

Datenbanktests mit H2-Datenbank



Ahmad Aljeld Zubaydi

Lerne, wie du mit der H2-Datenbank effektive Datenbanktests für deine Anwendungen durchführst. Entdecke die Vorteile einer schnellen und unkomplizierten Testumgebung.

Was Dich in diesem Beitrag erwartet

1. Was ist ein Datenbanktest?
2. Warum ist das Testen von Datenbanken essenziell?

3. Was muss getestet werden?
4. H2-Datenbank
5. Welche Vorteile bietet die H2-Datenbank beim Testen?
6. Welche Nachteile ergeben sich beim Einsatz der H2-Datenbank beim Testen?
7. Einrichtung der H2-Datenbank in einem Spring Boot-Projekt



Datenbanken bilden das Herzstück vieler Software-Systeme und Anwendungen. Sie dienen als strukturiertes System zur Speicherung, Organisation und Verwaltung von Daten, auf die Anwendungen zugreifen, um sie abzurufen, zu speichern und zu verarbeiten.

Die Daten sind so gestaltet, dass sie effizient abgerufen, aktualisiert und verwaltet werden können. Da Datenbanken oft als Rückgrat geschäftskritischer Anwendungen fungieren, ist eine gründliche Überprüfung und Validierung der Datenbanken unerlässlich, um sicherzustellen, dass die Systeme ordnungsgemäß funktionieren und die Datenintegrität gewahrt bleibt.

Dieser Leitfaden bietet einen umfassenden Überblick über das Testen mit Datenbanken. Wir behandeln wesentliche Konzepte, Aufgaben und Prozesse im Zusammenhang mit dem Datenbanktest. Von der Planung und Vorbereitung von Datenbanktests bis hin zur Durchführung verschiedener Testarten beleuchten wir die wichtigsten Aspekte des Datenbanktests, die für Entwickler, Qualitätssicherungsingenieure und Datenbankadministratoren von Interesse sind. Darüber hinaus stellen wir bewährte Methoden und Werkzeuge vor, die bei der Durchführung effektiver Datenbanktests hilfreich sein können. In diesem Kontext widmen wir uns insbesondere dem Testen relationaler Datenbanken, erläutern, warum dies von entscheidender Bedeutung ist, welche Bereiche getestet werden müssen und wie spezielle Tools wie die H2-Datenbank dabei eingesetzt werden kann.

Zum Abschluss erläutern wir, wie Datenbanktests mit einer H2-Datenbank in Java eingerichtet werden können, um ein tieferes Verständnis zu fördern.

Was ist ein Datenbanktest?

Datenbanktests sind eine spezifische Methode innerhalb des Softwaretests, die darauf abzielt, die Funktionalität, Performance, Zuverlässigkeit und Sicherheit einer Datenbank zu evaluieren. Das Hauptziel besteht darin, sicherzustellen, dass die Datenbank ordnungsgemäß funktioniert und den festgelegten Anforderungen entspricht. Bei Datenbanktests werden mehrere Aspekte berücksichtigt: Es wird überprüft, ob die Datenbank in der Lage ist, Daten korrekt zu speichern, abzurufen, zu aktualisieren und zu löschen. Zudem wird sichergestellt, dass die Datenbank die definierten Geschäftsregeln einhält und die Integrität der Daten sicherstellt. Die Leistungsfähigkeit der Datenbank unter verschiedenen Lastbedingungen sowie die Wirksamkeit von Sicherheitsmechanismen werden ebenfalls bewertet.

Darüber hinaus wird die Zuverlässigkeit der Datenbank in Bezug auf Ausfallsicherheit und Wiederherstellbarkeit überprüft. Durch die Durchführung von Datenbanktests können potenzielle Fehler und Schwachstellen frühzeitig identifiziert und behoben werden, was zur Verbesserung der Qualität und Stabilität der Datenbank sowie der Gesamtqualität der Softwareanwendung beiträgt.

Warum ist das Testen von Datenbanken essenziell?

