COMP90050 ADBS

授课老师： Rita

0 – Preparation

课程介绍

2024 S1 – Prof Egemen Tanin

2024 S2 – Prof Farhana Choudhury

winter

课程特点

- 知识点杂
- 知识点偏理论（理解）
- 计算题少
- 做题不要跳步

课程资料

- Lecture (Pre-recorded + Live session)
- Tutorial (Start from week 2 – on campus)
- Assessment
  - Quiz: 2% * 5 = 10% (Week 3, 5, 7, 9, 11) – 20 min each
  - Project: presentation 15% + report 25% = 40% (4 members)
  - Final Exam: 50%
- 墨学资料
  - 课前ppt+课后ppt
  - 往年quiz
  - 往年真题

答疑时间 & 方式

- **上课时间**：墨尔本时间 每周六晚7-9pm
- 答疑时间：**每周一晚**，群里统一答疑
- 需要详细讲解的问题，每周六晚上课时统一讲解

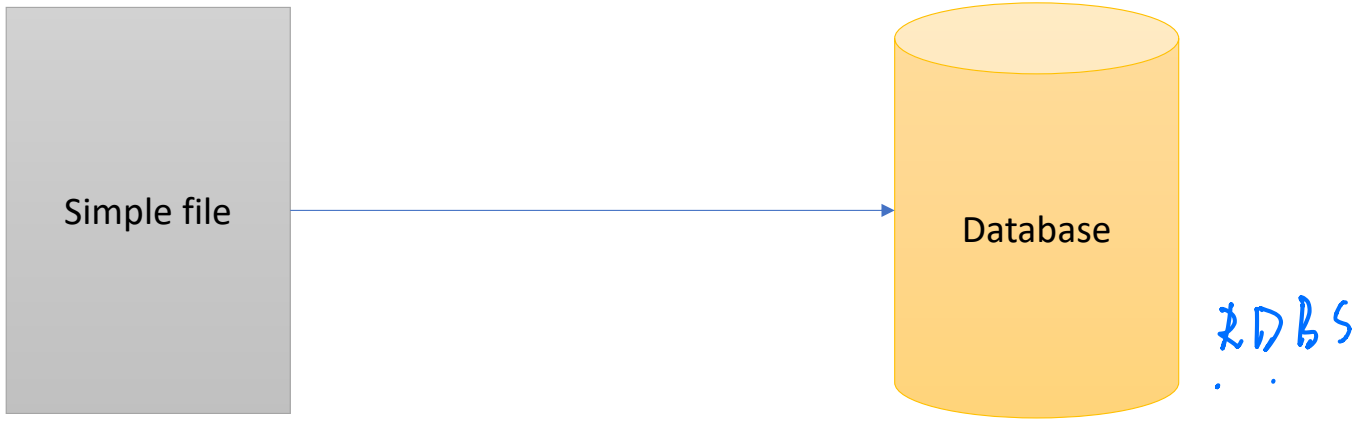
课程反馈

- 联系Lemon或其他助教

学习方法建议

- 按时上课 – Learn in English
- 不要堆积知识
- 做一个自己的笔记
- 及时做好总结

Why ADBS?



- Size of data increases
- Relationship of data becomes complex
- Storage can be in the cloud
- Interaction with end users are complex

What we will learn?

- Knowledge of how DB systems work at the architectural level
- Essentials to achieve correct behavior and the best possible performance
- Mechanisms used by current systems to provide many useful features
- Techniques for achieving reliability and good concurrency among other things

Topics

- 不同种数据库存储结构
- 几种RAID（fault tolerance）
- Recovery Algorithm
- ACID性质
- Transaction Model的区分
- CAP Theorem
- Deadlock
- Isolation & Dependency

