Datenbanken I & II

Vorlesung DHBW

Cedric Weber, IT Consultant

22.08.2023 | confidentiality level: none

Vorlesung Organisatorisches

Kurze Vorstellung

Klausur: 31.10.23, 14-16 Uhr

Vorlesungstermine:

Di, 22.08., 14-19

Mi, 23.08., 14-19

Di, 12.09., 14-19

Di, 19.09., 14-19

Di, 26.09., 14-19

Mi, 04.10., 14-19

Di, 10.10., 14-19

Di, 17.10., 14-19

Di, 24.10., 14-19

Datenbanken I + II: Google Drive Share

LucidSpark Board

Vorlesungsagenda

1. Tune-In
2. Vorlesung / Übungen
3. Retro

Es wird **keine Hausaufgaben** geben, die Vorlesungen bauen jedoch aufeinander auf

Cedric Weber

Kurze Vorstellung

B. Sc. Wirtschaftsinformatik (WWI16AMC)

⇒ seit 2016 bei Roche Diagnostics GmbH in Mannheim

⇒ seit 2019 Internal IT Consultant für Supply Chain

- DB Admin & Owner für WMS Solution
- Projekte für Systemintegration & Neueinführungen
- Product Owner Analytics Data Mesh Supply Chain

“Get-to-know”

5min Prep | 20min Vorstellung

Bitte die drei folgenden Fragen auf dem Lucid Board individuell beantworten:

1. Vorstellung der Person (Unternehmen, Interessen, Interessantes, etc.)
2. Welchen Kontakt hattest du bereits mit Datenbanken (Arbeit & Privat)
3. Wie glaubst du, kann dir das Thema Datenbanken in deinem zukünftigen Beruf helfen

Arbeitet bitte im zugewiesenen Space auf https://lucid.app

Den **Lucid Space**, werden wir im Laufe der Vorlesung immer mal wieder verwenden. Ein kostenloser Gastzugang ohne Anmeldung ist möglich. **Es ist daher nicht nötig sich zu registieren!**

Vorlesung Datenbanken I & II

Was ist vorgegeben / Was machen wir

I

Datenbanken I

1. Grundlagen
2. Relationen
3. ER-Modell
4. Datenbankkonzeption
5. SQL

II

Datenbanken II

1. Einbettung von Datenbanken in Programmlogik (Java)
2. Analytics mit Datenbanken in Python

Verhältnis Datenbanken I & Datenbanken II: 70/30 ⇒ Auch in der Klausur :)

Was bringt euch diese Vorlesung?

Zukünftige Jobprofile für Wirtschaftsinformatiker

1. **Business Analysten/Consultants:** Sie übersetzen Geschäftsanforderungen in technische Anforderungen. Datenbankwissen hilft bei der Lösung datenbezogener Probleme.
2. **Datenbankadministrator:** Dieser Job benötigt Wissen über Datenbankmanagement, Datenmodellierung und SQL. Eine Datenbankvorlesung liefert dafür die Grundlage.
3. **IT-Projektmanager:** Sie leiten technische und nicht-technische Teams. Datenbankkenntnisse erleichtern das Verständnis von technischen Herausforderungen.
4. **Data Scientist/Data Analyst:** Sie sammeln, analysieren und interpretieren Daten für Geschäftsentscheidungen. Datenbankwissen ist hierfür essenziell.
5. **ERP-Spezialist:** Da viele ERP-Systeme auf Datenbanken basieren, ist ein gutes Datenbankverständnis wichtig für effiziente Nutzung und Anpassung dieser Systeme.

+

Nutzung des Basiswissens für private Projekte

Brainstorming

10min Prep | 15min Pause | 15min Vorstellung

1. Was sind Datenbanken?
2. Wo werden Datenbanken angewandt?
3. Was sind Vorteile/Chancen von Datenbanken?
4. Was sind Nachteile/Risiken von Datenbanken?
5. Was sind Begriffe, die euch interessieren?

Arbeitet bitte im zugewiesenen Space auf https://lucid.app

Den **Lucid Space**, werden wir im Laufe der Vorlesung immer mal wieder verwenden. Ein kostenloser Gastzugang ohne Anmeldung ist möglich. **Es ist daher nicht nötig sich zu registieren!**

Datenbanken I: Grundlagen

1. Anforderungen an moderne Datenbanken
2. Historie
3. DBS & DBMS
4. Zugriffsmöglichkeiten

DHBW
Duale Hochschule
Baden-Württemberg

Datenbanken I: Anforderungen an moderne Datenbanken

Definition Datenbanken

Datenbanken I: Grundlagen

DHBW
Duale Hochschule
Baden-Württemberg

Datenbank

„**Selbstständige**, auf **Dauer** und flexiblen und sicheren Gebrauch ausgelegte Datenorganisation, die sowohl eine Datenbasis als auch eine zugehörige Datenverwaltung (DBMS) umfasst. Eine Datenbank dient dazu, eine große Menge von Daten **strukturiert** zu speichern und zu **verwalten**."