In vielen Softwareanwendungen wird das Testen ihrer Datenbanken während des Entwicklungsprozesses vernachlässigt, was potenzielle Schwachstellen und Risiken wie Datenverlust, Sicherheitslücken, Systemfehler und Anfälligkeiten für Fehler oder Angriffe offenlegt.

Um die genannten potenziellen Probleme zu vermeiden, bevor sie sich im Live-Betrieb auswirken, sind umfassende Tests für Datenbanken unerlässlich. Sie helfen, die Integrität der Daten zu wahren, die Funktionalität zu überprüfen und eine optimale Leistung sicherzustellen. Das unterstützt Unternehmen dabei, Probleme frühzeitig zu erkennen und zu beheben, um ihre wertvollen Daten zu schützen und einen reibungslosen Betrieb zu gewährleisten.

Das Testen von Datenbanken ist aus verschiedenen Gründen von wesentlicher Bedeutung:

- **Gewährleistung von Datenintegrität und -Qualität:**

Datenbanktests stellen sicher, dass die in der Datenbank gespeicherten Daten korrekt sind und den erwarteten Standards entsprechen. Dies hilft, Fehler und Inkonsistenzen zu vermeiden,

wodurch die Genauigkeit und Zuverlässigkeit der darauf aufbauenden Geschäftsprozesse gewährleistet werden.

- **Sicherheit:** Datenbanktests ermöglichen die Identifizierung und Behebung potenzieller Sicherheitslücken und Schwachstellen in der Datenbank. Dies ist besonders wichtig, um unbefugten Zugriff auf sensible Informationen zu verhindern und die Vertraulichkeit der Daten zu wahren.
- **Leistung und Skalierbarkeit:** Durch das Testen der Datenbankleistung unter verschiedenen Lastbedingungen können Engpässe und Flaschenhälse frühzeitig erkannt und behoben werden. Dies gewährleistet eine optimale Leistung, auch wenn die Datenbank mit zunehmender Datenmenge oder Benutzerzahl skaliert wird.
- **Compliance:** In vielen Branchen existieren gesetzliche Anforderungen und Vorschriften bezüglich des Umgangs mit Daten, insbesondere sensiblen Benutzerdaten. Datenbanktests tragen dazu bei, sicherzustellen, dass die Datenbank diesen Anforderungen entspricht und die Compliance mit den relevanten Vorschriften gewährleistet ist.
- **Kosteneffizienz:** Durch das frühzeitige Erkennen und Beheben von Fehlern in der Datenbankarchitektur und -Funktionalität können erhebliche Kosteneinsparungen erzielt werden. Fehler, die erst in späteren Phasen des Entwicklungszyklus oder nach der Bereitstellung entdeckt werden, sind oft teurer und zeitaufwendiger zu beheben.

Zusammenfassend trägt das Testen von Datenbanken dazu bei, die Qualität, Sicherheit, Leistung und Skalierbarkeit von Datenbankanwendungen zu verbessern und somit das Vertrauen der Benutzer in die Integrität ihrer Daten zu stärken.

Was muss getestet werden?

- **Funktionalität:** Ein beträchtlicher Teil der Datenbankfunktionalität, besonders Trigger, wird automatisch durch Modellierungswerkzeuge generiert. Die Frage ist, wie viele Tests für den generierten Code notwendig sind. Empfohlene Tests umfassen die Überprüfung der Existenz von Methoden, Testen der referenziellen Integrität (RI), Überprüfung von Berechnungen und Testen von View-Definitionen, um sicherzustellen, dass sie die erwarteten Ergebnisse liefern.
- **Beziehungen:** Beim Testen von Datenbankbeziehungen geht es vor allem um die Sicherstellung der referenziellen Integrität (RI). Wichtige Aspekte dabei sind:
 - Kaskadierende Löschungen: Sichern, dass beim Löschen eines Datensatzes auch alle verknüpften Datensätze korrekt entfernt werden, um ungewollten Datenverlust zu vermeiden.
 - Existenzregeln: Überprüfen, ob beim Hinzufügen eines Datensatzes die referenzierten Datensätze bereits existieren, was die richtige Reihenfolge des Einfügens beeinflusst.
 - Datensammlungen: Achten auf die RI innerhalb von Gruppen stark verknüpfter Datensätze und sicherstellen, dass sie bei Manipulationen erhalten bleibt.
- **Datenqualität:** Datenqualitätstests sind entscheidend, um sicherzustellen, dass die in der Datenbank gespeicherten Werte eine hohe Qualität haben. Wichtige Bereiche dafür sind:
 - Dateninhalt und -format: Überprüfung von Standardwerten, Nullwerten, Leerwert, Dateninvarianten, Attributgröße und Formatierung.
 - Datenaktualität und -Integrität: Sicherstellung der Aktualität, korrekter Datentypen, Verwendung von Einzweckfeldern, Normalisierung und das Fehlen von Daten.

