

*Fast, Consistent, Online Backups for
MySQL*

INNODB®



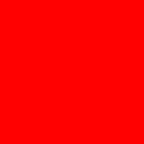
ORACLE®

Backup Strategies with MySQL Enterprise Backup

John Russell
Oracle/InnoDB

Calvin Sun
Oracle/InnoDB

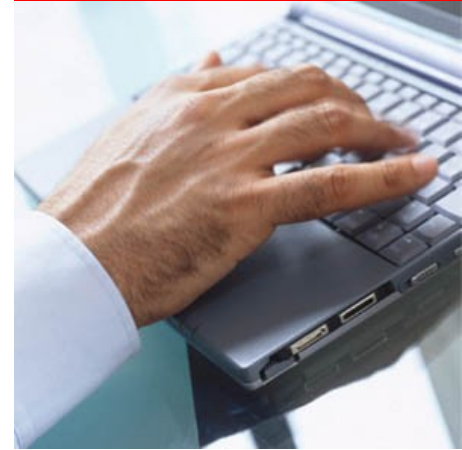
Mike Frank
Oracle/MySQL



The preceding is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Agenda

- What is “MySQL Enterprise Backup”?
- How It Works - Backup
- How It Works - Restore
- Incremental Backup
- Partial Backup of InnoDB Data
- Backup Tips and Strategies
- MySQL Enterprise Backup Availability



What is MySQL Enterprise Backup?

- A feature of “MySQL Enterprise”
- Formerly known as “InnoDB Hot Backup”, a backup tool for InnoDB
- Backs up running databases without locking InnoDB data
- Supports InnoDB and MyISAM
- Features:
 - Compressed backup
 - Partial backup
 - Point-in-time recovery

New Features in 3.5

- Incremental backup
- Support of Barracuda file format
 - Backup of compressed tables
- Backup of additional files, such as partition files
 - Backup of in-memory database with `--exec-when-locked` option



Email mike.frank@oracle.com to register for beta.

How It Works - Backup

Backup steps:

- 1) Backup data files
- 2) Copy log records created during data file copy
- 3) Backup MyISAM files with innobackup
- 4) Additional user-defined action under --exec-when-locked option

Step 1: Backup Data Files

- Copy and compress data files
 - Produces “fuzzy” backup
 - ✓ Backup of data files doesn't correspond to any specific log sequence number (LSN)
 - ✓ Different database pages are copied at varying times
- Omit free space in blocks & empty pages
- Backup files are typically 30% smaller
- Note oldest and newest LSN

Step 2: Copy Log Records

- All redo records with LSNs during data file copy
- Portions of the log file that contain all the required redo information are copied
- Includes time from beginning to end of backup
- All data blocks that were modified after they were copied can be recovered