Anforderungen an moderne Datenbanken

- ✓ Leistungsfähigkeit und Geschwindigkeit
- ✓ Skalierbarkeit
- ✓ Datensicherheit
- ✓ Zuverlässigkeit und Verfügbarkeit
- ✓ Sicherstellung der Datenintegrität
- ✓ Ermöglichung konkurrierender Zugriffe
- ✓ Interoperabilität
- ✓ Strukturelle Änderungen

Leistungsfähigkeit und Geschwindigkeit

Datenbanken I: Grundlagen

Datenbanken müssen in der Lage sein, große Mengen von Daten schnell und effizient zu verarbeiten, um Abfragen in Echtzeit zu ermöglichen.

Metriken, die die Leistungsfähigkeit objektiv messbar machen:

- **Abfrageleistung**: Zeit nach absenden einer Anfrage an die Datenbank bis zum erhalt der Antwort
- **Transaktionsrate**: Anzahl der Transaktionen, die eine Datenbank pro Sekunde ausführen kann
- **Datenladegeschwindigkeit:** Geschwindigkeit, in der die Datenbank große Datenmengen verarbeiten kann
- **Benchmarking:** Durchführung standardisierte Benchmarking Tests zur Leistungsermittlung (z.B. TPC oder SPEC Benchmark)

Skalierbarkeit

Datenbanken sollten das Potenzial besitzen, sowohl das **Datenvolumen** und die **Datenvielfalt** als auch die **Anzahl der Benutzer** zu **erhöhen, ohne dass es zu Leistungseinbußen kommt.**

Scale-Up

Hinzufügen von mehr Ressourcen, wie z.B. CPU oder Speicher, zu einem einzelnen Knoten in einem System.
Beispiel: Abnehmen der Leistung eines Datenbanksystems wird mit hinzufügen von RAM oder CPU behandelt

Scale-Out

Hinzufügen von mehr Knoten zu einem System und verteilt die Last über diese Knoten.
Beispiel: Abnehmen der Leistungs eines Datenbanksystems, wird mit dem Hinzufügen mehrerer Server mit gleicher Leistungskapazität behandelt

Datensicherheit

Datenbanken I: Grundlagen

Es soll sichergestellt werden, dass Daten **geschützt** und **sicher** gehalten werden.

Zugriffskontrolle

Es ist wichtig, dass nur autorisierte Benutzer Zugang zu den Daten. Nur autorisierte Nutzer haben die Möglichkeit den Status der Datenbank zu verändern
Beispiel: Benutzernamen und Passwörter, Zwei-Faktor-Authentifizierung oder biometrische Identifikation

Datenverschlüsselung

Das lesen der Daten ist nur mit der Verwendung des korrekten Schlüssels möglich. Selbst bei unautorisiertem Zugriff sind die Daten geschützt
Beispiel: Symmetrische Verschlüsselung, Asymmetrische Verschlüsselung, Hashing

Sicherstellung der Datenintegrität

Datenbanken I: Grundlagen

Korrektheit (Unversehrtheit) von Daten und der **korrekten Funktionsweise** von Systemen. Es gibt 4 Arten der Integrität bei Datenbanksystemen:

1. **Korrekter Inhalt**
   Die Gewährleistung konsistenter Daten setzt die korrekte Darstellung der abgerufenen Inhalte voraus.
2. **Unmodifizierter Zustand**
   Informationen sollen so übertragen werden, wie es der Nutzer gewollt und erklärt hat. Ein ummodifizierter Zustand ist gegeben, wenn das Ergebnis eines Datenverarbeitungsvorgangs unverfälscht versendet wurde.
3. **Erkennung von Modifikation**
   Dieser Integritätszustand tritt ein, wenn eine Information zwar verfälscht ist, das System dies jedoch erkennt.
4. **Temporale Korrektheit**
   Eine Datei ist temporal wiedergegeben, wenn sie die zum Versenden angegebene Zeitdauer nicht überschritten hat und dem Verwender in der richtigen Reihenfolge erscheint.

Möglichkeit konkurrierender Zugriffe

Datenbanken I: Grundlagen

Fähigkeit eines Datenbanksystems, **gleichzeitige** Datenanforderungen von mehreren Benutzern oder Anwendungen zu bearbeiten.

Locking (Sperren)

Bei diesem Ansatz wird ein Datensatz gesperrt, wenn ein Benutzer oder eine Anwendung darauf zugreift, um eine Änderung vorzunehmen. Während die Sperre in Kraft ist, können andere Benutzer oder Anwendungen diesen Datensatz nicht ändern.

OCC (Optimistic Concurrency Control)

Bei jeder Transaktion wird eine Prüfung durchgeführt, ob Konflikte aufgetreten sind. Wird ein Konflikt erkannt, wird die Transaktion abgebrochen und wiederholt

2PL (Two-Phase-Locking)

Dies ist eine spezielle Art von Sperren, bei der jede Transaktion durch zwei Phasen geht: eine Sperrenphase, in der die Transaktion alle benötigten Sperren anfordert und erhält, und eine Freigabephase, in der alle Sperren freigegeben werden.