- **Datenvolumen:** Überprüfung des Dateninhalts auf Übereinstimmung mit Erwartungen und das Erkennen von ungewöhnlichen Schwankungen im Volumen.
- **Performance:** Performance-Testing zielt darauf ab, die Effizienz und Geschwindigkeit einer Datenbank sicherzustellen. Zu den wichtigen Punkten gehören Zugriffszeiten, Ausführungszeiten von Abfragen, Indizes, Transaktionszeiten, der Testumfang (nur Datenbank oder gesamtes System) und die Skalierbarkeit (Leistung bei steigender Datenmenge und Benutzeranzahl).

Wir haben viele Aspekte untersucht, aber nicht alle sind in jedem Anwendungsfall erforderlich. Daher ist es wichtig, zu bestimmen, welche spezifischen Anforderungen relevant sind. In dieser Hinsicht ermöglicht gründliche Analyse der Anforderungen es uns, die prioritären Bereiche zu identifizieren und Ressourcen effizient zu nutzen, um die bestmöglichen Ergebnisse zu erzielen.

Neben technischen Aspekten müssen Datenbanktests sowohl die technischen Anforderungen als auch die Erwartungen der Benutzer und die Geschäftsziele berücksichtigen, um sicherzustellen, dass die Datenbank in beiden Aspekten einwandfrei funktioniert.

Im Folgenden werden wir einen Ansatz für Datenbanktests mit H2-Datenbanken untersuchen und dessen Vor- und Nachteile erläutern.

H2-Datenbank

Die H2-Datenbank ist ein leichtes, relationales SQL-Datenbankmanagementsystem mit Open-Source-Lizenz, das in Java entwickelt wurde. Sie kann nahtlos in Java-Anwendungen integriert oder im Client-Server-Modus betrieben werden. Eine besondere Eigenschaft von H2-Datenbank ist die Fähigkeit, als In-Memory-Datenbank zu agieren, wobei die Daten

nicht auf einem physischen Datenträger, sondern im Arbeitsspeicher gespeichert werden. Diese Funktionalität macht die H2-Datenbank besonders nützlich für Entwicklungszwecke und in Testumgebungen, insbesondere bei der Durchführung von Unit-Tests. Die einfache Konfiguration und Benutzerfreundlichkeit der H2-Datenbank ermöglichen eine schnelle und effiziente Codeprüfung und machen sie zu einer bevorzugten Option für Entwickler, die eine zuverlässige, aber dennoch leichtgewichtige Datenbanklösung suchen.

Die H2-Datenbank wird in verschiedenen Szenarien eingesetzt, die von ihrer Flexibilität und Effizienz profitieren. Nachfolgend sind die Hauptanwendungsbereiche aufgeführt:

- **Entwicklung und Testen:**

Die H2-Datenbank eignet sich aufgrund ihrer Einfachheit und schnellen Einrichtung ideal für Entwicklungs- und Testumgebungen, wo es oft wichtig ist, schnell eine Datenbank bereitzustellen und wieder zu entsorgen.

