Sistemas de Bases de Dados

José Júlio Alferes
Departamento de Informática

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Objetivos

- Em Bases de Dados (1º ciclo) pretendia-se:
  - Que os estudantes fossem capazes de desenhar uma bases de dados relacional
  - Que conseguissem implementar uma base de dados relacional (centralizada) usando um sistema de gestão de bases de dados
  - Que dominassem o uso de linguagens de manipulação e interrogação de bases de dados
  - Que os estudantes soubessem os *fundamentos para desenharem uma base de dados e usarem um sistema de gestão de bases de dados relacional*
- Em Sistemas de Bases de Dados (2º ciclo) pretende-se que os estudantes conheçam as *"entranhas"* dum tal sistema de gestão de bases de dados
  - *"[…] bases necessárias à compreensão dos problemas envolvidos na construção e funcionamento de sistemas de gestão de bases de dados, dando ênfase à utilização eficiente de sistemas de bases de dados"*

Programa

- Implementação de sistemas de Bases de Dados
  - Armazenamento e estrutura de ficheiros
  - Indexação e hashing em bases de dados
- Processamento e otimização de perguntas
- Processamento de transações em bases de dados
  - Propriedades ACID
  - Controlo de concorrência
  - Sistemas de recuperação
- Arquiteturas de sistemas de gestão de bases de dados
  - Bases de dados distribuídas
  - Bases de dados paralelas
  - Sistemas Cloud-Based

Armazenamento de Dados

- Em Bases de Dados, o SGBD aparece como uma “caixa preta” que oferece um nível de abstração (nível lógico)
- Em Sistemas de Bases de Dados começa-se desvendando o que está dentro dessa caixa preta
  - Como é que as bases de dados são de facto armazenadas num computador
  - Com que estrutura de ficheiros em disco
  - Com que estruturas de dados para permitir acesso rápido
- Porque é importante saber isto?
  - Porque um Mestre em Eng. Informática não é um mero utilizador de sistemas, e deve conhecer os sistemas que utiliza
  - Porque só conhecendo minimamente como um SGBD é implementado se pode tirar partido eficiente dele
  - Porque só conhecendo as suas estruturas de dados internas se pode tirar partido do sistema e pode melhorar a sua eficiência

Processamento e Otimização de Perguntas

- Não basta saber como é que os dados são armazenados!
  - Um SGBD processa perguntas, em geral feitas numa linguagem declarativa de alto nível (e.g. SQL). Como é que são implementadas?
  - No 1º ciclo ensinou-se como usar a linguagem declarativa SQL para perguntas.
- Em SBD ensina-se como é que se implementam essas perguntas:
  - Como é que se passa duma pergunta SQL para código (imperativo) a executar sobre a base de dados? (processamento)
  - Como se podem transformar perguntas noutras equivalentes, que executem de forma mais eficiente? (otimização)
- Porque é importante saber isto?
  - Porque um Eng. Informático com Mestrado não é um mero utilizador e deve conhecer o que utiliza
  - Porque isto é essencial para usar as perguntas SQL de forma eficiente!
  - Em grandes bases de dados é essencial aferir da eficiência das perguntas

Processamento de Transações

- O conceito de transação é essencial em bases de dados
  - Só ele permite fornecer um nível lógico em que vários utilizadores acedem em simultâneo a uma base de dados
  - Na prática do dia a dia, os sistemas de bases de dados são acedidos por milhares de utilizadores, muitos deles em simultâneo
  - Como garantir que isto é possível sem gerar inconsistências?
- Em BD aflora-se muito brevemente o tema de transações.
- Em SBD aprofunda-se o tema quer do ponto de vista de uso em SGBD existentes, quer da sua implementação
- Porque é importante saber isto?
  - Porque para tirar verdadeiro partido das funcionalidades oferecidas pelos SGBD no que toca a acessos simultâneos há que conhecer o que estes disponibilizam, e como é que o implementam
  - Há também que ter consciência do que é (e do que não é) garantido pelo sistema

Arquiteturas de SGBDs

- Num mundo em que cada vez mais há grandes quantidades de informação disponível, e a informação está cada vez mais distribuída, não faz sentido pensar em bases de dados apenas como repositórios centralizados
- Existem várias arquiteturas de SGBDs, que serão estudadas em SBD
  - Bases de dados distribuídas e Cloud Based
  - Bases de dados heterogéneas
- Quando os dados não estão centralizados, o próprio processamento das perguntas e das transações não pode ser centralizado
  - Mas mesmo que os dados estejam centralizados, por eficiência pode fazer sentido paralelizar o processamento
- Porque é importante saber isto?
  - Porque nos dias de hoje as bases de dados distribuídas são uma realidade, e só estudando se podem conhecer

