Databases

Susanne Bleisch

23. Juni 2011

Inhaltsverzeichnis

1 Einführung in Datenbanksysteme

Datenverwaltung und insbesondere die Verwaltung von *Geodaten*[1] ist grundsätzlich nicht an eine bestimmte Technologie gebunden. Denkbar wären somit analoge Planarchive oder auch dateibasierte Ablagesysteme. Der Begriff Geoinformationssystem impliziert jedoch gewisse Anforderungen, die weit über das Speichern, Archivieren und Abrufen von Daten hinausgehen und nur mit Datenbanksystemen sinnvoll befriedigt werden können.

Der Fokus dieser Lektion und des gesamten Moduls liegt deshalb auf datenbankbasierten Konzepten und Architekturen. Nach der Einführung und Definition einiger wichtiger Begriffe in der Unit „Begriffsdefinitionen (Seite 7)“wenden wir uns den spezifischen Eigenschaften des Datenbankansatzes (Unit „Eigenschaften des Datenbankansatzes (Seite 12)“) zu. Die Betrachtung verschiedener Anwendungsgebiete in der Unit „Anwendungsgebiete (Seite 20)“soll ermöglichen, verschiedene Einsatzgebiete von Datenbanken kennen zu lernen und die Datenbankeigenschaften zu vertiefen. Ein Vergleich des Datenbankansatzes mit dateibasierten Lösungsansätzen erfolgt in der Unit „Vorteile und Nachteile (Seite 29)“.

Lernziele

- Sie sind mit der Begriffswelt im Umfeld Daten, Information und Datenverwaltung vertraut und können die wichtigsten Begriffe erklären.
- Sie kennen und verstehen die charakteristischen Eigenschaften von Datenbanksystemen und können diese auf Anwendungen im eigenen Umfeld übertragen.

[1] Unter Geodaten oder raumbezogenen Daten versteht man Datenobjekte, die durch eine Position im Raum direkt oder indirekt referenzierbar sind. Der Raum ist dabei definiert durch ein Koordinatensystem, das den Bezug zur Erdoberfläche herstellt. Geodaten werden in der Regel grafisch in Papierform oder an grafikfähigen Bildschirmen präsentiert. Aus informationstechnischer Sicht kann man die Daten, die wiederum zu Geodaten gehören, einteilen in: •Geometrie (Position und geometrische Ausprägung) •Topologie (explizit gespeicherte geometrisch-topologische Beziehungen) •Präsentation (graphische Ausprägungen wie Signaturen, Farbe, Typographie) •Sachdaten (alphanumerische Daten zur Beschreibung der Semantik). Geodaten stellen in der Informationsverarbeitung eine besondere Herausforderung dar. Die Gründe dafür sind: •der hohe Aufwand für die Erfassung •die große Menge der anfallenden Daten •die geforderten Antwortzeiten beim Zugriff auf Geodaten •die Verarbeitung nach räumlichen Kriterien sowie •die Komplexität der Beziehungen.

1.1 Begriffsdefinitionen

Von Daten zu nutzbaren Informationen

Bevor wir uns mit dem Einsatz und der Architektur von Datenverwaltungslösungen beschäftigen, wollen wir uns mit Begriffen wie Information, Daten und Datenbanksystemen auseinandersetzen - dem Basisvokabular für unser Modul. Viele dieser Begriffe werden zwar täglich verwendet, oft aber in einem falschen Zusammenhang.

1.1.1 Daten versus Informationen

Daten (Computerdaten): Die Darstellung von Tatsachen (zum Beispiel Messwerte) oder Konzepten, die in einer durch den Computer lesbaren Form erzeugt oder in eine entsprechende Form gebracht werden.

Information: Information ist nutzbare Antwort auf eine konkrete Fragestellung [**?**] . Information wird meistens aus Daten extrahiert oder abgeleitet.

Von Information wird gesprochen, wenn auf eine spezifische Frage eine Antwort gegeben wird, die das Verständnisniveau der Fragenden erhöht und sie befähigt, einem bestimmten Ziel näher zu kommen. [**?**]

Information hat die folgenden Aspekte:

- *strukturelle*[2] und *syntaktische*[3]
- *semantische*[4] (inhaltliche)
- *pragmatische* (anwendungsrelevante)

Das Verhältnis von Daten und Information

Die Begriffe Daten und Information werden sehr oft vermischt und im falschen Zusammenhang eingesetzt. Daher sollen nachfolgend ein paar Unterscheidungsmerkmale aufgeführt werden obwohl es nicht möglich ist strikt zwischen Daten und Information zu trennen:

- Semantische Aspekte (die Bedeutung) werden in Daten oft codiert, wobei diese Codes nach vorgängig vereinbarten Konventionen definiert und interpretiert werden müssen (z. B. Notenskala von 1 bis 6 mit der Konvention 6 = sehr gut).

[2] Die Struktur ist der abstrakte innere Aufbau.

[3] Syntax ist id Lehre vom Satz, Lehre von den Regeln, wonach in Sprache(n) aus den Wörtern zusammengehörige Wortgruppen gebildet werden, und die Lehre von der Kombination von Wörtern zu Sätzen. „Syntaxis“heisst auf Griechisch Zusammenstellung, Anordnung, Aneinanderreihung.

[4] Die Informationstheorie unterscheidet drei Dimensionen einer Information, die syntaktische, die semantische und die pragmatische. Nehmen wir als Beispiel eine Verkehrsampel. In der syntaktischen Dimension unterscheiden wir die drei Farben Rot, Gelb und Grün. Ihren Sinn bekommt die Ampel aber erst in der semantischen Dimension. Hier wird den Farben ihre Bedeutung zugeordnet. Rot hat die Bedeutung Halt, grün bedeutet freie Fahrt. Erst in der pragmatischen Dimension aber wird das Ampelsignal für die Verkehrsregelung brauchbar. Pragmatisch gesehen bedeutet das rote Licht, als Autofahrer muss ich anhalten.

- Information muss in der Regel zuerst aus Daten rekonstruiert oder abgeleitet werden. Dies geschieht in Bezug auf eine bestimmte Fragestellung. Ein Beispiel: Die Daten enthalten Angaben zu den Niederschlagsmengen in jedem Monat während der letzten zehn Jahre. Jemand möchte wissen, wieviel es im Durchschnitt während dem Monat Juli geregnet hat (über die letzten zehn Jahre). Somit ist die durchschnittliche Niederschlagsmenge im Monat Juli der letzten zehn Jahre Information die die konkrete Fragestellung beantwortet. Aber die berechnete Durchschnittszahl ohne entsprechende Frage ist den Daten zuzuordnen.

- Daten enthalten in der Regel keine anwendungsrelevanten Aspekte (z. B. lassen sich aus den Koordinaten eines Grundstücks keine Informationen für unterschiedliche Anwendungen wie Steuern, Erschliessung, Überflutungsrisiken etc. ableiten).

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 1: Ein Brief in chinesischer Sprache

Bei einem Brief in fremder Sprache erkennen wir zwar die strukturellen und syntaktischen Aspekte, d. h. die Abschnitte, Sätze und Wörter, nicht aber den Sinn des Geschriebenen.

Auch ein Schreiben in der eigenen Sprache kann nicht in jedem Fall als Information bezeichnet werden. Wir mögen zwar dessen Inhalt (*Semantik*) verstehen, wenn dieser für uns aber irrelevant oder uninteressant ist, dann fehlt der wichtige Aspekt des Nutzens.

1.1.2 Die Komponenten eines Informationssystems

Aus *konzeptioneller*[5] Sicht hat ein Informationssystem einen schalenförmigen Aufbau.

Fahren Sie mit der Maus über die Begriffe in der nachstehenden Interaktion und erkennen Sie, aus welchen Teilen ein Informationssystem aufgebaut ist.

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 2: Aufbau eines Informationssystems

Die Datenverwaltungs-Komponenten eines Informationssystems sind:

Datenbasis und Datenbank: Eine Datenbasis ist eine Menge von Daten, die aus Sicht der Systembetreiber in irgendeiner Weise als zusammengehörig betrachtet werden. Angereichert um weitere Daten, die das DBMS zur Erfüllung seiner Aufgaben benötigt, bilden sie eine Datenbank (DB).

Datenbankverwaltungssystem: Datenbankverwaltungssysteme (engl. database management system oder DBMS) sind Softwareprodukte für die dauerhafte, integre und anwendungsunabhängige Speicherung und Verwaltung sowie die flexible und bequeme Verwendung von grossen, mehrfach benutzbaren integrierten Datenbasen.

Datenbanksystem: Ein Datenbanksystem (DBS) besteht aus einem DBMS und einer oder mehreren Datenbanken.

Datenbankverwaltungssysteme und Datenbanksysteme stehen im Zentrum dieses Moduls.

Informationssystem: Ein Informationssystem erweitert die Datenbank um eine Reihe von Werkzeugen (engl. software tools) zur Abfrage, Darstellung, Transformation und Analyse von Daten.

Gemäss der im ersten Teil der Unit behandelten Unterscheidung zwischen Daten und Informationen (Seite 8), bereichern die Werkzeuge eines Informationssytems die Daten um *semantische* und *pragmatische* bzw. anwendungsrelevante Aspekte.

[5] Nach Duden: die einem Werk zugrunde liegende Anschauung, Leitidee; für das Datenmodell gilt: unabhängig von einer bestimmten Technologie

Sicherlich haben Sie auch schon den Begriff Geoinformationssystem gehört und möglicherweise auch schon Definitionen dazu gelesen. Der nächste Abschnitt definiert den Begriff Geoinformationssystem nochmals zur Erinnerung und vergleicht diese Definition mit der des Informationssystems.

Geoinformationssystem: "*Ein Geoinformationssystem dient der Erfassung, Speicherung, Analyse und Darstellung aller Daten, die einen Teil der Erdoberfläche und die darauf befindlichen technischen und administrativen Einrichtungen sowie geowissenschaftliche, ökonomische und ökologische Gegebenheiten beschreiben.*" [**?**]

Diese Definition enthält die wichtigsten Aspekte der obigen Definition eines Informationssystems, jedoch mit einer Fokussierung auf Daten mit einem Raumbezug.

1.2 Eigenschaften des Datenbankansatzes

Der Datenbankansatz hat einige ganz charakteristische Eigenschaften, die in dieser Unit näher betrachtet werden.

In der Unit „Anwendungsgebiete (Seite 20)“werden verschiedene Datenbankanwendungen vorgestellt und anhand des jeweiligen Beispiels noch einmal auf die wichtigsten Eigenschaften des Datenbankansatzes verwiesen.

Der Vergleich zwischen dem dateibasierten Ansatz und dem Datenbankansatz ist in der Unit „Vorteile und Nachteile (Seite 29)“zu finden.

1.2.1 Mehrfachnutzung

Ein Datenbanksystem erlaubt mehreren Benutzern gleichzeitig den Zugriff auf eine Datenbank. Die Beantwortung unterschiedlicher Fragestellung (der diversen Benutzer) mit den gleichen (Basis-) Daten ist ein zentraler Aspekt eines Informationssystems.

Eine solche Mehrfachnutzung erhöht auch die Wirtschaftlichkeit eines Systems. Die Datenerfassung und -haltung ist nicht *redundant*[6], das System kann zentralisiert betreut und die Daten können einfacher aktualisiert werden. Ausserdem werden auf diese Weise die oftmals sehr teuren *(Geo-)Daten* besser genutzt.

Bei einer Mehrfachnutzung von Daten stellt sich aber die Frage, wie bei konkurrierenden Änderungen (z. B. gleichzeitiges Verändern der gleichen Daten von zwei verschiedenen Nutzern mit verschiedenen Anwendungen) vorgegangen werden soll. Ausserdem besteht auch ein grösseres Gefahren- bzw. Missbrauchspotenzial zum Beispiel im Bereich Datenschutz.

Datenbank-Änderungen werden im Fachjargon Transaktionen genannt und an anderer Stelle dieser Unit erklärt.

Ein Beispiel für Mehrfachnutzung ist die Reise-Datenbank eines grösseren Reisebüros. Die Angestellten verschiedener Zweigstellen können gleichzeitig auf diese Datenbank zugreifen und Reisen buchen bzw. verkaufen. Jeder sieht sofort wie viele Plätze für eine bestimmte Reise noch erhältlich sind oder ob diese schon ausgebucht ist.

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 3: Mehrfachnutzung der gleichen Daten

[6] Bezeichnet die mehrfache Speicherung gleicher Daten an verschiedenen Stellen (einer Datenbank).

1.2.2 Strukturierte und beschriebene Daten

Eine grundsätzliche Eigenschaft des Datenbankansatzes ist, dass ein Datenbanksystem nicht nur die Daten enthält, sondern auch eine komplette Definition oder Beschreibung der Daten. Diese Beschreibung besteht aus Angaben über den Umfang, die Struktur, die Art und das Format aller Daten sowie über die Beziehungen der Daten untereinander. Diese gespeicherten Informationen werden auch Metadaten („Daten-über-Daten“) genannt.

Metadaten[7] werden von der DBMS-Software, aber auch von Anwendungsprogrammen (z. B. GIS) und Anwendern einer Datenbank verwendet. Da DBMS-Software nicht für eine spezielle Datenbankanwendung geschrieben ist, muss sie die Metadaten einer Datenbank verwenden, um deren Umfang, Struktur etc. zu kennen.

Hier nun ein einfaches Beispiel dafür, wie in einer Datenbank Daten beschrieben sein können.

Nachstehend ist eine Datenbanktabelle aufgeführt. Durch die Struktur dieser Tabelle (1. Spalte = Vorname, 2. Spalte = Name, 3. Spalte = PLZ, 4. Spalte = Wohnort) weiss man, dass ein Eintrag in der ersten Spalte ein Vorname als Zeichenkette (String) und ein Eintrag in der dritten Spalte eine Postleitzahl (Nummer) sein muss.

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 4: Beispiel einer Datenbanktabelle

[7] Meta ist ein Präfix, das in den meisten informationstechnologischen Anwendungen „eine zugrundeliegende Definition oder Beschreibung“ meint. Metadaten sind also eine Definition oder Beschreibung von Daten. Man spricht auch von „Daten über Daten“.

1.2.3 Trennung von Daten und Anwendungen

Wie in der Eigenschaft „Strukturierte Daten (Seite 14)“beschrieben, wird die Struktur einer Datenbank durch die *Metadaten*, die ebenfalls in der Datenbank abgelegt sind, beschrieben.

Das Anwendungsprogamm benötigt keine Kenntnis über die physikalische Datenspeicherung (Codierung, Format, Speicherort etc.). Es kommuniziert mit dem Verwaltungssystem einer Datenbank (DBMS) über eine normierte Schnittstelle mittels einer standardisierten Sprache (z. B. SQL). Den Zugriff auf die eigentlichen Daten und die *Metadaten* übernimmt dabei das DBMS.

Auf diese Weise können die Anwendungen völlig von den Daten getrennt werden, und datenbank-interne Effizienzverbesserungen und Reorganisationen haben keinen Einfluss auf die Anwenderprogramme.

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 5: Trennung der Anwendungen von den Daten

1.2.4 Datenintegrität

Datenintegrität ist ein Begriff für die Qualität und Zuverlässigkeit von Daten eines Datenbanksystems. Im weiteren Sinne zählt zur Integrität auch der Schutz der Datenbank vor unberechtigtem Zugriff (Vertraulichkeit) und Veränderungen.

Daten widerspiegeln Sachverhalte der realen Welt. Logischerweise wird verlangt, dass sie dies korrekt tun. Ein DBMS soll Unterstützung bieten bei der Aufgabe, nur korrekte und widerspruchsfreie („konsistente“) Daten in die Datenbank gelangen zu lassen. Ausserdem wird mit korrekten Transaktionen (Seite 17) die Konsistenz auch während des Systembetriebs aufrechterhalten.

In derselben Datenbank dürfen keine widersprüchlichen Aussagen stehen.

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 6: Zwei Datenbanktabellen mit widersprüchlichen Datensätzen

Diese Datenbank besteht aus den zwei Tabellen „Wohnstrassen“und „Strassenplanung“. In der Tabelle „Wohnstrassen“ist die Gartenstrasse von 19.00-7.00 mit einem Fahrverbot belegt. In der Tabelle „Strassenplanung“jedoch bestehen für die Gartenstrasse keine Einschränkungen. Dies ist ein Beispiel für einen Widerspruch, wie er in einer Datenbank nicht vorkommen darf.

Solche Widersprüche können verhindert werden, indem die Datenstruktur sorgfältig modelliert wird. In diesem Fall müsste man schauen, wie man die Informationen über die Gartenstrasse in einer Tabelle zusammenfassen könnte. Dies erleichtert auch die Nachführung, da dann nur noch eine Tabelle geändert werden muss, wenn sich an den Informationen zur Gartenstrasse etwas ändert.

1.2.5 Transaktionen

Eine Transaktion ist ein Bündel von Aktionen, die in der Datenbank durchgeführt werden, um diese von einem konsistenten Zustand wieder in einen konsistenten (widerspruchsfreien) Zustand zu überführen. Dazwischen sind die Daten zum Teil zwangsläufig inkonsistent.

Eine Transaktion ist atomar, d. h. nicht weiter zerlegbar. Innerhalb einer Transaktion werden entweder alle Aktionen oder keine durchgeführt. Nur ein Teil der Aktionen würde zu einem inkonsistenten Datenbankzustand führen.

Ein Beispiel einer Transaktion ist das Verschieben einer bestimmten Summe Geld von einem Konto auf ein anderes. Die Abbuchung des Geldes von einem Konto und die Gutschrift auf dem anderen Konto machen zusammen eine konsistente Transaktion aus. Diese Transaktion ist ausserdem atomar. Die Abbuchung oder die Gutschrift alleine würde zu einem inkonsistenten Zustand führen. Nach Abschluss der Transaktion (Abbuchung und Gutschrift) wird die Änderung an beiden Konti dauerhaft, und der Geldgeber sieht nun einen kleineren Kontostand, während der Empfänger des Geldes sich über seinen höheren Kontostand freuen kann.

Probieren Sie es aus, indem Sie mit den Buttons unten links durch die einzelnen Schritte dieses Beispiels navigieren.

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 7: Transaktion am Beispiel einer Geldverschiebung von einem Konto auf ein anderes

1.2.6 Datenpersistenz

Datenpersistenz meint, dass in einem DBMS einzelne Daten solange aufbewahrt werden müssen, bis sie explizit gelöscht werden. Die Lebensdauer von Daten muss also von den Benutzern direkt oder indirekt bestimmbar sein und darf nicht von irgendwelchen Systemgegebenheiten abhängen. Ebensowenig dürfen einmal in die Datenbank aufgenommene Daten verloren gehen.

Änderungen, die eine Transaktion (Seite 17) in einer Datenbank vornimmt, sind dauerhaft. Wenn die Transaktion abgeschlossen ist, kann auch ein darauf folgender Systemabsturz die Daten nicht mehr gefährden.

1.2.7 Datensichten

Typischerweise hat eine Datenbank mehrere Benutzer, und jeder benötigt, je nach Zugangsrechten und Bedürfnissen, eine individuelle Ansicht der Daten (Inhalt und Form). Eine solche Datensicht kann aus einer Teilmenge der gespeicherten Daten oder aus daraus abgeleiteten (nicht explizit gespeicherten) Daten bestehen.

Beispiel: An einer Hochschule werden Daten über die Studierenden verwaltet. Neben Matrikelnummer, Name, Adresse etc. werden auch andere Informationen erfasst, wie die Semesterzugehörigkeit oder die Angabe darüber, ob der oder die Studierende das Semester repetiert.

Diese umfassende Datenbank wird von Personen mit unterschiedlichen Bedürfnissen und Berechtigungen benützt.

Klicken Sie auf einen der vier Buttons, um die verschiedenen Sichten auf die Datenbank dargestellt zu bekommen.

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 8: Verschiedene Datensichten

Der Datenbankadministrator hat eine Sicht auf die gesamte Datenbank, während andere Nutzer in diesem Beispiel jeweils eine sehr eingeschränkte Sicht auf die Datenbank haben. Die Administration benötigt für die Erstellung einer Statistik über die Repetenten in den Abteilungen keine Namen oder Matrikelnummern. In einer Liste der Studierenden sollten heikle Daten wie der Repetentenstatus keinesfalls sichtbar sein. Der Dozent einer Klasse jedoch wünscht sehr detaillierte Auskunft über die Studierenden dieser Klasse.