Interoperbilität

Datenbanken I: Grundlagen

Fähigkeit eines Datenbanksystems, effizient mit anderen Systemen zu kommunizieren und zu interagieren. Dies lässt sich auf mehrere Weisen erreichen:

Standardisierte Schnittstellen & Protokolle

Verwendung von standardisierten Schnittstellen und Protokollen, die es den verschiedenen Systemen ermöglichen, miteinander zu kommunizieren. Beispielsweise ermöglicht das Structured Query Language (SQL)

APIs

APIs ermöglichen es den Anwendungen, Funktionen und Dienstleistungen von anderen Systemen oder Diensten zu nutzen. Durch APIs kann ein Datenbanksystem Daten von einem anderen System abrufen oder Daten an ein anderes System senden.

Strukturelle Änderungen

Datenbanken I: Grundlagen

Grundbegriff der relationalen Datenbank ist die Datenbankstruktur. Diese muss auch während des Betriebs noch änderbar sein. Dies wird in der Regel durch die “CRUD” Operationen sichergestellt

Create (Erstellen)

Hierbei werden neue Daten in der Datenbank erstellt und gespeichert, beispielsweise das Hinzufügen eines neuen Datensatzes.

Update (Aktualisieren)

Diese Operation ändert vorhandene Daten in der Datenbank, etwa wenn Benutzerinformationen aktualisiert werden.

Read (Lesen)

Bei dieser Operation werden Daten aus der Datenbank abgerufen. Dabei werden die Daten selbst nicht geändert. Beispielsweise das Auslesen eines Kontostandes

Delete (Löschen)

Bei dieser Operation werden spezifische Daten aus der Datenbank entfernt, beispielsweise das Löschen eines Benutzerkontos.

Zuverlässigkeit und Verfügbarkeit

Datenbanken I: Grundlagen

Zuverlässigkeit

Bezieht sich auf die Konsistenz der Leistung eines Datenbanksystems. Ein zuverlässiges System liefert **immer** das erwartete Ergebnis und widersteht verschiedenen Arten von Fehlern, um die Integrität und Konsistenz der Daten zu gewährleisten. Es implementiert Maßnahmen zur Fehlererkennung und -behebung.

Verfügbarkeit

Verfügbarkeit ist die Fähigkeit eines Datenbanksystems, kontinuierlich und ohne Unterbrechung Zugriff auf Daten zu gewähren. Ein hochverfügbares System ist so konzipiert, dass es Ausfallzeiten minimiert, auch wenn Komponenten ausfallen.
Beispiele: Redundanz und Failover-Mechanismen, um seine Funktionen auch bei Ausfällen aufrechtzuerhalten.

Diese Anforderung bezieht sich aktiv auf die Frequenz der Einhaltung der anderen Anforderungen

Aufgabe: Einführung eines Datenbanksystems

15min Prep | 5min Vorstellung pro Team | 15min Pause

Bildet 4 Teams, und wendet die “Anforderungen an moderne Datenbanken” auf die unten stehenden Use Cases an. Könnt ihr aus der Anforderung des Unternehmers nicht ableiten, dürft ihr gerne fragen.

Arbeitet bitte im zugewiesenen Space auf https://lucid.app

1 E-Commerce Unternehmen	“Wir brauchen eine Datenbank, für unser schnell wachsendes Unternehmen. Die Daten die wir sammeln sollen auch B2B Kunden auf der ganzen Welt verkauft werden
2 Bank	“Wir brauchen eine Datenbank, mit der wir unsere Finanztransaktionen abwickeln können. Die Daten dürfen unter keinen Umständen and die Öffentlichkeit”
3 Forschungsinstitut	“Wir brauchen eine Datenbank, in der wir die Rohdaten unserer Forschung ablegen können um diese anschließend zu analysieren”
4 Healthcare Unternehmen	“Wir brauchen eine Datenbank um unsere Patientendaten zu speichern. Die Daten sind extrem sensibel, müssen jedoch in seltenen Fällen angepasst werden”

DHBW
Duale Hochschule
Baden-Württemberg

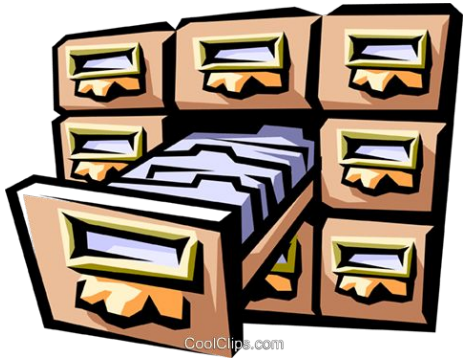
Datenbanken I: Historie von Datenbanken

Bevor es Datenbanken gab..

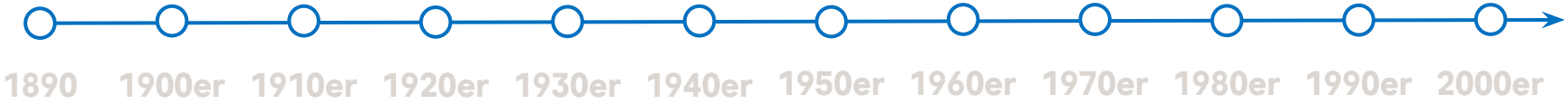
Datenbanken I: Historie

Bevor es Möglichkeiten der digitalen Aufzeichnung gab, wurden Daten vorwiegend manuell “gespeichert”

Unstrukturierte “Datenspeicher”:
Manuelle Aufzeichnungen (Schrift & Symbolik)
Strukturierte “Datenspeicher”: Karteikarten



