DB101 – Database consistency checking

Goetz Graefe – Madison, Wis.

Agenda

- Introduction
- In-page checking
- B-tree structure
- Indexes of a single table
- Cross-table checks
- Offline, online, continuous
- Remedies
- Summary

Topics omitted

1. Checking recovery and log archive
2. Free-space management, database catalogs
3. Bitmap indexes, hash indexes, spatio-temporal indexes
4. Column storage, read- and write-optimized storage
5. Log-structured merge-forests, stepped merge forests
6. Distributed databases
7. Privacy and security policies
8. Query processing: optimization and execution
9. Data quality and cleaning

Database consistency checks: why, when, what, how

- After hardware failures (e.g., HDD crash, SSD wear-out)
- As defense against software defects (e.g., locking & latching)

- Local checks (e.g., within a database page)
- Physical consistency (e.g., b-tree structure)
- Logical consistency (e.g., multiple indexes, multiple tables)

- Offline, online, continuous
- Database, backup, backup and restore pipelines

In-page checks

- All header fields (plausibility checks)
  - Index identifier & tree level
  - Record count & byte count
  - …
  - PageLSN (version & recovery information)
- All offsets and record sizes (plausibility, overlaps)
- Key ranges, null values, check constraints
- Sort order & uniqueness

B-tree structure

- Free-space management
- Leaf & branch nodes
- Branch keys & key ranges (uniqueness)
  both siblings and cousins
- Fence keys or neighbor pointers

Algorithm approaches:

- Index navigation
- Storage scan + “aggregation” of facts

Aggregation of facts

- Complementary pairs of facts, e.g.,
  - forward + backward pointers
  - fence keys in parent + child
  - …
- Sort- and hash-based grouping, parallel algorithms

- Database check at storage bandwidth – also database backups
- In-pipeline checks during backup and restore

Indexes of a single table

- Including primary storage structure or heap
- Counts of entries?
- Consistent values – anti-joins
- Bit vector filtering? Approximate pre-filter?

Cross-table checks

- e.g., foreign key integrity constraints, materialized views
- join, bit vector filtering (approx)

Offline, online, continuous

Offline algorithms:

- A scan for each check vs a single scan for all checks
- Disk-order scan, backup & restore pipelines

Continuous comprehensive consistency check:

- Side effect of query execution
- Only for top-down consistency constraints (e.g., fence keys)?

Online algorithms (also required for “fuzzy” backups):

- Database snapshot vs “facts” from updates

Remedies

- Drop & re-create (secondary index, materialized view)
- Media recovery (restore)
- Node recovery (failover, re-init)
- Single-page recovery (repair)

Summary and conclusions

Comprehensive consistency checks are

- required defense against software and hardware faults
- useful for databases, backups, and backup / restore pipelines
- local and global (in-page, cross-index, cross-table, cross-node)
- logical and physical (e.g., joins of indexes, b-tree structure)

Algorithms are

- offline vs online (e.g., continuous)
- iterative vs bulk (e.g., disk-order scans + parallel aggregation)