### **Effort Estimation for Manual Audit Implementation**

Implementing manual auditing for 40 tables in an SQL Server backend would require significant development effort. The key tasks involved include:

1. **Designing Audit Tables (8-12 hours)**
   * Creating separate audit tables or a generic audit structure to track changes.
   * Defining schema for audit tables with metadata such as timestamps, user details, and change type.
2. **Writing Triggers for Each Table (40-60 hours)**
   * Implementing AFTER INSERT, UPDATE, and DELETE triggers for all 40 tables.
   * Ensuring performance optimization and handling bulk operations efficiently.
3. **Implementing Application-Level Audit Logging (40-50 hours)**
   * Writing custom services in Spring Boot to capture changes before persisting.
   * Modifying repository methods to track changes in all 40 tables.
4. **Testing and Debugging (30-40 hours)**
   * Ensuring correctness, integrity, and rollback mechanisms.
   * Writing automated and manual test cases to validate auditing.
5. **Deployment and Maintenance (20-30 hours)**
   * Setting up logging and monitoring.
   * Handling schema changes in the future.

### **Total Manual Effort: 140-190 hours (~4-6 weeks for 1 developer)**

* **High cost & effort**: Manual tracking increases development complexity and maintenance overhead.
* **Error-prone**: More risk of missing updates or incorrect tracking.

### **Benefits of Using GitHub Copilot & Spring Data Envers**

#### **1. GitHub Copilot Benefits**

* **Accelerates code writing**: Suggests optimized SQL triggers, repository methods, and audit table structures.
* **Reduces errors**: Helps prevent common pitfalls in manual implementation.
* **Saves ~30-40% of development time**: Faster coding for triggers and logging.

#### **2. Spring Data Envers Benefits**

* **Automatic Audit Logging**: Hibernate Envers tracks entity changes without writing explicit triggers.
* **Less Development Effort**: No need to create audit tables manually.
* **Easier Maintenance**: Works with Spring Boot’s existing repository structure.
* **Estimated Effort Reduction**: Reduces manual coding effort by **60-70%**, bringing total effort down to **50-70 hours (~2 weeks)**.

### **Conclusion for C-Level Executives**

* **Manual Implementation**: 4-6 weeks, high complexity, long-term maintenance cost.
* **Using Spring Data Envers & GitHub Copilot**: 2 weeks, significantly reduced development time, and more robust auditing.
* **Recommendation**: Adopt **Spring Data Envers** to ensure an efficient and scalable solution with minimal effort.

4o

##### **You said:**

Generate a step-by-step guide to present a usecase of implementing Auditing using Spring Dta envers. Provide optimised Github co-pilot prompts to modify pomxml and Entity class and any other files. Also provide the output of prompts. This is a step-by-step guide for developers to implement auditing using github copilot using spring data envers

###### **ChatGPT said:**

### **Step-by-Step Guide: Implementing Auditing Using Spring Data Envers with GitHub Copilot**

This guide provides a structured approach for developers to implement auditing in a **Spring Boot** project using **Spring Data Envers** and **GitHub Copilot**.

## **1. Add Dependencies to pom.xml**

### **GitHub Copilot Prompt:**

*"Modify the pom.xml file to include Spring Data Envers dependency along with Hibernate Envers for auditing support."*

### **Copilot-Suggested Code:**

xml

CopyEdit

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-envers</artifactId>

</dependency>

### **Explanation:**

* spring-boot-starter-data-jpa: Required for JPA operations.
* hibernate-envers: Provides automatic auditing of entity changes.

## **2. Enable Auditing in the Spring Boot Application**

### **GitHub Copilot Prompt:**

*"Create a Spring Boot configuration class to enable Hibernate Envers auditing."*

### **Copilot-Suggested Code:**

java

CopyEdit

import org.springframework.context.annotation.Configuration;

import org.springframework.data.jpa.repository.config.EnableJpaAuditing;

@Configuration

@EnableJpaAuditing

public class AuditConfig {

}

### **Explanation:**

* @EnableJpaAuditing: Enables JPA auditing in Spring Boot.

