



openHPI: **Web-Technologien** Persistenz-Layer und Datenbanken

Prof. Dr. Christoph Meinel

Hasso-Plattner-Institut

Universität Potsdam

Datenbank-basierte Webanwendungen

HTTP ist zustandsloses Protokoll

- Aber: Applikationen befinden sich immer in einem bestimmten Zustand
 - ➔ Daten des Zustands müssen vorgehalten werden
 - Clientseitig: Storage-Mechanismen beim Browser oder im Arbeitsspeicher
 - Serverseitig: Datenbanken und Sessions

Unterschiedliche Arten von Datenbanken:

- Relationale Datenbanken, z.B. MySQL, PostgreSQL, Oracle, MS SQL, ...
- NoSQL-Datenbanken
 - Graph Stores, z.B. Neo4J, Giraph, ...
 - Document Stores, z.B. MongoDB, Elastic, OrientDB, CouchDB...
 - Key-Value Stores, Redis, BerkeleyDB, ...
 - Column Stores, Hbase, Cassandra, Hadoop...
 - ...

Relationale Datenbanken (1/3)

■ Tabellenstruktur

- Reihe – Repräsentiert einen Datensatz (z.B. User)
- Spalte – ein Attribut der Datensätze, z.B. Name, Vorname, E-Mail, ...
- Tabelle – Menge von Datensätzen mit denselben Attributen

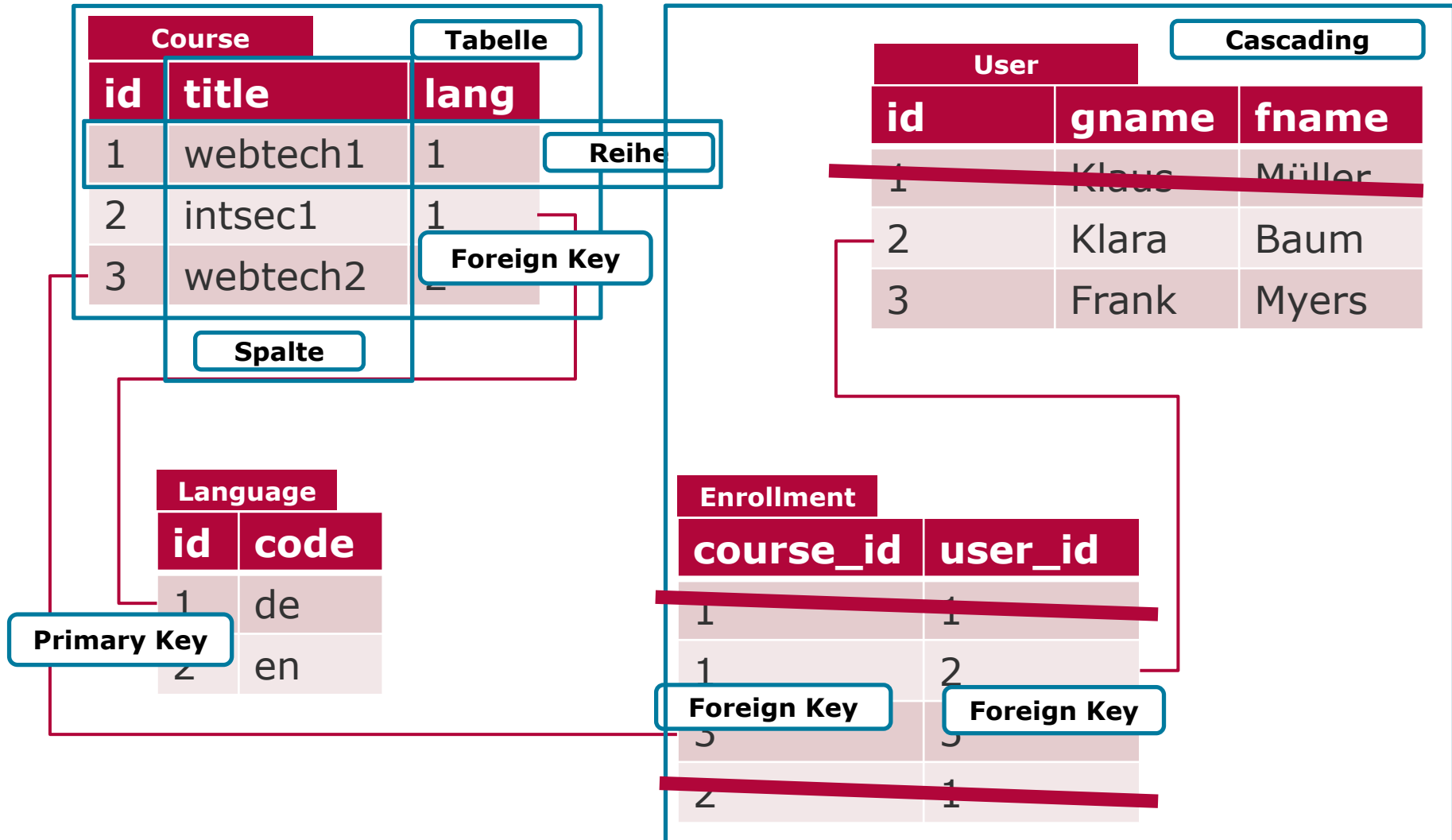
■ Jeder Datensatz in Tabelle hat eindeutigen Schlüssel: **Primary Key**

