Database Systems II B: DBMS-Implementation

Chapter 4: Disks

Prof. Dr. Stefan Brass

Martin-Luther-Universität Halle-Wittenberg

Wintersemester 2019/20

http://www.informatik.uni-halle.de/~brass/dbi19/

Objectives

After completing this chapter, you should be able to:

- explain how disks work (list their main parts).
- evaluate disks, explain performance parameters.
- explain and evaluate different RAID configurations.
- create and use tablespaces in Oracle.
- explain the storage hierarchy and compare the characteristics of different storage media.
- explain how buffering (caching) works.
- find disk/buffer-related bottlenecks in Oracle.

Inhalt

Disks (1)

- A disk consists of a stack of circular plates ("platters", "data disks") each coated on one or both sides with magnetic recording material.

  As an example, we use the Hitachi Ultrastar 15K147. This is a server disk with SCSI interface (also available with FC and SAS interface). The capacity is 147 GB, the price (2005) about 700-1000$.
  This disk has 5 platters of 3.5 inch diameter and 10 heads.
  There are also versions with 73.4 GB (3 platters, 5 heads) and 36.7 GB (2 platters, 3 heads).

- The platters are mounted to a rotating spindle.

  As the name says, the disks rotate with 15000 rpm in the 15K147.
  Other speeds are e.g. 4400 (2.5" disks for notebooks), 5400 (low noise), 7200, and 10000 rpm.

Disks (2)

- There is one read-write head for each magnetic surface, flying on an air cushion e.g. 15 nm above the surface.

  The Ultrastar 15K147 has 10 heads (in the 147 GB version).
  Micron $= 10^{-6}$ inch. The information about the distance from the surface is not for the Ultrastar, Source:
  http://www.techportal.de/uploads/publications/497/phystech59.pdf

- The heads are mounted to an arm-assembly that looks like a comb and can move in and out.

  Only all heads together can be moved. Only one head can read or write at the same time.

Disks (3)

- The data is written on each surface in the form of concentric circles called tracks.

  The Ultrastar 15K147 has 40 935 tracks per surface.

  The track density is 90 000 tracks/inch (TPI) (average).

  The areal data density is 61.7 Gbit per square inch (9.7 Gbit/cm^2).

- The tracks with the same distance from the center on all surfaces together are called a cylinder.

  If they were connected, they would have this geometrical shape.

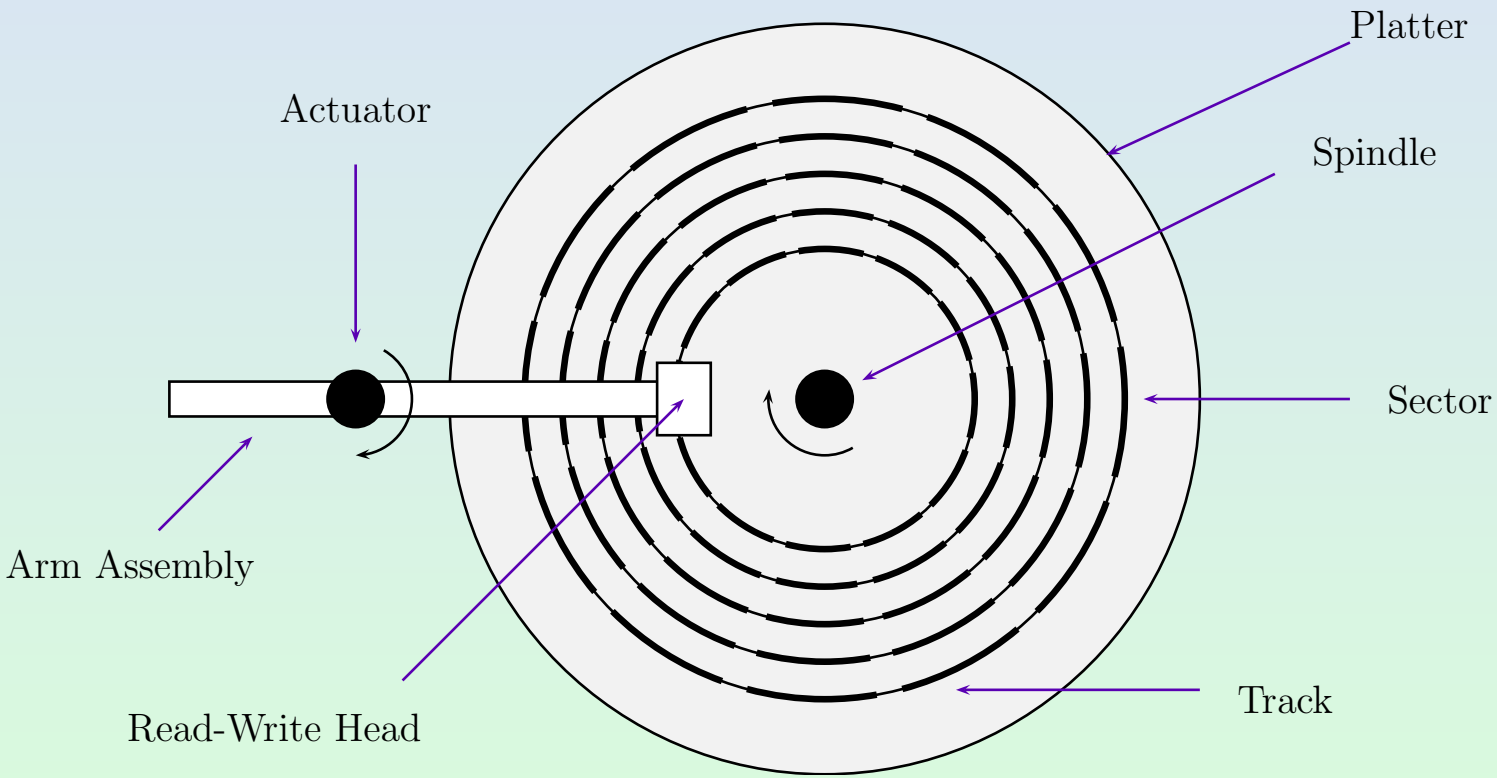
Disks (4)

- Each track is divided into sectors: This is the smallest unit of information that can be read/written.

  Sectors are small arcs of the circle. They often consist of 512 Bytes. The Ultrastar 15K147 permits 512–528 Byte/sector. Modern disks have more sectors on the outer tracks ("mutiple zone recording"), since the outer tracks are longer. The Ultrastar 15K147 has 24 zones, containing from 560 to 840 sectors per track.
  In total, it has 287 140 277 sectors.

