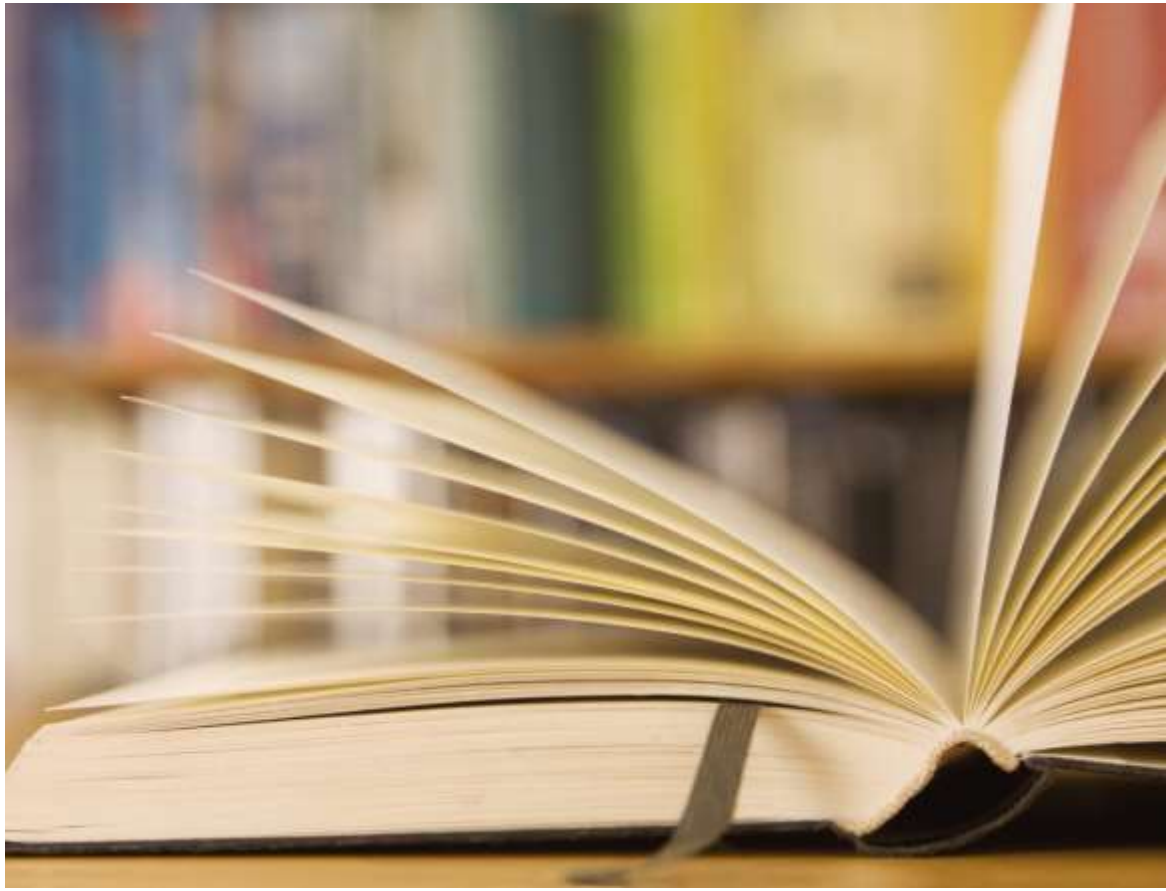
CSCI317 – Database Performance Tuning

Range Partitioning

17 August 2022

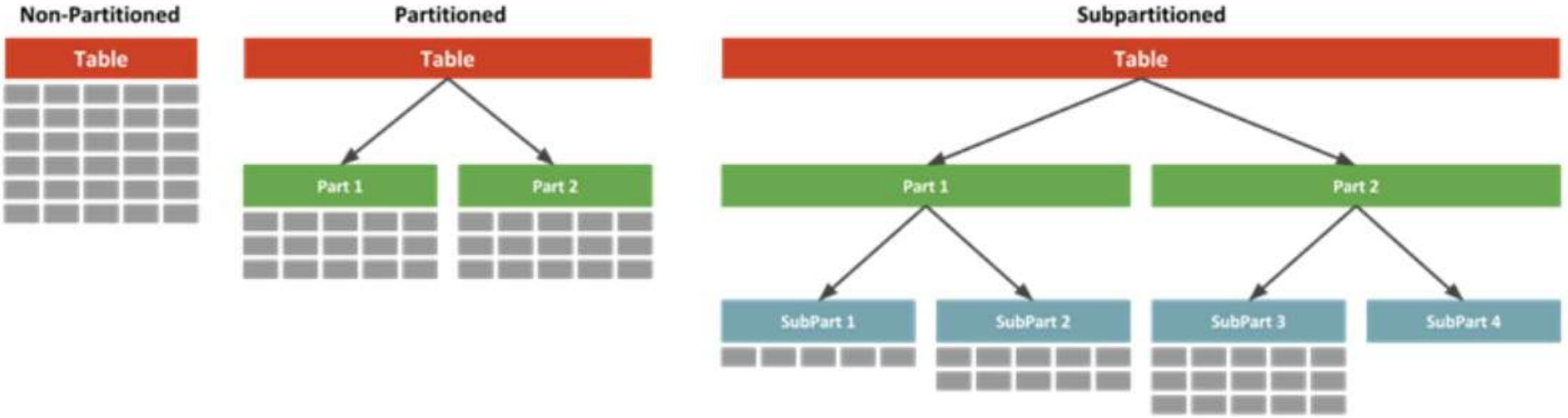


Range Partitioning

Range Partitioning

- As data becoming large, maintenance of large database tables become time and resource intensive. This may affect the database performance, in particular, data access time.
- Partitioning database tables and indexes can benefit the performance and maintenance issues.
- When a table is portioned by range, it is partitioned in such a way that each partition contains rows for which the partitioning expression value lies within a given range.

Range Partitioning



Range Partitioning

- For example, Suppose that we create a table such as the following to hold personnel records for a chain of 20 video stores, numbered 1 through 20:

```
CREATE TABLE employees (
    id              INT NOT NULL    primary key,
    fname           VARCHAR(30),
    lname           VARCHAR(30),
    hired           DATE NOT NULL DEFAULT '1970-01-01',
    separated       DATE NOT NULL DEFAULT '9999-12-31',
    job_code        INT NOT NULL,
    store_id        INT NOT NULL );
```

Source: https://docs.oracle.com/cd/E17952_01/mysql-5.7-en/partitioning-range.html

