MySql Concrete Architecture (Storage Management)

November 1, 2017

Team

Ante Pimentel

Dusan Birtasevic

- Francis Okoyo
- Hashim Al-Helli

Richard Van

Introduction

- This presentation illustrates the divergences between the conceptual and concrete architecture of a system.
- Our topic of interest is the storage management component of the MySQL Server.
- The learning outcome of this presentation is a better understanding of why certain differences exist between architectures.

Layout

- Subsystems & Interactions

Derivation Process

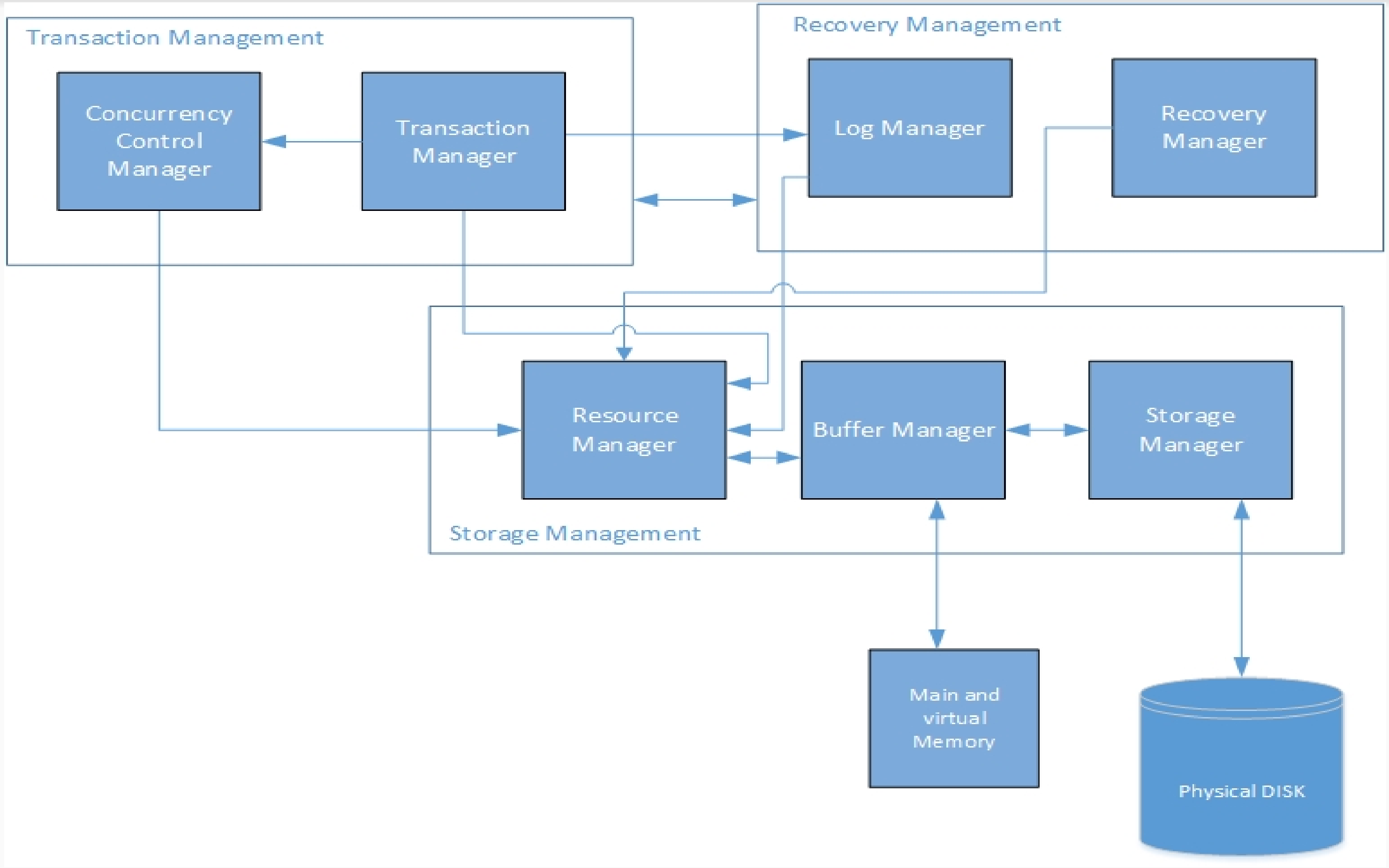
- Architecture Styles & Design Patterns

Storage Management

- Architecture Differences
- LsEdit Screenshots
- Use Case

Lessons Learned

Subsystems and Interactions



Derivation Process

- Visualized interactions using lsedit.
- Simplified the view of lsedit by removing unneeded file dependences.
- Analyzed the code structure and file names and tried to map file names to components.
- If filename wasn’t clear we went inside the code and analyzed the comments.
- When we found the right files, we contained and visualized them.

Some Important files in MySQL Server Folder

client - command line client utility code.

sql - main folder of MySQL it contains the parser and optimizer,execution engine and handler to choose a storage engine and many other things.

Sql-common - files that used by client and server.

Storage - The storage engines like **Inndob** and **MyISSAM** implemented here.

Vio - virtual input output contains implementation of low level network I/O code.

Architecture Styles and Design Patterns

- Layered architecture (top-level and Storage management internals)
- Pipe-and-filter architecture (query processing)
- Client-server architecture (application layer)
- Facade design pattern. The file mysqld.cc act as an interface between the client and the other components.

Storage Management - Storage Manager

Storage Manager

- Responsible for providing fast read/write access to tablespaces and logs of the database.
- fil0fil.cc located in the fil directory contains the implementation of the low-level file system. Contains the methods and structures needed to manipulate them.
- fsp0fsp.cc located in the fsp directory takes care of the file space management, which handles the allocation of pages and keeps track of which files are used, open, and closed.

Storage Management - Buffer Manager

Buffer Manager

- The Buffer Manager is responsible for efficiently storing the data in memory for manipulation
- it accepts requests from the Resource Manager and decides how much memory to allocate.
- The buffer manager is located in the buf directory.
- buf0buf.cc file is the buffer pool, buf0flu.cc flush the buffer pool and buf0lru.cc contains the replacement algorithm that decides which blocks should be shifted back to disk.
- buf0rea.cc calls the storage manager to initiate a file read.

