

Datenbanken - Zusammenfassung

1. Charakterisierung
 - 1.1 Aufgaben:
 - große Datenmengen speichern
 - nach beliebigen Kriterien wiederfinden
 - verändern
 - 1.2 Anforderung: Alle Informationen müssen auf dem neusten Stand verfügbar sein
 - 1.3 Definition:

Eine Datenbank ist eine integrierte Ansammlung von Daten, die allen Benutzern eines Anwendungsbereiches als gemeinsame Basis aktueller Information dient. Integriert: Die Daten sind entsprechend den natürlichen Zusammenhängen in der Anwendungswelt strukturiert - und nicht danach, wie einzelne Anwendungen die Daten benötigen

gemeinsame Basis: Die Daten können durch viele Benutzer (auch parallel) genutzt werden. Die verschiedenen Benutzer haben unterschiedliche Sichten auf die Daten.
2. Motivation
 - 2.1 Probleme
 - Speicherverschwendung durch Mehrfachspeicherung
 - Möglicherweise Inkonsistenz in den Daten
 - Schwierige Änderung/Updates
 - Integration der Daten für neue Anwendungen
 - 2.2 Beschränkte Zugriffsmöglichkeiten
 - Schwierige Verknüpfung der Daten
 - Mehrbenutzerbetrieb (kritische Abschnitte, paralleler Zugriff)
 - 2.3 Sicherheit
 - Regelung der Zugriffsrechte auf die Daten
 - Recovery der Daten nach einem Fehlerfall (Festplatte hatte einen headcrash und einzelne Dateien sind verloren)
 - Sicherstellen von Integritätsbedingungen
 - 2.4 Software-Engineering
 - Programmierer muss interne Darstellung der Daten kennen
 - Hoher Entwicklungsaufwand für neue Anwendungen
 - 2.5 Datenabstraktion
 - 2.5.1 Sicht
 - Teilmenge an Informationen, die für eine Anwendung erforderlich sind
 - 2.5.2 Logische Ebene
 - In einem Datenbankschema wird festgelegt, welche Daten gespeichert werden
 - 2.5.3 Physische Ebene
 - Legt fest, wie Daten auf den Speichermedien organisiert, codiert und abgelegt werden
 - 2.5.4 Physische Datenunabhängigkeit
 - Änderungen an der physischen Ebene wirken sich nicht auf die anderen Ebenen aus (z.B. nachträglich einen Index einfügen)
 - 2.5.5 Logische Datenunabhängigkeit
 - Änderungen in der logischen Ebene wirken sich nicht auf die Anwendungen/Sichten aus (z.B. Hinzufügen oder Entfernen eines Datenfeldes) - komplexere Änderungen der logischen Ebene lassen sich vor den Anwendungen nicht verbergen. Deshalb ist besondere Sorgfalt bei der Erstellung dieser Ebene erforderlich

- 2.6 Aufgaben eines Datenbanksystems nach Codd
 - 2.6.1 Integration
 - Einheitliche Verwaltung aller von den Anwendungen benötigten Daten (ohne Redundanz)
 - 2.6.2 Operationen
 - Erstellen, Suche, Ändern und Löschen von Daten ist möglich
 - 2.6.3 Katalog bzw. Schema (engl. data dictionary)
 - Eine Beschreibung der Daten in der Datenbank ist vorhanden (Metadaten)
 - 2.6.4 Benutzersichten (engl. views)
 - Das System kann spezielle Sichten auf die Daten für einzelne Anwendungen bereitstellen
 - 2.6.5 Integritätssicherung (engl. integrity)
 - Das System überwacht selbständig die Konsistenz der Daten, also die Korrektheit von Datenbankinhalten
 - 2.6.6 Zugriffskontrolle
 - Überwachung und Steuerung von Zugriffen auf die Daten durch Anwender
 - 2.6.7 Transaktionen (engl. transaction)
 - Das System kann bestimmte Operationen zusammenfassen, so daß sie als Gruppe entweder ganz oder gar nicht ausgeführt werden (Atomizität)
 - 2.6.8 Synchronisation (engl. synchronization)
 - Das System koordiniert parallele Zugriffe, so daß gegenseitige Beeinflussungen vermieden werden
 - 2.6.9 Datensicherung
 - Das System unterstützt die Wiederherstellung (engl. recovery) der Daten nach einem Absturz oder Fehler
- 3. Datenbanken-Technologien
 - 3.1 Begriffserklärung
 - 3.1.1 Datenbank (DB)
 - Strukturierter Datenbestand, der von einem Datenbankmanagementsystem verwaltet wird
 - 3.1.2 Datenbankschema
 - Legt die Struktur der Datenobjekte in der Datenbank fest (Metadaten)
 - 3.1.3 Datenbankausprägung (Instanz)
 - Konkreter Inhalt/Zustand der Daten in einer Datenbank
 - 3.1.4 Datenbankmanagementsystem(DBMS)
 - Software zur Verwaltung von Datenbanken (Daten definieren, Daten speichern/ändern/löschen, Anfragen implementieren, Sicherheit)
 - 3.1.5 Datenbanksystem (DBS)
 - DBMS und Datenbanken
 - 3.2 Schema-Architektur
 - 3.2.1 externes Schema
 - Anwendungsspezifische Teilsichten auf den Datenbestand
 - 3.2.2 Konzeptuelles Schema
 - Implementierungsunabhängige Datenmodellierung. Beschreibt die Struktur der Datenbank vollständig
 - 3.2.3 Internes Schema
 - Beschreibt systemspezifische Realisierung, z.B. Zugriffspfade