- **Lehrzwecke:**

Aufgrund ihrer unkomplizierten Konfiguration und Benutzerfreundlichkeit wird die H2-Datenbank oft in Bildungseinrichtungen eingesetzt, um Studierenden und Lehrenden das Erlernen und Lehren von Datenbankkonzepten und SQL zu erleichtern.

- **Prototypen:**

Die H2-Datenbank wird häufig für die Prototypenerstellung von Software-Anwendungen genutzt, weil sie es Entwicklern ermöglicht, schnell eine einsatzbereite Datenbankumgebung zu schaffen, ohne langwierige Einrichtungsprozesse. Dies fördert die zügige Entwicklung und Präsentation von Anwendungskonzepten.

- **Mobile und Embedded-Geräte:**

Aufgrund ihres geringen Speicher- und Verarbeitungsbedarfs ca. 2,5

MB Jar-Dateigröße ist die H2-Datenbank gut geeignet für den Einsatz auf mobilen Geräten und in eingebetteten Systemen, bei denen Ressourceneffizienz von großer Bedeutung ist.

Im Folgenden werde ich die Vor- und Nachteile des Einsatzes der H2-Datenbank für Tests erläutern.

Welche Vorteile bietet die H2-Datenbank beim Testen?

- 1. Schnelle Einrichtung und Einfachheit:** H2 ist sehr leichtgewichtig und kann schnell eingerichtet werden. Sie erfordert keine komplexe Installation oder Konfiguration, was den Prozess des Testaufbaus erheblich beschleunigt.
- 2. In-Memory-Funktion:** Eine der größten Stärken von H2 ist die Möglichkeit, vollständig im Speicher (RAM) zu operieren. Dies ermöglicht extrem schnelle Datenzugriffe und -manipulationen, was ideal für Tests ist, bei denen die Performanz der Datenbankzugriffe kritisch sein kann.
- 3. Kompatibilität mit SQL-Standards:** H2 unterstützt eine breite Palette von SQL-Funktionen und ist weitgehend kompatibel mit anderen SQL-Datenbanken wie PostgreSQL, MySQL etc. Dies erleichtert das Schreiben von wieder verwendbaren Tests.
- 4. Transaktionsmanagement:** H2 unterstützt vollständige Transaktionskontrolle, was bedeutet, dass Tests isoliert und wiederholbar durchgeführt werden können, ohne Seiteneffekte zu verursachen.
- 5. Einfache Integration in Entwicklungs- und Testumgebungen:** H2 kann leicht in gängige Entwicklungsumgebungen und Frameworks wie Spring, Hibernate und andere integriert werden.

Welche Nachteile ergeben sich beim Einsatz der H2-Datenbank beim Testen?

- 1. Abweichungen in der SQL-Implementierung:** Trotz der hohen Kompatibilität mit anderen Datenbanken bestehen weiterhin Unterschiede in der SQL-Implementierung und im Verhalten. Dies kann zu Problemen führen, wenn Anwendungen, die auf H2 getestet wurden, in Produktionsumgebungen mit anderen Datenbanken betrieben werden.
- 2. Eingeschränkte Funktionen:**
H2 bietet möglicherweise nicht alle spezialisierten Funktionen, die in größeren DBMS wie Oracle oder SQL-Server vorhanden sind, oder implementiert diese anders. Dies kann die Testabdeckung beeinträchtigen.
- 3. Datenpersistenz:** Obwohl H2 die Möglichkeit bietet, Daten auf einer Disk zu speichern, ist sie primär für ihren In-Memory-Betrieb bekannt. Dies kann bei Tests, die eine dauerhafte Datenhaltung benötigen, zu Herausforderungen führen.
- 4. Skalierbarkeits- und Performanz-Grenzen:** In Szenarien, in denen die Datenbank unter hoher Last oder mit großen Datenvolumen betrieben wird, kann die H2-Datenbank an ihre Grenzen stoßen, insbesondere im In-Memory-Modus.
- 5. Mangel an Unterstützung in manchen Umgebungen:** Obwohl H2 in vielen Java-basierten Umgebungen gut unterstützt wird, kann es sein, dass sie in anderen Programmiersprachen oder Plattformen nicht die gleiche Unterstützung oder Integration bietet.