Storage Management - Resource Manager

Resource Manager

- The responsibility of the Resource Manager is to accept requests from the higher level layers and translate them into an appropriate format that can be understood by the Buffer Manager.
- The resource manager is found inside the dict directory.
- It can be seen from the diagram that there is a direct interaction between storage manager and resource manager where the resource manager skips the buffer and accesses the storage manager directly.

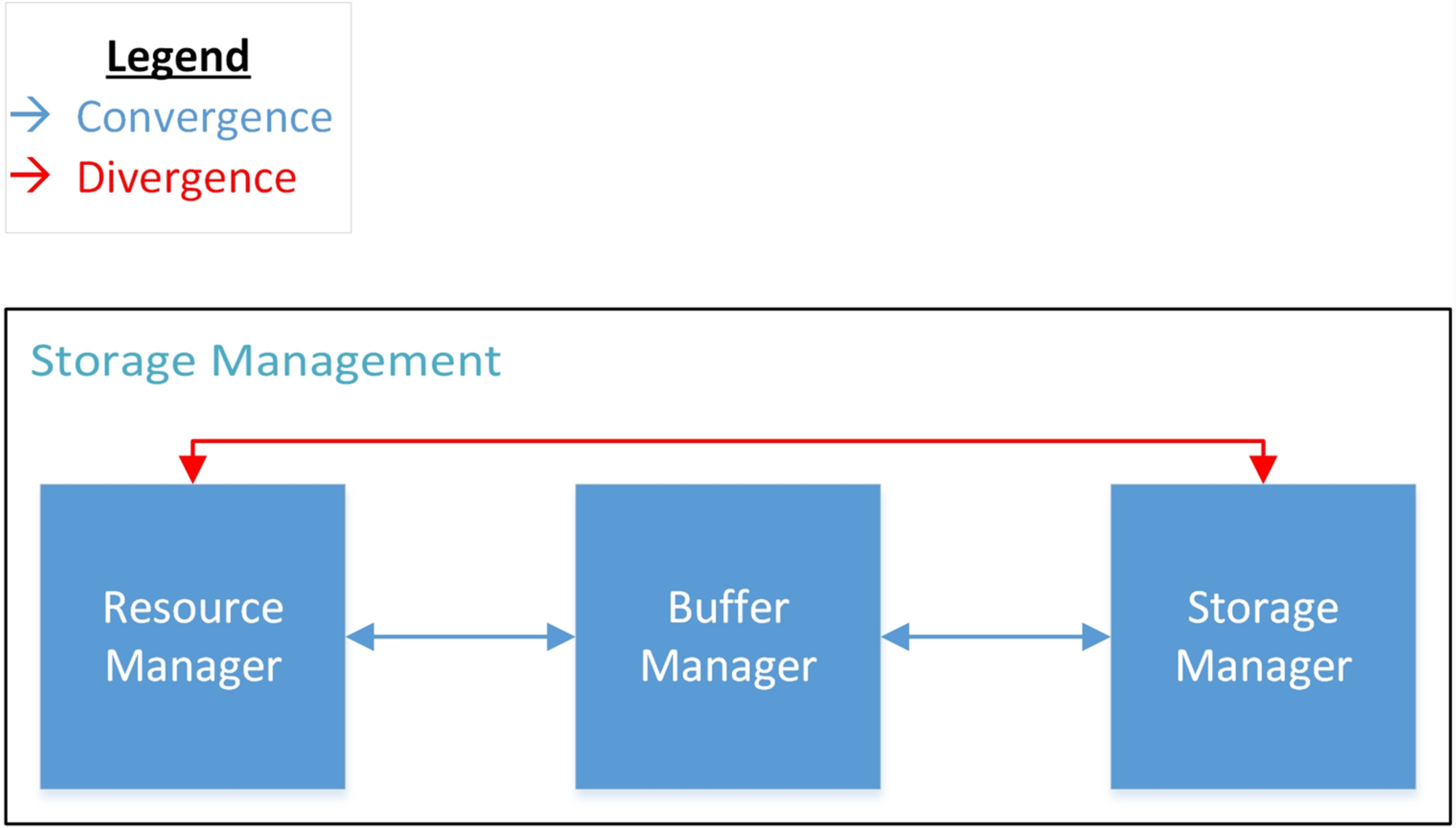
Architecture Differences: Sticky Notes

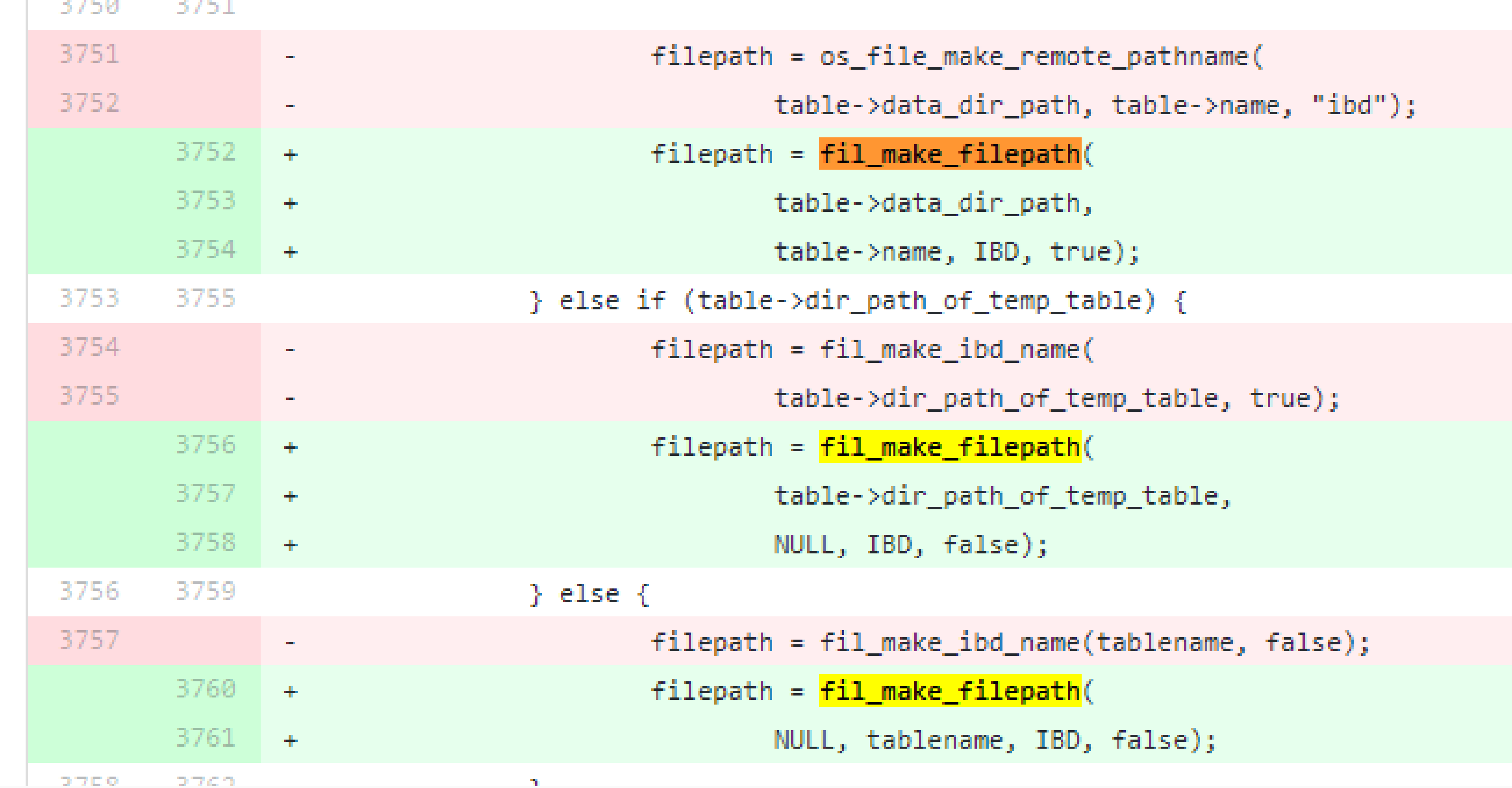
Which: dict0dict.cc (ResourceManager) -> fil0fil.cc (StorageManager) fil_make_filepath