InnoDB Logfile

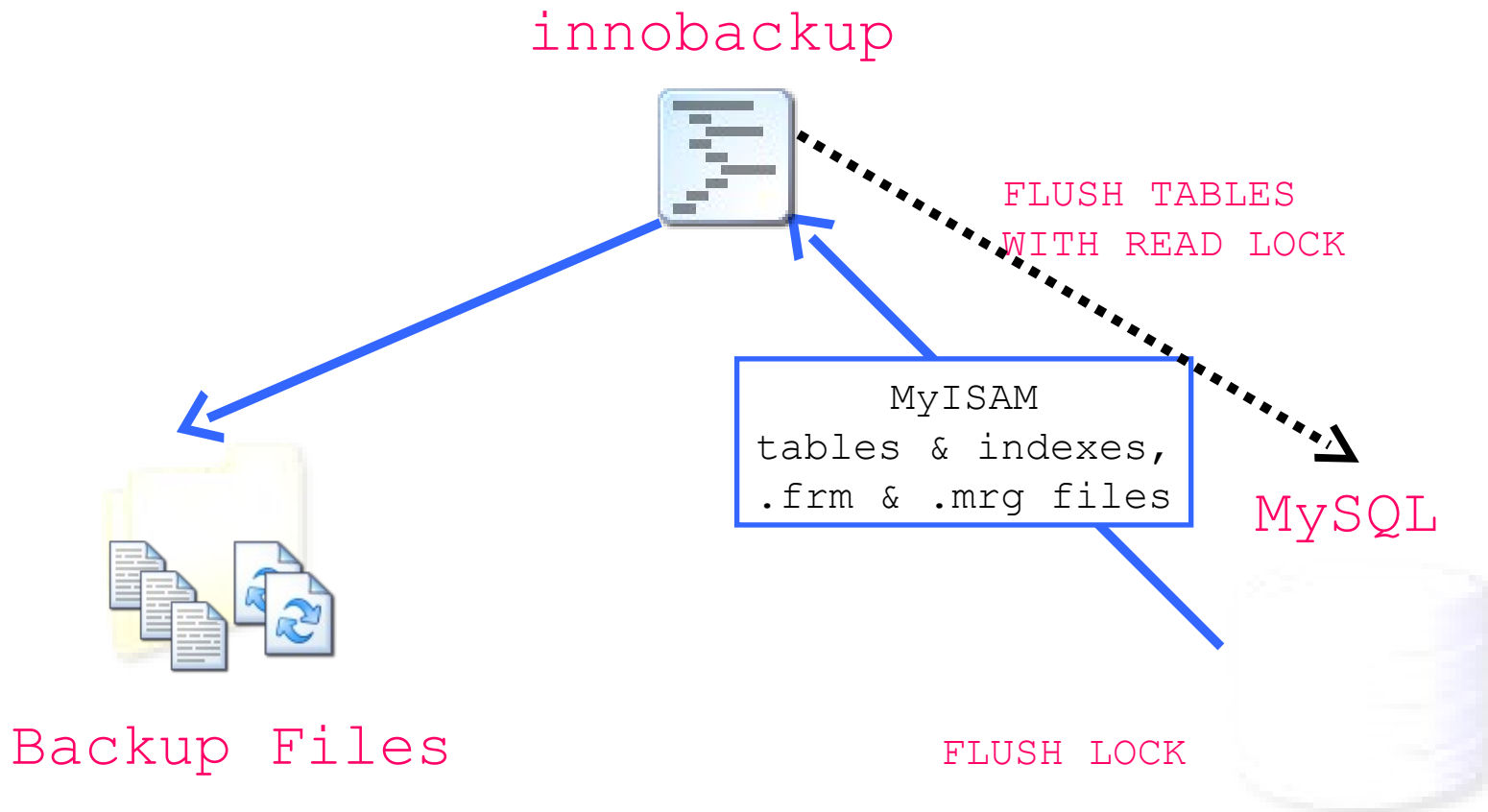


ibbackup_logfile



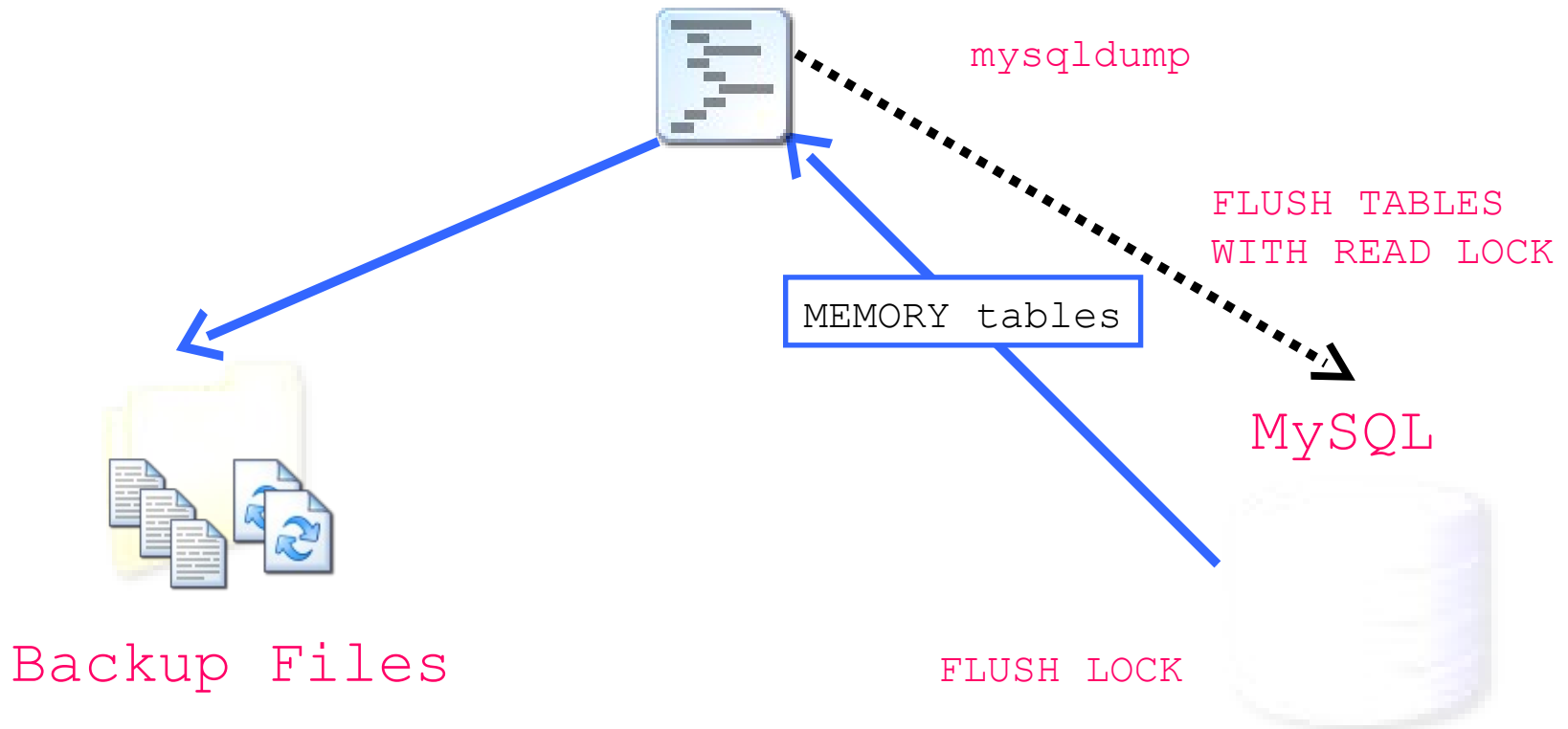
ORACLE®

Step 3: Backup MyISAM Files



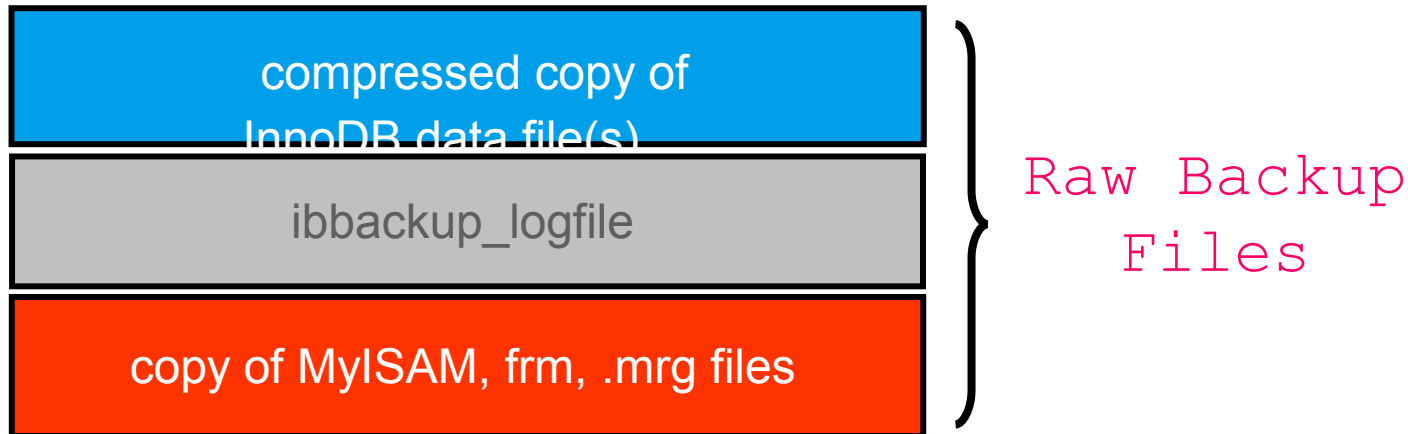
Step 4: Backup MEMORY Tables

Innobackup with
`--exec-when-locked`



“Raw Backup” Files

- The “raw backup” files from backup phase cannot be directly consumed by MySQL
- These files can be copied to media
- The database must be “restored” first



