

Benutzerhandbuch: Der KI-gestützte ERM-Editor

Michael Niedermair und andere, realisiert mit KI-Unterstützung



generiert mit Google AI Studio



1	Was ist dieses Werkzeug?	3
1.1	Dein persönlicher Assistent für Datenbank-Modelle	3
1.2	Die vier Spalten im Überblick	3
2	Dein erster Start	4
2.1	Was du brauchst	4
2.2	Die Konfiguration verstehen und anpassen	4
2.3	Start über die Kommandozeile	5
3	Der Weg zum fertigen SQL-Skript	5
3.1	Schritt 1: Von der Idee zur Strukturanalyse	5
3.2	Schritt 2: Das Konzept visualisieren (ERM)	6
3.3	Schritt 3: Der Bauplan für die Datenbank (Logisches Modell)	7
3.4	Schritt 4: Der fertige Code (SQL-DDL)	8
4	Deine Werkzeuge für den Durchblick	9
4.1	Der Wissens-Tutor: Dein persönlicher Experte	9
4.2	Die kontextbezogenen Tutoren: Hilfe genau da, wo du sie brauchst	10
4.3	Speichern und Laden: So verlierst du nie deinen Fortschritt	10
4.4	Exportieren: Deine Ergebnisse für die Weiterverwendung	11
5	Was tun, wenn...? (Erste Hilfe)	11
5.1	Probleme beim Start der Anwendung	11
5.2	Probleme mit der KI-Antwort	12
5.3	Probleme mit den Diagrammen	12
6	Ein Blick unter die Haube (für Entwickler)	13
6.1	Die Architektur im Überblick	13
6.2	Die KI-Konfiguration anpassen	14
6.3	Die Prompts der Agenten verändern	14
6.4	Eigene Erweiterungen	14



1 Was ist dieses Werkzeug?

1.1 Dein persönlicher Assistent für Datenbank-Modelle

Stell dir vor, du hast eine Idee für ein Projekt – eine App für deine Schulbibliothek, eine Verwaltung für deinen Sportverein oder ein digitales Kochbuch. Für all diese Dinge brauchst du eine gut strukturierte Datenbank. Aber wie kommt man von einer vagen Idee zu einem professionellen Datenbank-Design?

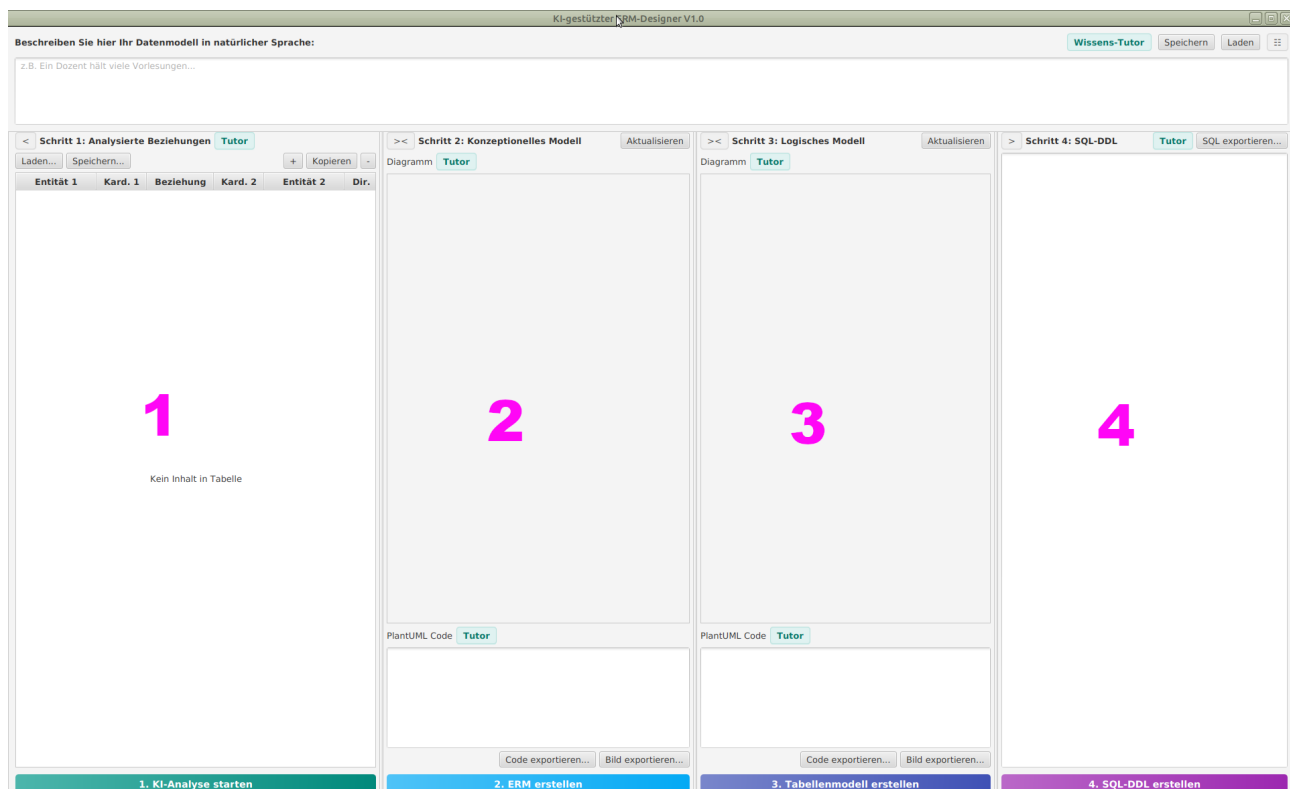
Genau hier hilft dir der ERM-Editor. Er ist ein Lern- und Arbeitswerkzeug, das dich durch die vier zentralen Phasen des Datenbank-Entwurfs führt:

1. **Analyse:** Du beschreibst deine Idee in normalen Sätzen und eine KI hilft dir, die wichtigen Objekte (z. B. „Buch“, „Mitglied“) und ihre Beziehungen zu finden.
2. **Konzeptioneller Entwurf:** Du erstellst ein übersichtliches Diagramm (ein ER-Modell), das die Struktur deiner Daten wie ein Bauplan zeigt.
3. **Logischer Entwurf:** Du übersetzt den Bauplan in ein konkretes Tabellenmodell, das schon fast wie eine echte Datenbank aussieht.
4. **Physischer Entwurf:** Du generierst den fertigen SQL-Code, mit dem du die Datenbank auf einem Server erstellen kannst.

Am Ende dieses Prozesses hast du nicht nur ein fertiges Ergebnis, sondern auch verstanden, **warum** deine Datenbank so aufgebaut ist, wie sie ist.

1.2 Die vier Spalten im Überblick

Die Oberfläche des Editors ist in vier Spalten aufgeteilt, die genau diesen vier Schritten entsprechen. Du arbeitest dich also logisch von links nach rechts durch dein Projekt.



2 Dein erster Start

Bevor du loslegen kannst, musst du dem Editor kurz sagen, welche KI er verwenden soll und wie er sie erreichen kann. Das klingt kompliziert, ist aber ganz einfach.

2.1 Was du brauchst

1. Das Programm:

Die `erm-editor-VERSION.jar`-Datei.

2. Eine Konfigurationsdatei:

Du hast eine `config-example.properties`-Datei erhalten. Kopiere diese Datei und nenne sie z. B. `meine-config.properties`.

3. Eine „Geheimnis“-Datei:

Erstelle eine leere Textdatei namens `ki-tokens.properties`. Hier kommen deine persönlichen API-Schlüssel hinein. **Wichtig:** Diese Datei solltest du an einem sicheren Ort speichern, außerhalb deines Projektordners, und sie niemals mit anderen teilen!

2.2 Die Konfiguration verstehen und anpassen

Öffne deine beiden neuen Dateien in einem Texteditor (`meine-config.properties` und `ki-tokens.properties`).

1. Die `ki-tokens.properties` befüllen:

Trage hier deinen persönlichen API-Schlüssel ein, den du z.B. vom Google AI Studio bekommen hast.

```
# Beispiel für ki-tokens.properties
gemini_api_key=AIzaSy_DEIN_PERSOENLICHER_SCHLUESSEL_HIER
github_api_key=ghp_DEIN_PERSOENLICHER_SCHLUESSEL_HIER
```

2. Die `meine-config.properties` anpassen:

Hier musst du nur eine einzige Zeile ändern.

Suche die Zeile `ermssystem.gemini.secure.properties.path` und trage dort den vollständigen Pfad zu deiner `ki-tokens.properties`-Datei ein.

```
1# =====
2# Beispiel-Konfigurationsdatei für den ERM-Editor
3#
4# ANLEITUNG:
5# 1. Kopieren Sie diese Datei an einen sicheren Ort (z.B. /home/user/erm-config.properties).
6# 2. Erstellen Sie eine separate Datei für Ihre API-Schlüssel (z.B. /home/user/.secrets/ki-tokens.properties).
7# 3. Passen Sie die Pfade und Werte in Ihrer Kopie dieser Datei an.
8# 4. Starten Sie die Anwendung mit (Version anpassen):
9#   java -jar erm-0.0.0.jar <präfix> <pfad-zu-ihre-datei>
10#   Beispiel: java -jar erm-0.0.0.jar ermsystem.gemini /home/user/erm-config.properties
11# =====
12
13# -----
14# Konfigurations-Set 1: Verwendung von Google Gemini für alle Modelle
15# Präfix: ermsystem.gemini
16# -----
17
18# Pfad zur separaten, sicheren Datei, die die eigentlichen API-Schlüssel enthält.
19ermssystem.gemini.secure.properties.path=/tmp/ki-tokens.properties
20
21# --- Modell für die Analyse (Schritt 1) ---
22ermssystem.gemini.analysis.system=GEMINI
23ermssystem.gemini.analysis.model.name=gemini-1.5-flash-latest
24ermssystem.gemini.analysis.model.temperature=0.1
25ermssystem.gemini.analysis.model.topP=0.9
26ermssystem.gemini.analysis.model.timeoutSeconds=120
27ermssystem.gemini.analysis.model.logRequests=false
28# Verweis auf den Schlüssel in der 'secure.properties.path'-Datei
29ermssystem.gemini.analysis.model.apiKeyLookup=gemini_api_key
30
```



2.3 Start über die Kommandozeile

Öffne nun ein Terminal oder eine Kommandozeile. Navigiere in das Verzeichnis, in dem deine `erm-editor-VERSION.jar`-Datei liegt.