In dieser Form der Ablage, kann man noch **nicht** von einer Form der Datenbank sprechen

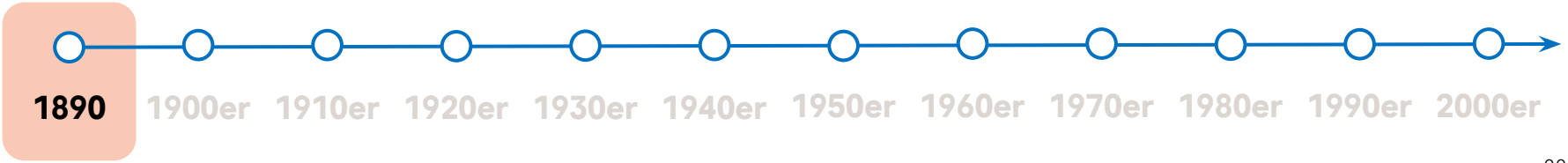


Historie der Datenbanken

Datenbanken I: Historie

Die von Hermann Hollerith entwickelten **Lochkarten** können als erster Datenspeicher gewertet werden

- Speicherkapazität war sehr begrenzt (60-100 Bytes)
- Erstellung lief manuell
- Keine nachträglichen Anpassungen möglich
- Lange Zugriffszeiten
- Starke Verschleißerscheinungen

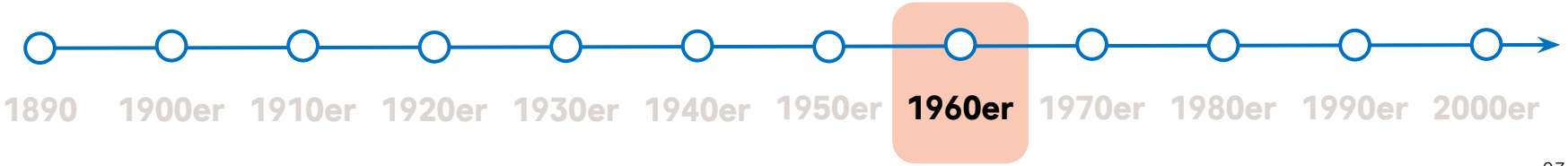


Historie der Datenbanken

Datenbanken I: Historie

Hierarchische- & Netzwerk-Datenbankmodelle

- große Konzerne entwickelten die ersten richtigen Datenbankmodelle (z.B. IBM)
- Sehr unflexibel
- Hohe Zugriffszeiten
- Vergeleichsweise hohe Datenmengen
- Als Datenspeicher werden meist Magentbänder verwendet



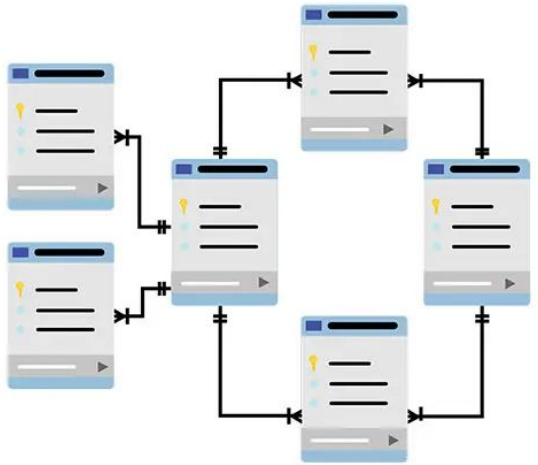
Historie der Datenbanken

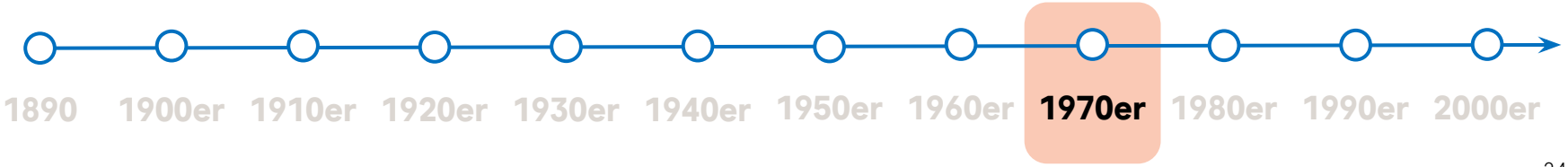
Datenbanken I: Historie

Edgar F. Codd entwickelte das erste Relationale Datenbankmodell auf dessen Basis kurz später SQL entwickelt wurde

- Dieses Modell stellt die grundlage für die meisten der heute aktiven Datenbanken
- Markiert den Startpunkt für den weltweiten Erfolg von Datenbanken