1 – Hard Disk & Memory Hierarchy

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

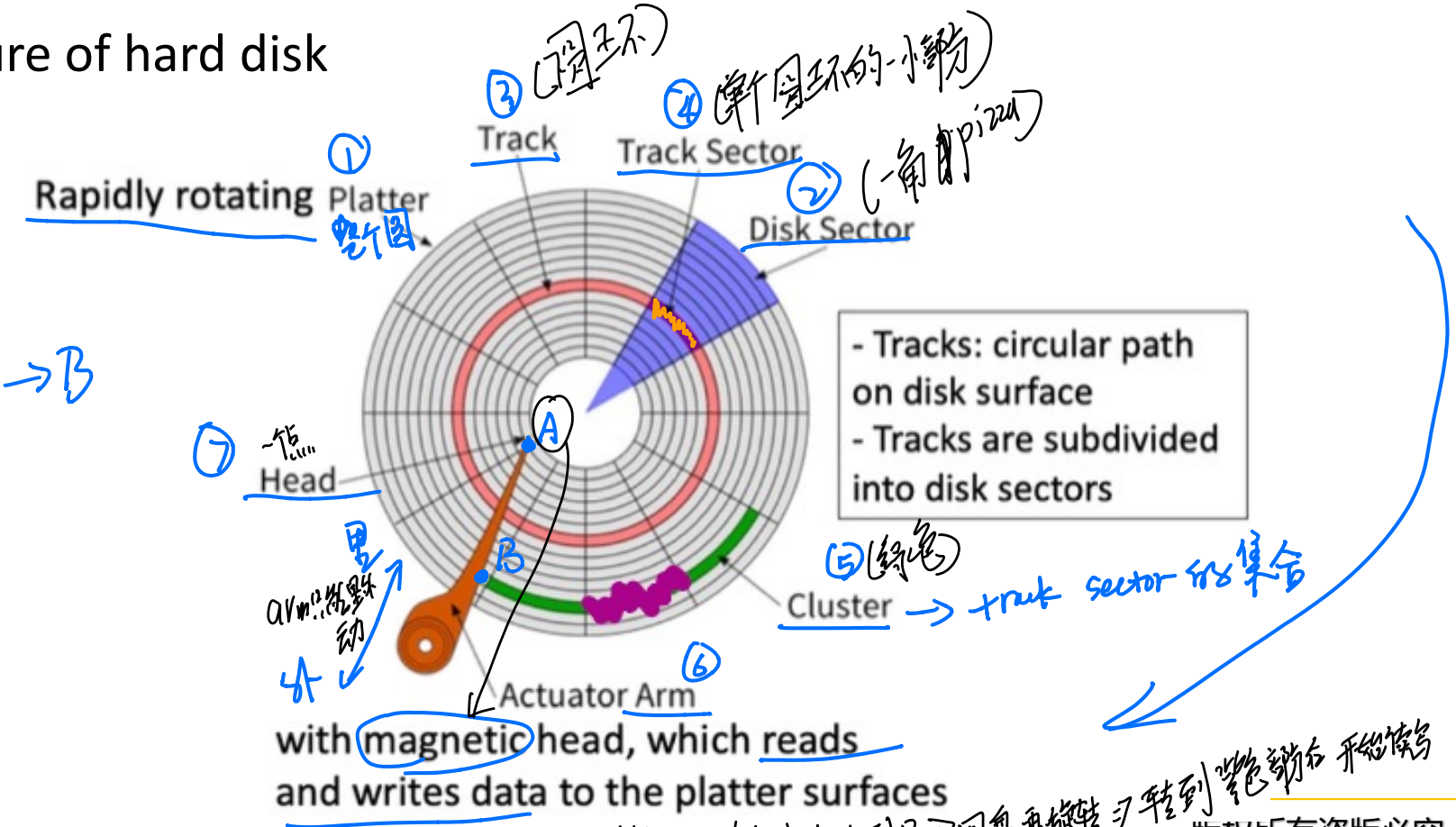
认识单位与数量级

大小(容量)的单位 Size

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

MB < GB < TB

Structure of hard disk



Disk Access Time

$$\text{Disk access time} = \text{seek time} + \text{rotational time} + \frac{\text{transferlength}}{\text{bandwidth}}$$

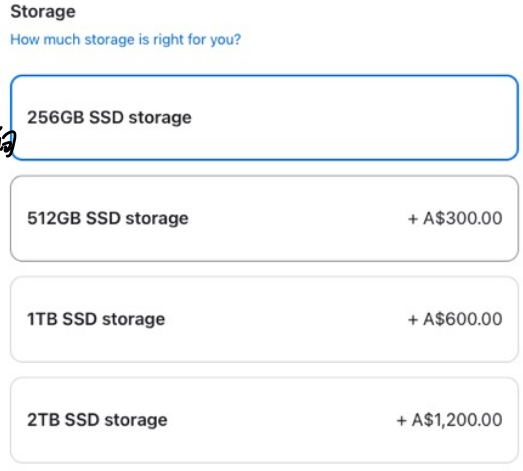
e.g., What is the Disk access time for a transfer size of 4KB, when average seek time is 12 ms, rotation delay 4 ms, transfer rate 4MB/sec?

$$\text{Disk access time} = 12\,\text{ms} + 4\,\text{ms} + \frac{4\,\text{KB}}{4\,\text{MB/sec}}$$

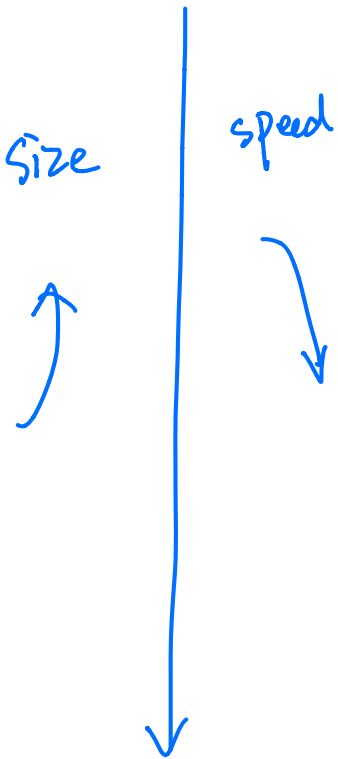
SSD (Solid State Disk/Driver)

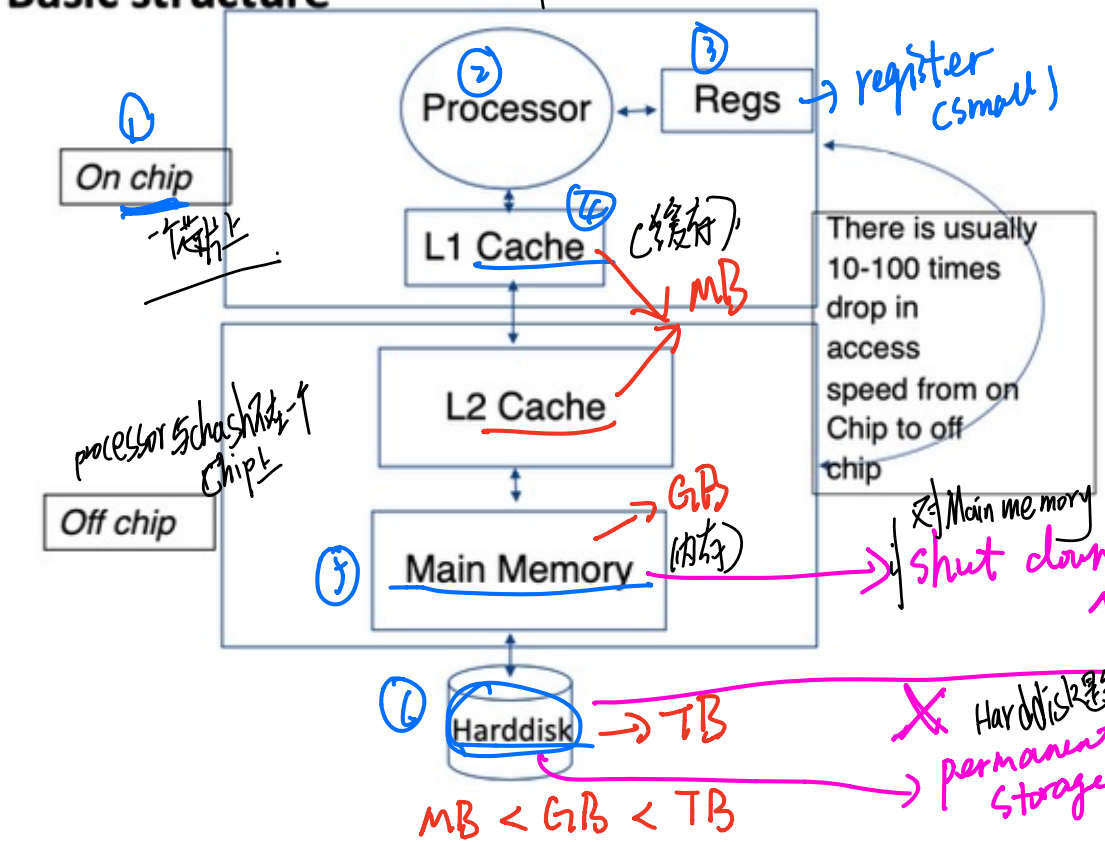
$$\text{Disk access time} = \frac{transferlength}{bandwidth}$$

- Silicon
- No seek/rotational time
- No start-up time like HDD
- Runs silently
- Very expensive



Memory Hierarchy





Effective memory access time

EA = H*C+(1-H)*M

H = hit ratio

C = cache access time

M = memory access time

$$Hit\ ratio = \frac{references\ satisfied\ by\ cache}{total\ references}$$

Q: Why not have large cache?

