认识单位与数量级

1MB=1024KB
1s=1000ms

Metric	Value		Bytes
Byte (B)	1	2^0	1
Kilobyte (KB)	$1,024^1$	2^{10}	1,024
Megabyte (MB)	$1,024^2$	2^{20}	1,048,576
Gigabyte (GB)	$1,024^3$	2^{30}	1,073,741,824
Terabyte (TB)	$1,024^4$	2^{40}	1,099,511,627,776
Petabyte (PB)	$1,024^5$	2^{50}	1,125,899,906,842,624
Exabyte (EB)	$1,024^6$	2^{60}	1,152,921,504,606,846,976
Zettabyte (ZB)	$1,024^7$	2^{70}	1,180,591,620,717,411,303,424
Yottabyte (YB)	$1,024^8$	2^{80}	1,208,925,819,614,629,174,706,176

版权所有盗版必究
课程报名联系我们

Disk Access Time

一段时间内可以转移的数据

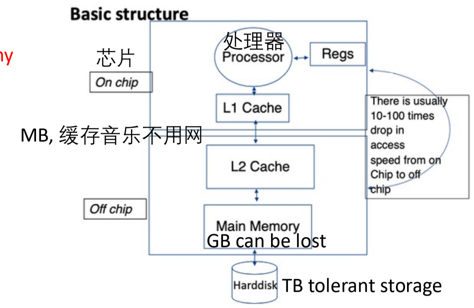
圆盘在转 要转移的数据

Disk access time=seek time + rotational time + (transfer length/ bandwidth)

Head从A-B的时间

对SSD (solid state disk/driver): Disk access time=transfer length/ bandwidth(无旋转 无seek 只剩一部分)
1.Silicon 2. No seek/rotation time 3.No start-up time like hardware disk (不需启动, 从转到不转) 4. run silently 5. very expensive

Memory Hierarchy



版权所有盗版必究
课程报名联系我们

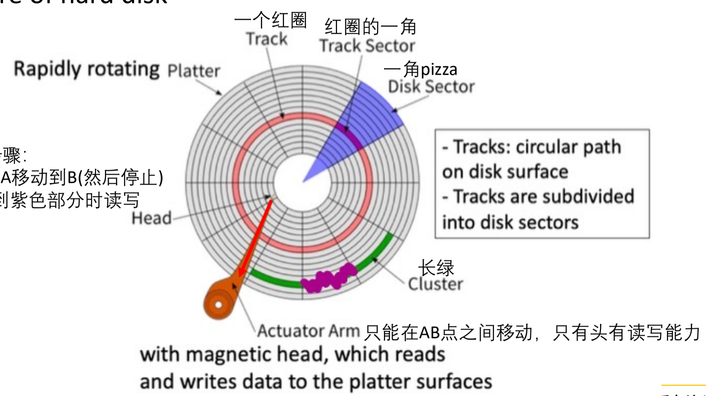
Performance of a DB system comes from

- **Hardware**
  - – The speed of the processor
  - – Number of processors
  - – Number of disk drives and I/O bandwidth
  - – Size of main memory
  - – Communication network
  - – Type of architecture
- **Software**
  - – Type of database technology used for a given application
- **Database tuning, crash recovery**
  - – Indexing parameters
  - – Data duplication
  - – Sharing data, etc

版权所有盗版必究
课程报名联系我们

Structure of hard disk

读写紫色部分的步骤:
1.Actuator Head 从A移动到B(然后停止)
2.Platter rotate 转到紫色部分时读写



版权所有盗版必究
课程报名联系我们

Q: Option 1: 250 GB solid state drive, 8 MB cache; Option 2: 500 GB hard disk drive, 32 MB cache; Usage: play online games, listen to music, watch movies, work on word documents for his studies, and browse social networks.

Option1
Pros: 1. Faster on playing large scale online games, as faster disk access time. 2. Better using experience, run silently less noise. Cons: Limited storage space, may not download too much movies or games compared with 2.
Option2
Pros: 1. Larger space and cache, store more movies, games and music and play them, so that we can still have fun even in the case when network is down. Cons: 1. Slower running speed (hard disk), may lead to a bit worse experience on playing games, and disk may become noisy especially when running large scale online games.

版权所有盗版必究
课程报名联系我们

- Cloud: flexible- you pay as you go -Cost Efficiency: Users pay only for the resources they actually use, reducing upfront costs and eliminating expenses for unused capacity. This is ideal for fluctuating or unpredictable demands. Scalability and Flexibility: Resources can be scaled up or down easily based on current needs, allowing businesses to adapt quickly without long-term commitments.

