



www.DBTechNet.org
DBTech VET



Lifelong
Learning
Programme

SQL Transactions Teacher’s Guide

Authors

Martti Laiho and Dimitris Dervos

Disclaimers

The DBTech VET teachers project has been funded with support from the European Commission. This publication [communication] reflects the views only of the authors, and the Commission cannot be held responsible for any use that may be made of the information contained therein. Trademarks of products mentioned are trademarks of the product vendors.

Objectives

Reliable data access must be based on properly designed SQL transactions with ZERO TOLERANCE to incorrect data in database. Knowledge and skills needed for developing reliable and secure data access in applications in ICT industry are critical for the success of trainees on their career in working life, whereas reliability failure of some developed application can be a catastrophe affecting the careers. Professional growing starts by proper education, and teachers are in key position as coaches of future professionals.

In this basic course, the challenge for the teacher/trainer is (a) to identify the current level of knowledge and skills in his student audience, and (b) to provide an overview on the basic concepts and the key challenges involved, in a way that raises the students’ interest and motivation in the underlying technology.

Target user groups

The target groups of this guide include teachers, trainers in vocational institutes and industry-oriented higher education institutes.

Prerequisites

Teachers are encouraged to (a) have attended a DBTech VET Teacher training course, (b) have hands-on skills on the use of the DebianDB database laboratory, and (c) have passed the experiments series using at least the DBMS product to be used on the course to be taught, but more preferably they need to have used at least 2 DBMS products: one that implements concurrency control via MGL (e.g. DB2 or SQL Server), and one that utilizes the MVCC technology (e.g. MySQL/InnoDB or Oracle).

Course Materials

Course Descriptions

SQL Transactions – Teacher Guide (this document, made available in six languages: English, Estonian, Finnish, Greek, Russian, and Spanish).

SQL Transactions – Theory and hands-on exercises

(SQL-Transactions_handbook_<language code>.pdf, made available in six languages: English, Estonian, Finnish, Greek, Russian, and Spanish)

SQL Transactions Intro – Powerpoint slides for presentations: In English, only.

Basics of SQL Transactions – PDF slides for presentations: in English, only.

Portable DBTechLab virtual database laboratory (OVA + documents of DebianDB 6)

Quick Start to DebianDB (pdf, introduction to DBTechLab 6): in English, only.

Learning outcomes based multiple-choice questions web form on SQL Concurrency Technologies: in English, only.

Evaluation forms (three types of generic evaluation web forms/questionnaires): in English, only.

Teaching Pedagogy

The key topics need first be presented in an **orientating session** utilizing carefully selected slides and demonstrations on using the virtual database lab and a selected DBMS product.

The orientation sessions should not be long (max 60 minutes) and after it the students should have the opportunity for **hands-on experimentation** with a selected DBMS product in the DBTechLab environment. The pedagogy is mainly based on hands-on experimentation whereby students can verify the problems and their solutions via a series of learning steps:

orientation/theory – experiments – new problems – need for more knowledge-

Free access to modern real world DBMS products that appear in the corresponding job market preferred skills list is expected to increase the trainees’ motivation and justify the need for investing time and effort in the skills training course.

Even if the SQL standard offers a common model for the SQL dialects of the DBMS products, the latter behave differently from the standard and from each other. The most important learning outcome is that the trainee learns to experiment with the selected DBMS product, and **verifies its behavior/functionality in practice**. The latter directly relates to transactional programming in application development tasks.

Teaching hints

You need to understand what you teach, and to be able to answer the questions.

Don’t try to teach everything if you don’t have enough time.

Don’t hurry!

Make sure that students understand COMMIT and ROLLBACK.

Focus on the issues marked with boldface letters above.

Remember that students can also read!

Topics and Learning Objectives

The topics and learning objectives of the SQL Transactions course module comply with the Information Management / Transaction Processing definition in the Computer Science Curricula 2013 by ACM and IEEE (<http://ai.stanford.edu/users/sahami/CS2013//strawman-draft/cs2013-strawman.pdf>, page 91) defining an elective course module on transaction processing for undergraduate level education.

The best benchmark for the DBTech VET basic level course module on SQL Transactions is that the course’s SQL Transactions Tutorial and Student’s Guide address the aforementioned CS2013 module topics.

