Database and information systems

Michal Krátký & Radim Bača

Department of Computer Science
Faculty of Electrical Engineering and Computer Science
VŠB – Technical University of Ostrava

2020/2021

Content

Contact information

- Michal Krátký
- tel.: +420 597 326 090
- room: EA434
- e-mail: michal.kratky@vsb.cz
- Web of database courses: `http://dbedu.cs.vsb.cz/`
  login / password are the same as the LDAP login and password

Hardware and software available

1. Servers:
   - **dbsys.cs.vsb.cz**: 2 $\times$ Intel Xeon E5 2690 2.9GHz 12C, 288GB RAM.
   - **bayer.cs.vsb.cz**: 2 $\times$ Intel Xeon X5670 2.93GHz 6C, 512GB RAM
   - RAID Disk Array (12TB)
2. Oracle 18c R2 x64 Enterprise Edition, SQL Server 2016.
3. Oracle SQL Developer[1], Microsoft Management Studio

[1] `https://www.oracle.com/database/technologies/appdev/sql-developer.html`

Database and Information Systems

1. PL/SQL (Oracle)
2. T-SQL (SQL Server)
3. Recovery management, transactions, ACID
4. JDBC, ADO.NET
5. Object-relational mapping
6. Physical database design
7. Object-relational data model

Classification of practices

- The course is finished by a written examination.
  The minimum **30 points from 55p**.

- The maximal classification from the practices is 45 points:
  1. One real-time test: PL/SQL.
     The minimum: **8 points from 15p**.
  2. Development of an information system implemented in the ASP.NET platform with a special attention to the database layer.
     The minimum: **16 points from 30p**.

References

1. H. Garcia-Molina, J.D. Ullman, J. Widom: Database systems: the complete book. Prentice Hall, 2002.
2. C.J. Date: An Introduction to Database Systems. Addison Wesley, 2003.
3. Oracle 18c documentation: `https://docs.oracle.com/en/database/oracle/oracle-database/18/books.html`

PL/SQL

1. The PL/SQL language[2] represents a procedural extension of SQL.
2. PL/SQL syntax is based on the ADA language.
3. Similar procedural extensions have been also released for other relational database systems: T-SQL for Sybase and MS SQL Server, PL/pgSQL for PostgreSQL and SQL PL for IBM DB2.

[2] `https://docs.oracle.com/en/database/oracle/oracle-database/18/books.html` – PL/SQL Language Reference

PL/SQL Features

Advantages:

- A combination of procedural logic and SQL.
- An application is stored in a DBMS:
  - Lower amount of data is transferred since only the final result is returned through a connection.
  - Code can be shared among applications.
  - Platform independent.

Disadvantages:

- Bad portability of the code among various database systems.
- **Do we need the portability?**

Basic structure of PL/SQL block

1. `DECLARE` – optional, a declaration of local variables and cursors.
2. `BEGIN` – required, the start of PL/SQL commands.
3. `EXCEPTION` – optional, exception handling.
4. `END` – required, PL/SQL block end.

Example, Transactions in PL/SQL I

```
CREATE TABLE Person(
   login char(5) PRIMARY KEY,
   email VARCHAR(20) NOT NULL,
   password VARCHAR(15) NOT NULL,
   fname VARCHAR(15) NOT NULL,
   mname VARCHAR(15),
   lname VARCHAR(15) NOT NULL,
   street VARCHAR(30),
   city VARCHAR(30));

CREATE TABLE Role (
   idRole INT PRIMARY KEY,
   role VARCHAR(30) NOT NULL);

CREATE TABLE PersonRole (
   login CHAR(5) REFERENCES Person,
   idRole INT REFERENCES Role,
   PRIMARY KEY(login, idRole));

INSERT INTO Role VALUES(1, 'Author');
COMMIT;
```

Example, Transactions in PL/SQL II

Transaction: A block of database operations which is executed completely or not at all.

We want to insert a person and in the same time assign a role with id 1 to this person. The transaction is the following:

```
BEGIN
  INSERT INTO Person VALUES('sob28', 'jan.sobota@vsb.cz',
    'password', 'Jan', NULL, 'Sobota', NULL, NULL);
  INSERT INTO PersonRole VALUES('sob28', 1);
  COMMIT;
EXCEPTION
  WHEN OTHERS THEN
    ROLLBACK;
END;
```

If any insertion fails then both operations are cancelled.

