Query Optimization Techniques

@data insights

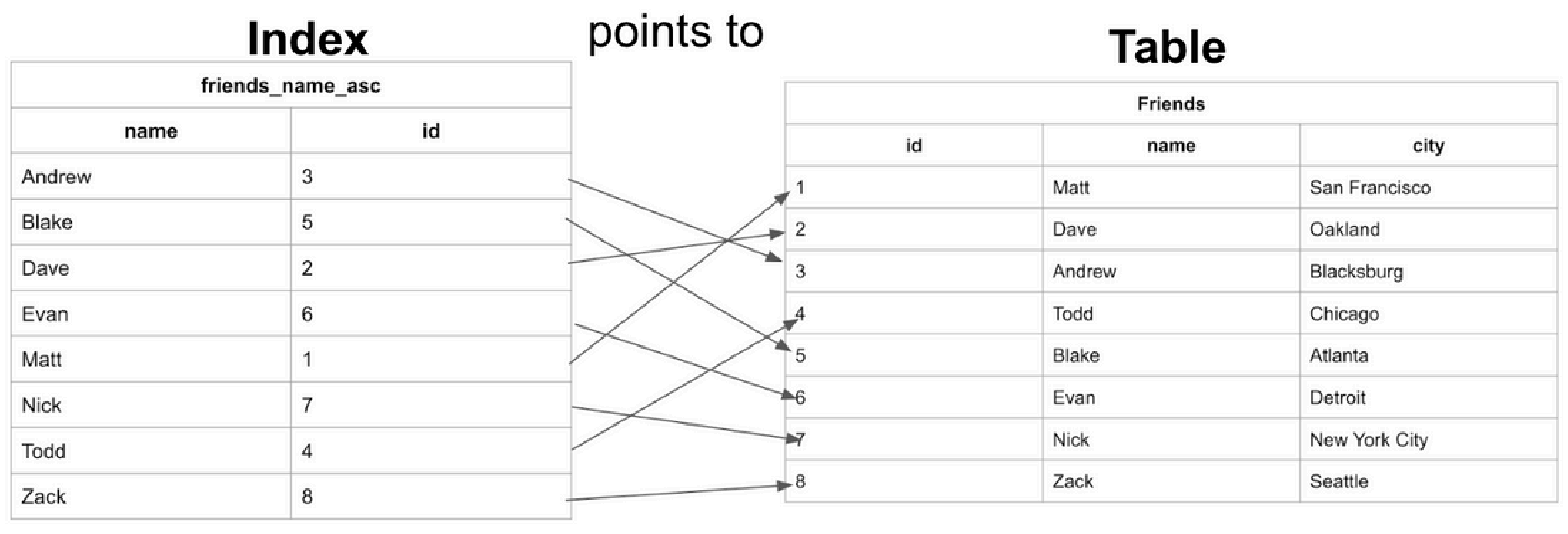
What is Query Optimization?

- Process of enhancing database queries for faster execution.
- Reduces resource usage (CPU, memory, I/O).
- Improves user experience and system scalability.

$$\begin{tikzcd}
\text{Query} \arrow[d] \\
\text{Optimization} \arrow[d] \\
\text{Execution}
\end{tikzcd}$$