<p>IM/TransactionProcessing [elective]</p> <p>Topics:</p> <ul style="list-style-type: none">• Transactions• Failure and recovery• Concurrency control <p>Learning Objectives:</p> <ol style="list-style-type: none">1. Create a transaction by embedding SQL into an application program.2. Explain the concept of implicit commits.3. Describe the issues specific to efficient transaction execution.4. Explain when and why rollback is needed and how logging assures proper rollback.5. Explain the effect of different isolation levels on the concurrency control mechanisms.6. Choose the proper isolation level for implementing a specified transaction protocol.

Figure 1 CS2013 Information Management (IM) / Transaction Processing

The learning objectives listed in Figure 1, when considered in the context of the current European VET education, are met as follows:

1. For the objective of embedding SQL transaction into an application program, we provide listing of a Java/JDBC program in Appendix 2, which need to be explained by the teacher. However, embedded SQL (ESQL) as a topic is not included in the current version of the DBTech VET course material.
2. With regard to the concept of implicit commits, it is assumed that in CS2013 means to imply the AUTOCOMMIT mode. In the DBTech VET course yet another implicit commit functionality is addressed: the automatic commit on DDL statements (e.g. in Oracle and MySQL/InnoDB).
3. Efficient transaction execution is addressed by the “Big Picture” of Database Server and Transaction processing in Appendix 3, which the teacher should explain to the students, and by the Chapter 3 entitled “Some Best Practices” in the “SQL Transactions” handbook.
4. The use of ROLLBACK is explained in Chapter 1 of the “SQL Transactions” handbook and in the hands-on experiments. Appendix 3 explains the role of transaction logging for transaction rollback and database recovery operations.

5. Chapter 2 in the “SQL Transactions” handbook presents the typical concurrency control problems, isolation levels of ISO SQL standard and implementations in DBMS products as solutions to these problems, plus explains the effects in terms of the basic concurrency control mechanisms, Multi-Granular Locking (MGL, LSCC) and Multi-Versioning (MVCC).
6. Choosing the proper isolation level to solve specific concurrency problems is addressed by a series of exercises and hands-on experimentation.

Note: “transaction protocol” in the 6th learning objective in CS2013 is not accurate concept in this context.

The following quote from “Recent Trends” in chapter 3.1 of the CC2008 report (predecessor to the CS2013 one), can be applied also to database transactions and data access technologies:

“The growing relevance of concurrency

The development of multi-core processors has been a significant recent architectural development. To exploit this fully, software needs to exhibit concurrent behavior; this places greater emphasis on the principles, techniques and technologies of concurrency. Some have expressed the view that all major future processor developments will include concurrent features, and with even greater emphasis on the concurrency elements. Such a view implies the increased emphasis on concurrency will not be a passing fashion but rather it represents a fundamental shift towards greater attention to concurrency matters.”

(<http://www.acm.org/education/curricula/ComputerScience2008.pdf>)

On this basic level course module on SQL Transactions the target learning objectives level of students is Level 2 (“Understand”), and partly Level 3 (“Apply”) of the CC2008’s variant of Bloom’s taxonomy (see CC2008 chapter 4.1.1).

Course Module Topics Outline - Instructor notes

Part 1 SQL Transaction – a Logical Unit of Work (LUW)

*Part 1 introduces transactions in **single-user environment**, without competition of the concurrent transactions.*

Hint: you may start by first introducing the DBTechLab virtual database laboratory (see 1.6), or by following the numbered order of the SQL Transactions handbook.

1.1 Problems and need for transactions (motivation)

- Lost data
- Incorrect data into database
- Incorrect results from database

1.2 Client/Server concepts [theory]

- Concepts relating to the ISO SQL standard: SQL-server, SQL-client, SQL-session, diagnostics
- Client-side: driver, dialogue: request, result
- Server-side: threads, caches

1.3 Introduction to SQL Transactions

SQL transaction [theory]

- Transaction as a logical unit of work, unit of recovery, unit of consistency.
- Transactions as building blocks of reliable applications, transferring the database from one consistent state to another consistent state.
- Implicit or explicit start of transaction – product dependent.

Explain the AUTOCOMMIT mode:

- Autocommit => every command is a transaction => no rollback services, more I/O

- Explicit transactions can be started on AUTOCOMMIT mode.

Transactional mode:

- Transactions are started implicitly.