Disks (5)

Top View:

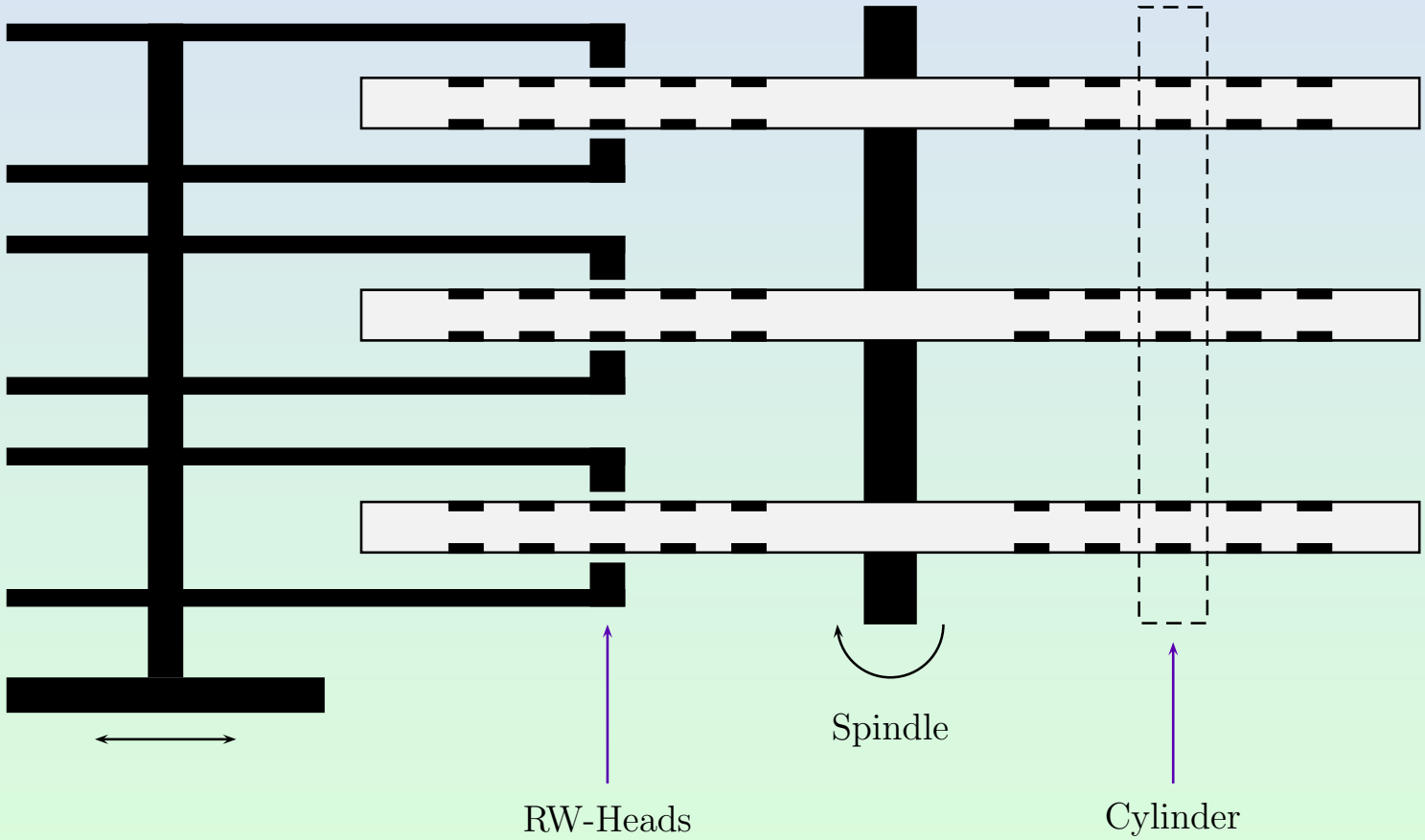


Ultrastar 15K147:

IBM Ultrastar 36ZX:

Disks (8)

Side View:



Disks (9)

- Modern disk drives have a disk controller built-in. This is simply a small computer.
- It translates relatively high-level commands (such as read the sector with address defined by cylinder, surface, and sector number) into the commands for the real hardware (e.g. the motor for the arm-assembly).

Disks (10)

- The disk controller attaches checksums to the sectors.

  It also attaches address information to ensure that the head is really on the right track (embedded servo technology).

- It is not economically feasible to produce 100% defect-free media. Disks have a limited number of replacement sectors.

  During initialization of the disk, each sector is tested and bad sectors are found. The controller manages a defect map which uses one of the spare sectors in place of a bad sector.

Disks (11)

- Modern disks have a cache (RAM) for recently read sectors.

  If the sector is still in the cache, it can be sent immediately to the computer without the mechanical delay needed to read a sector from the disk.

- Read ahead: The controller usually reads sectors following a requested sector into the cache.

  They are anyway available while the disk continues to spin and often required next. If the program does not fast enough request the next sector, it might have passed already under the read/write-head. Without caching, one would have to wait an entire turn of the disk.

Disks (12)

- Except for the read-ahead, the disk cache is not very important for database systems, since the DBMS has a cache of its own.

  The Ultrastar 15K147 has 16 MB buffer cache (of which 2.5 MB are used for drive firmware). This is "multisegmented" which means that the cached data can be from different locations on the disk. The Ultrastar 15K147 supports 1–256 buffer segments.

- Disks may be configured to cache write requests, too. That is dangerous.

  I.e. the request is remembered for delayed execution. But then it might be lost due to a power failure before it is really executed. When the disk tells the DBMS that the data were written, the DBMS must be able to rely on that.

Disks (13)

- Blocks are a collection of consecutive sectors.

  The sector size is often 512 Byte, but it is more economical for the operating system to read/write larger units. The block size is often 8 KB under UNIX, and e.g. 2 KB under DOS ("cluster size"). Block size is usually determined during formatting as a multiple of the fixed sector size.

- In case of a power failure, it can happen that blocks are written only partially.

  This leaves one with neither the old version nor the new version, which is a problem for recovery. One technique to discover such destroyed blocks is to write the same bit pattern at the beginning and end of each block, and invert it whenever a new version is written. One can also write version numbers or checksums.

Disk Capacity

- Disk manufactureres define 1 MByte as $1000 * 1000$ Byte, otherwise 1 MByte is $1024 * 1024$ Byte.
- The operating system needs part of the disk space for control information (not available for user data).
- Thus, the disk will appear smaller than advertised.
- The required disk space will grow over time.

