REPRESENTATION OF HIERARCHICAL DATA

PROBLEM STATEMENT

- Many real-world applications require structuring and managing hierarchical data.
- Hierarchical relationships exist in organizational structures, file systems, taxonomies, and more.
- Traditional relational databases struggle with efficiently querying and managing hierarchical data.
- This presentation explores two primary methods of representing hierarchical data: Adjacency List and Nested Set Model.

WHAT IS HIERARCHICAL DATA ?

- Definition: Data arranged in a tree-like structure where each node has a parent-child relationship.
- Examples:
  - Organizational charts (CEO → Managers → Employees)
  - File directory structures (Root → Folder → Subfolder → File)
  - Product categories in e-commerce platform.

METHOD 1 – ADJACENCY LIST MODEL

- **Concept:** Each node in the hierarchy contains a reference to its immediate parent node, allowing you to navigate up the hierarchy by following these parent links.

- **Example**: In an employee database, each employee record would have a "managerID" field that points to the ID of their direct supervisor.

- Schema Example:

ID	Name	Parent_ID
1	CEO	NULL
2	Manager	1
3	Employee	2

- Pros:
  Simple to implement, efficient for adding or deleting nodes.

- Cons:
  Can be less efficient for querying large subtrees or determining the depth of a node within the hierarchy.

METHOD 2 – NESTED SET MODEL

- **Concept:** Uses left and right values to represent hierarchy levels. Each node in the hierarchy is assigned a left and right value, which helps in efficiently querying hierarchical relationships without recursion.
- **Example**: In a file system, each folder and file is assigned a left and right value to define its hierarchical position. For instance, a 'Documents' folder might have Left=1 and Right=6, while a subfolder 'Projects' inside it could have Left=2 and Right=5.
- **Pros**: Fast hierarchical queries (no recursion needed) Good for read-heavy applications
- **Cons:** Complex insertions and deletions. Requires recalculating left/right values on modification

COMPARISION

Feature	Adjacency List	Nested Set Model
Query Complexity	Requires recursive queries	Faster queries
Insert/Delete	Easy	Complex, requires updates
Best For	Dynamic data with frequent updates	Static data with frequent reads

REAL-WORLD USE CASES

- Adjacency List:
  - Social media connections (friends/followers)
  - Employee reporting structures
  - Menu structures

- Nested Set Model:
  - Product categories in e-commerce
  - Hierarchical tagging systems
  - Multi-level marketing systems

CONCLUSION

- Hierarchical data can be represented using multiple models depending on the use case.
- Adjacency List is easier for updates but requires recursive queries.
- Nested Set Model is efficient for querying but complex to modify.
- Choosing the right model depends on read vs. write optimization.

THANK YOU!!