Common Query Optimization Techniques

- Indexing


Index
points to
Table

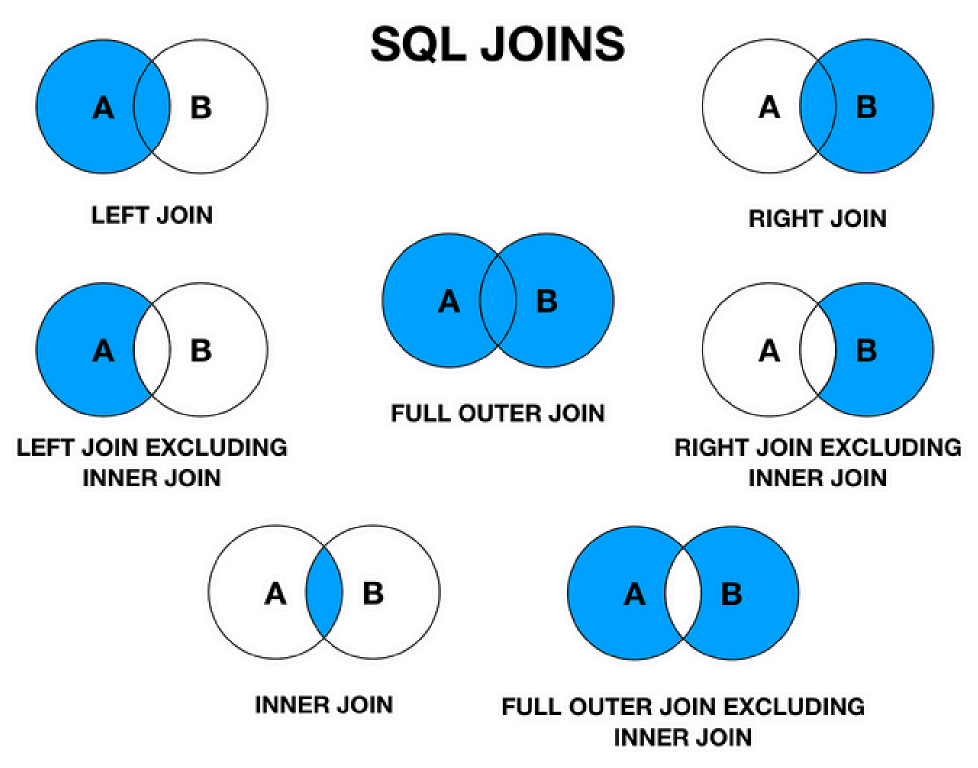
friends_name_asc

name	id
Andrew	3
Blake	5
Dave	2
Evan	6
Matt	1
Nick	7
Todd	4
Zack	8

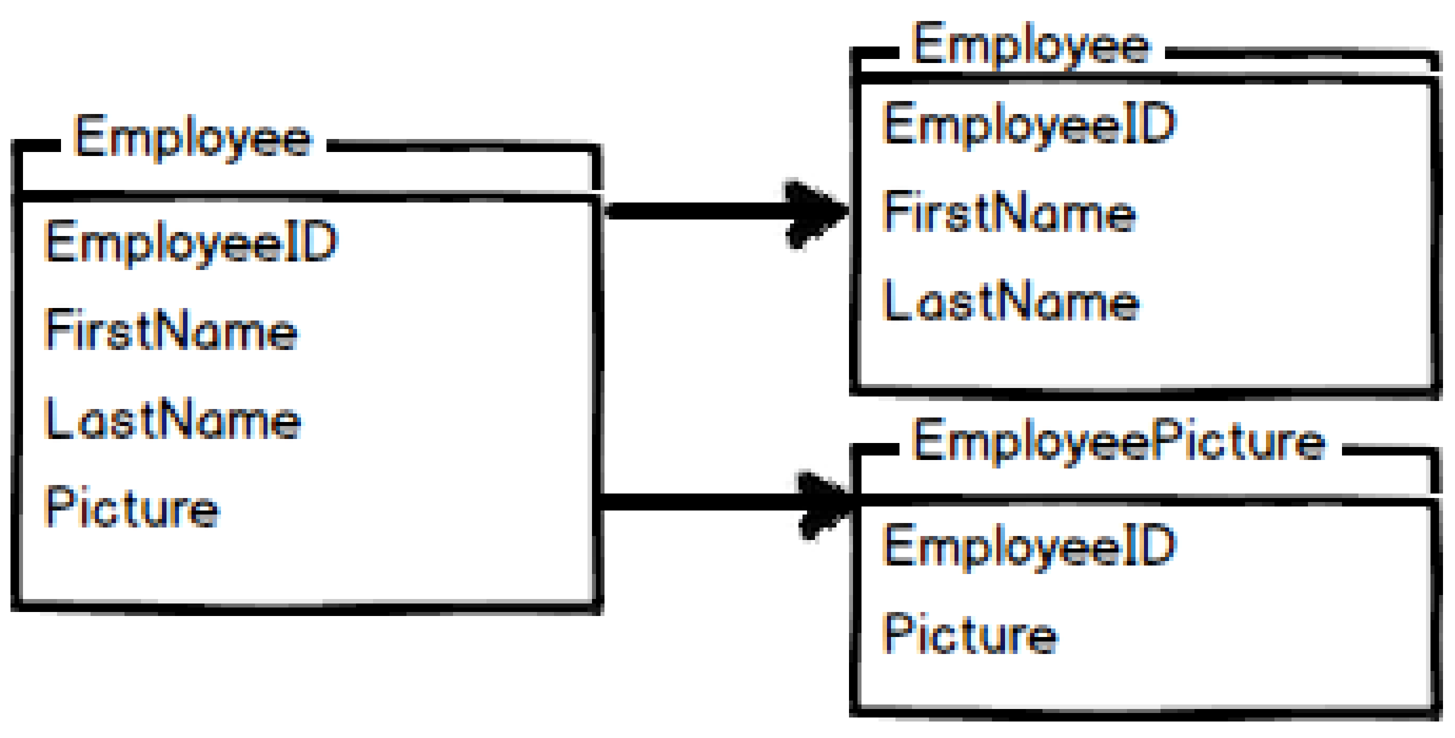
Friends

id	name	city
1	Matt	San Francisco
2	Dave	Oakland
3	Andrew	Blacksburg
4	Todd	Chicago
5	Blake	Atlanta
6	Evan	Detroit
7	Nick	New York City
8	Zack	Seattle

- Query Refactoring
- Avoiding SELECT *