- 3.3 Phasen des Datenbankentwurfs
 - 3.3.1 Das Fachproblem liegt normalerweise vor
 - 3.3.2 Anforderungsanalyse
 - Welche Informationen werden in der Datenbank gespeichert, welche Operationen werden auf den Daten ausgeführt usw.
 - 3.3.3 Konzeptioneller Entwurf
 - Beschreibe das Schema der Daten unabhängig von der späteren Implementierung
 - 3.3.4 Logischer Entwurf
 - Detailliere den konzeptionellen Entwurf, optimiere die Schemata
 - 3.3.5 Schema-Entwicklung für ein spezielles DBMS, Deklaration der Daten
 - 3.3.6 Physischer Entwurf
 - Lege (Speicher-)Zugriffstrukturen fest
 - 3.3.7 Implementierung und Wartung
 - Installation des Datenbanksystems, Anpassung an neue Anforderungen
- 3.4 Entwurfsmodelle
 - 3.4.1 Entity Relationship Modell (ER Modell)
 - Basiert auf den Grundkonzepten Entity (Informationseinheit), Attribut (Eigenschaft des Entity) und Relation (Beziehung zwischen Entities)
 - 3.4.2 Erweitertes Entity Relationship Modell (EER Modell)
 - Mehr Attributtypen
 - Generalisierung (engl. is-a), Spezialisierung und Aggregation (engl. part-of)
 - 3.4.3 Unified Modelling Language (UML)
 - Allgemeine Sprache, nicht nur zur Modellierung im Datenbankenbereich
 - 3.4.4 Relationenmodell (RDBMS)
 - Basiert auf dem mathematischen Modell der Relation (z.B. Oracle 8i/9i/10g, MySQL, PostgreSQL)
 - 3.4.5 Netzwerkmodell
 - CODASYL 1971, eingeschränktes Relationenmodell (keine n:m Beziehungen), darstellbar als Graph
 - 3.4.6 Hierarchisches Modell
 - Schema ist eine Menge von Bäumen (spezielle Graphen), z.B. IBM IMS
 - 3.4.7 Objektorientiertes und Objekt-Relationales Modell (OODBMS)
 - Mehr Konzepte (Mengen, Tupel, Listen, Vererbung)
- 3.5 Schema
 - Beschreibt die Struktur einer Datenbank, also den Aufbau eines Inhalts/der Daten. Wird in der Regel in der Datenbank zusammen mit den Daten gespeichert (Metadaten im sogenannten Katalog, engl. data dictionary)
- 3.6 Konzeptuelles Schema
 - Beschreibt die Struktur der Wirklichkeit bzw. eines Ausschnittes aus der Wirklichkeit (Miniwelt) in einer darstellungsunabhängigen Notation (Entwurfsmodell)
- 3.7 Darstellungsschema
 - Beschreibt die Struktur der Daten in einer Datenbank und hängt von der Darstellungstechnologie ab (Implementierungs- oder Darstellungsmodell)
- 3.8 Instanz
 - Konkrete Menge an Daten in einer Datenbank (plus Metadaten)

4. Das konzeptuelle Modell
 - 4.1 Schema-Architektur
 - Externes Schema: Anwendungsspezifische Teilsichten auf den Datenbestand
 - Konzeptuelles Schema: Implementierungsunabhängige Datenmodellierung.
 - Beschreibt die Struktur der Datenbank vollständig
 - Internes Schema: Beschreibt systemspezifische Realisierung, z.B. Zugriffspfade
 - 4.2 Miniwelten
 - 4.2.1 Definition
 - Eine Miniwelt ist ein relevanter Ausschnitt aus der realen Welt (Realität)
 - Erfassen der Realität = Verstehen der Informationen
 - 4.2.1.1 Information
 - Etwas Drittes neben Materie und Energie (philosophisch)
 - Information setzt den Menschen über seine Außenwelt in Kenntnis, ist also der "Stoff", der Erkenntnis ermöglicht.
 - Entropie einer Nachrichtenquelle (Theorie von Shannon)
 - 4.2.2.2 Daten
 - Daten sind letztlich Zeichenfolgen über einem Alphabet
 - Wenn Daten eine Bedeutung bekommen, werden sie zu Informationen
 - 4.2.2 Methoden zur Informationsgewinnung
 - Interviews
 - Analyse bestehender Arbeitsabläufe (Vorgänge)
 - Analyse genutzter Formulare oder Standarddokumente
 - 4.2.3 Ergebnis: Nicht-formale Beschreibungen des Fachproblems
 - Texte
 - Tabellen
 - Formblätter

Man erkennt Gegenstände/Objekte der realen Welt, die

 - bestimmte Eigenschaften (Attribute) haben
 - in bestimmten Verbindungen zueinander stehen
 - 4.3 Das konzeptuelle Modell
 - strukturiert den zukünftigen Anwendungsbereich
 - beschreibt die Gesamtheit derjenigen Daten, die in der Datenbank verwaltet werden
 - unabhängig von Gesichtspunkten einzelner Benutzer
 - unabhängig von Gesichtspunkten der physischen Speicherung
 - beschreibt die Integritätsbedingungen, also Bedingungen oder Vorschriften, die
 - für die Daten immer gelten oder
 - für die Änderungen von Daten gelten
 - 4.3.1 Teile des konzeptuellen Modells
 - Objekttypen mit Attributen
 - z.B. Lehrveranstaltung mit Attributen Bezeichnung, Semester, ...
 - Beziehungen zwischen den Objekttypen
 - z.B. bietet_an zwischen ProfessorInnen und Lehrveranstaltungen
 - Integritätsbedingungen
 - z.B. Matrikelnummer 6stellig, Geburtsdatum nach 1900
 - 4.3.2 Konzept der Schlüssel
 - verschiedene Objekte der Miniwelt müssen unterscheidbar sein
 - was macht das Objekt eigentlich aus?
 - welche Attribute identifizieren das Objekt eindeutig?
 - Beispiel: Studierende und ihre Attribute
 - Vorname, Nachname
 - Matrikelnummer
 - Geburtsdatum
 - Anschrift
 - Telefonnummer

Schlüssel sind Teilmengen von Attributen, die ein Objekt eindeutig identifizieren

 - Beispiel: Studierende und ihre möglichen Schlüssel
 - Vorname, Nachname und Geburtsdatum

4.3.3 Modellierungskonzepte im ER Modell

- Entity-Mengen, Entity-Typen (Objekte)
- Wertebereiche, Attribute
- Primärschlüssel
- Beziehungsmengen

Modellierungsebene:

- Das ER Modell modelliert auf der Typ-Ebene, nicht auf der Instanz-Ebene
- Aussagen werden über Mengen getroffen, nicht über einzelne Elemente einer Menge (Menge der Studierenden vs. Student Klaus Müller)
- Integritätsbedingungen und Constraints ergänzen das Modell

4.4 Entities (Objekte)

4.4.1 Entity

- wohlunterscheidbare Dinge der Miniwelt
- sie besitzen Eigenschaften, deren konkrete Ausprägung als Werte bezeichnet werden

4.4.2 Entity-Menge

- Menge von gleichartigen Entities
- sie besitzen gemeinsame Eigenschaften

4.4.3 Beispiel:

Entity Elisabeth Dennert-Möller mit Eigenschaft Größe und Wert 178 cm
Entity-Menge der Anwesenden. Jedes Entity hat eine Größe.