Einrichtung der H2-Datenbank in einem Spring Boot-Projekt

Das nachfolgende Beispiel demonstriert, wie die H2-Datenbank in einem Spring Boot-Projekt eingerichtet werden kann. Hierfür müssen bestimmte Schritte durchgeführt werden.

1. Hinzufügen der benötigten Abhängigkeiten

Da Maven als Build-Tool verwendet wird, sollen die folgenden Abhängigkeiten zur pom.xml hinzugefügt werden, um die H2-Datenbank und weitere benötigte Bibliotheken zu integrieren.

```
1 <dependencies>
2     <dependency>
3         <groupId>org.springframework.boot</groupId>
4         <artifactId>spring-boot-starter-data-
5             jpa</artifactId>
6     </dependency>
7     <dependency>
8         <groupId>org.springframework.boot</groupId>
9         <artifactId>spring-boot-starter-web</artifactId>
10    </dependency>
11    <dependency>
12        <groupId>com.h2database</groupId>
13        <artifactId>h2</artifactId>
14        <scope>runtime</scope>
15    </dependency>
16    <dependency>
17        <groupId>org.projectlombok</groupId>
18        <artifactId>lombok</artifactId>
19        <optional>true</optional>
20    </dependency>
21    <dependency>
22        <groupId>org.springframework.boot</groupId>
23        <artifactId>spring-boot-starter-test</artifactId>
24        <scope>test</scope>
25    </dependency>
</dependencies>
```

2. Erstellen einer Entitätsklasse, die eine Datenbanktabelle darstellt

Wir verwenden die Lombok-Bibliothek, die durch Annotationen die automatische Generierung von Gettern, Settern, Konstruktoren und weiteren Methoden ermöglicht. Dies vereinfacht die Codebasis und macht sie lesbarer:

```
1 import jakarta.persistence.Entity;
2 import jakarta.persistence.Id;
3 import lombok.AllArgsConstructor;
4 import lombok.Data;
5 import lombok.NoArgsConstructor;
6
7 @Entity
8 @Data
9 @AllArgsConstructor
10 @NoArgsConstructor
11 public class Car {
12
13     @Id
14     private Long id;
15
16     private String brand;
17
18     private String color;
19
20     private String transmission;
21 }
```

3. Repository für die Entitätsklasse erstellen

Das Repository ermöglicht den Zugriff auf die Daten der Car-Klasse und wird in Spring Boot in Verbindung mit dem Hibernate-Framework verwendet, um die Datenbankoperationen zu verwalten und die Datenbankabfragen zu definieren.

```
1 import org.springframework.data.jpa.repository.JpaRepository;
2
3 import java.util.Optional;
4
5 public interface CarRepository extends JpaRepository<Car, Long> {
6
7     Car save(Car car);
8
9     Optional<Car> findById(Long id);
10
11     void deleteById(Long id);
12 }
```

In diesem Beispiel erweitert `CarRepository` das `JpaRepository`, ein integraler Bestandteil des Spring Data JPA-Frameworks, das eine Vielzahl von Methoden für Datenbankoperationen wie Speichern, Aktualisieren, Löschen und Suchen bereitstellt. In diesem Repository sind drei einfache Methoden definiert: eine zum Speichern eines einzelnen Eintrags, eine weitere zum Abfragen eines Eintrags anhand seiner ID und eine zum Löschen eines Eintrags unter Angabe seiner ID.

