In this session, I am recreating the scd lecture step to step to understand it more deeply. This covers day2 of dimensional data modelling.

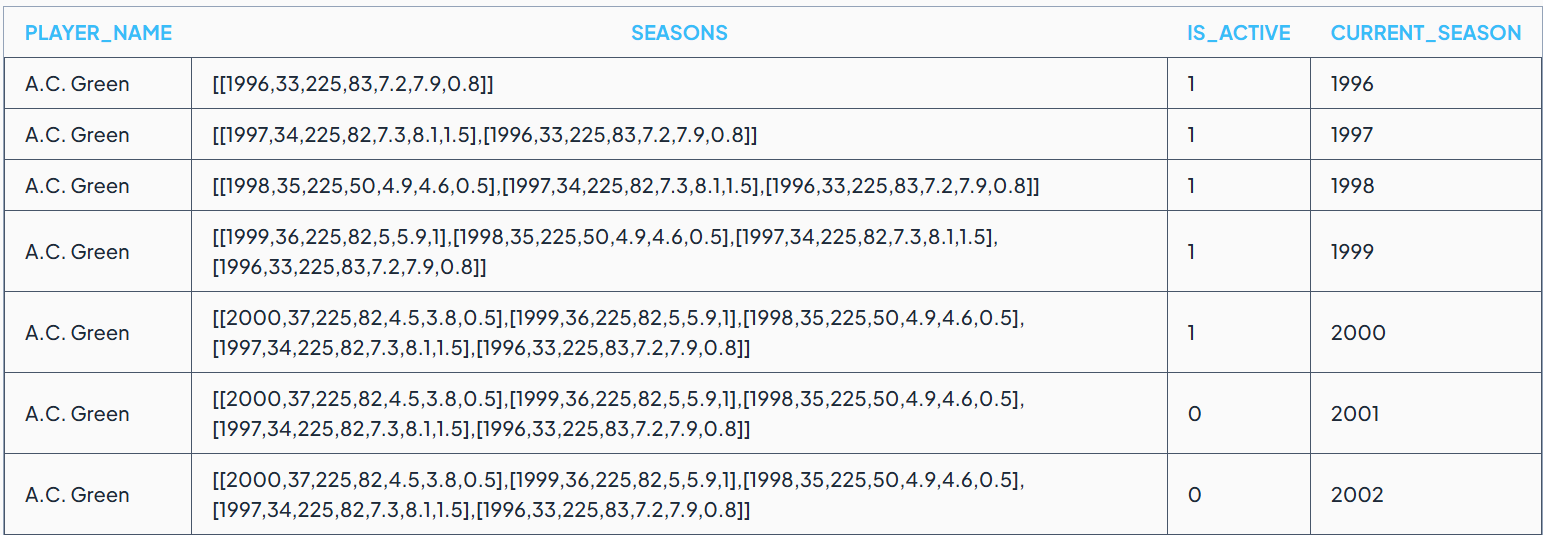
A slowly changing dimension can be modelled in different ways, but we are choosing the best way, SCD type 2 which maintains the history of the dimension from start to finish and hence, idempotent.

Recap from the first session, we created Prithvi.nba\_players, a cumulative table.

SELECT player\_name,seasons,is\_active,current\_season FROM prithvi.nba\_players

order by player\_name, current\_season

LIMIT 50



If you look at a sample entry for ac green, seasons, is\_active are the changing columns. We would like to understand for a particular player, how his dimensions are changing with time, here season.

Seasons column changes every season, changes very often and could be modelled using yearly snapshot. It does not fit the criteria for how often the dimension is changing. It should be slowly changing.

Now, is\_active column for A.C.Green is changing only twice for his entire history of his time as a player. It fits the bill for scd, so, we chose is\_active to scd.

The final result should have 2 rows for A.C. Green, with 1 row’s is\_active column set to 1 and another row’s set to 0. We would need two additional columns: start\_season and end\_season to track the starting and ending point for is\_active column. These two columns help maintain the entire history.

We will start with creating a schema for implementing this scd.

