Comparison of SQL and NoSQL Outputs for a Sample Employee Dataset

August 2025

1 Introduction

This document compares the output of SQL (using MySQL) and NoSQL (using MongoDB) databases when performing basic operations on a sample ”Employee” dataset. The dataset includes employee details such as ID, name, department, and salary. We demonstrate insert, select, update, and delete operations, highlighting differences in syntax, structure, and output.

2 Sample Dataset

The dataset represents employee records with the following fields:

- `employee_id`: Unique identifier (integer).
- `name`: Employee name (string).
- `department`: Department name (string).
- `salary`: Annual salary (integer).

Sample data:

- Employee 1: ID = 1, Name = Alice Smith, Department = HR, Salary = 50000
- Employee 2: ID = 2, Name = Bob Johnson, Department = IT, Salary = 60000

3 SQL (MySQL) Operations

SQL databases, such as MySQL, use a relational model with structured tables and a fixed schema. Below are the operations performed on the `Employee` table.

3.1 Table Creation

```sql
CREATE TABLE Employee (
    employee_id INT PRIMARY KEY,
    name VARCHAR(50),
    department VARCHAR(50),
    salary INT
);
```

3.2 Insert Operation

Inserting two employee records:

```
INSERT INTO Employee (employee_id, name, department, salary)
VALUES (1, 'Alice Smith', 'HR', 50000),
       (2, 'Bob Johnson', 'IT', 60000);
```

Output: No direct output; records are added to the table.

3.3 Select Operation

Retrieving all employees:

```
SELECT * FROM Employee;
```

Output:

```
+-------------+-------------+------------+--------+
| employee_id | name        | department | salary |
+-------------+-------------+------------+--------+
|           1 | Alice Smith | HR         | 50000  |
|           2 | Bob Johnson | IT         | 60000  |
+-------------+-------------+------------+--------+
```

3.4 Update Operation

Updating Alice's salary:

```
UPDATE Employee SET salary = 55000 WHERE employee_id = 1;
```

Output: No direct output; one row updated. Verify with:

```
SELECT * FROM Employee WHERE employee_id = 1;
```

Output:

```
+-------------+-------------+------------+--------+
| employee_id | name        | department | salary |
+-------------+-------------+------------+--------+
|           1 | Alice Smith | HR         | 55000  |
+-------------+-------------+------------+--------+
```

3.5 Delete Operation

Deleting Bob's record:

```
DELETE FROM Employee WHERE employee_id = 2;
```

Output: No direct output; one row deleted. Verify with:

```
SELECT * FROM Employee;
```

Output:

```
+-------------+-------------+------------+--------+
| employee_id | name        | department | salary |
+-------------+-------------+------------+--------+
|           1 | Alice Smith | HR         | 55000  |
+-------------+-------------+------------+--------+
```

4 NoSQL (MongoDB) Operations

MongoDB, a NoSQL database, uses a document-based model with collections of JSON-like documents and a flexible schema.

4.1 Collection Creation

No explicit schema creation is needed; documents are inserted directly into a collection.

4.2 Insert Operation

Inserting two employee documents:

```
db.employee.insertMany([
   { employee_id: 1, name: "Alice Smith", department: "HR", salary:
      50000 },
   { employee_id: 2, name: "Bob Johnson", department: "IT", salary:
      60000 }
]);
```

Output:

```
{
  "acknowledged": true,
  "insertedIds": [
    ObjectId("..."),
    ObjectId("...")
  ]
}
```

4.3 Select Operation

Retrieving all employees:

```
db.employee.find().pretty();
```

Output:

```
{
    ”_id”: ObjectId(”...”),
    ”employee_id”: 1,
    ”name”: ”Alice Smith”,
    ”department”: ”HR”,
    ”salary”: 50000
}
{
    ”_id”: ObjectId(”...”),
    ”employee_id”: 2,
    ”name”: ”Bob Johnson”,
    ”department”: ”IT”,
    ”salary”: 60000
}
```

4.4 Update Operation

Updating Alice’s salary:

```
db.employee.updateOne(
   { employee_id: 1 },
   { $set: { salary: 55000 } }
);
```

Output:

```
{ ”acknowledged”: true, ”matchedCount”: 1, ”modifiedCount”: 1 }
```

Verify with:

```
db.employee.find({ employee_id: 1 }).pretty();
```

Output:

```
{
    ”_id”: ObjectId(”...”),
    ”employee_id”: 1,
    ”name”: ”Alice Smith”,
    ”department”: ”HR”,
    ”salary”: 55000
}
```

4.5 Delete Operation

Deleting Bob’s document:

```
db.employee.deleteOne({ employee_id: 2 });
```

Output:

```
{ ”acknowledged”: true, ”deletedCount”: 1 }
```

Verify with:

```
db.employee.find().pretty();
```

Output:

```
{
    ”_id”: ObjectId(”...”),
    ”employee_id”: 1,
    ”name”: ”Alice Smith”,
    ”department”: ”HR”,
    ”salary”: 55000
}
```

5 Comparison of SQL and NoSQL

- **Data Model**: SQL uses a table-based structure with a fixed schema, requiring predefined columns and data types. MongoDB uses a document-based model with flexible schemas, allowing varied document structures within the same collection.
- **Syntax**: SQL uses standardized Structured Query Language for operations. MongoDB uses JavaScript-like syntax, which is less standardized but intuitive for developers familiar with JSON.
- **Output Format**: SQL outputs are tabular, with rows and columns, ideal for structured data. MongoDB outputs are JSON-like documents, suitable for hierarchical or unstructured data.
- **Scalability**: SQL databases (e.g., MySQL) are vertically scalable, requiring more powerful hardware for increased load. NoSQL databases (e.g., MongoDB) are horizontally scalable, distributing data across servers, making them better for large-scale data.
- **Performance**: For small, structured datasets, SQL performs well, especially for complex queries with joins or aggregates. MongoDB excels with large, unstructured datasets and simple key-value operations but may underperform for complex aggregates [1].
- **Consistency**: SQL databases ensure ACID compliance, guaranteeing strong consistency. MongoDB offers eventual consistency, prioritizing availability and scalability [2].

6 Conclusion

SQL (MySQL) is ideal for structured data requiring complex queries and strong consistency, as seen in its tabular output and standardized syntax. NoSQL (MongoDB) is better suited for flexible, unstructured data with high scalability needs, reflected in its document-based output. The choice depends on the application's requirements for structure, scalability, and consistency.

References

[1] A performance comparison of SQL and NoSQL databases, IEEE Conference Publication, 2013. https://ieeexplore.ieee.org/document/6637690

[2] SQL vs NoSQL - 7 Key Differences You Must Know, Algomaster, 2024. https://blog.algomaster.io/p/sql-vs-nosql