Database internals - Basics to advanced

- Hrishikesh Kulkarni

Row oriented database
vs
column oriented database

-
  ```
  SELECT *
  FROM employees
  WHERE profile_id=100
  ```

-
  ```
  SELECT SUM(salary)
  FROM employees
  ```

Keys in RDBMS

- Primary key
- Candidate key
- Foreign key

We can put constraints on columns (NOT NULL, UNIQUE)

Transaction

- Collection of queries
- Represents one unit of work
- Can have - SELECT, INSERT, UPDATE, DELETE (Data Modificn/Manipuln Lang)
- Example
- Read lock (Shared lock)
- Write lock (Exclusive lock) - Locking at row level, table level or DB level

Commands:

- Transaction BEGIN
- Transaction COMMIT
- Transaction ROLLBACK - Can be user defined or used by system for unexpected ending

Can transaction contain only SELECT (read) statements?

ACID

- A: Group of statements in transaction should be treated as single unit. Either success or failed transaction.
- C: Database should move from one state to other which is consistent.
- I: Result of execution of concurrent transactions == Result of executing transactions serially
- D: Once commit is done, data should be persisted

Optimistic lock: No locking. Keep track of value using version/snapshot/last update time (epoch). If change in data - fail transaction. SerializableException

Pessimistic lock: Lock at table/row level - so dirty reads are avoided

Index

- Index has pointers to the page (physical location of record's page)
- Index can be on single column or multiple columns
- B+ tree is used internally
- Index improves lookup performance. How?
- Index reduces write performance. How?
- Types of index:
  - Clustered index (Physical location storage sorted)
  - Non-clustered index (Use dense mapping)

How tables and indexes stored on disk?

Logical Table

emp_id	emp_name	emp_dob	emp_salary
1	Alice	01.01.1990	$$$
2	Bob	05.04.1992	$$$
...	...	...	...
1000	...	...	...

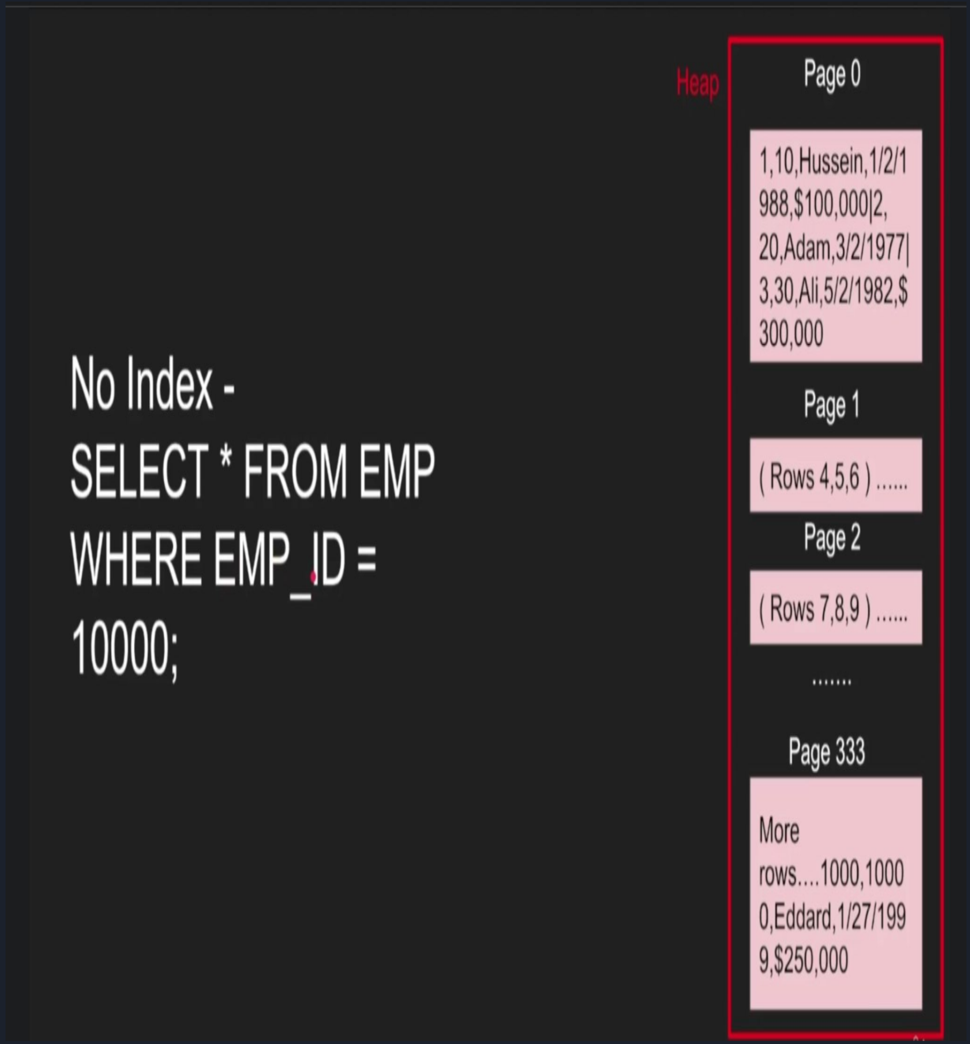
I/O can't read single record, reads page

