CS307
Principles of Database Systems

Chapter 13

Shiqi YU 于仕琪

yusq@sustech.edu.cn

13.1 Query Optimizer

Shiqi Yu 于仕琪

yusq@sustech.edu.cn

ORACLE

1978 Version 1

1985 Version 5

SYBASE

1987 Version 1

PostgreSQL

1988 Version 6

1989

1988 Prototype

Microsoft SQL Server

MySQL

1994 First release

ALL major DBMS products were designed in the 1980s (MySQL is more recent but was based on SQL Server). They CANNOT change their architecture because it would be, business-wise, suicidal for them to require from customers dumping and reloading today's massive databases for a migration. Only incremental improvements are possible.

Mid 1980s looking like in the 1980s.

VAX 8600 (high-end)

32-bit architecture

Processor, ~10 to 20MHz clock

4 to 256 Mb of memory

I/Os ~10 to 30 Mb/s

This is what a $M 1 machine was

digital

Beginning of multi-processor computers

Keep in mind that the big DBMS products were designed for this.
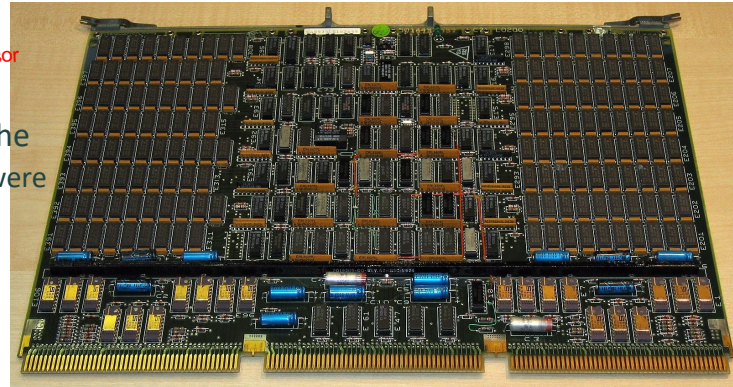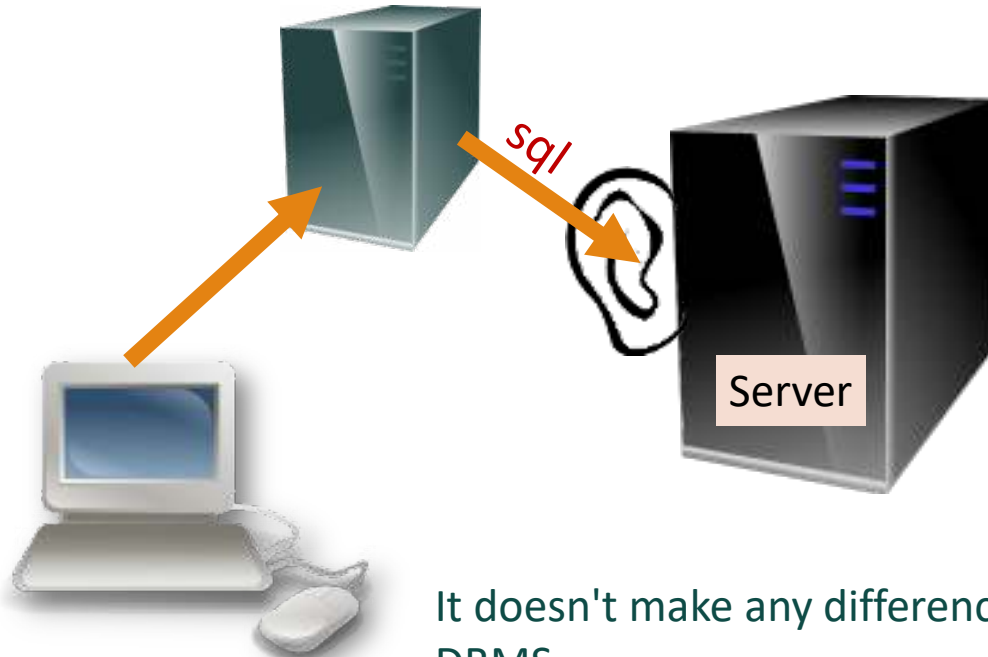
A 4 MB memory board for the VAX 8600

SQL queries will be directly sent to this server. The listener is only here for new connections.