  A rule of thumb is that the needed disk space doubles every year. However, when planning a database, you should do a much more careful calculation. Since disk space becomes cheaper over time, it is also not a good idea to buy disk space for many years in advance. One author recommends to buy disk space for the next two years.

Reading a Sector (1)

- The operating system sends the read request over the disk interface to the disk controller.
- The arm is moved to the right cylinder (seek time).

  The average seek time is the time needed to move the arm one third of the maximal distance. For the Ultrastar 15K147, it is 3.7 ms (read) / 4.1 ms (write). The disk needs 0.6 ms to position the head on the next track and 6.7 ms (read)/7.0 ms (write) for the full distance. The average seek time for write commands is slightly larger than for read commands, since for write commands the controller must be sure that the head is positioned on the right track ("it is locked into the track"). Sectors can be read earlier (when the position is not yet 100% sure) and then checked for the embedded address. The Ultrastar 15K147 needs 1.4 ms to switch from the last sector in the current track to the next sequential sector on another surface ("head skew").

Reading a Sector (2)

- The disk then waits until the needed sector shows up under the read-write head (latency time).

  In average, half a turn is needed. So 15 000 rpm give an average latency of $60\mathrm{s}/(15\,000 * 2) = 2\,\mathrm{ms}$.

- The drive then reads the data.

  Modern disks can read an entire track in one revolution (interleave factor 1:1), i.e. the speed is (140 205MB/409 360)∗250(turns/s) = 85.6 MB/s. The speed is higher on the outer tracks and lower on the inner tracks. Hitachi specifies: Outermost zone: 93.3 MB/sec, innermost zone: 62.3 MB/sec (measured typical values, 1 MB=10^6 Byte).

Disk Interfaces (1)

- After the disk controller has read the sector, it must transfer the data to the computer's main memory.
- The example disk has an Ultra320 SCSI (pronounced "scuzzy") interface which allows to transfer 320 MB/s.

  Previous versions: SCSI2 (Fast, Wide): 5/10/20 MB/s (1986), Ultra SCSI: 20/40 MB/s, Ultra-2 SCSI: 80 MB/s, Ulta160 SCSI.
- Multiple disks (and other devices) can be connected over the same SCSI bus.

  Otherwise the high bandwidth would not be interesting, because it cannot be used up by a single disk (except possibly from the cache).

Disk Interfaces (2)

- PC interfaces: IDE/ATA, Ultra ATA, SATA

  IDE/ATA: 2–4MB/s,
  Ultra ATA: 33MB/s, Ultra ATA/66: 66MB/s,
  Ultra ATA/100: 100MB/s, Ultra ATA/133: 133 MB/s
  S-ATA: 150 MB/s, 300 MB/s

- In future, there will probably be only one physical medium for networking and disk-computer connection, e.g. “Fibre Channel” (FC).

  There are versions with 1 Gbit/s, 2 Gbit/s, 4 Gbit/s, 5 Gbit/s. We just bought (in 2005) a disk subsystem with 2 Gbit/s connectors, but our main machine has two parallel connections. 1 Gbit/s corresponds to about 400 MByte/s of data.

Disk Performance (1)

- It is much faster to access consecutive blocks than blocks scattered randomly over the entire disk.

  Consecutive means first the following sectors on the same track, then another track (surface) in the same cylinder, and then an adjacent cylinder. Normally the sectors are ordered such that when we move to an neighbouring cylinder, no or only a minimal latency time is needed.

- E.g. reading 16 MB which are stored in one piece takes 0.25 seconds (255 ms).

  In the Ultrastar 15K147, a track contains on average 359136 Bytes, thus nearly 47 tracks must be read. This takes one random seek (3.7 ms), one average latency time (2 ms), 47 revolutions (47∗4 ms), 42 head switches (42∗1.4 ms), and 4 cylinder switches (4∗0.6 ms), giving in total 254.9 ms. Hitachi specifies 186 ms for reading 16 MB sequentially in the outermost zone and 275 ms in the innermost zone.

Disk Performance (2)

- (Older) Textbooks say that 10 MB/s can be effectively read from a disk (if the data is stored in consecutive blocks).
- The above computation gives 48 MB/s.
- Reading the same amount of data from randomly scattered blocks needs about 100 times as much time.

  Assume that we need to read 4096 blocks of 4 KByte.
  The time needed is $4096 * (3.7 + 2.0 + 0.05)$ ms $= 23.55$ s.
  The Hitachi documentation specifies 24.7 sec as typical time for 4096 random single block reads (they also calculate a command overhead) (a single block is 512 Byte).

Disk Performance (3)

- Some authors say that if a disk has constantly more than 50 independent accesses per second, it becomes a bottleneck.

  This leaves 20ms for every access. The disk is faster, but when the requests are randomly distributed, and you keep the disk operating near its limits, a queue will build up. Probably today this amount has increased to 70–100 accesses per second.

- Thus, even if the entire database would fit on a single disk, it might be necessary to buy several disks if the system load is high.

  An alternative might be to increase the buffer cache (RAM), see below.

Disk Performance (4)

- To improve the performance, data that are needed together should be stored near to each other:
  - Ideally in the same block.
  - If several blocks are needed (e.g. full table scan), the data should be stored in consecutive blocks.
- Multiple disks can at least do the seek in parallel.
  So if the data cannot be stored together, it might be an option to store it on different disks.

  Depending on the maximal transfer rate of the interface, also the transfer can be done interleaved. Larger computers have several disk interfaces which can work in parallel.

Disk Performance (5)

- Expected future performance improvements:
  - Disk Capacity: 27–60% per year.

    Doubles every 1.5–2 years.
  - Transfer Rate: 22–40% per year.

    Doubles every 2–3.5 years.
  - Rotation/Seek Time: 8% per year.

    Halves every 7–10 years.
  - \$/MB: $> 60\%$ per year.

    Halves in less than 1.5 years.

Disk Performance (6)

- This shows that the difference between random and sequential accesses becomes greater and greater.
- In 1970, this factor was about 30, in 2000 it is about 140.

  Data of the IBM 3330 (1970): Capacity: 93.7 MB, average seek: 30 ms, next track: 10 ms, max. seek: 55 ms, 3600 rpm, 13 KB/track, 19 heads, 411 cylinders, 806 KB/s transfer rate.

Solid State Disks (SSDs)

- Solid-State Disks (SSDs) use flash memory (like USB-sticks).

  There are no moving parts.

- While for magnetic disks, sequential reading or writing is much faster than accessing distant blocks, read accesses on SSDs are everywhere equally fast.

  For write accesses it is better when larger units are written, because these have to be deleted and written again (e.g. 2 MB).

