Session 0 – Vorstellung: Mein Weg zu Datenbanken & interaktivem OER

> **Session-Typ:** Einführung / Vorstellung (keine vollständige Vorlesung) **Dauer:** ca. 30–45 Minuten
> **Fokus:** Persönlicher Background, technologische Reise, Motivation für diese Vorlesung

Hinweis: Diese Session ist **kein Pflichtbestandteil** der Vorlesung, sondern eine persönliche Einladung, meinen Hintergrund kennenzulernen und zu verstehen, warum diese Vorlesung so gestaltet ist, wie sie ist.

Zusammenfassung

In dieser Session stelle ich mich vor – André Dietrich, Ihr Dozent für diese Vorlesung. Ich gebe Ihnen einen Einblick in meinen akademischen und beruflichen Werdegang, von meiner Promotion in eingebetteten Systemen und Robotik über meine Arbeit mit verteilten Datenbanksystemen bis hin zur Entwicklung interaktiver Lehr- und Lernmaterialien.

Dabei erzähle ich, wie meine Leidenschaft für Programmiersprachen, Datenbankparadigmen und Web-Technologien mich zu Projekten wie LiaScript, SelectScript, cassandra_ros, Industrial eLab und Edrys-Lite geführt hat – und warum ich überzeugt bin, dass interaktive Open Educational Resources die Zukunft des Lernens sind.

Über mich – André Dietrich

Promotion: Eingebettete Systeme & Robotik

Ich habe in Embedded Systems und Robotics an der Otto-von-Guericke-Universität Magdeburg promoviert. Der Fokus meiner Dissertation lag auf dem Internet of Things und dem Zugriff auf verteilte Systeme.

Zentrale Frage meiner Forschung:

> Wie können wir große Mengen an Sensordaten aus verschiedenen Quellen effizient sammeln, speichern und abfragen?

Herausforderungen:

- **Heterogenität:** Sensoren liefern Daten in unterschiedlichen Formaten (JSON, Binär, Protobuf)
- **Verteilung:** Daten kommen von vielen Quellen (Roboter, Mikrocontroller, Cloud-Services)
- **Skalierbarkeit:** Millionen von Datenpunkten pro Sekunde
- **Abfragbarkeit:** Wie machen wir diese Daten sinnvoll zugänglich?

Während meiner Promotion entwickelte sich eine Faszination für Programmiersprachen und Paradigmen. Eine zentrale Frage war: Wie können wir Daten holistisch und intuitiv zugänglich machen – ohne uns in technischen Details zu verlieren?

Interesse: Programmiersprachen & Paradigmen

Leitfragen:

- Wie können Abfragesprachen über klassische Datenbanken hinausgehen?
- Welche Paradigmen eignen sich für welche Anwendungsfälle?
- Wie können wir Komplexität reduzieren, ohne Mächtigkeit zu verlieren?

Meine Projekte: Von NoSQL zu OER

cassandra_ros – NoSQL & Sensordaten

Mein erstes größeres Datenbankprojekt war ein ROS-Adapter für Apache Cassandra, einen NoSQL Wide Column Store. Das Ziel: Sensorsignale aus Robotern in einer verteilten Datenbank speichern und abfragen.

Projekt-Steckbrief:

Aspekt	Details
Projekt	ROS-Adapter für Apache Cassandra
Ziel	Sensorsignale aus Robotern speichern
Technologie	Cassandra (NoSQL Wide Column Store)
Abfragesprache	CQL (Cassandra Query Language)
Link	cassandra_ros Wiki

Was habe ich dabei gelernt? Wide Column Stores bieten flexible Schemas und horizontale Skalierbarkeit – perfekt für write-heavy Workloads wie Sensordaten. CQL ermöglicht SQL-ähnliche Abfragen auf NoSQL-Systemen. Aber es gibt Trade-offs: Eventual Consistency versus starke Konsistenz – das CAP-Theorem in der Praxis.

Wichtigste Erkenntnisse:

Wide Column Store (Cassandra)
✓ Flexible Schemas ✓ Horizontale Skalierbarkeit ✓ Write-heavy Workloads ✓ SQL-ähnliche Abfragen (CQL) ✗ Eventual Consistency (CAP-Theorem) ✗ Komplexe Joins schwierig

Trade-off (CAP-Theorem):

- **Consistency:** Alle Knoten sehen dieselben Daten zur gleichen Zeit
- **Availability:** System antwortet immer (auch bei Netzwerkpartitionierung)
- **Partition Tolerance:** System funktioniert trotz Netzwerkausfällen

> Cassandra wählt: **Availability + Partition Tolerance** → Eventual Consistency

SelectScript – Deklarative Abfragesprache

Parallel entwickelte ich SelectScript – eine Lua-ähnliche, eingebettete Programmiersprache für Simulationsumgebungen. Die Besonderheit: SELECT-Statements werden nicht nur für Datenabfragen verwendet, sondern für allgemeine Problemlösungen.

Projekt-Steckbrief:

Aspekt	Details
Projekt	Deklarative Abfragesprache für Simulationen
Besonderheit	SELECT-Statements für Problemlösung
Anwendung	Rekursive Abfragen, hierarchische Strukturen, robotische Weltmodelle
Beispiele	Türme von Hanoi, 4-Farben-Problem, Graphtraversierung
Link	SelectScript auf GitHub

SelectScript zeigt, dass SQL-ähnliche Syntax weit über klassische Datenbanken hinausgehen kann. Rekursive Queries sind mächtiger, als man denkt – ob Türme von Hanoi, Graphtraversierung oder Constraint-Satisfaction-Probleme. Deklarative Sprachen ermöglichen intuitive Problemlösung.