1.3 Anwendungsgebiete

Aktuelle Informatiklösungen zeichnen sich aus durch viele verteilte Benutzer, die laufend aktualisierte Datenbestände gemeinsam nutzen möchten. Dafür werden in den verschiedensten Bereichen Datenbanksysteme eingesetzt, und es gibt kaum eine IT-Anwendung, die noch ohne sie auskommt.

Nachstehend sind einige Beispiele für den Einsatz von Datenbanksystemen aufgeführt.

Es wird jeweils kurz die Anwendung beschrieben und danach auf für diese Anwendung speziell wichtige Datenbankeigenschaften eingegangen. Die jeweiligen Eigenschaften sind mit den Beschreibungen derselben in der Unit „Eigenschaften des Datenbankansatzes (Seite 12)“verknüpft.

1.3.1 Verwaltung von Bankkonten

Die Verwaltung von Bankkonten ist eine anspruchsvolle Aufgabe, die schon längere Zeit Datenbanksysteme als Hilfsmittel verwendet. Heutzutage könnte sich vermutlich niemand mehr vorstellen, in der komplexen Finanzwelt zu operieren, ohne von Datenbanksystemen unterstützt zu werden.

Speziell wichtige Eigenschaften dieses Datenbanksystems:

Transaktionen (Seite 17)	Der erfolgreiche und korrekte Ablauf von Transaktionen ist ein wichtiger Punkt in der Bankkontenverwaltung. Man stelle sich nur vor, dass Gutschriften nicht richtig verbucht oder Abzüge mehrfach getätigt würden.
Datenintegrität (Seite 16)	Dabei spielt natürlich auch die Datenintegrität eine grosse Rolle. Es muss klar festgelegt sein, wie die Konsistenzbedingungen und -regeln aussehen und wie diese eingehalten werden können.
Datenpersistenz (Seite 18)	Als Bankkontenbesitzer ist es beruhigend zu wissen, dass die Persistenz der Daten gewährleistet ist. Daten werden nicht willkürlich gelöscht und gehen nicht auf mysteriöse Weise verloren.

1.3.2 Fahrplan-Informationssystem

Der Online-Fahrplan der SBB (Schweizerische Bundesbahnen) ist ein Beispiel eines Datenbanksystems aus dem Bereich des öffentlichen Verkehrs und eines darauf aufbauenden webbasierten Informationssystems. Seine Aufgabe ist es, die Benutzer jederzeit mit aktuellen und korrekten Informationen über optimale Verbindungen und den Zugbetrieb der SBB zu versorgen.

Speziell wichtige Eigenschaften dieses Datenbanksystems:

Mehrfachnutzung (Seite 13)	Das Datenbanksystem des SBB-Fahrplans kann von verschiedenen Anwendungen und Benutzern verwendet werden. Die Fahrplandaten werden SBB-intern verwendet, aber auch die Öffentlichkeit kann via Internet oder *WAP*[8] auf die Zugfahrpläne und weitere Auskünfte zugreifen.
Datenintegrität (Seite 16)	Die Benutzer des SBB-Fahrplans möchten immer eine aktuelle und korrekte Auskunft erhalten, dies stellt hohe Anforderungen an die Datenintegriät. Dafür müssen bei Fahrplan-, Gleis- oder anderen Änderungen die Daten des Datenbanksystems laufend aktualisiert werden.

Für ein solches Datenbanksystem, das in Spitzenzeiten deutlich mehr als eine Anfrage pro Sekunde erhält, ist neben den erwähnten Punkten auch die Performance (Leistung) eines Datenbanksystems eine speziell geforderte Eigenschaft.

Internet-Zugriff auf den SBB-Fahrplan: http://fahrplan.sbb.ch/ (fahrplan.sbb.ch/)

[8] Das Wireless Application Protocol - kurz: WAP - ist der wichtigste Standard, um Internetkommunikation und interaktive Dienste für Handys und andere mobile Endgeräte zu realisieren: Nachrichten aus dem Internet abrufen, Börsenkurse abfragen, Flugtickets buchen.

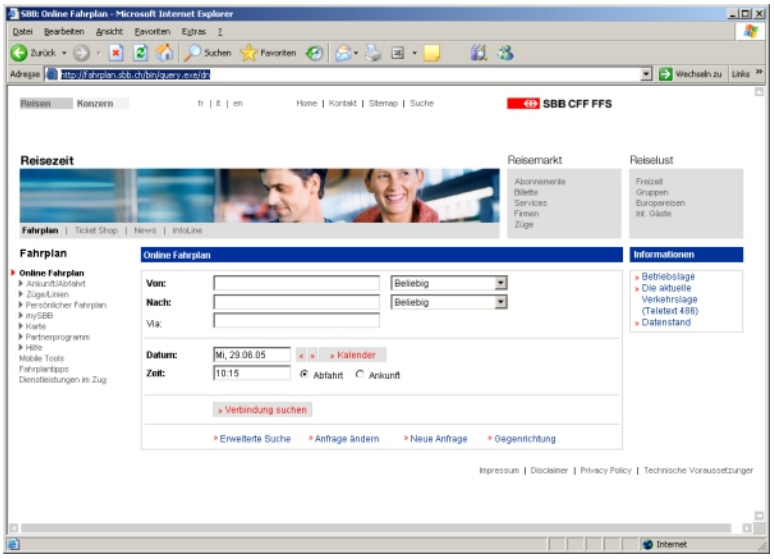

Abbildung 9: Screenshot SBB-Fahrplan-Informationssystem [**?**]

1.3.3 Bibliothekskatalog

Der Bibliothekskatalog NEBIS [**?**] ist der Katalog eines Netzwerks von Bibliotheken und Informationsstellen in der Schweiz. Auf diese Weise kann schweizweit in Bibliotheken nach bestimmten Büchern und Veröffentlichungen gesucht werden.

Speziell wichtige Eigenschaften dieses Datenbanksystems:

Strukturierte und beschriebene Daten (Seite 14)	Es ist mehr als hilfreich, wenn man sich bei der Erfassung und Nachführung von Tausenden von Büchern und Zeitschriften an eine klare und festgelegte Struktur halten kann. Die Beschreibung der Daten ermöglicht ausserdem eine gezielte Suche nach einzelnen Objekten. Eine saubere Struktur ist ausserdem redundanzfrei. Dies spart Arbeit, da schon eine minimale *Redundanz* zu einem Mehrfachem an Aufwand führen würde.
Datensichten (Seite 19)	Je nachdem, wie detailliert und spezialisiert eine Suche durchgeführt werden soll, braucht der Benutzer mehr oder weniger Informationen. So reichen für eine Grobsuche schon Titel und Autor, während für eine detailliertere Suche, zum Beispiel nach einer bestimmten Auflage eines Buches, schon mehr Informationen gewünscht sind. Zu diesem Zweck werden dem Benutzer verschiedene Datensichten zur Verfügung gestellt. Dazu kommen noch weitere Datensichten für die Verwalter des Katalogs.

Verschiedene Datensichten (für eine grössere Ansicht bitte auf das jeweilige Bild klicken)

Internet-Zugriff auf den Nebis-Bibliothekskatalog: http://www.nebis.ch/ (www.nebis.ch/)

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 10: Ergebnisliste der Suche [**?**]

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 11: Standardansicht eines ausgewählten Buches [**?**]

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 12: Katalogansicht eines ausgewählten Buches [**?**]

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 13: MARC'-Ansicht eines ausgewählten Buches [**?**]

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 14: Screenshot Bibliothekskatalog [**?**]

1.3.4 Zentrales Geodata-Warehouse

In städtischen und kantonalen Verwaltungen werden immer häufiger die raumbezogenen Daten aller Amtsstellen in einem zentralen Datenpool, dem sogenannten Geodata-Warehouse (einem raumbezogenen Datenbankverwaltungssystem), verwaltet und nachgeführt. Dies bringt enorme Einsparungen mit sich, da die Daten nun redundanzfrei sind und nur einmal nachgeführt werden müssen. Im bisherigen Betrieb wurden Daten häufig in den verschiedenen Ämtern mehrfach geführt, was deren Aktualisierung enorm erschwerte. Ausserdem müssen bei ämterübergreifenden Projekten die Geodaten nicht mehr mühsam zusammengesucht werden, sondern können einfach aus dem Datenpool geholt werden.

Speziell wichtige Eigenschaften dieses Datenbanksystems:

Mehrfachnutzung (Seite 13)	Ein zentrales Geodata-Warehouse ist in mehrfacher Hinsicht ein schönes Beispiel für die Mehrfachnutzung. Einerseits greifen mehrere Anwender darauf zu - die diversen Angestellten der verschiedenen Ämter. Andererseits werden für diese Zugriffe auch unterschiedliche Anwendungsprogramme (z. B. GIS-Systeme) verwendet. So kann es sein, dass das Forstamt mit der GIS-Software A auf die Daten zugreift, während das Vermessungsamt die GIS-Software B verwendet.
Trennung von Daten und Anwendungen (Seite 15)	Wie oben beschrieben, greifen verschiedene Benutzer mit unterschiedlichen Anwendungen auf die Daten zu. Dies ist nur möglich, wenn die Daten von den Anwendungen getrennt sind. Wenn nämlich die Daten mit einer Anwendung verbunden wären, gäbe es einen grossen Aufwand, diese Daten so aufzubereiten, dass auch andere Anwendungen sie lesen und verwenden könnten. Diese Unabhängigkeit wird besonders wichtig, wenn die DBMS-Software erneuert werden soll, ohne Dutzende von Anwenderprogrammen anpassen zu müssen. Beispiel: Ein Word-File ist sehr schwierig im Excel-Programm zu öffnen, und dies, obwohl beide Programme vom selben Hersteller kommen.

Datenpersistenz (Seite 18)	Die Erfassung von *Geodaten* und anderen Daten ist meistens mit einem grossen Aufwand und hohen Kosten verbunden. Deshalb ist die Datenpersistenz ein speziell geforderte Eigenschaft eines Geodata Warehouse. Damit wird sichergestellt, dass Daten nicht einfach verloren gehen und danach aufwendig und teuer wiederbeschafft werden müssen.
Datenintegrität (Seite 16)	Da die Daten eines Amtes häufig auch Auskunft über rechtliche Verhältnisse geben müssen, wie zum Beispiel die Grundbuchplandaten der amtlichen Vermessung, wird von ihnen gefordert, dass sie absolut korrekt und zuverlässig, also integer, sind. Dies wird durch die Definition und Einhaltung von ganz bestimmten Konsistenzbedingungen und -regeln erreicht.

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 15: Schematische Darstellung eines Geodata-Warehouse und der möglichen Zugriffe aus verschiedenen Ämtern

1.3.5 Übung

INSTALLATIONSHINWEISE (TUTOREN): **Diskussionstopic 'Datenbank-Anwendungen' aufsetzen, Feedback zu Postings geben**

Suchen Sie selber eine weitere Datenbank-Anwendung (in Ihrem Umfeld, im Internet, ...) und versuchen Sie abzuschätzen, welches speziell wichtige / speziell geforderte Eigenschaften für diese Anwendung sind.

Verfassen Sie eine kurze Zusammenstellung der Informationen (im gleichen Stil wie die Anwendungen in dieser Unit) und veröffentlichen Sie diese im Diskussionsforum unter dem Thema „Datenbank-Anwendungen“. Schauen Sie sich unbedingt auch weitere Zusammenstellungen ihrer Mitstudierenden an.

Ihre Zusammenstellung wird von einem Tutor angeschaut, und er wird ein kurzes Feedback dazu geben (ebenfalls im Forum).

1.4 Vorteile und Nachteile

Vom Karteikasten zur Datenbank

Die Art der Datenspeicherung und Verwaltung hat im Verlauf der Jahre eine grosse Entwicklung durchgemacht - vom manuellen Karteikartensystem über die erste (dateibasierte) Computerisierung desselben bis hin zu modernen Datenbanksystemen. Der erste Teil dieser Unit befasst sich mit dateibasierten Systemen im Vergleich zu Datenbanksystemen. Danach werden die Vor- und Nachteile von Datenbanksystemen aufgeführt.

1.4.1 DBS versus dateibasiert

Mit den Eigenschaften eines Datenbanksystems (Unit Eigenschaften des Datenbankansatzes (Seite 12) und Unit Anwendungsgebiete (Seite 20)) im Hinterkopf wenden wir uns den dateibasierten Systemen zu. Beim dateibasierten Ansatz definiert und kreiert jeder Anwender mit einem spezifischen Programm jeweils die Dateien, die er für eine spezielle Anwendung benötigt. Im Vergleich zum Datenbanksystemansatz ergeben sich daraus einige Einschränkungen.

Fahren Sie mit der Maus über die fetten Schlagwörter auf der linken Seite und sehen Sie die Einschränkungen gegenüber dem Datenbankansatz erklärt in Wort (rechts) und Bild (unten).

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 16: Datenbanksystem versus dateibasierte Systeme

Bei einem dateibasierten Ansatz kann es geschehen, dass die Administration einer Schule Daten pflegt (z. B. basierend auf einem Tabellenkalkulationssystem wie zum Beispiel Microsoft Excel), die Auskunft über die Registrierung und Gebührenbezahlung der Studenten geben. Währendessen verwalten die Dozenten Daten über die Studierenden und ihre Noten. Obwohl beide Anwender an Studierendendaten interessiert sind, haben sie doch unterschiedliche Dateien und unterschiedliche Programme, um diese zu verwalten und zu ändern. Diese *Redundanz* im Definieren und Speichern von Daten verschwendet Speicherplatz. Ausserdem ist ein mehrfacher Aufwand nötig, um die Daten zu aktualisieren. Änderungen an den Studierendendaten müssen von der Administration und von den Dozenten unabhängig voneinander vorgenommen werden.

1.4.2 Vorteile DBMS

Grundsätzlich können hier alle in der Unit Eigenschaften des Datenbankansatzes (Seite 12) erwähnten Punkte noch einmal als Vorteile aufgeführt werden.

- Mehrfachnutzung (Seite 13)
- Strukturierte und beschriebene Daten (Seite 14)
- Trennung von Daten und Anwendungen (Seite 15)
- Datenintegrität (Seite 16)
- Transaktionen (Seite 17)
- Datenpersistenz (Seite 18)
- Datensichten (Seite 19)

Dazu kommen noch weitere, bis jetzt nicht explizit erwähnte Vorteile.

Verwenden Sie für die Navigation die blauen Buttons.

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 17: vorteile.swf

Klicken Sie auf den nachstehenden Link für eine Liste der weiteren Vorteile. (PDF-File der weiteren Vorteile (www.gitta.info/IntroToDBS/de/multimedia/weitereVorteileDBMS.pdf))

1.4.3 Nachteile DBMS

Neben den zahlreichen Vorteilen eines Datenbanksystems dürfen aber auch die Nachteile nicht unerwähnt bleiben.

Verwenden Sie für die Navigation die blauen Buttons.

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 18: nachteile.swf

Klicken Sie auf den nachstehenden Link für eine Liste der Nachteile. (PDF-File der Nachteile (www.gitta.info/IntroToDBS/de/multimedia/nachteileDBMS.pdf))

1.5 Aufgaben

Die nachstehenden Aufgaben sollen Ihnen ermöglichen, zu testen, ob Sie die Inhalte dieser Lektion verstanden haben.

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 19: tabellenkalkulationVsDbms.swf

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 20: zentralesGeodataWarehouse.swf

Ziehen Sie in der nachstehenden Übung die Nummern neben den Begriffen auf die Kreise in der Grafik. Arbeiten Sie möglichst exakt, und wenn Sie alle Nummern platziert haben, können Sie mit „Check“testen, ob alles richig ist. Der Reset-Button setzt alle Nummern in die Ursprungsposition zurück.

Achtung: Der Check-Button funktioniert nur korrekt, wenn alle Kreise mit einer Nummer bedeckt sind.

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 21: dragDrop_test_informationssystemaufbau.swf

1.6 Zusammenfassung

In dieser Lektion wurden die Bedeutung einer datenbankgestützten Datenverwaltung erläutert. Dazu wurden eingangs Begriffe wie Datenbanksystem und Geoinformationssystem definiert bzw. repetiert und im Kontext der Begriffe Daten und Informationen positioniert.

Anschliessend wurden wichtige Gründe für den Einsatz von Datenbanksystemen bei der Verwaltung von Daten im Allgemeinen und von Geodaten im Speziellen aufgeführt und anhand von Beispielen aus verschiedenen Anwendungsbereichen illustriert. Zu den massgebenden Eigenschaften des Datenbankansatzes gehören die Mehrfachnutzung sowie die Strukturierung und Beschreibung der Daten. Weiter wurden die Vorteile der Trennung von Daten und Anwendungen und die verschiedenen Merkmale zur Gewährleistung einer hohen Zuverlässigkeit und Datensicherheit behandelt, namentlich die Konzepte der Transaktionen und der Datensichten.

1.7 Literaturempfehlungen

- ZEHNDER, C.A., 1998 *Informationssysteme und Datenbanken* [**?**, Einführung ins Thema Informationssysteme und Datenbanken auf Deutsch]

Glossar

Daten (Computerdaten): Die Darstellung von Tatsachen (zum Beispiel Messwerte) oder Konzepten, die in einer durch den Computer lesbaren Form erzeugt oder in eine entsprechende Form gebracht werden.

Datenbanksystem: Ein Datenbanksystem (DBS) besteht aus einem DBMS und einer oder mehreren Datenbanken.

Datenbankverwaltungssystem: Datenbankverwaltungssysteme (engl. database management system oder DBMS) sind Softwareprodukte für die dauerhafte, integre und anwendungsunabhängige Speicherung und Verwaltung sowie die flexible und bequeme Verwendung von grossen, mehrfach benutzbaren integrierten Datenbasen.

Datenbasis und Datenbank: Eine Datenbasis ist eine Menge von Daten, die aus Sicht der Systembetreiber in irgendeiner Weise als zusammengehörig betrachtet werden. Angereichert um weitere Daten, die das DBMS zur Erfüllung seiner Aufgaben benötigt, bilden sie eine Datenbank (DB).

Dimensions of Information: Die Informationstheorie unterscheidet drei Dimensionen einer Information, die syntaktische, die semantische und die pragmatische. Nehmen wir als Beispiel eine Verkehrsampel. In der syntaktischen Dimension unterscheiden wir die drei Farben Rot, Gelb und Grün. Ihren Sinn bekommt die Ampel aber erst in der semantischen Dimension. Hier wird den Farben ihre Bedeutung zugeordnet. Rot hat die Bedeutung Halt, grün bedeutet freie Fahrt. Erst in der pragmatischen Dimension aber wird das Ampelsignal für die Verkehrsregelung brauchbar. Pragmatisch gesehen bedeutet das rote Licht, als Autofahrer muss ich anhalten. [**?**]

Geodaten: Unter Geodaten oder raumbezogenen Daten versteht man Datenobjekte, die durch eine Position im Raum direkt oder indirekt referenzierbar sind. Der Raum ist dabei definiert durch ein Koordinatensystem, das den Bezug zur Erdoberfläche herstellt. Geodaten werden in der Regel grafisch in Papierform oder an grafikfähigen Bildschirmen präsentiert. Aus informationstechnischer Sicht kann man die Daten, die wiederum zu Geodaten gehören, einteilen in: •Geometrie (Position und geometrische Ausprägung) •Topologie (explizit gespeicherte geometrisch-topologische Beziehungen) •Präsentation (graphische Ausprägungen wie Signaturen, Farbe, Typographie) •Sachdaten (alphanumerische Daten zur Beschreibung der Semantik). Geodaten stellen in der Informationsverarbeitung eine besondere Herausforderung dar. Die Gründe dafür sind: •der hohe Aufwand für die Erfassung

•die große Menge der anfallenden Daten •die geforderten Antwortzeiten beim Zugriff auf Geodaten •die Verarbeitung nach räumlichen Kriterien sowie •die Komplexität der Beziehungen.

