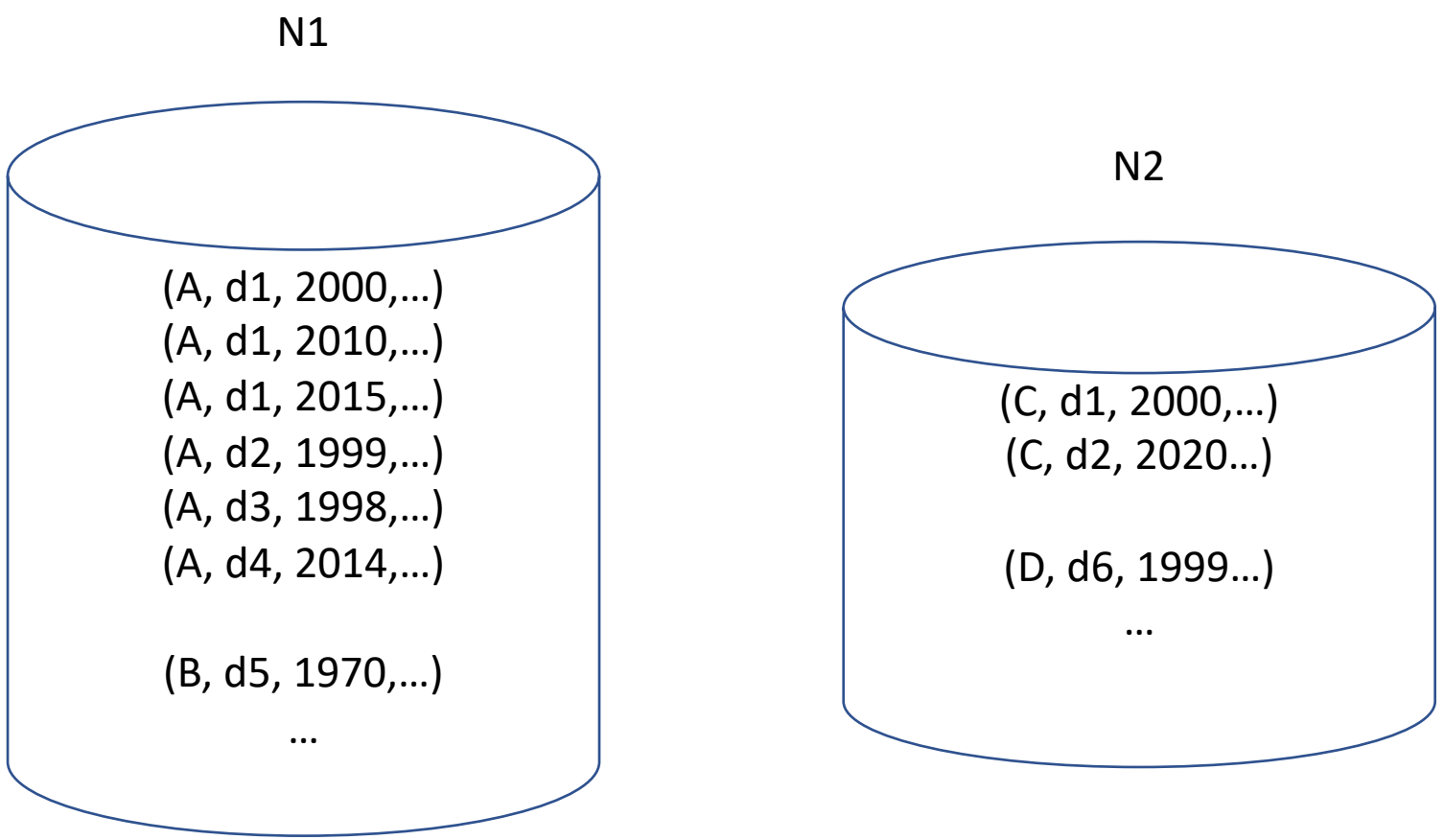
Understanding Cassandra queries

Examples

```
CREATE TABLE movies (
    title text,
    director text,
    year int,
    genre text,
    PRIMARY KEY (title, director, year);
```