1. cache is expensive
2. if processor & cache are not in a same chip, access time will be long.

Disk buffer / Disk Cache

- Disk cache is different from the cache in memory hierarchy
- Disk cache is embedded in the disk

0.5×4 + 0.5×2016
= 1008 + 2
= 1010

Effective disk buffer access time

disk cache

EA = HB*BC+(1-HB)*D

HB = hit ratio of the disk buffer
BC = buffer access time
D = disk access time

e.g., 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%?

EA = HB * BC + (1-HB) * D
= 30% * C + 70% * S
= 0.3C + 0.7 * 1000C
= 700.3C

Hardware Limitations

- Moore's Law: Memory chip capacity doubles **every 18 months** since 1970
- Joy's law for processors: Processor performance doubles **every two years** since 1984
- **Hard disk was the foundation stone** for many DBMS design choices
- This means, in database system implementations, **hardware limitations effect** how things are done and this changes over decades, slowly but surely.
- In future, more designs choices for databases will be made **based on SSDs and CPU problems** and these will likely dominate the arena eventually

2 – Two types of Classifications of DB

① Type of DB systems (How the data are stored)

- Simple file
- Relational DB system
- Object Oriented DB system
- NoSQL
- Deductive DB system

② DB Architectures (How different machines are arranged together)

- Centralized (Client - Server)
- Distributed
- WWW
- Grid
- P2P
- Cloud

Type 1 - How the data are stored

Simple file

As a plain text file. Each line holds one record, with fields separated by delimiters (e.g., commas or tabs)

RDBS Relational database

record ↑ attribute ↑

As a collection of tables (relations) consisting of rows and column. A primary key is used to uniquely identify each row.

Table 1 : Student

primary key

ID	Name
1	Alice
2	Bob
1	Cindy X

Table 2 : Subject

ID	Subject
1	maths
2	English
"3"	Chinese X

Data type 类型也要一致 "3" 和 3

e.g. ID : int

Table 3 : Student - Subject

(enroll) primary key 组合

Student	subject
1	1
1	2
2	1
2	1 X

OO DB System

Data stored in the form of 'objects' directly (like OOP)

Student

String name;

int age;

⋮

String getName() {

return name;

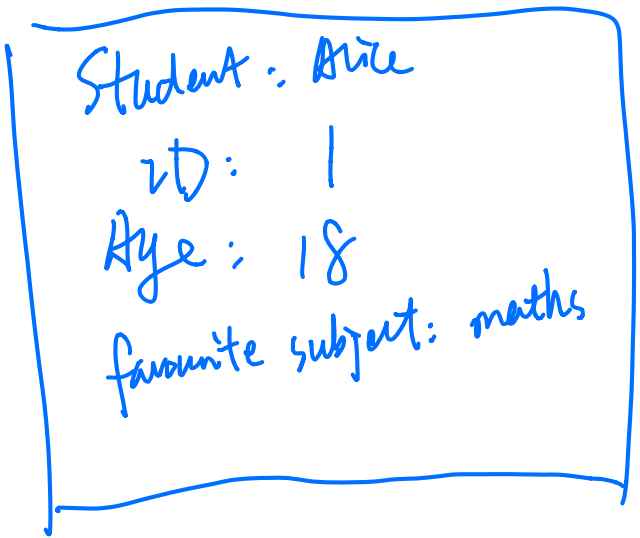
}

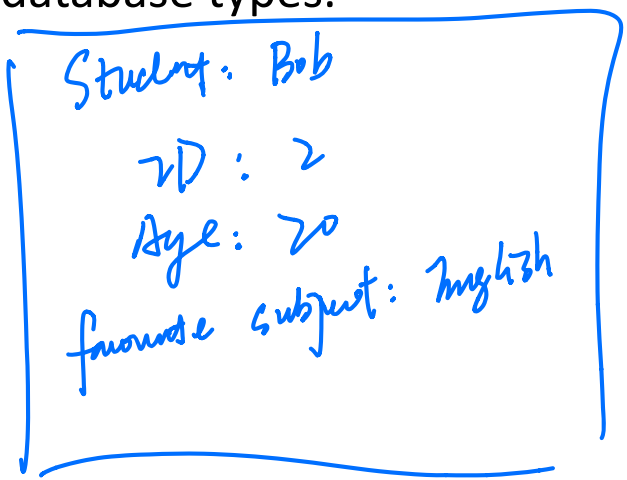
Java

这时了.

NoSQL (Key-value pair) 键值对 没有关联关系

Non relational – database modelled other than the tabular relations. Covers a wide range of database types.





演绎(推理数据库)

1:42前

Deductive DB system

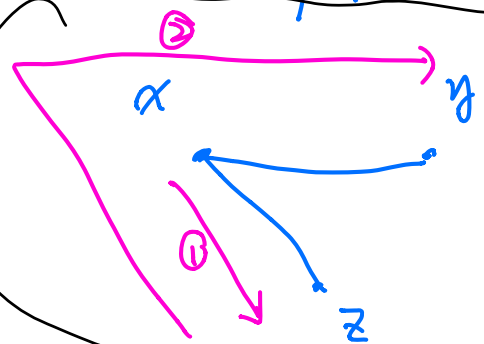
演绎的

A deductive database is a database system that can make deductions based on rules and facts stored in the database.

e.g.