Geoinformationssystem: "*Ein Geoinformationssystem dient der Erfassung, Speicherung, Analyse und Darstellung aller Daten, die einen Teil der Erdoberfläche und die darauf befindlichen technischen und administrativen Einrichtungen sowie geowissenschaftliche, ökonomische und ökologische Gegebenheiten beschreiben.*" [**?**]

Information: Information ist nutzbare Antwort auf eine konkrete Fragestellung [**?**] . Information wird meistens aus Daten extrahiert oder abgeleitet.

Informationssystem: Ein Informationssystem erweitert die Datenbank um eine Reihe von Werkzeugen (engl. software tools) zur Abfrage, Darstellung, Transformation und Analyse von Daten.

Konzeptionell: Nach Duden: die einem Werk zugrunde liegende Anschauung, Leitidee; für das Datenmodell gilt: unabhängig von einer bestimmten Technologie

Metadaten: Meta ist ein Präfix, das in den meisten informationstechnologischen Anwendungen „eine zugrundeliegende Definition oder Beschreibung“meint. Metadaten sind also eine Definition oder Beschreibung von Daten. Man spricht auch von „Daten über Daten“.

Redundanz: Bezeichnet die mehrfache Speicherung gleicher Daten an verschiedenen Stellen (einer Datenbank).

Struktur: Die Struktur ist der abstrakte innere Aufbau.

Strukturierter Datenbestand: Ein Datenbestand wird als strukturiert bezeichnet, wenn er systematische Untergliederungen und Verknüpfungen zulässt.

Syntax: Syntax ist id Lehre vom Satz, Lehre von den Regeln, wonach in Sprache(n) aus den Wörtern zusammengehörige Wortgruppen gebildet werden, und die Lehre von der Kombination von Wörtern zu Sätzen. „Syntaxis“heisst auf Griechisch Zusammenstellung, Anordnung, Aneinanderreihung. [**?**]

WAP: Das Wireless Application Protocol - kurz: WAP - ist der wichtigste Standard, um Internetkommunikation und interaktive Dienste für Handys und andere mobile Endgeräte zu realisieren: Nachrichten aus dem Internet abrufen, Börsenkurse abfragen, Flugtickets buchen.

2 Datenbanksysteme: Konzepte und Architekturen

Nachdem Sie die Vorteile einer datenbankgestützten Datenverwaltung und die vielseitigen Einsatzmöglichkeiten von Datenbanksystemen kennen gelernt haben, wollen wir uns in dieser Lektion mit einigen grundlegenden Konzepten und typischen Architekturen von Datenbanksystemen vertraut machen.

Dabei interessiert zunächst, wie ein konzeptionelles Schema in eine Datenbankumgebung übertragen werden kann und welche typischen Mittel dafür zur Verfügung stehen. Anschliessend lernen wir eine generische Datenbankarchitektur kennen, die es uns erlauben soll, wichtige Aspekte wie das Zusammenspiel der verschiedenen Schemas sowie Schnittstellen für die Kommunikation mit einem Datenbankverwaltungssystem verstehen zu können.

Lernziele

- Sie kennen den Zusammenhang zwischen Datenschemata, Datenbankmodellen und Datenbankinstanzen und können diese beschreiben.
- Sie können die Drei-Schema-Architektur aufzeichnen und erläutern.
- Sie kennen Bedeutung und Prinzip einer Datenbankschnittstelle und können deren typische Funktionalität aufzählen.

2.1 Datenbankmodelle, Schemas und Instanzen

Mit Datenbanken sollen Sachverhalte und Prozesse aus der Realwelt in computertechnischer Form beschrieben und gespeichert werden. Die dazu erforderliche Abbildung erfolgt wiederum mit Hilfe von Modellen, in diesem Fall den sogenannten Datenbankmodellen.

In dieser Unit werden die gebräuchlichsten vorgestellt. Im Anschluss daran werden Datenbankschemas als formelle Beschreibung einer konkreten Datenbank und Datenbankinstanzen als deren Inhalt bzw. Zustand eingeführt.

2.1.1 Datenbankmodelle

Datenbanksysteme können auf verschiedenen Datenmodellen bzw. Datenbankmodellen basieren. Ein Datenmodell ist eine Ansammlung von Konzepten und Regeln zur Beschreibung der Struktur einer Datenbank. Dabei verstehen wir unter der Struktur einer Datenbank die Datentypen, Bedingungen und Beziehungen zur Beschreibung bzw. Speicherung der Daten.

Die wichtigsten Datenbankmodelle sind:

Netzwerkmodell und Hierarchisches Modell

Sie sind Vorgänger des relationalen Modells. Sie bauen auf individuellen Datensätzen auf und können hierarchische Beziehungen oder auch allgemeinere netzartige Strukturen der Realwelt ausdrücken.

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 22: Netzwerk und Hierarchisches Datenmodell

Relationales Modell

Es ist das bekannteste und in heutigen DBMS am weitesten verbreitete Datenbankmodell. Es stellt die Datenbank als eine Sammlung von Tabellen (Relationen) dar, in denen alle Daten angeordnet werden.

Dieses Modul befasst sich vorwiegend mit dem relationalen Datenbankmodell und den darauf basierenden Datenbanksystemen.

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 23: Relationales Datenbankmodell

Objektorientiertes Modell

Objektorientierte Modelle definieren eine Datenbank als Sammlung von Objekten mit deren Eigenschaften und Methoden. Eine detailliertere Besprechung von objektorientierten Datenbanken folgt in späteren Modulen.

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 24: Schematische Darstellung eines objektorientierten Datenbankmodells

Objektrelationales Modell

Objektorientierte Modelle sind zwar sehr mächtig, aber auch recht komplex. Mit dem relativ neuen objektrelationalen Datenbankmodell wurde das einfache und weit verbreitete relationale Datenbankmodell um einige grundlegende objektorientierte Konzepte erweitert.

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 25: Schematische Darstellung des objektrelationalen Datenbankmodells

2.1.2 Datenbankschemas und Datenbankinstanzen

Unabhängig vom Datenbankmodell ist es wichtig, zwischen der Beschreibung der Datenbank und der Datenbank selber zu unterscheiden. Die Beschreibung einer Datenbank wird **Datenbankschema** oder auch *Metadaten*[9] genannt. Das Datenbankschema wird während des Datenbank-Design-Prozesses festgelegt und ändert später nur sehr selten.

Die eigentlichen Daten einer Datenbank verändern sich im Laufe der Zeit häufig. Der Datenbankzustand zu einem bestimmten Zeitpunkt, gegeben durch die aktuell existierenden Inhalte und Beziehungen und deren Attribute, wird **Datenbankinstanz** genannt.

Die nachfolgende Illustration zeigt auf, dass das Datenbankschema als Schablone oder Bauplan für eine oder mehrere Datenbankinstanzen betrachtet werden kann.

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 26: Analogie zwischen Datenbankschema und Bauplan

Beim Entwurf einer Datenbank wird zwischen zwei Abstraktionsstufen und ihren entsprechenden Datenschemas (konzeptionelles Datenschema und logisches Datenschema, siehe untenstehende Definitionen) unterschieden.

Konzeptionelles Datenschema: Ein konzeptionelles Datenschema („conceptual schema“) ist eine systemunabhängige Datenbeschreibung, d.h. sie ist unabhängig von den eingesetzten Datenbank- und Computersystemen. [**?**]

Logisches Datenschema: Ein logisches Datenschema („logical schema“) beschreibt die Daten in der Datenbeschreibungssprache (DDL = Data Definition Language) eines bestimmten Datenbank-Verwaltungssystems. [**?**]

Das konzeptionelle Datenschema orientiert sich ausschliesslich an der Datenbankanwendung und somit an der realen Welt, nicht aber an der datentechnischen Infrastruktur (DBMS und Computersysteme), die dafür allen-

[9] Meta ist ein Präfix, das in den meisten informationstechnologischen Anwendungen „eine zugrunde liegende Definition oder Beschreibung“meint. Metadaten sind also eine Definition oder Beschreibung von Daten. Man spricht auch von „Daten über Daten“.

falls zum Einsatz kommen. *Entitätenblockdiagramme*[10] und Relationen sind Werkzeuge für die Erstellung eines konzeptionellen Schemas.

Beim Datenbankentwurf wird aus dem konzeptionellen Datenschema das logische Schema abgeleitet (siehe Unit Relationales Datenbankdesign (www.gitta.info/LogicModelin/de/)). Am Ende dieser Ableitung steht das logische Datenschema für eine spezielle Anwendung und ein spezielles DBMS. Ein DB-Entwicklungssystem setzt danach das logische Schema direkt in Anweisungen für das DBMS um.

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 27: Schematische Darstellung der verschiedenen Schemas

[10]Eine Entität ist eine Einheit. In einer relationalen Datenbank ist eine Entität als Tabelle dargestellt.

2.1.3 Gegenüberstellung von räumlichen Modellen und Datenbankmodellen

Im Modul „Räumliche Datenmodellierung“haben Sie Konzepte und numerische Modelle zur Repräsentation raumbezogener Phänomene der Realwelt kennen gelernt.

Die nachfolgende Gegenüberstellung soll Ihnen die Unterscheidung von räumlichen Modellen und Datenbankmodellen erleichtern.

Räumliche Modelle

Räumliche Modelle ermöglichen eine Abbildung (bzw. Modellierung) von raumbezogenen Phänomenen der Realität (Gegebenheiten, Prozesse) auf einer primär konzeptuellen Stufe, d. h. es wird in der Regel noch keine Aussage über die Konkretisierung bzw. die Umsetzung dieser Modelle in die Informatik gemacht. Räumliche Modelle sind oft grafik-orientierte Modelle (z. B. Karten oder Pläne) und können sowohl digital als auch analog (z. B. in einem Gipsmodell) umgesetzt werden.

Beispiele für räumliche Modelle sind:

- das Vektormodell als Spezialfall eines Objektmodells
- das Rastermodell als Spezialfall eines feldbasierten räumlichen Modells
- kombinierte (hybride) Modelle

Datenbankmodelle

Datenbankmodelle gehören in die Kategorie der Informatikmodelle und werden auch als konkrete Modelle bzw. als Implementierungsmodelle bezeichnet. Datenbankmodelle sind sehr allgemein einsetzbar und somit nicht auf raumbezogene Aufgabenstellungen beschränkt.

Beispiele für Datenbankmodelle sind:

- das relationale Datenbankmodell
- das objekt-relationale Datenbankmodell

2.2 DBMS-Architektur und Datenunabhängigkeit

Datenbankverwaltungssysteme sind komplexe Software-Lösungen, die oft über Jahre entwickelt und optimiert wurden. Aus Anwendungs- und Benutzersicht weisen jedoch die meisten Systeme eine ähnliche Basisarchitektur auf. Die Auseinandersetzung mit dieser Basisarchitektur hilft, Querbezüge zur Datenmodellierung zu schaffen und die eingangs des Moduls postulierte „Datenunabhängigkeit“des Datenbankansatzes zu verstehen.

2.2.1 Drei-Schema-Architektur

Nachdem wir im Abschnitt „Datenmodelle, Schemas und Instanzen (Seite 39)“das konzeptionelle und das daraus abgeleitete logische Schema kennen gelernt haben, werden in dieser Unit zum Verständnis der DBMS-Architektur noch zwei weitere Schemas vorgestellt - das externe Schema und das interne Schema.

Externes Schema: Ein externes Datenschema („external schema“) beschreibt die Benutzersichtdaten bestimmter Benutzer(gruppen) sowie die damit verbundenen spezifischen Operationen und Bedingungen. [**?**]

Internes Schema: Das interne Datenschema („internal schema“) beschreibt den Inhalt der Datenbasis und die benötigten Dienstleistungsfunktionen, soweit dies für den Einsatz des DBMS nötig ist. [**?**]

Das interne Schema beschreibt somit die Daten aus computernaher Sicht bzw. aus der Sicht des Systems. Es ergänzt das zugehörige logische Schema um datentechnische Aspekte wie Speichermethoden oder Hilfskonstrukte zur Steigerung der Effizienz.

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 28: Drei-Schema-Architektur

Die rechte Seite der obigen Abbildung wird auch als Drei-Schema-Architektur bezeichnet. Bstehend aus internem, logischem und externem Schema.

Während das interne Schema die physische Gruppierung der Daten und die Speicherplatzbelegung beschreibt, gibt das logische Schema (abgeleitet aus dem konzeptionellen Schema) den grundlegenden Aufbau der Datenstruktur wieder. Das externe Schema einer konkreten Anwendung beleuchtet im Allgemeinen nur einen Teilbereich des logischen Schemas, der für die Anwendung relevant ist. Daraus folgt, dass jede Datenbank genau ein internes und ein logisches Schema hat, während jeweils mehrere externe Schemas möglich sind.

Das Ziel der Drei-Schema-Architektur ist die Trennung der Benutzeranwendungen von der physischen Datenbank, den gespeicherten Daten. Physisch sind die Daten lediglich auf der internen Ebene vorhanden, die anderen Repräsentationsformen werden jeweils bei Bedarf daraus berechnet bzw. abgeleitet. Das DBMS hat die Aufgabe, die Abbildung zwischen den einzelnen Ebenen zu realisieren.

2.2.2 Datenunabhängigkeit

Mit Kenntnis der Drei-Schema-Architektur lässt sich der Begriff der Datenunabhängigkeit so erläutern, dass eine höhere Ebene dieser Datenarchitektur immun gegenüber Änderungen auf der nächst tieferen Ebene ist.

Physikalische Unabhängigkeit: Das logische Schema kann demnach unverändert bleiben, wenn sich beispielsweise aus Gründen der Optimierung oder Reorganisation der Speicherort oder die Speicherform einzelner Daten ändern.

Logische Unabhängigkeit: Ebenso können bei (den meisten) Änderungen des logischen Schemas die externen Schemas unverändert weiterbestehen. Dies ist besonders deshalb erwünscht, weil dadurch Anwendungsprogramme nicht modifiziert oder neu übersetzt werden müssen.

Beispiel zur physikalischen und logischen Unabhängigkeit

Müssen beispielsweise in einem Geomatik-Ingenieurbüro mit unterschiedlichen Abteilungen (Amtliche Vermessung, Leitungskataster, usw.) aus irgendwelchen Gründen die Daten der verwendeten GIS-Datenbank auf einen neuen Datenbank-Server portiert werden, so darf dies keine Auswirkungen auf das logische Schema der Datenbank haben.

Wird wiederum das logische Schema verändert, müssen die unterschiedlichen Anwendergruppen des Geomatik-Ingenieurbüros ohne Einschränkungen mit den Daten der Datenbank weiterarbeiten können, obgleich sie verschiedene Anwender-Sichten auf die Daten haben.

2.3 Datenbanksprachen und Datenbankschnittstellen

Bis jetzt haben wir uns mit Datenmodellen bzw. Datenbankmodellen, Datenbeschreibungen und den Komponenten eines Datenbanksystems vertraut gemacht.

In dieser Unit wollen wir uns anschauen, wie ein Datenmodell in das Datenbanksystem und wie Informationen zu den Anwendern gelangen. Fachlich etwas präziser formuliert, sollen in dieser Unit die folgenden Fragen beantwortet werden:

- Wie erfolgt die Interaktion zwischen einer Anwendung und einem Datenbank-Verwaltungssystem?
- Wie sieht ein Datenbanksystem gegenüber einer Benutzerin aus?
- Wie können Anfragen durchgeführt und Resultate in einer Anwendung dargestellt werden?

2.3.1 Datenbanksprachen

DDL (Data Definition Language)

Wer Daten und Datenstrukturen beschreiben will, benötigt dazu ein geeignetes Beschreibungswerkzeug, eine **Datenbeschreibungssprache** . Diese dient der Definition und der Veränderung des Datenschemas.

Typische DDL-Operationen (mit den entsprechenden Schlüsselwörtern in der relationalen Datenbanksprache SQL (www.gitta.info/RelQueryLang/de/)) sind:

- Erzeugen von Tabellen und Festlegen von Attributen („create table ...“)
- Ändern von Tabellen durch Hinzufügen oder Entfernen von Attributen („alter table ...“)
- Löschen ganzer Tabellen mitsamt Inhalt (!) („drop table ...“)

DML (Data Manipulation Language)

Zusätzlich wird eine Sprache für die Beschreibung der Arbeitsmöglichkeiten mit Daten (Speichern, Suchen, Lesen, Ändern), den sogenannten Datenmanipulationen, benötigt. Solche Operationen können mit einer **Datenmanipulationssprache** durchgeführt werden. In solchen Sprachen treten typischerweise die erwähnten Stichworte auf, meist auf Englisch („insert, modifiy, update, delete, select“).

Typische DML-Operationen (mit den entsprechenden Schlüsselwörtern in der relationalen Datenbanksprache SQL (www.gitta.info/RelQueryLang/de/)) sind:

- Einfügen von Daten („insert“)
- Verändern von Einträgen („update“)
- Löschen von Einträgen („delete“)
- Datenabfragen („select“)

Häufig aber sind diese zwei Sprachen für Definition und Manipulation von Datenbanken in einer umfassenden Sprache zusammengefasst. Ein gutes Beispiel ist die relationale Anfragesprache SQL (Structured Query Language), die vertiefter in der Lektion „Relationale Anfragesprache SQL (www.gitta.info/RelQueryLang/de/)“behandelt wird.

2.3.2 Datenbankschnittstellen

Funktionsprinzip einer Datenbankschnittstelle

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 29: Funktionsprinzip einer Datenbankschnittstelle

Die Anwendung stellt mit Hilfe von SQL, einer Anfragesprache, eine Anfrage an das Datenbanksystem. Dort wird die entsprechende Antwort (Result-Set) bereitgestellt und ebenfalls mittels SQL an die Anwendung zurückgegeben. Diese Kommunikation kann interaktiv oder eingebettet stattfinden.

Art und Nutzung der Datenbankschnittstelle

Die zwei wichtigsten Einsatzmöglichkeiten einer Datenbankschnittstelle, wie SQL eine ist, sind nachfolgend kurz aufgeführt:

•interaktiv:	SQL kann als interaktive Anfragesprache, direkt von einem Terminal aus eingesetzt werden.
•eingebettet	SQL kann in Programmiersprachen („host language“) eingebettet („embedded“) werden, um damit Datenbankanwendungen zu erstellen.

2.3.3 Benutzungsoberflächen

Eine Benutzungsoberfläche ist die „Seite“der Datenbankschnittstelle die der Benutzer sieht. Diese ist häufig grafisch oder zumindest teilgrafisch aufgebaut und bietet Hilfsmittel welche die Benutzung der Datenbank vereinfachen.

Formular-basierte Oberfläche

Diese Oberflächen bestehen Formularen die den Benutzern angepasst sind. Der Benutzer kann entweder alle Felder ausfüllen und so neue Einträge machen, oder er kann einzelne Felder ausfüllen und auf diese Weise eine Anfrage für die restlichen Felder machen.

Formular-basierte Benutzungsoberflächen sind stark verbreitet und machen die wichtigste Interaktionsmöglichkeit mit DBMS aus. Formular-basierte Oberflächen sind sehr einfach zu bedienen und haben den grossen Vorteil, dass die Anwender keine Kenntnisse einer Datenbanksprache (z. B. SQL) benötigen.

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 30: Beispiel einer formular-basierten Benutzungsoberfläche

Text-basierte Oberflächen

Um die Datenbank administrieren zu können oder für andere professionelle Nutzer gibt es Möglichkeiten über Ein- und Ausgabefenster direkt in einer Anfragesprache (in Code-Form) mit der Datenbank zu kommunizieren.

Wir werden diese Möglichkeit in der Lektion Relationale Anfragesprache SQL (`www.gitta.info/RelQueryLang/de/`) näher kennen lernen.

Text-basierte Oberflächen sind sehr mächtig und ermöglichen eine sehr umfassende Interaktion mit einem DBMS. Ihre Bedienung erfordert aber aktive Kenntnisse der entsprechenden Datenbanksprache (z. B. SQL).