- Using Joins Instead of Subqueries


SQL JOINS
A
B
LEFT JOIN
A
B
RIGHT JOIN
A
B
LEFT JOIN EXCLUDING INNER JOIN
A
B
FULL OUTER JOIN
A
B
RIGHT JOIN EXCLUDING INNER JOIN
A
B
INNER JOIN
A
B
FULL OUTER JOIN EXCLUDING INNER JOIN

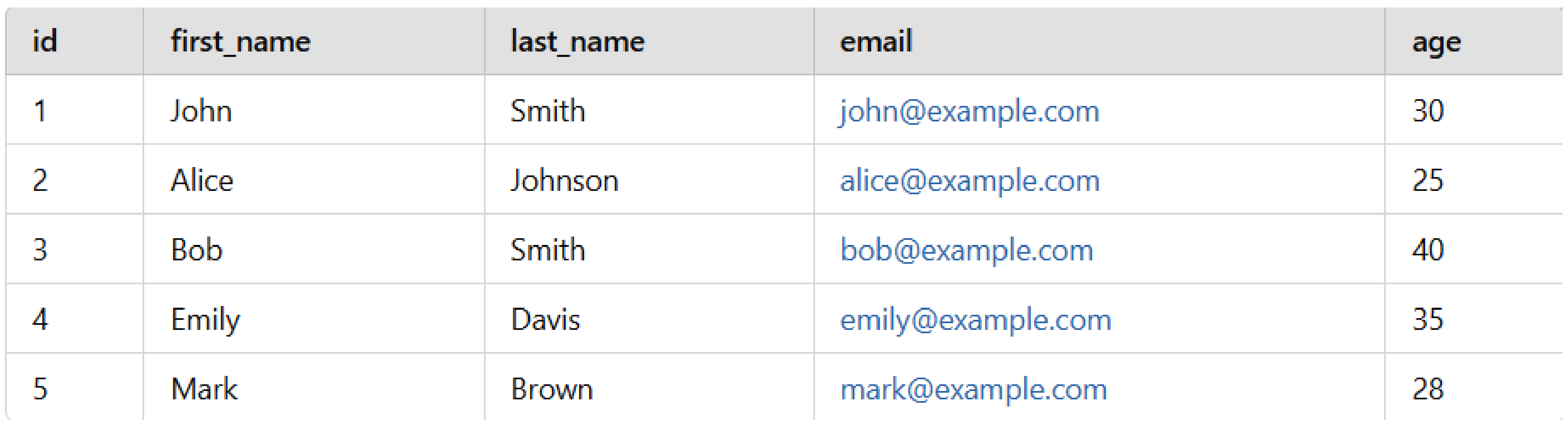
- Partitioning Large Tables



Before/After Example 1 - Indexing

Before:

- **Query: SELECT * FROM users WHERE last_name = 'Smith';**
- **Execution Plan: Full Table Scan.**
- **Performance: Slow (e.g., 500ms).**

