nbackup evolution

Backup and restore: overview

- Database backup and restore
  - “Logical”
    - Backup: export all data and metadata
    - Restore: create new database and populate it with metadata and data from backup file
  - “Physical” at OS\filesystem level
    - Backup: create copy of database file(s) at filesystem level
    - Restore: no need - filecopy is ready to use
  - “Physical” at page-level
    - Backup: create page-by-page copy of database file(s)
    - Restore: copy is ready to use

Backup and restore: overview

- Logical (gbak) backup\restore
  - Pros
    - Restored database is always in newest format (ODS) supported by the engine
    - Useful for migration between engine versions
      - Both upgrade and downgrade is possible
    - New (restored) database could occupy less space, data is not fragmented
  - Cons
    - Performance is far from optimal
    - Restore must build all indices
    - Whole database is processed

Backup and restore: overview

- Physical backup\restore at OS\filesystem level
  - Pros
    - Fastest
  - Cons
    - Impossible for active database – filecopy will not be consistent
    - Whole database should be copied
    - *Rumors: original database could be damaged*
      - *Never confirmed*

Backup and restore: overview

- Physical backup\restore using volume snapshots
  - Pros
    - Fast
    - Differential\incremental backups (if supported by OS)
  - Cons
    - Requires support from engine to make consistent on-disk image
    - Restored whole volume
    - Whole volume is processed
    - Whole database is processed

Backup and restore: overview

- Physical backup\restore at page-level
  - Pros
    - As fast as file-level copy
    - Incremental backup is possible
    - No need to build indices at restore
  - Cons
    - Not applicable for migration between engine versions
    - Database is not compacted
    - Data is not defragmented

Physical backup

- nbackup
  - Original idea is of Sean Leyne
  - Designed and implemented by Nickolay Samofatov
  - Funded through a grant from BroadView Software, Inc.
  - Introduced in Firebird 2.0

Physical backup

- Main goals:
  - Possibility to “freeze” database file on-disk contents while backup working
    - Any existing utility could copy database file
    - File on disk should be consistent
    - File copy is ready to use database
  - Make differential\incremental backup
  - Safety
    - Server shutdown and\or crash during backup should not corrupt database
    - After restart server should be able to continue to work with database

Physical backup

- IO redirection, delta file
  - During backup all writes (and some reads) are redirected into separate file
  - Delta file contains:
    - Changed database pages
    - Mapping table between delta and database

Physical backup

- Physical backup state
  - Needs to make IO redirection work correctly
  - Possible values
    - Normal
    - Stalled
    - Merge
  - Stored at Header page
    - gstat

Physical backup: Backup State

- Normal
  - Backup is not running
  - IO is not redirected
  - No delta file

Physical backup: Backup State

- Stalled
  - Backup is in progress
  - IO is redirected into delta-file:
    - All writes go to delta-file
    - Changed pages are read from delta file
    - Not changed pages are read from database file

Physical backup: Backup State

- Merge
  - Backup is finishing
  - Changes are merged into main database file
    - All changed pages are copied from delta to database file
  - IO is still redirected:
    - Changed pages are written to both database and delta
    - Not changed pages are written to database file only
    - Changed pages are read from delta file
    - Not changed pages are read from database file

Physical backup: Allocation Table

- How to detect IO read source\write target ?
  - Page Allocation Table
    - Contains numbers of pages changed since backup start
    - Stored in delta-file
    - Fully cached in memory
  - Read page N:
    - Lookup for N in Allocation Table
      - Found: read page from delta
      - Not found: read page from database
  - Write page N:
    - Lookup for N in Allocation Table
      - Not found: store N into Allocation Table
    - Write page into delta (and into database)

Physical backup: kind of backups

- Full backup
  - Backup all pages in database
    - Could be done using any file copy utility
- Differential backup
  - Backup pages changed since some previous full backup
- Incremental backup
  - Backup pages changed since some previous backup of any level
  - Multilevel
    - Full backup is level 0 backup
    - Increment since full backup is level 1 backup
    - Increment since level N backup is level N+1 backup

Physical backup: System Change Number

- Is page changed ?
  - System Change Number (SCN)
    - Incremented when Backup State is changed
    - Stored at Header page
    - Every database page is marked with current SCN value
    - Current SCN value is assigned to the page when it changed
  - Compare SCN of given page with SCN of previous backup
    - After each successful backup record about it is put into RDB$BACKUP_HISTORY
    - Each backup is marked by SCN that was current before backup started

Physical backup: operations

- Begin of backup
  - Utility (nbackup, isql, etc)
    - Attach to database
    - Run ALTER DATABASE BEGIN BACKUP statement
  - Engine
    - Create delta file
    - Page cache(s) is flushed to disk
    - Backup State is changed from “normal” to “stalled”
  - Utility
    - Detach from database (optional)

Physical backup: operations

- Produce backup
  - copy, xcopy, etc – full backup
  - nbackup
    - Full backup (level 0)
      - Copy database file page by page
    - Incremental backup of level L > 0
      - Query RDB$BACKUP_HISTORY for SCN of previous backup with level L - 1
      - Read database file and put into backup pages with SCN greater than found SCN of previous backup