GIS-Oberfläche

Eine GIS-Benutzungsoberfläche integriert in aller Regel auch Elemente einer Datenbankschnittstelle. Die Datenbankinteraktion erfolgt dabei über eine Kombination von verschiedenen Oberflächen:

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 31: Beispiel einer text-basierten Benutzungsoberfläche

- grafische Interaktion via Selektionen innerhalb der Kartendarstellung
- Kombination von formular-basierter und text-basierter Interaktion (z.B. in der Form von Query-Wizards, speziellen Masken, die die Erstellung von Datenbankabfragen unterstützen und erleichtern)

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 32: Beispiel einer GIS-Oberfläche

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 33: Beispiel eines Query-Wizards in einem GIS

2.4 Aufgaben

Begriffszuordnung

Verschieben Sie die untenstehenden Begriffe so in die leeren Kästchen, dass jeweils zwei zusammenpassende Begriffe ein Paar bilden. Klicken Sie den Check-Button an, um ihre Anordnung zu überprüfen. Mit dem Reset-Button können Sie alle Begriffe wieder an die Ausgangsposition zurücksetzen.

Hinweis: Die Begriffe werden hier so verwendet, wie sie in dieser Lektion eingeführt wurden. In anderer Fachliteratur werden diese Begriffe möglicherweise leicht anders verwendet.

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 34: Interaktion: Begriffzuordnung

Drei-Schema-Architektur

Ziehen Sie die Nummern bei den Begriffen (blau) auf die richtigen weissen Kreise in der Grafik. Mit dem „Check“-Button können Sie ihre Anordnung überprüfen. Mit dem „Reset“-Button können Sie die Nummern an ihre Ausgangsposition zurücksetzen.

Achtung: Für ein korrektes Funktionieren des „Check“-Buttons sollten Sie diesen erst verwenden, wenn Sie alle Nummern den Kreisen in der Grafik zugeordnet haben.

Tipp: Beginnen Sie mit einfachen/bekannten Begriffen.

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 35: Interaktion: Drei-Schema-Architektur

2.5 Übung zur Datenunabhängigkeit

INSTALLATIONSHINWEISE (TUTOREN): **Diskussionstopic 'Übung zur Datenunabhängigkeit' aufsetzen, Feedback zu den Postings geben**

Versuchen Sie in eigenen Worten die Architektur eines Datenbankmanagement-Systems zu beschreiben. Was ist Datenunabhängigkeit und wie wird sie erreicht? Ergänzen Sie Ihre Diskussion wenn möglich mit ein oder zwei Beispielen aus Ihrem bisherigen Informatikalltag, die den Nutzen der Datenunabhängigkeit aufzeigen.

Ihre Besprechung, die nicht mehr als 200 bis 400 Wörter umfassen soll, veröffentlichen Sie bitte auf dem Diskussionsforum unter dem Thema „Übung zur Datenunabhängigkeit". Lesen Sie auch die Mitteilungen ihrer Mitstudierenden und kommentieren Sie diese gegebenenfalls.

2.6 Zusammenfassung

In dieser Lektion wurden die wichtigsten Datenbankmodelle (relational, objektorientiert und objekt-relational) und die darauf basierenden Datenbanktechnologien vorgestellt. Ein solches Datenbankmodell stellt die Grundregeln zur Verfügung, um ein konzeptionelles Schema in eine bestimmte Datenbankumgebung zu übertragen und es mittels eines logischen Schemas zu beschreiben. Diese Datenbeschreibung bildet die Grundlage, um anschliessend entsprechende Datenbanken (Instanzen) mit den eigentlichen Daten zu erzeugen und zu verwalten. Die logische Modellierung auf der Basis des relationalen Datenmodells lernen wir in der Lektion Logische Modellierung (www.gitta.info/LogicModelin/de/) kennen.

In der Folge haben wir eine generische Datenbankarchitektur kennen gelernt, die auf einem 3-Schema-Konzept basiert. Diese Drei-Schema-Architektur erweitert das logische Schema um ein internes und ein externes Schema. Die Dreiteilung wiederum bildet eine wichtige Voraussetzung für die bereits früher postulierte Datenunabhängigkeit und für die Realisierung von Schnittstellen für die Kommunikation zwischen Anwendung und Datenbankverwaltungssystem. Ein Grossteil der Systeme bietet eine Schnittstelle auf der Basis der relationalen Datenbanksprache SQL (Structured Query Language). Die Grundelemente und den Einsatz von SQL lernen Sie in der Lektion „Die relationale Anfragesprache SQL“(www.gitta.info/RelQueryLang/de/) kennen.

2.7 Literaturempfehlungen

- ZEHNDER, C. A., 1998 *Informationssysteme und Datenbanken* [**?**, Einführung ins Thema Informationssysteme und Datenbanken inklusive Datenmodellierung, auf Deutsch]

Glossar

Entität: Eine Entität ist eine Einheit. In einer relationalen Datenbank ist eine Entität als Tabelle dargestellt.

Externes Schema: Ein externes Datenschema („external schema“) beschreibt die Benutzersichtdaten bestimmter Benutzer(gruppen) sowie die damit verbundenen spezifischen Operationen und Bedingungen. [**?**]

Internes Schema: Das interne Datenschema („internal schema“) beschreibt den Inhalt der Datenbasis und die benötigten Dienstleistungsfunktionen, soweit dies für den Einsatz des DBMS nötig ist. [**?**]

Das interne Schema beschreibt somit die Daten aus computernaher Sicht bzw. aus der Sicht des Systems. Es ergänzt das zugehörige logische Schema um datentechnische Aspekte wie Speichermethoden oder Hilfskonstrukte zur Steigerung der Effizienz.

Konzeptionelles Datenschema: Ein konzeptionelles Datenschema („conceptual schema“) ist eine systemunabhängige Datenbeschreibung, d.h. sie ist unabhängig von den eingesetzten Datenbank- und Computersystemen. [**?**]

Logisches Datenschema: Ein logisches Datenschema („logical schema“) beschreibt die Daten in der Datenbeschreibungssprache (DDL = Data Definition Language) eines bestimmten Datenbank-Verwaltungssystems. [**?**]

Logische Unabhängigkeit: Ebenso können bei (den meisten) Änderungen des logischen Schemas die externen Schemas unverändert weiterbestehen. Dies ist besonders deshalb erwünscht, weil dadurch Anwendungsprogramme nicht modifiziert oder neu übersetzt werden müssen.

Metadaten: Meta ist ein Präfix, das in den meisten informationstechnologischen Anwendungen „eine zugrunde liegende Definition oder Beschreibung“meint. Metadaten sind also eine Definition oder Beschreibung von Daten. Man spricht auch von „Daten über Daten“.

Physikalische Unabhängigkeit: Das logische Schema kann demnach unverändert bleiben, wenn sich beispielsweise aus Gründen der Optimierung oder Reorganisation der Speicherort oder die Speicherform einzelner Daten ändern.

3 Das relationale Datenmodell

Das Relationenmodell dient den meisten derzeitigen Datenbanken als Grundlage. In komerziellen Datenbanken wird das Relationenmodell seit etwa 1981 eingesetzt. Es wurde von E. F. Codd um 1970 vorgestellt mit dem Ziel, die Datenunabhängigkeit zu gewährleisten, und es basiert auf einer Variante des mathematischen Konzepts der Relation, in der Relationen auf einfache Weise als Tabellen interpretiert werden.

Der Fokus dieser Lektion liegt in der Umwandlung eines konzeptionellen Schemas wie zum Beispiel des Entity-Relationship-Schemas (Gegenstands-Beziehungs-Schema) in das logische Schema, das relationalen Datenbankmodell. Erklärungen zu den verschiedenen Schema-Typen finden Sie in der Unit Datenmodelle, Schemata und Instanzen (www.gitta.info/DBSysConcept/de/).

Lernziele

- Sie kennen und verstehen die grundsätzlichen Konzepte des Relationenmodells und können Begriffe wie Domäne, Tupel oder Relation erläutern.
- Sie sind in der Lage, ein ER-Schema auf ein relationales Datenbankschema abzubilden und die damit verbundenen Regeln in der Praxis anzuwenden und einzuhalten.
- Sie erkennen die Bedeutung der Datenintegrität und können die verschiedenen Integritätsbedingungen sowie die Begriffe des Primär- und Fremdschlüssels erläutern. Sie sind hinsichtlich integritätsgefährdenter Operationen (Hinzufügen, Löschen und Ändern von Tupeln) sensibilisiert und erkennen deren Auswirkungen.
- Sie sind in der Lage die Unterschiede der einzelnen Normalformen im Normalisierungsprozess des relationalen Datenmodells zu erklären und können eine Normalisierung an praktischen Beispielen anwenden. Sie sind zudem mit den Begriffen der verschiedenen Abhängigkeiten vertraut und können diese im Normalisierungsprozess richtig unterscheiden.

3.1 Konzepte des Relationenmodells

Im Gegensatz zum Entity-Relationship-Modell (oder Gegenstands-Beziehungs-Modell), das ein konzeptuelles Modell darstellt, handelt es sich beim Relationenmodell um ein logisches Datenmodell. Dieses Modell liegt in gewisser Hinsicht eine Stufe „tiefer“. Es werden hier keine abstrakten Gegenstände oder Gegenstandstypen mehr betrachtet, sondern nur noch deren konkrete, datenmässige Umsetzung. Das Ziel der logischen Datenmodellierung ist das Anordnen der zu speichernden Informationen mit Hilfe eines Modells, das eine möglichst redundanzfreie Speicherung unterstützt und geeignete Operationen für die Datenmanipulation zur Verfügung stellt.

3.1.1 Organisation der Daten im relationalen Datenbankmodell

In der Regel basiert die logische Datenmodellierung auf einem fertiggestellten konzeptuellen Schema, das mit Hilfe bestimmter Richtlinien und Regeln in möglichst eindeutiger Weise in einem relationalen Schema abgebildet wird. Die grundlegende Organisationsform der Daten im relationalen Datenbankmodell ist die Relation. Eine Relation kann als Tabelle dargestellt werden, wobei zu beachten ist, dass sich die Definition einer Relation nicht mit der einer Tabelle deckt und umgekehrt.

Es sind im Wesentlichen zwei Gründe, die für das Relationenmodell als logisches Datenbankmodell sprechen:

- **Einfachheit:** Die gesamte Information einer relationalen Datenbank wird einheitlich durch Werte repräsentiert, die mittels eines einzigen Konstrukts (nämlich der „Relation“) strukturiert sind.
- **Systematik:** Das Modell besitzt eine fundierte mathematische Grundlage - die Mengentheorie.

Im Folgenden werden die grundlegenden Begriffe des relationalen Datenbankmodells eingeführt.

3.1.2 Definitionen

Domäne: Eine Domäne besteht aus einem Namen D und einer Menge atomarer Werte. Ein anderer Name für Domäne ist Wertebereich. Domänen definieren den Wertebereich von Attributen.

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 36: Beispiel Domäne

Tupel: Ein Tupel t ist eine Liste mit n Werten t = <d1, d2, ..., dn>, wobei jeder Wert di ein Element der Domäne Di, oder NULL sein muss.

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 37: Beispiel Tupel

Attribut: Ein Attribut A bezeichnet die Funktion, die eine Domäne D in einem Relationenschema R ausübt. Es kann auch als Abbildung der Tupel einer Relation auf den Wert des jeweiligen Tupels (für dieses Attribut) verstanden werden, wobei jeder Wert di ein Element der Domäne oder NULL sein muss.

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 38: Beispiel Attribut

Relationenschema: Ein Relationenschema R, Schreibweise: R(A1, A2, ..., An), bezeichnet eine Menge von Attributen A1, A2, ..., An.

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 39: Beispiel Relationenschema

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 40: Beispiel Relation

Relation: Eine Relation r ist eine Instanz (Ausprägung) des Relationenschemas R(A1, A2, ..., An). Sie ist eine Teilmenge des kartesischen Produkts (Kreuzprodukt) der beteiligten Domänen.

Relationales Datenbankschema: Ein relationales Datenbankschema ist eine Menge von Relationenschemata S = R1, ..., Rn zusammen mit einer Menge von Integritätsbedingungen. Eine relationale Datenbankinstanz ist die Menge r1, ..., rn , wobei ri Instanz von Ri ist und alle Integritätsbedingungen erfüllt sind. Eine relationale Datenbank ist ein relationales Datenbankschema mit einer entsprechenden Datenbankinstanz.

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 41: Begriffe des relationalen Datenschemas

Das relationale Schema eines Gegenstandes (Entität) kann als Tabelle (Relation) abgebildet werden. In diesem Beispiel sind die Entität die „Studierendennoten“. Diese Entität wird mit den Attributen Name, Fach und Note beschrieben. Die Domäne oder der Wertebereich der Attribute Name und Fach sind alle Gross- und Kleinbuchstaben des Alphabets, die des Attributs Note sind Zahlen von 1 bis 6 mit einer Kommastelle. Diese Struktur der Entität mit ihren Attributen, ohne den eigentlichen Inhalt, nennt man Relationenschema. Wenn nun Inhalte in die Relation eingefügt werden, dürfen nur Werte verwendet werden, die in der Domäne definiert sind. Ein Tupel

sind die zusammengehörenden Werte verschiedener Attribute. Es entspricht in der Tabelle einer Zeile.

INSTALLATIONSHINWEISE (TUTOREN): **Diskussionsforum 'Relationales Datenmodell' aufsetzen**

Zwischenfrage: In der obigen Tabelle hat sich ein kleiner Fehler eingeschlichen. Können Sie ihn entdecken?

Falls Sie die Antwort zu dieser Frage diskutieren möchten, steht Ihnen dazu das Diskussionsforumstopic „Relationales Datenmodell"zur Verfügung.

3.2 Abbilden eines ER-Schemas auf ein relationales Datenbankschema

In dieser Unit werden nun die Regeln eingeführt, mit deren Hilfe die Konstrukte des Entity-Relationship-Modells auf Konstrukten des relationalen Modells abgebildet werden können. Danach sind wir in der Lage, jedes beliebige Entity-Relationship-Modell in ein relationales Datenbankschema zu überführen.

Jede der folgenden Abbildungsregeln ist einem Konstrukt des Entity-Relationship-Modells gewidmet. Um ein Entity-Relationship-Modell in ein relationales Datenbankschema umzuwandeln, müssen alle 8 Abbildungsregeln abgearbeitet werden. Jede Regel wird für alle im zu bearbeitenden Entity-Relationship-Modell gefundenen und der Regel entsprechenden Gegenstandstypen (Regel 1,2,7 und 8) oder Beziehungen (Regel 3,4,5 und 6) angewendet. Man beachte die korrekte Reihenfolge der Anwendung der Abbildungsregeln in der Praxis (1,7,8,2,3,4,5 und zum Schluss 6).

Zu jeder Abbildungsregel wird die Definition angegeben und ein Anwendungsbeispiel gezeigt.

3.2.1 Repetition: ER-Konzepte

Erweitertes Entity-Relationship-Diagramm

Um die Abbildungsregeln zu verstehen, müssen Sie die Konzepte des ER-Modells kennen. Hier folgt ein Flash-Beispiel, anhand dem Sie Ihre Kenntnisse auffrischen können.

Falls Sie die Konzepte des ER-Modells nicht kennen, schauen Sie im folgenden Buch nach:

- Fundamentals of Database Systems, Elmasri, Ramez; Navathe, Shamakant B. (1994)

Sie sollten folgende Konstrukte erkennen: starker Gegenstandstyp, schwacher Gegenstandstyp, Beziehungstyp, identifizierender Beziehungstyp, Eigenschaft, abgeleitete Eigenschaft, mehrwertige Eigenschaft, zusammengesetzte Eigenschaft, Partialschlüsseleigenschaft und Schlüsseleigenschaft.

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 42: Interaktion: Erweitertes Entity-Relationship Diagramm

3.2.2 Regel 1

In diesem ersten Schritt werden alle starken Gegenstandstypen ins relationale Datenbankschema umgewandelt. Bei Subklassen benutzen sie Regel 8.

Definition Regel 1

Man erstelle ein Relationenschema R für jeden starken Gegenstandstyp G, wobei die Eigenschaften von G die Attribute von R bilden. Bei mehrwertigen Eigenschaften verfahre man nach Regel 7. Wähle einen Primärschlüssel (Identifikationsschlüssel) aus.

In diesem Beispiel sehen Sie die Anwendung der Regel 1:

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 43: Regel 1

Zuerst wird der starke Gegenstandstyp Kunde ausgewählt und eine Relation dafür erstellt.

```
Kunde
```

Dann werden alle Eigenschaften von Kunde, in diesem Fall „KundNr“, „Vorname“, etc. zu Attributen dieser Relation.

```
Kunde(KundNr, Vorname, Nachname,Strasse, StrNr)
```

Um ein einzelnes Tupel in dieser Relation ansprechen zu können, wird ein Primärschlüssel (mehr Informationen siehe Unit „Datenintegrität“) ausgewählt. In diesem Fall ist das Attribut KundNr als Primärschlüssel verwendbar.

```
Kunde( KundNr , Vorname, Nachname, Strasse, StrNr)
```

3.2.3 Regel 2

In diesem Schritt werden alle schwachen Gegenstandstypen ins relationale Datenbankschema umgewandelt.

Definition Regel 2

Man erzeuge für jeden schwachen Gegenstandstyps S mit Eigentümer G ein Relationenschema R, wobei die Eigenschaften von S die Attribute von R bilden. Bei mehrwertigen Eigenschaften verfahre nach Regel 7. Übernimm den Primärschlüssel des Relationenschemas, das dem Eigentümer G entspricht, und füge ihn als Fremdschlüssel R hinzu. Wähle eine Attributkombination (Partialschlüssel) aus, die dann zusammen mit diesem Fremdschlüssel den Primärschlüssel der Relation bildet. Erst die Kombination von Fremdschlüssel- und Partialschlüsselattributen bildet den Primärschlüssel von R.

In diesem Beispiel sehen Sie die Anwendung der Regel 2:

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 44: Regel 2

Zuerst wird der schwache Gegenstandstyp Teil ausgewählt und eine Relation dafür erstellt.

```
Teil
```

Dann werden alle Eigenschaften von Teil, in diesem Fall Name und Redakteur zu Attributen dieser Relation.

```
Teil(Name, Redakteur)
```

Dann wird der Primärschlüssel des Relationenschemas, das dem Eigentümer entspricht, als Fremdschlüssel hinzugefügt (Eigentümer ist hier die Relation „Zeitung(Name, Auflage, Preis)“).

```
Teil(ZeitungName, Name, Redakteur)
```

Das Attribut Name bildet zusammen mit dem Fremdschlüssel den Primärschlüssel der Relation.

```
Teil( ZeitungName, Name , Redakteur)
```

3.2.4 Regel 3

Hier werden nun alle binären Beziehungstypen der Art (1,1)(1,1), (0,1)(1,1) oder (0,1)(0,1) ins relationale Datenbankschema umgewandelt.

Definition Regel 3

Man suche alle regulären, binären „(1,1)(1,1)“-, „(0,1)(1,1)“- und „(0,1)(0,1)“-Beziehungstypen B. Man finde die Relationenschemata S und T für die beteiligten Gegenstandstypen und wähle eines davon aus (zum Beispiel S) um dort den Primärschlüssel von T als Fremdschlüssel sowie die Eigenschaften von B als Attribute hinzuzufügen.

In diesem Beispiel sehen Sie die Anwendung der Regel 3:

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 45: Regel 3

Zuerst wird ein Beziehungstyp der Art (1,1)(1,1), (0,1)(1,1) oder (0,1)(0,1)ausgewählt. Im Beispiel ist es der Beziehungstyp „leitet“.

Dann wählen wir einen an der Beziehung „leitet“beteiligten Gegenstandstyp aus.

`Zeitung( Name , Auflage, Preis)`

Den Primärschlüssel des zweiten Gegenstandstyps („ChefRedakteur“) fügen wir als Fremdschlüssel in die Relation Zeitung ein.

`Zeitung( Name , Auflage, Preis, ChefRedakteur_PersNr)`

Am Ende fügen wir die Eigenschaften der Beziehung „leitet“als Attribute der Relation Zeitung hinzu.