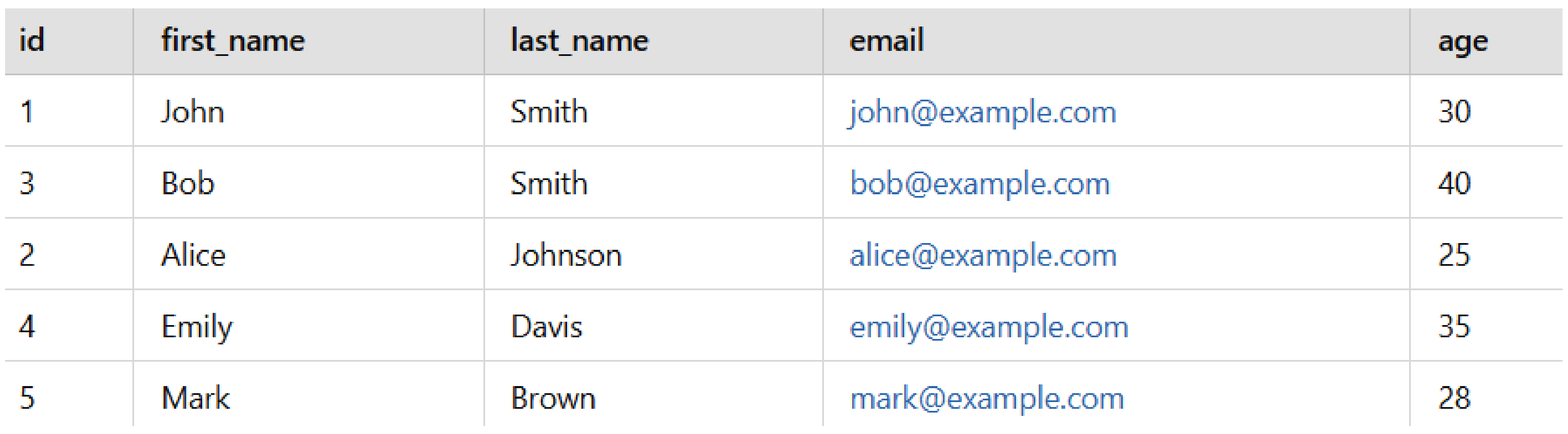


id	first_name	last_name	email	age
1	John	Smith	john@example.com	30
2	Alice	Johnson	alice@example.com	25
3	Bob	Smith	bob@example.com	40
4	Emily	Davis	emily@example.com	35
5	Mark	Brown	mark@example.com	28

Since there is no index on last_name, the database performs a Full Table Scan—checking each row one by one.

After:

- **Query: CREATE INDEX idx_last_name ON users(last_name);**
- **Execution Plan: Index Seek.**
- **Performance: Fast (e.g., 50ms).**



id	first_name	last_name	email	age
1	John	Smith	john@example.com	30
3	Bob	Smith	bob@example.com	40
2	Alice	Johnson	alice@example.com	25
4	Emily	Davis	emily@example.com	35
5	Mark	Brown	mark@example.com	28

Since an index exists on last_name, the database can directly find matching rows.