Gib nun den folgenden Befehl ein und passe die Teile in `<...>` an:

```
java -jar erm-editor-VERSION.jar <präfix> <pfad-zu-deiner-config>
```

- **<präfix>:**
Das Konfigurations-Set, das du verwenden möchtest.
Für den Anfang ist das `ermsystem.gemini`.
- **<pfad-zu-deiner-config>:**
Der vollständige Pfad zu deiner `meine-config.properties`-Datei.

Ein konkretes Beispiel könnte so aussehen:

```
java -jar erm-editor-0.1.0.jar ermsystem.gemini /schule/db/meine-config.properties
```

Nachdem du Enter gedrückt hast, sollte der ERM-Editor starten. Herzlichen Glückwunsch, du bist bereit für dein erstes Projekt!

3 Der Weg zum fertigen SQL-Skript

In diesem Kapitel gehen wir den gesamten Weg gemeinsam durch. Wir erstellen ein einfaches Datenmodell für eine **Schulbibliothek**. Am Ende wirst du sehen, wie aus einer einfachen Beschreibung ein komplettes, lauffähiges SQL-Skript wird.

3.1 Schritt 1: Von der Idee zur Strukturanalyse

Jedes Projekt beginnt mit einer Idee. Unsere Idee ist, die Ausleihen von Büchern an Schüler zu verwalten.

Deine Aufgabe:

Gib die folgende Beschreibung in das obere Textfeld der Anwendung ein. Du kannst den Text einfach kopieren und einfügen.

- Ein Schüler kann mehrere Bücher ausleihen.
- Jedes Buch kann von mehreren Schülern ausgeliehen werden.

Beschreiben Sie hier Ihr Datenmodell in natürlicher Sprache:

- Ein Schüler kann mehrere Bücher ausleihen.
- Jedes Buch kann von mehreren Schülern ausgeliehen werden.

Entität 1	Kard. 1	Beziehung	Kard. 2	Entität 2	Dir.
Schüler	0..*	leiht aus	0..*	Buch	>

1. KI-Analyse starten

Klicke anschließend auf den Button **1. KI-Analyse starten**.



Was passiert jetzt?

Die KI liest und versteht deinen Text. Sie extrahiert die wichtigsten Objekte (genannt **Entitäten**) und die Handlungen (genannt **Beziehungen**) dazwischen.

Das Ergebnis:

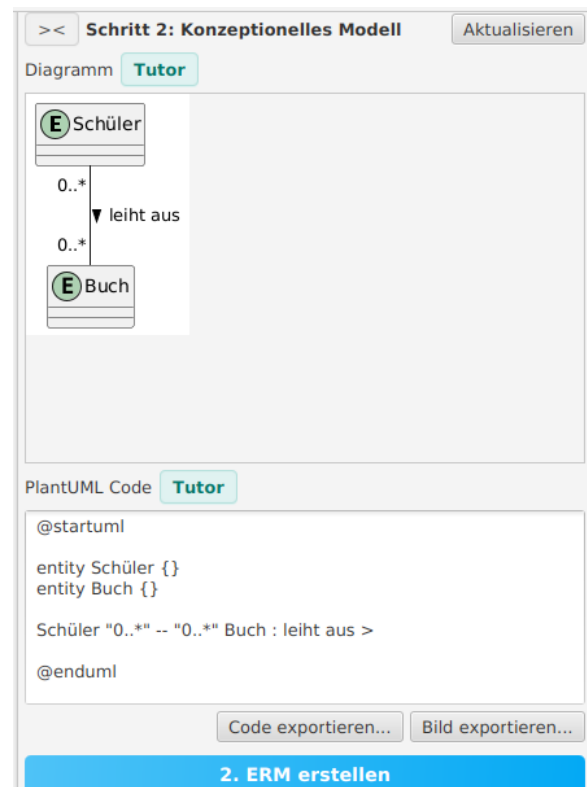
Die Tabelle in der ersten Spalte füllt sich automatisch. Du siehst jetzt die erkannten Beziehungen, zum Beispiel, dass ein „Schüler“ in einer Beziehung zu einem „Buch“ steht. Die Spalten „Kard.“ (Kardinalität) zeigen an, wie viele Objekte an der Beziehung beteiligt sind („*“ bedeutet „viele“).

3.2 Schritt 2: Das Konzept visualisieren (ERM)

Eine Tabelle ist gut, aber ein Bild ist oft besser. Jetzt erstellen wir aus den analysierten Daten einen visuellen Bauplan, das **Entity-Relationship-Modell (ERM)**.

