course:
Database Systems (NDBI025)
SS2011/12

lecture 5:

SQL – embedded SQL, external applications, SQL/XML

doc. RNDr. Tomáš Skopal, Ph.D.
doc. RNDr. Irena Mlýnková, Ph.D.
RNDr. Michal Kopecký, Ph.D.

Department of Software Engineering, Faculty of Mathematics and Physics, Charles University in Prague

Today's lecture outline

- embedded SQL – „internal" database applications
  - stored procedures
  - cursors
  - triggers
- SQL/XML
  - XML data type + functions integrated into SQL SELECT
- „external" database applications (using interfaces/libraries)
  - ODBC, JDBC, ADO.NET
  - object-relational mapping libraries
    - Java Hibernate

Programming in embedded SQL

- procedural extensions of SQL
  - std. SQL is a subset (i.e., that's why embedded)
  - MS SQL Server – Transact SQL (T-SQL)
  - Oracle – PL/SQL
- benefits
  - controlling statements (if-then, for, while)
    - cannot be just scripted
  - cursors (iterative scan of tables)
  - smaller networking overhead (code on server), pre-compiled
  - triggers – general integrity constraints
  - better security (access rights control for server code)
- cons
  - proprietary extensions, cannot be simply transferred
  - SQL 1999 standard, but not respected by industry much

Structure (MS SQL Server)

DECLARE section
BEGIN … **END**

E.g.:

```
DECLARE @avg_age FLOAT
BEGIN
SELECT @avg_age = AVG(age) FROM Employee
END
```

Stored procedures (MS SQL Server)

CREATE PROCEDURE *procname* [; *number*]
[*declaration_parameter* [, ...]]
[WITH RECOMPILE]
AS commands [;]

- parameter declaration
  - *@name type* [= *expression*] [OUT[PUT]]
    - OUT[PUT] parameter is output
- *number* allows multiple versions of the same procedure
- procedure call
  - EXEC[UTE] procname [expression [, ...]]
    - parameters are passed w.r.t. order
  - EXEC[UTE] procname [@name=expression [, ...]]
    - parameters are passed w.r.t. names

Stored procedures, example

```
CREATE PROCEDURE Payment
    @accSource VARCHAR(25),
    @accTarget VARCHAR(25),
    @amount INTEGER = 0
  AS
  BEGIN
      UPDATE Accounts SET balance = balance - @amount
       WHERE account=@accSource;

      UPDATE Accounts SET balance = balance + @amount
       WHERE account=@accTarget;
  END

EXEC Payment '21-87526287/0300', '78-9876287/0800', 25000;
```

Stored procedures (MS SQL Server)

CREATE FUNCTION *funcname* [; *number*]
([*declaration_parameter* [, ...]]) **RETURNS** type
[WITH RECOMPILE]
AS commands [;]

- parameter declaration
  - *@name type* [= *expression*] [OUT[PUT]]
    - OUT[PUT] parameter is output
- *number* allows multiple versions of the same procedure
- procedure call
  - funcname([expression [, ...]])
    - parameters are passed w.r.t. order
  - funcname(procname [@name=expression [, ...]])
    - parameters are passed w.r.t. names

Stored procedures, example

```
CREATE FUNCTION AccBalance(
    @acc VARCHAR(25)
    ) RETURNS INTEGER
  AS
  DECLARE @ret INTEGER;
  BEGIN
      SELECT @ret =balance
       FROM Accounts
       WHERE account=@acc;

      RETURN @acc;
  END

SELECT AccBalance(account) AS bal FROM …
```

Cursors (MS SQL Server)

- declaration
  - C [SCROLL] **CURSOR FOR**
    SELECT …;
- data retrieval
  - **FETCH**
    {NEXT | PRIOR | ABSOLUTE n | RELATIVE n | LAST | FIRST}
    FROM C
    [INTO @variable [, …]]
  - If cursor not declared using SCROLL, only NEXT is allowed

Cursors, example (tax payment)

```
DECLARE
    Cur CURSOR FOR
        SELECT *
        FROM Accounts;
    BEGIN
        OPEN Cur
        DECLARE @acc int, @bal int;
        FETCH NEXT FROM Cur INTO @acc, @bal;
        WHILE @@FETCH_STATUS=0
        BEGIN
          Payment(@acc, '21-87526287/0300', @bal*0.01)
          FETCH NEXT FROM Cur INTO @acc, @bal;
        END;
        CLOSE Cur;
        DEALLOCATE Cur;
    END
```

Triggers – DML triggers

- event-executed stored procedure (table/view event)
- allows to extend integrity constraint logics
  - ***inserted, deleted*** – logical tables with the same structure as the table the trigger is bound on

CREATE TRIGGER trigger_name **ON** { table | view }
[WITH ENCRYPTION]
{ FOR | AFTER | INSTEAD OF }
{ [INSERT] [,] [UPDATE] [,] [DELETE] }
[WITH APPEND]
AS
[{IF UPDATE (column) [{ AND | OR } UPDATE (column)] ...
| IF (COLUMNS_UPDATED (bitwise_operator updated_bitmask
)}]
sql_statement [...]

