

# *InnoDB*

## *New InnoDB Features*

Heikki Tuuri / Innobase Oy

[www.innodb.com](http://www.innodb.com)

[heikki.tuuri@innodb.com](mailto:heikki.tuuri@innodb.com)

Talk at MySQL Users Conference 2005, Santa Clara

# *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
- 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
- *Mr/Ms ? ?* will implement **FULLTEXT** indexes

# *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; thank you to Ryan Huddleston!
- Allows you to place each InnoDB table into its own **.ibd** file; in this respect similar to MyISAM

# InnoDB

7

## *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

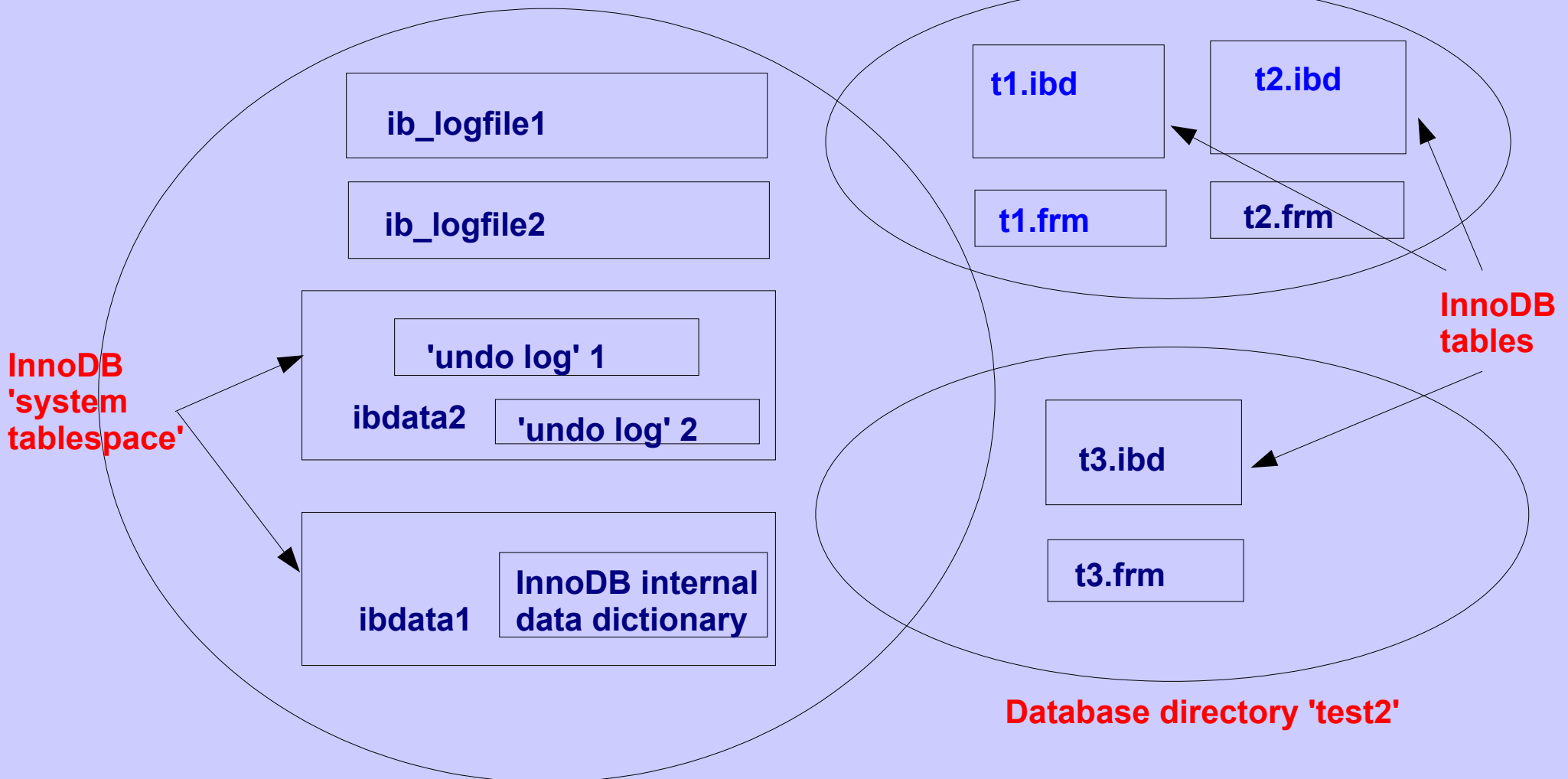
# InnoDB

## Multiple tablespaces (cont.)

Files when using multiple tablespaces

MySQL 'datadir'

Database directory 'test'



## *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.)

# InnoDB

## *Advantages of multiple tablespaces*

- 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.
- 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.
- No clear disadvantages.