Deine Aufgabe:

Klicke auf den Button **2. ERM erstellen**.



Das Ergebnis:

In der zweiten Spalte erscheinen zwei Dinge:

- Oben:**
Ein Diagramm mit Rechtecken (den Entitäten) und Linien (den Beziehungen). Das ist dein ER-Modell. Es gibt dir einen schnellen Überblick über die Struktur.
- Unten:**
Der sogenannte **PlantUML-Code**, aus dem das Diagramm erzeugt wird. Das Coole daran: Du kannst diesen Code direkt bearbeiten! Ändere z. B. den Text einer Beziehung, klicke auf **Aktualisieren** oben in der Spalte und beobachte, wie sich das Diagramm anpasst.

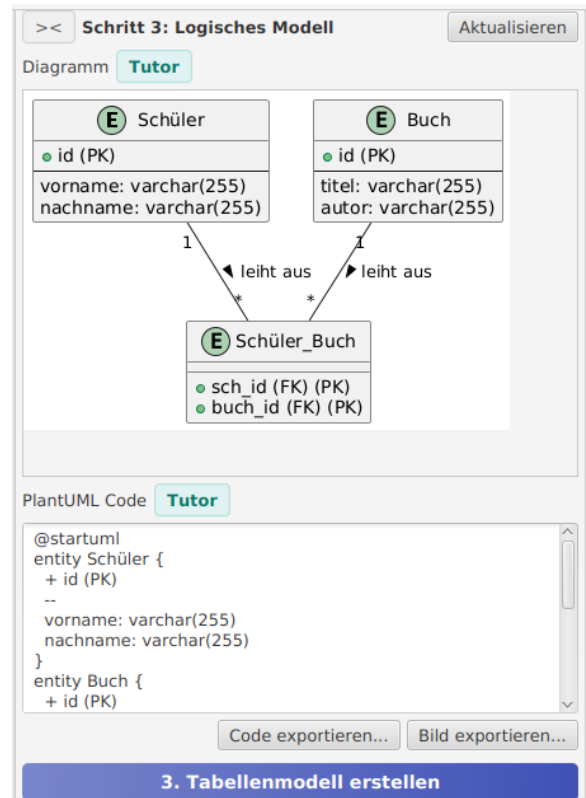
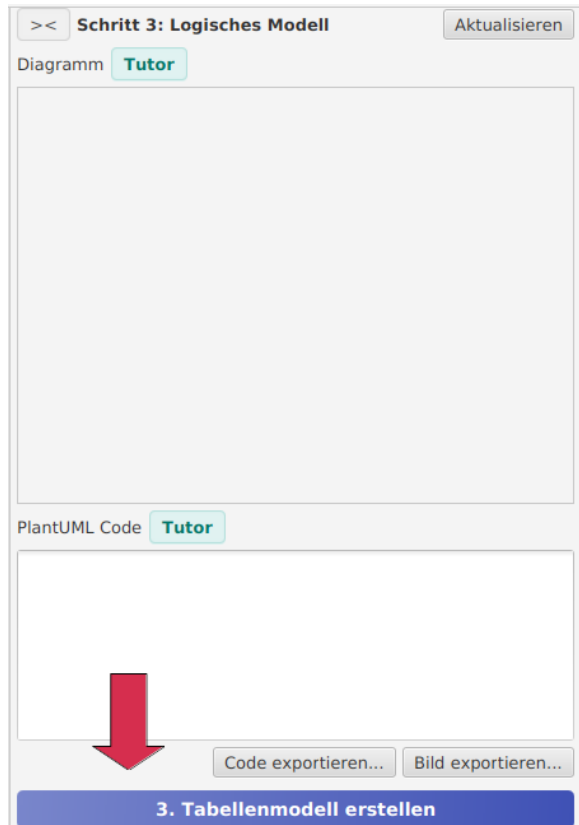


3.3 Schritt 3: Der Bauplan für die Datenbank (Logisches Modell)

Das ERM ist ein konzeptioneller Plan. Jetzt werden wir konkreter und erstellen ein **logisches Tabellenmodell**. Dieses Modell zeigt, wie die Tabellen in der Datenbank später tatsächlich aussehen werden.

Deine Aufgabe:

Klicke auf den Button **3. Tabellenmodell erstellen**.



Das Ergebnis:

In der dritten Spalte erscheint ein neues, detaillierteres Diagramm. Achte auf die wichtigen Unterschiede:

1. Jede Tabelle hat jetzt einen Primärschlüssel `id (PK)`, um jeden Eintrag eindeutig zu machen.
2. Die `n:m`-Beziehung („viele Schüler leihen viele Bücher“) wurde durch eine **Zwischentabelle** („Schüler_Buch“) aufgelöst. Diese neue Tabelle enthält nur die Verweise („FK“ für Foreign Key, also Fremdschlüssel) auf die beiden Originaltabellen. Das ist ein zentrales Konzept im Datenbankdesign!