Range Partitioning

```
CREATE TABLE employees (
    id              INT NOT NULL      primary key,
    fname           VARCHAR(30),
    lname           VARCHAR(30),
    hired           DATE NOT NULL DEFAULT '1970-01-01',
    separated       DATE NOT NULL DEFAULT '9999-12-31',
    job_code  INT NOT NULL,
    store_id        INT NOT NULL )
PARTITION BY RANGE (store_id) (
        PARTITION P0 VALUES LESS THAN (6),
        PARTITION P1 VALUES LESS THAN (11),
        PARTITION P2 VALUES LESS THAN (16),
        PARTITION P3 VALUES LESS THAN (21)
)
```

One way is to partition the table using the store_id column into 4 partitions.

Range Partitioning

```
CREATE TABLE employees (
    id              INT NOT NULL      primary key,
    fname           VARCHAR(30),
    lname           VARCHAR(30),
    hired           DATE NOT NULL DEFAULT '1970-01-01',
    separated       DATE NOT NULL DEFAULT '9999-12-31',
    job_code  INT NOT NULL,
    store_id        INT NOT NULL )
PARTITION BY RANGE (job_code) (
        PARTITION P0 VALUES LESS THAN (100),
        PARTITION P1 VALUES LESS THAN (1000),
        PARTITION P2 VALUES LESS THAN (10000)
)
```

Another way is to partition the table using the job_code column into 3 partitions.