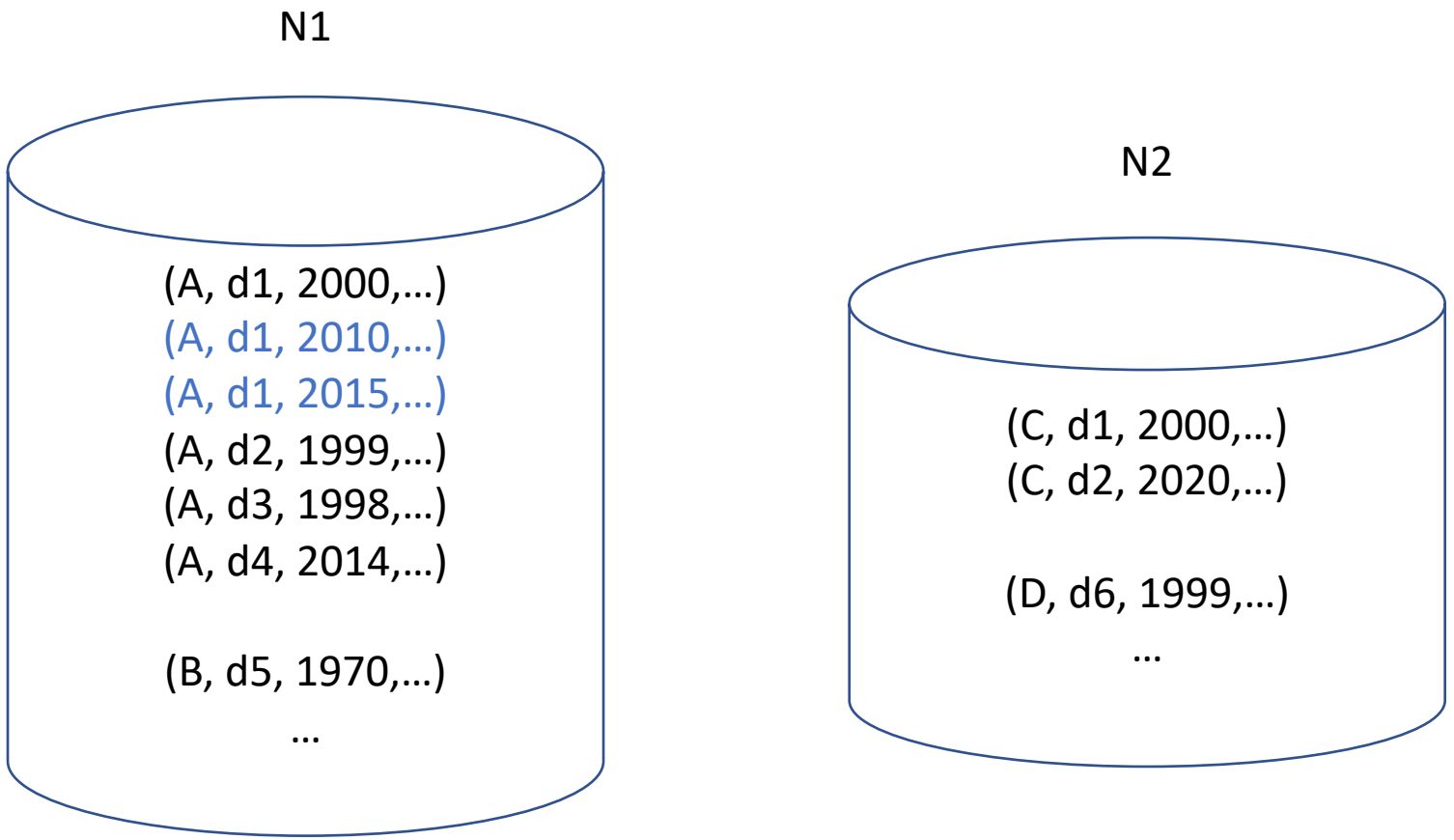
Inside each partition, ordered with respect to the partition key and the clustering column values **(order by title, director, year)**

Queries can be executed if

1. It is possible to locate the nodes storing the rows to be retrieved
2. In each node, the rows to be retrieved are sequentially stored

```
CREATE TABLE movies (
    title text,
    director text,
    year int,
    genre text,
    PRIMARY KEY (title, director, year);
```

```
SELECT*
FROM movies
WHERE  title = 'A' AND
       director= 'd1' AND
       year > 2000
```