Avaliação

- 3 testes (ou exame de recurso) **sem consulta** a valer 75% da nota
  - Nota mínima para aprovação na disciplina: 10 valores
  - Teste a: 1 de abril, 6 de maio e 17 de junho

- Trabalho prático em grupo de 3 alunos
  - A fazer fora das aulas
  - Com ênfase em relatório
  - Tem uma apresentação oral
  - Notas de trabalhos de anos anteriores são válidas este ano

- O enunciado concreto do trabalho será apresentado a meio do semestre.

Trabalho

- A componente prática, que ao todo vale 25% da nota, contém:
  - Trabalho prático feito em grupo
  - Apresentação do trabalho (com nota individual)
  - Avaliação, individual, de outro trabalho
    - Anónima
    - Feita em formulário próprio
- Datas importantes:
  - 23 de maio: Entrega dos trabalhos
  - 31 de maio: Entrega de relatórios de avaliação
  - Na semana de 8 ou 12 de junho: Apresentação do trabalho

Recursos

- Bibliografia
  - Database System Concepts, **7th edition**, 2019

- Sistema
  - Nos laboratório está instalado o Oracle
  - Para o trabalho terá que instalar um outro SGBD (à sua escolha)
    - Há vários opensource (Postgres, MySQL, MariaDB, ...)
    - Como estudante tem acesso a outros comerciais (SQLServer, DB2)

- Outras recursos serão afixados:
  - No CLIP
  - No Google Calendar
  - https://sites.google.com/fct.unl.pt/sbd2020/

Funcionamento

- As aulas práticas começam depois da 2ª aula teórica
  - 13 de março para o P1
  - 17 de março para P2 e P3

- Não haverá aula teórica:
  - 27 de março
  - 17 de abril

- Aulas teóricas extra:
  - **18** de março
  - Outra data a indicar

Chapter 12: Physical Storage Systems

Brief overview of Physical Storage Media for Databases, to be aware of its incidence on the design of DBMSs