Global Partitioned Index

Global Partitioned Index

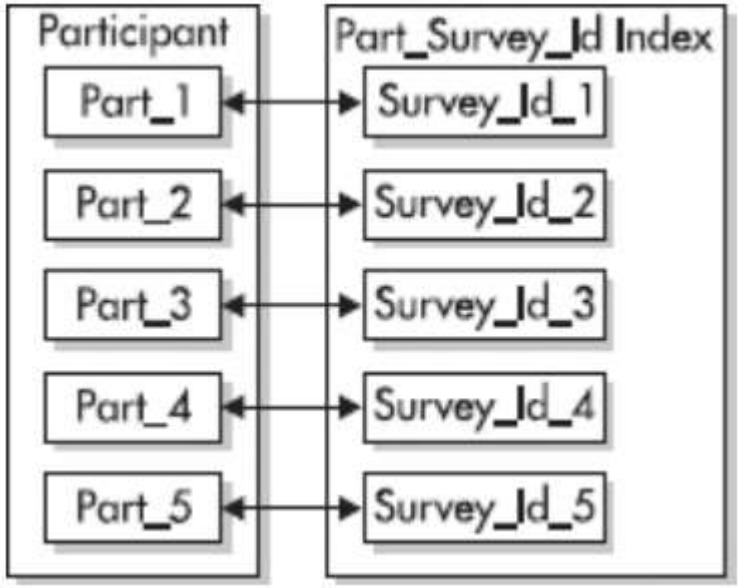
- A partitioned index is simply an index broken into multiple pieces.
- By breaking an index into multiple physical pieces, you are accessing much smaller pieces (faster), and you may separate the pieces onto different disk drives, and hence reducing I/O contention.
- Partitioning can work in several different ways:
  - The tables can be partitioned and the indexes are not partitioned;
  - The table is not portioned, but the index is;
  - Both the table and index are partitioned.

Global Partitioned Index

- Partitioned Index can be:
  - Local
    - Prefixed
    - Non-prefixed
  - Global
    - Prefixed
    - Non-prefixed
- A table can have any number or combination of the different types of indexes built on its columns.

Global Partitioned Index (Local, prefixed)

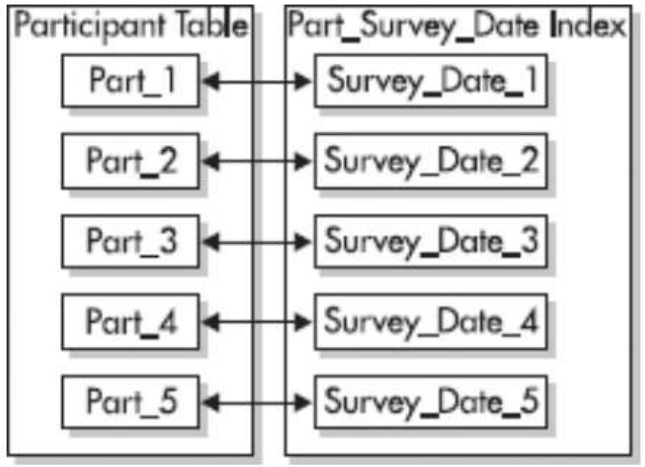
- Local indexes are indexes that are partitioned using the same partition key and same range boundaries as the portioned table.
- Each partition of a local index will only contain keys and ROWIDs from its corresponding table partition.



Prefixed indexes are indexes that contain keys from the partitioning key as the leading edge of the index.
The partitions of the index are equipartitioned, meaning the partitions of the index are created with the same range boundaries as those of the table.

Partitioned, prefixed indexes

Global Partitioned Index (Local, Non-prefixed)