Man spricht heute oft von einer Ära “prä” und “post” Codd





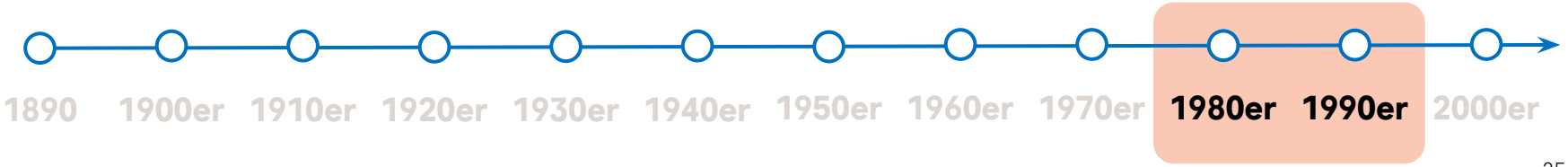
Historie der Datenbanken

Datenbanken I: Historie

Die 80er und 90er waren stark von einer **weltweiten Verbreitung und Weiterentwicklung der RDBMs & SQL** geprägt

Die technischen Fortschritte ermöglichten das Speichern von immer **mehr Daten** in immer **besser operierenden Systemlandschaften**

Firmen wie **Oracle, IBM, Microsoft oder Sybase** (von SAP übernommen) wurden zu globalen Playern und stellen noch heute in vielen Unternehmen die Datenbankstrukturen



Historie der Datenbanken

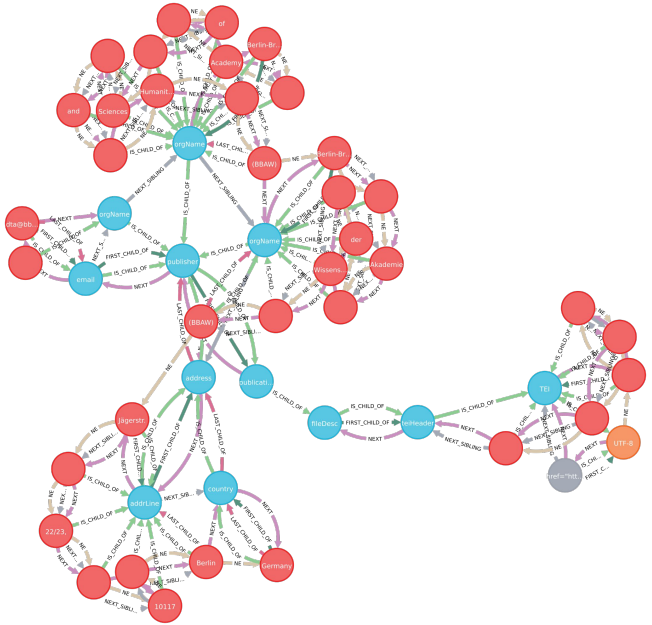
Datenbanken I: Historie

Entwicklung von sog. “**NoSQL” Datenbanken** um effizienter unstrukturierte Daten speichern zu können

Datenbanken werden vermehrt in der Cloud abgebildet und von Unternehmen als **DaaS** angeboten

In den letzten Jahren entwickelten sich vermehrt neue Modelle:

- **In-Memory Datenbanken** (z.B. SAP HANA)
- **Autonome Datenbanken** für Machine Learning
- **Graphendatenbanken** (z.B. Amazon Neptune)
- **Zeitreihendatenbanken** (für Live Daten, IoT Anwendungen)



1890 1900er 1910er 1920er 1930er 1940er 1950er 1960er 1970er 1980er 1990er **2000er**

DHBW
Duale Hochschule
Baden-Württemberg

Datenbanken I: DB, DBS & DBMS

DB, DBS & DBMS

Datenbanken I: Grundlagen

Datenbank

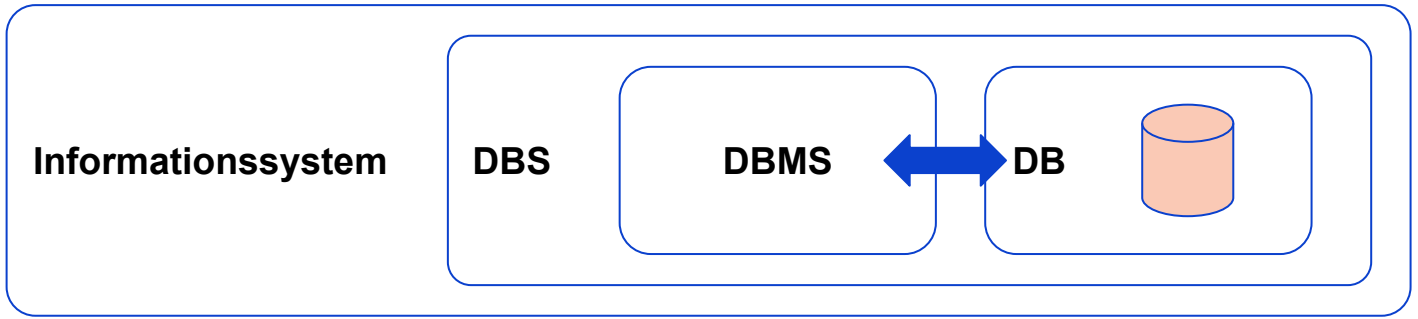
Eine Sammlung von Daten, die auf strukturierte Weise organisiert sind, so dass sie einfach abgerufen, verwaltet und aktualisiert werden können.

DBMS

Software, die für die Interaktion mit Endbenutzern, Anwendungen und der Datenbank selbst verwendet wird, um Daten abzufragen und zu manipulieren.