Who: Kevin Lewis

When: Jan 17, 2014

Why: Refactoring. Consolidating several functions that make a filename.


Legend
→ Convergence
→ Divergence
Storage Management
Resource Manager
Buffer Manager
Storage Manager



```diff
-		filepath = os_file_make_remote_pathname(
-			table->data_dir_path, table->name, "ibd");
+		filepath = fil_make_filepath(
+			table->data_dir_path,
+			table->name, IBD, true);
	} else if (table->dir_path_of_temp_table) {
-		filepath = fil_make_ibd_name(
-			table->dir_path_of_temp_table, true);
+		filepath = fil_make_filepath(
+			table->dir_path_of_temp_table,
+			NULL, IBD, false);
	} else {
-		filepath = fil_make_ibd_name(tablename, false);
+		filepath = fil_make_filepath(
+			NULL, tablename, IBD, false);
```

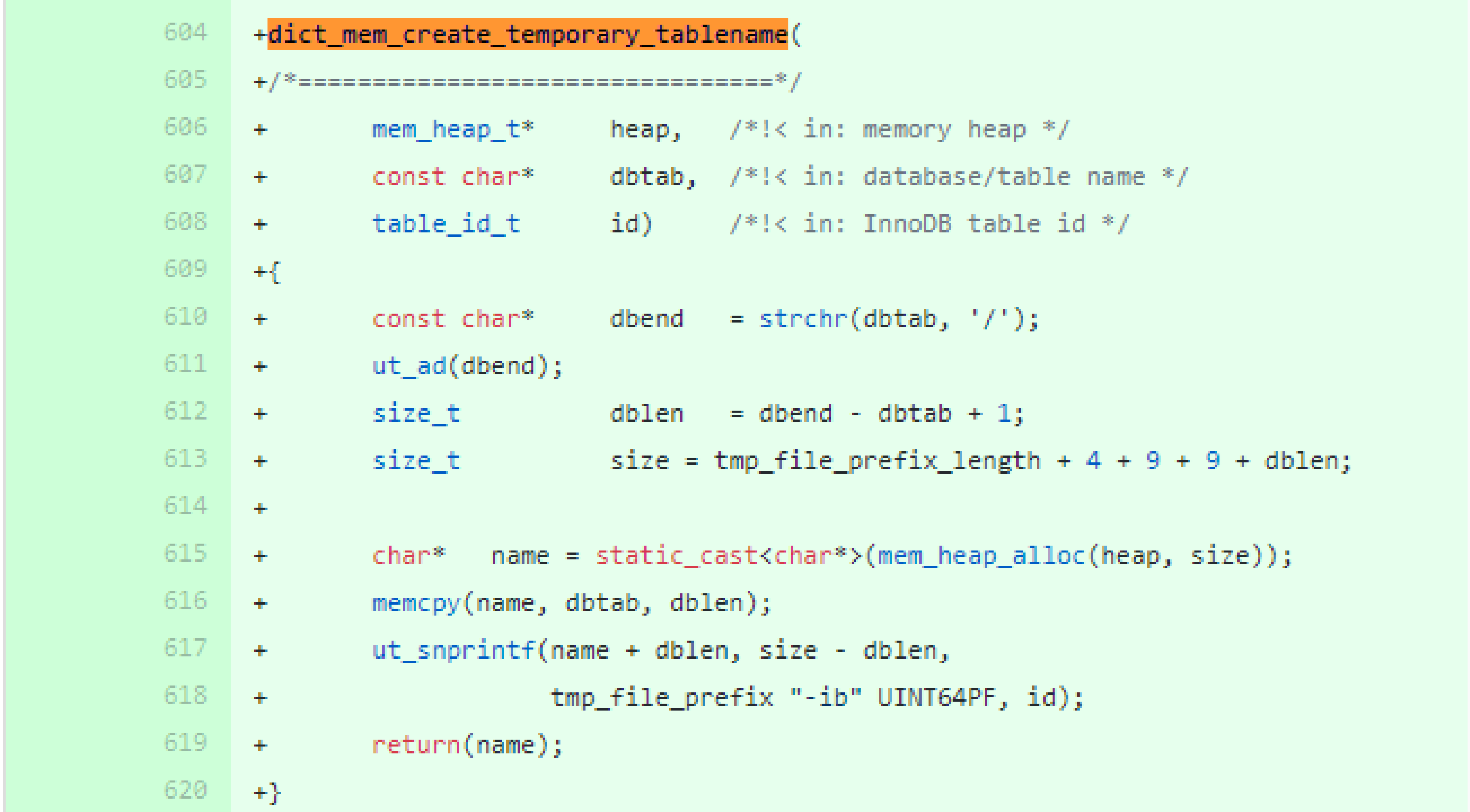
Architecture Differences: Sticky Notes

Which: fil0fil.cc (StorageManager) -> dict0mem.cc (ResourceManager) dict_mem_create_temporary_tablename