4.4.4 Eigenschaften von Entities

- Anzahl der später tatsächlich vorkommenden Entities
 - Schätzung der Größenordnung
 - wichtig, um die spätere Größe der Datenbank schätzen zu können
- Identifizierungseigenschaft(en)
- Zustandsbeschreibung(en), möglichst zeitrelevant

4.4.5 Wichtige Angaben zur Eigenschaft

- Name der Eigenschaft
- potentieller Wertebereich
 - Elementar: Ganzzahlig, Gleitkommazahl, String usw.
 - Eventuelle Einschränkungen

5. Attribute (Eigenschaften)
Attribute bezeichnen Eigenschaften , die alle Elemente einer Entity-Menge besitzen (gemeinsame Eigenschaften)
 - 5.1 Wertebereich
Als Wertebereich oder Domäne werden die zulässigen Werte eines Attributes bezeichnet
Ein Attribut ordnet jedem Entity aus einer Entity-Menge einen Wert aus einem Wertebereich zu.
 - 5.2 Entity Beschreibung
 - Name der Entity-Menge ist zeitinvariant
 - Name der Attribute ist zeitinvariant
 - Entity-Menge ist nicht zeitinvariant
 - 5.3 Attribut-Typen
 - 5.3.1 Einwertige Attribute (klassisches ER-Modell)
nehmen genau einen Wert aus dem Wertebereich an
 - 5.3.2 Mehrwertige Attribute (erweitertes ER-Modell)
können für ein konkretes Entity einen oder mehrere Werte aus dem Wertebereich annehmen (ein Buch kann mehrere Autoren haben)
 - 5.3.3 Zusammengesetzte Attribute (erweitertes ER-Modell)
nehmen in einem konkreten Entity für jede ihrer Komponenten einen Wert an (eine Adresse besteht aus einem Strassennamen, einer Postleitzahl und einem Ort)
6. Schlüssel
 - 6.1 Definition
Ein Schlüssel ist eine minimale Menge von Attributen, deren Werte das zugeordnete Entity eindeutig innerhalb aller Entities diesen Typs identifiziert.
Minimal: Keine Teilmenge des Schlüssels identifiziert die Entities eindeutig
Problem: Es kann mehrere Schlüssel für eine Entity-Menge geben (Schlüsselkandidaten)
Lösung: Wähle einen Schlüssel aus und nenne ihn Primärschlüssel
 - 6.2 Auswahl des Primärschlüssels
 - 6.2.1 Zeitinvarianz beachten
 - Haarfarbe könnte sich später mal ändern
 - Namen können sich ändern (Heirat)
 - neue Entities können hinzukommen (deren Eigenschaften man heute noch nicht kennt)
 - 6.2.2 künstlicher Primärschlüssel
 - findet man in der Praxis sehr häufig, wird bei Bedarf eingefügt
 - Beispiele: Matrikelnummer, Personalnummer, Bestellnummer, Kontonummer, ...
 - 6.2.3 Schlüsselattribute werden unterstrichen
Beispiel: Buch (Titel, Autor, Verlag, ISBN)

7. Beziehungen
- 7.1 Definition
 Beziehungen beschreiben Zusammenhänge zwischen Entities
 Beziehungstypen sind abstrahierte Beziehungen zwischen Entity-Typen.
 Ein Beziehungstyp R kann als kartesisches Produkt betrachtet werden. Jede Teilmenge des kartesischen Produktes ist eine mögliche Beziehungsmenge.
- 7.2 Kardinalität von binären Beziehungen
- 7.2.1 1:1 Beziehung
 Jedem Entity aus E1 ist höchstens ein Entity aus E2 zugeordnet und umgekehrt
- 7.2.2 1:n Beziehung (n:1 analog)
 Jedem Entity aus E1 können beliebig viele Entities aus E2 zugeordnet sein. Einem Entity aus E2 kann immer nur höchstens ein Entity aus E1 zugeordnet sein.
- 7.2.3 n:m Beziehung
 Jedes Entity aus E1 kann mit beliebig vielen Entities aus E2 in Beziehung stehen und umgekehrt.
- Beziehungen sind Integritätsbedingungen und müssen daher grundsätzlich gelten, nicht nur zufällig für die aktuellen Entities. Sie können auch innerhalb einer Entity-Menge vorhanden sein (z.B. Personen sind mit Personen verheiratet).
- 7.2.4 (min, max) Notation
- Ist präziser als die x:y Notation
 - In der x:y Notation wird nur "1" oder "viele" festgelegt
 - Für jedes Entity in einer Beziehung werden in der (min, max) Notation Ober- und Untergrenzen festgelegt
 - Wenn es Entities geben darf, die gar nicht an der Beziehung teilnehmen, so wird min mit 0 angegeben
 - Wenn ein Entity beliebig oft an einer Beziehung teilnehmen darf, so wird die max-Angabe durch * ersetzt
- 7.3 Schwache Entities
 Schwache Entities werden auch existenzabhängige Entities genannt.
- 7.3.1 Eigenschaften
- Sie sind von der Existenz eines übergeordneten Entitys abhängig
 - Häufig nur in Kombination mit dem Schlüssel des übergeordneten Entitys eindeutig identifizierbar
 - Schwache Entities sind doppelt umrandet
8. Erweiterte ER-Konzepte
- 8.1 Generalisierung
 Die Abstraktion auf Typ-Ebene ist vergleichbar mit dem Vererbungskonzept in der objektorientierten Programmierung. Gemeinsame Attribute werden "herausfaktoriert" und dem Obertypen zugeordnet. Untertypen erben die Attribute ihrer Obertypen.
 Generalisierung geht von unten nach oben (bottom up, z.B. Ein Professor ist ein Mitarbeiter ist eine Person)
- 8.2 Spezialisierung
 Gleiches Prinzip wie bei Abstraktion, allerdings von oben nach unten (top down), vom Allgemeinen zum Speziellen
 Die Spezialisierungen können überlappen (z.B. können Vereinsmitglieder mehreren Vereinen angehören), disjunkt (ein Mitarbeiter kann entweder ein Manager oder ein Azubi sein, beides geht nicht), total (ein Mitarbeiter kann Manager oder Nicht-Manager sein) oder partiell sein (ein Mitarbeiter kann Manager, Azubi aber auch was anderes sein).
- 8.3 Aggregation
 Aggregation modelliert die Tatsache, dass Objekte aus anderen Objekten zusammengesetzt sein können (engl. part of) (z.B. Lenker ist Teil von Rahmen ist Teil von Fahrrad)

