InnoDB

InnoDB: Taking advantage of InnoDB's New Table Formats & The InnoDB Roadmap

Heikki Tuuri / Innobase Oy
www.innodb.com
heikki.tuuri@innodb.com

Talk at MySQL ComCon Europe, Frankfurt Nov 10th, 2004

InnoDB

Speaker

- Born 1964 in Helsinki Finland
- PhD in mathematical logic from the University of Helsinki
- In 1988 – 1995 worked as a researcher and assistant professor at the departments of Mathematics and Computer Science
- Founded Innobase Oy in 1995

InnoDB

InnoDB history

- First line of code was written on Jan 20th, 1994
- The original goal was to make the world's fastest disk-based relational database
- In 1999, InnoDB was completed, Heikki Tuuri had written 110 000 lines of C
- Marketing efforts 1999 – 2000
- Heikki met Monty of MySQL in August 2000

InnoDB

InnoDB history (continued)

- Heikki wrote the InnoDB/MySQL interface ha_innodb.cc in Sept 2000 – March 2001
- March 12th, 2001 InnoDB was released in MySQL-3.23.34a

InnoDB

The company: Innobase Oy

- Owned by Heikki Tuuri
- Located in Helsinki, Finland
- Pekka Lampio, MSc., works with InnoDB Hot Backup development
- Marko Mäkelä, PhD in Computer Science from Helsinki Univ. of Tech., develops new space-saving InnoDB table formats
- Jan Lindström, PhD of Computer Science from Univ. of Helsinki improves locking

InnoDB

Innobase Oy (continued)

- Innobase Oy is an Original Equipment Manufacturer for MySQL AB (the product is the InnoDB storage engine in MySQL)
- Main sources of Innobase Oy revenue:
- 1) InnoDB Hot Backup (a non-free tool),
- 2) royalty from MySQL Pro licenses,
- 3) MySQL technical support contracts
- Innobase Oy is profitable :)

InnoDB

Multiple tablespaces

- A new feature that appeared in MySQL-4.1.1
- The feature was sponsored by RightNow Technologies, Inc. of Bozeman, Montana
- Allows you to place each InnoDB table into its own .ibd file; in this respect similar to MyISAM

InnoDB

Multiple tablespaces (cont.)

- You enable the feature by putting the line
  innodb_file_per_table
  to the [mysqld] section of your my.cnf file

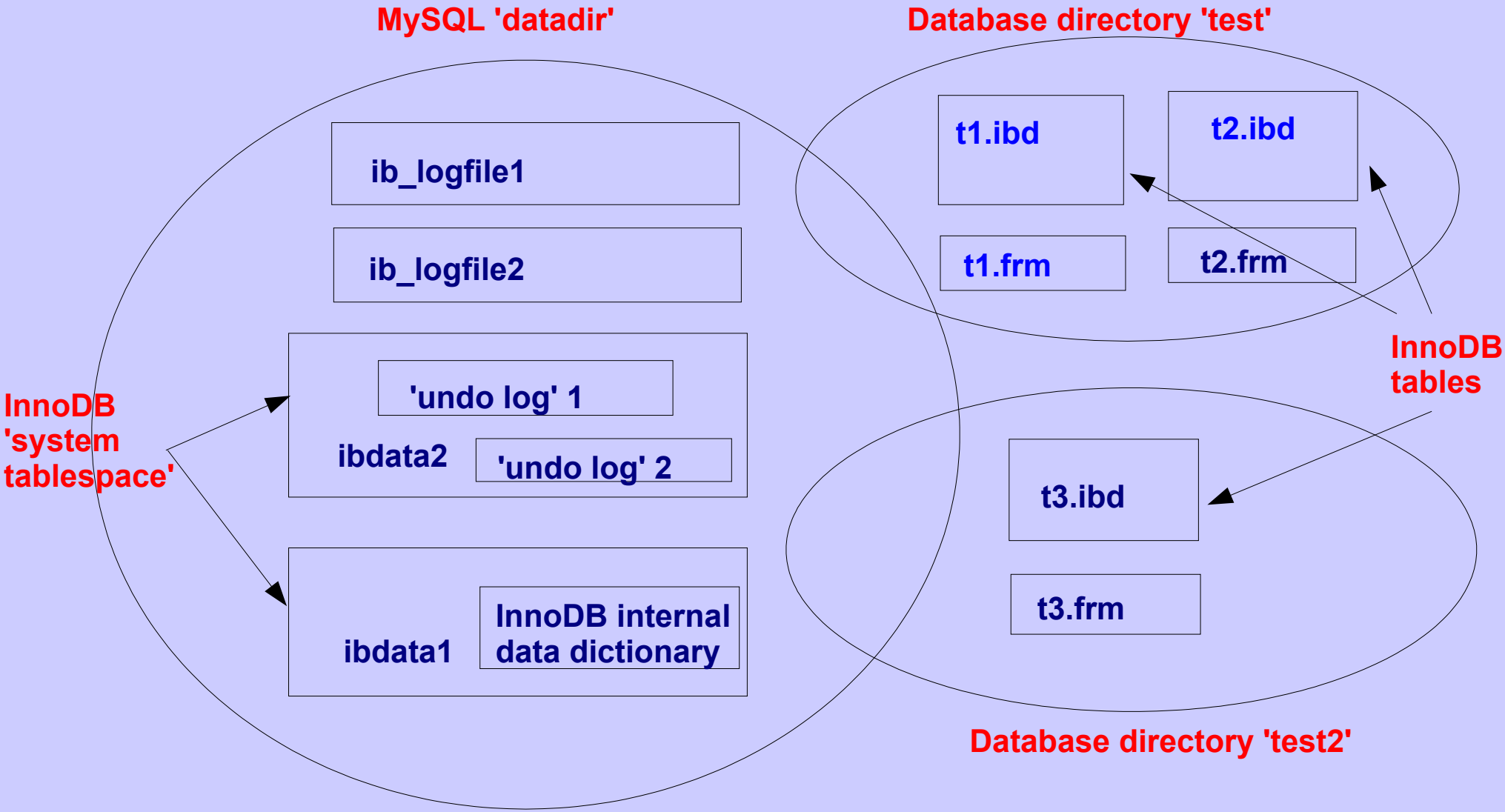
- InnoDB will still use the familiar ibdata files as the 'system tablespace'. The system tablespace contains the 'undo logs' (the 'rollback segment' of InnoDB) and the internal InnoDB data dictionary