`Zeitung( Name , Auflage, Preis, ChefRedakteur_PersNr , SeitDatum)`

3.2.5 Regel 4

Nun werden alle binären Beziehungstypen der Art (1,n)(1,1), (0,n) (1,1), (1,n)(0,1) oder (0,n)(0,1) ins relationale Datenbankschema umgewandelt.

Definition Regel 4

Man suche alle regulären, binären „(1,n)(1,1)“-, „(0,n)(1,1)“-, „(1,n)(0,1)“- und „(0,n)(0,1)“-Beziehungstypen B, sowie die jeweiligen Relationenschemata S und T der beteiligten Gegenstandstypen. Man wähle das Relationenschema S auf der Seite mit der Beziehung (1,1) oder (0,1) aus und füge dort den Primärschlüssel von T als Fremdschlüssel hinzu. Ausserdem werden allfällige Eigenschaften von B als Attribute zu S hinzugefügt.

In diesem Beispiel sehen Sie die Anwendung der Regel 4:

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 46: Regel 4

Zuerst wird ein Beziehungstyp der Art (1,n)(1,1), (0,n)(1,1), (1,n)(0,1) und (0,n)(0,1) ausgewählt. Bei uns der Beziehungstyp „gibt in Auftrag“.

Dann wählen wir den auf der Seite (1,1) oder (0,1) liegenden Gegenstandstypen aus. Bei uns ist dies die Relation Inserat.

`Inserat( AuftragsNr , Grösse, Preis, ErscheinDat)`

Den Primärschlüssel des zweiten Gegenstandstyps („Kunde“) fügen wir als Fremdschlüssel in die Relation Inserat ein.

`Inserat( AuftragsNr , Grösse, Preis, ErscheinDat, `*`AuftraggeberKundNr`*`
)`

3.2.6 Regel 5

Hier werden alle binäre Beziehungstypen der Art (0,n)(0,n), (1,n) (0,n) oder (1,n)(1,n) ins relationale Datenbankschema umgewandelt.

Definition Regel 5

Man suche alle regulären, binären „(0,n)(0,n)“-, „(0,n)(1,n)“- und „(1,n)(1,n)“-Beziehungstypen B, sowie jeweils die Relationenschemata S und T der beteiligten Gegenstandstypen. Definiere für jeden Beziehungstyp B ein neues Relationenschema R. Die Primärschlüssel der Relationenschemata der beteiligten Gegenstandstypen S und T werden als Fremdschlüssel übernommen. Sie bilden zusammen den Primärschlüssel des neuen Relationenschemas R. Füge Attribute, die Eigenschaften von B entsprechen, zu R hinzu.

In diesem Beispiel sehen Sie die Anwendung der Regel 5:

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 47: Regel 5

Zuerst wird ein Beziehungstyp der Art (0,n)(0,n), (0,n)(1,n) und (1,n)(1,n) ausgewählt. Bei uns der Beziehungstyp „erscheint in“.

Dann definieren wir für den Beziehungstyp „erscheint in“ein neues Relationenschema.

`InseratInZeitung`

Die Primärschlüssel der Relationenschemata der beteiligten Gegenstandstypen Inserat und Zeitung werden als Fremdschlüssel übernommen.

`InseratInZeitung ( Inserat_AuftragsNr, Zeitung_Name )`

Zum Schluss fügen wir Attribute, die Eigenschaften der Beziehung „erscheint in“entsprechen, zur Relation „InseratInZeitung“hinzu.

`InseratInZeitung ( Inserat_AuftragsNr, Zeitung_Name , Position)`

3.2.7 Regel 6

Hier werden alle n-stelligen Beziehungstypen ins relationale Datenbankschema umgewandelt. Dies erfolgt analog zu Regel 5.

Definition Regel 6

Man verfahre für n-stellige Beziehungstypen (n>2) analog zu Regel 5, d. h. bilde sie auf eigenständige Relationenschemas ab und übernehme den Primärschlüssel der Relationenschemas aller beteiligten Gegenstandstypen als Fremdschlüssel.

Siehe Beispiel von Regel 5.

3.2.8 Regel 7

Wenn man in Regel 1 auf mehrwertige Eigenschaften gestossen ist, wandelt man diese gemäss dieser Regel ins relationale Datenbankschema um.

Definition Regel 7

Man erzeuge für jede mehrwertige Eigenschaft E ein neues Relationenschema R'. R' enthält ein Attribut A, das der Eigenschaft E entspricht und den Primärschlüssel K jenes Relationenschemas (R), welches dem Gegenstandstyp entspricht, der E enthält. Der Primärschlüssel von R' ergibt sich aus der Kombination von A und K (man beachte, dass R keine Attributentsprechung für die mengenwertige Eigenschaft besitzt).

In diesem Beispiel sehen Sie die Anwendung der Regel 7:

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 48: Regel 7

Zuerst definieren wir für die mehrwertige Eigenschaft „TelNr“eine neue Relation „ChefTelNr“mit dem Attribut „TelNr“)

`ChefTelNr(TelNr)`

Der Primärschlüssel der Relation, an der die mehrwertige Eigenschaft „TelNr“hängt (bei uns „Chefredakteur“), wird in die Relation „ChefTelNr“als Fremdschlüssel übertragen.

`ChefTelNr( Chefredakteur_PersNr ,TelNr)`

Zusammen mit dem Attribut „TelNr“bildet das Attribut „Chefredakteur_PersNr“den Primärschlüssel.

`ChefTelNr( Chefredakteur_PersNr,TerlNr )`

3.2.9 Regel 8

Wenn man in Regel 1 auf Subklassen gestossen ist, wandelt man diese gemäss dieser Regel ins relationale Datenbankschema um.

Definition Regel 8

Man erzeuge ein Relationenschema R für die Superklasse C mit den Attributen A(R) = (K, A1, ..., An). Bestimme K zum Primärschlüssel von R. Erzeuge weiter ein Relationenschema Ri für jede Subklasse Si, (1<=i<=m) mit den Attributen A(Ri) = (K) vereinigt (Attribute von Si). Setze den Primärschlüssel von Si gleich K.

In diesem Beispiel sehen Sie die Anwendung der Regel 8:

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 49: Regel 8a

Die Relation „Angestellter“(„ANummer“) ist die Superklasse, „Techniker“und „Ingenieur“(„Ausbildung“) sind die Subklassen in diesem Beispiel.

`Angestellter( ANummer )`

`Techniker`

`Ingenieur(Ausbildung)`

Den beiden Gegenstandstypen der Subklasse fügen wir den Primärschlüssel der Superklasse hinzu.

`Angestellter( ANummer )`

`Techniker ANummer )`

`Ingenieur( ANummer ,Ausbildung)`

3.2.10 Anwendung der Abbildungsregeln

Die nachstehende Interaktion ermöglicht das Üben der acht Abbildungsregeln. In einem ersten Schritt sollen jeweils die zur aktuellen Regel passenden Gegenstandstypen selektiert werden. Danach muss das entsprechende relationale Schema für diese Gegenstandstypen gewählt werden.

Um die Flash-Animation zu starten, klicken Sie bitte auf die Grafik.

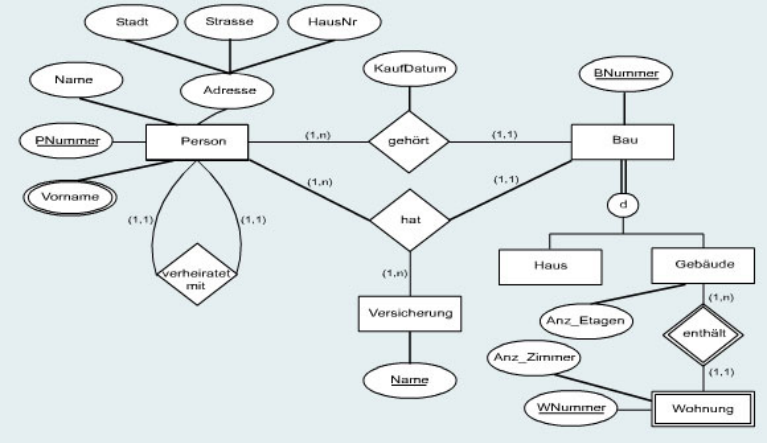

Abbildung 50: Interaktion: Anwendung Abbildungsregeln

3.2.11 Umwandlung vom konzeptionellen Schema zum logischen Schema

INSTALLATIONSHINWEISE (TUTOREN): **Diskussionsforum 'Uebung Umwandlung' aufsetzen**

Wandeln Sie das nachstehende konzeptionelle Schema in ein relationales Datenbankschema um, indem Sie die in dieser Unit eingeführten Regeln anwenden. Um die Darstellung zu vereinheitlichen unterstreichen Sie bitte die Identifikationsschlüssel und stellen Sie die Fremdschlüssel kursiv dar.

Dokumentieren Sie Ihre Lösung in einer Word- oder PDF-Datei und veröffentlichen Sie diese auf dem Diskussionsforum unter dem Thema „Übung Umwandlung". Schauen Sie sich die Lösungen Ihrer Mitstudenten an und kommentieren Sie diese gegebenenfalls. Falls Sie Probleme oder Fragen haben, können Sie diese ebenfalls im Diskussionsforum stellen und diskutieren. Die veröffentlichten Lösungen werden von einem Tutor angeschaut und in einem allgemeinen Feedback kommentiert.

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 51: Konzeptionelles Datenmodell

3.3 Datenintegrität

Unter dem Begriff Integrität oder Konsistenz (engl. integrity, consistency) versteht man die Widerspruchsfreiheit von Datenbeständen. Eine Datenbank ist integer oder konsistent, wenn die gespeicherten Daten fehlerfrei erfasst sind und den gewünschten Informationsgehalt korrekt wiedergeben. Die Datenintegrität ist dagegen verletzt, wenn Mehrdeutigkeiten oder widersprüchliche Sachverhalte zutage treten.

3.3.1 Schlüssel-Integritätsbedingung

Relationen sind Mengen von Tupeln, die allein durch ihre Werte unterschieden werden. Der Begriff Menge impliziert Eindeutigkeit der Elemente, d. h. es kann in einer Menge nicht zwei Elemente geben, die die gleichen Werte besitzen. Tupel müssen folglich eindeutig identifizierbar sein. Somit muss also auch jeder Schlüssel eindeutig sein.

Schlüsselkandidat: Jedes Attribut oder jede minimale Attributkombination, die alle Tupel einer Relation eindeutig identifiziert, ist ein Schlüsselkandidat. Dabei bedeutet „minimal“, dass kein Attribut ohne Verlust der eindeutigen Identifizierbarkeit weggelassen werden kann.

Primärschlüssel: Der unter den Schlüsselkandidaten ausgewählte Identifikationsschlüssel wird zum Primärschlüssel der Relation.

Primärschlüssel werden meist unterstrichen dargestellt.

3.3.2 Gegenstands-Integritätsbedingung

Die Gegenstands-Integritätsbedingung folgt direkt aus der Schlüssel-Integritätsbedingung und besagt, dass kein Primärschlüsselwert NULL (=kein Wert) sein darf. Erlauben wir NULL-Werte für Schlüsselattribute, so könnten mehrere Tupel NULL als Schlüsselwert besitzen. Damit wären diese Tupel nicht mehr eindeutig identifizierbar und die Schlüsselbedingung verletzt.

Beispiel:

ID	**Name**	**Vorname**	**Geburtsjahr**
NULL	Meier	Hans	1955
NULL	Meier	Hans	1985

Tabelle 1: Legende fehlt

Da der Primärschlüsselwert (Attribut ID) bei beiden Tupeln leer (NULL) ist, können wir die beiden Tupel nicht direkt identifizieren.

3.3.3 Referenzielle Integritätsbedingung

Im Gegensatz zum GBM gibt es im relationalen Modell kein Konstrukt, mit dem Beziehungen zwischen Tupeln explizit modelliert werden können. Beziehungen werden hier implizit mit Hilfe des Primär-Fremdschlüssel-Konzepts dargestellt.

Fremdschlüssel: Ein Attribut in einem Relationenschema R1 ist ein Fremdschlüssel wenn es in Beziehung zu einem Primärschlüsselattribut aus R2 steht und es gilt:

- Die Domäne des Fremdschlüssels aus R1 ist identisch mit der Domäne des Primärschlüssels aus R2.
- Die Menge der Fremdschlüssel-Attributwerte von R1 muss eine Teilmenge der vorhandenen Primärschlüssel-Attributwerte von R2 sein.

Fremdschlüssel werden meist gestrichelt unterstrichen dargestellt.

Eine Beziehung zwischen zwei Relationen wird nun so hergestellt, dass die Domänen des Primärschlüssels der einen Relation in die zweite Relation als Fremdschlüsseldomänen (mit entsprechenden Attributen) aufgenommen werden.

Referenzielle Integritätsbedingungen verlangen, dass aktuelle Fremdschlüsselwerte sich immer nur auf Primärschlüsselwerte von existierenden Tupeln beziehen.

Formal ausgedrückt heisst dies:

Gegeben sind:

- Ein Tupel `t1` einer Relation `R1` mit dem Schema `R1=(A1, ..., Am, ...)` , wobei `Ai` die Primärschlüssel-Attribute sind.
- Ein Tupel `t2` einer Relation `R2` mit dem Schema `R2=(B1, B2, ..., Bn, ..., A1, ..., Am, ...)` , wobei `Ai` die Fremdschlüssel-Attribute sind.

Das Tupel `t2` ist genau dann referenziell integer, wenn ein Tupel `t1` existiert mit der Eigenschaft `t1.Ai=t2.Ai (i=1, ..., m)` oder wenn `t2.Ai =NULL (i=1, ..., m)` gilt.

Bildlich betrachtet, muss jedes Tupel der Fremdschlüssel-Relation ein Tupel der Primärschlüssel-Relation referenzieren (oder auf NULL gesetzt sein).

Daher spricht man im Zusammenhang mit referenzieller Integrität auch davon, dass keine „hängenden Referenzen“existieren dürfen (also Verweise auf etwas, das nicht existiert).

Das Relationenschema mit dem Fremdschlüssel wird als *referenzierendes* , das mit dem entsprechenden Primärschlüssel als *referenziertes Relationenschema* bezeichnet. Ein Fremdschlüssel kann dabei auch das eigene Relationenschema referenzieren. Die meisten der heutigen relationalen Datenbanksysteme unterstützen die automatische Einhaltung der referenziellen Integrität.

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 52: Beispiel zur Referenziellen Integrität

Die Tabelle „Abteilung“hat die Abteilungsnummer als Primärschlüssel. Dieser wird in der Tabelle Mitarbeiter als Fremdschlüssel verwendet, um die Abteilungszugehörigkeit eines Mitarbeiters zu fixieren. Die Fremd-Primärschlüssel-Beziehung erfüllt die Regel der referenziellen Integrität, falls alle Abteilungsnummern des Fremdschlüssels aus der Tabelle „Mitarbeiter“in der Tabelle „Abteilung“als Primärschlüsselwerte aufgeführt sind. In unserem Beispiel ist also die Regel der referentiellen Integrität nicht verletzt.

Nehmen wir an, wir möchten in die Tabelle „Mitarbeiter“, ein neues Tupel „Id: 4, Weber, A5“einfügen. Unsere Einfügeoperation wird abgewiesen, wenn das Datenbanksystem die referenzielle Integrität unterstützt. Der Wert A5 wird nämlich als ungültig erklärt, da er in der referenzierten Tabelle „Abteilung“nicht vorkommt.

3.3.4 Integritätsgefährdende Operationen

Wir unterscheiden und erläutern drei Arten von Operationen, die die Integrität im besprochenen Sinne gefährden können:

- Einfügen von Tupeln
- Löschen von Tupeln
- Ändern von Attributwerten eines Tupels

Bei all diesen Operationen sind alle Arten von Integritätsbedingungen zu beachten und einzuhalten. Anfragen, also das Wiederfinden von Daten in einer Datenbank, stellen als reine Leseoperationen natürlich keine integritätsgefährdende Operation dar.

Einfügen von Tupeln

Bei dieser Operation sind alle drei Integritätsbedingungen betroffen. Folgende Verletzungen von Integritätsbedingungen sind möglich:

- In einem existierenden Tupel gibt es bereits den zur Einfügung vorgesehenen Primärschlüsselwert.
- Der Primärschlüsselwert des neuen Tupels ist NULL.
- Zu einem neuen Fremdschlüsselwert existiert kein zugehöriger Primärschlüsselwert.

Integritätsverletzende Operationen müssen vom Datenbanksystem entweder zurückgewiesen oder nach einem vereinbarten Protokoll in zulässige Operationen umgewandelt werden.

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 53: Beispiel für das Einfügen von Tupeln

Löschen von Tupeln

In diesem Fall kann nur die referenzielle Integrität verletzt werden. Referenzierende Tupel können durch das Löschen des referenzierten Tupels betroffen sein, indem ihr Fremdschlüsselwert ungültig wird. Geeignete Gegenmassnahmen bestehen entweder im Zurückweisen der Löschoperation, im kaskadierenden Löschen (das referenzierende Tupel wird automatisch mitgelöscht) oder im Setzen der Fremdschlüsselattribute auf NULL.

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 54: Beispiel für das Löschen von Tupeln

Durch das Löschen des Tupels mit der „Abt_Nr“„A1“in der Relation Abteilung würde in der Relation Mitarbeiter im Tupel mit der ID „1“die Abteilungsnummer ungültig werden.

Ändern von Attributwerten

Diese Operation betrifft wiederum alle drei Arten von Integritätsbedingungen. Da eine Änderung auch als Kombination aus dem Löschen des alten Tupels und dem Einfügen des geänderten Tupels aufgefasst werden kann, treten hier die gleichen Probleme wie beim Einfügen- und Löschen von Tupeln auf. Werteänderungen sind nur bei Primär- und Fremdschlüsseln kritisch. Alle anderen Attribute können problemlos geändert werden.

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 55: Beispiel Ändern von Tupeln

Durch das Ändern des Wertes „Abt_Nr“in der Relation Abteilung von „A1“zu „A4“würde in der Relation Mitarbeiter im Tupel mit der ID „1“die Abteilungsnummer ungültig werden.

3.4 Normalisierungsprozess

Unter Normalisierung versteht man die systematische Untersuchung einer Relation mit dem Zweck, qualitativ hochwertige Relationen zu erhalten. Eine Relation ist dann normalisiert, wenn sie folgende Eigenschaften aufweist:

- Sie ist redundanzfrei.
- Sie verursacht keine Probleme bei der Datenpflege.
- Sie beschreibt einen Ausschnitt aus der Realität angemessen und richtig.

Unnormalisierte Relationen enthalten nichtatomare Attribute, d. h. die Attribute selbst besitzen eine gewisse Struktur z. B. eine Menge von Werten. Dies kann aber bereits durch eine sorgfältige ER-Modellierung verhindert werden.

Normalerweise sollte der Normalisierungsprozess schon auf konzeptueller Ebene stattfinden. Sein Ziel besteht darin, eine gewisse Einheitlichkeit in den Entwurf zu bringen. Allerdings sind bisher die für diesen Prozess erforderlichen Schritte nur auf der Ebene des relationalen Datenmodells sauber definiert worden. Daher werden auch wir die Normalisierung auf logischer Ebene betrachten. Der Entwerfer eines konzeptuellen Schemas sollte sich über die Normalisierungsprinzipien im Klaren sein, um mit einem korrekten konzeptuellen Schema die Implementierung eines korrekten logischen Schemas zu erleichtern.

3.4.1 Abhängigkeiten

Um die Umwandlung der Relationen in die drei Normalformen zu verstehen, müssen wir zuerst das Konzept der Abhängigkeiten zwischen Attributen dieser Relationen einführen.

Funktionale Abhängigkeit: Attribut B eines Gegenstandstyps G ist von Attribut A funktional abhängig, wenn zu jedem Wert von A höchstens ein Wert von B auftreten kann.

G.A –> G.B

Beispiel:

ID	**Name**
S1	Meier
S2	Weber

Tabelle 2: Legende fehlt

Das Attribut Name ist funktional abhängig vom Attribut ID (ID –> Name).

Identifikationsschlüssel: Ein Attribut A für das gilt: Jedes Attribut von G ist von A funktional abhängig; kein Attribut von A ist von den übrigen A-Attributen funktional abhängig.