9. Das relationale Modell
 - 9.1 Modellentwurf
 - 9.1.1 Sichten

Sichten können z.B. Organisationsstruktur oder Buchhaltung sein
 - 9.1.2 Synonyme

Z.B. Manager und Mitarbeiter
 - 9.1.3 Widerspruchsfreiheit
 - Synonyme und Homonyme beachten
 - Attributwertebereiche einheitlich
 - Kardinalitäten bei Beziehungen einheitlich
 - Sachverhalte immer gleich modellieren
 - 9.1.4 Beziehungen

Können als Beziehungstyp im ER-Diagramm oder als Entitätstyp, der nur in Beziehung zu anderen Entities existieren kann, modelliert werden (z.B. Prüfung als Beziehung zwischen Student und Fach oder Bestellung als Beziehung zwischen Teil, Projekt und Lieferant)
Die Modellierungsvariante ist abhängig davon,

 - ob eine Beziehung auch Attribute hat
 - wie viele Entity-Typen an der Beziehung beteiligt sind
 - 9.1.5 Mehrstellige Beziehungen

Sind als Beziehungstyp einfach zu modellieren, aber nicht äquivalent zu mehreren zweistelligen Beziehungen. Die Beziehung kann auch als Entity modelliert werden.
 - 9.2 Übersicht über das relationale Modell
 - 9.2.1 Datenstrukturen

Ein relationales Modell definiert eine Datenstruktur

 - die Relation (Tabelle) ist die einzige Datenstruktur (außer atomare Werte)
 - alle Informationen werden ausschließlich durch Werte dargestellt
 - zeitinvariante Typinformation ist im Relationenschema festgehalten
 - 9.2.2 Implementierung von Beziehungen
 - 9.2.3 Operationen auf (mehreren) Relationen
 - Vereinigung, Differenz
 - Kartesisches Produkt
 - Projektion, Selektion
 - Grundoperationen
 - 9.2.4 Entwurfstheorie

Normalformen (was sind "gute" Relationenmodelle?)
 - 9.3 Relationen als Tabellen

Die Gesamtheit der Attribute (z.B. Name, Straße, Hausnummer) wird als Relationenschema bezeichnet. Eine Zeile (z.B. Herr Müller in der Bahnhofstrasse 125) ist ein Tupel, die Gesamtheit aller Tupel ist eine Relation.

Relationenschema Telefonbuch(Name, Strasse, Hausnummer)
 $\text{dom}(\text{Name}) = \text{dom}(\text{Strasse}) = \{\text{Zeichenreihe}\}$
 $\text{Dom}(\text{Hausnummer}) = \{\text{3stellige Zahl}\}$

Die Ordnung der Zeilen oder der Spalten ist ohne Bedeutung, alle Werte sind atomar und es gibt Schlüsselkandidaten sowie einen Primärschlüssel.
 - 9.4 Relationen

Gegeben seien n Wertebereiche (Domänen) D_1, D_2, \dots, D_n die nur atomare Werte enthalten (atomare Werte sind Zeichenketten und/oder Zahlen)
Eine Relation R ist definiert als Teilmenge des kartesischen Produktes (Kreuzprodukt) der Domänen (R ist Teilmenge aus $D_1 \times D_2 \times \dots \times D_n$)
 - 9.5 Relationales Datenmodell

Daten der realen Welt werden als Sammlung von Relationen dargestellt.

- 9.6 Relationenschema
 Das Relationenschema ist eine Festlegung der Eigenschaften von Relationen =>
 $R(A_1, A_2, \dots, A_n)$
 Es setzt sich zusammen aus paarweise verschiedenen Attributen A_1, A_2, \dots, A_n ,
 jedem Attribut A_i ist ein Wertebereich $\text{dom}(A_i)$ zugeordnet. Zu jedem Zeitpunkt
 existiert genau eine Relation r zum Relationenschema R : " r ist eine Instanz von
 R ". Die zu R gehörigen Relationen sind also sämtlich Relationen des Typs r
 Teilmenge von $\text{dom}(A_1) \times \text{dom}(A_2) \times \dots \times \text{dom}(A_n)$. Unterschied ist oft unwichtig.
 Die Gesamtheit der Relationenschemata einer Datenbank (+
 Integritätsbedingungen) heisst Schema der relationalen Datenbank oder
 Datenbankschema.
- 9.6.1 Unterschiede zum ER-Modell
 Einwertige Attribute, Definitionsbereiche (Domänen) und Primärschlüssel
 haben sie gemeinsam, statt zusammengesetzter und mehrwertiger
 Attribute werden im Relationenmodell nur unabhängige einwertige Attribute
 verwendet. Aus der Entity- und Beziehungsmenge wird die Relation
 (Tabelle).
- 9.7 Umwandlung ER-Modell in ein relationales Modell
- 9.7.1 Anforderungen
- Informationserhaltung
 Abbildung der ER-Konzepte auf relationale Konzepte
 Möglichst genaue Übereinstimmung in der Semantik
 - Minimierung der Redundanz
 - Minimierung des Verknüpfungsaufwands
 - Natürliche Abbildung
 - Keine Vermischung von Objekten
 - Verständlichkeit
- 9.8 Grundkonzepte des relationalen Modells
- 9.8.1 Jeder Entity-Typ wird als Relation mit seinen Attributen und eindeutigem
 Primärschlüssel dargestellt.
 Attribute des ER-Modells werden Attribute des relationalen Modells
 - atomare Attribute sind kein Problem
 - zusammengesetzte Attribute werden "flachgeklopft"
 - Generalisierung ist im relationalen Modell nicht vorgesehen und muss von
 Hand nachgebildet (imitiert) werden
- 9.8.2 Jeder Beziehungstyp kann als eigenständige Relation definiert werden,
 wobei die Primärschlüssel der zugehörigen Entitytypen in dieser Relation
 enthalten sein müssen.
 Primärschlüssel der Beziehungstypsrelation kann der aus den
 Fremdschlüsseln zusammengesetzte Schlüssel sein oder ein anderer
 Schlüsselkandidat (z.B. ein künstlicher Schlüssel). Die Attribute des
 Beziehungstyps werden zu Attributen der Relation.
 Die Relation zu einer Beziehung R enthält alle Primärschlüssel der
 beteiligten Entities E_i und alle Attribute der Beziehung R .
- 9.8.3 Jeder N:M-Beziehungstyp muss als eigenständige Tabelle definiert werden.
 Dabei treten mindestens die Primärschlüssel der zugehörigen Entity-Typen
 als Fremdschlüssel auf. Der Primärschlüssel der Beziehungstabelle ist
 entweder der aus den Fremdschlüsseln zusammengesetzte Schlüssel oder
 ein anderer Schlüsselkandidat. Die weiteren Attribute des Beziehungstyps
 gehen in die Attribute der Tabelle über.
- 9.8.4 Jeder 1:N-Beziehungstyp kann ohne eigenständige Tabelle ausgedrückt
 werden, indem in der Tabelle mit der N-Kardinalität die Schlüssel der
 referenzierten Tabelle und weitere Attribute des Beziehungstyps geführt
 werden.
- 9.8.5 Ein Beziehungstyp 1:1 kann ohne eigenständige Tabelle durch die beiden
 Tabellen der zugeordneten Entity-Typen ausgedrückt werden, indem der
 Primärschlüssel der einen Tabelle als Fremdschlüssel in die andere Tabelle
 eingebracht wird.