Who: Jimmy Yang

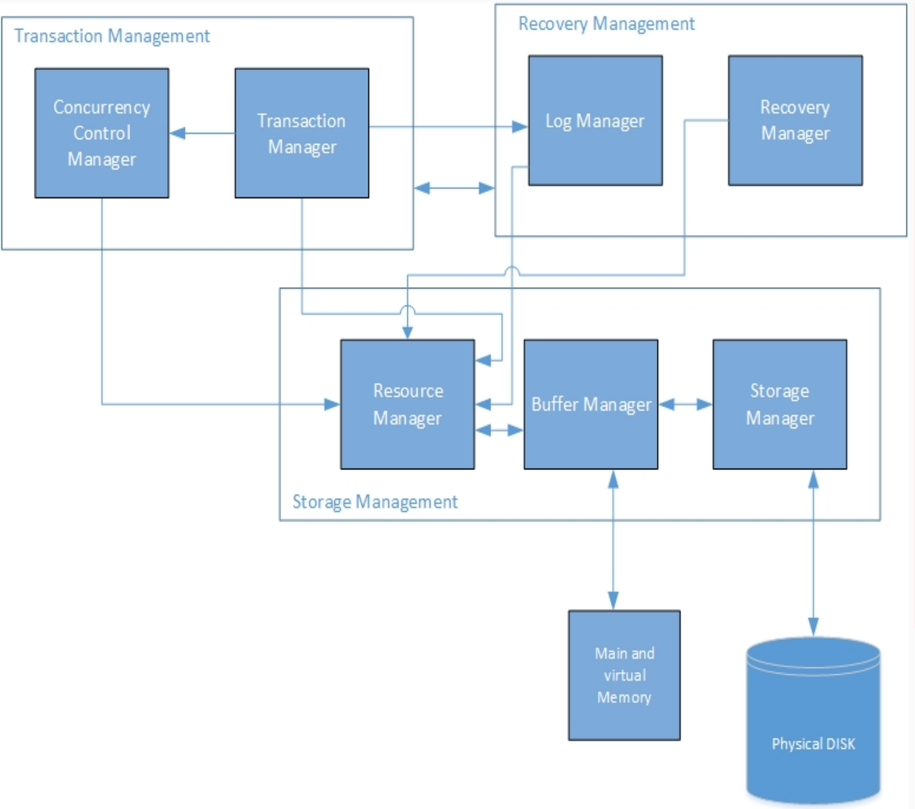
When: Sept 5, 2012

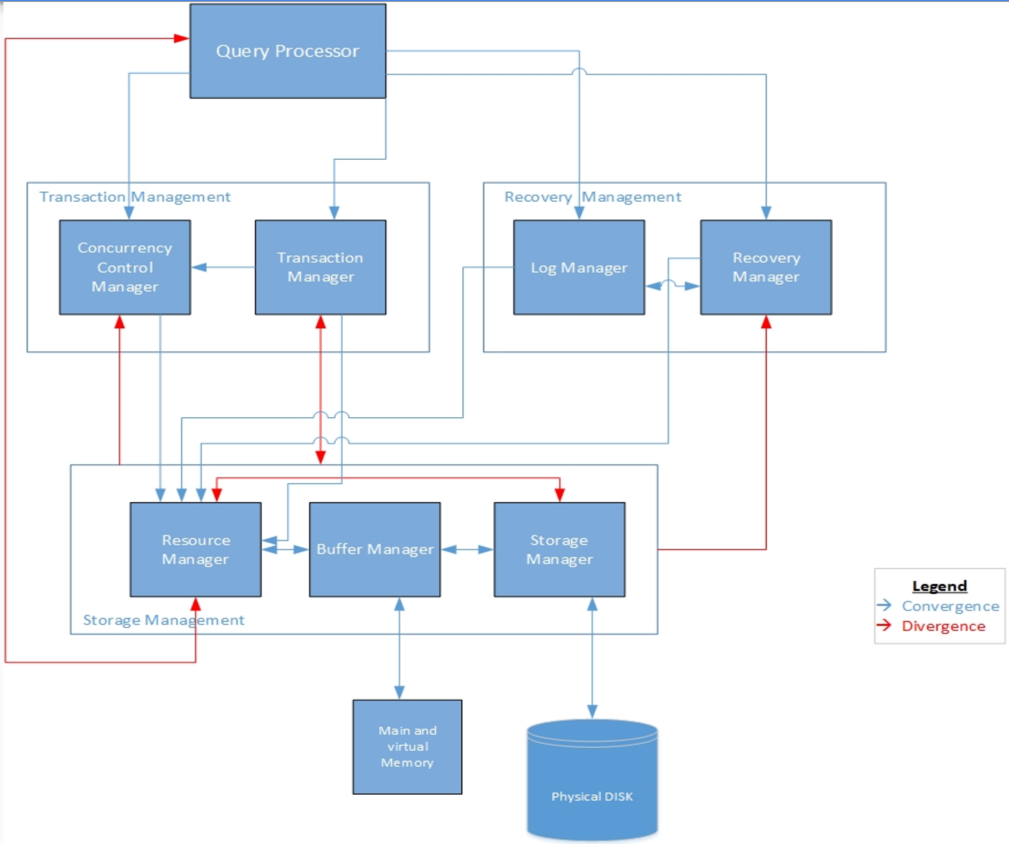
Why: Bug fixing



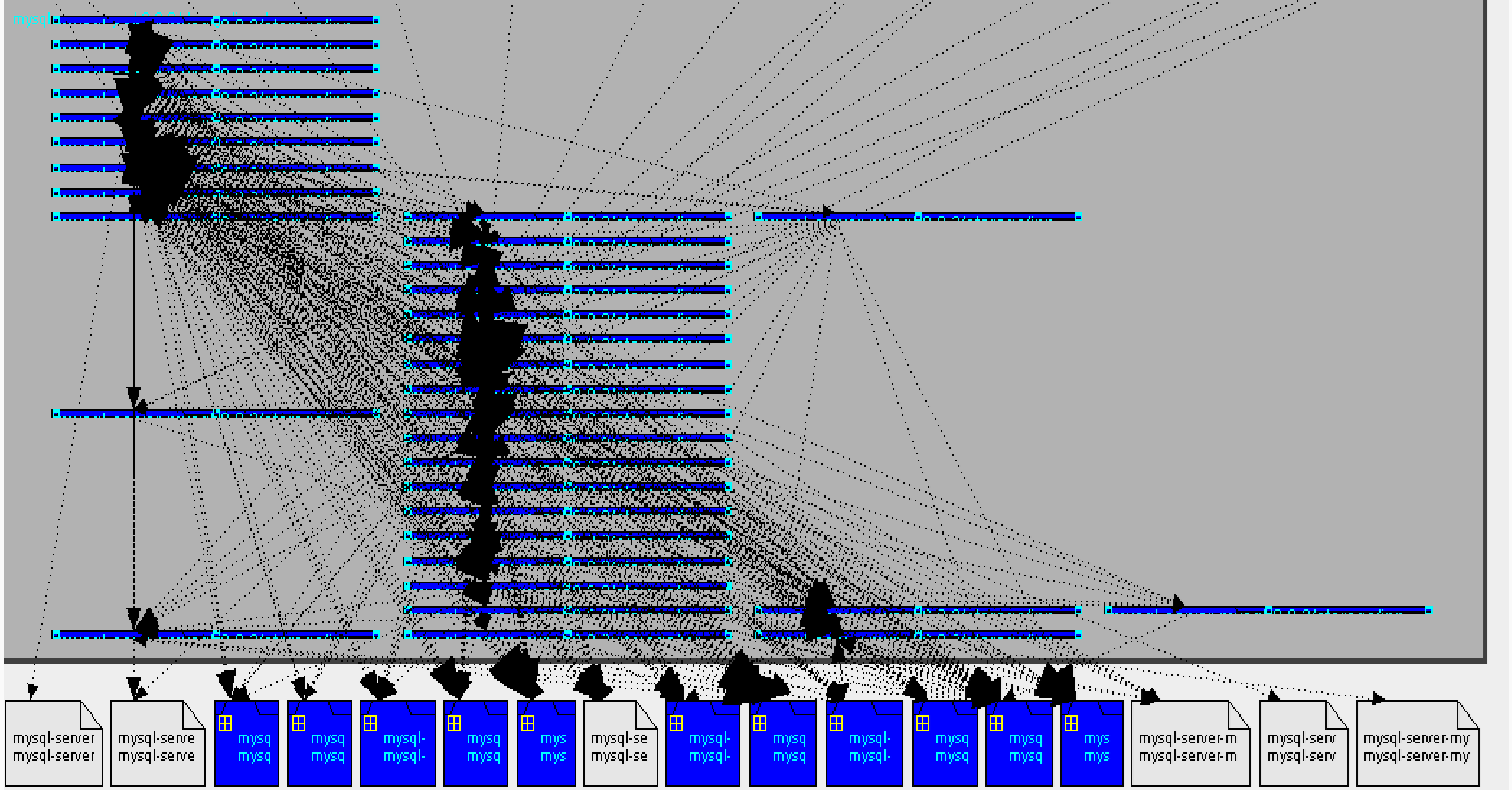
```
+dict_mem_create_temporary_tablename(
+/*====================================*/
+       mem_heap_t*     heap,   /*!< in: memory heap */
+       const char*     dbtab,  /*!< in: database/table name */
+       table_id_t      id)     /*!< in: InnoDB table id */
+{
+       const char*     dbend   = strchr(dbtab, '/');