How It Works - Restore

Restore steps:

- 1) Backup files are uncompressed and restored into specified data directory
- 2) InnoDB logfiles are recreated
- 3) Log files are applied
- 4) MyISAM and other files are restored

Restoring a Database – “Apply Log” Phase

Backup Files



Data dir



InnoDB data files
uncompressed

ibbackup_logfile restored

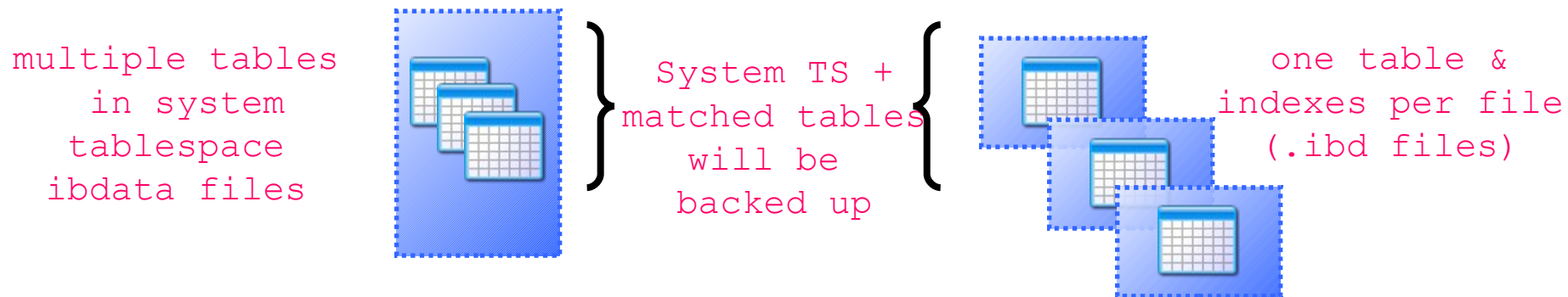
MyISAM, .frm, .mrg files restored

Incremental Backup

- A type of “hot backup”
- Only data changed since some point in time (last backup) are backed up
- Smaller backup file size and faster to backup
- Make the process more complex
- Take more time to carry out the restore process
 - Full backup and incremental backups must be restored in order
- Can be used for differential backup

Partial Backup of InnoDB Data

- When using the “file per table” option, backup a subset of tables
- Use regular expressions to specify tables (`--include`), or `--databases` option
- Tables in system tablespace plus individual tables that match the pattern are backed up



Backup Tips & Strategies

ibbackup Command

- Many parameters (compression, incremental, partial).
- InnoDB tables only; always system TS.
- Doesn't hold up any DB operations.
- Needs OS-level read permissions for files.
- Doesn't overwrite output files. Specify new directory each time.
- No .FRM files. Back these up separately via innobackup.
- You will probably end up scripting it yourself because of these last 2 points.

Applying the Log

- Backup is most important and tricky for big, busy databases.
- During the backup, changes happen in the database.
- The backup data includes data files + the logs to bring them fully up-to-date.
- Run `ibbackup` again with `--apply-log` option to form a complete backup.
- If backup is compressed, it gets uncompressed at this point.

innobackup Command

- Adds another level of functionality and flexibility on top of ibbackup -- a “one-stop shop”.
- Includes .FRM files for InnoDB tables.
- Includes data and .FRM files for MyISAM tables; this part does block DB operations.
- Stores backups under timestamped directories.
- Uses database credentials.
- Can perform backup and apply log in one step.

Compression

- Backups can be compressed (level 0-9).
- Can keep more backup data on hand.
- Applying the log uncompresses - need space for both compressed and uncompressed.
- When testing backups, evaluate time and space considerations for uncompressing.