- ermöglicht eindeutige Identifikation des Datensatzes
- kann aus mehreren Attributen zusammengesetzt werden
- kann in anderen Tabellen referenziert werden, um Bezüge zwischen Tabellen herzustellen (Foreign Key)

■ Cascading

- Löschen aller abhängigen Daten (in anderen Tabellen), wenn Datensatz gelöscht wird

Relationale Datenbanken (2/3)



Relationale Datenbanken (3/3)

■ Indizes

- vergleichbar mit Index in einem Buch
- ermöglichen direkten Zugriff auf Datensatz, anstatt Tabelle Reihe per Reihe zu durchsuchen
- deutlicher Performance-Gewinn

■ Transaktionen

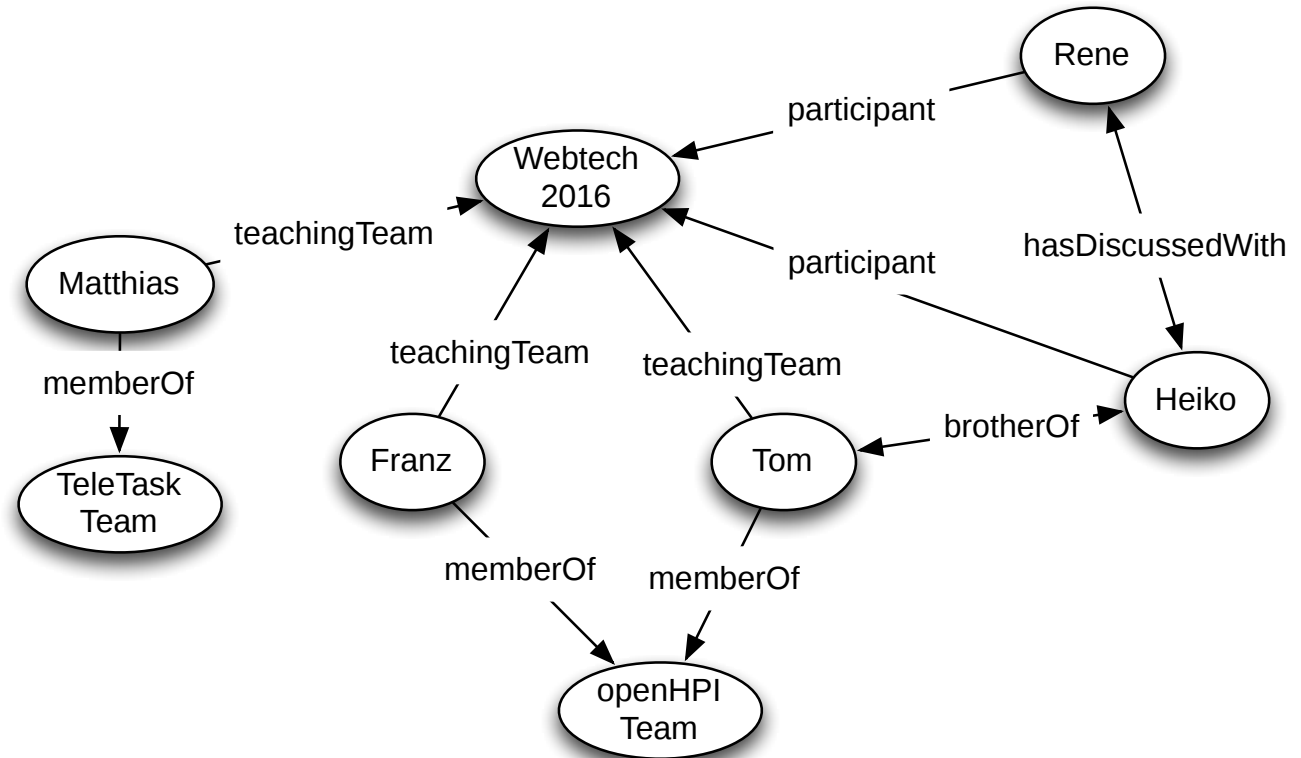
- zusammenfassen mehrerer Operationen
- Operation einer Transaktion schlägt fehl → Rollback

■ **Stored Procedures:** In Datenbank gespeicherter, ausführbarer Code

■ **Query-Sprache:** SQL (Structured Query Language)

- SELECT, UPDATE, DELETE, INSERT
- JOIN, UNION, INTERSECT, ...
- WHERE, ORDER BY, LIMIT

- Basieren auf Graph-Struktur → Knoten (Nodes), Kanten (Edges)



- Query-Sprachen: Gremlin, SPARQL, Cypher
- Anwendung: Soziale Netze, Semantic Web, etc.