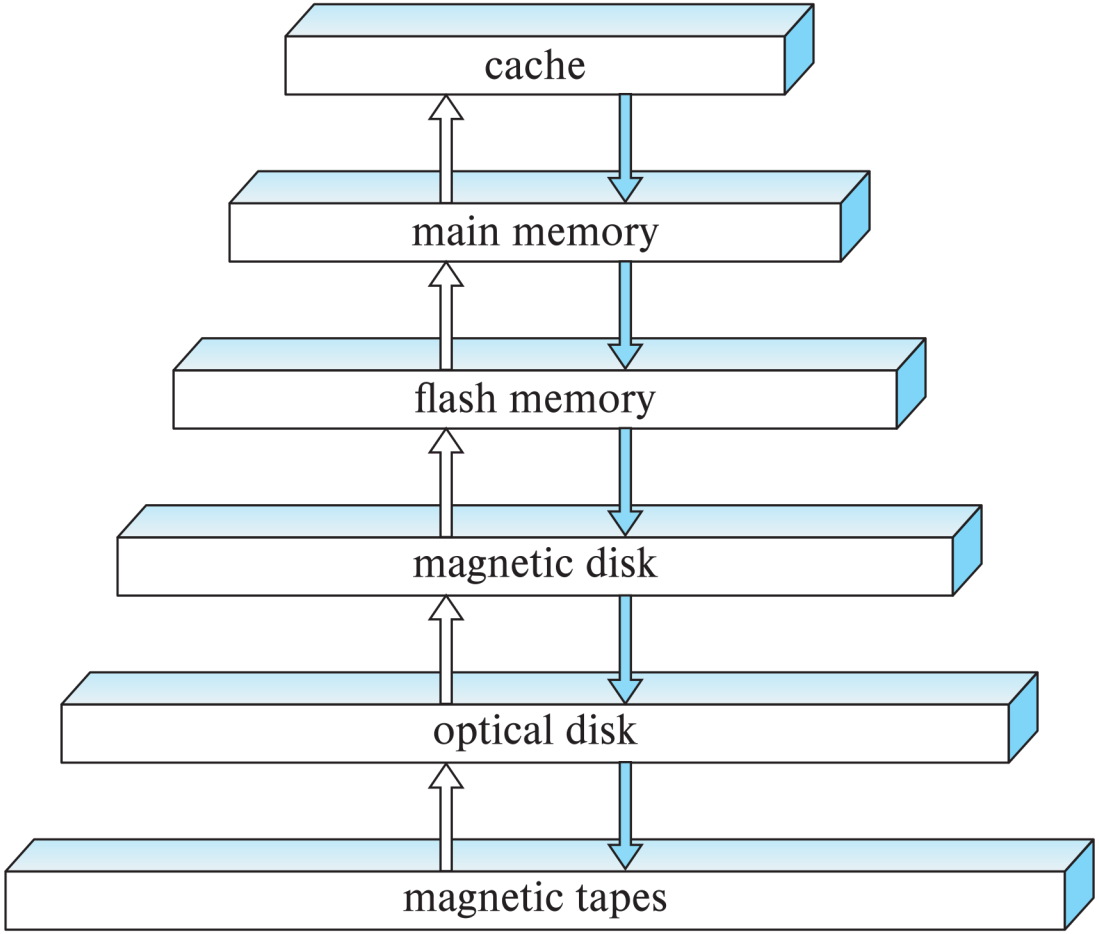
Sistemas de Bases de Dados 2019/20

Capítulo refere-se a: Database System Concepts, 7th Ed

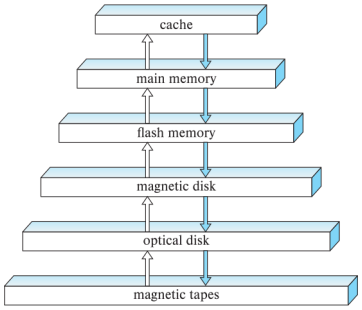
Classification of Physical Storage Media

- In the end, a database must be physically stored in computer(s)
- Can differentiate storage into:
  - **volatile storage:** loses contents when power is switched off
  - **non-volatile storage**:
    - Contents persist even when power is switched off.
    - Includes secondary and tertiary storage, as well as batter-backed up main-memory.
- Factors affecting choice of storage media include
  - Speed with which data can be accessed
  - Cost per unit of data
  - Reliability

Storage Hierarchy



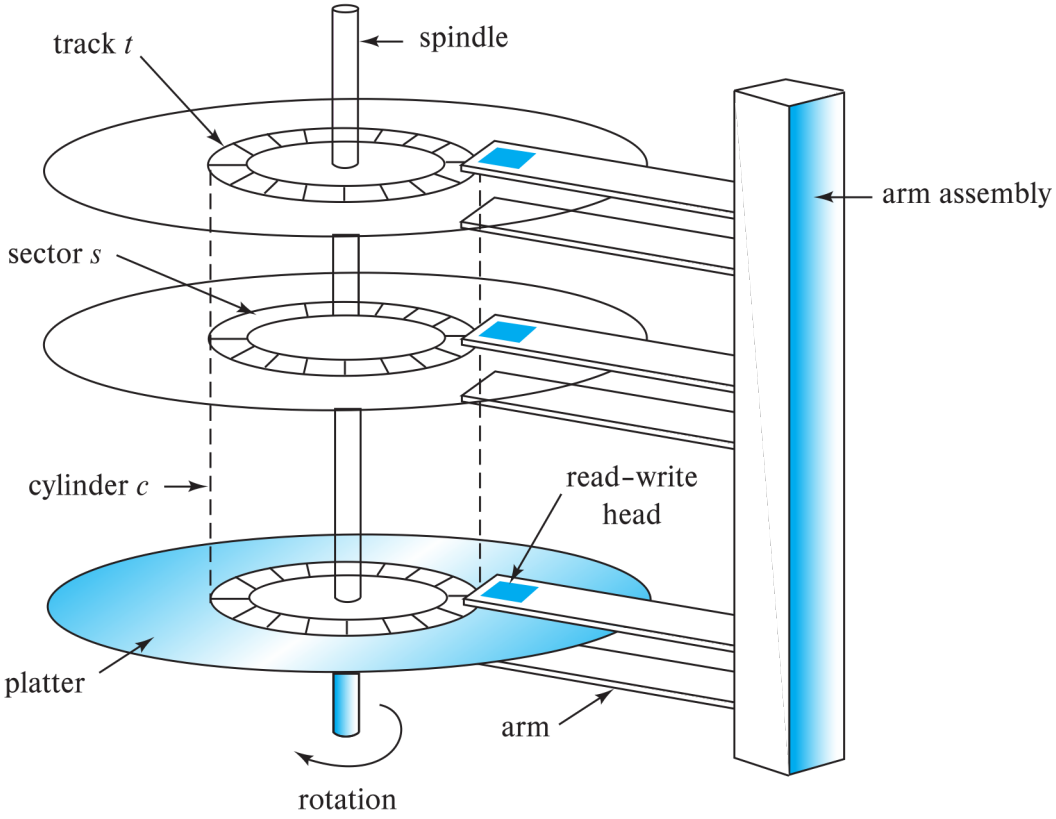
Storage Hierarchy (Cont.)