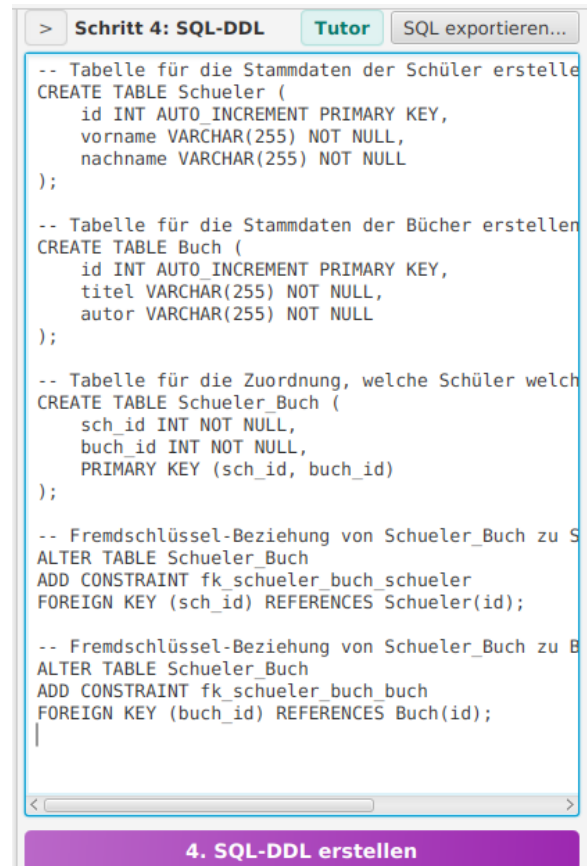
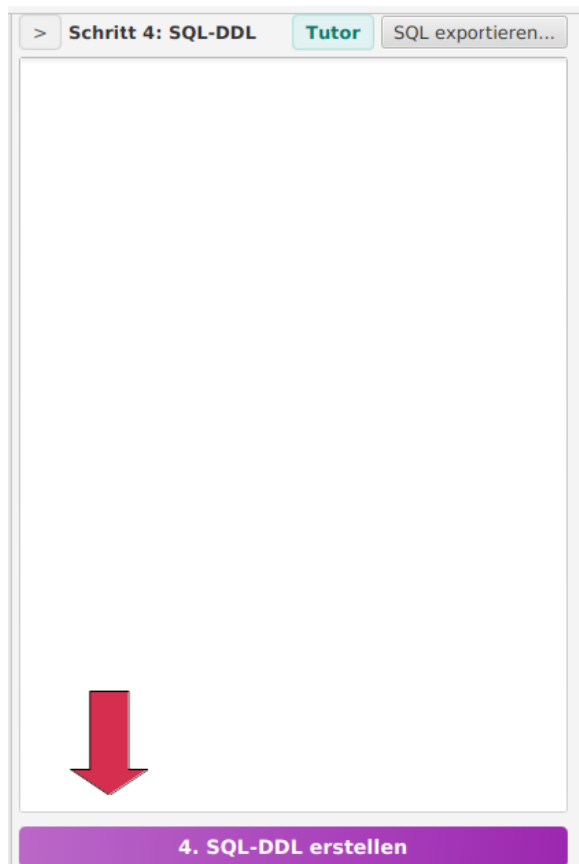


3.4 Schritt 4: Der fertige Code (SQL-DDL)

Der letzte Schritt! Jetzt wandeln wir den detaillierten Bauplan in die Sprache um, die Datenbanksysteme verstehen: **SQL**.

Deine Aufgabe:

Klicke auf den Button **4. SQL-DDL erstellen**.



Das Ergebnis:

In der vierten Spalte siehst du nun das fertige **SQL-DDL-Skript**. DDL steht für „Data Definition Language“ – es sind die Befehle, um die Struktur der Datenbank zu definieren. Du siehst hier die `CREATE TABLE`-Anweisungen, mit denen die Tabellen und ihre Spalten erstellt werden. Am Ende werden mit `ALTER TABLE` die Fremdschlüssel-Beziehungen hinzugefügt.

Diesen Code kannst du nun kopieren und in einem echten Datenbanksystem (wie MariaDB oder MySQL) ausführen, um deine Datenbank zu erstellen.



4 Deine Werkzeuge für den Durchblick

Der ERM-Editor ist mehr als nur ein Generator. Er bietet dir eine Reihe von Werkzeugen, die dich beim Lernen unterstützen und dir helfen, deine Arbeit zu organisieren.

4.1 Der Wissens-Tutor: Dein persönlicher Experte

Manchmal hast du Fragen, die über den aktuellen Schritt hinausgehen. Was genau ist „Normalisierung“? Was ist der Unterschied zwischen `VARCHAR` und `TEXT`? Für all diese Fragen gibt es den **Wissens-Tutor**.

Wissens-Tutor

Tutor: Hallo und willkommen! Ich unterstütze dich gerne beim Lernen rund um Datenbanken und Software-Entwicklung. Du kannst jederzeit eine Frage stellen oder ein Thema auswählen.

