

# Midterm Topics

- Ch. 1 Overview of Database Systems
- Ch. 2 Intro. to Database Design
- Ch. 3 The Relational Model
- Ch. 4 Relational Algebra and Calculus
- Ch. 5 SQL: Queries, Constraints, ~~Triggers~~
- Ch. 8 Overview of Storage and Indexing

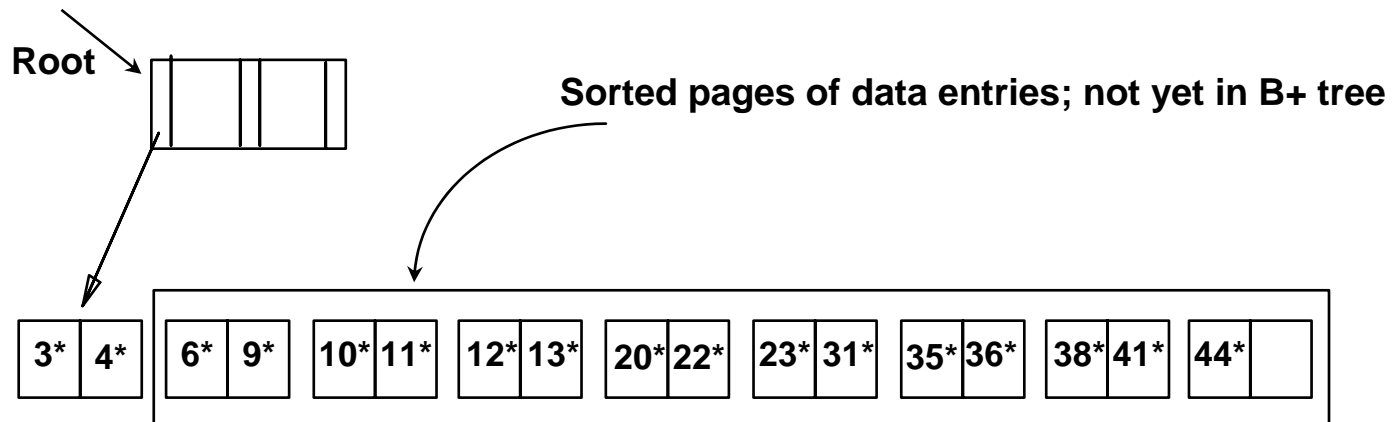
# B+ Tree (cont.)