DML triggers (example)

```
CREATE TRIGGER LowCredit ON Purchasing.PurchaseOrderHeader
AFTER INSERT
AS
DECLARE @error_count int

SELECT @error_count = count (*)
    FROM inserted i JOIN Purchasing.Vendor v on v.VendorID = i.VendorID
    WHERE v.CreditRating=5

IF @error_count > 0
BEGIN
    RAISERROR ('Vendor''s credit rating is too low to accept new purchase orders.', 16, 1)
    ROLLBACK TRANSACTION
END
```

DDL triggers

CREATE TRIGGER trigger_name **ON** { ALL SERVER | DATABASE }
[WITH <ddl_trigger_option> [,...n]]
{ FOR | AFTER } { event_type | event_group } [,...n] **AS**
{sql_statement [;] [,...n] | EXTERNAL NAME < method specifier > [;] }
<ddl_trigger_option> ::= [ENCRYPTION] [EXECUTE AS Clause]
    <method_specifier> ::= assembly_name.class_name.method_name

DDL triggers (example)

```
CREATE TRIGGER safety ON DATABASE FOR
   DROP_SYNONYM
AS
   RAISERROR ('You must disable Trigger "safety" to drop
   synonyms!',10, 1) ROLLBACK
```

What is SQL/XML

- An extension of SQL for XML data (SQL 2003)
  - New built-in data type called XML
  - Querying over XML data
- Note: SQL/XML ≠ SQLXML
  - SQLXML is Microsoft technology in MS SQL Server (not standard)
  - Similar strategy, but different approach
  - Not a standard
- The key aspect is an XML value
  - Intuitively: an XML element or a set of XML elements
  - Its semantics results from XML Infoset data model
    - Standard of XML data model = tree consisting of nodes representing elements, attributes, text values, ...
    - graph formalism for XML

XML Data Publishing

- Generating of XML data (of type XML) from relational data
  - XMLELEMENT – creating an XML element
  - XMLATTRIBUTES – creating XML attributes
  - XMLFOREST – creating a sequence of XML elements
  - XMLCONCAT – concatenation of XML values into a single value
  - XMLAGG – creating a sequence of elements from a group of rows

XMLELEMENT

```
Employees (id, first, surname, dept, start)
```

- Creating an XML value for:
  - Name of an element
  - (Optional) list of attribute declarations
  - (Optional) list of expressions declaring element content

```
SELECT E.id,
  XMLELEMENT ( NAME "emp",
    E.first || ' ' || E.surname ) AS xvalue
FROM Employees E WHERE ...
```

id	xvalue
1001	<emp>George Clooney</emp>
...	...

XMLELEMENT – subelements

```
SELECT E.id,
  XMLELEMENT ( NAME "emp",
    XMLELEMENT ( NAME "name",
      E.first || ' ' || E.surname ),
    XMLELEMENT ( NAME "date", E.start )
  ) AS xvalue
FROM Employees E WHERE ...
```

id	xvalue
1001	<emp> <name>George Clooney</name> <date>2000-05-24</date> </emp>
...	...

XMLATTRIBUTES

```
SELECT E.id,
  XMLELEMENT ( NAME "emp",
    XMLATTRIBUTES ( E.id AS "empid" ),
    E.first || ' ' || E.surname ) AS xvalue
FROM Employees E WHERE ...
```

id	xvalue
1001	<emp empid="1001">George Clooney</emp>
...	...

XMLFOREST

```
SELECT E.id,
  XMLELEMENT ( NAME "emp",
    XMLFOREST (
      E.first || ' ' || E.last AS "name",
      E.start AS "date" ) ) AS xvalue
FROM Employees E
WHERE ...
```

id	xvalue
1001	<emp> <name>George Clooney</name> <date>2000-05-24</date> </emp>
...	...