DBS

Ein System, das Datenbanken verwaltet. Es umfasst die Software (DBMS), die Hardware (Server, Speichersysteme usw.) und Datenbank selbst



DB, DBS & DBMS

Datenbanken I: Grundlagen

	Verantwortlichkeit	**Beispiel**
DB	Speicherung und Organisation von Daten.	Eine Kundendatenbank, die alle Informationen über die Kunden eines Unternehmens enthält
DBMS	Bietet Tools und Funktionen, um Datenbanken zu verwalten. - Durchführen der CRUD Operationen - Bereitstellen von Backup- und Wiederherstellungsfunktionen - Gewährleistung der Datensicherheit und Integrität - Verwaltung von Datenbanktransaktionen.	MySQL, Oracle, SQL Server, PostgreSQL
DBS	Beschreibt das Gesamtsystem, das die Daten (DB), das Managementsystem (DBMS), und die Hardware, auf der das DBMS und die Datenbank ausgeführt werden beinhaltet	Das Kundendatenbanksystem eines Unternehmens könnte die Kundendatenbank (DB), die DBMS-Software (z.B. mySQL), die zum Verwalten der Kundendatenbank verwendet wird, und die Server-Hardware, auf der die DB und das DBMS ausgeführt werden, umfassen.

RDBMS

Datenbanken I: Grundlagen

Das RDBMS (Relationales DBMS) ist die am häufigsten eingesetzte Form des DBMS. Es beschreibt eine Form des DBMS, die speziell auf die Belange von Relationalen Datenbanken zugeschnitten ist

Codd's 12 Rules:

- 1: Informationsregel
- 2: Garantierte Zugriffregel
- 3: Systematische Behandlung von NULL Werten
- 4: Relationaler Onlinekatalog
- 5: Umfassende Datensprache
- 6: Aktualisierbare Sichten
- 7: Hohe Stufe der Datensprache
- 8: Physikalische Datenunabhängigkeit
- 9: Logische Datenunabhängigkeit
- 10: Integritätsunabhängigkeit
- 11: Verteilungsunabhängigkeit
- 12: Nichtgefährdungsregel

Codd’s 12 Rules für RDBMS

Datenbanken I: Grundlagen

DHBW
Duale Hochschule
Baden-Württemberg

1. Informationsregel

Alle Informationen einer relationalen Datenbank (inklusive Metadaten) müssen auf eine einzige Art und Weise dargestellt werden. Gemeint ist hiermit die Darstellung in tabellarischem Format.
Beispiel: Telefonnummer eines Kunden in einer Kundentabelle

2. Garantierte Zugriffsregel

Jeder einzelne Dateneintrag (ein Wert in einer Tabelle) muss durch den Tabellennamen, den Primärschlüssel des Datensatzes und den Spaltennamen eindeutig erreichbar sein.
Beispiel: Zugriff auf Adresse eines spezifischen Kunden in der Kundentabelle über den Kunden-ID und die Spalte "Adresse"

Codd's 12 Rules für RDBMS

Datenbanken I: Grundlagen

DHBW
Duale Hochschule
Baden-Württemberg

3: Systematische Behandlung von NULL Werten

NULL Werte sollen unterstützt werden und verschiedene Arten von Unbekannten repräsentieren
Beispiel: Ein fehlendes Geburtsdatum kann als NULL dargestellt werden

4: Relationaler Onlinekatalog

Neben Benutzerdefinierten Tabellen enthält die Datenbank auch Daten über sich selbst (Metadaten). Diese können von authorisierten auf dieselbe Art abgefragt werden
Beispiel: Informationen über die Metadaten einer benutzerdefinierten Tabelle können mit SQL abgefragt werden

5: Umfassende Datensprache

Die Datenbank untersützt mindestens eine Sprache die Datenmanipulation und -definition, Integritätsregeln und Transaktionssteuerung zu verarbeiten kann
Beispiel: z.B. SQL

Codd's 12 Rules für RDBMS

Datenbanken I: Grundlagen

DHBW
Duale Hochschule
Baden-Württemberg

6: Aktualisierbare Sichten

Daten die über eine Ansicht dem Nutzer präsentiert werden duplizieren keine Inhalte, sondern wird dynamisch bei Abfrage erzeugt
Beispiel: Sicht mit Aggregatsfunktion berechnet die angezeigten Werte bei jeder Aktualisierung erneut

7: Hohe Stufe der Datensprache

Es müssen grundlegende relational algebraische Operationen (Selektionen, Projektionen und Joins) als auch Set Operationen wie Union, Intersektion, Division und Differenz unterstützt werden. Dies geschieht durch die Datenbanksprache.
Beispiel: SQL Ausführungen von Joins können verarbeitet werden.