- Old tables stay in the ibdata files, but new tables are created to .ibd files into the database directory of the table

InnoDB

Multiple tablespaces (cont.)

Files when using multiple tablespaces



InnoDB

Multiple tablespaces (cont.)

- An example:

```
USE test;
CREATE TABLE heikki(a INT) TYPE=
                                InnoDB
```

  creates a file 'heikki.ibd' in the directory 'test' under the datadir of the MySQL server. This is similar to MyISAM: a MyISAM table would create the files heikki.MYD and heikki.MYI in the same directory.

- The .ibd file contains both the data AND the indexes of the table.

InnoDB

Multiple tablespaces (cont.)

- If you remove the line innodb_file_per_table from my.cnf, then InnoDB again starts creating new tables inside the ibdata files. But the tables in .ibd files will stay visible and can still be used.

InnoDB

Multiple tablespaces (cont.)

- Advantages of using multiple tablespaces:
- If you DROP a table or run OPTIMIZE on it, the disk space is freed to the operating system. Recall that ibdata files never shrink, and you can never remove an ibdata file from an InnoDB installation.
- In Unix, database administrators can symlink database directories or individual tables to particular disks, and thus have more control in balancing disk I/O. (Though load balancing can also be achieved by creating several ibdata files and spreading them over the disks.)
- You can restore individual tables from a backup taken with InnoDB Hot Backup, or from a cold backup. Remember that the backup MUST be from the same InnoDB installation, and you must not have run ALTER TABLE on the table meanwhile.

InnoDB

Multiple tablespaces (cont.)

- Advantages of multiple tablespaces (cont.):
- In the future, you will be able to move 'clean' .ibd files also between InnoDB installations.
- Performance of multiple tablespaces seems to be as good as for a single ibdata file.
- There are no clear disadvantages in using multiple tablespaces.

InnoDB

Multiple tablespaces (cont.)

- IMPORTANT: you CANNOT move .ibd files manually around like you can MyISAM tables. Use

```
RENAME TABLE test.heikki
             TO test2.heikki
```

  to move heikki.ibd to database test2.

- You CANNOT yet move .ibd files from one MySQL instance to another. The reason is that InnoDB tables contain transaction id's and log sequence numbers, as well as internal table id's. It is in the TODO to allow moving .ibd files to another instance.

InnoDB

Multiple tablespaces (cont.)