Beispiel: Rekursive Query (Türme von Hanoi)

```
mov
  = PROC(Tower, frm, to)
    "A simple tower move function that returns a new tower configura
     mov([[3,2,1], [], []], 0, 1) -> [[3,2], [1], []]

     In case of an unalowed move a None value gets returned:
     mov([[3,2], [1], []], 0, 1)  -> None "
    : ( IF( $Tower == None, EXIT None);

        IF( not $Tower[$frm], EXIT None);

        IF( $Tower[$to],
            IF( $Tower[$frm][-1] > $Tower[$to][-1],
                EXIT None));

        $Tower[$to]@+( $Tower[$frm][-1] );
        $Tower[$frm]@pop();
        $Tower;
      );


# initial tower configuration
tower = [[3,2,1], [], []];

# allowed moves [from, to]
moves = [[0,1], [0,2], [1,0], [1,2], [2,0], [2,1]];

# goal configuration
finish = [[], [], [3,2,1]];

```

```
31
32
33  # vanilla-approach: recusively test all combinations for 7 moves
34  $start_time = time();
35  rslt1 = SELECT [$m1, $m2, $m3, $m4, $m5, $m6, $m7]
36            FROM m1:moves, m2:moves, m3:moves, m4:moves,
37                 m5:moves, m6:moves, m7:moves
38           WHERE finish == (tower
39                              |> mov($m1[0], $m1[1])
40                              |> mov($m2[0], $m2[1])
41                              |> mov($m3[0], $m3[1])
42                              |> mov($m4[0], $m4[1])
43                              |> mov($m5[0], $m5[1])
44                              |> mov($m6[0], $m6[1])
45                              |> mov($m7[0], $m7[1]))
46             AS list;
47
48  print("#################################################################
      ###");
49  print("first vanilla-approach search");
50  print("time:   ", time()-$start_time);
51  print("result: ", rslt1);
52
53
54
55  $start_time = time();
56  rslt2 = SELECT $m
57            FROM m:moves
58           WHERE finish == mov($tower, $m[0], $m[1])
59     START WITH $tower = tower
60     CONNECT BY $tower@mov($m[0], $m[1])
61      STOP WITH $tower == None OR $step$ > 6
62               AS list;
63
64  print("#################################################################
      ###");
65  print("simple CONNECT BY (recursive search)");
66  print("time:   ", time()-$start_time);
67  print("result: ", rslt2);
68
69
70
71  $start_time = time();
72  rslt3 = SELECT $tower
73            FROM m:moves
74           WHERE finish == mov($tower, $m[0], $m[1])
75     START WITH $tower = tower
76     CONNECT BY NO CYCLE
77                $tower@mov($m[0], $m[1])
78      STOP WITH $tower == None OR $step$ > 6
79               AS LIST;
```

```
80
81  print("#################################################################
      ###");
82  print("CONNECT BY with no cycles");
83  print("time:   ", time()-$start_time);
84  print("result: ", rslt3);
85
86
87  rslt4 = SELECT $step$, $tower, $m
88                FROM m:moves
89               WHERE finish == mov($tower, $m[0], $m[1])
90      START WITH $tower = tower
91      CONNECT BY UNIQUE
92                 $tower@mov($m[0], $m[1])
93       STOP WITH $tower == None OR $step$ > 7
94                 AS LIST;
95
96  print("#################################################################
      ###");
97  print("CONNECT BY with UNIQUE");
98  print("time:   ", time()-$start_time);
99  print("result: ", rslt4);
100
101 True;
```

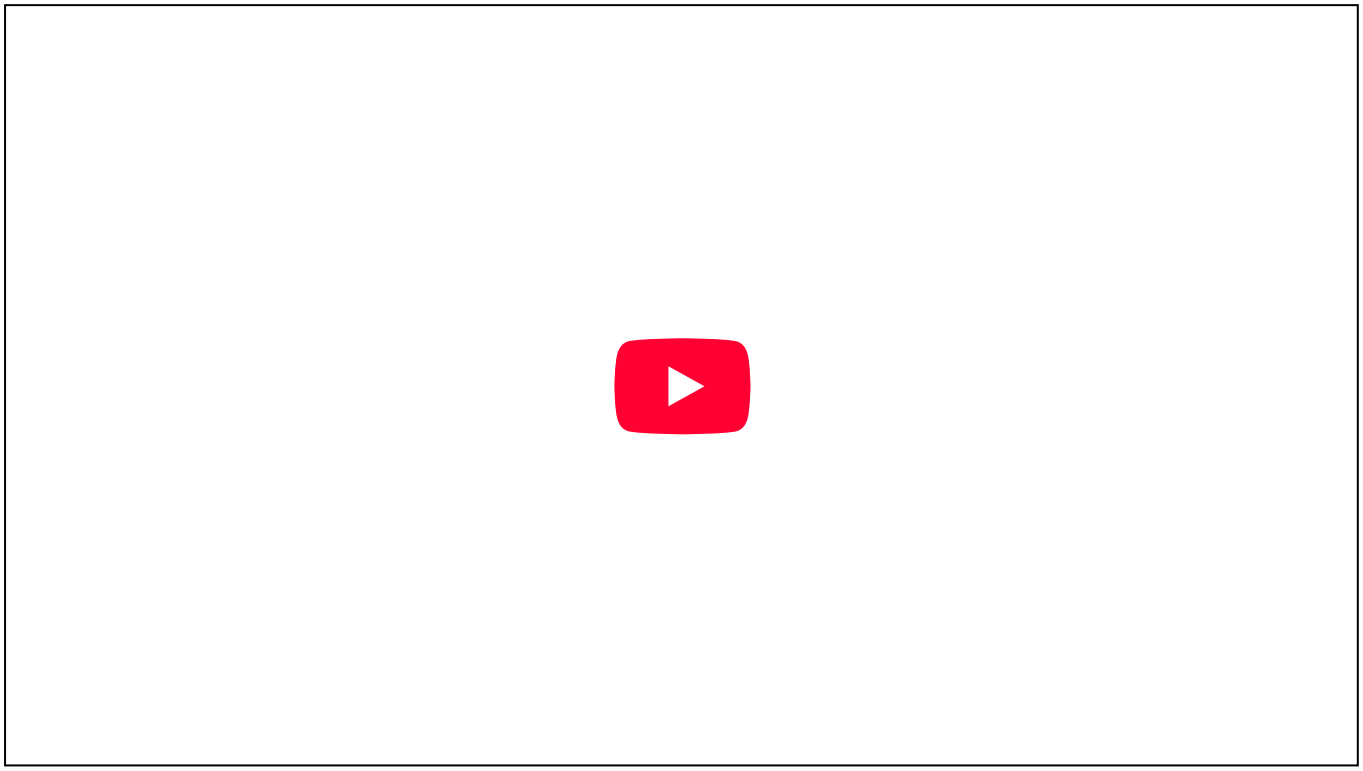
```
WARNING: A restricted method in java.lang.System has been called
WARNING: java.lang.System::loadLibrary has been called by
com.sun.jna.Native in an unnamed module (file:/usr/local/bin/S2c.jar)
WARNING: Use --enable-native-access=ALL-UNNAMED to avoid a warning for
callers in this module
WARNING: Restricted methods will be blocked in a future release unless
native access is enabled


 __      _            _   __           _       _
/ _\ ___| | ___  ___| |_/ _\ ___ _ __(_)_ __ | |_
\ \ / _ \ |/ _ \/ __| __\ \ / __| '__| | '_ \| __|
_\ \  __/ |  __/ (__| |__\ \ (__| |  | | |_) | |_
\__/\___|_|\___|\___|\__\__/\___|_|  |_| .__/ \__|
                                       |_|

######################################################################
first vanilla-approach search
time:    3.93045
result:  [[[0,2],[0,1],[2,1],[0,2],[1,0],[1,2],[0,2]]]
######################################################################
simple CONNECT BY (recursive search)
time:    0.3672
result:  [[[0,2],[0,1],[2,1],[0,2],[1,0],[1,2],[0,2]]]
######################################################################
CONNECT BY with no cycles
time:    0.552
result:  [[[[3,2,1],[],[]],[[3,2],[],[1]],[[3],[2],[1]],[[3],[2,1],[]],
[[],[2,1],[3]],[[1],[2],[3]],[[1],[],[3,2]]]]
######################################################################
CONNECT BY with UNIQUE
time:    0.2146
result:  [[0,[[3,2,1],[],[]],[0,1],1,[[3,2],[1],[]],[1,2],2,[[3,2],[],
[1]],[0,1],3,[[3],[2],[1]],[2,1],4,[[3],[2,1],[]],[0,2],5,[[],[2,1],
[3]],[1,0],6,[[1],[2],[3]],[1,2],7,[[1],[],[3,2]],[0,2]],[0,[[3,2,1],
[],[]],[0,2],1,[[3,2],[],[1]],[0,1],2,[[3],[2],[1]],[2,1],3,[[3],[2,1],
[]],[0,2],4,[[],[2,1],[3]],[1,0],5,[[1],[2],[3]],[1,2],6,[[1],[],
[3,2]],[0,1],7,[[],[1],[3,2]],[1,2]],[0,[[3,2,1],[],[]],[0,2],1,[[3,2],
[],[1]],[0,1],2,[[3],[2],[1]],[2,1],3,[[3],[2,1],[]],[0,2],4,[[],[2,1],
[3]],[1,0],5,[[1],[2],[3]],[1,2],6,[[1],[],[3,2]],[0,2]]]
RESULT: true
```