Success or failure need to be tested after every command!

Execution may be successful in terms of SQL but not in terms of the application!

End of transaction (success: **COMMIT**, failure: **ROLLBACK**).

Note:

- Some products generate implicit COMMITs on DDL commands.

- Most products generate ROLLBACK on deadlocks (to be explained later).

Explain a generic overview of a database server (“Big Picture”) in Figure A3.1 of Appendix 3, and the “magical service” of **ROLLBACK** implementations, using the PDF slides of “**Basics of SQL Transactions**”.

Explain the following components of the ACID principle:

- atomicity (A) achieved by ROLLBACK,
- consistency (C) achieved by CONSTRAINTS¹ and tested transaction logic,
- durability (D) achieved by COMMIT.

Note: Isolation will be covered in Part 2.

1.5 Diagnosing SQL errors/exceptions [theory]

By SQLcode: SQLSTATE, GET DIAGNOSTICS

[For more information on these see Appendix 2, and more on this in the “Stored Routines” paper: exception handling / condition handling]

1.6 Hands-on laboratory

Introduction to the DBTechLab (DebianDB)

Guided import of DebianDB file (.ova) to VirtualBox

Guided session with students demonstrating the use of the Quick Start Guide to DebianDB, plus the use of the selected DBMS:

MySQL/InnoDB, Oracle XE, DB2 Express-C, PostgreSQL, or Pyrrho.

The exercises in the SQL Transactions handbook utilize MySQL, but they can also be conducted utilizing any one of the other four DBMS products as well as the MS SQL Server DBMS (see Appendix 1). In the virtual laboratory at /home/student/Transactions directory one can find (and copy-paste) the scripts required for experimenting with the stated exercise material..

Exercises 1.1 - 1.7:

AUTOCOMMIT mode / Transactional mode

Automatic ROLLBACK on errors – available in some DBMS products only

Transaction recovery

Database recovery up to the latest committed transaction

Part 2 Transactions in the Multi-User Environment

A program tested to work properly in the single-user environment may fail in the multi-user environment! => need to understand concurrency anomalies and how to cope with them => need to understand isolation levels => need to understand concurrency control (CC) mechanisms of the DBMS product

¹ In addition to SQL constraints, such as PRIMARY KEY, UNIQUE, FOREIGN KEY, CHECK, [NOT] NULL and some “exotic” constraints in some products, additional ”business rules” can be implemented by programmed SQL triggers, but these are not covered in the “SQL Transactions” handbook and may be too difficult topic on basic level courses.

2.1 Concurrency Problems

Write anomalies:

2.1.1 Lost Update

- possible in file-based systems
- not possible in DBMS products **during** the transaction

Blind Overwriting

- **after** the transaction commits, concurrent transactions may overwrite the results (solution: sensitive updates or optimistic locking by RVV)

Experiment: 2.1

Read anomalies:

2.1.2 Dirty Read

2.1.3 Non-Repeatable Read

2.1.3 Phantoms | Phantom Read ²

2.2 The ACID principle

This is a good orientation concept in basic transaction education.

However, on advanced level, **Isolation** (I) is the challenging ACID component due to different interpretations of the isolation concept. Most DBMS products are marketed as "ACID compatible", but this is actually not true in cases where the DBMS product supports isolation below the REPEATABLE READ one. The full isolation from concurrent transactions is available only on OCC products, especially in Pyrrho which is included in the portable DBTechLab virtual database laboratory environment..

2.3 ISO SQL standard Isolation Levels

for implementing concurrency control (CC)

READ UNCOMMITTED

READ COMMITTED (typical default of the products)

REPEATABLE READ

SERIALIZABLE (default according to the SQL standard)

2.4 Concurrency Control Mechanisms

2.4.1 Multi-Granular Locking (MGL)

X-locks / S-locks + intent locks + index/predicate locks + schema locks

- may lead to **deadlocks**
- explicit locking (Lock Table)

Some products support explicit row-level locking (SELECT..FOR UPDATE)

2.4.2 Multi-Versioning (MVCC) with non-standard isolation levels:

Latest Committed (Currently Committed, Read Committed Snapshot)

Snapshot: does not protect for phantoms, but eliminates phantom reads

*Note: MVCC eliminates read locks, but **locking is still used for writing***

– *which may lead to deadlocks, for example in UPDATE-UPDATE scenarios!*

2.4.3 Optimistic Concurrency Control (OCC)

² Note the difference: the "Phantom" problem is solved by preventing writing of phantom rows (by means of MGL), whereas the "Phantom Read" problem is solved by "closing the eyes" from seeing the phantom rows (by means of MVCC)

All writes are applied to a private cache of the transaction, and they get synchronized into the database only at commit time! The first transaction to commit wins!