- 9.8.6 Jeder Entity-Typ einer Generalisierungshierarchie verlangt eine eigenständige Tabelle, wobei der Primärschlüssel der übergeordneten Tabelle auch Primärschlüssel der untergeordneten Tabelle wird.
- 9.8.7 Bei einer Aggregation müssen sowohl die Entity-Typen als auch der Beziehungstyp als je eigenständige Tabelle definiert werden, falls der Beziehungstyp N:M ist. Im Falle einer 1:N-Beziehung kann der Entity-Typ mit dem Beziehungstyp in einer Tabelle kombiniert werden
- 9.8.8 Integritätsbedingungen
 - Legen explizit oder implizit Regeln für Relationen oder Beziehungen zwischen Relationen fest.
 - Es gibt nicht zwei identische Tupel in einer Tabelle. Relation ist Menge
 - Jeder Wert eines Attributs gehört zu einem definierten Wertebereich. Für jedes Attribut wird ein Datentyp festgelegt.
 - Jedes Tupel muss eindeutig identifizierbar sein. Festlegung eines Primärschlüssels
 - Alle Attribute eines Fremdschlüssels tauchen in einer anderen Relation als Werte des Schlüssels auf
- 9.9 Begriffe und Erklärungen
 - Attribut ist eine Spalte einer Tabelle
 - Wertebereich/Domäne sind die möglichen Werte eines Attributes
 - Attributwert ist ein Element eines Wertebereichs
 - Relationenschema ist die Menge von Attributen
 - Relation ist eine Menge von Zeilen einer Tabelle
 - Tupel ist die Zeile einer Tabelle
 - Datenbankschema ist die Menge von Relationenschemata
 - Datenbank ist eine Menge von Relationen
 - Schlüssel ist eine minimale eindeutig identifizierende Attributmenge
 - Primärschlüssel ist ein beim Datenbankentwurf ausgezeichnete Schlüssel
 - Fremdschlüssel ist eine Attributmenge, die in einer anderen Relation Schlüssel ist
- 9.10 Operationen im relationalen Modell

Operationen müssen irgendwie benannt werden können, deshalb gibt es Sprachen, in denen Operationen beschrieben werden

 - 9.10.1 Operationsarten
 - Datenverwaltung (Datendefinition; Datenmanipulation; Zugriffs-, Integritäts- und Transaktionskontrolle)
 - Nutzung (Anfragen (engl. Queries))
 - 9.10.2 Sprachentypen
 - Relationenalgebra
 - Prozedurale Sprache, in der spezifiziert wird, wie eine Anfrage auszuwerten ist
 - (Mathematische) Operationen, die auf einer Relation möglich sind
 - Relationen sind Mengen, genauer gesagt Teilmengen, eines kartesischen Produktes
 - Man kann die typischen Mengenoperationen Vereinigung, Durchschnitt und Differenz definieren
 - Selektion
 - Auswahl von Zeilen (Tupeln) einer Relation über Prädikate (boolesche Ausdrücke, bestehend aus Operanden, Vergleichsoperatoren und logischen Operatoren)
 - => Selektion enthält all die Tupel aus R, für die das Prädikat den Wert wahr ergibt
 - Projektion
 - Auswahl von Spalten (Attribute) einer Relation mit Grad $n \geq k$
 - => Projektion enthält die Spalten aller Tupel der Relation. Das Ergebnis ist wieder eine Relation (also eine Menge) und enthält daher keine Duplikate

- Relationenkalkül
Deklarative Sprache, in der spezifiziert wird, welche Daten man erhalten will

9.10.3 "Klassische" Mengenoperationen

Vereinigungsverträglichkeit der beteiligten Relationen muss gegeben sein:

- gleicher Grad der Relationen, d.h. dieselbe Anzahl Attribute
- gleiche Bereiche, d.h. dieselben Domänen für die Attribute

Vereinigung (Union), Differenz, Durchschnitt und symmetrische Differenz

9.10.3.1 Kartesisches Produkt

Das kartesische Produkt zweier Mengen ergibt sich, indem man alle möglichen Paare bildet, deren erstes Element aus der einen Menge und deren zweites Element aus der anderen Menge stammt.