+       ut_ad(dbend);
+       size_t          dblen   = dbend - dbtab + 1;
+       size_t          size = tmp_file_prefix_length + 4 + 9 + 9 + dblen;
+
+       char*   name = static_cast<char*>(mem_heap_alloc(heap, size));
+       memcpy(name, dbtab, dblen);
+       ut_snprintf(name + dblen, size - dblen,
+                   tmp_file_prefix "-ib" UINT64PF, id);
+       return(name);
+}
```

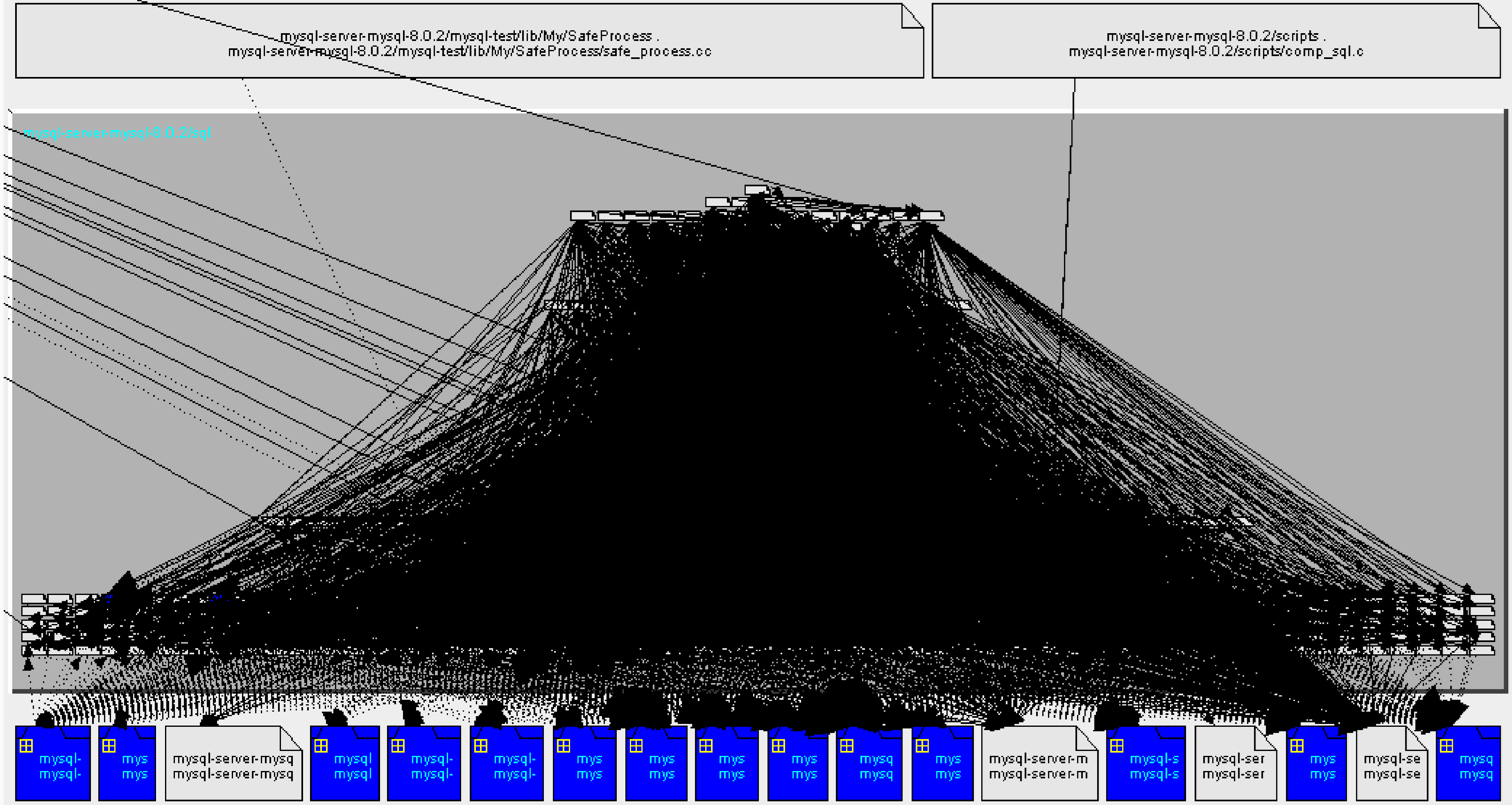
Side by side comparison: Conceptual and Concrete architectures