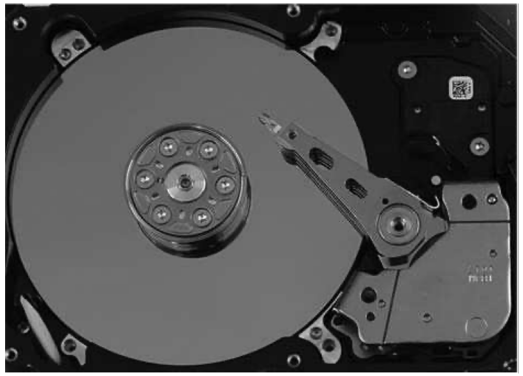


- **Cache and Main Memory**
  - Fast access (10s to 100s of nanoseconds)
  - Generally too small (or expensive) to store the entire database
  - **Volatile**
- **Flash memory**
  - Non-volatile fast read-access memory
  - Write is not as fast – and requires prior erasure
  - Limited number of erasures
  - Not yet with enough capacity or competitive price to be considered for big databases
- **Magnetic disks**
  - Still the medium of choice for database storage
  - Data moved to main memory for access, and written back to disk
  - Reliable
- **Optical disks and Tapes**
  - Mainly used for backups (off-line/archival storage)
  - Slow access time

Magnetic Hard Disk Mechanism

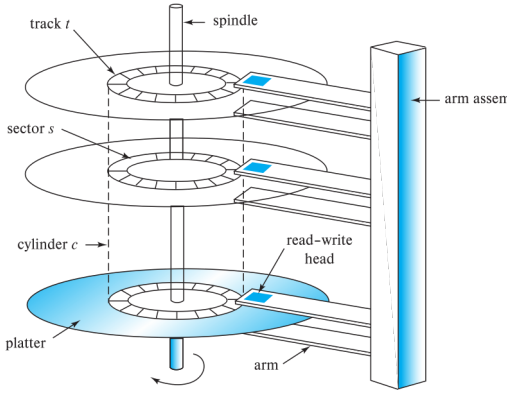
- It is worth looking at how magnetic disks work
  - After all they are the place where most databases are stored!





Magnetic Disks

- **Read-write head**
  - Very close to platter surface, reads and writes magnetically encoded info
- Surface of platter divided into circular **tracks**
  - Over 50K-100K tracks per platter on typical hard disks
- Each track is divided into **sectors.**
  - A sector is the smallest unit of data that can be read or written.
  - Sector size typically 512 bytes
  - Typical sectors per track: 500 to 1000 (on inner tracks) to 1000 to 2000 (on outer tracks)
- To read/write a sector
  - disk arm swings to position head on right track
  - platter spins continually; data is read/written as sector passes under head
- Head-disk assemblies
  - multiple disk platters on a single spindle (1 to 5 usually)
  - one head per platter, mounted on a common arm.
- **Cylinder** *i* consists of *i*th track of all the platters



Performance Measures of Disks

- **Access time** – the time it takes from when a read or write request is issued to when data transfer begins. Consists of:
  - **Seek time** – time it takes to reposition the arm over the correct track.
    - Average seek time is 1/2 the worst case seek time.
    - 4 to 10 milliseconds on typical disks
  - **Rotational latency** – time it takes for the sector to be accessed to appear under the head.
    - 4 to 11 milliseconds on typical disks (5400 to 15000 r.p.m.)
    - Average latency is 1/2 of the above latency.
  - Overall latency is 5 to 20 msec depending on disk model
- **Data-transfer rate** – the rate at which data can be retrieved from or stored to the disk.
  - 25 to 200 MB per second max rate, lower for inner tracks
- **Mean time to failure (MTTF)** – the average time the disk is expected to run continuously without any failure
  - Nowadays, typically 3 to 5 years

Performance Measures (Cont.)