edge：点到点

path：可以经过多个点



edge(x, y)

edge(x, z)

path(x → y) : edge(x, y)

path(x → y) : edge(x, z), ①

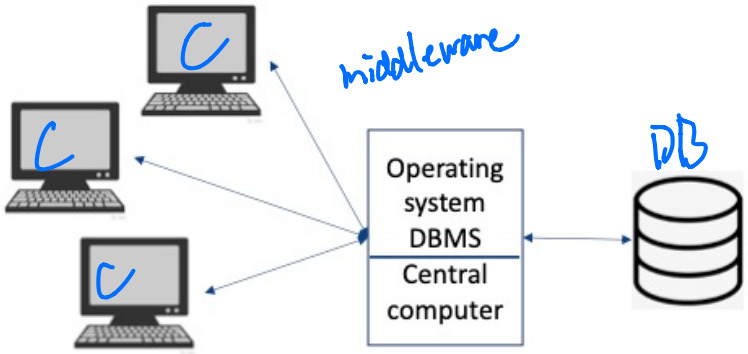
path(z, y) ②

Type 2 - How machines are arranged together

location (地点...)

administration (admin)

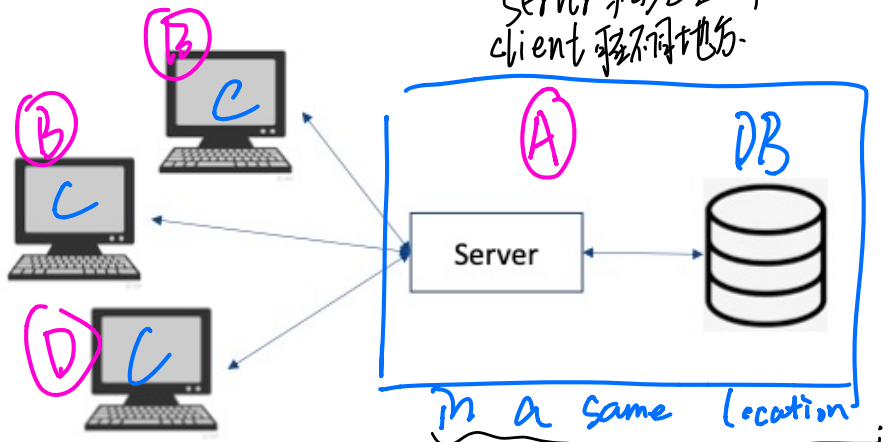
Centralized



in one location.

admin: simple

Client-Server



clients can be in different places

admin: simple => small company

provide concurrency, recovery and transaction processing different location

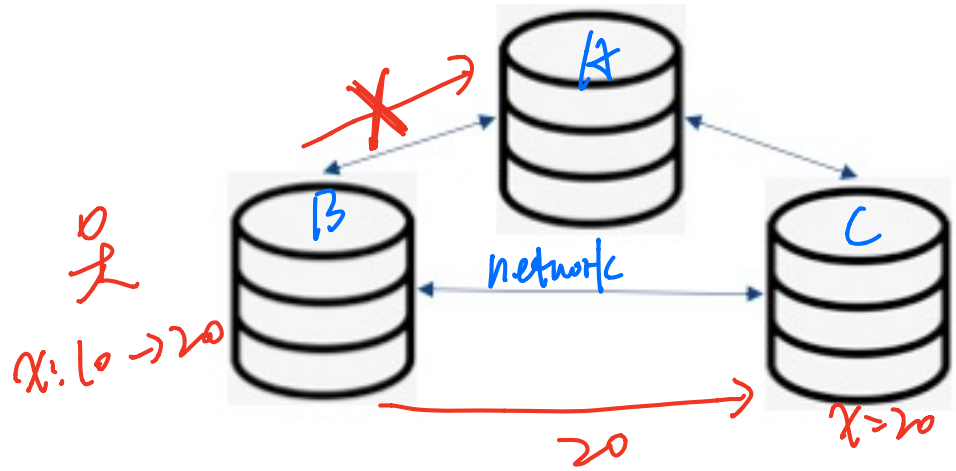
admin: 选一个 node

node / machine

从(A,B,C)

Distributed

three different locations 通过网络连接，选一个管理



问题：

→① inconsistency 数据不统一 (B→A, B→C, B→A 路径断了无法更新)

Dirty transaction,

Some nodes are updated,

but some are not.

→ crash recovery → complicated

down掉不好搜

Administration?

WWW world wide web

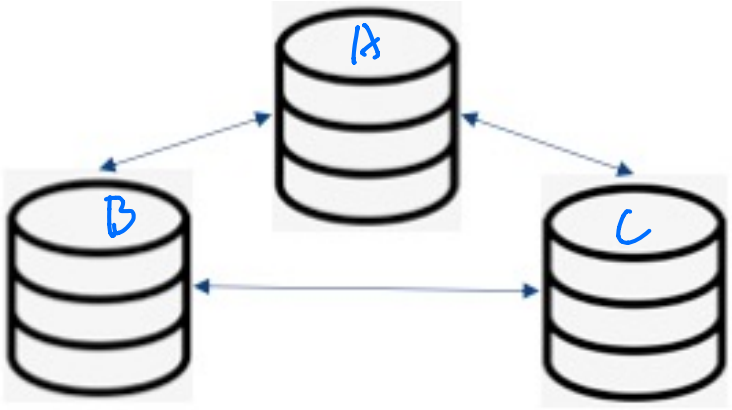
- easy & cheap to access from anywhere
- multiple owner (分布)
- e.g. unimelb.com.au
  - unimelb → uni
  - .au → gov
- admin: no central management
- security & privacy

1:53?

similar to Distributed DB

admin: done by Locally by each owner of the system

Grid → outdated



Administration? → manage data transfer & crash recovery

P2P Peer to Peer

different location

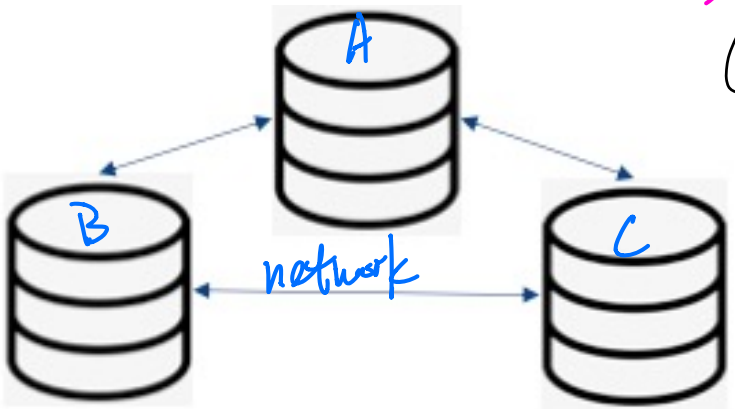
admin: done by owner (管理权谁管理)

=) nodes can join & leave the network

e.g. designed for particular use

相对特定的应用

distributed 和 Grid 无法做到



Administration?