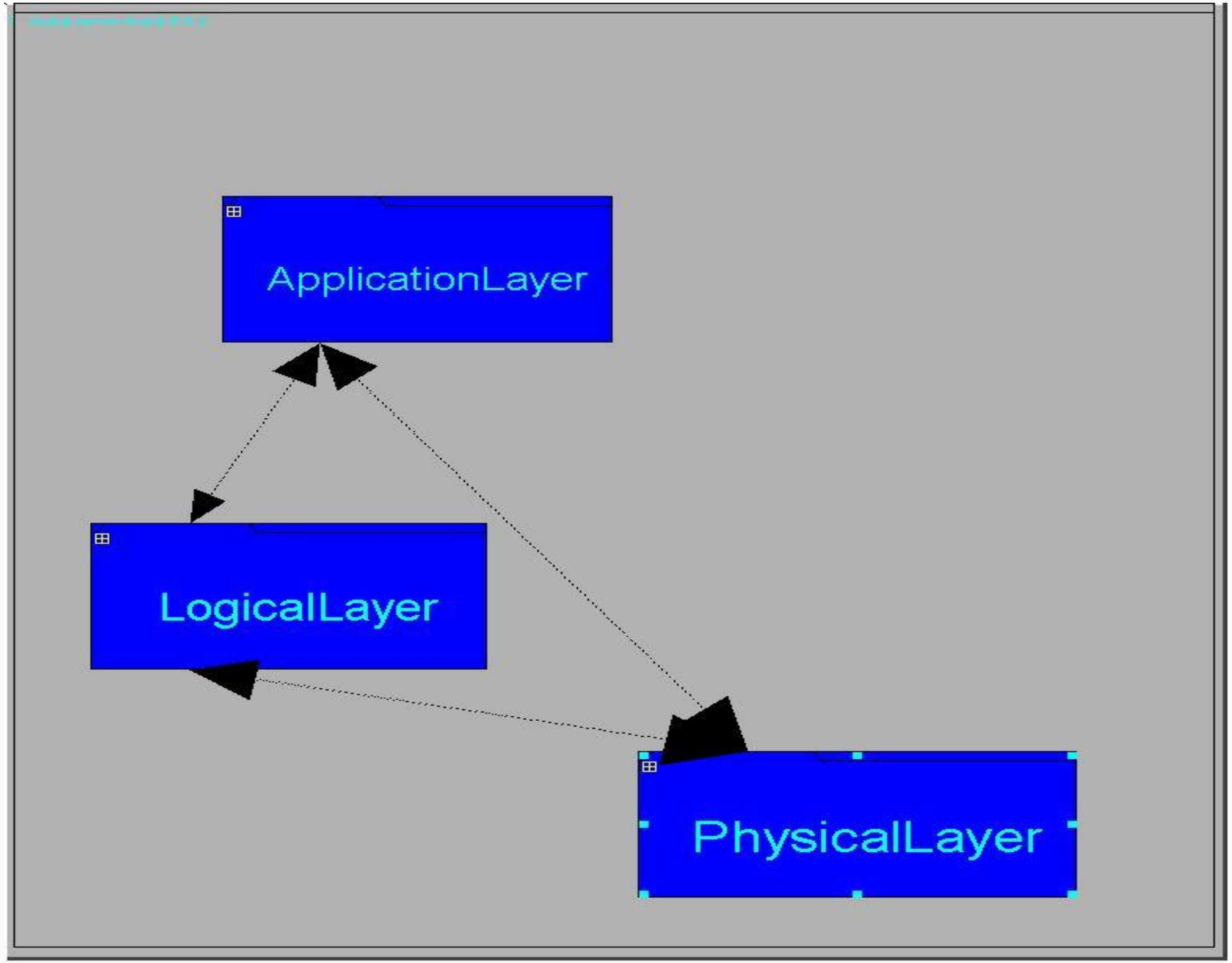


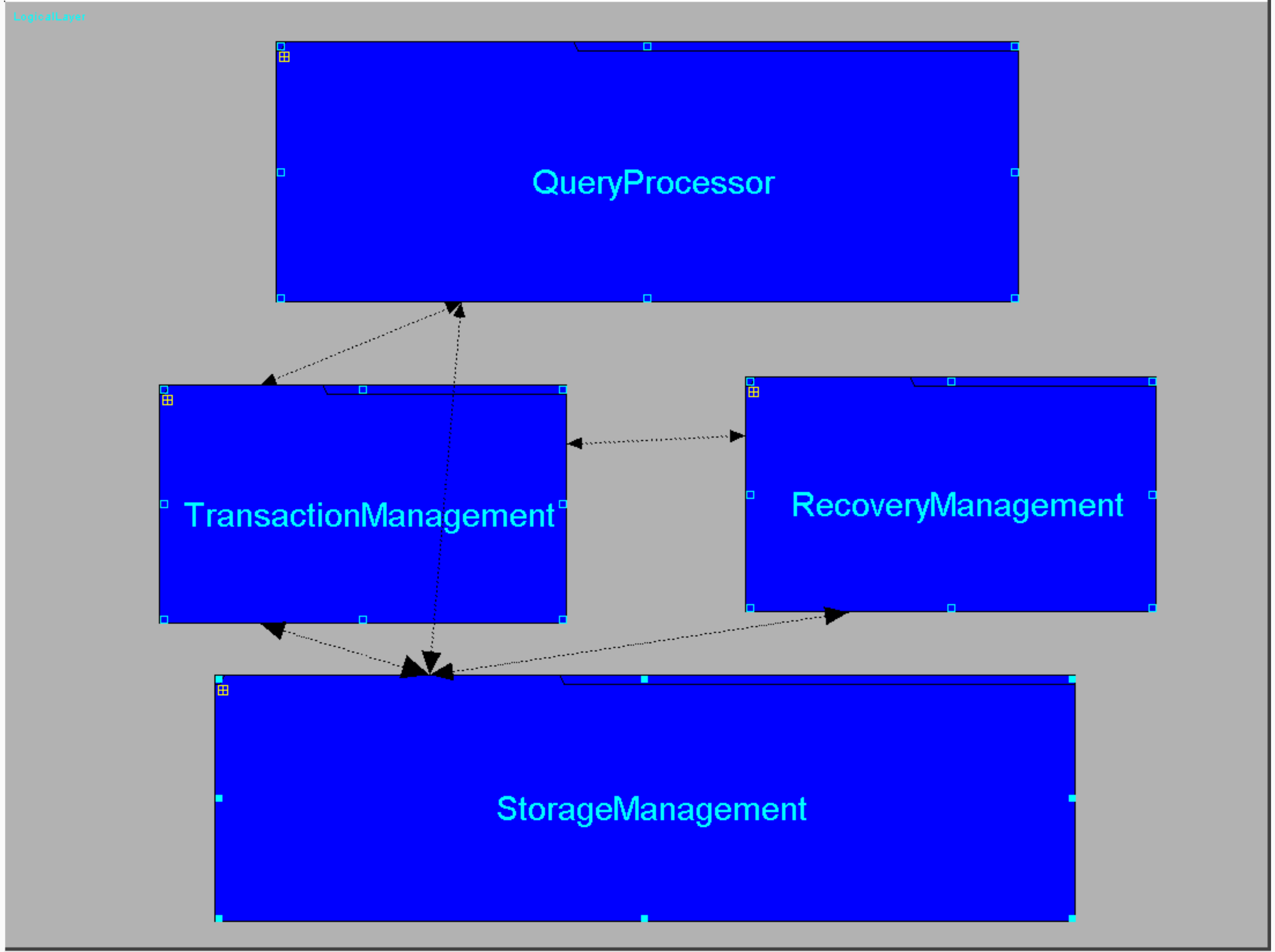


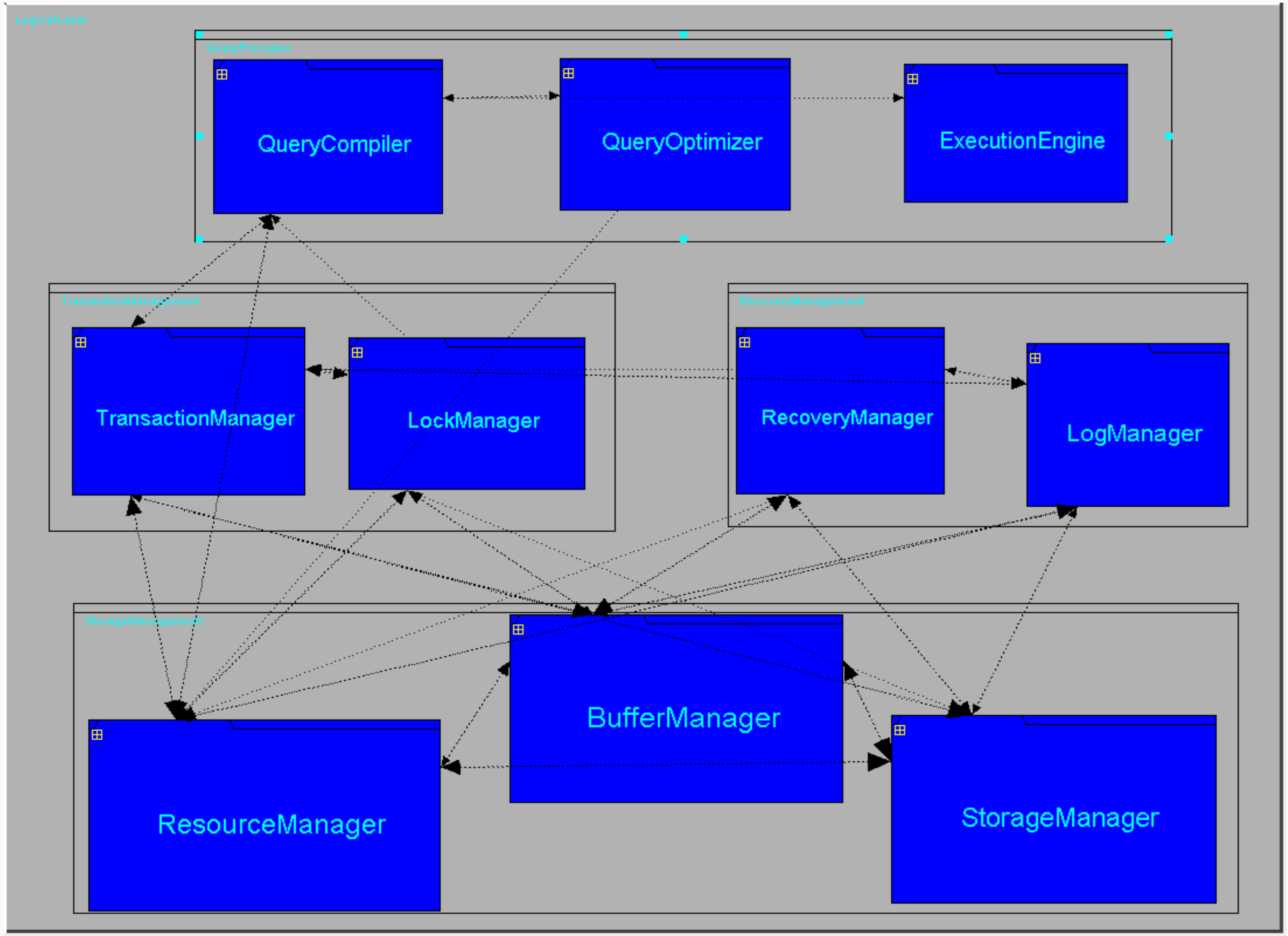
LsEdit Screenshots


mysql-server
mysql-server
mysql-serve
mysql-serve
mysq
mysq
mysq
mysq
mysql-
mysql-
mysq
mysq
mys
mys
mysql-se
mysql-se
mysql-
mysql-
mysq
mysq
mysql-
mysql-
mysq
mysq
mysq
mysq
mys
mys
mysql-server-m
mysql-server-m
mysql-serv
mysql-serv
mysql-server-my
mysql-server-my

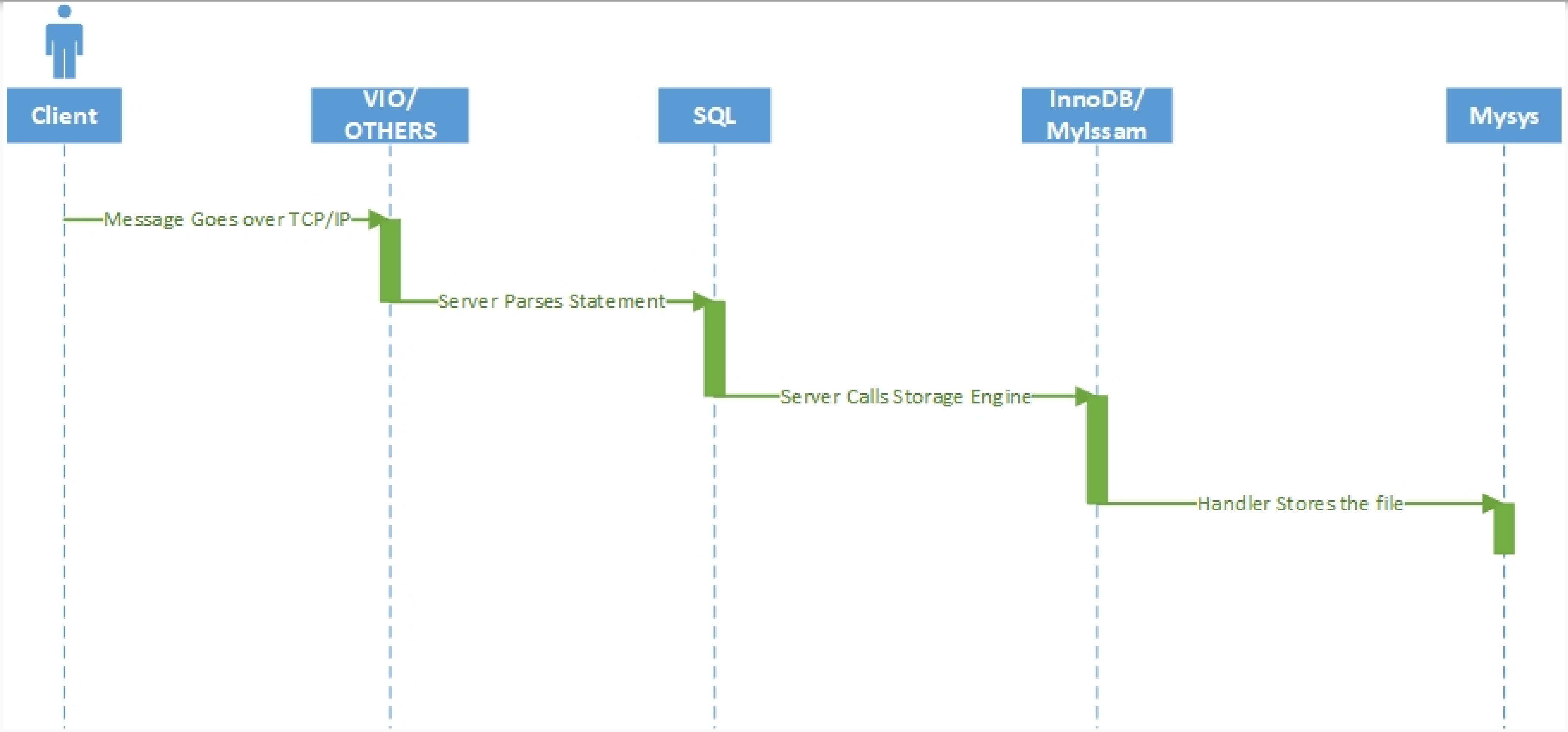

mysql-server-mysql-8.0.2/mysql-test/lib/My/SafeProcess .
mysql-server-mysql-8.0.2/mysql-test/lib/My/SafeProcess/safe_process.cc