- One block access takes e.g. 0.1 ms,
  a typical read/write-rate is 200–500 MB/s.

- The number of times a block can be written is limited.

  E.g. 3000 times. The controller tries to write all blocks equally often and might have more blocks than the specified capacity.

- The price per GB is about 10 times that of a magnetic disk.

Inhalt

RAID Storage (1)

- The speed and capacity of disks is limited.

  At least very large/fast disks become unproportionally more expensive.

- Especially the seek time is reduced very slowly compared to the processor speed.

  CPU speed: +60% per year, i.e. doubles every 1.5 years.

- Idea: Combine multiple disks to a larger storage component!

- RAID: Redundant Array of Independent (or Inexpensive) Disks.

RAID Storage (2)

- But: With 100 disks, one must expect one disk failure every 6 months.
- Therefore, RAID systems must include measures in order to prevent the loss of data when a disk fails.

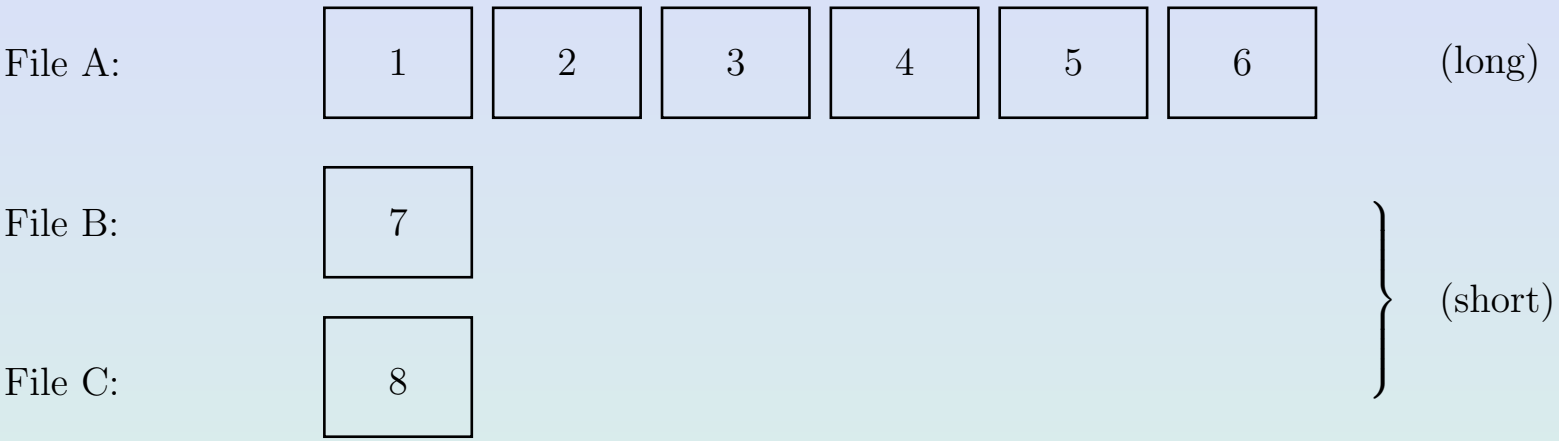
  In addition, good RAID systems allow to exchange faulty disks during the operation ("hot swap"), automatically manage a built-in spare disk ("hot spare"), and have also redundant controllers, power suplies, etc.
- There are different ways to couple the disks (RAID levels).

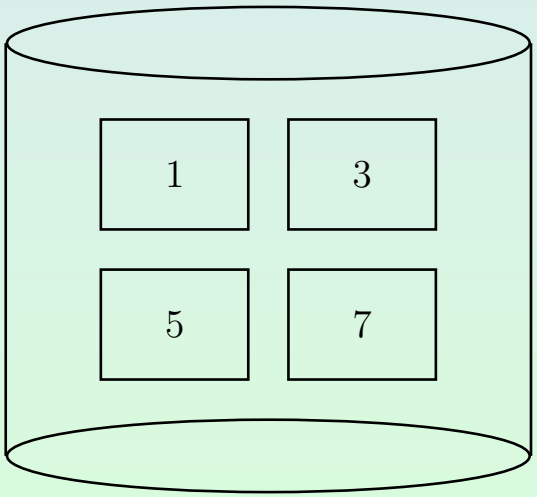
RAID 0: Striping (1)

- With two disks, the first block is written to disk 1, the second to disk 2, the third to disk 1, and so on.
- The speed of read/write accesses for long files scales up nearly linearly with the number of disk drives.

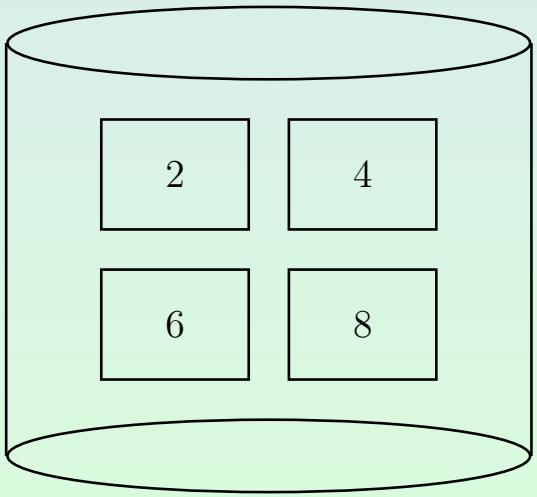
  With two drives, the time required for reading/writing a long file is nearly half of the time required with one disk.
- Random accesses to single blocks do not become faster, but they are distributed among multiple drives (more requests per second can be processed).
- A single disk failure makes all data unusable.

RAID 0: Striping (2)





Disk 1



Disk 2

RAID 1: Mirroring (1)

- Here disk 2 keeps a copy of disk 1.
- The time needed for write accesses is not improved.

  It required time might actually increase: Some controllers first write a block on disk 1, and only after that succeeded, also on disk 2 (to leave at least one in a consistent state).

- The time for read accesses is slightly improved.

  A read access can be scheduled for the disk where the head is nearer. For very large files, both disks could work in parallel.

- Read accesses can be distributed among the disks (reads/sec doubles), write commands must be done on both disks.

RAID 1: Mirroring (2)

- If one of the two disks fails, the controller can continue to work without interruption with the other disk.

  If the system has a "hot spare" disk, the controller can immediately start to copy the contents of the remaining disk on the spare disk, so that the data is again kept redundantly on two disks.

- Even with mirroring, backups must be done.

  There is no protection against a fire or errors of the DBA (`DROP TABLE ...`), software bugs, viruses etc.