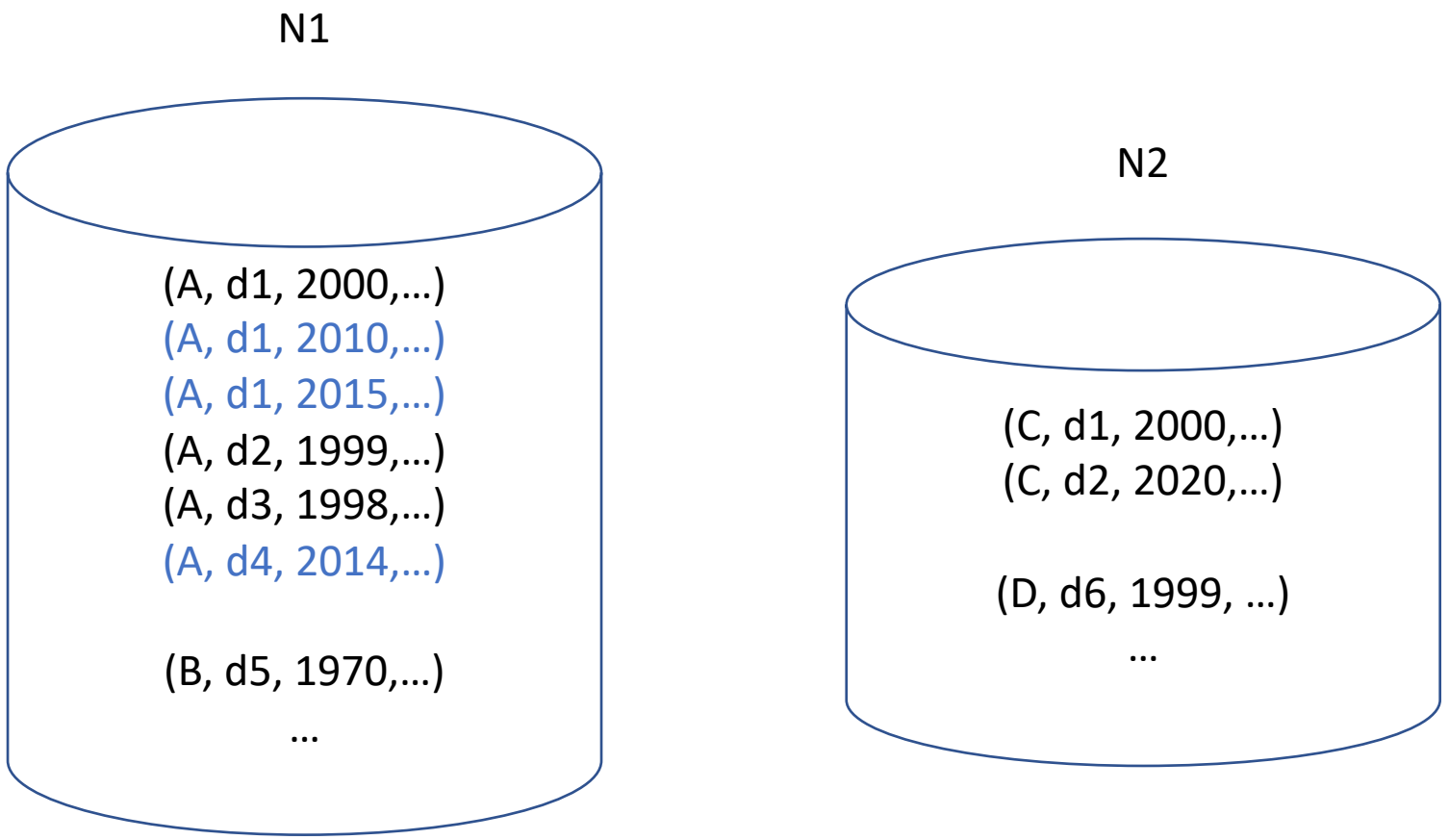


The query is ADMITTED because

1. It is possible to locate the nodes storing the rows to be retrieved (N1)
2. In N1, the rows to be retrieved are sequentially stored (two sequential tuples, in blue)

```
CREATE TABLE movies (
    title text,
    director text,
    year int,
    genre text,
    PRIMARY KEY (title, director, year);
```

```
SELECT*
FROM movies
WHERE  title = 'A' AND
                year > 2000
```



The query is NON ADMITTED because

1. It is possible to locate the nodes storing the rows to be retrieve (N1)
2. But, in N1 the rows to be retrieved are NOT sequentially stored (three tuples, in blue)

Conditions that guarantee the desired behaviour

- The WHERE clause must contain an equality-based selection condition on each attribute of the partition key
  - IN clause is also allowed (a set of nodes is identified)
  - The system is able to identify the nodes contributing to the result

- The WHERE clause can contain selection conditions over any prefix of the clustering column list, as defined in the primary key
  - If more than one clustering column exist, range restrictions are allowed only on the last clustering column being restricted in the WHERE clause
  - On the identified nodes, rows to be returned must be sequentially stored