SET AUTOCOMMIT ON/OFF

- After `SET AUTOCOMMIT ON` is any SQL command automatically committed. In other words, COMMIT and ROLLBACK commands are useless.
- After `SET AUTOCOMMIT OFF` is the automatic commit switched off and every transaction starts when the previous ends.

Comments

```
BEGIN
  -- one row comment
  INSERT INTO Person VALUES('sob28', 'jan.sobota@vsb.cz',
    'heslo', 'Jan', NULL, 'Sobota', NULL, NULL);
  /*
   * multi-row comment
   */
  INSERT INTO PersonRole VALUES('sob28', 1);
  COMMIT;
EXCEPTION
  WHEN OTHERS THEN
    ROLLBACK;
END;
```

Variables 1/3

We can define local variables in the section DECLARE:

```
varible_name varible_type [NO NULL := value];
```

- variable_name is the name of the variable. We often use the v_ prefix in the name.
- variable_type is a data type of the variable. We can use the CREATE TABLE data types.
- value is an optional part which initializes the value of the variable.

Variables 2/3

- Variables can be used to store the temporary values.
- There are two basic ways how to store a value in the variable:

Syntax	Example
`variable_name := value`	`v_age := 20`

`SELECT column`
`  INTO variable_name`
`  FROM table_name`
	`SELECT age INTO v_age`
	`  FROM student`
	`  WHERE login LIKE 'bon007'`

- SELECT has to return exactly one record otherwise the exception `NO_DATA_FOUND` or `TOO_MANY_ROWS` is raised.

Variables 3/3

- We can use standard arithmetic operators when working with numbers: +,-,*,/,...
- We can use the || operator for a string concatenation and standard SQL functions (`TO_CHAR`, `TO_DATE`, `SUBSTR`, `LENGTH`, and so on).

Variable, Example

```
DECLARE
   v_fname VARCHAR2(20);
   v_lname VARCHAR2(20);
   v_email VARCHAR2(60);
BEGIN
   SELECT fname, lname INTO v_fname, v_lname
          FROM student WHERE login = 'bon007';

   v_email := v_fname || '.' || v_lname || '@vsb.cz';

   UPDATE student set email = v_email
          WHERE login = 'bon007';
END;
```

Operator %TYPE

- Data types often correspond to the table attributes data types.
- We have to change the code when the attribute data type is changed.
- Therefore, we use the operator `%TYPE` instead of a specific data type.

Example:

```
DECLARE
 v_lg Student.login%TYPE;
 ...
```

Where variable `v_lg` have the same type as an attribute `login` of table `Student`.

Operator %TYPE, Example

```
DECLARE
  v_email VARCHAR(30);                    -- improper way
  v_email_2 Student.email%TYPE;
BEGIN
  SELECT email INTO v_email from Student
    WHERE login = 'kra228';

  SELECT email INTO v_email_2 from Student
    WHERE login = 'kra228';
END;
```

Operator %ROWTYPE

- In some cases we use a structured data type, which contains variables corresponding to a table.
- An instance of such a data type corresponds to one record of a table.
- In this case we can use the `%ROWTYPE`.

Example:

```
DECLARE
  v_st Student%ROWTYPE;
  ...
```

Where the variable `v_st` contains the same variables and data types as the table `Student`.

Operator %ROWTYPE, Example

```
DECLARE
   v_login CHAR(6);                              -- improper way
   v_email VARCHAR(30);                          -- improper way
   v_student Student%ROWTYPE;
BEGIN
   SELECT login, email INTO v_login, v_email from Student
     WHERE login = 'kra228';                     -- improper way

   SELECT * INTO v_student from Student
     WHERE login = 'kra228';
END;
```

Variables, example

```
DECLARE
   C_VCHAR_MAXLEN CONSTANT NUMBER := 32767;
   v_date DATE := SYSDATE;
   v_number NUMBER NOT NULL := 1;
   v_student Student%ROWTYPE;
   v_name Student.name%TYPE;
BEGIN
...
END;
```

References

- Oracle books: `https://docs.oracle.com/en/database/oracle/oracle-database/18/books.html`:
  - PL/SQL Language Reference
  - PL/SQL Packages and Types Reference