Page 1 (3 records)
1,Alice,01.01.1990,$|2,Bob,05.04.1992|3,Lorel,01.02.1993

Page 2
(Next 3 records)

…..

Page 334



No Index -
SELECT * FROM EMP
WHERE EMP_ID =
10000;

Heap

Page 0

1,10,Hussein,1/2/1988,$100,000|2,20,Adam,3/2/1977|3,30,Ali,5/2/1982,$300,000

Page 1

(Rows 4,5,6) ......

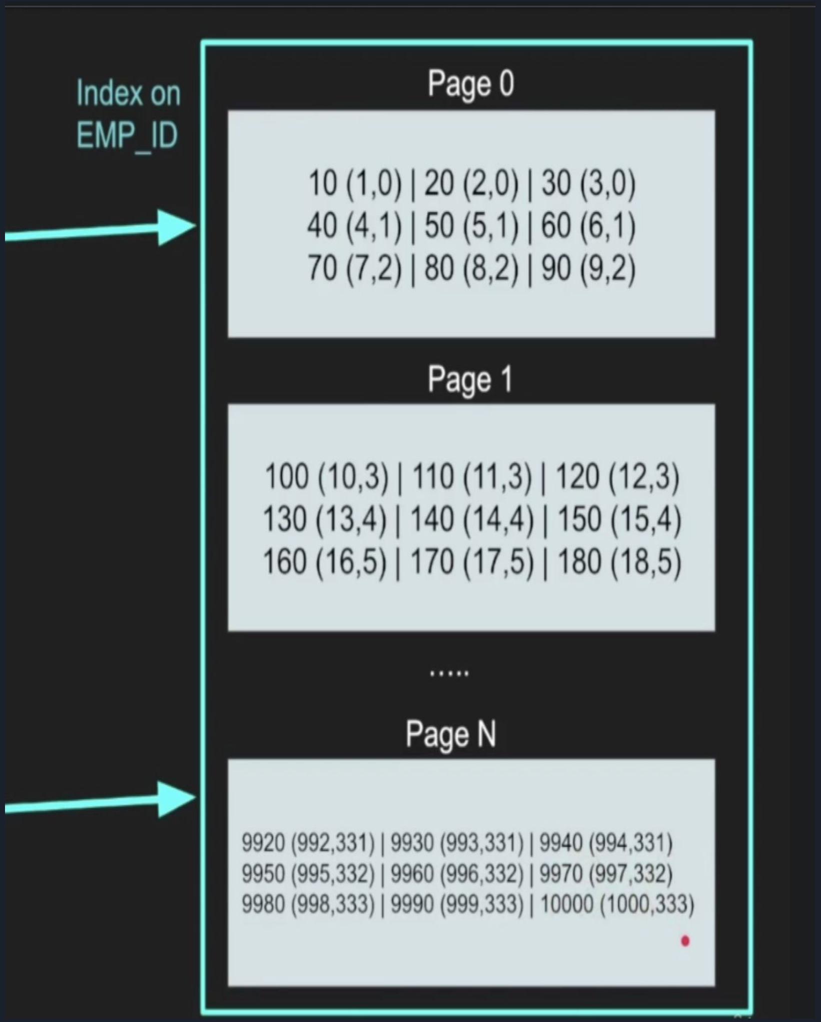
Page 2

(Rows 7,8,9) ......