G.A –> G.B

Beispiel:

ID	**Name**	**Vorname**
S1	Meier	Hans
S2	Weber	Ueli

Tabelle 3: Legende fehlt

Das Attribut ID ist Identifikationsssschlüssel.

Volle funktionale Abhängigkeit: A sei der Identifikationsschlüssel eines Gegenstandstyps G, B Attribut; B ist genau dann von A voll funktional abhängig, wenn B von A funktional abhängig ist, aber nicht bereits von Teilen von A.

G.A ==> G.B

Beispiel:

IDStudent	**Name**	**IDProfessor**	**Note**
S1	Meier	P2	5
S2	Weber	P1	6

Tabelle 4: Legende fehlt

Das Attribut „Note“ist voll funktional abhängig von den Attributen „IDStudent“und „IDProffesor“(„IDSt, IDProf ==> Note“).

Transitive Abhängigkeit: A sei der Identifikationsschlüssel eines Gegenstandstyps G, B und C sind weitere Attribute, alle untereinander verschieden/disjunkt; C ist transitiv abhängig von A wenn gilt:

G.A –> G.B ; G.B –> G.C ; G.B -/-> G.A

Beispiel:

ID	**Name**	**Konto_Nr**	**Bank_Clearing_Nr**	**Bank**
L1	Meier	1234-5	836	UBS
L2	Weber	5432-1	835	CS

Tabelle 5: Legende fehlt

Die funktionale Abhängigkeit bezüglich „Bank_Clearing_Nr –> Bank“ist eine transitive Abhängigkeit, da „Bank_Clearing_Nr“nicht Primärschlüssel der Relation ist.

3.4.2 Erste Normalform

1. Normalform: Ein Relationenschema befindet sich in der 1. Normalform, wenn alle seine Attribute einfach und einwertig sind.

Zur Verwaltung der Studenten sei folgende Relation gegeben:

`Student(Vorname, Nachname, Informatikkentnisse)`

Im Attribut „Informatikkenntnisse“können mehrere Werte stehen.

Die Attribute „Vorname“und „Nachname“sind einfach und einwertig.

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 56: Beispiel 1. Normalform

Um die 1. Normalform zu erreichen, muss für jeden Wert eines mehrwertigen Attributes ein separates Tupel erzeugt werden.

3.4.3 Zweite Normalform

2. Normalform: Ein Relationenschema ist in der 2. Normalform, wenn es in der 1. Normalform ist und wenn jedes nicht zum Identifikationsschlüssel gehörige Attribut von diesem voll funktional abhängig ist.

Zur Verwaltung der Prüfungsnoten sei folgende Relation gegeben:

`Student(IDSt, StudentNachname, IDProf, ProfessorNachname,Note)`

Die Attribute IDSt und IDProf bilden den Identifikationsschlüssel.

Alle Attribute sind einfach und einwertig.

Zudem ist bekannt, dass folgende funktionale Abhängigkeiten existieren:

1. Das Attribut „ProfessorNachname“ist funktional abhängig vom Attribut „IDProf“(„IDProf –> ProfessorNachname“)

2. Das Attribut „StudentNachname“ist funktional abhängig vom Attribut „IDSt“(„IDSt –> StudentNachname“)

3. Das Attribut „Note“ist voll funktional abhängig von den Attributen „IDSt“und „IDProf“(„IDSt, IDProf ==> Note“)

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 57: Beispiel 2. Normalform

Die obere Tabelle ist in der 1. Normalform, da alle Attribute einfach und einwertig sind. Wenn jedoch der Student 1 von der Schule abgeht und gelöscht wird, gehen auch alle Informationen über den Professor Schmid verloren. Das Attribut „Professor“ist nämlich nicht voll funktional abhängig vom Identifikationsschlüssel „IDSt“. Um dieses Problem zu lösen, wird eine neue Relation „Professoren“mit den Attributen „ProfID“und „Professor“geschaffen. Die Relation „Noten“dient dazu, die beiden Relationen „Studenten“und „Professoren“zu verbinden und die Noten zu verwalten. In die Relation „Noten“wird die ID des Professors und des Studenten eingefügt. Auf diese Weise können die drei Relationen miteinander verknüpft werden. Das Problem des Verlusts von Professoren-Informationen beim Löschen von Studenten ist damit behoben.

3.4.4 Dritte Normalform

3. Normalform: Ein Relationenschema befindet sich in der 3. Normalform, wenn es in der 2. Normalform ist und kein Attribut, das nicht zum Identifikationsschlüssel gehört, von diesem transitiv abhängt.

Zur Verwaltung der Bankverbindung von Lieferanten sei folgende Relation gegeben:

`Lieferant(ID, Name, Konto_Nr, Bank_Clearing_Nr, Bank)`

Das Attribut ID ist Identifikationsschlüssel. Alle Attribute sind einfach und einwertig.

Zudem ist bekannt, dass folgende funktionale Abhängigkeiten existieren:

1. Name, Konto_Nr, Bank_Clearing_Nr sind funktional abhängig von ID (ID –> Name, Konto_Nr, Bank_Clearing_Nr)
2. Bank ist funktional abhängig von Bank_Clearing_Nr (Bank_Clearing_Nr –> Bank)

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 58: Beispiel 3. Normalform

Da alle Attribute einfach und einwertig sind, befindet sich die Ausgangsrelation in der 1. Normalform. Ausserdem befindet sie sich ebenfalls in der 2. Normalform, da alle Attribute vom Identifikationsschlüssel voll funktional abhängig sind. Die funktionale Abhängigkeit bezüglich Bank_Clearing_Nr –> Bank ist eine transitive Abhängigkeit, da Bank_Clearing_Nr nicht Primärschlüssel der Relation „Lieferant“ist. Um die 3. Normalform zu erreichen, wird eine neue Relation erstellt und die bestehende geändert.

3.4.5 Übung Normalisierung

Installationshinweise (Tutoren): **Discussion topic „Normalisierungsübung“aufsetzen, Studentenpostings kommentieren**

Die nachstehende Tabelle ist bereits in der 1. Normalform. Sie enthält pro Zeile und Spalte nur einen Eintrag. Ihre Aufgabe ist es, diese Relation so zu normalisieren, dass sie am Ende in der 3. Normalform vorliegt.

Dokumentieren Sie Ihre Lösung in einer Word- oder PDF-Datei und veröffentlichen Sie diese auf dem Diskussionsforum unter dem Thema „Normalisierungsübung“. Schauen Sie sich die Lösungen Ihrer Mitstudierenden an und kommentieren Sie diese gegebenenfalls. Wenn Sie Probleme oder Fragen haben, können Sie diese ebenfalls im Diskussionsforum stellen und diskutieren.

Die veröffentlichten Lösungen werden von einem Tutor angeschaut und in einem allgemeinen Feedback kommentiert.

Relation zur Verwaltung der Noten von Studenten in den diversen Fächern

UnitID	**StudentID**	**Datum**	**TutorID**	**Fach**	**Raum**	**Note**	**Buch**	**TutEmail**
U1	St1	23.02.03	Tut1	GMT	629	4.7	Deumlich	tut1@fhbb.ch
U2	St1	18.11.02	Tut3	GIn	631	5.1	Zehnder	tut3@fhbb.ch
U1	St4	23.02.03	Tut1	GMT	629	4.3	Deumlich	tut1@fhbb.ch
U5	St2	05.05.03	Tut3	PhF	632	4.9	Dümmlers	tut3@fhbb.ch
U4	St2	04.07.03	Tut5	AVQ	621	5.0	Swiss-Topo	tut5@fhbb.ch

Tabelle 6: Legende fehlt

3.4.6 Zusammenfassung der Unit

Diese Unit hat aufgezeigt, wie und warum Relationen stufenweise normalisiert werden sollten. Unnormalisierte Relationen verursachen Probleme und damit auch Kosten. Eine sorgfältige Planung der Entitäten (im konzeptionellen Modell) und die Normalisierung des logischen Modells können helfen dies zu vermeiden. Relationen in der 3. Normalform werden oft als „normalisiert“bezeichnet. Im Grunde gäbe es noch weitere Normalformen nämlich die vierte und fünfte. Die Anomalien, die vermieden werden durch eine Normalisierung bis zum fünften Grad, sind aber sehr selten, und so genügt es meistens, bis zur dritten Normalform, zu normalisieren.

3.5 Zusammenfassung

Objekte der realen Welt werden im Relationenmodell durch Tabellen dargestellt. Jede Tabelle setzt sich aus Zeilen und Spalten zusammen. Die Tabelle ist also eine Sammlung aller zugehörigen Zeilen. Jede einzelne Zeile einer Tabelle, die auch als Tupel bezeichnet wird, setzt sich aus Datenfeldern zusammen, den Attributen. Die Attribute repräsentieren bestimmte Merkmale des entsprechenden Objektes der realen Welt. Jedes Attribut setzt sich aus einem bestimmten Attributnamen und einem Attributwert zusammen.

Relationen zwischen den einzelnen Tupeln sollen bestehende Beziehungen oder einen bestimmten Sachverhalt zwischen zwei Tabellen ausdrücken. Zusätzlich werden sogenannte Schlüsselattribute vergeben, um zum einen die Zuordnung (Relation oder Beziehung) zwischen Objekten (Tabellen) darzustellen und zum anderen den Zugriff auf eine Tabelle eindeutig zu definieren. Üblicherweise werden die Schlüsselattribute der jeweiligen Relation unterstrichen.

Unter dem Begriff Integrität oder Konsistenz (engl. integrity, consistency) versteht man die Widerspruchsfreiheit von Datenbeständen. Eine Datenbank ist integer oder konsistent, falls die gespeicherten Daten fehlerfrei erfasst sind und den gewünschten Informationsgehalt korrekt wiedergeben. Die Datenintegrität ist dagegen verletzt, wenn Mehrdeutigkeiten oder widersprüchliche Sachverhalte zutage treten.

Ein Relationsschema sollte einen Entity-Typ oder einen Beziehungstyp beschreiben. Das heisst eine Relation sollte Informationen beinhalten, die tatsächlich auch logisch zusammengehören. Wird dies nicht beachtet, kann es zu Anomalien kommen. Um die Anomalien mehr oder weniger vollständig zu vermeiden, werden verschiedene Normalformen für relationale Datenbankschemas vorgeschlagen.

3.6 Literaturempfehlungen

- ELMASRI, R.; NAVATHE, S. B., 1994 *Fundamentals of Database Systems* [**?**, Einführung ins Thema Datenbanken und SQL, auf Englisch]

Glossar

1. Normalform: Ein Relationenschema befindet sich in der 1. Normalform, wenn alle seine Attribute einfach und einwertig sind.

2. Normalform: Ein Relationenschema ist in der 2. Normalform, wenn es in der 1. Normalform ist und wenn jedes nicht zum Identifikationsschlüssel gehörige Attribut von diesem voll funktional abhängig ist.

3. Normalform: Ein Relationenschema befindet sich in der 3. Normalform, wenn es in der 2. Normalform ist und kein Attribut, das nicht zum Identifikationsschlüssel gehört, von diesem transitiv abhängt.

Attribut: Ein Attribut A bezeichnet die Funktion, die eine Domäne D in einem Relationenschema R ausübt. Es kann auch als Abbildung der Tupel einer Relation auf den Wert des jeweiligen Tupels (für dieses Attribut) verstanden werden, wobei jeder Wert di ein Element der Domäne oder NULL sein muss.

Domäne: Eine Domäne besteht aus einem Namen D und einer Menge atomarer Werte. Ein anderer Name für Domäne ist Wertebereich. Domänen definieren den Wertebereich von Attributen.

Fremdschlüssel: Ein Attribut in einem Relationenschema R1 ist ein Fremdschlüssel wenn es in Beziehung zu einem Primärschlüsselattribut aus R2 steht und es gilt:

- Die Domäne des Fremdschlüssels aus R1 ist identisch mit der Domäne des Primärschlüssels aus R2.
- Die Menge der Fremdschlüssel-Attributwerte von R1 muss eine Teilmenge der vorhandenen Primärschlüssel-Attributwerte von R2 sein.

Fremdschlüssel werden meist gestrichelt unterstrichen dargestellt.

Funktionale Abhängigkeit: Attribut B eines Gegenstandstyps G ist von Attribut A funktional abhängig, wenn zu jedem Wert von A höchstens ein Wert von B auftreten kann.

G.A –> G.B

Identifikationsschlüssel: Ein Attribut A für das gilt: Jedes Attribut von G ist von A funktional abhängig; kein Attribut von A ist von den übrigen A-Attributen funktional abhängig.

G.A –> G.B

Primärschlüssel: Der unter den Schlüsselkandidaten ausgewählte Identifikationsschlüssel wird zum Primärschlüssel der Relation.

Primärschlüssel werden meist unterstrichen dargestellt.

Relation: Eine Relation r ist eine Instanz (Ausprägung) des Relationenschemas R(A1, A2, ..., An). Sie ist eine Teilmenge des kartesischen Produkts (Kreuzprodukt) der beteiligten Domänen.

Relationales Datenbankschema: Ein relationales Datenbankschema ist eine Menge von Relationenschemata S = R1, ..., Rn zusammen mit einer Menge von Integritätsbedingungen. Eine relationale Datenbankinstanz ist die Menge r1, ..., rn , wobei ri Instanz von Ri ist und alle Integritätsbedingungen erfüllt sind. Eine relationale Datenbank ist ein relationales Datenbankschema mit einer entsprechenden Datenbankinstanz.

Relationenschema: Ein Relationenschema R, Schreibweise: R(A1, A2, ..., An), bezeichnet eine Menge von Attributen A1, A2, ..., An.

Schlüsselkandidat: Jedes Attribut oder jede minimale Attributkombination, die alle Tupel einer Relation eindeutig identifiziert, ist ein Schlüsselkandidat. Dabei bedeutet „minimal“, dass kein Attribut ohne Verlust der eindeutigen Identifizierbarkeit weggelassen werden kann.

Transitive Abhängigkeit: A sei der Identifikationsschlüssel eines Gegenstandstyps G, B und C sind weitere Attribute, alle untereinander verschieden/disjunkt; C ist transitiv abhängig von A wenn gilt:

G.A –> G.B ; G.B –> G.C ; G.B -/-> G.A

Tupel: Ein Tupel t ist eine Liste mit n Werten t = <d1, d2, ..., dn>, wobei jeder Wert di ein Element der Domäne Di, oder NULL sein muss.

Volle funktionale Abhängigkeit: A sei der Identifikationsschlüssel eines Gegenstandstyps G, B Attribut; B ist genau dann von A voll funktional abhängig, wenn B von A funktional abhängig ist, aber nicht bereits von Teilen von A.

G.A ==> G.B

4 Die relationale Anfragesprache SQL

SQL (Structured Query Language) ist eine standardisierte Abfragesprache, die alle erforderlichen Sprachelemente enthält, um sämtliche Arbeiten, die beim Umgang mit einer relationalen Datenbank anfallen, auszuführen. Erste Versionen von SQL hiessen SEQUEL, SEQUEL2. Sie gingen aus SQUARE hervor, einer eher mathematisch orientierten Sprache. SQL ist ein ISO- und ANSI-Standard. Da sich SQL im Laufe ihrer Geschichte weiterentwickelt hat, kann man jedoch nicht von einem einzigen Standard sprechen. Der derzeit gültige Standard ist SQL-99. In den folgenden Units geben wir eine vereinfachende Übersicht von SQL.

Diese Lektion orientiert sich am Standard SQL-99 [**?**] . Dieser wird jedoch nicht von allen Datenbanksystemen konsquent umgesetzt, z.B. sind gewisse Befehle anders benannt oder sind syntaktisch anders aufgebaut. Aus diesem Grund könnten SQL Befehle aus dieser Lektion in gewissen Systemen nicht funktionieren. In solchen Fällen muss anhand der jeweiligen Benutzeranleitungen festgestellt werden, ob der Befehl überhaupt unterstützt wird, anders benannt ist oder syntaktisch anders aufgebaut ist. Die meisten Datenbanksysteme gehen auch über den Standard hinaus. Bei diesen SQL Erweiterungen ist jedoch Vorsicht geboten, da sie sich von System zu System stark unterscheiden können.

Lernziele

- Sie verstehen die grundsätzlichen SQL-Konzepte und können den Einsatz von SQL in den Bereichen Datendefinition, Datenmanipulation und Datenkontrolle erläutern.
- Sie sind in der Lage, SQL für das Erstellen, Ändern und Löschen von Tabellen einzusetzen.
- Sie können mit Hilfe von SQL einfache und komplexe Anfragen formulieren und wissen, wie SQL-Anfragen in Schachtelungen oder im Verbund eingesetzt wird. Sie sind zudem in der Lage arithmetische Operatoren und Mengenoperatoren innerhalb von SQL-Anfragen korrekt anzuwenden.
- Sie beherrschen SQL beim Einfügen, Ändern und Löschen von Daten-Tupeln.

4.1 SQL-Konzepte

Die Sprache SQL (Structured Query Language) wird als einer der Hauptgründe für den kommerziellen Erfolg von relationalen Datenbanken in der Geschäftswelt angesehen. Die ANSI (American National Standards Institute) und die ISO (International Standards Organization) entwickelten eine Standard-Version von SQL (ANSI 1986) mit dem Namen SQL-86 oder SQL1. In den 90er Jahren wurde der erweiterte Standard SQL2 oder SQL-92 angewendet. SQL3 oder SQL-99 ist der im Moment gültige Standard. Durch den Einsatz von SQL in kommerziellen DBMS-Produkten wurde die Migration zwischen einzelnen DBMS-Produkten, die den Standard implementiert haben, für den Endbenutzer vereinfacht. Der Benutzer von relationalen Datenbanken muss sich im Idealfall keine Gedanken mehr machen, mit welchem DBMS-Produkt er momentan arbeitet, denn die Schnittstelle (SQL) zu den einzelnen Systemen bleibt für ihn immer die gleiche.

4.1.1 SQL-Grundlagen

Die Grundlage für die Datenmanipulation im Relationenmodell bildet die relationale Algebra. In ihrer unmittelbaren Form hat sie heute jedoch als Datenmanipulationsprache (DML) stark an Bedeutung verloren, da sie für Laien nur schwer zu durchschauen ist. SQL (Structured Query Language) hingegen hat sich seit Mitte der 80er Jahre als DML für relationale Systeme durchgesetzt. Es handelt sich dabei um eine deskriptive, mengenorientierte Sprache. Sie ist sowohl selbständig als auch eingebettet in eine Wirtssprache (C, C++, Java, PHP, ASP, u.v.m.) verfügbar.

Die Möglichkeiten von SQL beschränken sich nicht allein auf die Datenmanipulation (DML). SQL kann in drei konzeptionelle Einheiten aufgeteilt werden.

- Datendefinition (DDL - Data Definition Language) - zuständig für die Erstellung und Veränderung der Struktur der Datenbank
- Datenmanipulation (DML - Data Manipulation Language) - zuständig für den (Daten-) Inhalt der Datenbank (Daten hinzufügen, ändern, löschen, abfragen,...)
- Datenkontrolle / -steuerung (DCL - Data Controlling Language) - zuständig für die Sicherheit der Datenbank

Zusätzlich wurden in den meisten Implementierungen von SQL Konstrukte aus Programmiersprachen hinzugefügt (Fallabfragen, Iterationen etc.), so dass Spracherweiterungen wie PL/SQL (Oracle) schon fast als eigenständige Programmiersprachen angesehen werden können.

4.1.2 Datendefinition (DDL)

In SQL werden die Begriffe `TABLE` , `ROW` und `COLUMN` synonym für Relation, Tupel und Attribute gebraucht. Mittels des `CREATE TABLE` -Befehls wird ein Relationenschema in der Datenbank definiert. Das Relationenschema muss genau spezifiziert werden, oder in anderen Worten, die zur Relation gehörenden Attribute sowie deren Domänen müssen angegeben werden. Zusätzlich sind noch eine Reihe weiterer Deklarationen möglich wie z. B. Wertebeschränkungen (`CHECK` -Klausel), Standardwerte oder Primär- und Fremdschlüsseldeklarationen.