sql
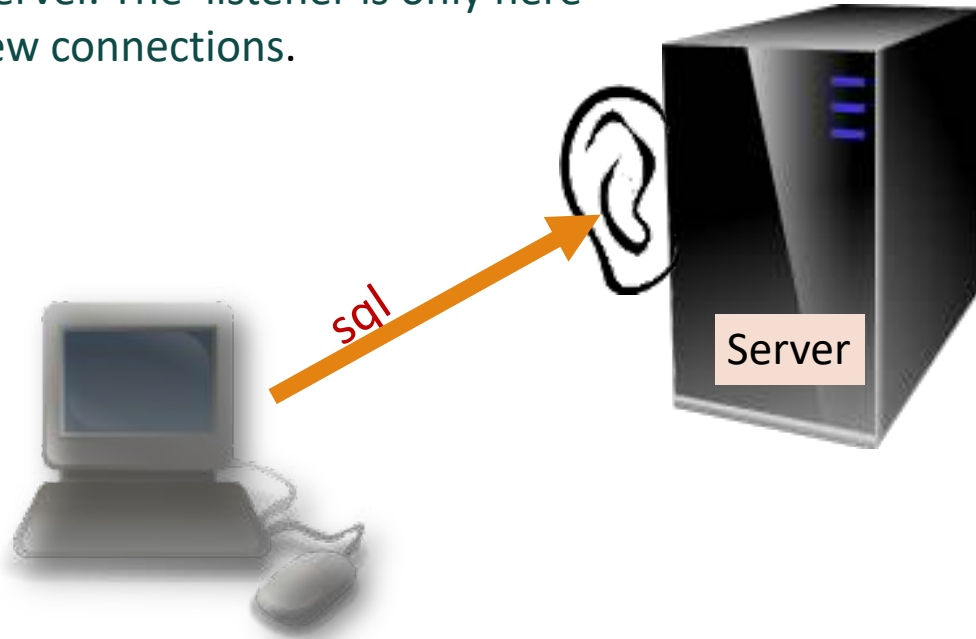
Server

In many cases (HTTP server, application server) the
end user isn't directly talking to the DBMS server.

sql

Server

It doesn't make any difference for the
DBMS.

```sql
select m.title, m.year_released
from movies m
    inner join credits c
    on c.movieid = m.movieid
    inner join people p
    on p.peopleid = c.peopleid
  where p.first_name = 'Tim'
   and p.surname = 'Burton'
   and c.credited_as = 'D'
```

Syntax ✓
Do tables exist? ✓
Right to access? ✓
Do columns exist? ✓
Indexes we can use? Best way
to access data? ✓

One way to improve efficiency is to keep data dictionary information (meta-data) in a shared cache to avoid additional queries.

Kept in memory

shared ➡ meta-data

```
select m.title, m.year_released
from movies m
    inner join credits c
    on c.movieid = m.movieid
    inner join people p
    on p.peopleid = c.peopleid
where p.first_name = 'Tim'
 and p.surname = 'Burton'
 and c.credited_as = 'D'
```

Syntax ✓
Do tables exist? ✓
Right to access? ✓
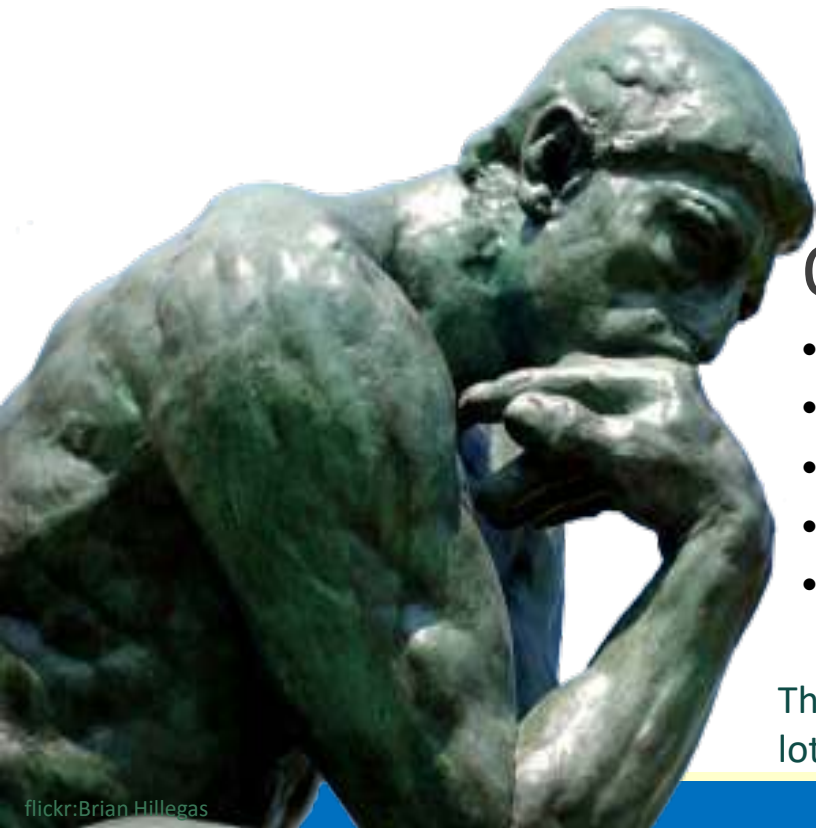**Do columns exist?** ✓
Indexes we can use? Best way to access data? ✓

Another crucial phase is the one when the optimizer tries to determine the most efficient way to access data.