- If you have a 'clean' backup of an .ibd file, you can restore it to the database with commands:

```
ALTER TABLE heikki DISCARD
                        TABLESPACE;
ALTER TABLE heikki IMPORT
                        TABLESPACE;
```

- 'Clean' means:
- 1) no uncommitted transactions in the .ibd file;
- 2) no unmerged insert buffer records;
- 3) purge has removed all delete-marked index records;
- 4) all pages written from the buffer pool to the .ibd file.
- Cleaning: let mysqld run idle on the installation. End all user transactions. When SHOW INNODB STATUS says that 'main thread is waiting for server activity', then every .ibd file is 'clean'.

InnoDB

Compressed InnoDB table formats

- Marko Mäkelä has reduced the space usage of InnoDB tables by about 20 % in MySQL-5.0, due to be released before the end of 2004. Marko has accomplished this saving by removing unnecessary column lengths stored into each InnoDB record.
- We will also implement a transparent, on-the-fly zip-like compression that will reduce space usage a further 50 %. This will appear in MySQL-5.1.
- Transparent online compression of both data and indexes seems to be a unique feature in the database world.

InnoDB

Compressed InnoDB table formats

- In MySQL-5.0, all newly created InnoDB tables will have the format that saves 20 % of space. Old tables will remain in the old format.

- The zip-like compression in 5.1 works like this:
  - Each 16 kB InnoDB data file page is squeezed to 8 kB on disk (the size may also be configurable).
  - We may also add a secondary buffer pool where the pages are kept compressed before bringing them to the ordinary buffer pool. The size of the secondary buffer pool would be configurable.
  - Each table will be in its own .ibd file.

InnoDB

Compressed InnoDB table formats

- Compression uses a lot of CPU. Under a write-intensive workload, compression may use almost the power of one CPU.
- Under a read-intensive workload CPU usage in much smaller, since decompression uses only 1 / 3 of the CPU time of compression.
- If the data on data pages compresses very little, compression will actually waste memory, because we cannot then put more than 8 kB of data into a 16 kB buffer pool page, since we must be able to squeeze it to 8 kB on disk. => Do not try to compress JPEG images, or other already compressed data!
- If we do not use a 'secondary' buffer pool, compression does not save main memory (RAM), because pages are in the uncompressed format in the buffer pool.
- File corruption of a compressed page is more fatal: if a single bit has changed in corruption, you lose the whole page

InnoDB

Upcoming features

http://www.innodb.com/todo.php

- Run an internal COMMIT for each 10 000 rows in ALTER TABLE. Removes the risk of a runaway rollback. Appears in 4.1.8.
- Remove unnecessary next-key locking in InnoDB. The my.cnf option innodb_locks_unsafe_for_binlog already does this in many cases in 4.1. When MySQL get 'row-level' binlogging in 5.1, we can remove the word 'unsafe'. (If we do not use next-key locks, phantom rows can appear, and these may in rare cases break MySQL's replication and roll-forward using the binlog.)
- Add support for a two-phase commit of distributed transactions. The interface is as in J2EE XA. Appears in 5.1/5.0.
- Optimize TRUNCATE TABLE to do DROP + CREATE. Appears in 5.1.

InnoDB

Upcoming features

http://www.innodb.com/todo.php

- Implement LOCK TABLES TRANSACTIONAL. As you know, in the ordinary MySQL LOCK TABLES you must lock every table that you are going to use. This new SQL command will be free from such restrictions, and the also InnoDB internal deadlock detection will work on it. The command can be used like the similar command in DB2, Oracle, etc.
- Use asynchronous file I/O on modern Linux kernels. This may improve disk I/O performance by 10 %.
- Add to InnoDB Hot Backup a backup daemon and a backup server process that can write the backup over a socket to another computer.

InnoDB

Upcoming features

http://www.innodb.com/todo.php

- Implement 'semi-synchronous' replication to MySQL. The replication master acknowledges a transaction COMMIT to a client only when the slave has received the binlog segment of the transaction. No transaction is lost if the master crashes. This is useful in building an InnoDB/MyISAM failover 'cluster' using MySQL's ordinary replication mechanism.
- In crash recovery, do the rollback of uncommitted transactions in the background. This speeds up recovery times.
- Implement SHOW INNODB LOCK STATUS.
- Enable the built-in multi-threaded rollback and purge in InnoDB.

InnoDB

Upcoming features

http://www.innodb.com/todo.php

- Implement more intelligent index read-ahead capabilities.
- Fast COUNT(*) from a table.
- Allow innodb_buffer_pool_size and innodb_log_file_size to be settable online.
- Allow new ibdata files to be added online.
- Fast, background index creation.
- Online reorganization of a table.
- Fulltext indexes.
- Partitioned tables.
- BMC Patrol Knowledge Module for monitoring MySQL/InnoDB.

Questions from audience

Thank you