Schema:

CREATE TABLE prithvi.nba\_player\_scd (

player\_name VARCHAR,

is\_active INTEGER,

start\_season INTEGER,

end\_season INTEGER,

current\_season INTEGER

)

WITH (

format ='PARQUET',

partitioning = ARRAY['current\_season']

)

Player\_name – the dimension which is changing over time

Is\_active – the column which changes slowly over time

Start\_season- the column which maintains the starting season when the is\_active is set to current value

End\_season- the column which maintains the ending season when the is\_active is set to current\_value. The is\_active flag changes next season.

Current\_season- the variable which tracks the time as season rolls on.

SCD can be implemented either in 1 load query or incrementally. Lets do the 1 load query first.

1 load Thought process:

First, lets see whether if the player was active last season or not. We use lag function to add an additional column is\_active\_last\_season. If there is a change we now know that its changed from its previous season. This is of utmost importance for implementing this scd on is\_active column.

SELECT player\_name,

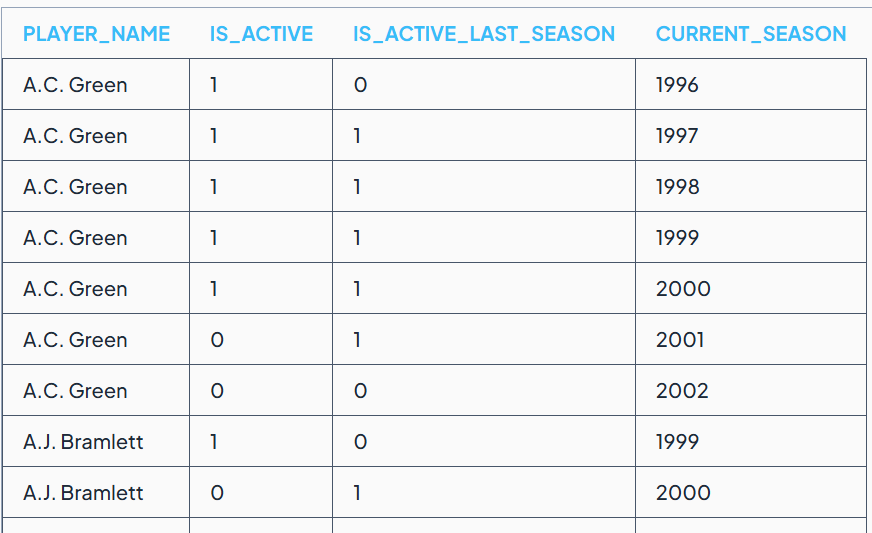
is\_active,

COALESCE(LAG(is\_active,1) OVER(PARTITION BY player\_name ORDER BY current\_season),0) as is\_active\_last\_season,

current\_season

FROM prithvi.nba\_players

The coalesce statement handles the nulls for the very first season.



Add a new column called did\_change to track the difference between the present and previous season.

WITH lagged AS(

SELECT player\_name,

is\_active,

COALESCE(LAG(is\_active,1) OVER(PARTITION BY player\_name ORDER BY current\_season),0) as is\_active\_last\_season,

current\_season

FROM prithvi.nba\_players

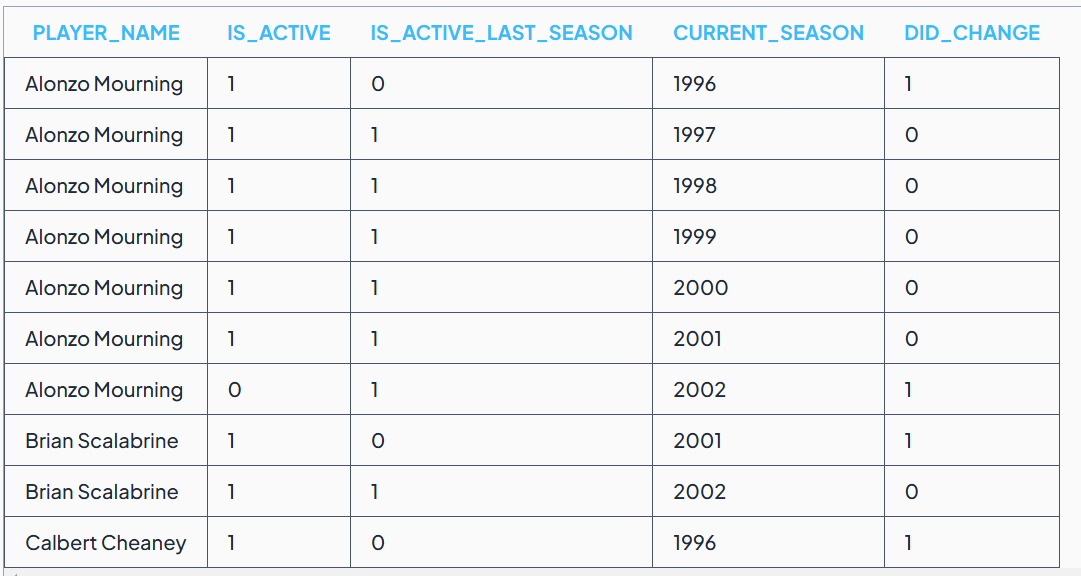
)