Codd's 12 Rules für RDBMS

Datenbanken I: Grundlagen

8. Physikalische Datenunabhängigkeit

Änderungen auf der phyischen Ebende der Datenbank soll keine Auswirkungen auf die Anwendungsprogramme haben
Beispiel: Hinzufügen von zusätzlichen Festplatten wirkt sich nicht auf die Applikation aus, die die Datenbank verwendent

9. Logische Datenunabhängigkeit

Änderungen auf der logischen Ebene der Datenbank sollen möglichst wenige Auswirkungen auf die Anwendungsprogramme haben
Beispiel: Hinzufügen einer weiteren Spalte einer Tabelle wirkt sich nicht/wenig auf die Applikation aus, die die Datenbank verwendet

ANSI/SPARC 3-Schicht-Architektur

Datenbanken I: Grundlagen

Externe Ebene

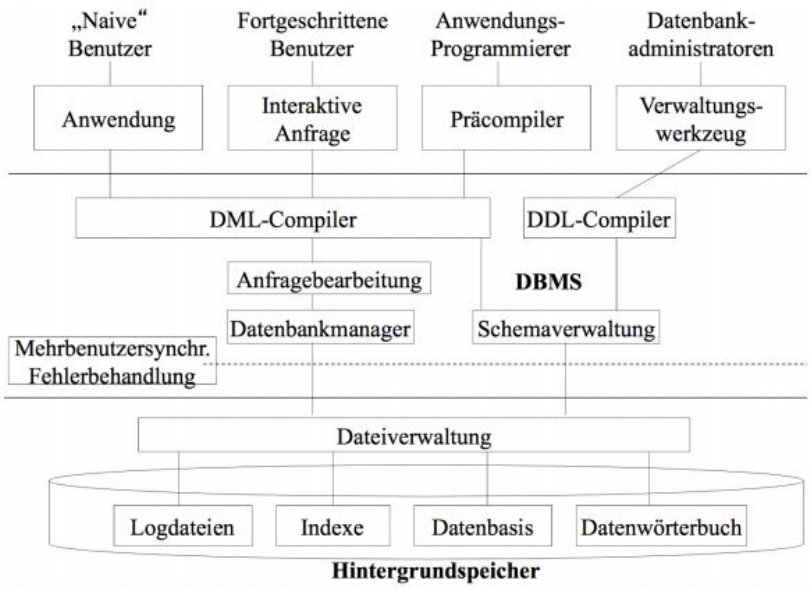
Sichten auf die Daten (Formulare, Layouts, APIs)

Kozeptionelle Ebene

system- und anwendungsunabhängige Struktur (Datenbankschema)

Interne Ebene

physikalische Speicherstrukturen einer Datenbank



Notenportal + Nutzerspezifische Datenstruktur

Gesamtes Datenmodell für Studierende

Indezierung mit B-Bäumen
Speicherung auf Cloud Architektur

Codd's 12 Rules für RDBMS

Datenbanken I: Grundlagen

DHBW
Duale Hochschule
Baden-Württemberg

10: Integritätsunabhängigkeit

Die Integritätsregeln müssen in das DBMS integriert sein und nicht in die Applikation
Beispiel: Das Sicherstellen von eindeutigen Produkt IDs wird in der Datenbank durchgeführt

11: Verteilungsunabhängigkeit

Applikationen müssen mit verteilten Abfragen homogener (anderes RDBMS) und heterogener Art (Tabellen außerhalb eines RDBMS) umgehen können
Beispiel: Die Verbindung einer Tabelle von Kunden auf einer RDBMS und den Sales aus einem XLS File kann per Join ausgeführt werden

Codd’s 12 Rules für RDBMS

Datenbanken I: Grundlagen

12: Nichtgefährdungsregel

Es darf keine Möglichkeit der Unterwanderung einer der bestehenden 11 Regeln geben.
Beispiel: Wird die Integrität der Datenbank durch die Applikation sichergestellt, ist sie kein RDBMS

1990: Regel 0

Für jedes System, das behauptet, ein relationales Datenbanksystem zu sein, muss es möglich sein, das System ausschließlich über sein relationales Modell zu verwalten, ohne dass auf irgendeine Weise auf Systemaspekte außerhalb dieses Modells zurückgegriffen werden muss.

Aufgabe: Codd's 12 Rules

15min Prep | 15min Pause | 5min Vorstellung pro Team

Codd's 12 Rules werden von den meisten Datenbanken gar nicht vollumfänglich erfüllt. Beim Einführen eines Datenbanksystems sollte dies zuvor durchdacht werden. Recherchiert in 4 Teams über die Möglichkeiten der Nicht-Erfüllung und zeigt unternehmensbezogen ein Beispiel wieso die Nicht-Erfüllung entweder kritisch ist oder sogar gewollt

1 E-Commerce Unternehmen	"Wir brauchen eine Datenbank, für unser schnell wachsendes Unternehmen. Die Daten die wir sammeln sollen auch B2B Kunden auf der ganzen Welt verkauft werden
2 Bank	"Wir brauchen eine Datenbank, mit der wir unsere Finanztransaktionen abwickeln können. Die Daten dürfen unter keinen Umständen and die Öffentlichkeit"
3 Forschungsinstitut	"Wir brauchen eine Datenbank, in der wir die Rohdaten unserer Forschung ablegen können um diese anschließend zu analysieren"
4 Healthcare Unternehmen	"Wir brauchen eine Datenbank um unsere Patientendaten zu speichern. Die Daten sind extrem sensibel, müssen jedoch in seltenen Fällen angepasst werden"

DHBW
Duale Hochschule
Baden-Württemberg

Zugriffe auf eine Datenbank

ACID Modell

Datenbanken I: Grundlagen

1 Kriterium **Atomicity**

Eine Transaktion ist eine unteilbare und unabhängige Arbeitseinheit.