Beispiele für Domänen in SQL sind `CHAR, NUMBER, LONG` und `DATE` . Durch die Deklaration `NOT NULL` wird festgelegt, dass für das jeweilige Attribut keine NULL-Werte zulässig sind. Folglich muss beim Einfügen eines Tupels grundsätzlich ein Wert für dieses Attribut angegeben werden (es sei denn, ein Wert wird vorgeschrieben oder automatisch generiert). Primärschlüssel werden durch eine sogenannte Relationenbedingung (`TABLE CONSTRAINT`) mit Hilfe der `PRIMARY KEY` -Klausel deklariert.

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 59: Code-Beispiel: CREATE TABLE

Dies ist ein Beispiel, wie eine Datenbanktabelle mit Hilfe von SQL definiert werden kann. Teile der Datendefinition (DDL) werden in der Unit Erstellen und Ändern von Tabellen (Seite 101) vertieft behandelt. Für den Moment müssen Sie dieses Beispiel-SQL-Statement nicht im Detail verstehen.

4.1.3 Datenmanipulation (DML)

Die Möglichkeiten der Datenmanipulation mit SQL können in zwei Kategorien eingeteilt werden. Die eine Gruppe bilden die Datenbankanfragen, welche Inhalte abfragen, aber keine Änderungen an den Daten vornehmen. Die zweite Gruppe sind die Datenmanipulationen, die die Datenbank verändern, indem sie zum Beispiel Daten einfügen, löschen oder abändern.

Die Datenbankanfragen werden in der Unit Datenbankanfragen (Seite 107) detaillierter behandelt. Die Unit Einfügen, Löschen und Ändern (Seite 125) befasst sich mit den Datenbankänderungen.

4.1.4 Datenkontrolle / -steuerung (DCL)

SQL-Befehle der Datenkontroll-Sprache (DCL) kontrollieren die Sicherheit und die Zugriffsrechte für Objekte oder Teile eines Datenbanksystems. Diese Befehle liegen näher bei der Sprache des DBMS und können in verschiedenen Implementierungen der SQL stark varieren.

Typische Befehle sind:

- `GRANT` - vergibt Zugriffsrechte
- `DENY` - verweigert Zugriffsrechte
- `REVOKE` - löscht vorher vergebene oder verweigerte Zugriffsrechte

Die Befehle zur Datenkontrolle oder -steuerung werden im Rahmen dieses Moduls nicht detailliert behandelt.

4.2 Erstellen und Ändern von Tabellen

In dieser Unit wird gezeigt, wie mittels SQL-Befehlen Tabellen erstellt, verändert und gelöscht werden können.

4.2.1 Tabellen erstellen

Mit `CREATE TABLE` kann in einer Datenbank eine neue Tabelle erstellt werden. Der Befehl hat folgende Grundstruktur:

```
CREATE TABLE <Tabellenname> (<Attributdefinitionen und Beschränkungen>);
```

Der Tabellenname muss innerhalb der aktuellen Datendank oder des aktuellen Schemas eindeutig sein.

Attributdefinition

Attribute werden mit einem Namen und einem Datentyp definiert, wobei der Namen innerhalb der Tabelle eindeutig sein muss. Diese Angaben sind für alle Attribute zwingend notwendig.

Die Reihenfolge der Attribute bei der Definition entspricht der Reihenfolge der Spalten in der erstellten Tabelle. Wird eine bestimmte Ordnung angestrebt ist diese bereits bei der Erstellung der Tabelle zu definieren. Danach bleibt sie, sofern keine Änderungen an der Struktur der Tabelle vorgenommen werden (siehe Tabellenstruktur ändern (Seite 104)) bestehen.

Beschränkungen

Es gibt zwei Arten von Beschränkungen: Tabellenbeschränkungen und Attributbeschränkungen. Der Unterschied liegt darin, dass Attributbeschränkungen sich auf nur ein Attribut beziehen und Tabellenbeschränkungen sich auf meherere Attribute beziehen können, dies aber nicht müssen. Mit diesen Beschränkungen kann der Wertebereich der Attribute eingeschränkt werden oder es wird verhindert, dass Werte eingegeben werden, die nicht erlaubt sind. Ein Datensatz kann nicht erfasst werden, wenn er eine Beschränkung verletzt.

Es gibt vier Arten von Beschränkungen, welche in der Folge kurz beschrieben werden:

- `UNIQUE` - das Attribut oder die Attributkombination muss innerhalb der Tabelle eindeutig sein
- `PRIMARY KEY` - das Attribut oder die Attributkombination ist Primärschlüssel der Tabelle
- `FOREIGN KEY` - das Attribut ist ein Fremdschlüssel
- `CHECK` - Bedingung die für ein Attribut oder eine Attributkombination erfüllt sein muss

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 60: CREATE TABLE Befehl

Die Beschränkugen können benannt werden. Dies ist jedoch nicht notwendig.

In diesem Beispiel wird einer Datenbank eine Tabelle hinzugefügt, weche mit einer bereits bestehenden Tabelle verknüpft ist. Im SQL Befehl kommt der Unterschied zwischen Attribut- und Tabellenbeschränkung zum Ausdruck. projekt_ID und leiter_ID haben eine Attributbeschränkung (Beschränkung wird direkt hinter die Attributdefiniton geschrieben). projekt_ID hat die Beschränkung PRIMARY KEY, ist also Primärschlüssel dieser Tabelle, d.h. das Attribut muss eindeutig und nicht NULL sein. leiter_ID hat die Beschränkung NOT NULL (Spezialfall einer CHECK Beschränkung), muss also immer einen Wert enthalten.

Der Verknüpfung mit der bestehenden Tabelle ist als Tabellenbeschränkung definiert (FOREIGN KEY) und hat einen Namen (projektleiter). Diese Beschränkung könnte jedoch auch als Attributbeschränkung definiert werden, da sie nur ein Attribut beinhaltet.

Das Beispiel zeigt, dass grundsätzlich kein Unterschied zwischen Attribut- und Tabellenbeschränkungen besteht, sofern nur ein Attribut betroffen ist. Es handelt sich dabei um zwei verschiedene Möglichkeiten Beschränkungen zu erfassen.

4.2.2 Tabellenstruktur verändern

Mit `ALTER TABLE` kann die Struktur einer Tabelle geändert werden. Es können somit die mit CREATE TABLE erzeugten Attribute und Beschränkungen geändert, neue hinzugefügt oder vorhandene gelöscht werden. Der Befehl hat folgende Syntax:

`ALTER TABLE <Tabellenname> <Änderung>;`

wobei `<Änderung>` verschieden Befehle beinhalten kann:

- `ADD [COLUMN] <Attributdefintion>`
  Attribut hinzufügen (Attributdefiniton wie bei CREATE)
- `ALTER [COLUMN] <Attributname> SET DEFAULT <Standardwert>`
  neuer Standardwert festlegen
- `ALTER [COLUMN] <Attributname> DROP DEFAULT`
  aktuellen Standardwert löschen
- `DROP [COLUMN] <Attributname> RESTRICT | CASCADE`
  löschen eines Attributes
- `ADD <Tabellenbeschränkung>`
  neue Tabellenbeschränkung hinzufügen (Tabellenbeschränkung wie bei CREATE)
- `DROP CONSTRAINT <Tabellenbeschränkung>`
  löschen einer Tabellenbeschränkung

Mit den oben genannten Befehlen können Attribute und Beschränkungen hinzugefügt bzw. gelöscht werden. Zudem können die Standardwerte für die Attribute gesetzt oder gelöscht werden. SQL sieht noch andere Befehle vor die hier nicht erwähnt sind.

SQL beinhaltet im Standard keine Befehle um Attribute zu ändern oder umzubnennen. Dies würde auch zu Problemen führen, wenn bereits Daten vorhanden sind. Denoch sind diese Befehle in einigen Datenbanken vorhanden (z.B. MODIFY oder RENAME). Die Syntax unterscheidet sich jedoch von System zu System. Wenn keine Daten vorhanden sind, kann das Attribut, das geändert werden soll, gelöscht und neu hinzugefügt werden.

In diesem Beispiel wird einer Tabelle ein neues Attribut hinzugefügt. Der angezeigte Datensatz enthält danach NULL für dieses Attribut, weil noch kein Wert zugewiesen wurde. Anschliessend wird dieses Attribut wieder aus

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 61: ALTER TABLE Befehl

der Tabelle gelöscht. Das Schlüsselwort `RESTRICT` bewirkt, dass nur Attribute gelöscht werden können, die nicht mit anderen Tabellen verbunden sind (Fremdschüssel). Alternativ kann das Schlüsselwort `CASCADE` verwendet werden. Dabei wird nicht nur die gewünschte Spalte, sondern auch die verbundene Spalte in der anderen Tabelle gelöscht.

4.2.3 Tabellen löschen

Mit DROP TABLE kann eine bestehende Tabelle gelöscht werden. Dabei werden die Daten und die Struktur der Tabelle gelöscht.

Der Befehl hat folgende Syntax:

```
DROP TABLE <Tabellenname>;
```

Achtung: Mit `DROP TABLE` wird die Tabellenstruktur samt den Daten gelöscht. Dies kann in den meisten Fällen nicht mehr rückgängig gemacht werden.

4.3 Datenbankanfragen

In dieser Unit wird besprochen, wie man mittels SQL Anfragen an die Datenbank stellt, um auf gegebene Fragestellungen die relevanten Daten aus der Datenbank herausziehen zu können.

4.3.1 SELECT-FROM-WHERE Struktur von SQL Anfragen

Mit Hilfe der SELECT-FROM-WHERE Struktur können Anfragen an eine Dabenbank gestellt werden. Anfragen bestehen aus einer Auswahl der gewünschten Spalten (SELECT) und einer Liste mit einer oder mehreren Relationen (FROM). In dieser einfachen Form werden immer alle Datensätze zurückgeben. Zusätzlich kann ein Suchkriterium übergeben werden (WHERE). Damit können gezielt Datensätze selektiert werden. Eine SQL Anfrage gibt als Resultat wiederum eine Relationen zurück.

Die Standardform einer Datenbankanfrage mittels SQL ist wie folgt aufgebaut:

```
SELECT <Attributliste>
FROM <Relationenliste>
WHERE <Bedingungen>;
```

Wobei:

- <Attributliste> besteht aus den Namen der Attribute, deren Werte man durch die Anfrage erhalten möchte.
- <Relationenliste> ist die Aufführung der Namen der Relationen, die für die Anfrage gebraucht werden.
- <Bedingungen> die jene Tupel identifizieren, die durch die Anfrage zurückgegeben werden sollen.

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 62: Standardanfrage

Damit haben wir die Grundstruktur von SQL als Anfragesprache bereits gezeigt. Es gibt nun eine Reihe von möglichen Erweiterungen des obigen Grundschemas, die die Flexibilität von Anfragen massiv erhöhen. Wir unterscheiden folgende Fälle:

- mehrere Bedingungen (konjunktiv oder disjunktiv verknüpft)
- komplexere Bedingungen (Unteranfragen, Bereichsanfragen, Verbundoperatoren)

- spezielle Selektionen (Verknüpfung von Attributen)
- nicht-relationale Konstrukte (Sortieren, Gruppierungen, Aggregatfunktionen)
- Mengenoperationen (Vereinigung, Durchschnitt, Differenz)

4.3.2 Verkettung von Bedingungen

Bedingungen im WHERE Teil einer Anfrage können beliebig durch die Anwendung der logischen Operatoren AND und OR miteinander verkettet werden. Einzelne Bedingungen können mit dem NOT-Operator negiert werden. Der Datensatz wird selektiert, wenn die gesamte Bedingung TRUE ergibt.

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 63: Verkettete Anfrage

Weiter Beispiele für verkettete Bedingungen:

- `PLZ = 8000 AND NOT Nachname = 'Schmidt'`
- `PLZ = 8000 OR PLZ = 8006`
- `(Nachname = 'Müller' OR Nachname = 'Meier') AND Ortsname = 'Zürich'`

In SQL Anfragen (z.B. in Bedingungen) muss Text in Hochkommas geschrieben werden. Numerische Werte werden ohne Hochkommas geschrieben. Dies ist in den Beispielen ersichtlich.

Statt einer langen Verkettung mit AND gibt es auch die Möglichkeit mehrere Attribute gleichzeitig miteinander zu vergleichen:

```
Vorname = 'Ursula' AND Nachname = 'Müller'
```

kann auch folgendermassen geschrieben werden:

```
(Vorname, Nachname) = ('Ursula', 'Müller')
```

4.3.3 Vergleichs- und Mengenoperationen

SQL unterstützt eine Reihe von Vergleichs- und Mengenoperatoren. Diese können dazu verwendet werden, Attribute mit Konstanten oder mit anderen Attributen zu vergleichen. Die Attribute können dabei aus derselben oder verschiedenen Relationen Stammen.

Folgende Vergleichsoperatoren für nummerische Attribute werden unterstützt:

- `=` gleich
- `>` grösser
- `<` kleiner
- `>=` grösser gleich
- `<=` kleiner gleich
- `<>` ungleich
- `BETWEEN` zwischen

Bedingungen haben grundsätzlich die Syntax `<Attribut> <Operator> <Wert>` . Der Operator `BETWEEN` hat eine etwas andere Syntax (siehe dazu das Beispiel).

Weitere Vergleichsoperatoren werden in den nächsten Abschnitten besprochen.

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 64: Vergleichsoperatoren

Im SELECT-Teil kann anstelle einer Attributliste auch ein * geschrieben werden. Der Effekt besteht darin, dass alle Attribute der im FROM Teil der Anfrage angegebenen Relationen ausgegeben werden.

4.3.4 Zeichenkettenvergleiche und arithmetische Operatoren

Vergleichsoperator LIKE

Durch den Vergleichsoperator LIKE kann eine Zeichenkette mit einem Ausschnitt aus einer Zeichenkette verglichen werden. Der Ausschnitt wird mit zwei reservierten Zeichen dargestellt:

- „%“ersetzt eine unbestimmte Anzahl von Zeichen (auch kein Zeichen)
- „_“ersetzt ein Zeichen

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 65: LIKE Anfrage

`NOT LIKE` untersucht ob das gegebene Ausschnitt nicht in der Zeichenkette vorhanden ist.

Arithmetische Operatoren

Die arithmetischen Standard-Operatoren für Addition (+), Subtraktion (-), Multiplikation (*) und Division (/) können für alle numerischen Werte oder Attribute mit numerischen Domänen eingesetzt werden. Damit können Attribute in einer Anfrage miteinander verrechnet werden.

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 66: Arithemische Operatoren Anfrage

Das obige Beispiel liefert eine Ergebnisrelation mit drei Attributen, deren drittes mit dem Namen „Wucher“versehen wurde (da „Preis / Grösse“kein sonderlich geeignete Bezeichnung ist) und den Preis je Grösseneinheit angibt. Als Wucher wird ein Verhältnis grösser als 15 angenommen.

Arithmetische Operatoren können im SELECT Teil einer Anfrage oder in Bedingungen verwendet werden.

Im SELECT oder im FROM Teil einer Anfrage kann mit dem Befehl `AS` Attributen oder Relationen ein anderer Namen gegeben werden: <`Attribut/Relation`> `AS` <`neuer Name`> (Das Schlüsselwort `AS` kann auch weggelassen werden). Dies kann dazu verwendet werden, um wie im Beispiel einem berechneten Wert einen sinnvollen Namen zu geben oder um SQL Anfragen lesbarer zu machen.

4.3.5 Geschachtelte Anfragen

Bedingungen haben prinzipiell die Struktur <Attribut> <Operator> <Wert> (z.B. Name=„John“) wobei die Werte wiederum Attribute, Konstanten oder aber das Resultat von Unteranfragen sein können, d. h. man muss die Werte von Bedingungen nicht unbedingt fix definieren, sondern kann sie mittels einer verschachtelten Anfrage generieren. Anfragen können so beliebig tief verschachtelt werden.

Es gibt drei Typen von Unteranfragen, welche sich durch ihr Resultat unterscheiden:

- Unteranfragen die einen Wert zurückgeben (eine Spalte und eine Zeile)
- Unteranfragen die eine Zeile zurückgeben
- Unteranfragen die meherere Zeilen zurückgeben

Wird nur ein Wert oder eine Zeile zurückgegeben, können die normalen Vergleichsoperatoren verwendet werden.

Werden mehrere Zeilen zurückgeben, kommen spezielle Operatoren zum Einsatz:

- `IN` sucht ob der Wert in der Unteranfrage vorkommt
- `<Vergleichsoperator> ALL` die Bedingung muss für alle Zeilen in der Unterabfrage TRUE ergeben
- `<Vergleichsoperator> ANY (SOME)` die Bedingung muss für mindestens eine Zeile in der Unterabfrage TRUE ergeben

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 67: Verschachtelte Anfrage

Unteranfragen können auch im FROM Teil einer Anfrage vorkommen. So kann eine Relation speziell für eine Anfrage zusammengestellt werden. Der Unteranfrage muss mit AS ein Namen zugewiesen werden: `(<Unterabfrage>) AS <Name>`

4.3.6 Verbund

Oft kommt es vor, dass in einer Anfrage Daten aus mehreren Relationen benötigt werden. Hierfür müssen die Relationen miteinander verknüpft werden. Dazu werden die identischen Attribute (Fremdschlüssel) in den beiden Relationen miteinander verbunden.

Es gibt zwei Möglichkeiten Relationen in Anfragen miteinander zu verküpfen:

im WHERE Teil

Bei dieser Möglichkeit werden die identischen Attribute mit dem Vergleichsoperator verknüpft und als "Bedingung" (keine eingentliche Bedingung) im WHERE Teil der Anfrage eingefügt.

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 68: Verbundssoperatoren

Grundsätzlich kann auch z.B. ein < für den Verbund verwendet werden. Dies ist geschieht jedoch meist in Kombination mit einer Verbundsbedingung mit `=` , weil ein solcher Verbund alleine keinen Sinn macht.

im FROM Teil

Eine andere Möglichkeit besteht darin, den Verbund im FROM Teil einer Anfrage vorzunehmen. Dies ist sinnvoller, da so die eigentlichen Suchkriterien (Bedingungen im WHERE-Teil) von den Tabellenverknüpfungen getrennt sind.

Dazu stehen folgende Befehle zur Verfügung:

- `<Relation> JOIN <Relation> USING (<Attribut>)`

  die Relationen werden über ein in beiden Tabellen gleich benanntes Attribut verknüpft

- `<Relation> NATURAL JOIN <Relation>`

  verknüpft automatisch alle Attribute die in beiden Relationen gleich benannt sind.

- `<Relation> JOIN <Relation> ON <Attribut> <Vergleichsoperator> <Attribut>`

bei dieser Variante kann angegeben werden, über welche Attribute beider Relationen verknüpft werden soll und welcher Operator verwendet werden soll (es können auch mehrere Verknüpfungen angegeben werden)

Das obige Beispiel würde mit diesen Befehlen folgendermassen aussehen:

```
SELECT Vorname, Nachname, Zeitung_Name
FROM Kunde JOIN Abonnement USING (KundNr);
```

oder

```
SELECT Vorname, Nachname, Zeitung_Name
FROM Kunde JOIN Abonnement ON Kunde.KundNr = Abonnement.KundNr;
```

oder

```
SELECT Vorname, Nachname, Zeitung_Name
FROM Kunde NATURAL JOIN Abonnement;
```

Die oben beschriebenen Befehle geben nur Datensätze aus, die in beiden Relationenen vorkommen. Sollen **alle** Datensätze der einen Relation mit den zugehörigen Datensätzen der zweiten Relation ausgegeben werden, kommen folgende Befehle zum Einsatz:

- `<Tabelle> RIGHT OUTER JOIN <Tabelle> USING (<Attribut>)`

  alle Datensätze der rechten Relation und die zugehörigen Datensätze der linken Relation

- `<Tabelle> LEFT OUTER JOIN <Tabelle> USING (<Attribut>)`

  alle Datensätze der linken Relation und die zugehörigen Datensätze der rechten Relation

Falls bei `RIGHT OUTER JOIN` in der linken Relation kein Datensatz verknüpft werden kann, wird für die Attribute dieser Relation NULL ausgegeben. Dies gilt ungekehrt auch für `LEFT OUTER JOIN` .

