Storing Lists, JSON, and Lists of JSON in PostgreSQL with Spring Boot

Piyush Kumar

July 16, 2025

1 Overview

To store lists, JSON, and lists of JSON in a PostgreSQL column using Spring Boot, you can use two primary approaches. The JSON structure to be stored is:

```
{
  "name": "John Doe",
  "city": "New York",
  "listOfRoll": [101, 102, 103],
  "listOfName": ["Alice", "Bob", "Charlie"],
  "quizJson": {
    "Math": ["Algebra", "Geometry"],
    "Science": ["Physics", "Biology"]
  },
  "quizListOfJson": [
    {
      "History": ["World War I", "Renaissance"]
    },
    {
      "Geography": ["Mountains", "Rivers"]
    }
  ]
}
```

There are two main approaches to achieve this:

2 Approach 1: Modern Spring Boot with Built-in Support

This approach supports the latest Spring Boot projects and uses built-in Hibernate support without requiring third-party dependencies. Version management is handled by Spring Boot.

In the model class, use:

```
@JdbcTypeCode(SqlTypes.JSON)
@Column(columnDefinition = "jsonb")
```

This method is dependency-free, and Spring Boot manages the versions, reducing maintenance overhead.

3 Approach 2: Using Hibernate-Types Library

This older approach supports all Spring Boot versions but requires the `hibernate-types` dependency.

Add to `pom.xml`:

```
<dependency>
    <groupId>com.vladmihalcea</groupId>
    <artifactId>hibernate-types-60</artifactId>
    <version>2.21.1</version>
</dependency>
```

In the model class, use:

```
@Type(JsonType.class)
@Column(columnDefinition = "jsonb")
```

Disadvantage: Version mismatches can cause runtime issues.

4 Why Use `@Column(columnDefinition = "jsonb")`?

If you omit `@Column(columnDefinition = "jsonb")`, Spring Boot defaults to `@Column(columnDefinition = "json")`, and the code will still run. However, `jsonb` is recommended for:

1. **Faster Querying**
2. **Automatic Key Sorting**
3. **Storage Efficiency**
4. **Powerful Operators**

Using `jsonb` ensures better performance and query capabilities, even though its optional.

5 Example Code for Approach 1

```
import jakarta.persistence.*;
import lombok.*;
import org.hibernate.annotations.JdbcTypeCode;
import org.hibernate.type.SqlTypes;
import java.util.*;

@Entity
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
public class Student {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String name;
    private String city;

    // 1. list of value like List<Long>
    @JdbcTypeCode(SqlTypes.JSON)
```

```
    @Column(columnDefinition = "jsonb")
    private List<Long> listOfRoll;

    // 2. list of value like List<String>
    @JdbcTypeCode(SqlTypes.JSON)
    @Column(columnDefinition = "jsonb")
    private List<String> listOfName;

    // 3. json of value like Map<String, List<String>>
    @JdbcTypeCode(SqlTypes.JSON)
    @Column(columnDefinition = "jsonb")
    private Map<String, List<String>> quizJson;

    // 4. list of json of value like List<Map<String, List<String>>>
    @JdbcTypeCode(SqlTypes.JSON)
    @Column(columnDefinition = "jsonb")
    private List<Map<String, List<String>>> quizListOfJson;
}
```

6 Example Code for Approach 2

```
import com.vladmihalcea.hibernate.type.json.JsonType;
import jakarta.persistence.*;
import lombok.*;
import org.hibernate.annotations.Type;
import java.util.*;

@Entity
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
public class Student {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String name;
    private String city;

    // 1. list of value like List<Long>
    @Type(JsonType.class)
    @Column(columnDefinition = "jsonb")
    private List<Long> listOfRoll;

    // 2. list of value like List<String>
    @Type(JsonType.class)
    @Column(columnDefinition = "jsonb")
    private List<String> listOfName;

    // 3. json of value like Map<String, List<String>>
    @Type(JsonType.class)
    @Column(columnDefinition = "jsonb")
    private Map<String, List<String>> quizJson;

    // 4. list of json of value like List<Map<String, List<String>>>
    @Type(JsonType.class)
```

```
    @Column(columnDefinition = "jsonb")
    private List<Map<String, List<String>>> quizListOfJson;
}
```

7 Additional Notes

The repository, service, and controller layers remain the same as in standard CRUD operations. **Approach 1** is preferred due to its simplicity and lack of external dependencies. Use **Approach 2** only if working with older Spring Boot versions, but be prepared to manage dependency versions manually. Both approaches yield the same result when configured correctly.

Thanks: Piyush Kumar