Kept in memory

shared

meta-data

QUERY OPTIMIZER

- Logical transforms
- Indexes Volumes Storage
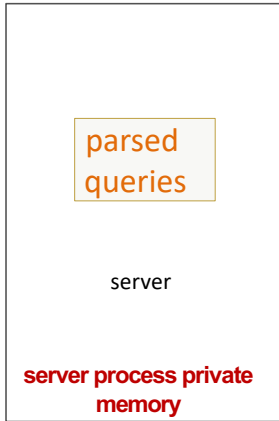- Hardware performance
- System load
- Settings

The optimizer has to (or can) take into account a lot of factors.

PARSING parsing
takes time

Let's put it another way: we'd rather not parse exactly the same query many times.

Keep parsed queries in memory

As most applications run exactly the same SQL statements again and again, a DBMS will cache a parsed query for reuse. For MySQL, it will be cached for a session.

Query cache management
LRU

Least Recently Used

Of course we cannot hold in cache zillions of parsed queries. We need to manage the cache, and replace queries that haven't been executed in a while with new ones.

```
select m.title, m.year_released
from movies m
    inner join credits c
    on c.movieid = m.movieid
    inner join people p
    on p.peopleid = c.peopleid
where p.first_name = 'Tim'
 and p.surname = 'Burton'
 and c.credited_as = 'D'
```

校验码
Checksum
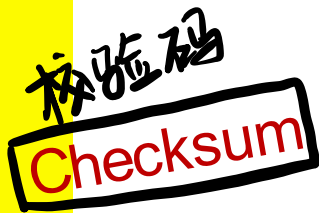
We primarily recognize
identical queries by
computing a text checksum.

+ check tables are same
and context identical

There are a lot of things to do for optimization because the resource is limited even we have a powerful server.

13.2 SCALING

Shiqi Yu 于仕琪

yusq@sustech.edu.cn

SCALING UP

For many years, the answer to a database outgrowing the processing power of its server has been to replace the server by a bigger server.

Having a single server is taking a risk if it breaks, and additionally migrating to another server is disruptive.

SCALING **OUT**

This is why people quickly thought of an alternative, adding more servers and making them share the load.

In the 1980s, Oracle tried connecting multiple servers to a single database. Complete disaster when the two servers wanted to modify the same data (or simply when one wanted to see what had just been modified by the other), data had to transit via the files.

Oracle Parallel Server
(1980s)
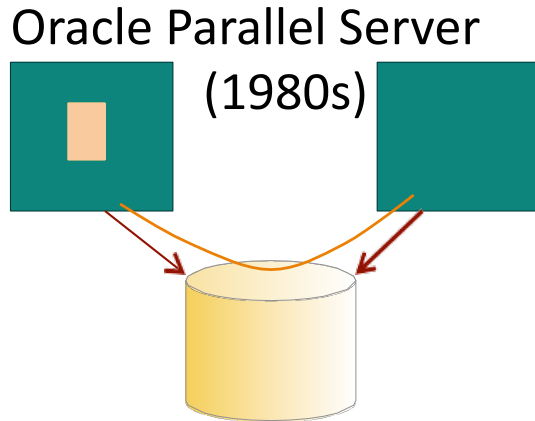
Oracle RAC
(2001)
Real Application Clusters

Oracle came out with a very good clustering offer, in which a very-high-speed private network was connecting the servers for exchanging data blocks.

Neil Gunther's
Universal Law of Computational Scalability

Relative capacity $C(N)$ of a computational platform:

$$C_{(N)} = \frac{N}{1 + \alpha\,(N-1) + \beta\,N\,(N-1)}$$

$$0 <= \alpha,\ \beta < 1$$

α Level of contention
β Coherency delay (latency for data to become consistent)

The problem with clustering is that it follows a law of diminishing returns: adding a second server will less than double your capacity, and in practice people have clusters of 2, 3 or 4 machines at most (Neil Gunther is a famous Australian consultant/academic specializing on performance)

One big problem is with transactions that involve several servers. Remember that transactions are meant to be atomic operations.

