DIFFERENT WAYS OF REPRESENTING HISTORY

An overview of techniques to track historical data in databases

CRYSTAL FERNANDES

DEFINING THE ISSUE

- **Maintaining historical data** is critical for accurate reporting, compliance, and business analysis.
- Different **data modeling techniques** offer various ways to represent history, each with trade-offs in complexity, storage, and performance.
- Organizations need to **choose the right method** to track data changes while ensuring data integrity and efficient querying.

Key Questions:

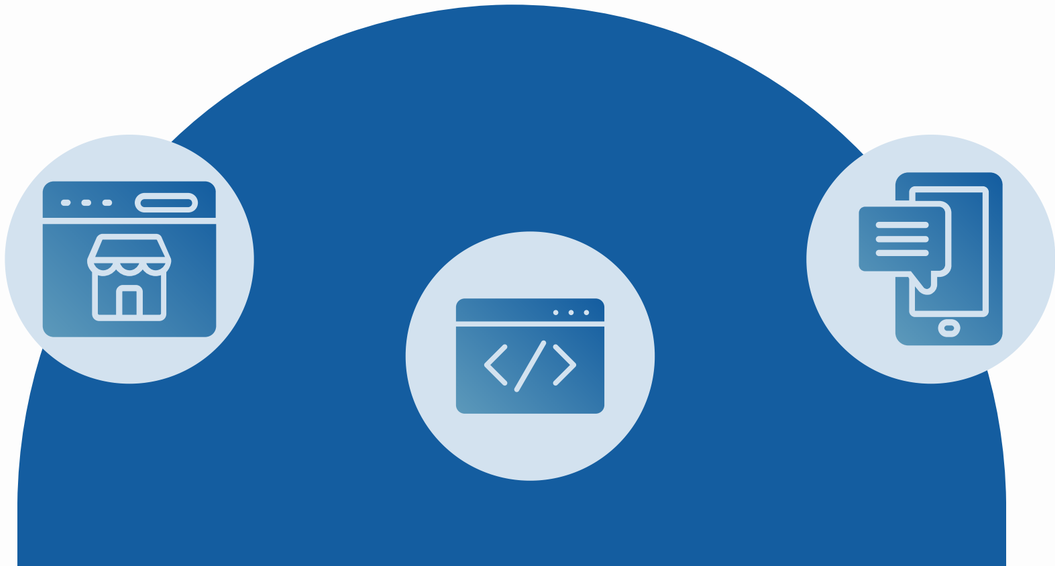
- How can we track changes in data over time effectively?
- What are the best techniques?
- How do we ensure historical accuracy without compromising performance?

What Does Representing History Mean?

- **Tracking Changes Over Time:** Capturing how data evolves or changes over time, not just storing current data.
- **Preserving Past Versions:** Storing past versions of data to retain a historical record of all updates or modifications.
- **Supporting Business Decisions:** Enabling analysis of data trends and behavior over time to make informed business decisions.
- **Ensuring Accountability:** Keeping records of who made changes to data, when, and why, often for auditing purposes.

Examples

1. Sales Data
2. Customer Data
3. Financial Records



Ways To Represent History In Data Modelling

01 Audit Tables

02 Slowly Changing Dimensions (SCD Types 1–6)

03 Temporal Tables

04 Snapshot Tables

AUDIT TABLES

- Separate tables
- Track and log every change (insert, update, delete) in the main table.
- Metadata about the changes (who, what, when).

STRUCTURE OF AN AUDIT TABLE:

- Audit ID: Unique identifier for each log entry.
- Table Name: The name of the table where the change occurred.
- Operation Type: The type of operation (INSERT, UPDATE, DELETE).
- Old Values: The previous values of the affected fields before the change.
- New Values: The new values of the affected fields after the change.
- Timestamp: The date and time when the change was made.
- User ID: The ID of the user who made the change.

Main Table (`orders`)

order_id	customer_id	status	updated_at
101	1001	Shipped	2025-01-05 10:00
102	1002	Delivered	2025-01-06 12:30

Audit Table (`orders_audit`)

audit_id	order_id	status	operation	modified_at	modified_by
1	101	Pending	INSERT	2025-01-01 09:00	admin
2	101	Processing	UPDATE	2025-01-02 14:30	admin
3	101	Shipped	UPDATE	2025-01-05 10:00	system
4	102	Pending	INSERT	2025-01-03 11:00	admin
5	102	Delivered	UPDATE	2025-01-06 12:30	system

SLOWLY CHANGING DIMENSIONS (SCD)

- Tracking History in Dimensions
- Capturing and storing changes to attributes in dimensional data (like customer, product, or location data) over time.

TYPES OF SCD:

SCD Type 1: Overwrite Data, No History

- Updates existing data with new values, overwriting the old record.
- No historical data is kept.

SCD Type 2: Full History with valid_from and valid_to

- Records each change as a new row, tracking the validity period of each record.
- Full history of changes is maintained.

SCD Type 3: Limited History with Extra Columns

- Keeps only the current and previous values in the same record.
- Typically uses extra columns (e.g., current_city, previous_city) to store the historical change.

SCD TYPE 1

Example: Customer Address Change

customer_id	name	city	updated_at
1	John Doe	New York	2025-01-01

Update Scenario:

- John moves from New York to Los Angeles on **2025-02-01**.
- The row is updated, and the old city (New York) is **lost**.

customer_id	name	city	updated_at
1	John Doe	Los Angeles	2025-02-01

SCD TYPE 2

Example: Customer Address History

customer_id	name	city	valid_from	valid_to
1	John Doe	New York	2024-01-01	2025-01-31
1	John Doe	Los Angeles	2025-02-01	9999-12-31

SCD TYPE 3

Example: Customer Address Change

customer_id	name	current_city	previous_city	updated_at
1	John Doe	Los Angeles	New York	2025-02-01

- **current_city** reflects John's current address (Los Angeles).
- **previous_city** stores his last known address (New York).
- **No further historical changes** are retained beyond one previous value.

TEMPORAL TABLES

- Temporal tables use valid_from and valid_to timestamps to track the validity period of each row.
- They allow you to query data as it existed at a specific point in time, preserving historical states.
- System-versioned tables that track data changes with timestamps in the same table

employee_id	name	role	salary	valid_from	valid_to
1	John Doe	Junior Developer	50000	2022-01-01	2023-12-31
1	John Doe	Developer	60000	2024-01-01	2024-12-31
1	John Doe	Senior Developer	75000	2025-01-01	9999-12-31

- What was John’s salary on January 1, 2024?
- When did John become a Senior Developer?

SNAPSHOT TABLES

- Periodic copies of the data at specific points in time

January Snapshot (`inventory_snapshot_2025_01`)

product_id	product_name	quantity	snapshot_date
1	Laptop	100	2025-01-31
2	Phone	200	2025-01-31

February Snapshot (`inventory_snapshot_2025_02`)

product_id	product_name	quantity	snapshot_date
1	Laptop	80	2025-02-28
2	Phone	250	2025-02-28

- Useful for Reporting: Great for point-in-time reporting, financial summaries, or compliance requirements.
- Storage-Heavy: Requires a lot of storage since each snapshot is a full copy of the table.
- Limited Granularity: Cannot track changes between snapshots (you only see the state at snapshot times).
- Complex Queries: Comparing snapshots can get complicated for detailed change tracking.

THANK YOU!