RAID 10: Stripe+Mirror (1)

- RAID Level 1+0 together, i.e. striping and mirroring, is sometimes called RAID Level 10.

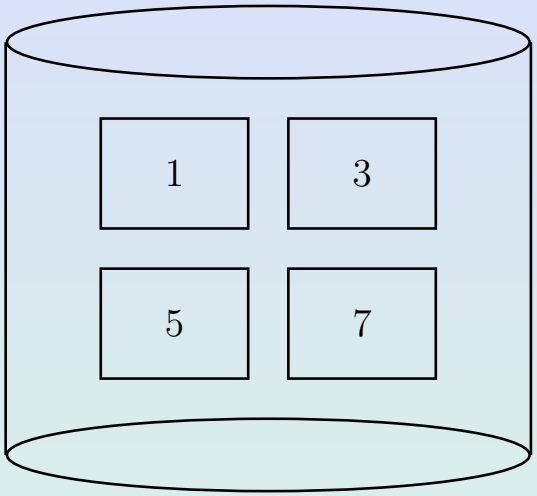
  RAID 10 is a RAID 0 built on top of two RAID 1 systems. It is also possible to build a RAID 1 on two RAID 0 systems (Level 0+1): The distribution of data blocks is the same, but if a disk fails, the blocks of the other disk in the RAID 0 system are considered unusable.

- The first block is stored on disk 1 and 3, the second on disk 2 and 4, the third on disk 1 and 3, etc.

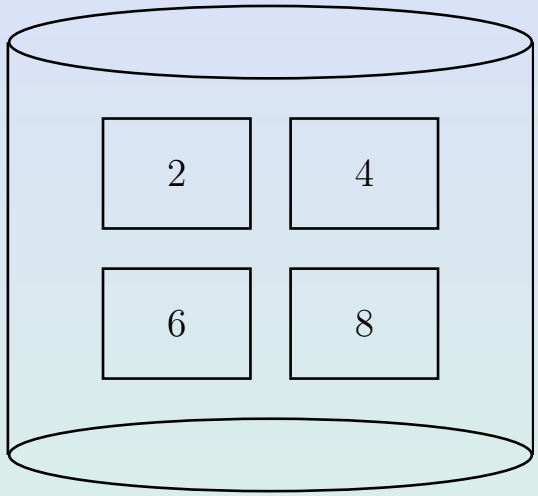
- This probably gives the best performance, but the mirroring is quite expensive.

RAID 10: Stripe+Mirror (2)

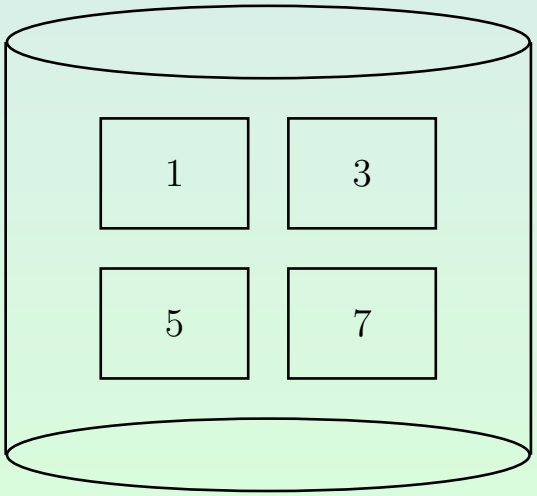
Disk 1:



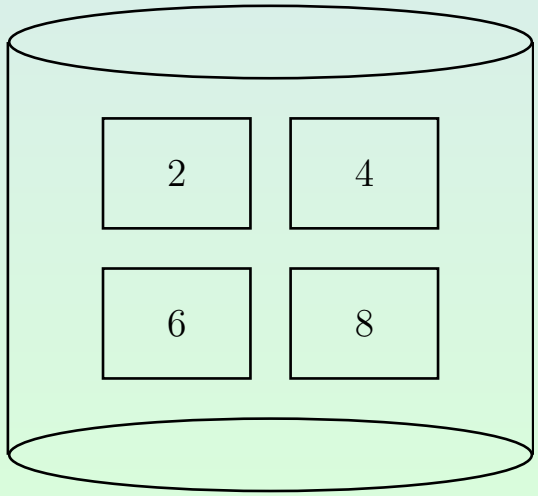
Disk 2:



Disk 3:



Disk 4:



RAID 4: Parity Disk (1)

- Mirroring doubles the required disk capacity (space overhead 100%).
- RAID levels 3–5 support striping of the data among n disks and add only a single disk with parity information.

  E.g. the XOR (exclusive or) of the data stored on the n disks is stored on the parity disk (i.e. the number of "1" in the corresponding bits on the $n+1$ disks is always even).
- So e.g. with four data disks, the space overhead is only 25%.

RAID 4: Parity Disk (2)

- If one disk fails, its information can be reconstructed from the other disks.

  E.g. if the number of "1" in the other n disks is even, the corresponding bit on the failed disk must have been "0". Two disk failures are always fatal. In RAID level $0+1$, one might be lucky that not both copies of the mirrored disk are hit. But for planning a safe system, it is probably more important what can be guaranteed.

- The performance for reading is similar to RAID 0:
  - The speed is improved only for long reads,
  - for small reads load-balancing is done (number of reads/second scales up lineraly with data disks).

RAID 4: Parity Disk (3)

- Writing becomes slower than with a single disk, since the old version of the block and the old version of the parity block must be read first to recompute the parity.

  NewParity = (OldData XOR NewData) XOR OldParity.
  So on each of the two disks, one needs to read a block, wait one rotation of the disk, and write it. On the ST318405 the read-modify-write cycle takes 56% more time than a simple write access. Caching helps (if the old version of the block is already in memory). Also long sequential writes do not have this problem, because they can compute the parity from each group of n written blocks.

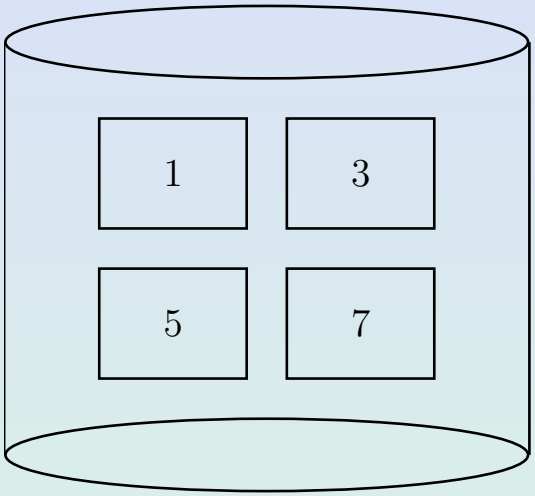
RAID 4: Parity Disk (4)

- If a disk fails, the performance goes down quite drastically (it halves although only one out of many disks failed).