XMLAGG

- XMLAGG is an aggregation function combined with GROUP BY
  - Similarly to SUM, AVG etc.
- XMLAGG accepts only XML expressions (values)
- The expression is evaluated for each row in group G created by the GROUP BY expression
- The results are concatenated into a single resulting value
- The result can be sorted using ORDER BY

XMLAGG

```
SELECT XMLELEMENT (
  NAME "dept",
  XMLATTRIBUTES (E.dept AS "name"),
  XMLAGG (
    XMLELEMENT (NAME "emp", E.surname)
    ORDER BY E.surname )
  ) AS xvalue
FROM Employees E
GROUP BY E.dept
```

xvalue
<dept name="hr"> <emp>Clooney</emp> <emp>Pitt</emp> </dept>
<dept name="accountant"> ... </dept>
...

XML Type

- Type XML can be used at same places as SQL data types (e.g., NUMBER, VARCHAR, …)
  - Column type, SQL variable, …
- The XML type can be
  - Queried (XMLQUERY)
  - Transformed into relational data (XMLTABLE)
  - Tested (XMLEXISTS)

Sample Data – Table EmpXML

id	ColEmpXML
1001	<emp> <first>George</first> <surname>Clooney</surname> <date>2000-05-24</date> <dept>hr</dept> </emp>
1006	<emp> <first>Brad</first> <surname>Pitt</surname> <date>2001-04-23</date> <dept>hr</dept> </emp>
…	…

XMLQUERY

```
SELECT
  XMLQUERY (                                  XQuery query
    'for $p in $col/emp return $p/surname',
    PASSING EmpXML.ColEmpXML AS "col"
    RETURNING CONTENT NULL ON EMPTY ) AS result
FROM EmpXML WHERE …
```

result
<surname>Clooney</surname>
<surname>Pitt</surname>
…

XMLTABLE

```
SELECT result.*
  FROM EmpXML, XMLTABLE (
    'for $p in $col/emp return $p',
    PASSING EmpXML.ColEmpXML AS "col"
    COLUMNS firstname VARCHAR(40) PATH 'first'
                      DEFAULT 'unknown',
            lastname  VARCHAR(40) PATH 'surname'
  ) AS TableResult
```

XQuery query

XPath expressions over XML data returned by the query

TableResult

firstname	lastname
George	Clooney
Brad	Pitt
unknown	Banderas

Assumption: We do not know first name of Banderas.

XMLEXISTS

```
SELECT id
FROM EmpXML
WHERE
  XMLEXISTS ( '/emp/date lt "2001-04-23"'
    PASSING BY VALUE EmpXML.ColEmpXML )
```

XPath query

id
1001
1006
…

Database applications

- DBI026
  - Oracle and MS SQL Server (alternatives)
    - embedded SQL, administration
    - external applications
    - indexing, optimisations
    - transactions
    - security
  - see http://www.ms.mff.cuni.cz/~kopecky/vyuka/dbapl/

External database programming

- external/standalone applications (i.e., outside DBMS environment) use standardized interfaces
  - ODBC (Open DataBase Connectivity)
    - 1992, Microsoft
  - JDBC (Java DataBase Connectivity)
    - using ODBC (mostly), or native driver/protocol, network driver
  - ADO.NET library (Active Data Objects .NET)
    - over OLE DB, ODBC, or directly drivers to MS SQL Server, Oracle
    - higher-level, faster and more reliable (than ODBC)
- „seminative" database object oriented programming using object-relational mapping
  - Java Hibernate
  - the same for Microsoft .NET

