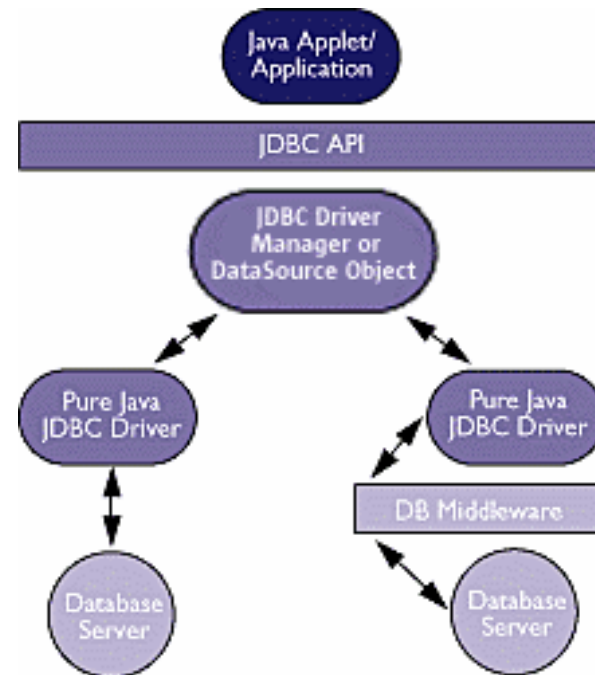


JDBC auf einer Folie

```
Connection con = DriverManager.getConnection (
    "jdbc:myDriver:myserver",
    "myLogin", "myPassword");
Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery(
    "SELECT a, b, c FROM table1");
while (rs.next()) {
    int x = rs.getInt("a");
    String s = rs.getString("b");
    float f = rs.getFloat("c");
}
rs.close();
stmt.close();
con.close();
```

JDBC Basics

- Zugriff von Java-Programmen auf SQL-basierte RDBMS
- Generische API für eine Vielzahl von Systemen
 - Inklusive JDBC-ODBC Bridge
 - Treiber sind in unterschiedlichen Stufen verfügbar
 - Auch unterschiedliche Implementierungen
- In Version 4 verfügbar, Teil von Enterprise Java
 - Packages **java.sql** und **javax.sql**



Wichtigste Konzepte

- DB-spezifischer JDBC-Treiber: **DriverManager**
- Verbindung zur DB: Interface **Connection**
- Anfragen formulieren: **Statement**
- Ergebnisse auswerten: **ResultSet**
- Metadaten über die DB

Verbindung zu Oracle

- Im Classpath: `classes12.jar`
 - Siehe Quellen
- Dann
- `DriverManager.registerDriver(
 new oracle.jdbc.OracleDriver());
server = „mangold.informatik.hu-
berlin.de:1521:DWH07“;
Connection con =
DriverManager.getConnection(
 „jdbc:oracle:thin:@“+server,
 "gruppenname",
 "kennwort");`

Statements

- Sind gewöhnliche SQL-Statements, die JDBC an die DB weiterleitet
- erzeugt wird ein Statement über das Interface **Connection**

```
Statement stmt =  
    con.createStatement( );
```

Beispiele

- `stmt.execute(`
 `"CREATE TABLE coworkers(`
 `c_id int,`
 `name varchar(25))"`
`);`
`stmt.execute(`
 `"INSERT INTO coworkers`
 `VALUES (1, 'Herbert')"`
`);`
`stmt.close();`

kein ';' am Ende des Statements

Anfragen

- Ausführung von **Statement.executeQuery()**;
- Erzeugt ein Objekt mit Interface `ResultSet`

```
ResultSet result =  
    stmt.executeQuery(  
        "SELECT c_id, name FROM  
        coworkers ORDER BY c_id")  
    );
```

- `ResultSet` speichert einen internen „aktuellen“ Pointer auf die Ergebnisliste
 - Buffergröße!
- Ergebnisse werden tupelweise durchlaufen über **result.next()**;
- Bereits das erste Tupel bedarf des vorherigen Aufrufs von **result.next()**;
- Aufruf liefert **true**, wenn weitere Tupel vorhanden; sonst **false**