Wichtigste Erkenntnisse:

- **Deklarative Sprachen** ermöglichen intuitive Problemlösung
- **SQL-ähnliche Syntax** kann über klassische Datenbanken hinausgehen
- **Rekursive Queries** sind mächtig (Graphtraversierung, Constraint-Solving)



Industrial eLab – Remote Labs & Web-Technologien

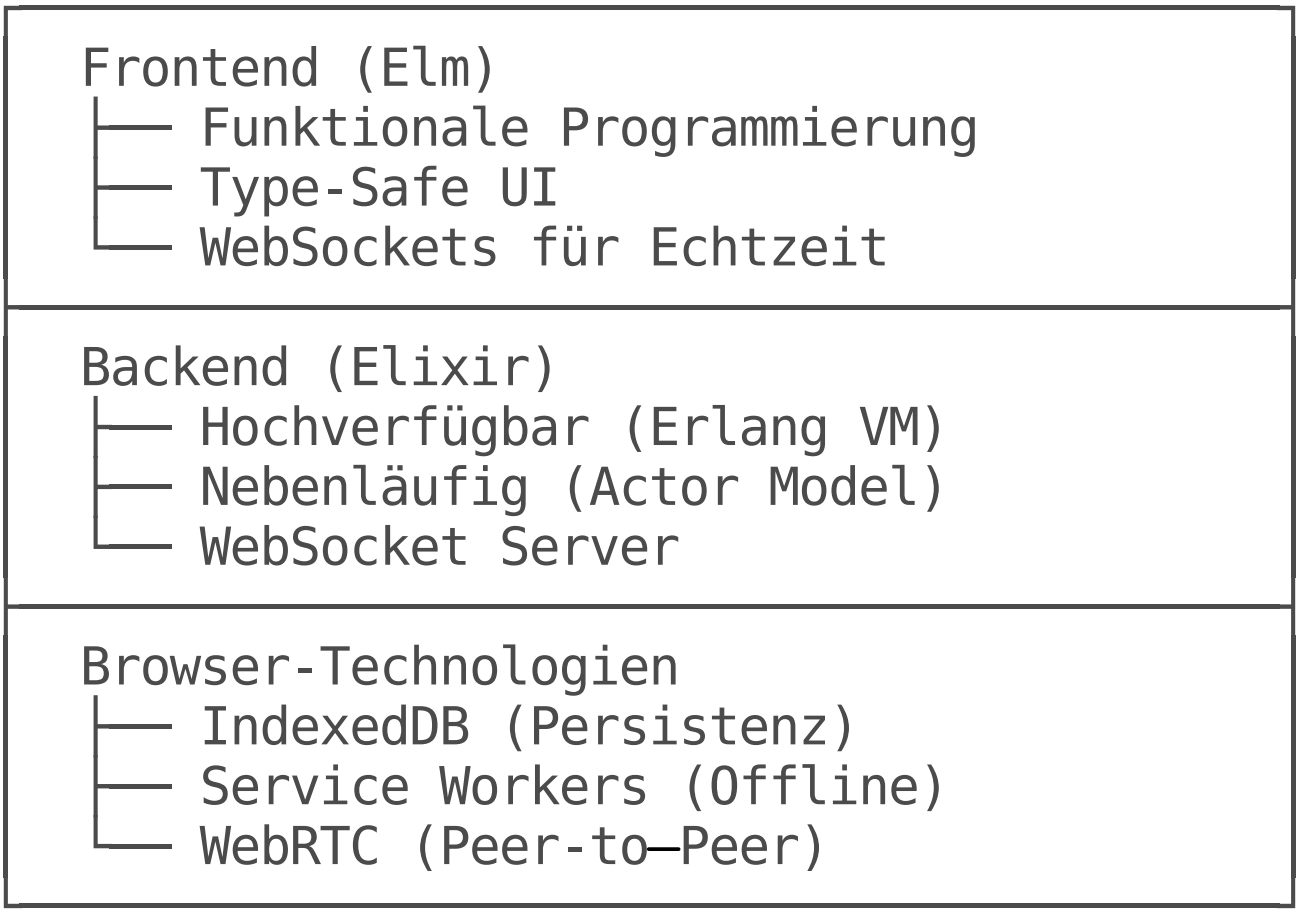
Von zweitausendsiebzehn bis zwanzig arbeitete ich am BMBF-geförderten Projekt Industrial eLab. Ziel war es, Remote Labs für die Ingenieurausbildung zu entwickeln – Studierende sollten zeit- und ortsunabhängig mit realer Hardware wie Robotern und Mikrocontrollern arbeiten können.