9.10.4 Umbenennung

Manchmal müssen Relationen oder Attribute umbenannt werden, damit Namenskonflikte aufgelöst werden.

Qualifizierte Attributnamen bestehen aus Relationenname und Attributname, getrennt durch einen Punkt, z.B. Student.Name, Professor.Name

9.10.5 Verbund (engl. Join)

Ein Verbund ist ein kartesisches Produkt zwischen zwei Relationen R mit Grad r und S mit Grad s, eingeschränkt durch eine Θ -Bedingung zwischen zwei Spalten i und j (Θ ist ein Vergleichsoperator aus $<, =, >, \leq, \neq, \geq$)

Ein Θ -Verbund zwischen R und S: $V = R_{A_i \Theta A_j} \bowtie S$

Für $\Theta = "="$ spricht man auch vom Gleichverbund (Equi-Join)

9.10.6 Natürlicher Verbund (engl. Natural Join)

Es werden diejenigen Tupel aus den Relationen R und S kombiniert, für die die Werte der Attribute gleichen Namens übereinstimmen. Im Ergebnis sind diese Attribute nur einmal vorhanden.

Beispiel:

Gegeben sind $R(A_1, A_2, \dots, A_m, B_1, B_2, \dots, B_n)$ und $S(B_1, B_2, \dots, B_n, C_1, C_2, \dots, C_k)$

$\pi_{A_1, A_2, \dots, A_m, R.B_1, \dots, R.B_n, C_1, \dots, C_k}(\sigma_{R.B_1=S.B_1 \wedge \dots \wedge R.B_n=S.B_n}(R \times S))$

9.10.7 Outer Join

Bisher konnten Joins unvollständig sein, d.h. Tupel aus einer Relation waren im Join nicht vertreten, wenn sie keinen Partner gefunden hatten.

Ein Outer Join nimmt auch partnerlose Tupel auf.

Outer Join Typen:

- linker äußerer Join (alle Tupel der linken Relation bleiben erhalten)
- rechter äußerer Join (alle Tupel der rechten Relation bleiben erhalten)
- (vollständiger) äußerer Join (alle Tupel bleiben erhalten)

9.11 Operatorbaum

In einem Operatorbaum kann man sich veranschaulichen, wie und in welcher Reihenfolge die Operatoren ausgeführt werden.

9.12 Optimierungsregeln für Anfragen

- Führe Selektionen so früh wie möglich aus
- Führe Projektionen frühzeitig aus
- Fasse einfache Selektionen auf einer Relation zusammen
- Verknüpfe bestimmte Selektionen mit einem vorausgehenden kartesischen Produkt zu einem Verbund
- Bestimme die Verbundreihenfolge so, dass die Anzahl und die Größe der Zwischenobjekte minimiert sind

9.13 Relationenkalkül

Der Benutzer gibt die Definition einer Relation an, die aus anderen abgeleitet werden soll (nicht die dazu notwendigen Operationen)

9.13.1 Allgemeiner Ausdruck

$\{t \mid q\}$ (mit t als Relationenschema der Ergebnisrelation und q als Prädikat zur Beschreibung der Eigenschaften der Tupel in der Ergebnisrelation)

q ist ein logischer Ausdruck aus Attributnamen, Konstanten, Vergleichsoperatoren, Booleschen Operatoren, Quantoren und Variablen. In q wird insbesondere festgelegt, aus welchen Ausgangsrelationen die Ergebnisrelation erzeugt werden soll.

Ergebnis der Anfrage ist diejenige Menge von Tupeln, die die im Prädikat beschriebenen Bedingungen erfüllen.

9.13.2 Beispiele für Bestandteile von Kalkülausdrücken

t Tupelvariable

$t \in R$ ein beliebiges Tupel t der Relation R

$t.E_i$ Wert des Tupels t für das Attribut E_i

$(\exists t \in R) q(\dots)$ Existenzquantor: Es existiert ein Tupel in der Relation R mit der Eigenschaft $q(\dots)$

$(\forall t \in R) q(\dots)$ Allquantor: Für alle Tupel der Relation R gilt, dass sie die Eigenschaft $q(\dots)$ besitzen

$\{(t.a, t.b) \mid t \in R\}$ Projektion der Tabelle R auf die Spalten a und b

$\{t \mid t \in R \text{ und } t \text{ erfüllt Bedingung(en)}\}$ Selektion der Spalten aus R , für die eine Bedingung erfüllt ist

9.14 Erweiterungen der relationalen Basissprachen

Relationenalgebra und Relationenkalkül sind Grundlage vor allem für den Abfrageteil einer relationalen Datenbanksprache.

Manipulationen wie Einfügen, Löschen, Verändern von Tupelmengen erfordern weitere Funktionen:

- Definition von Tabellen
- Einfügen, Löschen, Verändern
- Aggregierungsfunktionen wie Anzahl der Tupel einer Relation, Summenbildung, Maximumbestimmung, Minimal- und Durchschnittswerte einzelner Spalten
- Formatierung von Tabellen und Ausdrücken nach verschiedenen Kriterien, Sortierreihenfolgen, Gruppenwechsel zum Darstellen von Tabellen
- Berechnung arithmetischer Ausdrücke

9.15 Backus-Naur-Form (BNF)

Die BNF erlaubt die Beschreibung von kontextfreien Grammatiken

9.15.1 Bestandteile einer kontextfreien Grammatik

- Nicht-Terminal-Symbole werden durch andere Zeichen ersetzt
- Terminal-Symbole werden nicht mehr ersetzt
- Regeln beschreiben, wie ersetzt wird
Regeln bestehen aus einer linken und einer rechten Seite
 $\text{linkeSeite (Nicht-Terminal-Symbol)} ::= \text{rechteSeite}$
(Mischung aus Nicht-Terminalen und Terminalen)
- Startsymbol legt fest, bei welchem Nicht-Terminal die Ersetzung beginnt

9.15.2 Aufbau der rechten Seite einer Regel

- Terminal
- Sequenz (hintereinander vorkommende Bestandteile)
 $L ::= R_1 R_2$
- Optionales (ein Bestandteil kann vorkommen, muss aber nicht)
 $L ::= [R]$
- Auswahl (nur einer der Bestandteile darf vorkommen)
 $L ::= R_1 \mid R_2$
- Wiederholung (der Bestandteil darf mehrmals hintereinander vorkommen)
 $L ::= \{R\}$
- Klammerung (Sicherstellen der eindeutigen Interpretation der Regeln)

9.16 Normalisierung

Der Zweck der Normalisierung ist es, gute von weniger guten Relationenschemata zu unterscheiden. Schlechte Relationenschemata führen zu Anomalien, die zu verhindern sind.