# Prefix Key Compression

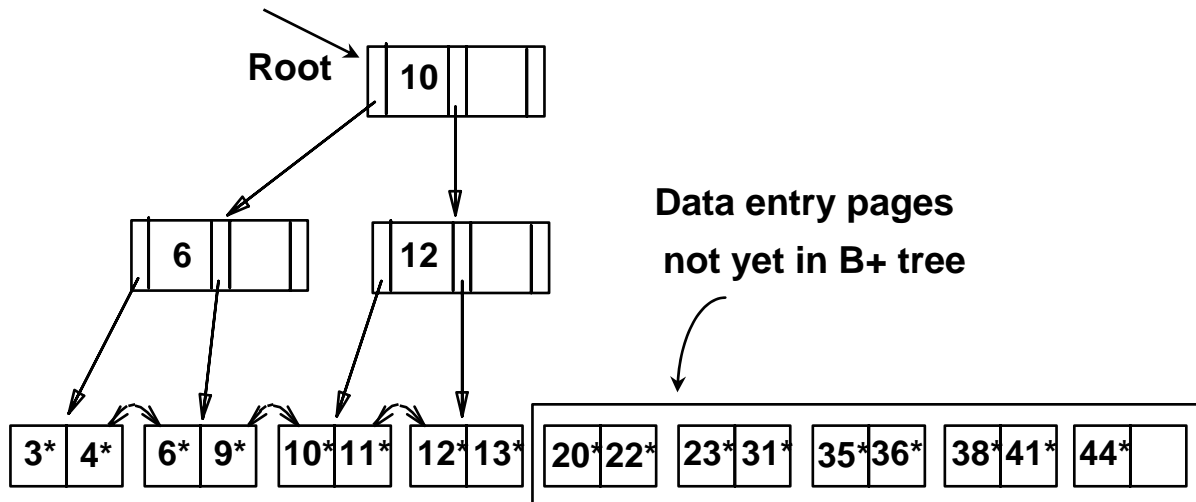
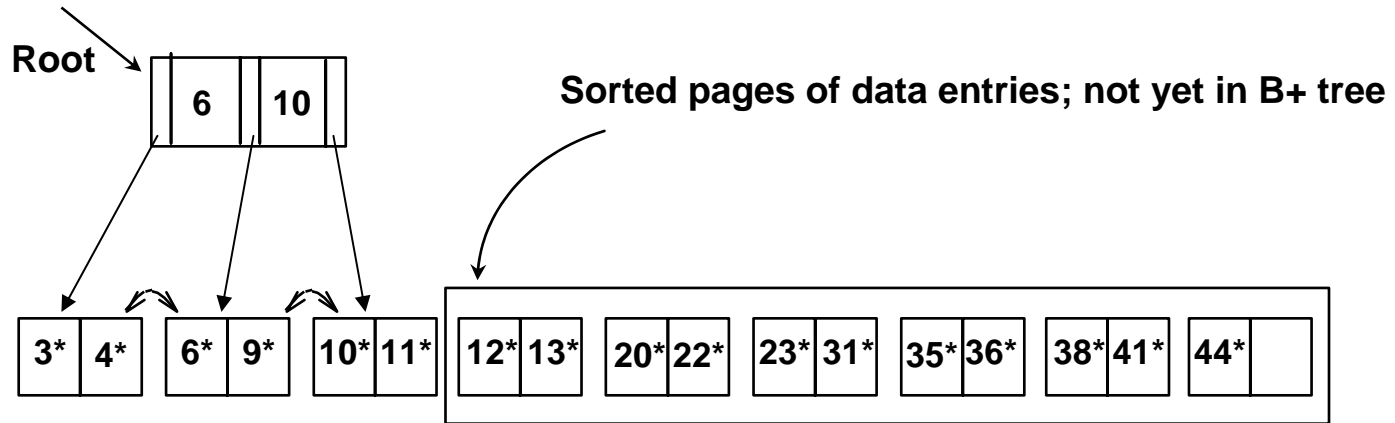
- Important to increase fan-out. (Why?)
- Key values in index entries only `direct traffic`; can often compress them.
  - e.g., If we have adjacent index entries with search key values *Dannon Yogurt*, *David Smith* and *Devarakonda Murthy*, we can abbreviate *David Smith* to *Dav*. (The other keys can be compressed too ...)
    - Is this correct? What if there is a data entry *Davey Jones*? (Can only compress *David Smith* to *Davi*)
    - In general, while compressing, must leave each index entry greater than every key value (in any subtree) to its left.
- Insert/delete must be suitably modified.

# Bulk Loading of a B+ Tree

- If we have a large collection of records, and we want to create a B+ tree on some field, doing so by repeatedly inserting records is very slow.
- Bulk Loading can be done much more efficiently.
- *Initialization*: Sort all data entries, insert pointer to first (leaf) page in a new (root) page.