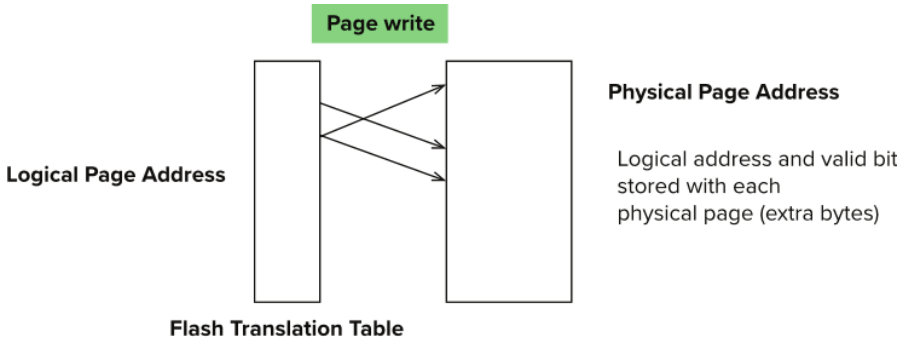
- **Disk block** is a logical unit for storage allocation and retrieval
  - 4 to 16 kilobytes typically
    - Smaller blocks: more transfers from disk
    - Larger blocks: more space wasted due to partially filled blocks
- **Sequential access pattern**
  - Successive requests are for successive disk blocks
  - Disk seek required only for first block
- **Random access pattern**
  - Successive requests are for blocks that can be anywhere on disk
  - Each access requires a seek
  - Transfer rates are low since a lot of time is wasted in seeks
- **I/O operations per second (IOPS)**
  - Number of random block reads that a disk can support per second
  - 50 to 200 IOPS on current generation magnetic disks

Flash Storage

- NOR flash and NAND flash
- NAND flash
  - used widely for storage, cheaper than NOR flash
  - requires page-at-a-time read (page: 512 bytes to 4 KB)
    - 20 to 100 microseconds for a page read
    - Not much difference between sequential and random read
  - Page can only be written once
    - Must be erased to allow rewrite
- **Solid state disks**
  - Use standard block-oriented disk interfaces, but store data on multiple flash storage devices internally
  - Transfer rate of up to 500 MB/sec using SATA

Flash Storage (Cont.)

- Erase happens in units of **erase block**
  - Takes 2 to 5 millisecs
  - Erase block typically 256 KB to 1 MB (128 to 256 pages)
- **Remapping** of logical page addresses to physical page addresses avoids waiting for erase
- **Flash translation table** tracks mapping
  - also stored in a label field of flash page
  - remapping carried out by **flash translation layer**



- After 100,000 to 1,000,000 erases, erase block becomes unreliable and cannot be used

SSD Performance Metrics

- Random reads/writes per second
  - Typical 4 KB reads: 10,000 reads per second (10,000 IOPS)
  - Typical 4KB writes: 40,000 IOPS
  - SSDs support parallel reads
    - Typical 4KB reads:
      - 100,000 IOPS with 32 requests in parallel (QD-32) on SATA
      - 350,000 IOPS with QD-32 on NVMe PCIe
    - Typical 4KB writes:
      - 100,000 IOPS with QD-32, even higher on some models
- Data transfer rate for sequential reads/writes
  - 400 MB/sec for SATA3, 2 to 3 GB/sec using NVMe PCIe
- **Hybrid disks**: combine small amount of flash cache with larger magnetic disk

Storage Class Memory

- 3D-XPoint memory technology pioneered by Intel
- Available as Intel Optane
  - SSD interface shipped from 2017
    - Allows lower latency than flash SSDs
  - Non-volatile memory interface announced in 2018
    - Supports direct access to words, at speeds comparable to main-memory speeds