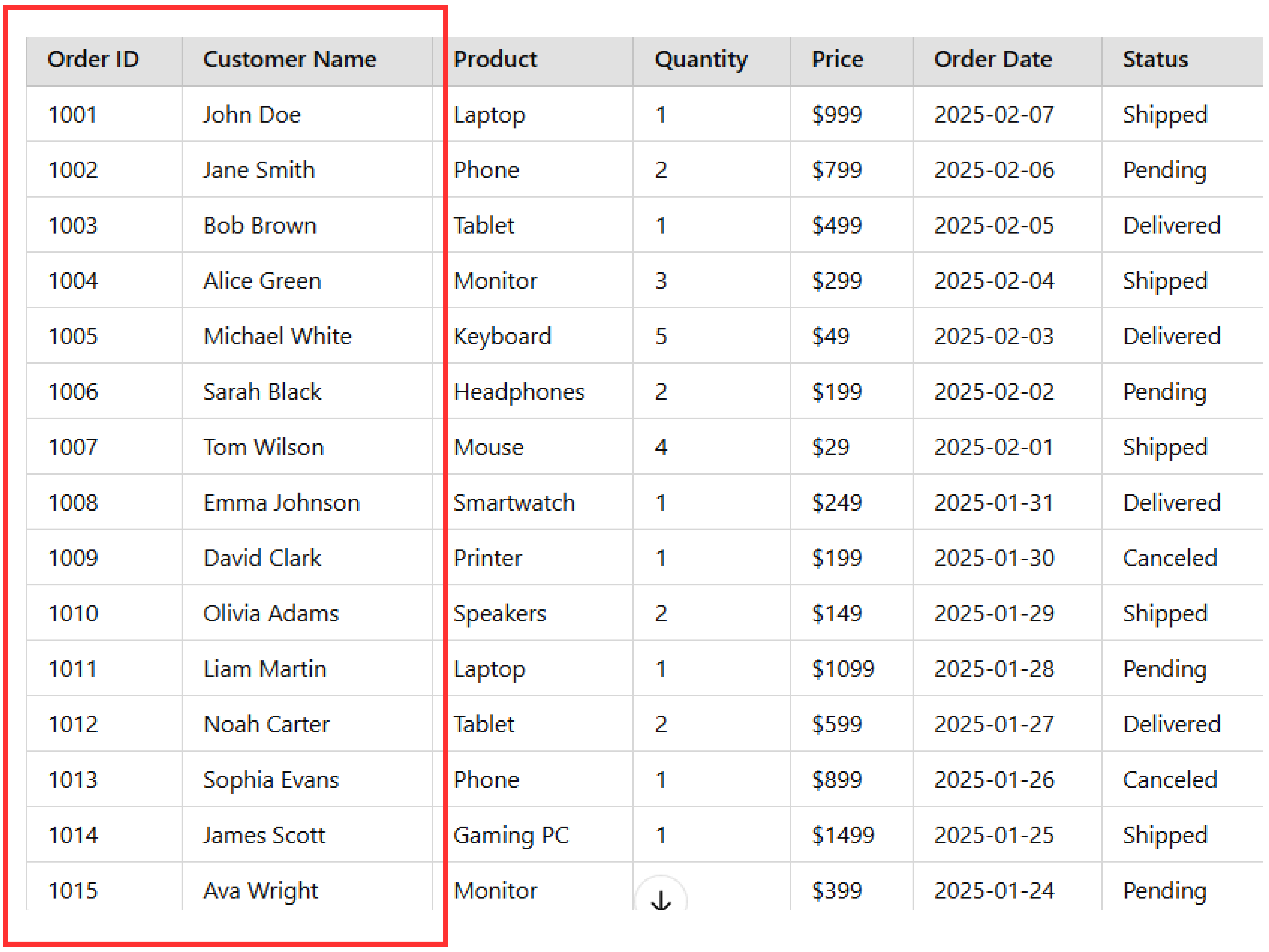
Before/After Example 2 - Avoiding SELECT

Before:

- **Query: SELECT * FROM orders;**
- **Execution Plan: Scans all columns.**
- **Performance: Slow (e.g., 300ms).**

After:

- **Query: SELECT “Order ID”, “Customer Name” FROM orders;**
- **Execution Plan: Scans only required columns.**
- **Performance: Fast (e.g., 100ms).**