Projekt-Steckbrief:

Aspekt	Details
Projekt	BMBF-gefördertes Remote Lab (2017–2020)
Partner	Otto-von-Guericke-Universität Magdeburg, Hochschule Magdeburg-Stendal
Technologie	Elixir Backend, Elm Frontend, WebSockets
Ziel	Zeit- und ortsunabhängiger Zugriff auf reale Hardware
Förderung	895.890 EUR
Link	Industrial eLab Projektseite

Aus diesem Projekt habe ich drei zentrale Erkenntnisse mitgenommen. Erstens: Der Browser ist ein vollwertiges Betriebssystem – moderne Web-Technologien wie WebRTC, IndexedDB und Service Workers ermöglichen komplexe Anwendungen. Zweitens: Progressive Web Apps mit IndexedDB und Caching ermöglichen Offline-Fähigkeit und Persistenz. Drittens: Die didaktische Herausforderung – Wie gestalten wir adaptive Lernumgebungen, die Studierende beim Problemlösen unterstützen?

Technologie-Stack:



Und genau aus diesem Projekt entstand die Idee für LiaScript – eine Markdown-basierte Beschreibungssprache für interaktive Lehr- und Lernmaterialien. Die Vision: Lehrinhalte sollten so einfach wie Markdown sein, aber so mächtig wie moderne Web-Apps.

Entstehung von LiaScript

> **Idee:** Markdown + Interaktivität + Browser-Power = LiaScript

Zentrale Frage:

Wie können Lehrende ohne Programmierkenntnisse interaktive, multimediale Kurse erstellen?

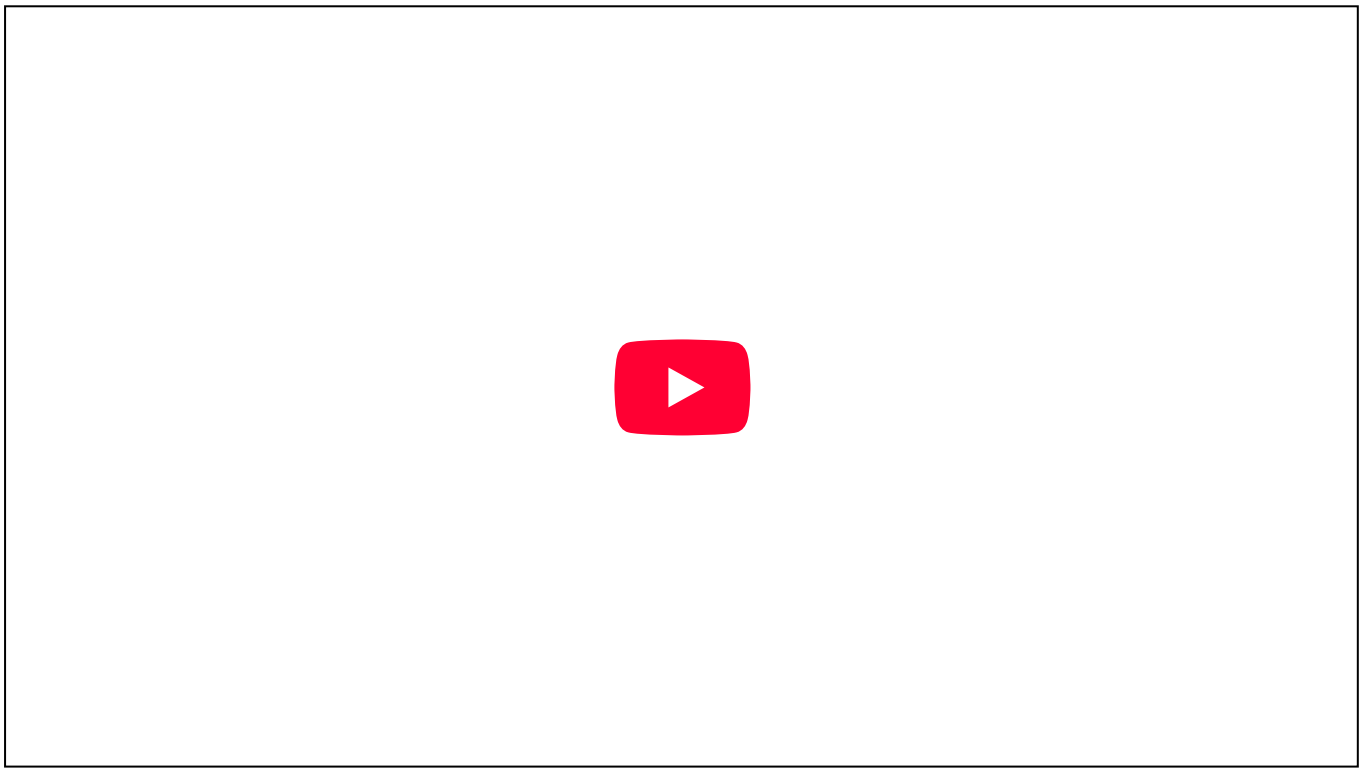
Antwort: Eine erweiterte Markdown-Syntax, die direkt im Browser interpretiert wird.

LiaScript – Interaktive OER im Browser

LiaScript ist heute ein Open-Source-Projekt für interaktive Kurse in Markdown. Die Vision: Lehrende erstellen Kurse als einfache Textdateien, zum Beispiel auf GitHub, ohne Build-Steps oder Content-Management-Systeme.