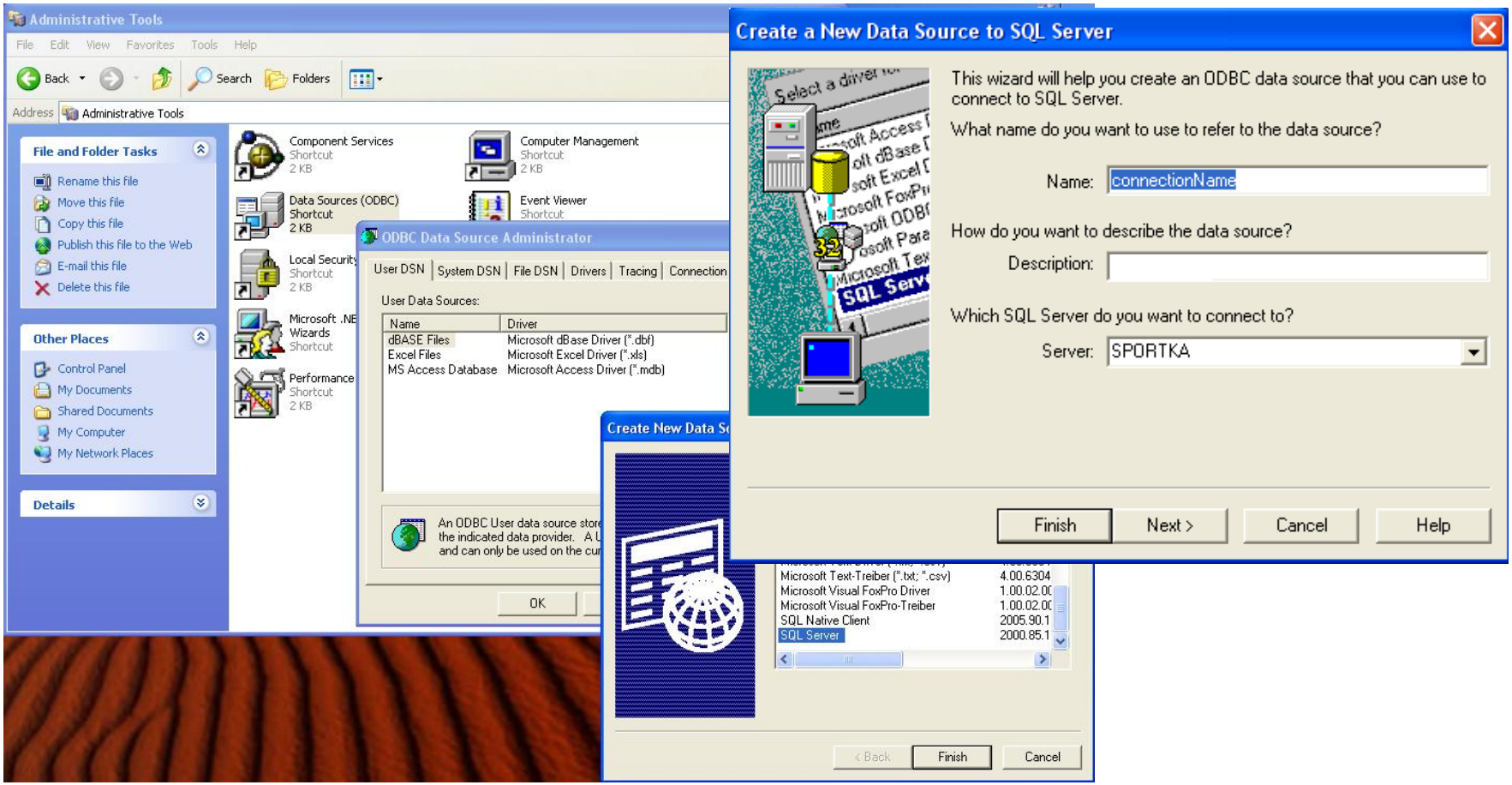
Java Hibernate Framework

- provides persistence to Java objects, i.e., provides „real" object oriented database programming
- mapping definition between object and its persistent state in database necessary (xml file for every class)
- simplification: memory manager organizes objects directly in database (+ uses main memory as a cache when accessing object)
- HQL (Hibernate query language)
  - object query language
  - Hibernate translates HQL into SQL

Relational vs. object approach

- object-to-tables mapping brings some overhead, not visible to the user (is both good and bad)
  - implementation of object DBMS using relational DBMS
- relational DBMS suitable for data-intensive applications and batch nature (uniform actions over many instances)
  - here object DBMS would be inefficient due to creation of many small objects that are created uniformly, i.e., individual „materialization" into general objects is not necessary
- object DBMS suitable for „Enterprise applications", where DBMS performance is not the bottleneck
  - relational DBMS provide low-level access to data, i.e., programmer forced to manage the database

ODBC, Windows configuration



ODBC, application (C#)

```
using System.Data.Odbc;
OdbcConnection DbConnection = new OdbcConnection("DRIVER={SQL
    Server};SERVER=MyServer;Trusted_connection=yes;DATABASE=northwind; ");
DbConnection.Open();
OdbcCommand DbCommand = DbConnection.CreateCommand();
DbCommand.CommandText = "SELECT * FROM Employee";
OdbcDataReader DbReader = DbCommand.ExecuteReader();

int fCount = DbReader.FieldCount;

while( DbReader.Read()) {
    Console.Write( ":" );
    for (int i = 0; i < fCount; i++) {
        String col = DbReader.GetString(i); Console.Write(col + ":");
    }
    Console.WriteLine();
}

DbReader.Close(); DbCommand.Dispose(); DbConnection.Close();
```