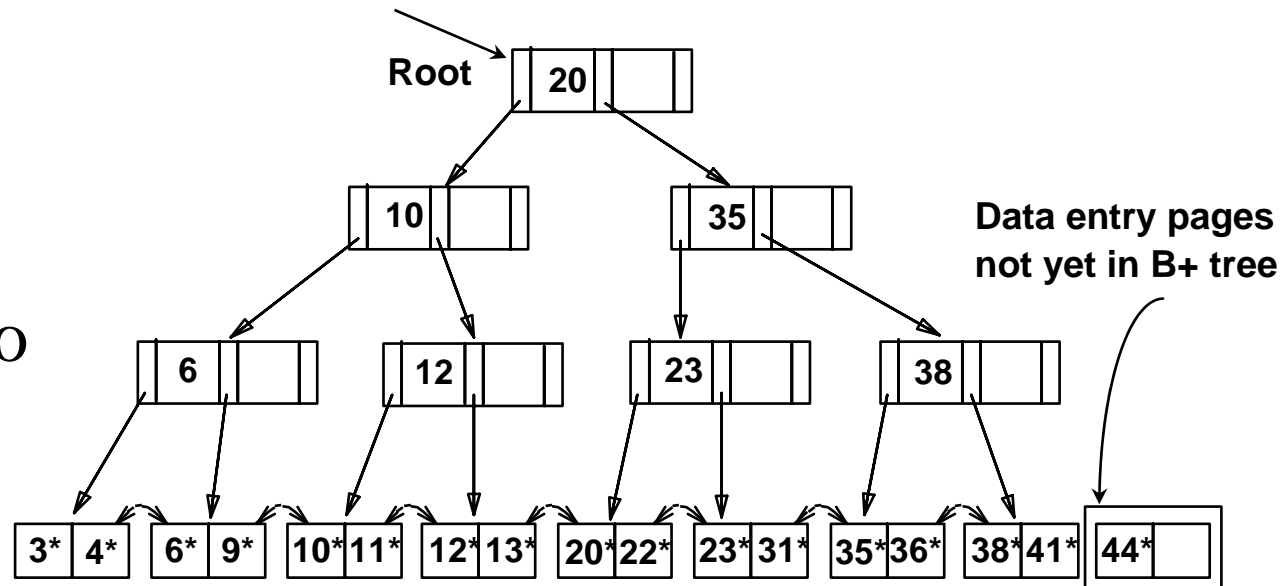
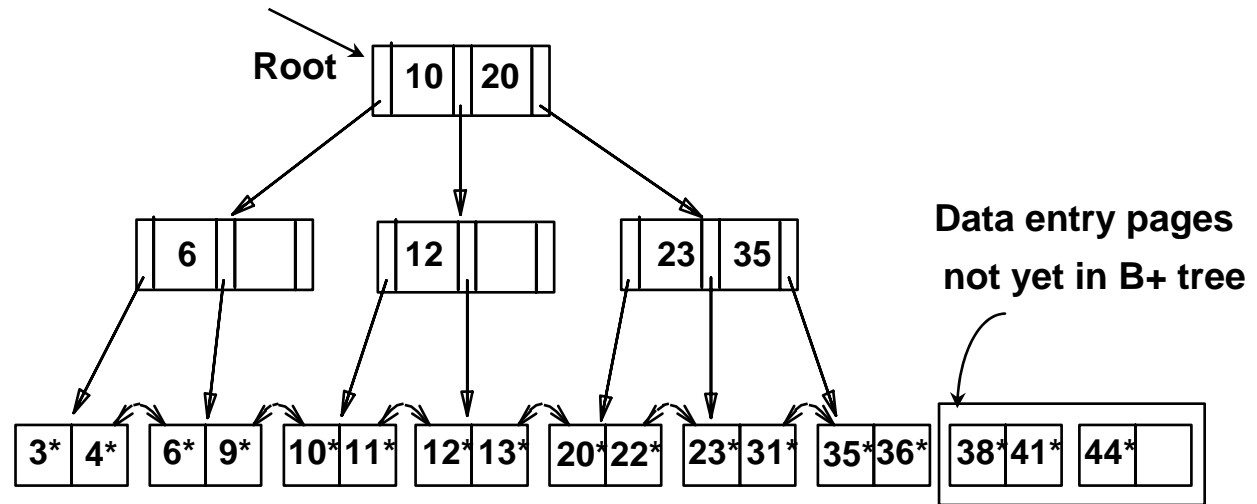


# Bulk loading (Cont.)



# Bulk Loading (Cont.)

- Index entries for leaf pages always entered into right-most index page just above leaf level. When this fills up, it splits. (Split may go up right-most path to the root.)



# Summary of Bulk Loading

- Option 1: multiple inserts.
  - Slow.
  - Does not give sequential storage of leaves.
- Option 2: Bulk Loading
  - Has advantages for concurrency control.
  - Fewer I/Os during build.
  - Leaves will be stored sequentially (and linked, of course).
  - Can control “fill factor” on pages.