## **3. Modify Entity Class to Enable Auditing**

### **GitHub Copilot Prompt:**

*"Modify the User entity to support Hibernate Envers auditing."*

### **Copilot-Suggested Code:**

java

CopyEdit

import jakarta.persistence.\*;

import lombok.\*;

import org.hibernate.envers.Audited;

import java.time.LocalDateTime;

@Entity

@Audited // Enables auditing

@Data

@NoArgsConstructor

@AllArgsConstructor

public class User {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

private String email;

private LocalDateTime createdAt;

private LocalDateTime updatedAt;

}

### **Explanation:**

* @Audited: Enables Hibernate Envers to track changes to this entity.
* @Entity: Defines a JPA entity.
* LocalDateTime createdAt, updatedAt: Tracks timestamps for auditing.

## **4. Create an Audited Repository**

### **GitHub Copilot Prompt:**

*"Create a Spring Data JPA repository for the User entity with audit support."*

### **Copilot-Suggested Code:**

java

CopyEdit

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.data.history.RevisionRepository;

public interface UserRepository extends JpaRepository<User, Long>, RevisionRepository<User, Long, Integer> {

}

### **Explanation:**

* JpaRepository: Standard CRUD repository.
* RevisionRepository: Provides audit history methods.

## **5. Retrieve Audit History in a Service Class**

### **GitHub Copilot Prompt:**

*"Create a service method to fetch audit history for a given user."*

### **Copilot-Suggested Code:**

java

CopyEdit

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.data.history.Revision;

import org.springframework.data.history.Revisions;

import org.springframework.stereotype.Service;

@Service

public class UserService {

@Autowired

private UserRepository userRepository;

public Revisions<Integer, User> getUserAuditHistory(Long userId) {

return userRepository.findRevisions(userId);

}

}

### **Explanation:**

* findRevisions(userId): Fetches all historical changes for a specific user.

## **6. Expose Audit History via a REST API**

### **GitHub Copilot Prompt:**

*"Create a REST API endpoint to fetch the audit history of a user by ID."*

### **Copilot-Suggested Code:**

java

CopyEdit

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.data.history.Revisions;

import org.springframework.web.bind.annotation.\*;

@RestController

@RequestMapping("/users")

public class UserController {

@Autowired

private UserService userService;

@GetMapping("/{id}/audit")

public Revisions<Integer, User> getUserAudit(@PathVariable Long id) {

return userService.getUserAuditHistory(id);

}

}

### **Explanation:**

* @GetMapping("/{id}/audit"): Fetches audit history for a user.
* Calls getUserAuditHistory(id) method from UserService.

## **7. Testing the Audit Trail**

### **Steps to Test:**

**Create a new user** bash  
CopyEdit  
curl -X POST "http://localhost:8080/users" -H "Content-Type: application/json" -d '{"name": "John Doe", "email": "john@example.com"}'

**Update the user** bash  
CopyEdit  
curl -X PUT "http://localhost:8080/users/1" -H "Content-Type: application/json" -d '{"name": "John Doe Updated", "email": "john@example.com"}'

**Fetch audit history** bash  
CopyEdit  
curl -X GET "http://localhost:8080/users/1/audit"

## **Expected JSON Output for Audit History**

json

CopyEdit

{

"content": [

{

"revisionNumber": 1,

"entity": {

"id": 1,

"name": "John Doe",

"email": "john@example.com"

}

},

{

"revisionNumber": 2,

"entity": {

"id": 1,

"name": "John Doe Updated",

"email": "john@example.com"

}

}

]

}

## **Conclusion**

| **Method** | **Manual Implementation** | **With Spring Data Envers & Copilot** |
| --- | --- | --- |
| Development Time | 4-6 weeks | 2 weeks |
| Maintenance Complexity | High | Low |
| Error-prone? | Yes | No |
| Scalability | Difficult | Easy |

### **Final Recommendation:**

Using **Spring Data Envers** and **GitHub Copilot** reduces development time by **60-70%** and ensures a more robust, maintainable auditing system with minimal effort. 🚀