....

Page 333

More rows....1000,10000,Eddard,1/27/1999,$250,000



Index on EMP_ID

Page 0

10 (1,0) | 20 (2,0) | 30 (3,0)
40 (4,1) | 50 (5,1) | 60 (6,1)
70 (7,2) | 80 (8,2) | 90 (9,2)

Page 1

100 (10,3) | 110 (11,3) | 120 (12,3)
130 (13,4) | 140 (14,4) | 150 (15,4)
160 (16,5) | 170 (17,5) | 180 (18,5)

.....

Page N

9920 (992,331) | 9930 (993,331) | 9940 (994,331)
9950 (995,332) | 9960 (996,332) | 9970 (997,332)
9980 (998,333) | 9990 (999,333) | 10000 (1000,333)

Partition

- What is partitioning?
- Horizontal partitioning vs Vertical partitioning
- Partitioning by
- Partition vs shard

Remember:

Quickest way to avoid querying billion rows is avoid querying billion rows

```
SELECT *
FROM employees
 WHERE id=700,001
```

employees

id	name	...
1	a1	...
2	b1	...
3	c1	...
...	...	...
...	...	...
700001	za	...
...	...	...
1M	xz	...

id	name	...
1	a1	...
2M	...	...

id	name	...
2M1	abc	...
4M	...	...

id	name	...
4M1	bcd	...
6M	...	...

id	name	...
1	a1	...
200,000	...	...

id	name	...
1	a1	...
200,000	...	...

Partition types

- By range (Dates, id range etc)
- By unique values (Location name, zip, gender, categorized values)
- By hash function

Horizontal Partition vs Sharding

- HP splits big table into multiple tables in the same db server (client is agnostic)
- Sharding splits big table into multiple table across multiple db servers. Sharding helps to change configuration of diff db servers resulting in scalable solution

Pros and cons

- Pros:
  - Improves query performance due to lesser data in each table
  - Archive older data by storing partition into cheap storage
  - Scalability
- Cons:
  - Slow write when updated row needs to move from one partition to another
  - Inefficient queries may end up looking across all partitions
  - Schema changes can be difficult
  - Difficult for transaction across multiple shards
  - Difficult for joins across multiple shards

Let's design BookMyShow!

Requirements (F and NF)

- Filtering via location, date and show all movies available on homepage.
- Provide ability to change location, date etc.
- After selecting movie, show all theatres. After selecting theatre with required time (start time), show seats.
- Select seats and provide ability to book them online.

Estimations (Come back after schema for storage estimation)

-

Exposed APIs

-

High Level Design and DB Schema

Detailed Discussion and Data Flow

-

Identifying and Resolving bottlenecks

-

Similar question: Design OYO/Airbnb/Goibibo/Redbus/Airline management

Requirements (F and NF)

- Onboard new hotels. Hotel related information can be updated. Hotel can have different types of rooms.
- Hotel owners should be able to see all bookings, revenue etc.
- User should be able to search in location, do filtering based on cost, tags (5*, waterfront etc.)
- User should be able to reserve rooms (for simplicity lets assume user can book only one kind of rooms)
- User can see his/her reservations and previous bookings

Requirements (F and NF)

- Low latency, high availability for search
- High consistency for booking
- 100k hotels -> 500 rooms/hotel
-
- DAU: 5 million users/month
- Global vs specified
-

Estimations

Exposed APIs

-

High Level Design and DB Schema

-

Detailed Discussion and Data Flow

-

Identifying and Resolving bottlenecks

-