# A Note on `Order`

- *Order* (**d**) concept replaced by physical space criterion in practice (*`at least half-full`*).
  - Index pages can typically hold many more entries than leaf pages.
  - Variable sized records and search keys mean different nodes will contain different numbers of entries.
  - Even with fixed length fields, multiple records with the same search key value (*duplicates*) can lead to variable-sized data entries.

# Summary

- Tree-structured indexes are ideal for range-searches, also good for equality searches.
- **ISAM** is a static structure.
  - Only leaf pages modified; overflow pages needed.
  - Overflow chains can degrade performance unless size of data set and data distribution stay constant.
- **B+ tree** is a dynamic structure.
  - Inserts/deletes leave tree height-balanced;  $\log_F N$  cost.
  - High fanout (**F**) means depth rarely more than 3 or 4.
  - Almost always better than maintaining a sorted file.

# Summary (Cont.)

- Typically, 67% occupancy on average.
- Usually preferable to ISAM; adjusts to growth gracefully.
- If data entries are data records, splits can change rids!
- Key compression increases fanout, reduces height.
- Bulk loading can be much faster than repeated inserts for creating a B+ tree on a large data set.
- Most widely used index in database management systems because of its versatility. One of the most optimized components of a DBMS.

# Ch.9 RAID

- **RAID: Redundant Arrays of Inexpensive Independent Disks**
  - disk organization techniques that manage a large numbers of disks, providing a view of a single disk of
    - high capacity and high speed by using multiple disks in parallel, and
    - high reliability by storing data redundantly, so that data can be recovered even if a disk fails

Originally a cost-effective alternative to large, expensive disks

- The chance that some disk out of a set of  $N$  disks will fail is much higher than the chance that a specific single disk will fail.
  - e.g., a system with 100 disks, each with MTTF of 100,000 hours (approx. 11 years), will have a system MTTF of 1000 hours (approx. 41 days)
  - Techniques for using redundancy to avoid data loss are critical with large numbers of disks.

# Improvement of Reliability via Redundancy

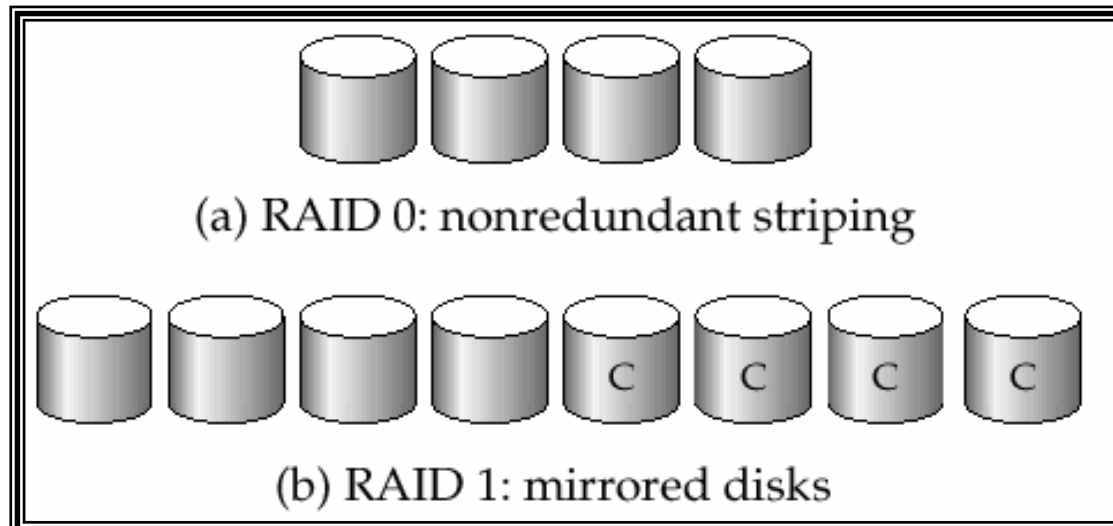
- **Redundancy** – store extra information that can be used to rebuild information lost in a disk failure
- e.g., **Mirroring** (or **shadowing**)
  - Duplicate every disk. Logical disk consists of two physical disks.
  - Every write is carried out on both disks
    - Reads can take place from either disk
  - If one disk in a pair fails, data still available in the other
    - Data loss would occur only if a disk fails, and its mirror disk also fails before the system is repaired
      - Probability of combined event is very small
        - » Except for dependent failure modes such as fire or building collapse or electrical power surges
- Mean time to data loss depends on mean time to failure and mean time to repair