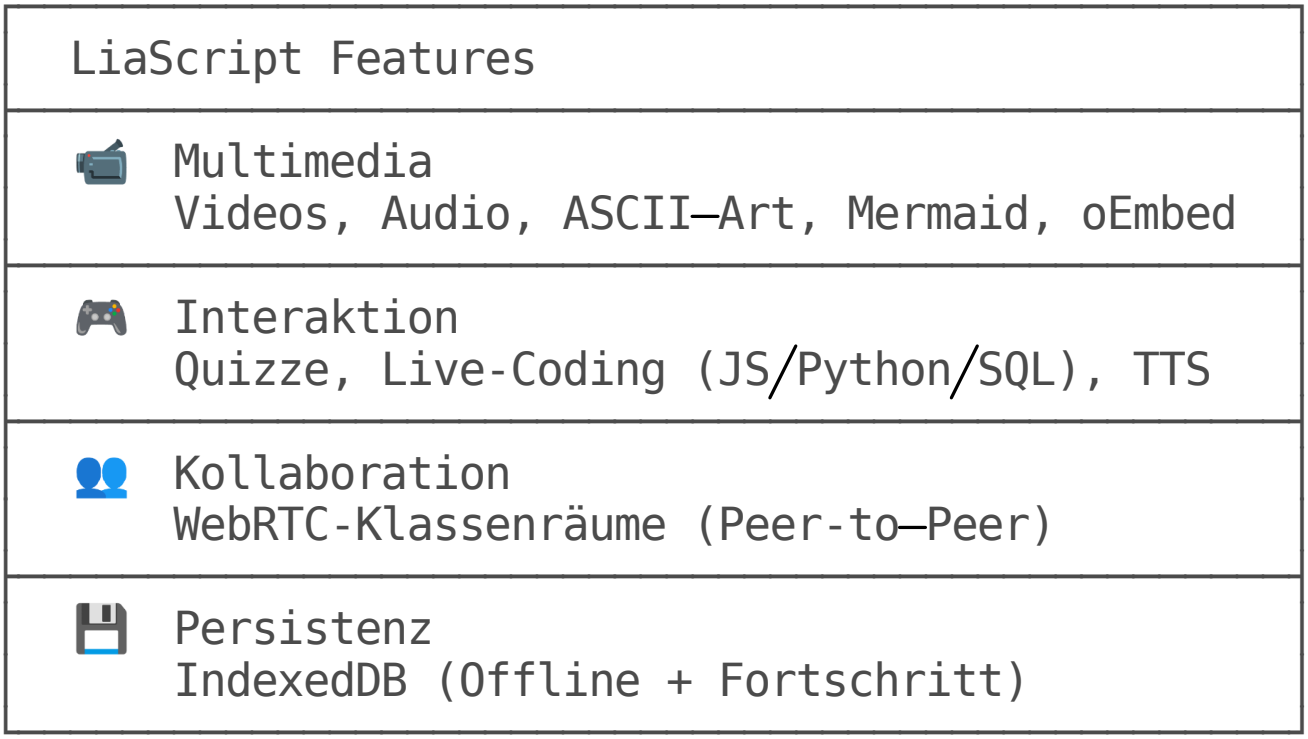
Projekt-Steckbrief:

Aspekt	Details
Projekt	Open-Source Markdown-Interpreter für interaktive Kurse
Vision	Kurse als Textdateien (z. B. auf GitHub), ohne Build-Steps
Features	Multimedia, Quizze, Live-Coding, TTS, Kollaboration
Technologie	IndexedDB, Service Workers, WebRTC
Link	LiaScript Homepage



LiaScript bietet eine Vielzahl von Features: Multimedia wie Videos, Audio, ASCII-Diagramme, Mermaid und LaTeX. Interaktion durch Quizze, Live-Coding in JavaScript, Python oder SQL, und Text-to-Speech. Kollaboration über WebRTC-basierte Klassenräume mit Peer-to-Peer-Kommunikation. Und Persistenz durch IndexedDB für Offline-Fähigkeit und Fortschritt speichern.

Feature-Übersicht:



Welche Technologien stecken dahinter? IndexedDB für lokale Datenhaltung – eine NoSQL-Datenbank direkt im Browser. Service Workers für Offline-Caching – Kurse funktionieren auch ohne Internetverbindung. Und WebRTC für Peer-to-Peer-Kommunikation in kollaborativen Klassenräumen.

Technologie-Stack:

Technologie	Zweck	Datenbankbezug
IndexedDB	Lokale Datenhaltung	NoSQL Object Store im Browser
Service Workers	Offline-Caching	Persistent Storage API
WebRTC	Peer-to-Peer	Dezentrale Datensynchronisation

Wichtigste Erkenntnisse:

- **Browser-basierte Datenbanken** (IndexedDB) sind mächtig, aber anders als traditionelle SQL-DBs
- **NoSQL im Browser:** Object Stores, Key-Value Zugriffe, Indexierung
- **Trade-offs:** Flexibilität vs. strukturierte Abfragen

Selbständiger Elm-Entwickler – Linked Data & SPARQL

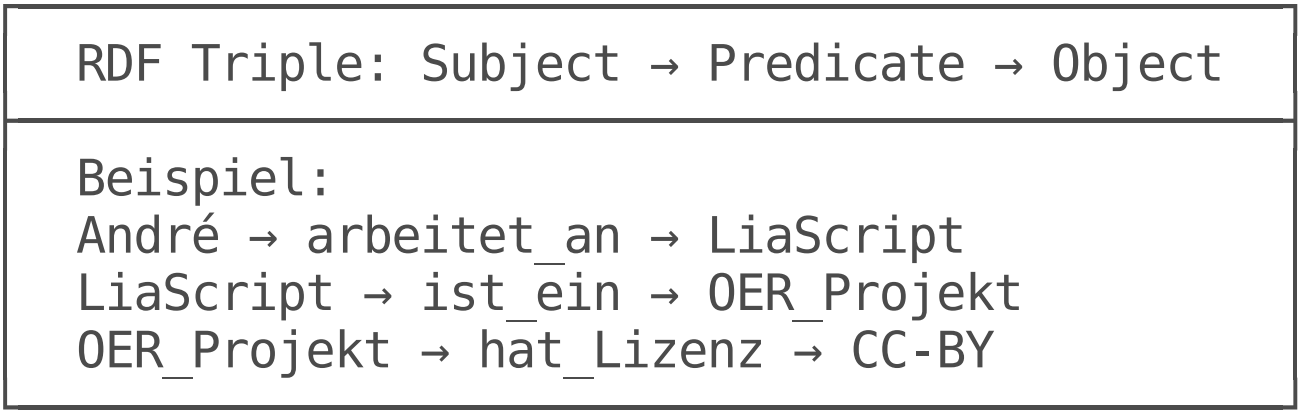
Nach dem Industrial eLab-Projekt war ich als selbständiger Elm-Entwickler tätig und arbeitete an Linked Data Anwendungen. Dabei kam ich mit dem Semantic Web und SPARQL in Kontakt.