Join and leave

No captical expenditure (no need for setting up Infra)

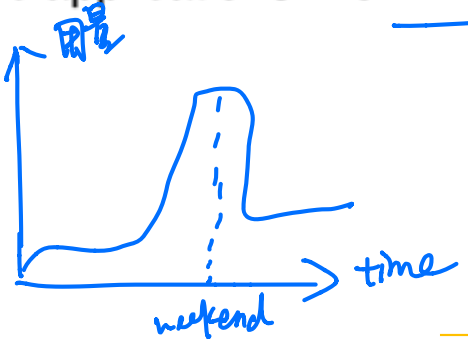
Cloud

cloud database → flexible → { you pay as you go ⇒ 按用量付费 ; 按用量 scale up/down }

Cloud services offered in several forms: EC2

- Iaas – Infrastructure as a service (provide virtual machines)
- Paas – Platform as a service (Provide environment like Linux, Windows, etc.)
- Saas – Software as a service (Specific applications like RDB, Mail, etc.)

you pay as you go



Type of DB systems (How the data are stored)

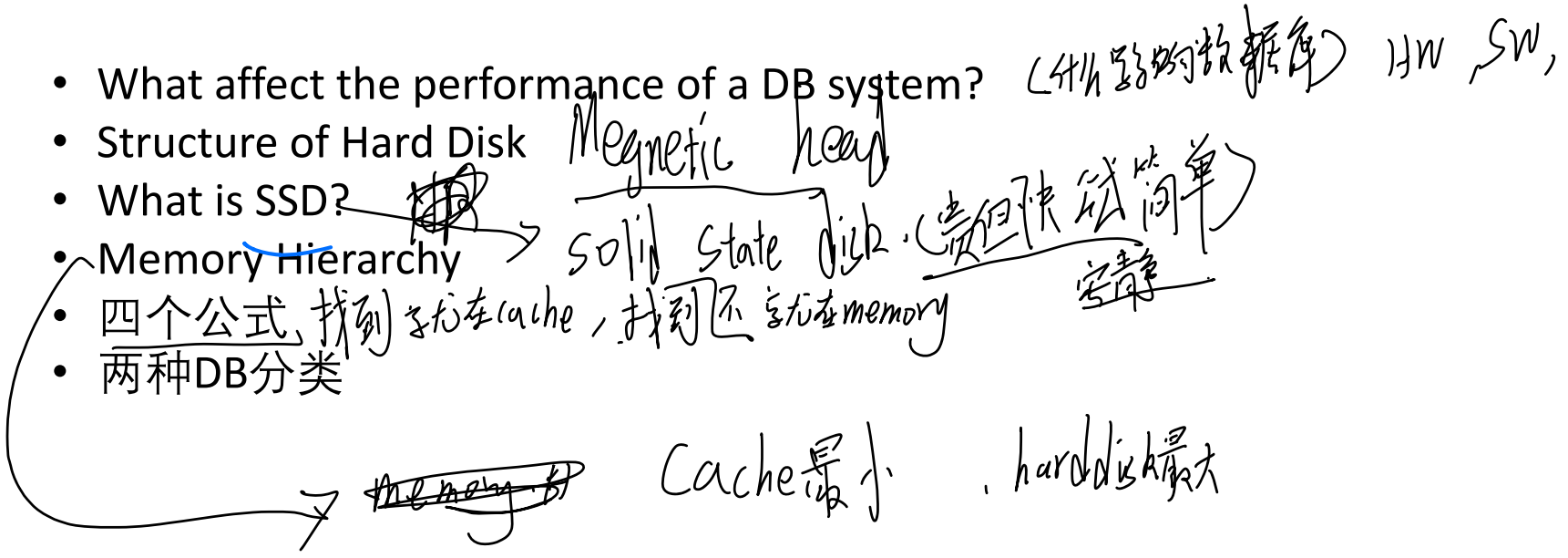
- Simple file
- Relational DB system
- Object Oriented DB system
- NoSQL
- Deductive DB system

DB Architectures (How different machines are arranged together)

- Centralized (Client - Server)
- Distributed
- WWW
- Grid
- P2P
- Cloud

Summary

- What affect the performance of a DB system?
- Structure of Hard Disk
- What is SSD?
- Memory Hierarchy
- 四个公式，找到字节在cache，找到不字节在memory
- 两种DB分类



Quiz

Q1

Why solid-state drives are faster than hard disk drives in general?

A. SSDs are expensive
B. Hard disk drives need to make backups more frequently
C. SSDs do not have any moving part
D. SSDs are smaller in size

Q2

Machine A has a higher cache hit ratio than Machine B. The cache access time and the memory access time of both machines are the same. Which machine has a faster effective memory access time?

effective memory access time = Hit Ration x Cache access time + (1-H) x M.
(M: memory access time.)

$H_A > H_B$ assume $H_A = 80\%$ $H_B = 0.3$

A. Machine A

B. Both machines have the same effective memory access time

C. Machine B

$E_A = 0.8C + 0.2M$

$E_B = 0.3C + 0.7M$

$\Rightarrow$ 为M=100C 接等数量级 $\Rightarrow$ C=1, M=100

$E_A = 0.8C + 20C < E_B = 0.3C + 70C$.

Q3

Jane wants to add an online shopping cart option to her local clothing store website. The users do not need to register to shop online there, but they get a temporary ID when they start adding items to the online shopping cart. Each time an item is added to a cart, an entry needs to be stored with that user's ID and the item's ID in a database. Which of the following options is a more suitable type of database in this context?

A. Deductive database systems
B. Document storages
C. Key-value storage
D. Graph databases

value

Key → 时间 Value → User's ID and item's ID

Q4

A database stores the email address of customers who want to receive marketing information in the future. Customers can login to the database via a web interface. The customers can create or modify their email addresses after logging in. The database has one million tuples currently. Which of the following should be achieved?