- Speicherung komplexer Daten, Arrays, Objekte, Key-Value-Paare
- Speicherung der Daten häufig im JSON Format

Dokument

```
{ "user":  
  {  
    "id": "a719f263-5ddd-4f96-a337-3883f79f1dee",  
    "personal_data": {  
      "last_name": "Meier",  
      "first_name": "Klaus",  
      "email": "klaus.meier@example.com"  
    },  
    "courses": [  
      "webtech1",  
      "webtech2",  
      "intsec1",  
      "java1"  
    ],  
    "kyu": "yellow"  
  }  
}
```

- Keine Query Sprache – Zugriff über OOP APIs
- Alternative: XML-Datenbanken – Query-Sprache: XPath

■ Relationale Datenbanken...

- ... unflexibler aufgrund ihrer fixen Datenstruktur
- ... mehr Möglichkeiten zur Sicherung der Datenkonsistenz
- ... schwierige horizontale Skalierbarkeit
- Mischformen (z.B. JSONB-Datentyp in Postgres)



■ NoSQL-Datenbanken...

- ... vereinfachen agile Entwicklung durch flexible Datenstruktur
- ... schlechtere Transaktionssicherheit
- ... einfachere horizontale Skalierbarkeit
- Übersicht: <http://nosql-database.org/>



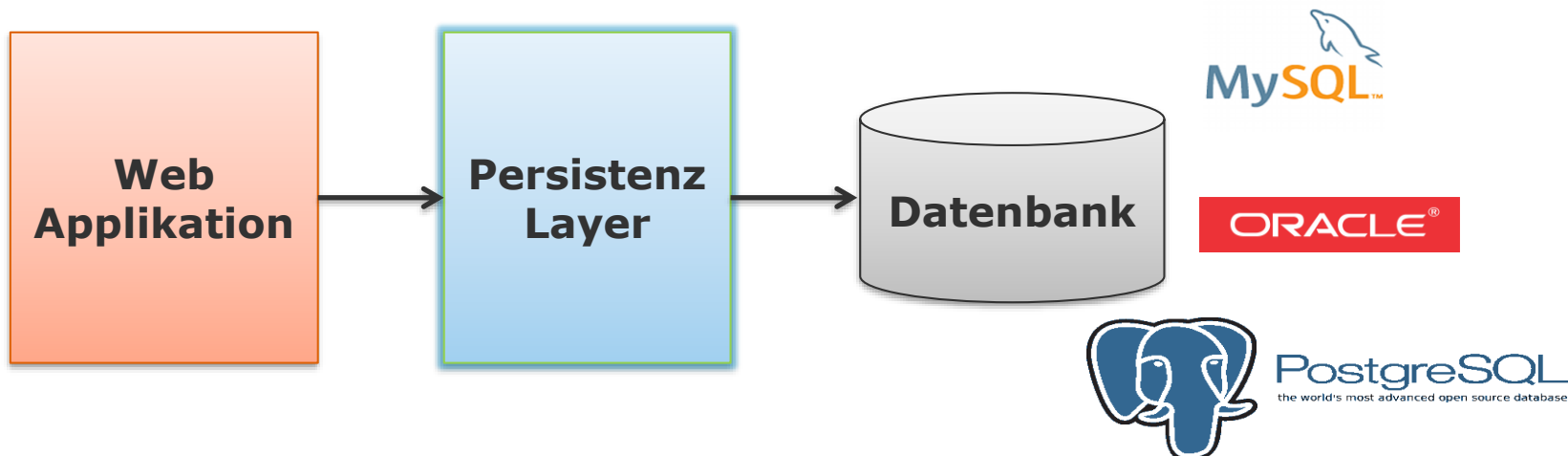
Empfehlung: Nutzen Sie das zum Zweck passende Werkzeug

In-Memory-Datenbanken → openHPI-Kurs im September

Persistenz-Layer (1/2)

Persistenz-Layer:

- Abstraktionsebene zwischen Datenbank und Applikation
- Gründe:
 - Vendor-Lock-In vermeiden: Abstraktion erleichtert Austausch der Datenbank (Maintainability)
 - Schwieriger: Umstieg von einem DB-Typ auf anderen
 - Komfortablerer Zugriff auf Daten für Entwickler (Produktivität)
 - Daten-Zugriffsoptimierung (Performance)



Persistenz-Layer (2/2)

Moderne Web-Applikationen werden in **objekt-orientierten** Sprachen entwickelt

- Anwendung von Konzepten der objekt-orientierten Programmierung verbessert
 - Kapselung & Wiederverwendbarkeit & Abstraktion
- Object-Relational-Mapping – **ORM**:
 - Abbildung von Objekten auf Datensätze
- ORM-Tools gibt es für alle verbreiteten Programmiersprachen – oft integriert in Persistenz-Layer der Frameworks
 - Ruby: ActiveRecord / Rails
 - Python: Django
 - Java: Hibernate
 - .NET: NHibernate

Caching

- Datenbank und Applikation laufen typischerweise auf verschiedener Hardware
- Performanceverlust
 - limitierte Geschwindigkeit des Netzwerks
 - komplexe Datenbank-Queries
- Performancesteigerung durch Caching der Daten auf Applikationsserver
- Verwendete Technologien
 - Key-Value-Store, z.B. Redis
 - Caching im Arbeitsspeicher, z.B. Memcached