Projekt-Steckbrief:

Aspekt	Details
Projekt	Web-Entwicklung für Linked Data Anwendungen
Technologie	Elm Frontend, Semantic Web, SPARQL
Datenbank	RDF-Stores (Triple Stores) für Wissensgraphen

Was habe ich dabei gelernt? Graphdatenbanken sind ideal für vernetzte, semantische Daten. SPARQL ist SQL für Graphen – aber mit eigenen Herausforderungen. RDF und Linked Data sind flexibel, aber komplex zu modellieren.

RDF Triple Store Konzept:



SPARQL Query Beispiel: Datenbank-Erfinder und ihre Geburtstage

```
# source: https://dbpedia.org/sparql

PREFIX dbo:  <http://dbpedia.org/ontology/>
PREFIX dbr:  <http://dbpedia.org/resource/>
PREFIX dbc:  <http://dbpedia.org/resource/Category:>
PREFIX dct:  <http://purl.org/dc/terms/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT DISTINCT ?person ?name ?birthDate ?work ?workLabel
WHERE {
  # a "database" work: software whose category contains “database”
  ?work a dbo:Software ;
        dct:subject ?cat .
  FILTER(CONTAINS(LCASE(STR(?cat)), "database"))

```

```
  # link people to the work via common creator-like properties
  ?person a dbo:Person ;
          (dbo:author|dbo:developer|dbo:designer|dbo:creator|dbo
            :notableWork|dbo:knownFor) ?work ;
          foaf:name ?name .

  OPTIONAL { ?person dbo:birthDate ?birthDate . }
  OPTIONAL { ?work rdfs:label ?workLabel . FILTER(LANG(?workLabel) =
    }
  FILTER(LANG(?name) = "en")
}
ORDER BY ?name
LIMIT 20
```

```
?person                                                 ?name
?birthDate                                              ?work
?workLabel
------------------------------------------------------------------------
------------------------------------------------------------------------
------------------------------------------------------------------------
----------------------------------------
http://dbpedia.org/resource/Adam_Kilgarriff        Adam Kilgarriff
1960-02-12
http://dbpedia.org/resource/Sketch_Engine          Sketch Engine
http://dbpedia.org/resource/Avi_Kivity             Avi Kivity
http://dbpedia.org/resource/ScyllaDB               ScyllaDB
http://dbpedia.org/resource/Brian_Aker             Brian Aker
1972-08-04
http://dbpedia.org/resource/MySQL                  MySQL
http://dbpedia.org/resource/Brian_Aker             Brian Aker
1972-08-04
http://dbpedia.org/resource/Drizzle_(database_ser… Drizzle (database
server)
http://dbpedia.org/resource/Brian_Aker             Brian Aker
1972-08-04
http://dbpedia.org/resource/Memcached              Memcached
http://dbpedia.org/resource/D._Richard_Hipp        D. Richard Hipp
1961-04-09
http://dbpedia.org/resource/SQLite                 SQLite
http://dbpedia.org/resource/Daniel_Weinreb         Daniel L. Weinreb
1959-01-06
http://dbpedia.org/resource/ObjectStore            ObjectStore
http://dbpedia.org/resource/George_Armitage_Miller George Armitage
Miller                              1920-02-03
http://dbpedia.org/resource/WordNet                WordNet
http://dbpedia.org/resource/Gjergji_Kasneci        Gjergji Kasneci
http://dbpedia.org/resource/YAGO_(database)        YAGO (database)
http://dbpedia.org/resource/Jaan_Tallinn           Jaan Tallinn
1972-02-14
http://dbpedia.org/resource/Kazaa                  Kazaa
http://dbpedia.org/resource/Janus_Friis            Janus Friis
1976-06-26
http://dbpedia.org/resource/Kazaa                  Kazaa
http://dbpedia.org/resource/Janus_Friis            Janus Friis
1976-06-26
http://dbpedia.org/resource/Rdio                   Rdio
http://dbpedia.org/resource/Jeff_Dean              Jeff Dean
```

```
http://dbpedia.org/resource/Bigtable                Bigtable
http://dbpedia.org/resource/Jim_Starkey             Jim Starkey
1949-01-06
http://dbpedia.org/resource/Falcon_(storage_engin… Falcon (storage
engine)
http://dbpedia.org/resource/Kaj_Arnö                Kaj Arnö
http://dbpedia.org/resource/MySQL                   MySQL
http://dbpedia.org/resource/Kaj_Arnö                Kaj Arnö
http://dbpedia.org/resource/MariaDB                 MariaDB
http://dbpedia.org/resource/Kevin_P._Ryan           Kevin P. Ryan
1963-10-12
http://dbpedia.org/resource/MongoDB                 MongoDB
http://dbpedia.org/resource/Martin_L._Kersten       Martin Kersten
1953-10-25
http://dbpedia.org/resource/MonetDB                 MonetDB
http://dbpedia.org/resource/Michael_Stonebraker     Michael Stonebraker
1943-10-11
http://dbpedia.org/resource/Illustra                Illustra
http://dbpedia.org/resource/Michael_Stonebraker     Michael Stonebraker
1943-10-11
http://dbpedia.org/resource/PostgreSQL              PostgreSQL
```

Wichtigste Erkenntnisse:

- **Graphdatenbanken** sind ideal für vernetzte, semantische Daten
- **SPARQL** ist SQL für Graphen – aber mit eigenen Herausforderungen
- **RDF/Linked Data:** Flexibel, aber komplex zu modellieren