9.16.1 Anomalien sind:

- nicht normales Verhalten
- Verhalten ist anders, als man es sinnvollerweise erwarten würde

9.16.2 Typische Anomalien sind:

9.16.2.1 Einfüge Anomalie

Entsteht, wenn man Informationen zweier Entitätstypen miteinander vermischt. Das Einfügen eines einzelnen Entitys wird dann schwierig, man kann nicht das eine Entity einfügen, ohne ein anderes angelegt zu haben (z.B. Professor → Vorlesung)

9.16.2.2 Update Anomalie

Entsteht, wenn man zwei verschiedene Entitätstypen miteinander verknüpft. Das Aktualisieren eines Entitys erfordert viel mehr Aktionen/Operationen als man sinnvollerweise braucht (z.B. Ändern des Alters einer Person)

9.16.2.3 Lösch Anomalie

Entsteht, wenn man zwei verschiedene Entitätstypen miteinander verknüpft. Das Löschen eines Entitys löscht möglicherweise auch Informationen eines anderen Entitys (z.B. werden beim Löschen einer Vorlesung auch der dazugehörige Professor gelöscht)

9.16.3 Redundanz

Ein Attribut einer Relation ist redundant, wenn einzelne Werte dieses Attributes innerhalb der Tabelle ohne Informationsverlust weggelassen werden können.

9.16.4 Anforderungen an den Entwurf von Relationenschemata

- die Bedeutung lässt sich leicht erklären
Es sollte keine Attribute aus mehreren Entity-Typen oder Beziehungstypen enthalten
- Einfüge-, Lösch- oder Update-Anomalien vermeiden
Falls Anomalien auftreten, müssen sie identifiziert werden und es muss sichergestellt sein, dass die Programme, die die Datenbank aktualisieren, korrekt arbeiten
- möglichst keine Attribute, die oft Nullwerte annehmen

9.16.5 Konsequenzen aus den Anforderungen

Prozess bei der Normalisierung:

1. Entwurf eines ER-Diagramms (Entity-Typen und Beziehungstypen)
2. Transformation in Relationenschema (Entity-Typ → Tabelle, Beziehungstyp → Tabelle, evtl. 1:1 und 1:n Beziehungstypen in eine beteiligte Entity-Tabelle integrieren)
3. Überprüfung jeder Tabelle des Relationenschemas und gegebenenfalls Aufteilung einer Relation in mehrere Relationen

9.17 Funktionale Abhängigkeit

Eine funktionale Abhängigkeit β ist funktional abhängig von α zwischen zwei Attributmengen α und β und schreibt man als $\alpha \rightarrow \beta$.

9.17.1 Bedeutung

Aus den Werten von α kann man immer eindeutig auf die Werte von β schließen (für alle Tupel einer Relation)

Alternative Erklärungen:

- Es können keine Tupel in der Relation vorkommen, die für α dieselben Werte haben, dann aber für β verschiedene Werte annehmen
- Zwei Tupel, deren Komponenten in α übereinstimmen, stimmen auch in β überein.

9.17.2 Eigenschaften

- triviale funktionale Abhängigkeit: $\alpha \rightarrow \alpha$
- für jeden Schlüsselkandidaten α gilt: $\alpha \rightarrow x$ (x ist beliebiges Attribut der Relation)
- funktionale Abhängigkeiten werden nicht durch Analyse einer Relation gewonnen, sondern sie werden vom Entwerfenden festgelegt
- funktionale Abhängigkeiten müssen jederzeit erfüllt sein, nicht nur für einen konkreten Zustand der Tabelle
- funktionale Abhängigkeiten sind Integritätsbedingungen

9.17.3 Hülle der funktionalen Abhängigkeiten

F sei die Menge der funktionalen Abhängigkeiten (functional dependencies)

F wird vom Schemadesigner spezifiziert

F impliziert $X \rightarrow Y$, wenn $X \rightarrow Y$ in allen Relationen eines Relationenschemas gültig ist, in denen auch F gültig ist.

Die Menge F^+ aller funktionalen Abhängigkeiten, die von F impliziert werden, heisst Hülle von F (engl. closure).

9.18 Volle funktionale Abhängigkeit

Falls $X \rightarrow Y \in F^+$, kann Y auch nur von einer Teilmenge der Attribute, die X bilden, funktional abhängig sein.

Für eine Fd-Menge F und eine funktionale Abhängigkeit $X \rightarrow Y \in F^+$ heisst Y voll funktional abhängig von X genau dann, wenn es keine echte Teilmenge X' von X gibt, sodass $X' \rightarrow Y \in F^+$.

9.18.1 Beispiel

ADRESSE ist voll funktional abhängig von {KDNR, ZEITSCHRIFT}

PREIS ist voll funktional abhängig von ZEITSCHRIFT

PREIS ist nicht voll funktional abhängig von {KDNR, ZEITSCHRIFT}

9.18.2 Neue Definition Schlüssel

$X \in R$ ist Schlüssel von $R=\{A_1, A_2, \dots, A_n\}$ genau dann, wenn

- $X \rightarrow \{A_1, A_2, \dots, A_n\} \in F^+$ und

- $\{A_1, A_2, \dots, A_n\}$ ist voll funktional abhängig von X (das heisst, X kann nicht verkleinert werden)

A heisst Schlüsselattribut des Relationenschemas R , wenn A Element irgendeines Schlüssels von R ist. Andernfalls heisst A Nichtschlüsselattribut. Jeder Schlüssel von R heisst Kandidatenschlüssel oder Schlüsselkandidat von R .

Ein Schlüssel wird als Primärschlüssel ausgezeichnet.

9.19 Normalformen

9.19.1 Erste Normalform

Eine Relation ist in erster Normalform, wenn alle Attribute einen atomaren Wertebereich haben.