# Improvement in Performance via Parallelism

- Two main goals of parallelism in a disk system:
  1. Load balance multiple small accesses to increase throughput
  2. Parallelize large accesses to reduce response time.
- Improve transfer rate by striping data across multiple disks.
- **Bit-level striping** – split the bits of each byte across multiple disks
  - In an array of eight disks, write bit  $i$  of each byte to disk  $i$ .
  - Each access can read data at eight times the rate of a single disk.
  - But seek/access time worse than for a single disk
    - Bit level striping is not used much any more
- **Block-level striping** – with  $n$  disks, block  $i$  of a file goes to disk  $(i \bmod n) + 1$ 
  - Requests for different blocks can run in parallel if the blocks reside on different disks
  - A request for a long sequence of blocks can utilize all disks in parallel

# RAID Levels

- Schemes to provide redundancy at lower cost by using disk striping combined with parity bits
  - Different RAID organizations, or RAID levels, have differing cost, performance and reliability characteristics
- **RAID Level 0:** Block striping; non-redundant.
  - Used in high-performance applications where data lost is not critical.
- **RAID Level 1:** Mirrored disks with block striping
  - Offers best write performance.
  - Popular for applications such as storing log files in a database system.



# RAID Levels (Cont.)

- **RAID Level 2:** Memory-Style Error-Correcting-Codes (ECC) with bit striping.
- **RAID Level 3:** Bit-Interleaved Parity
  - a single parity bit is enough for error correction, not just detection, since we know which disk has failed
    - When writing data, corresponding parity bits must also be computed and written to a parity bit disk
    - To recover data in a damaged disk, compute XOR of bits from other disks (including parity bit disk)



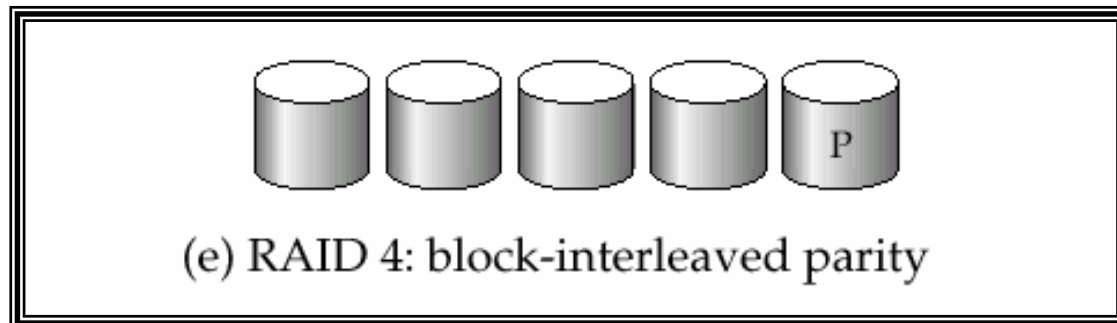
(c) RAID 2: memory-style error-correcting codes



(d) RAID 3: bit-interleaved parity

# RAID Levels (Cont.)

- RAID Level 3 (Cont.)
  - Faster data transfer than with a single disk, but fewer I/Os per second since every disk has to participate in every I/O.
  - Subsumes Level 2 (provides all its benefits, at lower cost).
- **RAID Level 4: Block-Interleaved Parity**; uses block-level striping, and keeps a parity block on a separate disk for corresponding blocks from  $N$  other disks.
  - When writing data block, corresponding block of parity bits must also be computed and written to parity disk
  - To find value of a damaged block, compute XOR of bits from corresponding blocks (including parity block) from other disks.