Q In one paragraph, compare Relational DB systems with Object Oriented DB Systems.
RDBMs stress relationship among tables, the data stored by RDBMs have relationship with each other, using structured schema where relationships between tables are established through primary and foreign keys. Data are manipulated by structure query language. OODBMs Store data as object, similar to the objects used in object-oriented programming, allowing for more complex data models that align with programming languages. This approach supports inheritance, encapsulation, and polymorphism directly within the database, which is advantageous for applications with complex data relationships and when a tight integration between the database and application logic is needed.

Jane has developed an online game which does not have many users at this moment. She is expecting a rapid growth in the number of users in near future, but cannot confirm the exact number or the time of growth. Currently the game is hosted in a server placed in Jane's garage. Jane does not have the space to accommodate more servers in her garage, but needs more computing and storage capacity for the game if a growth happens. Which one of the following solutions is the most suitable (cost, profit, and space effective) choice for this scenario?

- Establish a cluster for higher computing and storage capacity
- Buy more machines, place them somewhere else and establish a distributed system
- If the system is about to overload, refuse service to some users
- $\boxed{\text{Buy cloud computing and storage services}}$

版权所有盗版必究
课程报名联系我们

Effective memory access time

EA = H*C+(1-H)*M;   H = hit ratio=references satisfied by cache/ total references
C = cache access time;   M = memory access time

Q: Why not have large cache? 1.expensive 2. limited spaces on chip, even though more cache equipped, the access time still slow as extra caches and processor are not on the same chip.

Disk buffer/Disk Cache (是disk的一小部分(embedded), 与cache不同)
Effective disk buffer access time(在disk buffer上找不到了再去disk上找): EA = HB*BC+(1-HB)*D
HB = hit ratio of the disk buffer ; BC = buffer access time; D = disk access time
Q:
1. Assume disk access time is S, buffer access time is C, hit ratio is H, and S = 1000C. What is the effective access time, EA, as multiple of C when H = 30%? Ans: EA=0.3C+1000C(1-0.3)=700.3C
2. What is the Disk access time for a transfer size of 4KB, when average seek time is 12 ms, rotation delay 4 ms, transfer rate 4MB/sec? Ans: 1MB=1024KB; 1s=1000ms 4MB=4096KB 12ms+4ms+ [(4KB)/(4096KB/sec)]*1000ms

Q In one paragraph, discuss the importance of disks in DBMS design in its early years
1.Permanent storage on data and logs, even though the system crash, the data on main memory will likely to be lost but the storage in disks will not lost.
2.RAID redundant array of independent disks: fault tolerance of whole system can be improved by disk configuration.

版权所有盗版必究
课程报名联系我们

Type of DB systems (How the data are stored)

- Simple file:

Pros: Easy to implement; Low cost; light weight;
Cons: Concurrency problem, multiple users may change it simultaneously, causing data lost or inconsistency. Limited Querying Capabilities: Simple files lack efficient query functions making complex data retrieval difficult, especially with large datasets.

- Relational DB system: attribute, column row, primary key (strong relationship among tables)
- Object Oriented DB system
- NoSQL: {key-value pair "name": alice; "age":18}
- Deductive DB system

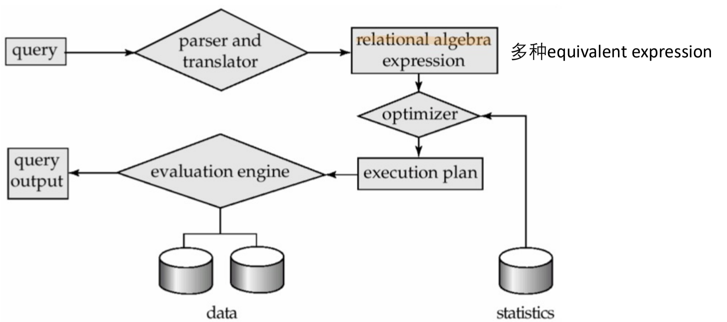
DB Architectures (How different machines are arranged together)

- Centralized (Client - Server): client 不同地方，serve与DB一个地方
- Distributed:一个admin多个participates, consistency for different partition
- WWW: security pravacy
- Grid:与distributed不同, admins perform management tasks on each individual node (local node) rather than on a global or centralized console.
- P2P: join and leave conveniently

版权所有盗版必究
课程报名联系我们

Query Processing Steps

E.g., select Salary From Employees Where Salary < 60000



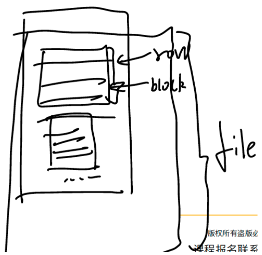
This process allows the SQL optimizer to identify and select expressions with lower execution costs, thereby improving performance by reducing computation time and resource consumption. Logically equivalent expressions also enable the optimizer to utilize indexes more effectively, speeding up data retrieval by restructuring queries to align better with existing data structures. Ultimately, this approach helps simplify the query plan, leading to faster and more efficient execution. So that the optimizer can make use of this expression to compare the cost of query plan