9.19.1.1 Konsequenzen

- keine zusammengesetzten Attribute
- keine mengenwertigen Attribute
- alle Relationen, die wir bisher betrachtet haben, waren in erster Normalform
- erste Normalform ist Bestandteil der Relationendefinition

9.19.2 Zweite Normalform

Eine Relation ist in zweiter Normalform, wenn jedes Nichtschlüsselattribut voll funktional abhängig von jedem Schlüssel ist.

Ist immer dann verletzt, wenn der Wert eines Nichtschlüssel-Attributs funktional von einer Teilmenge eines Schlüssels abhängt.

Tritt auf, wenn mehr als ein Konzept in einer Relation modelliert wird.

9.19.2.1 Konsequenzen

- Redundanz tritt auf
- Anomalien treten auf

9.19.3 Dritte Normalform

Ein Relationenschema R mit Fd-Menge F ist in dritter Normalform, wenn für alle $X \rightarrow A \in F^+$ mit $A \notin X$ gilt:

X enthält einen Schlüssel für R oder A ist Schlüsselattribut.

9.13.3.1 Äquivalente Definition

Ein Relationenschema R mit Fd-Menge F ist nicht in dritter Normalform, wenn es eine funktionale Abhängigkeit $X \rightarrow A$ gibt, so dass X keinen Schlüssel enthält und A Nichtschlüsselattribut ist.

9.13.3.2 Konsequenzen aus "nicht in dritter Normalform":

- Redundanz tritt auf
- Anomalien treten auf

9.19.4 Vierte Normalform (Boyce-Codd Normalform)

Allgemein gilt: Jede Relation die in (i)-ter Normalform ist, ist automatisch auch in (i-1)-ter Normalform.

9.19.5 Zusammenfassung der Normalformen

1NF: Nur atomare Attribute

2NF: Keine Abhängigkeiten von einem Teil des Schlüssels

3NF: Keine Abhängigkeit eines Nichtschlüsselattributs von einer Attributmenge, die keinen Schlüssel enthält

Zerlegungen können Normalformen erzwingen

9.20 Zerlegungen

Nicht jede Zerlegung von einer Tabelle in zwei (oder mehr) neue Tabellen ist erlaubt und macht Sinn. In einem Entwurfsprozess sollen weder Attribute verloren gehen noch hinzugefügt werden und funktionale Abhängigkeiten auf die entstehenden Schemata vererbt werden.

9.20.1 Anforderungen an vernünftige Zerlegungen von Relationen

- Verlustlosigkeit, d.h. alle Informationen müssen sich rekonstruieren lassen
Die Zerlegung einer Tabelle R in Tabellen R_1 und R_2 ist verlustlos, wenn für jede gültige Ausprägung R gilt: $R = R_1 \bowtie R_2$
- Abhängigkeitsbewahrung, d.h. alle Abhängigkeiten aus der großen Relation sind auch in den zerlegten (kleineren) Relationen vorhanden

10. Indexe

Unter dem Index eines Attributes versteht man eine Zugriffsstruktur, die in einer bestimmten Reihenfolge für jeden Attributwert die internen Adressen der Datensätze liefert, die diesen enthalten.

Stichwortverzeichnis eines Buches: Auf jedes Stichwort - in alphabetischer Reihenfolge aufgeführt - folgen die Nummern der Seiten, auf denen es im Text vorkommt.

Zu jedem Namen einer Tabelle wird in alphabetischer Reihenfolge in einer Indexstruktur (für den Benutzer verborgen) entweder der Identifikationsschlüssel oder die interne Adresse der Tupel festgehalten. Das Datenbanksystem benutzt bei einer entsprechenden Anfrage oder bei einem Verbund diesen Index der Namen. Option UNIQUE überwacht die Eindeutigkeit.

10.1 Richtlinien für Indizes

Eine Spalte sollte indiziert werden, wenn sie

- häufig zum Suchen und Sortieren verwendet wird
- zum Verknüpfen von zwei oder mehreren Tabellen verwendet wird
- Eindeutigkeit sichergestellt werden soll

Eine Reihe sollte nicht indiziert werden, wenn sie

- weniger als 200 Zeilen enthält und sie nicht für einen Verbund verwendet wird
- eine große Anzahl mehrfacher Werte enthält (extrem zum Beispiel Boolesche Variable)

11. Integritätsbedingungen in SQL

11.1 Referentielle Integrität durch Fremdschlüssel

Forderung: Jeder Wert eines Fremdschlüssels muss auch in der referenzierten Tabelle vorkommen

Lösung: Im create table statement wird der Fremdschlüssel als solcher markiert (foreign key). Das DBMS prüft nun bei allen Operationen auf den beteiligten Tabellen, ob die Forderung noch erfüllt ist.

11.2 Mehrere Schlüsselkandidaten

Wenn in einer Tabelle mehrere Schlüsselkandidaten existieren, dann kann man der Datenbank dies zusätzlich bekannt machen (UNIQUE).

Vorteil: Das DBMS erkennt versehentlich falsche Eingaben und kann diese zurückweisen.

11.3 Weitere Integritätsbedingungen für Tupel

Forderung: Bestimmte Attribute dürfen nicht alle Werte einer Domäne annehmen oder zwischen Attributen soll immer ein bestimmter Zusammenhang bestehen

Lösung: Im create table statement wird eine check-Bedingung formuliert

Man kann den Integritätsbedingungen (engl. integrity constraints) auch Namen geben und die Bedingungen später dynamisch ein- oder ausschalten.

11.4 Reaktionen auf Integritätsverletzungen

Würde beim Einfügen eines neuen Tupels eine Integritätsverletzung auftreten, dann wird die Einfügeoperation abgebrochen.

Entsteht beim Ändern oder Löschen von Tupeln eine Integritätsverletzung, dann sind auch andere Reaktionen möglich:

- Fortschreiben der Änderung (engl. propagate, cascade), so dass die Integritätsverletzung dadurch aufgehoben wird => cascade
- Attribute auf null oder ihren Defaultwerten setzen => set null oder set default
- Abbruch der Operation (wie beim Einfügen). Das ist auch die Defaultreaktion => no action

Diese Reaktionen können in Abhängigkeit von der Operation (Ändern oder Löschen) spezifiziert werden. Häufigstes Einsatzgebiet sind die foreign key constraints.