mysql-server-mysql-8.0.2/scripts .
mysql-server-mysql-8.0.2/scripts/comp_sql.c
mysql-server-mysql-8.0.2/sql
mysql- mysql-
mys mys
mysql-server-mysq mysql-server-mysq
mysql mysql
mysql- mysql-
mysql- mysql-
mys mys
mys mys
mys mys
mys mys
mysq mysq
mys mys
mysql-server-m mysql-server-m
mysql-s mysql-s
mysql-ser mysql-ser
mys mys
mysql-se mysql-se
mysq mysq


ApplicationLayer
LogicalLayer
PhysicalLayer


LogicalLayer
QueryProcessor
TransactionManagement
RecoveryManagement
StorageManagement


LogicalLayer
QueryProcessor
QueryCompiler
QueryOptimizer
ExecutionEngine
TransactionManagement
TransactionManager
LockManager
RecoveryManagement
RecoveryManager
LogManager
StorageManagement
BufferManager
ResourceManager
StorageManager

Use Case - INSERT Statement



Lessons Learned

- Retrieving Concrete architecture of a good documented software is possible to achieve with less problems.
- Concrete architecture may contain components that don’t exist in the conceptual architecture.
- There are more interactions between components in the concrete architecture than the conceptual.
- Lsedit is a powerful tool but not as user friendly.

[illegible]