版权所有盗版必究
课程报名联系我们

Weather service Australia has installed multiple sensors in Melbourne with a small computing device in them. Each computing devices store and manage the data of its own sensor. When the sensors record any new data, they connect with the other nearby sensors in an ad-hoc network, share those data, and then disconnect. Which of the following database architecture is the most suitable choice for this scenario? P2P Join and leave

- 8 continuous bits (b) = 1 byte (B)
- 4000/8000 continuous bytes = 1 block
- Bit – b; byte – B; block – A

How data is stored in disk?



e.g., size of each record: 55 byte; fixed size of 1 data block: 4096 byte
4096/55=74.47≈74 records cannot be split into blocks

condition

$\Pi_{*}(\sigma_{Employees.ID=Managers.ID}$ (Employees X Managers))

select　Select all　Join table

版权所有盗版必
课程报名联系

When a database needs to join two tables using page-oriented nested loop join algorithm, which one of the following strategies will provide better efficiency?

$\boxed{\text{Take the table with the smaller number of pages as the outer relation}}$

Take the table with the smaller number of pages as the inner relation

b_r *dominates the whole equation, smaller* b_r *brings lower cost block transfers*: $b_r * b_s + b_r$ *and seeks*: $b_r + b_r$;

Q: We have seen two join algorithms in class. Describe in your own words how each of these algorithms work given two tables to join, with one paragraph to describe each algorithm. Then explain which one of these algorithms commonly works more efficiently than the other and why this is the case, explain with one paragraph.

Nested loop join: iterates over each record in the outer table, for each of these records, it goes through every record in the inner table to check for matching conditions. (The method is simple but slow, the time complexity is O(m*n), where m,n are # of records for two tables.); Page nested loop join: optimizes the standard nested loop join by networking data in blocks rather than individual records. Instead of loading one record from each table at a time, it loads a full block of records from the outer table and compare it with each block from the inner table, which reduces the # of I/O operations. Since multiple records are loaded into memory at once. It is a more efficient, by processing multiple records at once, the page oriented method minimizes the time spent accessing the inner table and reduces the I/O operations and leverage memory more efficiently.

Natural join: a join operation that can be performed of a column that is common in two tables.

$r \bowtie_{\theta} s$ Natural join

r: outer relation
s: inner relation
r and s are two tables
Theta is the common column.

Nested-loop Join

- Requires no indices because it checks everything in r against everything in s
- Expensive since it examines every pair of tuples in the two relations.
- Could be cheap if you do it on two small tables where they fit to main memory (disk brings the whole tables with first block access).

Estimated cost: *block transfers*: $n_r * b_s + b_r$ *and seeks*: $n_r + b_r$; n: record b: block

Block Nested-loop Join (Page-Oriented Nested-loop join) ($n_r \rightarrow b_r$)

block transfers: $b_r * b_s + b_r$ *and seeks*: $b_r + b_r$;

Q: Relation A has 1,000 records stored in 40 blocks. Relation B has 800 records stored in 50 blocks. For a join operation that joins these two relations, the query optimizer chooses to use block nested-loop join. Should the outer relation be A or B based on the costs? (smaller block transfer and seeks=lower cost better)

A is outer: block transfer 40*50+40=2040; seeks: 2*40=80

B is outer: block transfer 50*40+50=2050; seeks: 2*50=100

Q: # records of *customer*: 1000 # blocks of *customer*: 100 # records of *depositor*: 500 # blocks of *depositor*: 50

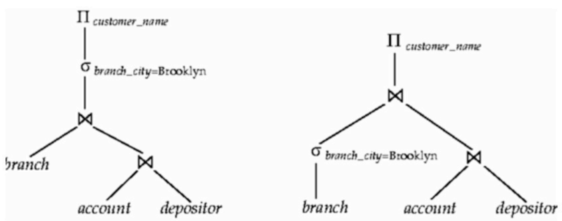
block nested loop join, choose which as outer relation?

Outer is customer: block transfer: 100(50+1)=5100 seeks: 2*100;

Outer is depositor: block transfer 50(100+1)=5050 seeks=2*50=100

Question 10: [4 Marks]

In the following figure we see two equal expressions a query optimizer is looking at to decide which one to run eventually:



Please describe which one of these plans the optimizer should choose and why. Is this choice true in general? If so, where in query optimization such an observation can be used? Briefly explain.

Right plan will be chosen. Because the right plan executes the select condition in branch table before it join with other tables, which helps to reduce the number of records during the second join operation. Therefore, the speed of the second join operation for right table is faster than that for the left plan, the whole speed of the query is faster. When we compare the speed of query plans, need to consider whether applying selection operation at earlier stage of query plan can help to reduce the # of intermediate tables.