Can be tested in the DBTechLab environment using the Pyrrho DBMS

2.4.4 Summary – see table 2.4 in the textbook.

2.5 Hands-on labs on Concurrency

Exercise 2.1 Lost Update problem

Exercise 2.2 SELECT-UPDATE scenarios a) and b)

Exercise 2.3 UPDATE – UPDATE scenarios in opposite order => deadlock

Exercise 2.4 Dirty Read problem

Exercise 2.5 Non-Repeatable Read

Exercise 2.6 Insert-Phantom Problem

Exercise 2.7 A SNAPSHOT study with different kinds of Phantoms

3 Best Practices

No user dialogues in transactions

Short transactions

Set the proper isolation level in the beginning of the transaction

Test for errors after every SQL command

Know the behavior of your DBMS product – different products behave differently

Know the SQL dialect of your DBMS product

Try to comply with the ISO/SQL standard, if possible

Appendix 1

Exercises of Part 1 and Part 2 are applied to Microsoft **SQL Server Express** and **the results are compared** to those obtained using other DBMS products.

Appendix 2

Short introduction to transactional programming using Java and the JDBC API.

Examples make use of a sample BankTransfer program.

- Programming try – catch for trapping SQLExceptions
- Connecting to the database server
- Retry Pattern (retry wrapper) invoking the integer typed transaction method.
- Every transaction begins by turning OFF the AUTOCOMMIT mode
- Passing SQL statements as parameters to methods
- Preparing parameterized SQL statements + passing parameter values (prevents SQL injections for security and improves performance)
- Retrieving the set of affected rows
- COMMIT and ROLLBACK are methods of the connection object.

The JDBC API provides unifying data access to differently behaving native DBMS products, but still some differences need to be taken into consideration by the programmer.

Appendix 3

Database recovery utilizing the transaction log.

The “Big Picture” of a generic database server is explained, including the caches and the files used. The transaction log is the most important file/filesset used, since it contains the latest committed transactions made persistent onto the disk.

Checkpoint processing is explained in relation with the database rollback/recovery information recorded in the transaction log.

[Optional topics]

Cursor Processing – automatic close at end of transaction.

This is essential technology in Embedded SQL for accessing the set-based results of SQL queries by record-oriented programming languages, and in the extended procedural SQL language (see the Stored Routines paper).

In the modern data access APIs, Server-side cursor processing materializes as wrapped by the Resultset object of the JDBC API, and as the Recordset object in some other APIs.

Topics not to be covered in the basic level course

The following topics belong to an advanced level VET oriented courses:

- Savepoints – transaction logic that one may rarely need to implement
- Locking details, timeouts, lock escalation
- Nested transactions – not supported in today’s products
- Distributed transactions
- Stored Routines in the transactional context

For more information on the above, see the following documents at:

- http://www.dbtechnet.org/papers/SQL_StoredRoutines.pdf
- <http://www.dbtechnet.org/papers/ConcurrencyTechnologies.pdf> (to become available in 2015)
- http://www.dbtechnet.org/papers/RVV_Paper.pdf
- <http://www.dbtechnet.org/papers/Distributed Transactions.pdf> (to become available in 2015)

Serializability theory, 2PL, etc suit as preliminary orientation and research topics for students in science HE programs, but not for VET students. For the latter, these topics would be unnecessary material to exploit, considering the time limitations in their curricula, plus since the topics do not relate directly to practical application code development in the modern DBMS products.

Evaluation

- Students fill in the proper DBTech VET evaluation form which is made available in three versions: course, seminar, and workshop. The form in question is not meant for assessing student skills and credit point assignment.

- Students fill in the Multiple-choice questions form and submit their answers to the teacher for measuring the learning outcomes achieved. The form may also be used for student assessment.
- The teacher fills in the DBTech VET Lab summary and returns it to the [local] DBTech VET/DBTechNet coordinator.