Du: Wie funktionieren relationale Datenbanken?

Tutor: Relationale Datenbanken speichern Daten in Tabellen, die aus Zeilen und Spalten bestehen. Jede Tabelle repräsentiert eine Entität, wie zum Beispiel 'Kunde' oder 'Bestellung'. Die Spalten enthalten die Attribute dieser Entität, zum Beispiel Name oder Adresse. Jede Zeile stellt einen Datensatz dar.

Die Beziehungen zwischen den Tabellen werden durch Schlüssel hergestellt:

1. Primärschlüssel: Ein eindeutiges Attribut, das jede Zeile einer Tabelle identifiziert.
2. Fremdschlüssel: Ein Attribut, das auf den Primärschlüssel einer anderen Tabelle verweist und so die Verknüpfung herstellt.

Mit SQL (Structured Query Language) können Daten abgefragt, eingefügt, geändert oder gelöscht werden. Die Integrität der Daten wird durch Regeln wie Referentielle Integrität und Normalisierung gewahrt.

Relationale Datenbanken sind besonders geeignet für strukturierte Daten und unterstützen komplexe Abfragen und Transaktionen.

Stellen Sie eine Frage oder wählen Sie einen Vorschlag:

- Was ist eine Tabelle in einer relationalen Datenbank?
- Wie funktionieren Primär- und Fremdschlüssel?
- Was bedeutet referentielle Integrität?
- Was ist SQL und wie wird es verwendet?
- Was versteht man unter Normalisierung?
- Welche Vorteile bieten relationale Datenbanken?

Ihre Frage... Senden

So nutzt du ihn:

Klicke oben rechts auf den Button **Wissens-Tutor**.

Es öffnet sich ein großes Dialogfenster. Hier kannst du jede beliebige Frage zum Thema Datenbanken, SQL oder Datenmodellierung stellen. Die KI wird dir eine ausführliche Antwort geben und dir direkt passende Folgefragen vorschlagen, um noch tiefer in das Thema einzutauchen.



4.2 Die kontextbezogenen Tutoren: Hilfe genau da, wo du sie brauchst

Zusätzlich zum allgemeinen Wissens-Tutor findest du in jeder der vier Spalten einen kleineren **Tutor**-Button.

Diese Tutoren sind Spezialisten für genau den Schritt, in dem du dich gerade befindest. Wenn du zum Beispiel in Spalte 2 (ER-Modell) auf den Tutor-Button klickst, kannst du gezielt Fragen zum ER-Modell stellen, das du gerade siehst. Der Tutor kennt den Kontext deiner Arbeit und kann dir so viel präziser helfen.

Lernassistent: ERM-Diagramm

Tutor:

Herzlich willkommen! Ich unterstütze dich gerne beim Verständnis von konzeptionellen ERM-Diagrammen. Ein ERM-Diagramm hilft dabei, die Struktur von Daten und deren Beziehungen zu modellieren. Dabei sind Entitäten, Beziehungen und Kardinalitäten besonders wichtig. Hast du schon einmal mit ERM-Diagrammen gearbeitet oder möchtest du die Grundlagen kennenlernen?

Du:

Wie unterscheiden sich Beziehungen von Entitäten?

Tutor:

Entitäten sind die zentralen Objekte oder Dinge, die im ERM-Diagramm dargestellt werden, wie zum Beispiel 'Schüler' und 'Buch'. Sie repräsentieren reale oder abstrakte Dinge, über die Informationen gespeichert werden. Beziehungen hingegen verknüpfen diese Entitäten miteinander und beschreiben, wie sie zueinander stehen. Im gegebenen Diagramm verbindet die Beziehung 'leiht aus' die Entitäten 'Schüler' und 'Buch' und zeigt, dass Schüler Bücher ausleihen können.

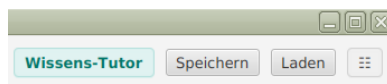
Stellen Sie eine Frage oder wählen Sie einen Vorschlag:

- Was bedeutet die Kardinalität 0..* in diesem Diagramm?
- Wie würde man Attribute zu einer Entität hinzufügen?
- Kann eine Beziehung auch eigene Attribute besitzen?

Ihre Frage... **Senden**

4.3 Speichern und Laden: So verlierst du nie deinen Fortschritt

Ein Datenbankmodell entsteht selten in fünf Minuten. Damit du deine Arbeit jederzeit unterbrechen und später fortsetzen kannst, gibt es die Funktionen zum Speichern und Laden.



Speichern: Klicke oben rechts auf **Speichern**. Du kannst deinen gesamten Fortschritt – von der Beschreibung über die Tabellen bis zum SQL-Code – in einer einzigen Datei mit der Endung

.ermproj (ERM-Projekt) sichern.

Laden: Klicke auf **Laden**, um eine .ermproj-Datei auszuwählen. Die Anwendung stellt sofort den gespeicherten Zustand in allen vier Spalten wieder her.