Zugriff auf Werte

- Über die Methoden
`ResultSet.getXXX("<attrib>")`
- also `getString()`, `getInt()`, ...

```
stmt.executeQuery("SELECT c_id,name ..  
while(result.next()) {  
    int c_id = result.getInt("c_id");  
    String name = result.getString("name");
```

- Man muss also den Datentypen der Spalte kennen
 - Zugriff möglich über Metadaten

Prepared Statements

- Normale SELECT Statements werden in der DB jedes Mal neu geparkt, optimiert und kompiliert
- Bei mehrfacher Ausführung eines Statements mit nur geänderten Parametern ist das ineffektiv
- Besser: Anstelle von Statement

```
PreparedStatement pstmt =
    con.prepareStatement(
        "INSERT INTO coworkers
        (c_id, name) VALUES (?, ?)");
{LOOP}
    // prepare tuples:
    pstmt.setInt(1, anInt);
    // string as 2nd attribute:
    pstmt.setString(2, aString);
    // execute prepared statement
    pstmt.execute();
{POOL}
pstmt.close();
```

Exceptions

- JDBC Methoden können SQLException werfen
 - Fangen!
- Beispiele
 - "Geschlossene Anweisung"
 - Wo? `Statement.execute()`
 - Was? Es wird auf ein Instanz von Statement zugegriffen, die zuvor mit `Statement.close()` geschlossen wurde
 - "ORA-00001: Verstoß gegen Eindeutigkeit"
 - Wo? `Statement.execute("INSERT ..");`
 - Was? Einfügen bereits vorhandener Werte in eine als `unique/primary key` deklarierte Spalte

Ein paar weitere Dinge

- Aufruf von PL/SQL Routinen über CallableStatements
- CachedConnections
- CachedResultSets
 - SELECT ohne Datenbankverbindung
- Scrollable ResultSets
- Updateable ResultSets
- Batch Updates
- Mapping von Datentypen (BLOB, structured records, Table functions, arrays, date, ...)
- ...

Quellen

- <http://www.jdbc-tutorial.com/>
- JDBC API Tutorial and Reference. (Java Series) von Jonathan Bruce, Jon Ellis, und Maydene Fisher. Addison Wesley (2004)
- Datenbanken und Java von Gunter Saake und Kai-Uwe Sattler. Dpunkt Verlag (2003)
- http://www.oracle.com/technology/software/tech/java/sqlj_jdbc/index.html
- <http://java.sun.com/javase/technologies/database/index.jsp>
- <http://java.sun.com/developer/Books/JDBCTutorial/index.html>

PL/SQL

- Kurze Einführung -



Ulf Leser
Wissensmanagement in der
Bioinformatik



PL/SQL

- Oracle-eigene Programmiersprache mit enger Integration in SQL
- Sämtliche prozeduralen Konzepte sind verfügbar: Funktionen/ Prozeduren, Abfragen/bedingte Ausführung, Schleifen, etc.
- Wird verwendet für SQL Funktionen, Trigger, Stored-Procedures, etc.
- Code wird in der Datenbank, uU während der Ausführung einer SQL-Anfrage ausgeführt
 - Keine Benutzerschnittstellen
 - Keine Ausgabe auf Bildschirm etc.
 - Logging ist gar nicht so einfach (Transaktionen)
- Alternative: Java in der Datenbank
 - Oder SQLJ

PL/SQL-Programme

- Grundaufbau

```
DECLARE
.. /* Variablen-&Typdeklarationen */
BEGIN
.. /* Programmcode, Funktionen .. */
EXCEPTION
.. /* Exception-Handling */
END;
```

SQL in PL/SQL

- Erlaubte SQL-Statements:
 - DML: `select`, `insert`, `delete`, `update`
- Nicht (direkt) erlaubt
 - DDL: `create`, `drop`, `alter`, etc.
 - Dazu verwendet man Dynamic SQL
- Einfaches Beispiel

```
DECLARE
```

```
  cnt number;
```

```
BEGIN
```

```
  SELECT count(*) INTO cnt FROM USER_TABLES;
```

```
  cnt := cnt*10;
```

```
  INSERT INTO mytabelle VALUES (cnt, 'constant');
```

```
END;
```