A. The data will not be lost if one of the database servers broke down
B. The database stores the true email address of customers
C. Different customers cannot modify their email addresses at the same time
D. Full backup of the database is created every second

Q5 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?

{store and manage the data of its own sensor → admin by its own
multiple sensors in melbourn → different location
(connect → share → disconnect) → join and leave

- (●) P2P databases
- ( ) Cloud storage
- ( ) Centralised database
- ( ) Distributed database

COMP90050 ADBS

授课老师：Rita

1 – Query Processing & Optimisation

基本概念

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

How data is stored in disk?

- Files – A database is mapped into different files. A file is a sequence of records.
- Data blocks – Each file is mapped into fixed length storage units, called data blocks (also called logical blocks, or pages)

e.g., size of each record: 55 byte; fixed size of 1 data block: 4096 byte

Table – "**Employee**"

ID	Name	Age
1001	A	25
1002	B	32
1003	C	19
1004	D	27
1005	E	40
1006	F	36

Table – "**Manager**"

ID	Department
1001	a
1003	b
1004	c
1006	d

Table – "**Location**"

ID	Country	City
1001	China	Beijing
1002	Australia	Melbourne
1004	China	Shanghai
1006	Australia	Sydney

- Attribute (column)
- Record (row)
- Query
  - Please tell me the name of all employees who is over 25
  - please tell me the name of all managers
  - Please tell me the name of all managers located in Australia
- Join

```
Select Name
from (Employee inner join Manager)
On Employee.ID = Manager.ID;

Select Name
from (A inner join location)
On A.ID = Location.ID
Where Location.country == 'Australia';
```


Join

Join is a very common and very expensive operation

Different types of join: inner join, outer join… (we focus on inner join)

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

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

Relational Algebra Expressions

Select * *from* T1

inner join T2

on T1.a = T2.b

```
SELECT *
FROM Employees
INNER JOIN Managers
ON Employees.ID = Managers.ID;
```

…. Is same as the following in relational algebra expression:

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

Relational Algebra Expressions

select A1, A2, ..., An
from r1, r2, ..., rm
where P

Select salary
From Employees
Where salary < 60000

.... Is same as the following in relational algebra expression:

$$\Pi_{A_1,A_2,\ldots,A_n}(\sigma_P(r_1 \times r_2 \times \ldots \times r_m))$$

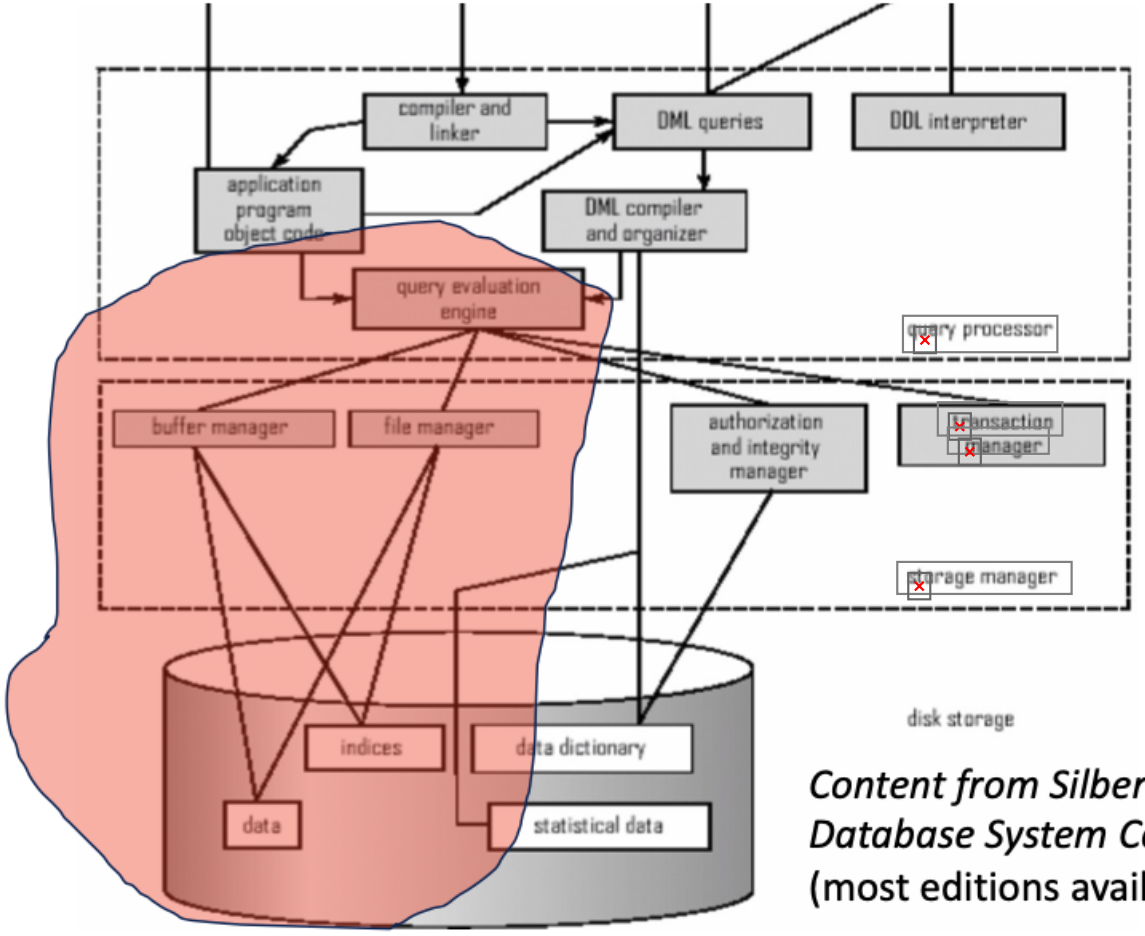
Equivalent Expressions

Can also be written as:

$$\Pi_{salary}(\sigma_{salary<60000}(\text{Employees}))$$

$$\sigma_{salary<60000}(\Pi_{salary}(\text{Employees}))$$

DB Engine



Content from Silberschatz et al Database System Concepts
(most editions available online)