# RAID Levels (Cont.)

- RAID Level 4 (Cont.)
  - Provides higher I/O rates for independent block reads than Level 3
    - block read goes to a single disk, so blocks stored on different disks can be read in parallel
  - Provides high transfer rates for reads of multiple blocks than no-striping
  - Before writing a block, parity data must be computed
    - Can be done by using old parity block, old value of current block and new value of current block (2 block reads + 2 block writes)
    - Or by recomputing the parity value using the new values of blocks corresponding to the parity block
      - More efficient for writing large amounts of data sequentially
  - Parity block becomes a bottleneck for independent block writes since every block write also writes to parity disk

# RAID Levels (Cont.)

- **RAID Level 5:** Block-Interleaved Distributed Parity; partitions data and parity among all  $N + 1$  disks, rather than storing data in  $N$  disks and parity in 1 disk.
  - e.g., with 5 disks, parity block for  $n^{\text{th}}$  set of blocks is stored on disk  $(n \bmod 5) + 1$ , with the data blocks stored on the other 4 disks.

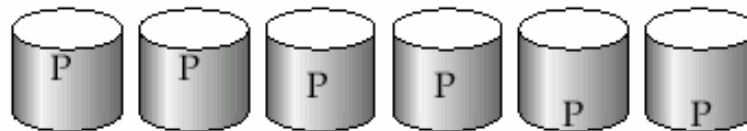


(f) RAID 5: block-interleaved distributed parity

P0	0	1	2	3
4	P1	5	6	7
8	9	P2	10	11
12	13	14	P3	15
16	17	18	19	P4

# RAID Levels (Cont.)

- **RAID Level 5 (Cont.)**
  - Higher I/O rates than Level 4.
    - Block writes occur in parallel if the blocks and their parity blocks are on different disks.
  - Subsumes Level 4: provides same benefits, but avoids bottleneck of parity disk.
- **RAID Level 6: P+Q Redundancy scheme; similar to Level 5, but stores extra redundant information to guard against multiple disk failures.**
  - Better reliability than Level 5 at a higher cost; not used as widely.



(g) RAID 6: P + Q redundancy

# Choice of RAID Level

- Factors in choosing RAID level
  - Monetary cost
  - Performance: Number of I/O operations per second, and bandwidth during normal operation
  - Performance during failure
  - Performance during rebuild of failed disk
    - Including time taken to rebuild failed disk
- RAID 0 is used only when data safety is not important
  - e.g. data can be recovered quickly from other sources
- Level 2 and 4 never used since they are subsumed by 3 and 5
- Level 3 is not used anymore since bit-striping forces single block reads to access all disks, wasting disk arm movement, which block striping (level 5) avoids
- Level 6 is rarely used since levels 1 and 5 offer adequate safety for almost all applications
- So competition is between 1 and 5 only

# Choice of RAID Level (Cont.)

- Level 1 provides much better write performance than level 5
  - Level 5 requires at least 2 block reads and 2 block writes to write a single block, whereas Level 1 only requires 2 block writes
  - Level 1 preferred for high update environments such as log disks
- Level 1 had higher storage cost than level 5
  - disk drive capacities increasing rapidly (50%/year) whereas disk access times have decreased much less (x 3 in 10 years)
  - I/O requirements have increased greatly, e.g. for Web servers
  - When enough disks have been bought to satisfy required rate of I/O, they often have spare storage capacity
    - so there is often no extra monetary cost for Level 1!
- Level 5 is preferred for applications with low update rate, and large amounts of data
- Level 1 is preferred for all other applications

# Hardware Issues

- **Hot swapping:** replacement of disk while system is running, without power down
  - Supported by some hardware RAID systems,
  - reduces time to recovery, and improves availability greatly
- Many systems maintain spare disks which are kept online, and used as replacements for failed disks immediately on detection of failure
  - Reduces time to recovery greatly
- Many hardware RAID systems ensure that a single point of failure will not stop the functioning of the system by using
  - Redundant power supplies with battery backup
  - Multiple controllers and multiple interconnections to guard against controller/interconnection failures