4. Aufsetzen der Testklasse

In dieser Klasse konfigurieren wir die Verbindung zur H2-Datenbank und fügen das Repository als Bean in unseren Test hinzu. Dadurch können wir Operationen auf der Datenbanktabelle durchführen, die durch die Car-Klasse dargestellt wird. Der Code implementiert einen Integrationstest für das `CarRepository`, indem er die H2-Datenbank konfiguriert und einen einfachen Testfall ausführt, der das Speichern, Abrufen und Löschen von Datenbank-Eintrag demonstriert.

```
1 import org.assertj.core.api.Assertions;
2 import org.junit.jupiter.api.Test;
3 import org.junit.jupiter.api.extension.ExtendWith;
4 import org.springframework.beans.factory.annotation.Autowired;
5 import org.springframework.boot.autoconfigure.EnableAutoConfiguration;
6 import org.springframework.boot.test.autoconfigure.orm.jpa.DataJpaTest;
7 import org.springframework.test.context.ContextConfiguration;
8 import org.springframework.test.context.TestPropertySource;
9 import org.springframework.test.context.junit.jupiter.SpringExtension;
10
11 import java.util.Optional;
12
13 @ExtendWith(SpringExtension.class)
14 @EnableAutoConfiguration
15 @ContextConfiguration(classes = {CarRepository.class})
16 @TestPropertySource(properties = {
17     "spring.datasource.url=jdbc:h2:mem:h2TestDb",
18     "spring.jpa.database-platform=org.hibernate.dialect.H2Dialect",
19     "spring.jpa.hibernate.ddl-auto=create-drop"})
20
21 @DataJpaTest
22 class CarRepositoryH2Test {
23
24     @Autowired
25     private CarRepository carRepository;
26
27     @Test
```

```
28 |     void givenCarWhenSaveThenGet() {
29 |         Car bmw = new Car(1L, "BMW", "red", "auto");
30 |         Car savedCar = carRepository.save(bmw);
31 |
32 |         Optional<Car> foundCar = carRepository.findById(1L);
33 |         Assertions.assertThat(foundCar)
34 |             .isPresent()
35 |             .contains(savedCar);
36 |
37 |         carRepository.deleteById(1L);
38 |
39 |         Assertions.assertThat(carRepository.findById(1L))
40 |             .isEmpty();
41 |     }
42 | }
```

Nachdem wir die Grundkonfiguration abgeschlossen haben, besteht nun eine funktionierende Verbindung zur H2-Datenbank, welche eine Tabelle für die Car-Klasse bereitstellt. Sie können nun Tests gegen diese In-Memory-Datenbank durchführen und sicherstellen, dass Ihre Anwendungslogik wie erwartet funktioniert.

Wir haben festgestellt, dass die H2-Datenbank ideal für schnelle und unkomplizierte Testumgebungen, insbesondere bei Java-Anwendungen, geeignet ist. Sie ermöglicht schnelle Iterationen ohne aufwändige Datenbankkonfiguration. Dennoch sollte bei einem Übergang in eine Produktionsumgebung die Kompatibilität der Produktionsdatenbank mit H2 sorgfältig geprüft werden, um Überraschungen zu vermeiden.

Das könnte Dich auch noch interessieren

Keycloak SPIs implementieren – Schritt für Schritt

Die Open-Source Identity- und Accessmanagement-Lösung Keycloak kann leicht erweitert werden. Dieser Beitrag zeigt, wie man die Service Provider Interface (SPI)

...

[WEITERLESEN](#)

Software-Qualität und Automatisierung – der Rückblick

Impressionen und Folien zum Conciso-Event “Software-Qualität und Automatisierung” am 12.06.24 ...

WEITERLESEN

Wie sieht zeitgemäßes Software Engineering aus?

Welche Antworten Conciso momentan auf diese Frage hat, habe ich im Rahmen der Digitalen Woche Dortmund 2020 bei einem Webinar ...

WEITERLESEN

Conciso GmbH

Pariser Bogen 7
44269 Dortmund

[Google Maps](#) | [OpenStreetMap](#) | [Apple Karten](#)

Tel.: +49 231 226175-0
Fax: +49 231 226175-10
E-Mail: info@conciso.de

Wichtige Inhalte

[Agile Advisory](#)

[Viable Architecture](#)

[Pragmatic Development](#)

[Wissensbeiträge](#)

[Jobs](#)

[Kontakt](#)

Contentletter abonnieren

Ihr Name

Ihre E-Mail-Adresse

ABONNIEREN