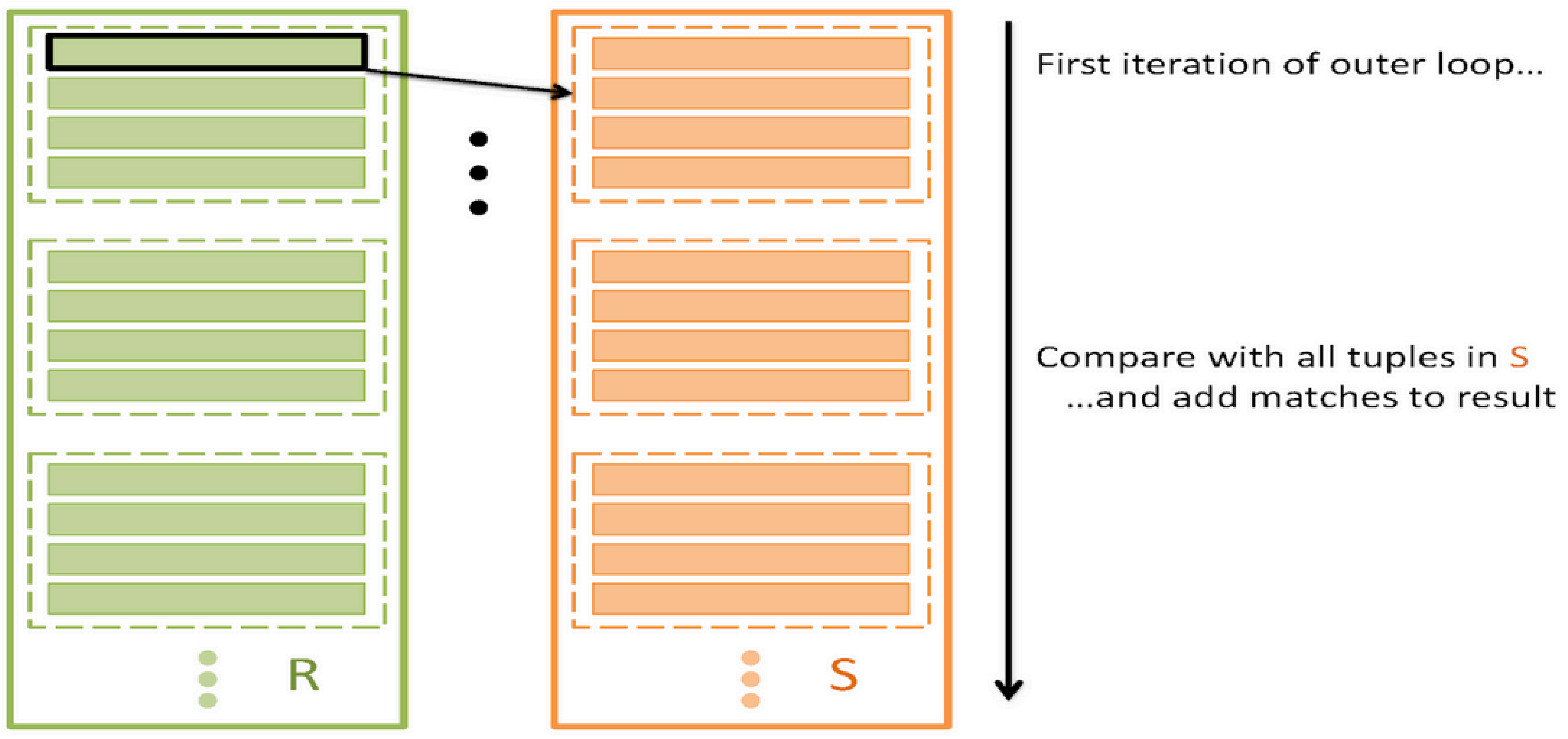


Order ID	Customer Name	Product	Quantity	Price	Order Date	Status
1001	John Doe	Laptop	1	$999	2025-02-07	Shipped
1002	Jane Smith	Phone	2	$799	2025-02-06	Pending
1003	Bob Brown	Tablet	1	$499	2025-02-05	Delivered
1004	Alice Green	Monitor	3	$299	2025-02-04	Shipped
1005	Michael White	Keyboard	5	$49	2025-02-03	Delivered
1006	Sarah Black	Headphones	2	$199	2025-02-02	Pending
1007	Tom Wilson	Mouse	4	$29	2025-02-01	Shipped
1008	Emma Johnson	Smartwatch	1	$249	2025-01-31	Delivered
1009	David Clark	Printer	1	$199	2025-01-30	Canceled
1010	Olivia Adams	Speakers	2	$149	2025-01-29	Shipped
1011	Liam Martin	Laptop	1	$1099	2025-01-28	Pending
1012	Noah Carter	Tablet	2	$599	2025-01-27	Delivered
1013	Sophia Evans	Phone	1	$899	2025-01-26	Canceled
1014	James Scott	Gaming PC	1	$1499	2025-01-25	Shipped
1015	Ava Wright	Monitor		$399	2025-01-24	Pending