RAID

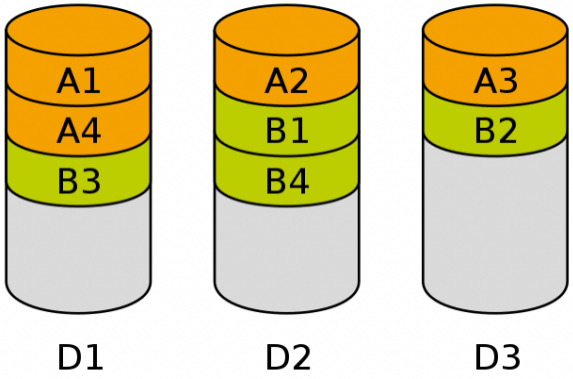
- **RAID: Redundant Arrays of Independent Disks**
  - disk organization techniques that manage a large numbers of disks, providing a view of a single disk of
    - **high capacity** and **high speed** by using multiple disks in parallel,
    - **high reliability** by storing data redundantly, so that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

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 is still available in the other
    - Data loss occurs 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 (e.g. fire or building collapse or electrical power surges, …)
- **Mean time to data loss** depends on mean time to failure, and **mean time to repair**
  - E.g., MTTF of 100,000 hours, mean time to repair of 10 hours gives mean time to data loss of $500*10^6$ hours (or 57,000 years) for a mirrored pair of disks (ignoring dependent failure modes)

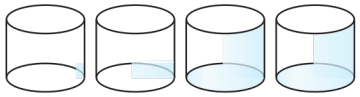
Improvement in Performance via Parallelism

- 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.
- **Block-level striping** – with *n* disks, block *i* of a file goes to disk (*i* mod *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

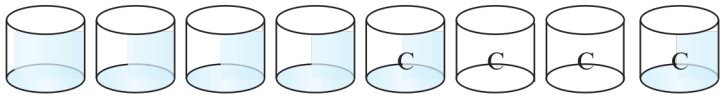


RAID Levels

- 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 loss 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.



(a) RAID 0: nonredundant striping



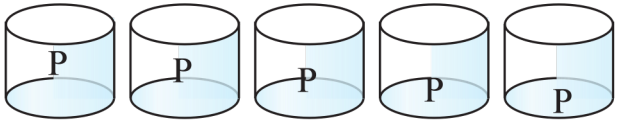
(b) RAID 1: mirrored disks

RAID Levels (Cont.)

- **Parity blocks**: Parity block *j* stores XOR of bits from block *j* of each disk
  - When writing data to a block *j*, parity block *j* must also be computed and written to disk
    - 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
  - To recover data for a block, compute XOR of bits from all other blocks in the set including the parity block

RAID Levels (Cont.)

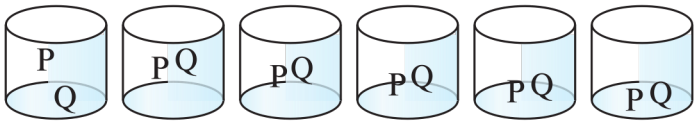
- **RAID Level 5:** Block-Interleaved Distributed Parity; partitions data and parity among all *N* + 1 disks, rather than storing data in *N* disks and parity in 1 disk.
  - E.g., with 5 disks, parity block for *n*th set of blocks is stored on disk (*n mod* 5) + 1, with the data blocks stored on the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



(c) RAID 5: block-interleaved distributed parity

RAID Levels (Cont.)

- **RAID Level 5** (Cont.)
  - Block writes occur in parallel if the blocks and their parity blocks are on different disks.
- **RAID Level 6**: P+Q Redundancy scheme; similar to Level 5, but stores two error correction blocks (P, Q) instead of single parity block to guard against multiple disk failures.
  - Better reliability than Level 5 at a higher cost
    - Becoming more important as storage sizes increase



(d) 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

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 had higher storage cost than level 5
- Level 5 is preferred for applications where writes are sequential and large (many blocks), and need large amounts of data storage
- RAID 1 is preferred for applications with many random/small updates
- Level 6 gives better data protection than RAID 5 since it can tolerate two disk (or disk block) failures
  - Increasing in importance since latent block failures on one disk, coupled with a failure of another disk can result in data loss with RAID 1 and RAID 5.

Optimization of Disk-Block Access

- **Buffering:** in-memory buffer to cache disk blocks
- **Read-ahead:** Read extra blocks from a track in anticipation that they will be requested soon
- **Disk-arm-scheduling** algorithms re-order block requests so that disk arm movement is minimized
  - **elevator algorithm:** move disk arm in one direction (from outer to inner tracks or vice versa), processing next request in that direction, until no more requests in that direction, then reverse direction and repeat