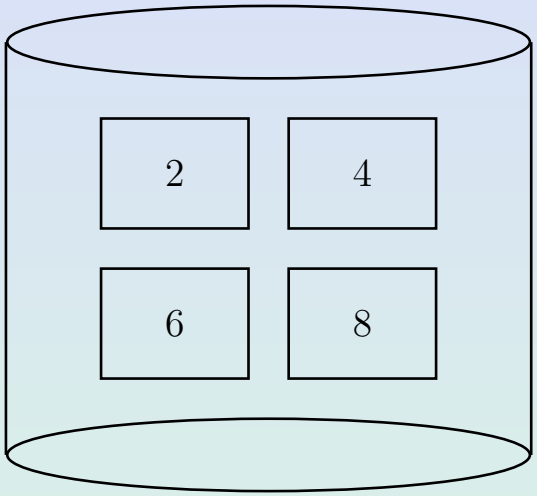
  E.g. the array consists of 10 disks plus one parity disk, each disk can process 50 reads/second. Thus, the performance of whole array is 500 reads/second (assuming optimal distribution). Suppose one of the 10 data disks fails. In one second, one can read 25 times from each of the 9 working disks plus 25 times from all disks to reconstruct the blocks on the failed disk. So the performance is only 250 reads/second. If the parity disk fails, performance is not reduced.

- Larger RAID systems partition the data disks into groups, and have one parity disk for every group.

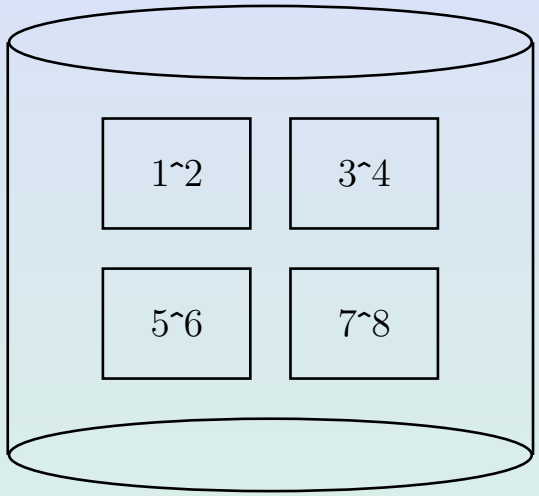
RAID 4: Parity Disk (5)



Disk 1



Disk 2



Disk 3

Disk 1	0	0	1	1
Disk 2	0	1	0	1
Disk 3	0	1	1	0

RAID 5: Distributed Parity (1)

- In RAID Level 4, the parity disk becomes a bottleneck for write operations.

  Each write operation, no matter on which disk, requires in addition a read and a write on the parity disk.

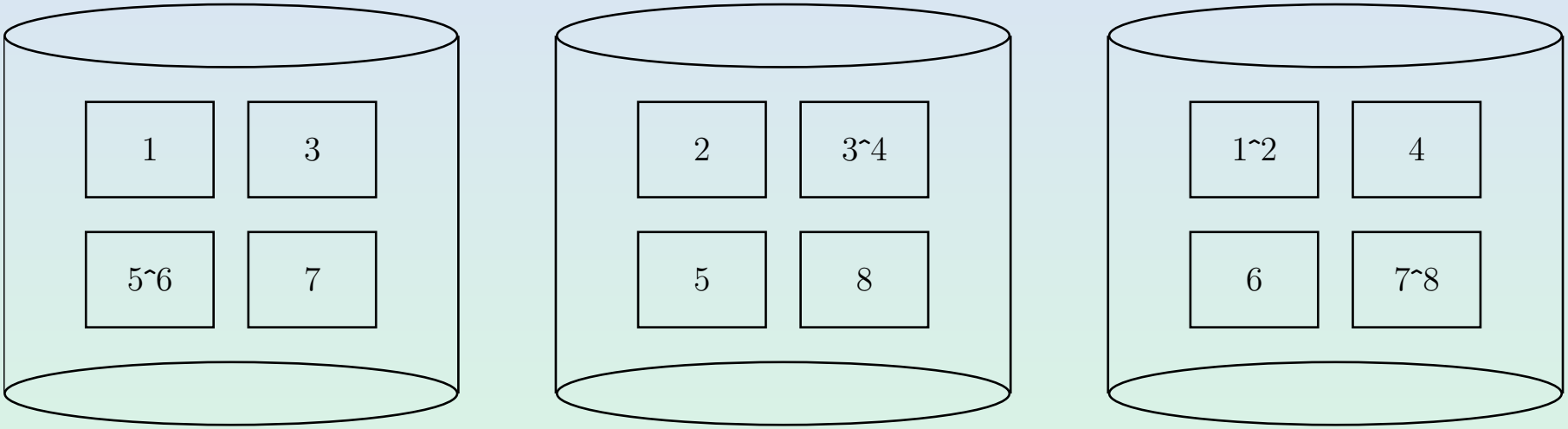
- RAID 5 distributes the parity information evenly among the disks. Otherwise, it is like RAID 4.

  Actually, reads can be distributed now over all $n + 1$ disks, instead of only the n data disks.

- Some write operations can now be done in parallel.

  In RAID Level 4, writes are serialized by the necessity to access the parity disk.

RAID 5: Distributed Parity (2)



Disk 1 Disk 2 Disk 3

Other RAID Levels

- RAID 2 uses error detecting and correcting codes.

  This is not used in practice, since each disk normally knows that it is faulty.

- RAID 3 is similar to RAID 4, but does the striping bit-wise instead of block-wise.

  So instead of putting e.g. the first 4 KB on disk 1, and the second 4 KB on disk 2, the bits of a block of 8 KB are distributed between both disks. But then it is always necessary to access all disks (no increase in accesses/second).

- RAID 6 uses two disks with error correcting codes: It can recover from two simultaneous disk failures.

RAID Storage: Evaluation (1)

- RAID systems are easy to administer:
  They look like one big disk.
- But better performance can be reached if database objects are explicitly distributed over multiple disks.

  This is an important part of physical database design.
- RAID Level 10 might be ok for databases.

  For redo log files, the stripe size should be small. But redo log files are much better kept on disks of their own, and not mixed with the data files. They must be mirrored or otherwise protected, but the DBMS software can do that.
- It might take quite long to restore the data on a spare disk in a RAID 5 system (hours). During this time, there is no protection against another disk failure.

More Remarks about RAID Systems

- All disks must be of the same size.