Tipp: Es gibt auch die Möglichkeit, **nur** die analysierten Beziehungen aus Schritt 1 zu speichern und zu laden. Die entsprechenden Buttons findest du direkt in der ersten Spalte. Das ist nützlich, wenn du verschiedene Modellierungs-Ansätze mit derselben Analyse ausprobieren möchtest.



4.4 Exportieren: Deine Ergebnisse für die Weiterverwendung

Wenn du mit einem Schritt fertig bist, möchtest du die Ergebnisse vielleicht in anderen Dokumenten oder Programmen verwenden. Dafür gibt es in den Spalten 2, 3 und 4 Export-Buttons.

Code exportieren...

Bild exportieren...

5 Was tun, wenn...? (Erste Hilfe)

Auch bei bester Planung kann mal etwas schiefgehen. Keine Sorge! Hier findest du Lösungen für die häufigsten Probleme, die beim Arbeiten mit dem ERM-Editor auftreten können.

5.1 Probleme beim Start der Anwendung

Problem: Nach dem Ausführen des Startbefehls erscheint eine Fehlermeldung wie `Error: A JNI error has occurred` oder `Hauptklasse ... konnte nicht gefunden werden`.

- **Lösung 1:** Befehl prüfen

Überprüfe deinen Befehl ganz genau auf die richtige Reihenfolge und Leerzeichen. Er muss immer diesem Muster folgen:

```
java -jar <jar-datei> <präfix> <pfad-zur-config>
```

- **Lösung 2:** Richtiger Ordner?

Stelle sicher, dass du dich im Terminal auch wirklich in dem Ordner befindest, in dem deine `.jar`-Datei liegt.

- **Lösung 3:** Java-Version

Prüfe, ob du eine aktuelle Java-Version installiert hast. Gib dazu im Terminal `java --version` ein. Es sollte mindestens Version '21' sein.

Problem: Die Anwendung startet, meldet aber sofort einen Fehler, dass die Konfigurationsdatei nicht geladen werden konnte.

- **Lösung:** Der Pfad zu deiner `.properties`-Datei im Startbefehl ist sehr wahrscheinlich falsch. Überprüfe ihn auf Tippfehler.

Probleme mit der KI und den Ergebnissen

Problem: Ich klicke auf einen der „Erstellen“-Buttons, aber es passiert nichts, es dreht sich nur der Lade-Indikator oder es kommt eine Fehlermeldung.

- **Lösung 1:** Internetverbindung

Die meisten KI-Modelle laufen online. Stelle sicher, dass du eine aktive Internetverbindung hast.

- **Lösung 2:** API-Schlüssel

Öffne deine `ki-tokens.properties`-Datei. Ist der richtige API-Schlüssel eingetragen? Ist er vielleicht aus Versehen unvollständig kopiert worden oder enthält Leerzeichen am Anfang oder Ende?

- **Lösung 3:** Geduld

Manchmal kann ein KI-Dienst kurzzeitig überlastet sein. Warte einen Moment und versuche es einfach noch einmal.



5.2 Probleme mit der KI-Antwort

Problem: Die KI-Analyse in Schritt 1 liefert seltsame oder unvollständige Ergebnisse.

- **Lösung:** Die KI ist nur so gut wie deine Anweisung.
Formuliere deine Beschreibung um. Manchmal helfen klarere und einfachere Sätze. Statt „Schüler leihen sich Bücher“ versuche es mit „Ein Schüler kann mehrere Bücher ausleihen.“
- **Korrigiere von Hand!**
Die Tabelle ist vollständig editierbar. Du kannst jederzeit Zeilen hinzufügen ('+'), löschen ('-') oder die Inhalte direkt in den Zellen ändern. Die KI gibt dir nur einen ersten Vorschlag.

5.3 Probleme mit den Diagrammen

Problem: In Spalte 2 oder 3 wird kein Diagramm angezeigt, obwohl im Textfeld darunter Code steht.

- **Lösung: Syntaxfehler!**
Das ist fast immer ein Zeichen für einen Fehler im PlantUML-Code.
 - Überprüfe den Code im Textfeld. Oft fehlt eine geschweifte Klammer { }, ein Anführungszeichen " oder eine Zeile endet nicht korrekt.
 - Klicke auf den **Aktualisieren**-Button oben in der jeweiligen Spalte, nachdem du den Code korrigiert hast, um das Diagramm neu zu zeichnen.



6 Ein Blick unter die Haube (für Entwickler)

Dieses Kapitel richtet sich an alle, die tiefer in die Technik des ERM-Editors einsteigen, ihn anpassen oder erweitern möchten. Ein grundlegendes Verständnis von Java, Maven und dem Konzept von APIs ist hierfür hilfreich.

6.1 Die Architektur im Überblick

Der ERM-Editor basiert auf einer klaren Trennung von Zuständigkeiten, um den Code wartbar und erweiterbar zu halten. Die drei wichtigsten Säulen sind:

1. **Die Benutzeroberfläche (View):**

Besteht aus JavaFX (`.fxml`-Dateien für das Layout, `MainController.java` für die Logik). Die View ist „dumm“: Sie zeigt nur Daten an und leitet Benutzeraktionen an den `ErmGeneratorService` weiter. Sie enthält selbst keine Logik zur Datentransformation.