SELECT \*,

CASE WHEN

is\_active <> is\_active\_last\_season THEN 1 ELSE 0 END AS did\_change

FROM lagged limit 50



We now count the lifetime ‘did\_change’ for a particular player. We can do this by running sum over the did\_change column as follows:

WITH lagged AS(

SELECT player\_name,

is\_active,

COALESCE(LAG(is\_active,1) OVER(PARTITION BY player\_name ORDER BY current\_season),0) as is\_active\_last\_season,

current\_season

FROM prithvi.nba\_players

)

SELECT \*,

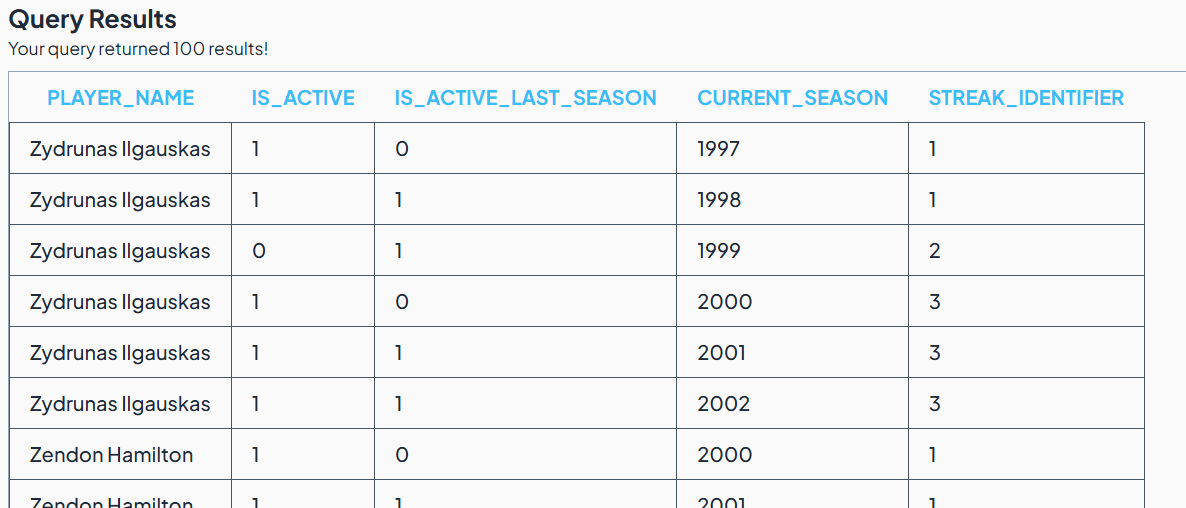
SUM(CASE WHEN

is\_active <> is\_active\_last\_season THEN 1 ELSE 0 END) OVER(PARTITION BY player\_name ORDER BY current\_season)AS streak\_identifier

FROM lagged

ORDER BY player\_name desc,current\_season

limit 100



The streak\_identifier column now accurately depicts the players behaviour vs time. If there’s a change it increments, if not, it stay the same.