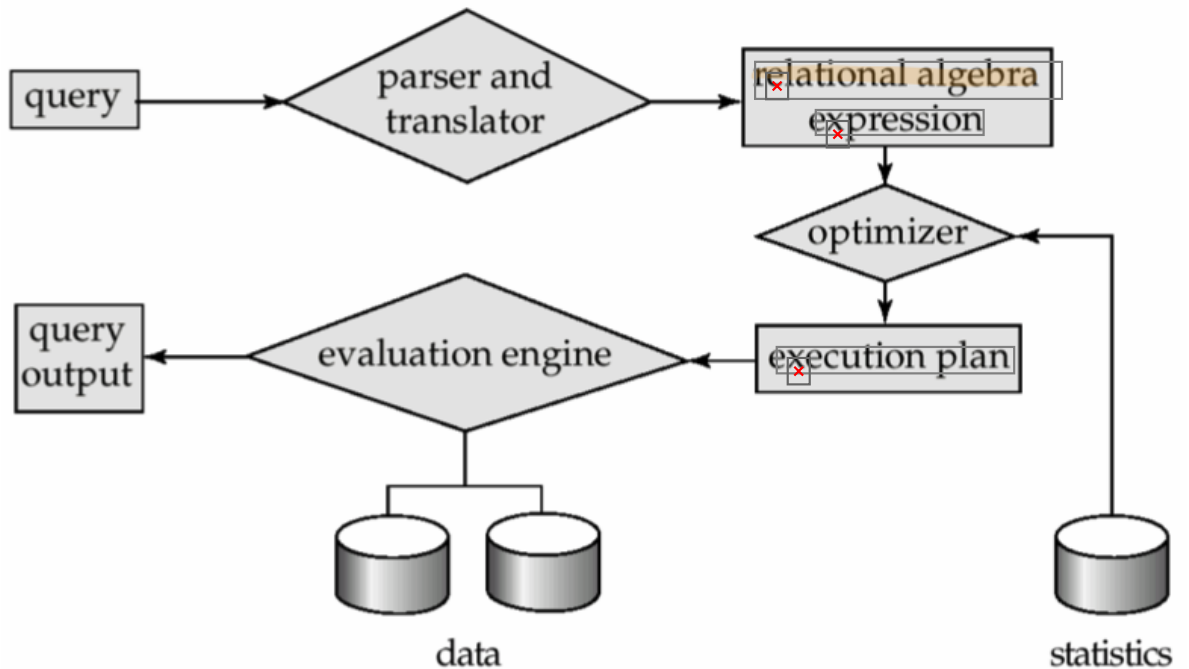
- **Hardware**
  - – The speed of the processor
  - – Number of processors
  - – Number of disk drives and I/ O bandwidth
  - – Size of main memory
  - – Communication network
  - – Type of architecture

DB Engine

- **Software**
  - – Type of database technology used for a given application
- **Database tuning, crash recovery**
  - – Indexing parameters
  - – Data duplication
  - – Sharing data, etc

Query Processing Steps

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



- Translate query to relational algebra expression
- Make execution plan

Nested-loop Join

for each tuple t_r **in** r **do begin**
 for each tuple t_s **in** s **do begin**
  test pair (t_r, t_s) to see if they satisfy the join condition theta (θ)
  if they do, add $t_r \bullet t_s$ to the result.
 end
end

- 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).

In the worst case, if there is enough memory only to hold one block of each table, the estimated cost is

$n_r * b_s + b_r$ block transfers, and
$n_r + b_r$ seeks

Block Nested-loop Join

```
for each block B_r of r do begin
    for each block B_s of s do begin
        for each tuple t_r in B_r do begin
            for each tuple t_s in B_s do begin
                Check if (t_r, t_s) satisfy the join condition
                if they do, add t_r • t_s to the result.
            end
        end
    end
end
```

- Variant of nested- loop join in which every block of inner relation is paired with every block of outer relation.

$$b_r * b_s + b_r \text{ block transfers} + 2 * b_r \text{ seeks}$$

n_r - >b_r

e.g.,

- Number of records of *customer*: 10,000 *depositor*: 5000
- Number of blocks of *customer*: 400 *depositor*: 100

Nested-loop Join

Depositor as the outer relation

Customer as the outer relation

Block Nested-loop Join

Depositor as the outer relation

Customer as the outer relation

Query Optimization

Query optimization is about the right choices and annotations

What to optimize?

- e.g., I have 3 tables, which two can I join first?
- Join algorithms

Query Plans

Alternatives

Execution Plan

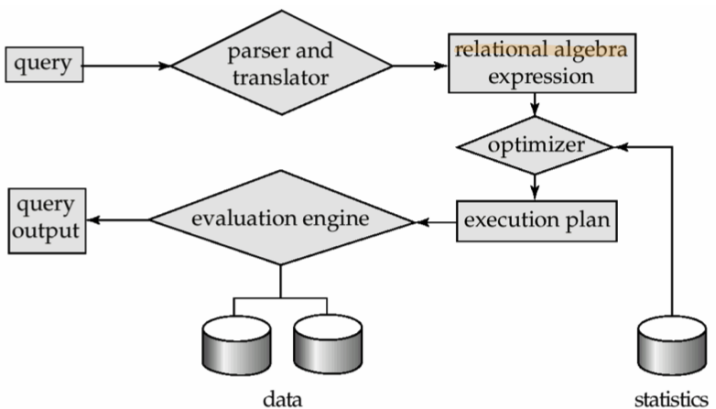
Final decision

How do we make the choices?

steps in cost-based query optimization

Cost: how many blocks are read from disk

1. Generate logically equivalent expressions of the query
2. Annotate resultant expressions to get alternative query plans
3. Choose the cheapest plan based on estimated cost



Estimation of plan cost based on:

- Statistical information about tables. Example:

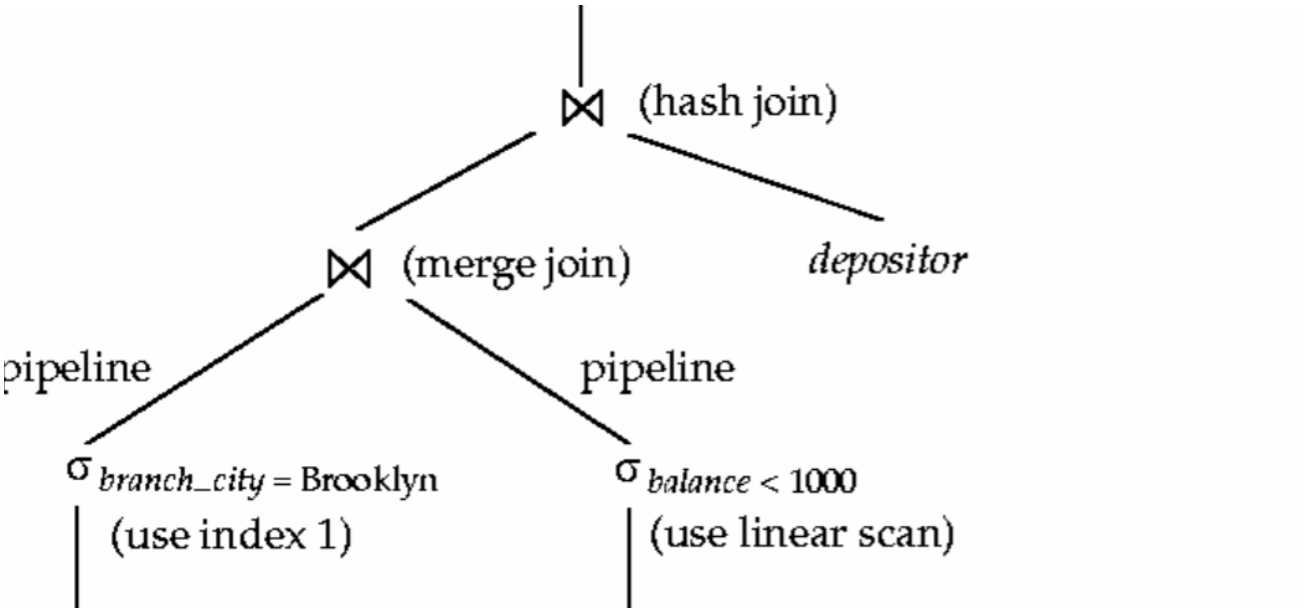
  number of distinct values for an attribute

- Statistics estimation for intermediate results to compute cost of complex expressions

- Cost formulae for algorithms, computed using statistics again

2 – Query Optimization in Real Life

Query Optimization is about the right choice on a graph



How to generate alternatives?

- Query optimizers use equivalence rules to systematically generate expressions equivalent to the given expression
- One can generate all equivalent expressions exhaustively
- The above approach is very **expensive** in space and time though

(In query optimizer, some expressions are not generated if they are for sure very complex)

But

- Must consider the interaction of evaluation techniques when choosing evaluation plans
- Choosing the cheapest algorithm for each operation independently may not yield best overall algorithm (每步最好的，不一定overall最好)

E.g., merge- join may be costlier than hash- join, but may provide a sorted output which could be useful later (the sorted result may be benefit to the later operation)

In Real Life

- Practical query optimizers incorporate elements of the following two broad approaches:
  1. Search all the plans and choose the best plan in a cost- based fashion.
  2. Uses **heuristics** to choose a plan.

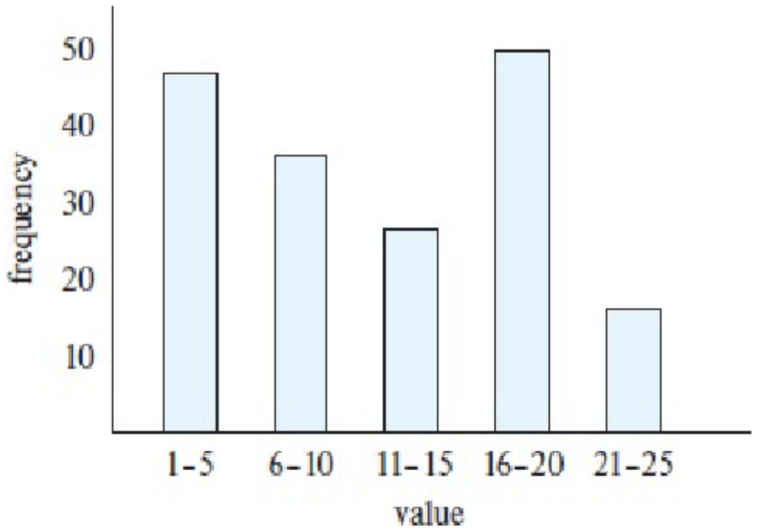
- Systems may use heuristics to **reduce the number of choices** that must be made in a cost- based fashion (because cost- based optimization is expensive)

- Heuristic optimization transforms the query- tree by using a set of rules that typically (but not in all cases) improve execution performance:
  1. **Perform selections early** (reduces the number of tuples)
  2. **Perform projections early** (reduces the number of attributes)
  3. **Perform most restrictive selection and join operations** (i.e. with smallest result size) before other similar operations

- Some systems use only heuristics, others combine heuristics with cost- based optimization

- Optimizers often **use simple heuristics for very cheap queries** , and perform **exhaustive enumeration for more expensive queries**

- Further Optimisations
  - Sampling
  - Histogram



- Adaptive Plans - Wait for one/ some parts of a plan to execute first, then choose the next best alternative

3 – Query Cost in Practice

Troubleshooting to manage costs (e.g., Query Store in Microsoft)

(Query store: SQL server management studio for monitoring)

- Identify ‘regressed queries’ - Pinpoint the queries for which execution metrics have recently regressed (for example, changed to worse).

- Track specific queries - Track the execution of the most important queries in real time. (e.g., most frequently asked queries)

When you identify a query with suboptimal performance
(没有达到最好的performance)

- Force a query plan instead of the plan chosen by the optimizer
- Do we need an **index**? - - - quickly find the data in the query
- Enforce statistic recompilation (重新编译)
- Rewrite query? (with parameters)

Parameter in query

Query rewriting with parameters for execution plan reuse

SELECT * FROM Product WHERE categoryID = 1;	SELECT * FROM Product WHERE categoryID = 4;

They use the same plan!

We expect the optimizer to generate essentially the same plan and reuse the plans - parameterize

"1" & "4"

```
DECLARE @MyIntParm INT
SET @MyIntParm = 1
EXEC sp_executesql
  N'SELECT -
  FROM Product
  WHERE categoryID = @Parm',
  N'@Parm INT',
  @MyIntParm
```


To further lower query cost

- **Store derived data**
  - When you frequently need derived values
  - Original data do not change frequently

- **Use pre-joined tables**
  - When tables need to be joined frequently
  - Regularly check and update pre- joined table for updates in the original table
  - May still return some ‘outdated’ result (pre- joined tables are not updated)