Before/After Example 3 – Using Joins Instead of Subqueries

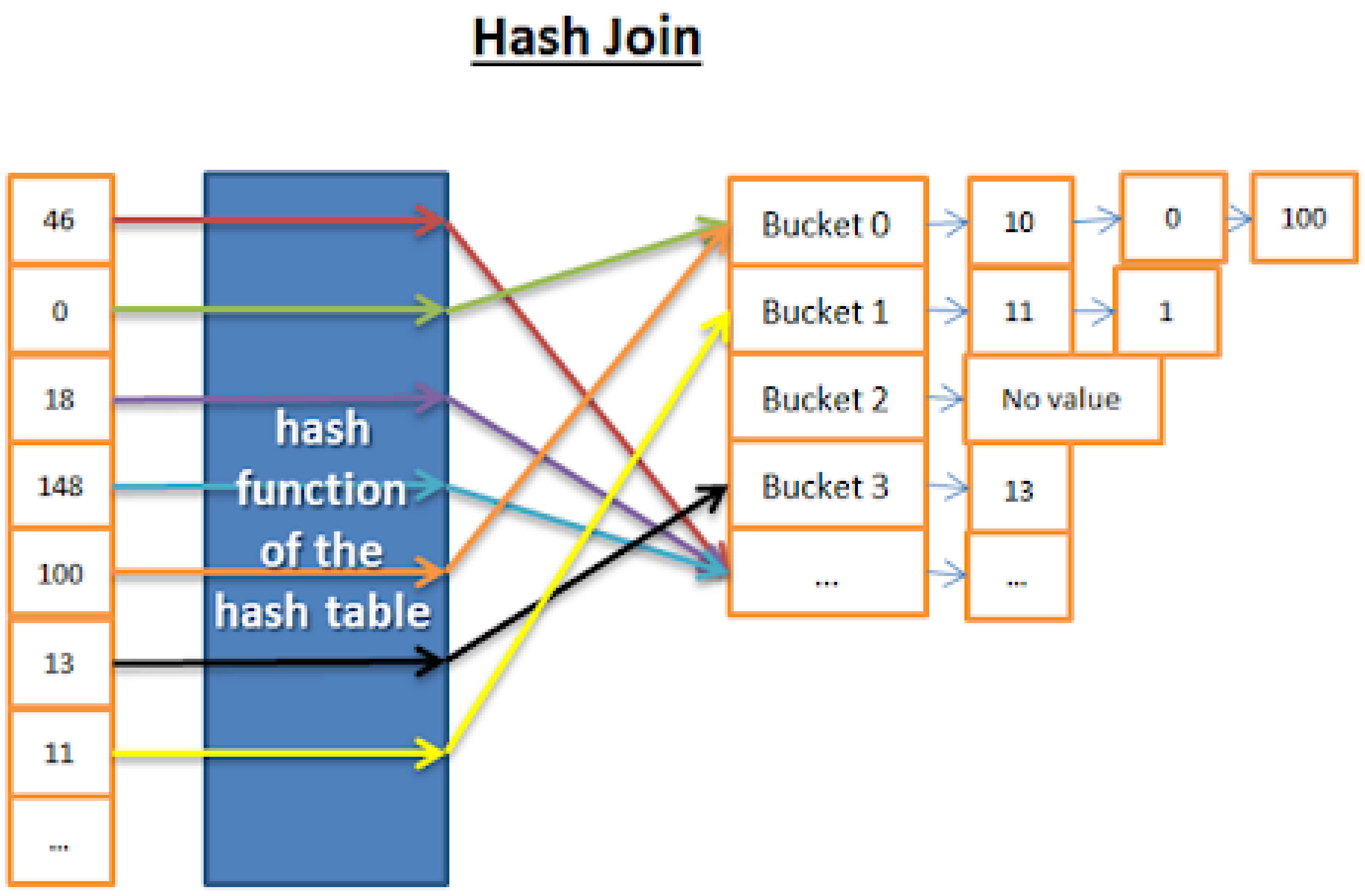
Before:

- Query: SELECT * FROM users WHERE user_id IN (SELECT user_ic FROM orders);
- Execution Plan: Nested Loop.
- Performance: Slow (e.g., 400ms).


First iteration of outer loop...
Compare with all tuples in S
...and add matches to result
R
S

After:

- Query: SELECT u.* FROM users u JOIN orders o ON u.user_id = o.user_id;
- Execution Plan: Hash Join.
- Performance: Fast (e.g., 150ms).