Distributed Transactions

It may happen that when we commit we know for sure it worked on one server, and we get no acknowledgment from the other.

commit

OK

Change

?
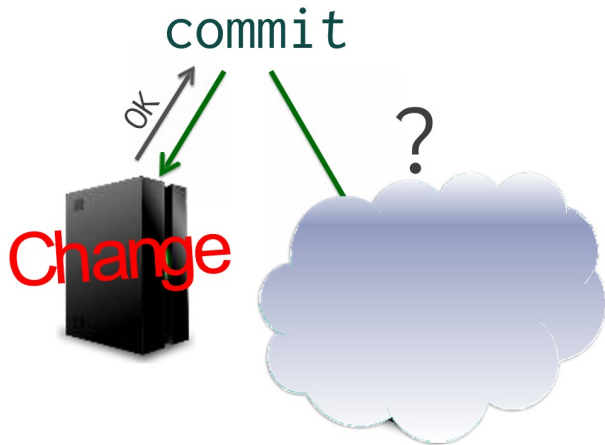
No way to rollback where it's committed, and we don't know if the other node failed before or after having committed the change.
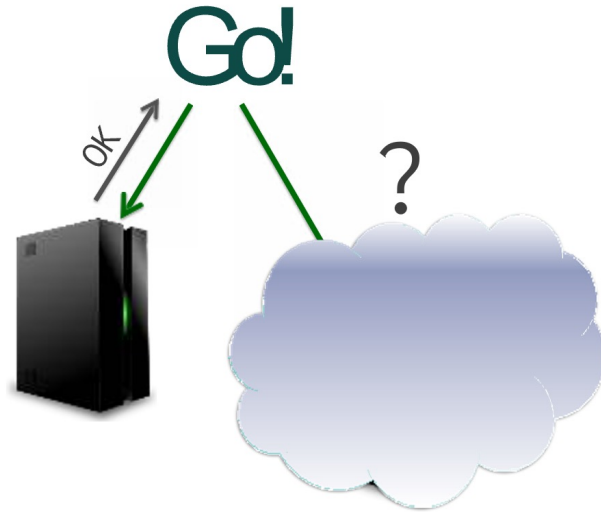
TWO-PHASE COMMIT

One algorithm was devised (a long time ago), called a "two-phase commit".
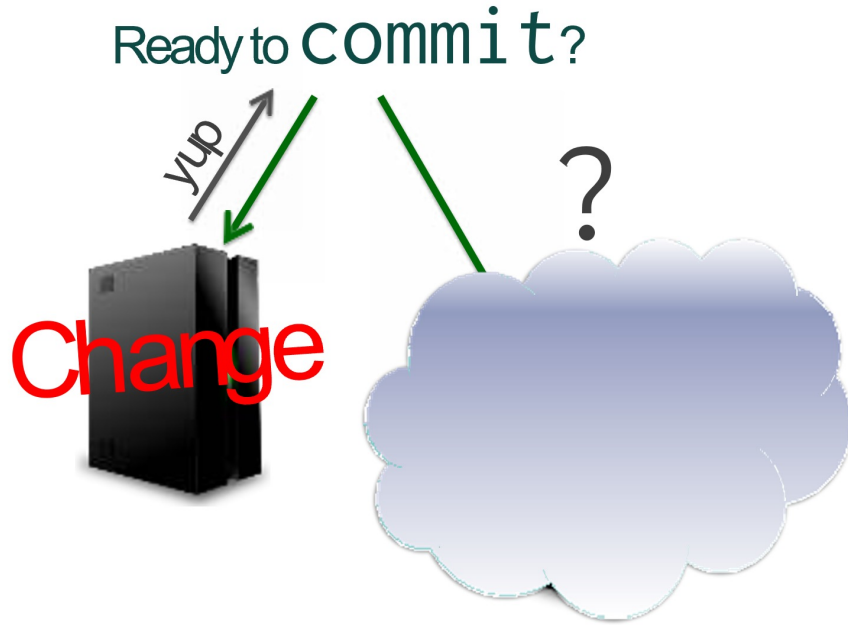
JUST before committing, you ask to all servers involved whether they are ready to commit. If you don't get an all clear, you can rollback.

Otherwise, you can then (2nd phase) send the official "commit" signal

Odds that something will fail are far lower than with a single-phase commit, but not zero.

Latency

Additionally, you have latency issues. All machines in a cluster may not be sitting next to each other, they may be a few miles apart in different data centers for security reasons (fire, flood …)

1 KM = 0.005ms
2000 KM = 10ms

Distance from Shenzhen to Beijing

Even if information travels fast, multiplying exchanges (two-phase commit) may become a sensitive issue.