  It might be possible to plug in larger disks, but use only the minimum of the sizes of all disks in the RAID system. If one uses very large disks (the newest technology), it might be difficult to get replacement disks.

- One should ask what possibilities there are to access the data when the RAID controller is faulty.

  Sometimes the data can be read only with this controller.

RAID Storage: Evaluation (3)

- If one uses RAID Level 3–5, one should benchmark the system also with a simulated disk failure.

  Will the performance still be sufficient if one disk should fail?

- RAID Level 3–5 is not good for redo log files, since these are only written.

Inhalt

Tablespaces (1)

- In Oracle, one can use tablespaces to control on which disk(s) a table is stored.

  Tablespaces are physical containers for tables. When a table is created, the tablespace in which the table is stored can be defined.

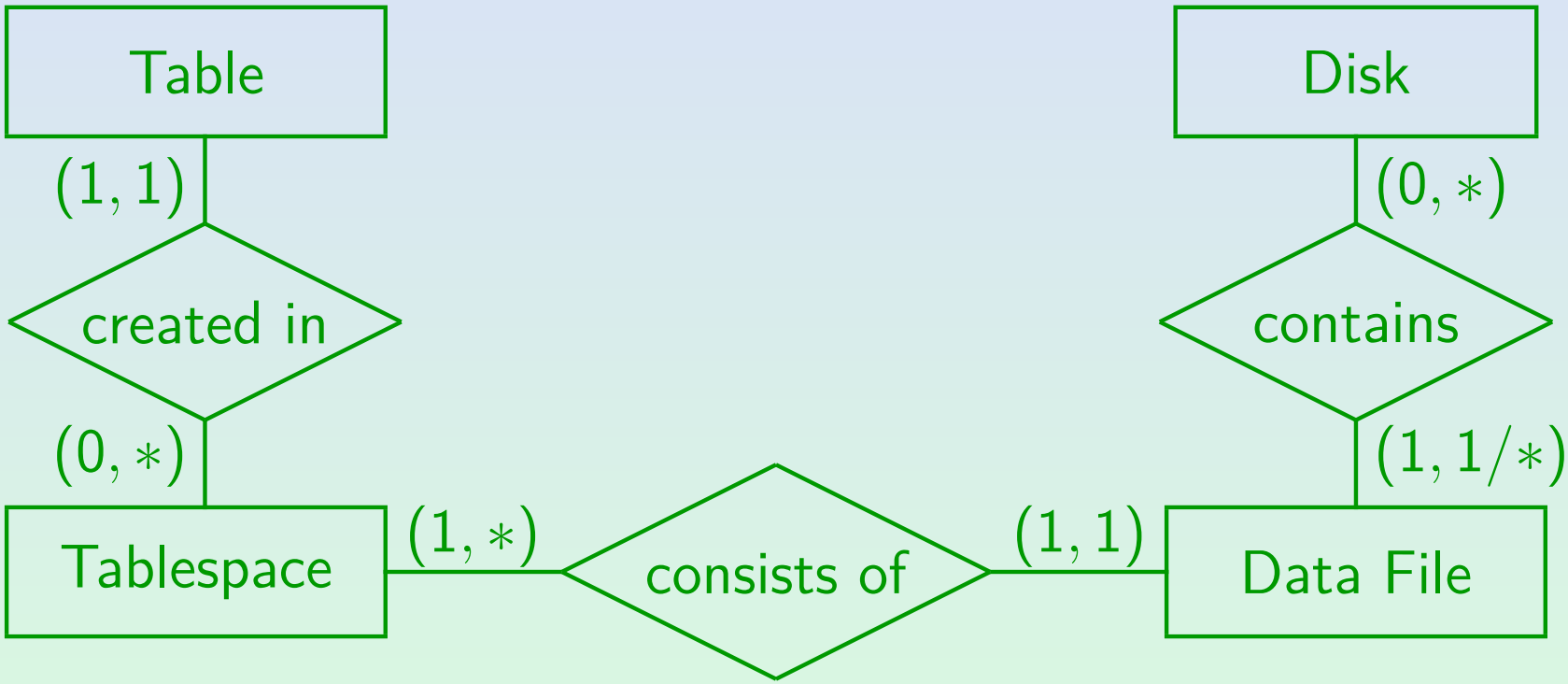
- Tablespaces are groups of data files. The files can be on the same disk or spread across serveral disks.

  A tablespace is something like a logical disk.

- Every data file can belong to only one tablespace.

- It is possible to have data files from different tablespaces on the same physical disk.

Tablespaces (2)



A data file is spread across several disks only in case of RAID systems.

Tablespaces (3)

- Example for defining the tablespace for a table:

```
CREATE TABLE STUDENTS(STUD_ID NUMERIC(5), ...)
                    TABLESPACE USER_DATA;
```

- Clauses setting physical attributes are specific to a DBMS (in this case Oracle), they are not contained in the SQL standard.
- The data file cannot be specified.

  Actually, the data file cannot be specified only for the first extent (piece of storage) allocated for the table. One can manually allocate additional extents in specified data files in order to stripe a table between different disks.

Tablespaces (4)

- If the tablespace consists of more than one file, Oracle may store part of the table in one data file, and part in the other.

  For this reason, most DBAs prefer to have only one data file per tablespace, if possible.

- Every Oracle database has a tablespace "SYSTEM", which contains e.g. the data dictionary.

  Simple DBs have only this tablespace. However, it is recommended to store user data in a different tablespace.

Tablespaces (5)

- It is normally easiest to have only one data file per tablespace.

  If a single disk is not large enough, it might be better to explicitly distribute the data among different tablespaces.

- It increases the flexibility if not too many tables (at least no unrelated ones) are put into the same tablespace.

  Tablespaces can be separately taken online or offline, separately exported and recovered. Backup copies are taken of datafiles. Performance statistics are available for datafiles.

Managing Tablespaces (1)

- A tablespace is created with a command like the following:

```
CREATE TABLESPACE USER_DATA
DATAFILE 'D:\User1.ora' SIZE 20M;
```

- Oracle will automatically create the datafile.

  Use "`SIZE 20M REUSE`" if the file exists and can be overwritten (the size is optional in this case).

- A data file can be added to a tablespaces with the following command:

```
ALTER TABLESPACE USER_DATA
ADD DATAFILE 'D:\User2.ora' SIZE 20M;
```

Managing Tablespaces (2)

- It is possible to let Oracle extend the datafile whenever the tablespace becomes full:

```
CREATE TABLESPACE USER_DATA
DATAFILE 'D:\User1.ora' SIZE 20M
         AUTOEXTEND ON NEXT 5M MAXSIZE 50M;
```

  The file is created with 20 MB size. When it is full, it is increased to 25 MB, 30 MB, and so on until 50 MB. When the 50 MB are used up, further commands that need additional storage (e.g. insertions) will fail. Without `MAXSIZE`, the entire disk is filled.