CrossLab & Edrys-Lite – Peer-to-Peer Remote Labs

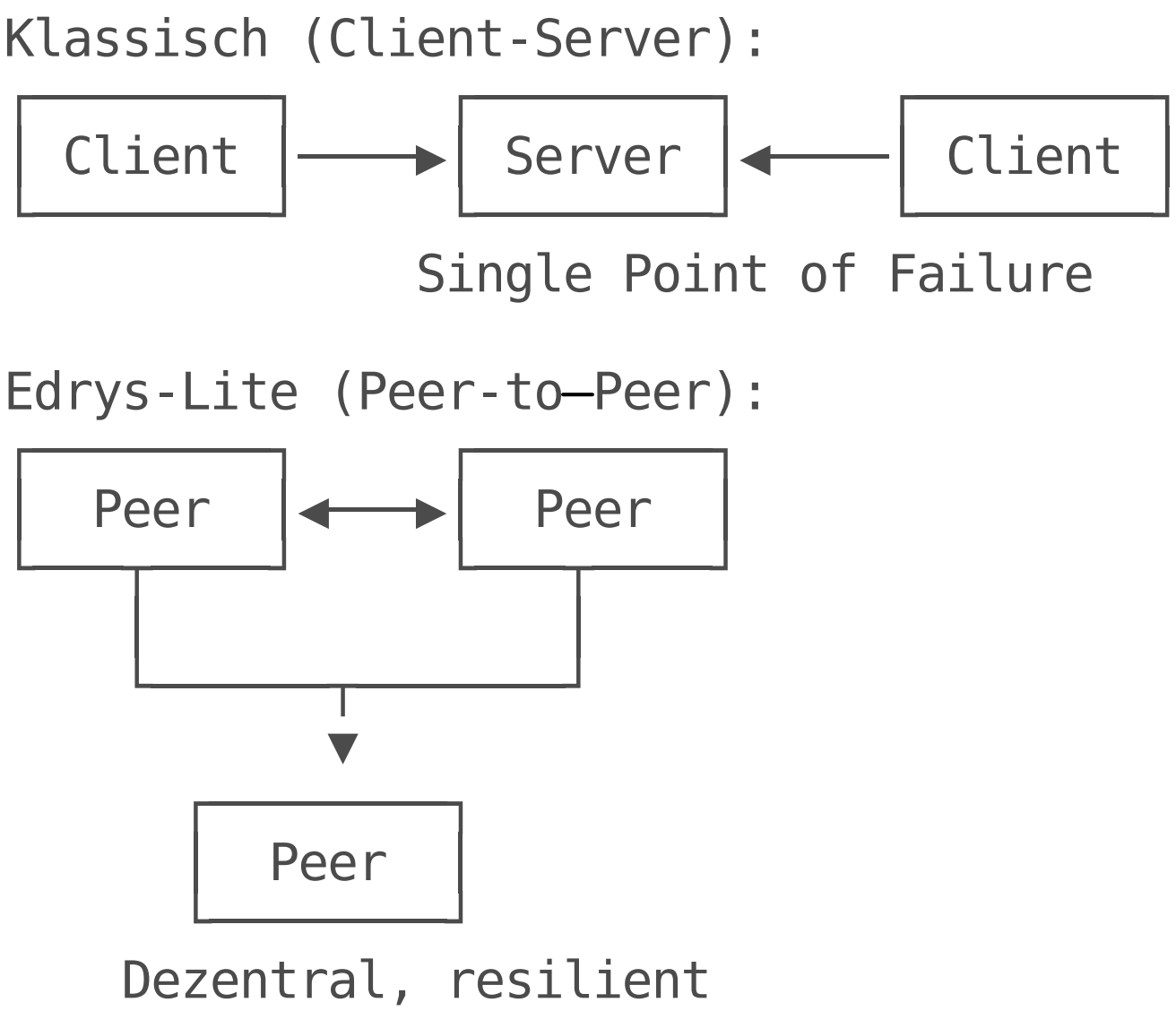
In Freiberg hatte ich dann die Möglichkeit, LiaScript in verschiedenen Projekten zu erweitern und Peer-to-Peer-Mechanismen zu untersuchen. Daraus entstand Edrys-Lite – ein dezentrales Peer-to-Peer-System zum Teilen von Remote Labs.

Projekt-Steckbrief:

Aspekt	Details
Projekt	Dezentrales Peer-to-Peer Remote Lab System
Technologie	WebRTC für direkte Browser-zu-Browser-Kommunikation
Ziel	Lehre und Labore dezentral organisieren
Besonderheit	Jeder Browser ist ein potenzieller „Server“
Link	Edrys-Lite

Die wichtigste Erkenntnis? Dezentrale Architekturen mit Peer-to-Peer reduzieren Abhängigkeiten von zentralen Servern. Browser-Technologien wie WebRTC, WebSockets und IndexedDB ermöglichen verteilte Anwendungen. Aber es gibt Trade-offs: Konsistenz versus Verfügbarkeit – das CAP-Theorem in der Praxis.

Architektur-Vergleich:



Trade-off (CAP-Theorem):

- **Consistency:** Schwierig bei P2P (Eventual Consistency)
- **Availability:** Hoch (kein Single Point of Failure)
- **Partition Tolerance:** Hoch (funktioniert bei Netzwerkausfällen)

Wichtigste Erkenntnisse:

- **Dezentrale Architekturen:** Peer-to-Peer reduziert Abhängigkeiten
- **Browser-Technologien:** WebRTC, WebSockets, IndexedDB ermöglichen verteilte Apps
- **Trade-offs:** Konsistenz vs. Verfügbarkeit (CAP-Theorem)

Meine Motivation für diese Vorlesung

Interaktive OER als Lernformat

Ich bin überzeugt: Interaktive Materialien mit Quizzen, Live-Coding und Visualisierungen fördern das Verständnis. LiaScript ermöglicht es, direkt im Browser mit Datenbanken zu arbeiten – DuckDB, SQLite, IndexedDB. Kein Setup, kein Installieren – nur Browser öffnen und loslegen.

Vorteile interaktiver OER:

Aspekt	Klassische Vorlesung	Interaktive OER (LiaScript)
Setup	Installation erforderlich	Browser genügt
Feedback	Verzögert	Sofort (Quizze, Live-Code)
Exploration	Begrenzt	Unbegrenzt (eigene Queries)
Persistenz	Notizen auf Papier	Automatisch im Browser
Kollaboration	Schwierig	WebRTC-Klassenräume

> **In dieser Vorlesung:** Sie arbeiten direkt mit DuckDB, SQLite, IndexedDB – alles im Browser!

Spec-Driven Development mit GitHub Copilot

Ich nutze GitHub Copilot als Co-Autor für diese Vorlesung. Der Ansatz: Spec-Driven Development – ich definiere Struktur, Lernziele und Didaktik, Copilot hilft bei den Inhalten. Das Experiment: Wie kann KI Lehrende bei der Erstellung von OER unterstützen?