Physical backup: operations

- End of backup
  - Utility
    - Attach to database (if not attached)
    - Put record into RDB$BACKUP_HISTORY
    - Run ALTER DATABASE END BACKUP statement
  - Engine
    - Backup State is changed from “stalled” to “merge”
    - Copy (merge) pages from delta into database
    - Backup State is changed from “merge” to “normal”
    - Delta file is deleted
  - Utility
    - detach from database

Physical backup: recovery

- Every new attachment look at backup state
  - Normal, Stalled
    - No extra actions needed
  - Merge
    - Run “merge” part of end backup process
      - Copy (merge) pages from delta into database
      - Backup State is changed from “merge” to “normal”
      - Delta file is deleted
    - Only one attachment
    - Run synchronously

Physical backup: synchronization

- Backup State lock
  - Control changes of Backup State
  - Fix Backup State while IO is in progress
- Allocation Table lock
  - Guard access to the Allocation Table
- End Backup lock
  - Allow only one process to end backup
- Implemented in Firebird Lock Manager
  - Supports different lock modes (Shared, Exclusive)
  - Supports lock caching and cross-process notifications (via AST handlers)

Physical backup: synchronization

- Backup State lock
  - Shared mode
    - While IO operation is in progress
      - IO direction should not be changed
    - While dirty page exists in cache
      - Consistency of database file on disk
  - Exclusive mode
    - When Backup State is changing

Physical backup: synchronization

- Allocation Table lock
  - Shared mode – read Allocation Table from delta file
  - Exclusive mode – add entry into Allocation Table
- Usage depends on Backup State
  - Normal: not used
  - Stalled: both Shared and Exclusive
  - Merge: Shared only

Physical backup: synchronization

- End Backup lock
  - Used in Exclusive mode only
  - Normal end of backup
  - Recovery check by every new attachment
    - Actually, run almost the same code as normal end of backup
    - Only one attachment performs merge

Physical backup: evolution

- Firebird 1.5
  - Initial development, private builds, no public releases
- Firebird 2.0
  - First public release

Physical backup: evolution

- Firebird 2.1
  - Synchronization reworked
  - Support for RAW devices on Linux

Physical backup: evolution

- Firebird 2.5
  - Synchronization reworked
    - Backup State lock is acquired early - for every page fetch, to avoid deadlocks
    - Attachment-private counters for state lock
  - Direct IO for database file scan
  - Forced Writes setting for delta file
  - Support in Services API

Physical backup: evolution

- Firebird 2.5.1, many bugs fixed
  - CORE-3466 : Some changes could be lost during the merge of delta file into main database file
  - CORE-3521 : Delta file contents is not flushed to disk
  - CORE-3535 : Write target of dirty page could be undefined if error happens when nbackup state is changed. Also prevent overwriting of first page of allocation table by data page contents.

Physical backup: evolution

- Firebird 2.5.3, bugs fixed, improvements
  - CORE-4431 : Reduce contention for allocation table lock while database is in stalled physical backup state
  - CORE-4432 : Let attachments to not block others when allocation table is read first time
  - CORE-4444 : Engine could hung and block all attachments in out of disk space condition during physical backup
  - CORE-4445 : Extend main database file faster when physical backup state changed from stalled to merge
  - Flush delta file implicitly before closing it.

Physical backup: evolution

- Firebird 3.0
  - Synchronization... is not reworked ;-)
  - Introduced SCN's inventory, allow to read pages changed since previous backup only and to not scan whole database
  - CORE-4462 : Make it possible to restore compressed .nbk files without explicitly decompressing them (Linux only)
  - CORE-4939 : Make IO operations with backup file aligned at page size boundary

Physical backup: evolution

- Firebird 3.0.1
  - Synchronization reworked, yes again ;-)
    - Backup State lock : in most cases local RW lock could be used instead of heavy LM's lock
    - No need to acquire Backup State lock at every page fetch

Physical backup: evolution

- Firebird 4.0
  - GUID-based backup and in-place restore
    - Allows to continuously “apply” increments to the read-only database
    - Not need to keep and apply all increments since full backup
    - Do not affect existing multilevel backup scheme
    - Could be used as kind of “physical replication”

Physical backup: evolution

- GUID-based backup...
  - Use backup GUID of target database as GUID of previous backup
    - *gstat -h <target database>*
  - Create backup
    - *nbackup -B <GUID> <source database> <backup file>*
      - Locate record in RDB$BACKUP_HISTORY using GUID of previous backup
      - Create backup using SCN from found history record
- ...and in-place restore
  - Apply backup to the target database
    - *nbackup -R -INPLACE <backup file> <target database>*

Physical backup: evolution

- Firebird 4+ : what else could be done
  - Re-think “direct IO” option, consider backup file too
  - Try to avoid state locking for dirty pages
  - Try to avoid cache flushing when backup state changing
  - Develop standby (cold- or even hot-) solution based on shipping and applying increments
    - Garbage collection and metadata consistency problems should be solved
  - Ship increments using network only

THANK YOU FOR ATTENTION

Questions ?

Firebird official web site

Firebird tracker

hvlad@users.sf.net