Incremental

- Don't have to take full backup every time.
- Incremental backup copies pages changed since LSN of previous backup.
- For minimal size, can keep track of LSN and specify new one each time.
- For simplicity, could always use LSN of full backup, and take a new full backup periodically.
- Testing: apply all incremental backups to full backup, then test restore.

Partial

- You always get the InnoDB system tablespace and any tables inside it.
- In addition, can specify only selected databases or databases+tables.
- Can also specify tables whose names match a regular expression pattern.
- To include/exclude at the table level, must set `innodb_file_per_table=1` so these tables are in separate `.ibd` files.

Partial - Restore Considerations

- The always-present system tablespace makes restoring a partial backup something of an expert operation.
- To make sure all changes are reflected in .ibd files, run mysqld on the restored data and let it perform all recovery.
- Ideally, restore a single InnoDB table (a .ibd file) via `ALTER TABLE ... DROP TABLESPACE` and `ALTER TABLE ... IMPORT TABLESPACE`.

Restore

- ibbackup does not have restore capability; would have to script that yourself.
- innobackup can do the restore for you:
 - Shut down the database.
 - innobackup with --copy-back option.
 - Copies everything back to its original location.
 - Restart the database.
 - Can also restore to different location, just specify directory in a .cnf file.

Configuration Files

- Always need 1, often 2, MySQL configuration files.
 - One to specify original file locations and properties.
 - One to specify location for backup files.
 - Maybe more for test restores.
- Even if you have to construct them from scratch and deduce file size parameters.
- For typical installations, fill in the same path to the data dir 3 times in each.
- The “backup-my.cnf” file isn’t picky about [mysqld] header and only examines certain options.

Strategy Decisions: 1

- Compress or not?
 - More CPU during backup.
 - Can keep more sets of data on same box.
 - Faster to transfer to another box.
 - Unless transfer does compression too.
 - More space, CPU to uncompress for apply log.
 - More time during restore if only uncompress at that point.

Strategy Decisions: 2

- Incremental or not?
 - Seems like no-brainer if incremental backup data significantly smaller than full data.
 - Some trouble to keep track of LSNs.
 - Less space savings if run several backups from the same LSN.
 - Might have to run several apply steps before restoring.
 - Thus, for good testing, probably do apply after each incremental backup.

Strategy Decisions: 3

- Full or partial?
 - Partial backup much trickier to restore.
 - Restoring of partial backups mainly possible for individual tables.
 - Plus run mysqld first on restored data to clean up partial backups.
 - Thus full backup should be default strategy, with partial backups only for more frequent backups of critical tables.

Strategy Decisions: 4

- How to configure database for convenient backups?
 - Set `innodb_file_per_table` option so every table is in its own `.ibd` file.
 - Smaller files easier to work with.
 - Don't get giant system tablespace with empty space in the middle.
 - Required for some types of partial backups.
 - Set up instance with a `my.cnf` that specifies parameters needed for backup.

Strategy Decisions: 5

- When to do backups?
 - InnoDB-only: any time that CPU and I/O are lightly loaded enough. Database can be busy.
 - InnoDB + MyISAM: need the MyISAM tables to be lightly loaded.
 - Run backups at a time when ALTER TABLEs are not being performed.
 - But good idea to have backups of before & after ALTER TABLE, DROP TABLE.

Strategy Decisions: 6

- When/how to restore?
 - Restore on test servers to verify backup data.
 - Disaster recovery:
 - Keep backup data uncompressed for fastest restore.
 - Keep backup data with all logs & incremental backups applied for fastest restore, comprehensive testing.
 - Point-in-time restore: restore earlier backup, then replay binlog up to desired time.
 - Compressed backups & unapplied incremental backups for restores that are less time-sensitive.


MySQL Enterprise Backup Availability

MySQL Enterprise Backup 3.1

- Rebranded - Hotbackup 3.0
- Includes MySQL Enterprise Backup Installers and Packages
- Also, executable version on Innobackup functionality
 - Supports Windows and all other platforms
- General Availability: Summer 2010

MySQL Enterprise Backup 3.5

- Beta - Starting Now
 - Send requests to mike.frank@oracle.com
- Includes
 - Barracuda file format support & backup of compressed tables
 - Incremental Backup
- General Availability: stay tuned!



Q&A
QUESTIONS
ANSWERS