Sie wird entweder ganz oder gar nicht ausgeführt

Bei Fehlern wird ein “Rollback” durchgeführt

2 Kriterium **Consistency**

Eine Transaktion führt von einem Konsistenten Zustand in einen neuen konsistenten Zustand

Alle Integritäts-bedingungen müssen nach der Transaktion erfüllt sein

3 Kriterium **Isolation**

Eine Transaktion wird isoliert von konkurrierenden Transaktionen ausgeführt

Das Ergebis ist immer wie nach sequenzieller Ausführung der Transaktionen, auch wenn diese Parallel laufen

4 Kriterium **Durability**

Eine Transaktion ist dann abgeschlossen, wenn ihre Änderungen persistent in der übernommen wurden

Selbst bei einem Systemausfall bleiben die Änderungen erhalten

Ohne die Befolgung des ACID Modells, kann die Zuverlässigkeit eines DBS nicht gewährleistet werden

SQL (Structured Query Language)

Datenbanken I: Grundlagen

SQL ist eine **standardisierte Programmiersprache** (Abfragesprache), die speziell für die Interaktion mit Datenbanken entwickelt wurde. SQL wird verwendet, um Daten zu **abzufragen**, **einzufügen**, zu **aktualisieren** und zu **löschen**, sowie um Datenbankstrukturen zu **erstellen** und zu **ändern**.

DML Data Manipulation Language	DDL Data Definition Language	DCL Data Control Language	TCL Transaction Control Language
DML ist für das Abrufen, Einfügen, Löschen und Aktualisieren von Daten in einer Datenbank verantwortlich.	**Datenbankstrukturen oder -schemata zu definieren. Mit DDL-Anweisungen können Sie Tabellen erstellen, ändern oder löschen.**	**DCL wird verwendet, um Zugriffsrechte in der Datenbank zu steuern.**	**TCL wird verwendet, um Transaktionen in der Datenbank zu verwalten. Sie können Transaktionen beginnen, rückgängig machen und bestätigen.**
`SELECT, INSERT, UPDATE, DELETE`	`CREATE, ALTER, DROP`	`GRANT, REVOKE`	`COMMIT, ROLLBACK, SAVEPOINT;`

Sonstige Datenbankzugriffe

Datenbanken I: Grundlagen

Meist unterstützen Datenbanken neben SQL noch weitere Möglichkeiten um Daten abzufragen:

SOAP API Simple Object Access Protocol	REST API Representational State Transfer	Angepasste API	ABAP Advanced Business Application Programming
Protokoll zur Übertragung von strukturierten Informationen in Webservice Anwendungen. Es wird primär HTTP/HTTPS verwendet Zieldatenformat ist XML Unterstützt ACID im Standard	**Architektur für die Übertragung von Daten in netzwerkbasierter Software** Im Gegensatz zu SOAP ein Protokoll, mehr ein Designpinzip Verwendet ebenfalls HTTP/S REST APIs sind stateless	**Basieren oft auf REST oder SOAP, werden aber für spezifische Geschäftsanforderungen optimiert und sind effizienter** Hoher Aufwand in der Entwicklung Beispiel: Twitter API, Slack API	**Ist eine Programmiersprache und bietet neben der Verwendung von SQL auch eigene Möglichkeiten Transaktionen zu verwalten** Im Gegensatz zu SOAP ein Protokoll, mehr ein Designpinzip

Finale Übung

15min Prep | 15min Pause | 5min Vorstellung pro Team

Entwerfen Sie eine Datenbank für folgendes Fallbeispiel:
Mitarbeiterdatenbank: Jeder Mitarbeiter arbeitet in einer Abteilung. Jede Abteilung ist in einem Bereich zugeordnet. Jeder Bereich hat einen zugeordneten Kundenkreis. Jede Abteilung und jeder Bereich hat einen Vorgesetzten, der auch ein Mitarbeiter ist. Kunden kaufen Dienstleistungen bei einer Abteilung ein, die dann durch einen Mitarbeiter ausgeführt werden. Eine Dienstleistung kann durch fast jede Abteilung ausgeführt werden, nicht jede Abteilung kann jede Dienstleistung durchführen.

Wie ihr diese Datenbank entwerft, liegt vollkommen in eurem Ermessen, versucht mittels des gelernten, die Anforderungen in irgendeiner Form zu erfüllen

Retro

15min

Zum Ende jeder Vorlesung werden wir gemeinsam eine sogenannte Retrospektive durchführen. Diese hat das Hauptziel, den Arbeitsprozess zu **reflektieren** und **kontinuierliche** Verbesserungen zu identifizieren.

Hier wird die Retro aus zwei Teilen bestehen

1. Stimmungsbarometer
2. Start/Stop/Continue Board

Retro DO's	Retro DONT's
Reflektion zu Vorlesungstempo, Inhalte (Anspruch), Struktur Konkrete Beispiele und Vorschläge Ideen und Wünsche für zukünftige Vorlesungen	Verallgemeinernde Bezeichnungen (Alles Schlecht/Alles Gut) Fachliche Beiträge / Sachdiskussionen