- The data file size can also be manually increased:

```
ALTER DATABASE
DATAFILE 'D:\User2.ora' RESIZE 100M
```

Managing Tablespaces (3)

- Tablespaces can be taken offline (i.e. made not available):

```
ALTER TABLESPACE USER_DATA OFFLINE;
```

The `SYSTEM` tablespace cannot be taken offline.

- The following command deletes a tablespace with all data in it:

```
DROP TABLESPACE USER_DATA
INCLUDING CONTENTS;
```

Managing Tablespaces (4)

- The following command is needed when data files are renamed or moved to another disk:

```
ALTER DATABASE
RENAME FILE 'C:\User2.ora' TO 'D:\User2.ora'
```

  The file cannot be currently in use. E.g. the tablespace is offline or the DBMS server is in the `MOUNT` state, but not `OPEN`. The command only changes the file name that Oracle uses to access the file (stored in Oracle's control file), one must use OS commands to actually move the data file.

- See the Oracle SQL Reference for more options.

Tablespaces in the Data Dictionary (1)

- DBA_TABLESPACES is list of all tablespaces. Columns are, e.g.
  - TABLESPACE_NAME: Name of the tablespace.
  - INITIAL_EXTENT, NEXT_EXTENT, MIN_EXTENTS, MAX_EXTENTS, PCT_INCREASE:
    Default storage parameters for tables created in this tablespace (see next chapter).
  - MIN_EXTLEN: Minimal size for storage pieces allocated in this tablespace.

Tablespaces in the Data Dictionary (2)

- Selected columns of DBA_TABLESPACES, continued:
  - STATUS: ONLINE, OFFLINE, READ ONLY.
  - CONTENTS: PERMANENT or TEMPORARY.

    TEMPORARY: For sorting during query evaluation.
  - LOGGING: LOGGING or NOLOGGING.

    If changes are not logged, they cannot be recovered.
  - EXTENT_MANAGEMENT: DICTIONARY or LOCAL.

    Extends are pieces of storage allocated for tables or other database objects. Originally, free space was managed by entries in the data dictionary. Oracle 8i has introduced local extend management (probably a bitmap inside the datafile) that is supposed to be more efficient.

Tablespaces in the Data Dictionary (3)

- USER_TABLESPACES lists all tablespaces for which the current user has write permission.
- USER_TS_QUOTAS lists the current file space usage and the allowed maximum for every tablespace writeable by the user.
- USER_FREE_SPACE: Pieces of free space in tablespaces.
- TABS (= USER_TABLES) contains the tablespace in which a table is stored:

```
SELECT TABLE_NAME, TABLESPACE_NAME
FROM   TABS
```

Tablespaces in the Data Dictionary (4)

- See also:
  - DBA_DATA_FILES,
  - V$DATAFILE,
  - V$DATAFILE_HEADER,
  - DBA_FREE_SPACE,
  - DBA_FREE_SPACE_COALESCED,
  - DBA_TS_QUOTAS,
  - V$FILESTAT,
  - V$TABLESPACE.

Performance Aspects (1)

- One should try to distribute the load (i.e. accesses per second) evenly between the disks.

  It is ok if a disk containing an often accessed file remains half empty.

- Distribute the tables among relatively many tablespaces to increase the flexibility for later changes.

  E.g. one can move a tablespace with all its tables to a different disk. If one has only a single tablespace with multiple files, it is not predictable, which table is contained in which file.

- One can manually stripe a table over multiple disks.

  See `ALTER TABLE ... ALLOCATE EXTENT ....`
  Necessary if the table requires more accesses/sec. than a single disk can deliver and one does not use a RAID system.

Performance Aspects (2)

- Before a `COMMIT` operation can be completed, a disk block must be written to the redo log files.

  This is actually the only case in which a disk block must be written synchronously no matter how much memory one has.

- If some disks are faster than others, think carefully what to put on them.

  If there are many transcations per minute, it would make sense to place the redo log on the fastest disks. But it is even more important not to place anything else on the disks with the redo log: The redo log is sequentially written, so the read/write head can remain at the current track and does not have to move around.

Performance Aspects (3)

- It is not recommended to store indexes on the same disk as their tables.

  This is advantageous only if there is a series of several index lookups, either because of many queries of this type or because of a join evaluated with the index. In that case the disk head does not have to move back and forth between index and table. This also gives load distribution: Both disks can work in parallel. Also, whenever the table is modified, the index must also be updated, and this can be done parallel on the two disks.

- In the same way, tables that are often joined together should be on different disks.

  Unless they can be put into a cluster, i.e. stored jointly in the same disk blocks, see below.

Performance Aspects (4)

- The data files should be stored in consecutive disk blocks.

  This automatically happens if the filesystem on the disk was created directly before the DB files were created. Otherwise there might be OS defragmentation tools.

- The number of DB blocks read in one I/O operation for full table scans is determined by the initialization parameter `DB_FILE_MULTIBLOCK_READ_COUNT`.

  It should be set relatively high, e.g. 8, 16, 32.

References

- Elmasri/Navathe: Fundamentals of Database Systems, 3rd Edition. Section 5.1–5.4.
- Ramakrishnan/Gehrke: Database Management Systems, 2nd Ed. Section 7.1, 7.2, 7.4.
- Härder/Rahm: Datenbanksysteme, Konzepte und Techniken der Implementierung.
- Mark Gurry, Peter Corrigan: Oracle Performance Tuning, 2nd Edition (with disk).
- Seagate: http://www.seagate.com/products/discsales/
- Quantum/Maxtor: http://www.maxtor.com/Maxtorhome.htm
- Hitachi Global Storage Technologies: http://www.hitachigst.com/
- Adaptec (RAID- and HDD-controller): http://adaptec.com, http://www.adaptec.de/
- Wikipedia (RAID systems): http://en.wikipedia.org/wiki/Redundant_Array_of_Independent_Disks
- The PC Guide: Hard Disk Performance http://www.pcguide.com/ref/hdd/perf/index.htm
- Storage Review: http://www.storagereview.com, http://198.76.30.88/jive/sr/
- Patterson/Keeton: Hardware Technology Trends and Database Opportunities. SIGMOD'98. http://www.cs.berkeley.edu/~pattrsn/talks/sigmod98-keynote-color.pdf