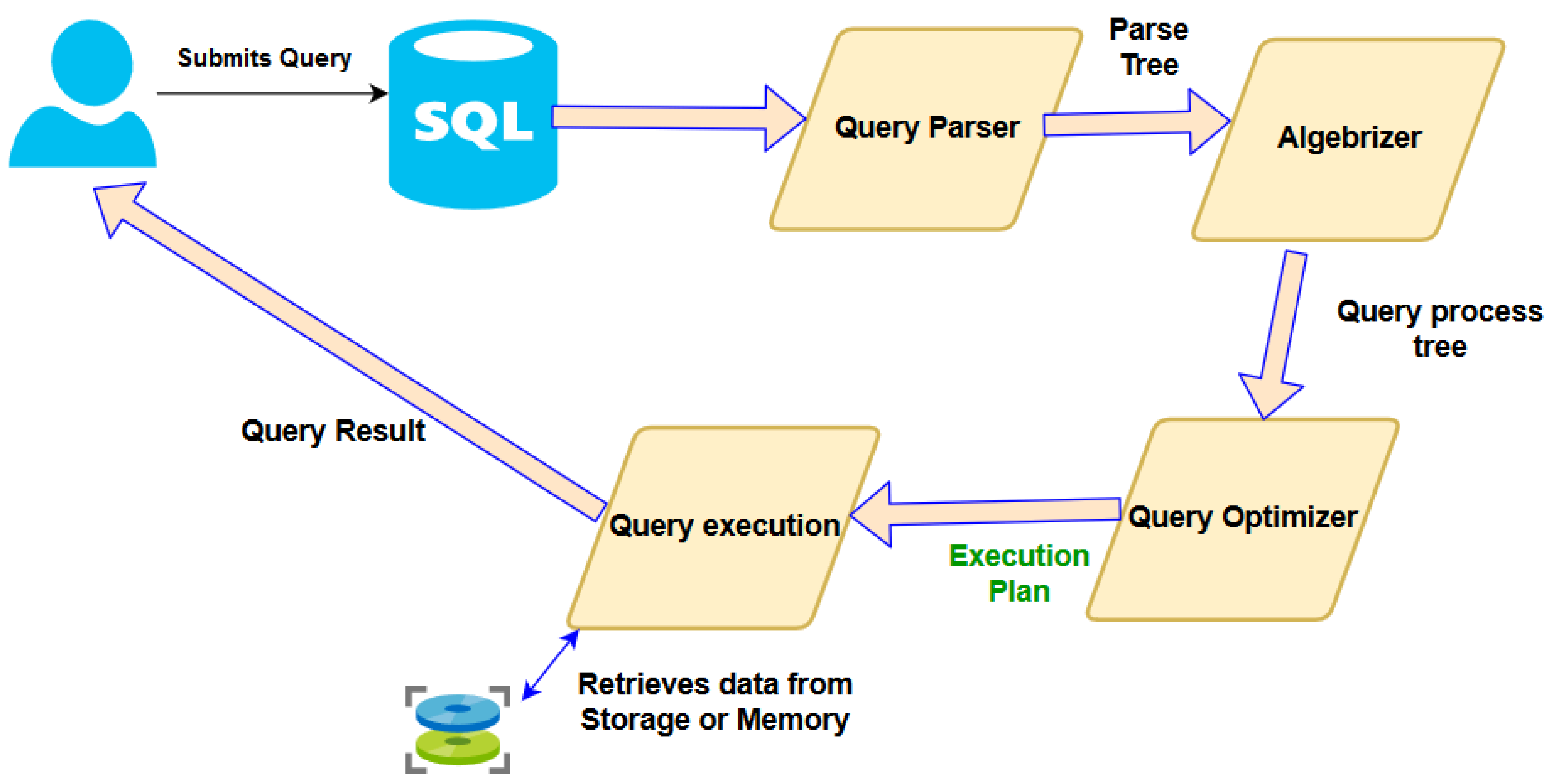
Execution Plans Explained

An execution plan is a roadmap of how a database query is executed, showing the steps the optimizer chooses to retrieve data efficiently.

How to Read It:
Operations:

- **Scan (Full table/index scan) → Reads all rows, less efficient.**
- **Seek (Index seek) → Quickly finds specific rows using an index.**
- **Join (Nested Loop, Hash, Merge) → Combines data from multiple tables.**

Cost: Represents the estimated resource usage; lower is better.
Rows: Estimated number of rows processed at each step.



Conclusion

Key Points:

- **Query Optimization is Critical:** Optimized queries reduce execution time and resource consumption.

- **Small Changes, Big Impact:** Indexing, rewriting queries, or restructuring joins can dramatically improve performance.

- **Measure & Test:** Always analyze execution plans and benchmark performance before and after changes.

Like

Comment

Share

Data INSIGHTS