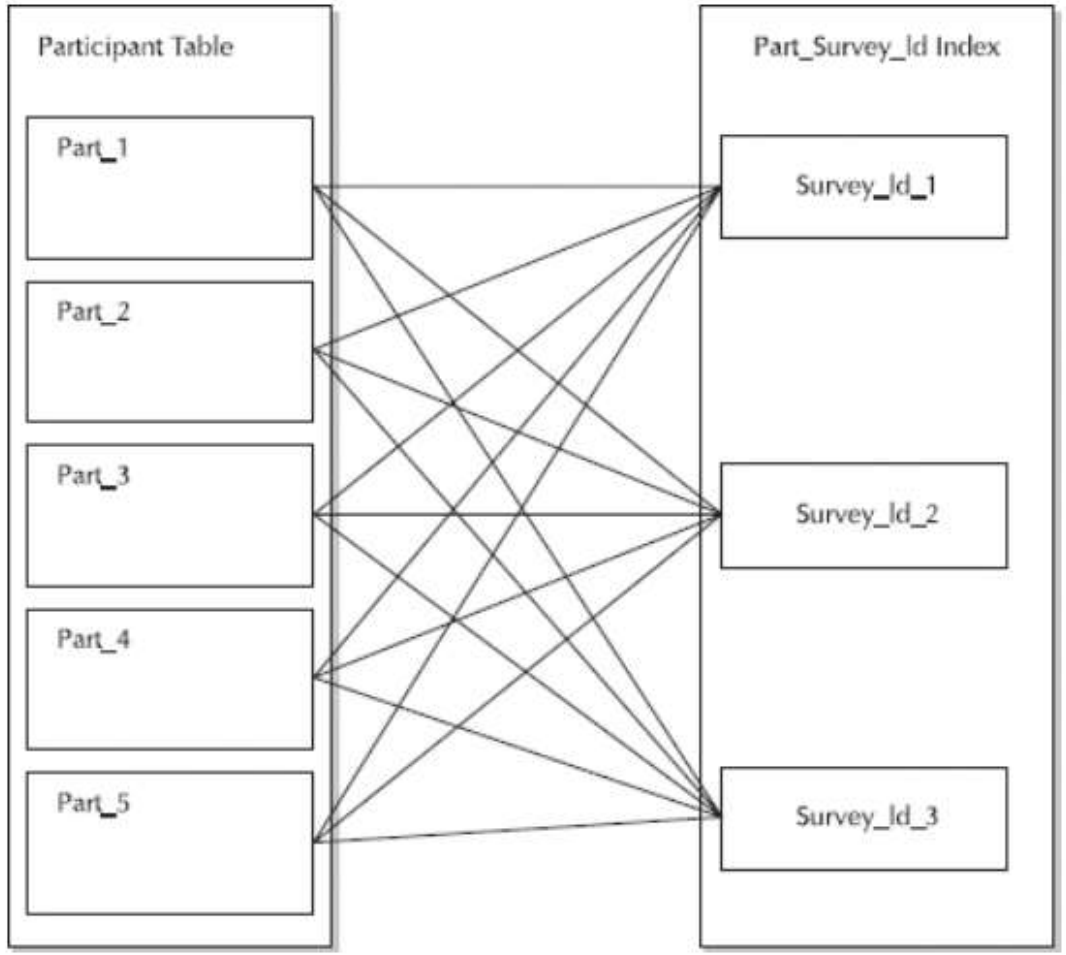


- Non-prefixed indexes are indexes that do not have the leading column of the partitioning key as the leading column of the index.
- A local non-prefixed index can be created on any column in the table, but each partition of the index only contains the keys for the corresponding partition of the table.

Global Partitioned Index (global, prefixed)

- **Global** partitioned indexes contain keys from multiple table partitions in a single index partition.
- The partitioning key of a global partitioned index is different or specifies a different range of values from the partitioned table.
- The creator of the global partitioned index is responsible for defining the ranges and values for the partitioning key.
- Prefixed, normally global prefixed indexes are not equipartitioned with the underlying table. If the index is going to be equipartitioned, it should be created as a local index to allow Oracle to maintain the index and use it to help prune partitions that will not be needed.

Global Partitioned Index (global, prefixed)



Source: https://logicalread.com/oracle-11g-partitioned-indexes-mc02/#.XFKO3rgRXIU

Global Partitioned Index (global, prefixed)

```
CREATE INDEX IDX ON employees (store_id) GLOBAL
PARTITION BY RANGE (store_id)
(PARTITION PIDX0 VALUES LESS THAN (6),
PARTITION PIDX1 VALUES LESS THAN (11),
PARTITION PIDX2 VALUES LESS THAN (16),
PARTITION PIDX3 VALUES LESS THAN (21)
);
```