Workflow:

1. Outline erstellen (Titel, Zielgruppe, Lernziele)
2. Didactics definieren (Persona, Stil, Methoden)
3. Agenda strukturieren (Sessions, Meilensteine)
4. Co-Authoring mit Copilot (Inhalte, Beispiele, Übungen)
5. Iteration & Feedback (Studierende, Selbstreflexion)

Forschungsfrage:

> Kann KI den Prozess der OER-Erstellung beschleunigen, ohne Qualität zu verlieren?

Aktuelle Rolle

Ich bin Dozent für Datenbanken im Wintersemester zweitausendfünfundzwanzig-sechsundzwanzig an der TU Bergakademie Freiberg. Mein Ziel: Eine praxisnahe, vergleichende Vorlesung mit Fokus auf Browser-basierte Technologien.

Vorlesungsziele:

Diese Vorlesung ist anders: Browser-first, interaktiv, OER, spec-driven mit Copilot.

populate

```sql
-- Erstelle Vorlesungsziele-Tabelle
DROP TABLE IF EXISTS course_objectives;
CREATE TABLE course_objectives (
  id INTEGER PRIMARY KEY,
  category TEXT NOT NULL,
  objective TEXT NOT NULL,
  keywords TEXT NOT NULL,
  description TEXT
);

-- Füge Vorlesungsziele ein
INSERT INTO course_objectives (id, category, objective, keywords,
  description) VALUES
  (1, 'Paradigmen', 'Paradigmen verstehen', 'File, KV, Document, Colu
    Relational, Graph',
   'Verschiedene Datenbank-Paradigmen kennen und deren Einsatzszenari
     verstehen'),
  (2, 'Relationale DB', 'Relationale Datenbanken meistern', 'SQL,
    Normalisierung, Transaktionen, Indexe',
   'Tiefes Verständnis relationaler Systeme mit praktischer SQL-Kompe
     ),
  (3, 'Praxis', 'Praktisch arbeiten', 'DuckDB, SQLite, IndexedDB, Bro

   'Hands-on Erfahrung mit modernen Browser-basierten Datenbanksystem
  (4, 'Bewertung', 'Vergleichen & bewerten', 'ACID, CAP, Trade-offs',
   'Kritische Analyse und Bewertung verschiedener Datenbankansätze'),
  (5, 'Anwendung', 'Anwenden', 'Polyglot Persistence, Architektur',
   'Praktische Anwendung in einem durchgängigen Projekt');

-- Zeige alle Vorlesungsziele mit ihren Kategorien
SELECT
  id,
  category,
  objective,
  keywords,
  description
FROM course_objectives
ORDER BY id;
```

```
DROP TABLE IF EXISTS course_objectives
DROP OK

CREATE TABLE course_objectives (
  id INTEGER PRIMARY KEY,
  category TEXT NOT NULL,
  objective TEXT NOT NULL,
  keywords TEXT NOT NULL,
  description TEXT
)
CREATE OK

INSERT INTO course_objectives (id, category, objective, keywords,
description) VALUES
  (1, 'Paradigmen', 'Paradigmen verstehen', 'File, KV, Document,
Column, Relational, Graph',
   'Verschiedene Datenbank-Paradigmen kennen und deren Einsatzszenarien
verstehen'),
  (2, 'Relationale DB', 'Relationale Datenbanken meistern', 'SQL,
Normalisierung, Transaktionen, Indexe',
   'Tiefes Verständnis relationaler Systeme mit praktischer SQL-
Kompetenz'),
  (3, 'Praxis', 'Praktisch arbeiten', 'DuckDB, SQLite, IndexedDB,
Browser',
   'Hands-on Erfahrung mit modernen Browser-basierten
Datenbanksystemen'),
  (4, 'Bewertung', 'Vergleichen & bewerten', 'ACID, CAP, Trade-offs',
   'Kritische Analyse und Bewertung verschiedener Datenbankansätze'),
  (5, 'Anwendung', 'Anwenden', 'Polyglot Persistence, Architektur',
   'Praktische Anwendung in einem durchgängigen Projekt')
Query OK, 5 rows affected (last id = 5)

SELECT
  id,
  category,
  objective,
  keywords,
  description
FROM course_objectives
ORDER BY id
```

	id	category	objective	keywords	description
0	1	Paradigmen	Paradigmen verstehen	File, KV, Document, Column, Relational, Graph	Verschiedene Datenbank-Paradigmen kennen und deren Einsatzszenarien verstehen
1	2	Relationale DB	Relationale Datenbanken meistern	SQL, Normalisierung, Transaktionen, Indexe	Tiefes Verständnis relationaler Systeme mit praktischer SQL-Kompetenz
2	3	Praxis	Praktisch arbeiten	DuckDB, SQLite, IndexedDB, Browser	Hands-on Erfahrung mit modernen Browser-basierten Datenbanksystemen
3	4	Bewertung	Vergleichen & bewerten	ACID, CAP, Trade-offs	Kritische Analyse und Bewertung verschiedener Datenbankansätze
4	5	Anwendung	Anwenden	Polyglot Persistence, Architektur	Praktische Anwendung in einem durchgängigen Projekt

Diese Tabelle zeigt alle fünf Lernziele der Vorlesung strukturiert als Datenbank. Probieren Sie eigene Abfragen aus!

Interaktive Exploration:

```sql
-- Beispiel: Suche nach Paradigmen-bezogenen Zielen
SELECT objective, keywords
FROM course_objectives
WHERE keywords LIKE '%Relational%';
```

```
SELECT objective, keywords
FROM course_objectives
WHERE keywords LIKE '%Relational%'
```

	objective	keywords
0	Paradigmen verstehen	File, KV, Document, Column, Relational, Graph

Nächste Schritte

Nach dieser Vorstellung starten wir in Session eins mit der eigentlichen Vorlesung.

Session 1 Vorschau:

1. **Was sind Datenbanken?** – Grundbegriffe, Paradigmen, Einsatzszenarien
2. **Erste Hands-on-Beispiele** – CSV, JSON, IndexedDB im Browser
3. **DIKW-Pyramide** – Daten, Information, Wissen, Weisheit

> **Bereit für die Reise?** Lassen Sie uns gemeinsam Datenbanken „unlocken“! 🎓

Willkommen zur Vorlesung „Databases Unlocked: A Beginner's Journey“! 🚀