Variablen und Datentypen

- Alle von Oracle unterstützen Typen + NUMBER, BOOLEAN
- Ermittlung zur Compilezeit möglich

```
DECLARE
```

```
    name          VARCHAR(20);
```

```
    gruppe        NUMBER;
```

```
    punktezahl    user_tables.table_name%TYPE;
```

Kontrollflüsse

- Bedingungen
 - IF THEN ...
ELSIF <B2> THEN ...
ELSE ... END IF;
- Schleifen
 - LOOP EXIT WHEN ... END LOOP;
 - WHILE LOOP ... END LOOP;
 - FOR <V> IN <C1>..<C2> LOOP
... END LOOP;
- Ausnahmen (Blöcke können geschachtelt werden)
 - BEGIN
...
EXCEPTION
...
END;

Cursor

- Meistens iteriert man über alle Tupel einer Anfrage
- Dazu verwendet man Cursor

```
DECLARE
  name user_tables.table_name%type;
  b user_tables.blocks%type;
  CURSOR Gruppencursor IS
    SELECT table_name,blocks
    FROM user_tables;
BEGIN
  OPEN Gruppencursor;
  LOOP
    FETCH Gruppencursor INTO name, b;
    EXIT WHEN Gruppencursor%NOTFOUND;
    dbms_output.put_line( ,Gelesen:` || name || b);
  END LOOP;
  CLOSE Gruppencursor;
END;
```

Kompilieren

- Folgenden Befehl auf einer SQL Konsole ausführen
 - CREATE OR REPLACE PROCEDURE myproc IS
... // Variablen
BEGIN
...
end;
- Entweder geht es gut, oder es gibt Fehler
 - SHOW ERRORS;
- Ausführen
 - EXEC myproc;
- Einfacher: IDE nehmen
 - Z.B. Aqua Data Studio: My Schema -> proecures -> Rechte Maustaste
 - Anlegen, Kompilieren, Ändern, DBMS_OUTPUT ansehen, ...

EXPLAIN PLAN

- Erste Schritte -



Ulf Leser
Wissensmanagement in der
Bioinformatik



EXPLAIN PLAN

- EXPLAIN PLAN
`SELECT * FROM user_tables;`
- Parst und optimiert die Query, ohne sie auszuführen
- Speichert die Abfolge sämtlicher gewählter Operationen in einer Tabelle ab
 - Referenzierte Tabellen
 - Verwendete Zugriffsmethoden
 - Join-Methode
 - Geschätzte Kosten
 - ...

Ausgabe

- Der Execution plan als Ergebnis der Operation liegt in der (vorher anzulegenden) Tabelle `plan_table`
- ```
CREATE TABLE plan_table (
 statement_id varchar(30),
 operation varchar(30),
 cost numeric ..);
```
- Script ist auf der Webseite

# Anzeigen eines Plans

- Verschiedene Skripte gibt's im Web
- Am einfachsten: `SELECT * FROM PLAN_TABLE`
- Empfehlung: Graphische Tools benutzen

**Selected Operation**

| Property        | Value            |
|-----------------|------------------|
| Operation       | SELECT STATEMENT |
| Node Cost       | 464.0            |
| IO Cost         | 451              |
| CPU Cost        | 91232567         |
| Cardinality     | 1624             |
| Bytes           | 930552           |
| Position        | 464              |
| Obj Owner       |                  |
| Obj Name        |                  |
| Options         |                  |
| Obj Type        |                  |
| Optimizer       | ALL_ROWS         |
| Obj Instance    |                  |
| Remarks         |                  |
| Obj Node        |                  |
| Search Columns  |                  |
| Other Tag       |                  |
| Partition Start |                  |
| Partition Stop  |                  |
| Partition Id    |                  |
| Other           |                  |
| Distribution    |                  |
| Temp Space      |                  |

# Statistiken

---

- Der Optimierer benötigt aktuelle Statistiken über die Tabellen
  - Wenn nur eine fehlt – regelbasierte Optimierer
- Die erneuert er nicht automatisch, sondern das muss man selber triggern
  - `analyze table <table_name> compute statistics`
  - `exec`  
`dbms_utility.analyze_schema('DWH_TEST', 'COMPUTE')`
  - Seit Oracle 9i empfohlen:
  - `DBMS_STATS.GATHER_TABLE_STATS('DWH_TEST', 'TEST');`