JDBC, application (Java)

```
Class.forName( "com.somejdbcvendor.TheirJdbcDriver" );

Connection conn = DriverManager.getConnection( "jdbc:somejdbcvendor:other data needed by
    some jdbc vendor", "myLogin", "myPassword" );

Statement stmt = conn.createStatement();

try {
    stmt.executeUpdate( "INSERT INTO MyTable( name ) VALUES ( 'my name' ) " );
    }
finally { stmt.close(); }
```

ADO.NET, aplikace (C#)

using ODBC:

```
SqlConnection connection = new
    SqlConnection("server=localhost;database=myDatabase;uid=sa;pwd=");
```

using OLE DB (sqloledb = SQL Server, msdaora = Oracle):

```
OleDbConnection connection = new OleDbConnection
      ("provider=sqloledb;server=localhost;database="+"myDatabase;uid=sa;pwd=");

connection.Open();

SqlCommand command = new SqlCommand("SELECT * FROM table", connection);
command.ExecuteNonQuery();
```

Java Hibernate – example

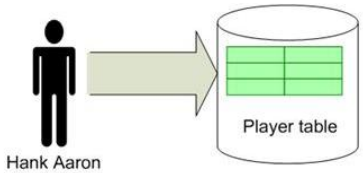
```
public class BallPlayer {
 private Long id;
 private String name;
 private String nickname;
 private Calendar dob;
 private String birthCity;
 private short uniformNumber;
 //getter and setter methods removed here for brevity.
}
```

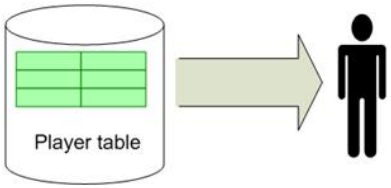
BallPlayer
id
name
nickname
dob
birthCity
uniformNumber

```
create table player (
 id integer primary key,
 name varchar not null,
 nickname varchar,
 date_of_birth date,
 city_of_birth varchar,
 uniform_number integer
);
```

Player	
PK	id
	name nickname date_of_birth city_of_birth uniform_number

```
Calendar dob = Calendar.getInstance();
dob.set(1934, Calendar.FEBRUARY, 5);
BallPlayer player = new BallPlayer();
player.setName("Henry Louis Aaron");
player.setNickname("Hammerin' Hank");
player.setDob(dob);
player.setBirthCity("Mobile, AL");
player.setUniformNumber((short) 44);
```






```
SessionFactory sessionFactory = new
Configuration().configure().  buildSessionFactory();
Session session = sessionFactory.openSession();
Transaction transaction = session.beginTransaction();
session.save(player);
transaction.commit();
session.close();
```

```
SessionFactory sessionFactory = new
Configuration().configure().buildSessionFactory();
Session session = sessionFactory.openSession();
Transaction transaction = session.beginTransaction();
BallPlayer aPlayer = (BallPlayer) session.get(BallPlayer.class,
transaction.commit();
session.close();
```

```
INSERT INTO PLAYER VALUES(5,'Henry Louis Aaron','Hammerin''
Hank','1934-02-05','Mobile, AL',44)
```

```
select ballplayer0_.id as id0_0_, ballplayer0_.name as name0_0_,
ballplayer0_.nickname as nickname0_0_, ballplayer0_.date_of_birth as
date4_0_0_, ballplayer0_.city_of_birth as city5_0_0_,
ballplayer0_.uniform_number as uniform6_0_0_ from Player ballplayer0_
where ballplayer0_.id=1
```

Java Hibernate – example

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-configuration
 PUBLIC "-//Hibernate/Hibernate Configuration DTD//EN"
 "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
 <session-factory>
   <property name="dialect"> org.hibernate.dialect.HSQLDialect</property>
   <property name="connection.driver_class"> org.hsqldb.jdbcDriver</property>
   <property name="connection.url"> jdbc:hsqldb:hsql:
//localhost/baseballdb</property>
   <property name="connection.username">sa</property>
   <property name="connection.password"></property>
   <property name="show_sql">true</property>
   <mapping resource="com/intertech/domain/BallPlayer.hbm.xml" />
 </session-factory>
</hibernate-configuration>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
   "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
   "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
       <class name="com.intertech.domain.BallPlayer" table="Player">
               <id name="id">
                       <generator class="sequence">
                               <param name="sequence">common_seq</param>
                       </generator>
               </id>
               <property name="name" />
               <property name="nickname" />
               <property name="dob" column="date_of_birth" />
               <property name="birthCity" column="city_of_birth" />
               <property name="uniformNumber" column="uniform_number" />
       </class>
</hibernate-mapping>
```

Equivalents of Java Hibernate for Microsoft.NET Framework

- ADO.NET Entity Framework
- NHibernate
- Persistor.NET
- etc.