## *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.

## *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'.

## *Compressed InnoDB table formats*

- Marko Mäkelä has reduced the space usage of InnoDB tables by about 20 % in MySQL-5.0.3. Marko has accomplished this saving by removing unnecessary column lengths stored into each InnoDB record.
- Marko will also implement a transparent, on-the-fly zip-like compression that will reduce space usage a further 50 – 90 %. This will appear in MySQL-5.1. In a test, a set of real-world data from a customer actually fit in 10 % space after the compression!

# InnoDB

## Compressed InnoDB table formats

- In MySQL-5.0.3 and later, 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 2 - 8 kB on disk (the size will 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.

## Compressed InnoDB table formats

- Under a write-intensive workload, compression may use almost the power of one CPU. Under a read-intensive workload, maybe 30 % of one CPU.
- 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 very fatal: if a single bit has changed in corruption, you lose the whole page

## 'Semi-synchronous' replication

- As Brian Aker mentioned yesterday, the MySQL binlog replication is 'asynchronous'. This means that the slave may not have even received a transaction that was committed in the master and for which the client has been informed of a successful commit.
- The above makes failover hard. If the master crashes and we let a slave take over, then the application must be prepared to check which transactions actually made it to the slave, and rerun the ones that did not!

## *'Semi-synchronous' replication*

- Solution: before telling the client that a transaction has been committed, make sure that the slave receives its binlog segment first. This is called 'semi-synchronous' replication.
- We can add different degrees of synchronicity: fully synchronous replication would wait for the slave to process the transaction first, before telling the client that it has been committed. The downside: delays in commits.

## 'Semi-synchronous' replication

- A patch for semi-sync was committed today. We are talking about a really new feature now :).

<http://lists.mysql.com/internals/24172>.

May appear in 5.0.x after a review. The patch is not fool-proof, because the TCP/IP protocol does not guarantee that the data really has been received by the slave. A better replication protocol is needed to make that certain.

## *'Semi-synchronous' replication*

- How to write a failover script? People would like to hear that database failover is possible without any changes in the application. Unfortunately, implementing such failover in the database server or a cluster makes the server slow, because all information must reach 2 nodes before the operation can be said to have succeeded.
- In a more efficient solution, the application must be prepared for a situation where the connection to the database server breaks, and the application must be able to rerun the whole transaction in the failover server.

## 'Semi-synchronous' replication

- Imagine that the connection to the server breaks just when the application is waiting for the server to tell that a **COMMIT** was processed ok. In that case, the client cannot know whether the transaction made it to the failover server or not! The client must ask from the failover server whether a transaction with **ID 8736487546** actually was committed there.

## 'Semi-synchronous' replication

- Performance advantage of semi-synchronous replication when compared to more distributed failover solutions:
  - \* Updates are made in a single master database node: no need for slow inter-node communication. Multi-core processors will make communications within a single computer even faster, and thus further add to the advantage.
- Unlimited scalability for read-only transactions who do not need the absolute latest data: they can be run in slave nodes.

# InnoDB

## *Upcoming features*

24

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

- Add support for a two-phase commit of distributed transactions. The interface is as in J2EE XA. Appeared in 5.0.3.
- 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.)

## Upcoming features

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

- Implement 'transactional' **LOCK TABLE t IN {SHARE | EXCLUSIVE} MODE**. 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 then also InnoDB internal deadlock detection will work on it. The command can be used like the same command in DB2, Oracle, etc. The patch has already been written. May be included in 5.0 or 5.1.

## *Upcoming features*

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

- Add to InnoDB Hot Backup a backup daemon and a backup server process that can write the backup over a socket to another computer. Appears in May 2005.

## *Upcoming features*

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

- Use asynchronous file I/O on modern Linux kernels. This may improve disk I/O performance by 10 %. The patch was written by Christoffer Hall-Frederiksen at the Univ. of Copenhagen. Appears in 5.1.

## *Upcoming features*

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

- **FULLTEXT** indexes for InnoDB: the financing exists, we are looking for a guru to implement them. Should be ready April 2006. Salary is competitive. Any volunteers :)?
- Partitioned tables: Mikael Ronström has promised them in 5.1 for all MySQL table types. They make mass deletion of old data in a table almost instantaneous.

## Upcoming features

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

- Allow new `ibdata` files to be added online. (2006)
- Fast `COUNT(*)` from a table. (2006?)
- Implement more intelligent index read-ahead capabilities. (2007?)
- Allow `innodb_buffer_pool_size` and `innodb_log_file_size` to be settable online. (2007?)
- Online index creation (2007?).
- Online reorganization of a table (2008?).  
Can reuse the code from online index creation.

# InnoDB

30

## *Upcoming features*

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

- Implement **SHOW INNODB LOCK STATUS**. (2006?)
- Enable the built-in multi-threaded rollback and purge in InnoDB. (2007?)

***Questions from audience***

***Thank you***