The pair of values: (player\_name,streak\_identifier) are unique for each streak. In the above example:

* Zydrunas IIgauskas + 1 (streak\_identifier) indicates his streak of playing for the seasons of 1997-1998
* Zydrunas IIgauskas + 2 (streak\_identifier) indicates his inactive strek for year 1999
* Zydrunas IIgauskas + 3 (streak\_identifier) indicates his active streak from 2000 to 2002

We can write a query using group by player\_name,streak\_identifier to get the final scd table

WITH lagged AS(

SELECT player\_name,

is\_active,

COALESCE(LAG(is\_active,1) OVER(PARTITION BY player\_name ORDER BY current\_season),0) as is\_active\_last\_season,

current\_season

FROM prithvi.nba\_players

),

streaked AS(

SELECT \*,

SUM(CASE WHEN

is\_active <> is\_active\_last\_season THEN 1 ELSE 0 END) OVER(PARTITION BY player\_name ORDER BY current\_season)AS streak\_identifier

FROM lagged

)

SELECT player\_name,

MAX(is\_active) AS is\_active,

MIN(current\_season) as start\_season,

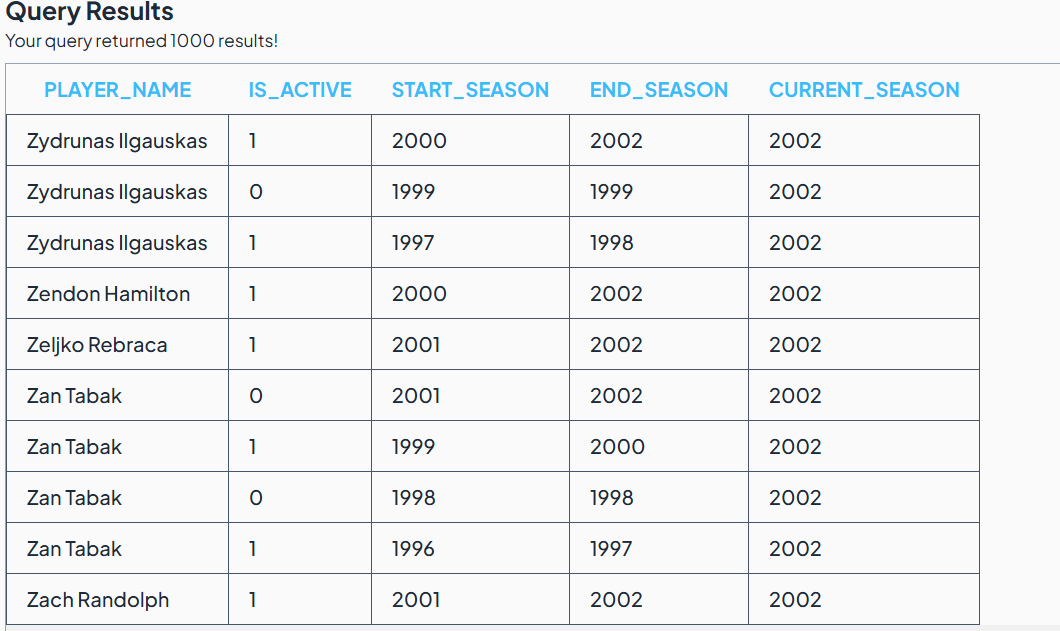
MAX(current\_season) as end\_season,

'2002' as current\_season

FROM streaked

GROUP BY player\_name,streak\_identifier

order by player\_name desc



Time to load the data into our earlier schema :Prithvi.nba\_player\_scd

Insert into Prithvi.nba\_player\_scd

WITH lagged AS(

SELECT player\_name,

is\_active,

COALESCE(LAG(is\_active,1) OVER(PARTITION BY player\_name ORDER BY current\_season),0) as is\_active\_last\_season,

current\_season

FROM prithvi.nba\_players

),

streaked AS(

SELECT \*,

SUM(CASE WHEN

is\_active <> is\_active\_last\_season THEN 1 ELSE 0 END) OVER(PARTITION BY player\_name ORDER BY current\_season)AS streak\_identifier

FROM lagged

)

SELECT player\_name,

MAX(is\_active) AS is\_active,

MIN(current\_season) as start\_season,

MAX(current\_season) as end\_season,

'2002' as current\_season

FROM streaked

GROUP BY player\_name,streak\_identifier

This is the 1-time load for the scd table, now let’s look at the incremental approach.

Incremental Load thought process