2. **Die Service-Schicht (Service):**

Das Herzstück der Anwendung ist der `ErmGeneratorService.java`. Er agiert als Vermittler zwischen der Oberfläche und den KI-Agenten. Wenn der Controller sagt „Erstelle ein ERM“, ruft der Service den entsprechenden Agenten auf, verarbeitet dessen Antwort und gibt das Ergebnis (z. B. ein Bild) an den Controller zurück.

3. **Die KI-Agenten (Agents):**

Jede Kernfunktion ist in einem eigenen KI-Agenten gekapselt (z. B. `AnalysisAgent`, `SqlDdlAgent`). Diese sind als Java-Interfaces definiert und nutzen die Bibliothek **LangChain4j**. `LangChain4j` übersetzt unsere Java-Methodenaufrufe (z. B. `analysisAgent.analyzeRelationships(...)`) automatisch in eine Anfrage an eine große Sprachmodell (LLM) API. Der eigentliche „Prompt“ – also die Anweisung an die KI – ist direkt in der `@SystemMessage` oder `@UserMessage` Annotation innerhalb des Agenten-Interfaces definiert.

Der Datenfluss ist also immer:

```
Benutzeraktion (View)
→ Aufruf an Service
→ Service ruft Agenten
→ Agent (LangChain4j) kommuniziert mit LLM
→ Antwort geht zurück an Service
→ Service verarbeitet Antwort
→ Ergebnis wird an View zurückgegeben
```



6.2 Die KI-Konfiguration anpassen

Wie du in Kapitel 2 gesehen hast, wird die gesamte KI-Konfiguration über eine `.properties`-Datei gesteuert. Dies wird durch die `KiModelFactory.java` ermöglicht.

- **Neue Modelle hinzufügen:**

Um ein neues Modell (z.B. für eine neue Funktion) hinzuzufügen, musst du in der `MainApp.java` einfach einen weiteren Aufruf an `KiModelFactory.createFromPrefix(...)` mit einem neuen Modell-Typ (z.B. `documentation`) hinzufügen. Anschließend musst du die entsprechenden Einträge in deiner `.properties`-Datei ergänzen (z.B. `ermsystem.gemini.documentation.model.name=...`).

- **Andere KI-Anbieter nutzen:**

Die `KiModelFactory` unterstützt aktuell Gemini, Ollama und GitHub. Um einen weiteren Anbieter (z.B. OpenAI) hinzuzufügen, müsstest du die Factory um eine Methode `createOpenAiModel` erweitern und den entsprechenden LangChain4j-Konnektor in der `pom.xml` hinzufügen.

6.3 Die Prompts der Agenten verändern

Die „Intelligenz“ der Anwendung liegt in den Prompts. Wenn du das Verhalten der KI ändern möchtest, ist dies der richtige Ort.

Beispiel: Du möchtest, dass der `SqlDdlAgent` SQL für PostgreSQL anstatt für MariaDB erzeugt.

1. Öffne die Datei `SqlDdlAgent.java`.
2. Finde die `@SystemMessage`-Annotation.
3. Passe den Text an: Ändere die Anweisungen von „...standardkonformes SQL (MariaDB-Dialekt)“ zu „...standardkonformes SQL (PostgreSQL-Dialekt)“. Du könntest auch die Datentypen im Beispiel anpassen (z.B. `SERIAL` statt `INT AUTO_INCREMENT`).

Das ist alles. Beim nächsten Start wird LangChain4j diese geänderte Anweisung an die KI senden, und das Ergebnis wird ein anderes sein. Dies ist die große Stärke eines prompt-basierten Ansatzes: Die Kernlogik ist in natürlicher Sprache formuliert und leicht anpassbar.

6.4 Eigene Erweiterungen

Möchtest du eine ganz neue Funktion hinzufügen, zum Beispiel einen Agenten, der aus dem SQL-DDL eine Dokumentation im Markdown-Format generiert?

1. **Neuen Agenten erstellen:** Erstelle ein neues Interface, z.B. `DocumentationAgent.java`. Formuliere in der `@SystemMessage` genau, was die KI tun soll.
2. **Service erweitern:** Füge dem `ErmGeneratorService` eine neue Methode hinzu, z.B. `generateDocumentation(String sqlDdl)`, die den neuen Agenten aufruft.
3. **Controller und View anpassen:** Füge in der `MainView.fxml` einen neuen Button hinzu und verknüpfe ihn im `MainController` mit deiner neuen Service-Methode.
4. **Konfiguration erweitern:** Füge in der `MainApp` und der `.properties`-Datei die Konfiguration für das Modell des neuen Agenten hinzu.

Dieses modulare Design macht es relativ einfach, die Anwendung zu erweitern, ohne den bestehenden Code grundlegend ändern zu müssen.



Komponentendiagramm: ERM-Editor V1.0