4.3.7 Nicht relationale Konstrukte

„ORDER BY“-Klausel

SQL unterstützt auch Konstrukte, die mit der ursprünglichen relationalen Algebra wenig bis gar nichts zu tun haben. So besitzt eine Menge per Definition keine Ordnung. Mittels des Konstrukts „ORDER BY“kann man jedoch eine Ordnung auf der Ergebnisrelation definieren. Die Schlüsselwörter „ASC“und „DESC“bezeichnen aufsteigende- resp. absteigende Anordung der Werte im Resultat. Wird keines dieser Schlüsselwörter angegeben, wird standardmässig aufsteigend sortiert.

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 69: ORDER BY Anfrage

Alle Kunden werden sortiert nach ihrem Namen (aufsteigend, wobei ASC als Standardeinstellung auch entfallen könnte) und ihrem Vornamen (absteigend, DESC) ausgegeben.

„GROUP BY“-Klausel

Gruppierungen („GROUP BY“-Klausel) dienen dazu, die Menge aller Tupel einer Relation nach bestimmten Kriterien in Teilmengen zu unterteilen, um für diese Teilmengen bestimmte Statistiken zu berechnen. Als Gruppierungskriterium dienen die Werte eines bestimmten Attributs. Alle Tupel, die für dieses Attribut den gleichen Wert besitzen, werden zu einer Gruppe zusammengefasst. Diese Gruppen können dann weiter bearbeitet werden (im Spezialfall auch durch eine weitere Gruppierung, um Gruppen von Gruppen zu bearbeiten etc.). Dazu dienen sogenannte Aggregatfunktionen, die nur auf numerische Attribute angewendet werden können. Folgende Aggregatfunktionen stehen üblicherweise zur Verfügung:

- `min` zum Bestimmen des minimalen Werts eines Attributs
- `max` zum Bestimmen des maximalen Werts eines Attributs
- `sum` zum Berechnen der Summe der selektierten Werte eines Attributs
- `count` zeigt die Anzahl der selektierten Werte eines Attributs (nicht der verschiedenen Werte!)

- `avg` zum Berechnen des Mittelwertes aller selektierten Werte eines Attributs, wobei NULL-Werte nicht in die Berechnung einfliessen.

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 70: GROUP BY Anfrage

Mittels dieser Anfrage werden die Nummern aller Kunden bestimmt, die mit Kleinanzeigen (Preis des einzelnen Inserats weniger als 300 Franken) schon Inserate für mehr als 200 Franken in Auftrag gegeben haben. Bei der Bearbeitung dieser Anfrage wird erst die Restriktion berücksichtigt (A004 wird aussortiert), dann werden die verbliebenen Tupel gruppiert, und zum Schluss wird geprüft, welche Tupel, die Gruppen repräsentieren, tatsächlich ausgegeben werden sollen (die Gruppen der Kunden 002 und 005 werden aussortiert, da sie noch nicht das Limit erreicht haben).

Anfragen, die eine GROUP BY-Klausel enthalten, werden folgendermassen abgearbeitet. Zuerst wird die Bedingung im WHERE Teil ausgeführt, falls eine solche vorhanden ist. Danach wird nach den angegeben Spalten gruppiert. Mit der Bedingung im HAVING Teil der Anfrage kann anschliessend, falls nötig, noch eine Bedingung für die gruppierten Attributwerte angegeben werden. Auf Grund dieser Reihenfolge ist ersichtlich, dass die Bedingung im WHERE Teil der Anfrage keine Aggregatfunktionen enthalten kann, weil zu diesem Zeitpunkt noch gar nicht gruppiert wurde. Alle Attribute, die in der Anfrage vorkommen müssen aber trotzdem entweder in der GROUP BY-Klausel vorkommen oder in einer Aggregatfunktion stehen.

4.3.8 Mengenoperationen

Mengenoperationen werden verwendet, um Tupel in einer Ergebnisrelation zusammenzufassen, die aus im Grunde verschiedenen Anfragen stammen. Voraussetzung dafür ist natürlich, dass die Anzahl der selektierten Attribute jeder beteiligten Anfrage identisch ist. Ausserdem müssen die jeweiligen Domänen kompatibel sein (und zwar jeweils in der gleichen Reihenfolge). Für die Verknüpfung von Tupelmengen stehen die üblichen Operatoren zur Verfügung, wie man sie von der Mengenlehre her kennt:

- `UNION` für die Vereinigung von Tupelmengen
- `INTERSECT` für den Durchschnitt von Tupelmengen
- `EXCEPT` für die Differenz von Tupelmengen

Standardmässig werden bei einer Anfrage mit diesen Mengenoperationen Duplikate nicht ausgegeben. Wenn die Duplikate ausgegeben werden sollen, muss hinter `UNION` , `INTERSECT` oder `EXCEPT` das Schlüsselwort `ALL` angegeben werden.

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 71: Mengenoperationen Anfrage

Durch diese Anfrage werden die Vor- und Nachnamen aller Kunden bestimmt, die sowohl ein Abonnement besitzen als auch schon einmal ein Inserat aufgegeben haben.

4.3.9 Zusammenfassung

In dieser Unit wurde in einzelen Schritten gezeigt wie SQL-Anfragen formuliert werden. Um all diese Teile im Gesamtkontext aufzuzeigen ist nachstehend die Syntax für eine gesamte SQL-Anfrage aufgeführt.

Komplette Syntax einer SQL Anfrage:

```
SELECT [DISTINCT | ALL]
<Attribut> [AS <Name>] [, ...] | *
FROM <Relation> [, <Relation>]
[WHERE <Bedingungen>]
[GROUP BY <Spalten> [HAVING <Bedingung>]
[ORDER BY <Spalte> [ASC | DESC], [, ...]];
```

Ein grosser Teil dieser Syntax ist optional und muss deshalb nur in gewissen Fällen angegeben werden. Die einfachste syntaktisch korrekte Anfrage, besitzt nur einen SELECT- und einen FROM-Teil. Nur bei komplexeren Fragestellungen werden alle Teile in der gleichen Anfrage verwendet.

Es kann vorkommen, dass im Resultat einer Anfrage gewisse Datensätze mehrmals vorkommen. Das Schlüsselwörter `DISTINCT` bewirkt, dass solche Duplikate aus dem Resultat gelöscht werden. Das Schlüsselwort `ALL` bewirkt, dass die Duplikate nicht gelöscht werden. Da dies standardmässig der Fall ist, muss es nicht unbedingt angegeben werden.

4.3.10 Übung „Datenbankanfragen“

INSTALLATIONSHINWEISE (TUTOREN): **Discussion topics 'Datenbankanfragen', 'eLSQL' und 'SQL Probleme' einrichten**

Diese Übung soll Ihnen die Möglichkeit geben, die verschiedenen SQL Anfragen noch besser kennen zu lernen und zu vertiefen. Dazu wird folgende Beispieldatenbank (www.gitta.info/RelQueryLang/de/multimedia/Datenbasis.html) verwendet. Behalten Sie diese Übersicht während der Bearbeitung der Übung offen.

Dokumentieren Sie die Lösungen für sich in einem Word- oder PDF-Dokument. Wenn Sie möchten können Sie das Dokument am Schluss auf dem Diskussionsforum unter dem Thema „Datenbankanfragen“veröffentlichen.

Wenn Sie Fragen oder Probleme haben mit dieser Übung können Sie auf dem Diskussionsforum unter dem Thema „SQL Probleme“eine Nachricht hinterlegen. Gerne dürfen Sie auch eventuelle Fragen Ihrer Mitstudierenden beantworten.

Aufgaben

Finden Sie die korrekten Lösungstabellen für die nachstehenden SQL Anfragen.

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 72: sql_I.gif

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 73: sql_II.gif

Lösungen zu den Aufgaben

Lösungsdownload (www.gitta.info/RelQueryLang/de/download/Loesung_SQLAnfrage.pdf)Grösse: 183KB. Typ: pdf.

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 74: sql_III.gif

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 75: sql_IV.gif

eLSQL Übung 'Datenbankanfragen'

Um diese Übung durchführen zu können, müssen Sie entweder das Tool eLSQL (Download Tool (www.gitta.info/RelQueryLang/de/download/elsql-0.9.4.zip)Grösse: 1281KB. Typ: zip. , Download Dokumentation (www.gitta.info/RelQueryLang/de/download/elsql-0.9.4-doc.zip)Grösse: 106KB. Typ: zip.) auf einem Server installieren und den Studierenden die entsprechende Webadresse bekanntgeben.

Diese Übung befasst sich mit allen bis jetzt kennengelernten SQL Anfragen. Dazu wird die folgende Beispieldatenbank (www.gitta.info/RelQueryLang/de/multimedia/Datenbasis.html) verwendet. Es ist empfohlen diese Übersicht während der Bearbeitung der Übung offen zu behalten. Die Übersicht finden Sie auch im eLSQL-Tool unter dem Tab 'Datensatz'. Die Übung wird mittels eines Web-Interfaces auf einer MySQL-Datenbank durchgeführt. Die nötigen Informationen dazu (Webadresse und Registrierung) erhalten Sie von ihrem Tutor.

Hinweise zum eLSQL-Interface

- Der Strichpunkt am Ende eines SQL Befehls oder einer SQL Anfrage ist optional.

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 76: sql_V.gif

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 77: sql_VI.gif

- Ihre Änderungen an der Datenbank werden gespeichert und sind beim nächsten Login wieder verfügbar.
- Unter dem Tab 'Datensatz' finden Sie Informationen zum Inhalt der Originaldatenbank.
- Unter dem Tab 'Einstellungen' können Sie Ihre Benutzerinformation ändern, sowie die Datenbank in den Originalzustand zurücksetzen. Alle Ihre Änderungen an der Datenbank werden gelöscht. Achtung: Dieser Befehl kann nicht rückgängig gemacht werden.
- Unter dem Tab 'Protokoll' finden Sie eine Auflistung aller von Ihnen gemachten SQL-Anfragen.

Ziel dieser Übung ist, dass Sie selbständig einfache und schwierigere SQL Anfragen formulieren können, um Antworten auf Problemstellung zu bekommen. Dazu sollen Sie die untenstehenden Aufgaben lösen. Die Vorgehensweise, Überlegungen, SQL Anfragen, Lösungen etc. sollen in einem Word oder PDF Dokument zusammengefasst werden (editiertes und kommentiertes Protokoll). Veröffentlichen Sie Ihre Lösung auf dem Diskussionsforum unter dem Thema 'eLSQL' und schauen Sie sich auch Lösungen Ihrer Mitstudierenden an. Wenn Sie Fragen oder Probleme haben mit dieser Übung können Sie auf dem Diskussionsforum unter dem Thema 'SQL Probleme' eine Nachricht hinterlegen. Gerne dürfen Sie auch eventuelle Fragen Ihrer Mitstudierenden beantworten.

- Selektieren Sie alle Nachnamen der Angestellten und sortieren Sie diese absteigend (Z-A).
- Finden Sie die Vornamen aller Angestellten, die in Dübendorf wohnen und mehr als 30'000 Franken verdienen.
- Selektieren Sie alle Kinder der in Zürich wohnhaften Angestellten und sortieren Sie diese aufsteigend nach dem Geburtsdatum.
- Finden Sie die gesamte Arbeitszeit an allen Projekten der Forschungsabteilung heraus. (in einer Abfrage!)

- Finden Sie heraus wie viele Kinder der Leiter der Abteilung Verwaltung hat.
- Welche Projekte sind in Zürich beheimatet an denen schon mindestens 50 Stunden gearbeitet wurde?
- Finden Sie selber eine einfache oder schwierigere SQL-Abfrage (in Textform als Beschreibung) und veröffentlichen Sie diese auf dem Diskussionsforum unter dem Thema „eLSQL“. Allfällige Lösungen zu solchen Aufgaben können im selben Diskussionsthema veröffentlicht werden. Verwenden Sie für die beiden Nachrichten den gleichen Titel jeweils mit der Endung „- Abfrage“oder „- Lösung“.
- Lösen Sie einige Aufgaben Ihrer Mitstudierenden.

Mögliche Lösungen

Lösungsdownload (www.gitta.info/RelQueryLang/de/download/Loesung_eLSQL1.pdf)Grösse: 179KB. Typ: pdf.

4.4 Einfügen, Löschen und Ändern

Um eine Datenbank immer auf dem aktuellen Stand zu halten, müssen von Zeit zu Zeit die Daten nachgeführt werden (alte, nicht mehr gebrauchte Einträge müssen gelöscht, neue Einträge erfasst oder Einträge müssen geändert werden). In SQL gibt es drei Kommandos, um die Datenbasis zu manipulieren. Diese Kommandos sind Einfügen (INSERT), Löschen (DELETE) und Ändern (UPDATE).

4.4.1 Einfügen von Tupeln

In seiner einfachsten Form wird durch das Kommando „Einfügen“(INSERT INTO) ein Tupel in eine Relation hinzugefügt.

Form des Statements:

```
INSERT INTO <Relationenname> (<Liste von Attributen>)
VALUES (<Liste von Werten>);
```

Wobei die Liste der Attribute nur dann angegeben werden muss, wenn keine vollständigen Tupel eingegeben werden oder wenn die Werte in einer anderen Reihenfolge eingegeben werden als die zugehörigen Attribute bei der Definition der Relation.

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 78: Einfügen von Tupeln

Mit der oben gezeigten Form des INSERT Kommandos werden neue Tupel in die Datenbank eingefügt. Es kommt auch häufig vor, dass die Werte die eingefügt werden müssen, als Resultatrelation einer Anfrage vorhanden sind. Dafür kann das Kommando folgendermassen geänderte werden.

```
INSERT INTO <Relationenname> (<Liste von Attributen>)
SELECT ...  (normale Anfrage)
```

VALUES wird also durch eine SQL-Anfrage ersetzt. Das Resultat dieser Anfrage wird in die angegebene Tabelle eingefügt. Falls die Anfrage eine komplettes Tupel mit der richtigen Reihenfolge liefert, muss auch hier die Liste der Attribute nicht angegeben werden.

4.4.2 Löschen von Tupeln

Das Kommando „Löschen“(DELETE FROM) entfernt ein oder mehrere Tupel aus einer Relation. Die WHERE-Klausel spezifiziert, welche Tupel betroffen sind. Tupel werden explizit immer nur aus einer Relation gleichzeitig gelöscht. Bei einer fehlenden WHERE-Klausel werden alle Tupel aus der angegebenen Relation gelöscht.

Form des Statements:

```
DELETE FROM <Relationenname>
WHERE <Bedingung>;
```

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 79: Loeschen von Tupeln

Achtung: Wird `DELETE FROM` ohne `WHERE` verwendet, werden alle Datensätze der Relation gelöscht. Dies kann in den meisten Systemen nicht rückgängig gemacht werden.

4.4.3 Ändern von Tupeln

Das Kommando „Ändern“(UPDATE) wird gebraucht, um Werte von Attributen von einem oder mehreren Tupeln zu ändern. Die WHERE-Klausel spezifiziert, welche Tupel von der Änderung betroffen sind. Die SET-Klausel spezifiziert die Attribute, die geändert werden sollen und bestimmt deren neue Werte.

Form des Statements:

`UPDATE <Relationenname>`

`SET` einem oder mehreren Attributen einen Wert zuweisen

`WHERE <Bedingung>;`

Hinweis:
Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar.

Abbildung 80: Aendern von Tupeln

Die Zuweisung des neuen Wertes erfolgt wie im Beispiel ersichtlich folgendermassen:

`<Attributname> = <Wert>`

Mehrer Zuweisungen werden durch Komma getrennt. Der zugewiesen Wert kann entweder ein Konstante sein oder das Resultat einer Anfrage, d.h. es kann auch eine Anfrage rechts vom Gleichheitszeichen stehen.

Wird eine ganze Zeile geändert sieht die Zuweisung wie folgt aus:

`ROW = ROW(<Liste von Werten>)`

4.4.4 eLSQL Übung 'Insert, Delete und Update'

INSTALLATIONSHINWEISE (TUTOREN): **Discussion topics 'Insert/Delete/Update' und 'SQL Probleme' einrichten**

Um diese Übung durchführen zu können, müssen Sie entweder das Tool eLSQL (Download Tool (www.gitta.info/RelQueryLang/de/download/elsql-0.9.4.zip)Grösse: 1281KB. Typ: zip. , Download Dokumentation (www.gitta.info/RelQueryLang/de/download/elsql-0.9.4-doc.zip)Grösse: 106KB. Typ: zip.) auf einem Server installieren und den Studierenden die entsprechende Webadresse bekanntgeben.

Diese Übung befasst sich vorwiegend mit den SQL Befehlen INSERT , DELETE und UPDATE . Um die Übung lösen zu können, müssen aber auch die SQL Anfragen aus der Unit Datenbankanfragen bekannt sein. Während der folgenden Übung arbeiten Sie mit dieser Beispieldatenbank (www.gitta.info/RelQueryLang/de/multimedia/Datenbasis.html) Es ist empfohlen diese Übersicht während der Bearbeitung der Übung offen zu behalten. Die Übersicht finden Sie auch im eLSQL-Tool unter dem Tab 'Datensatz'. Die Übung wird mittels eines Web-Interfaces auf einer MySQL-Datenbank durchgeführt. Die nötigen Informationen dazu (Webadresse und Registrierung) erhalten Sie von ihrem Tutor.

Hinweise zum eLSQL-Interface

- Der Strichpunkt am Ende eines SQL Befehls oder einer SQL Anfrage ist optional.
- Ihre Änderungen an der Datenbank werden gespeichert und sind beim nächsten Login wieder verfügbar.
- Unter dem Tab 'Datensatz' finden Sie Informationen zum Inhalt der Originaldatenbank.
- Unter dem Tab 'Einstellungen' können Sie Ihre Benutzerinformation ändern, sowie die Datenbank in den Originalzustand zurücksetzen. Alle Ihre Änderungen an der Datenbank werden gelöscht. Achtung: Dieser Befehl kann nicht rückgängig gemacht werden.
- Unter dem Tab 'Protokoll' finden Sie eine Auflistung aller von Ihnen gemachten SQL-Anfragen.

Aufgaben

- Stellen Sie sich vor, Sie seien neu angestellt bei der Firma welche durch die Beispieldatenbank repräsentiert wird. Fügen Sie Ihren Eintrag in die Tabelle 'Angestellter' ein und fügen Sie mindestens einen Ihrer Angehöriger in die Tabelle 'Angehoeriger' ein. Danach schreiben Sie in Ihrem Namen eine Anzahl von Projektstunden auf eines der Projekte auf.

  Hinweis: Lassen Sie die Geburtsdaten weg, da diese das Format Datum verwenden und kompliziert zu definieren sind.

- Selektieren Sie die von Ihnen gemachten Einträge in den entsprechenden Tabellen.

- Die Arbeit an Projekt 20 wurde eingestellt. Löschen Sie alle Hinweise auf dieses Projekt aus der Datenbank.

  Wie sind Sie vorgegangen und weshalb?

- Beate Tell verlässt die Firma. Ihre Leitungsposition in der Abteilung Forschung wird nun von Sonja Maradona übernommen. Ändern Sie die Datenbank und löschen Sie nicht mehr benötigte Einträge, so dass alle Tabellen auf dem aktuellsten Stand sind.

  Wie sind Sie vorgegangen und weshalb?

- Erhöhen Sie das Salär von allen Angestellten, die Boris Frisch zum Vorgesetzten haben, um 10'000 Fr.

- Stellen Sie sich eine weitere mögliche Aufgabe und führen Sie die entsprechende Änderungen in der Datenbank durch.

Mögliche Lösungen

Lösungsdownload (www.gitta.info/RelQueryLang/de/download/Loesung_eLSQL2.pdf)Grösse: 265KB. Typ: pdf.

4.5 Lernkontrolle

Die nachfolgende Interaktion kann das bisher gelernte an einem konkreten Beispiel angewendet werden.

Um die Flash-Animation zu starten, klicken Sie bitte auf die Grafik.

Abbildung 81: Anwendung Datenbankanfragen

Die Daten der Übung können als Access-Datenbank (www.gitta.info/RelQueryLang/de/multimedia/SQL-SelfAss2.mdb)Grösse: 180 KB. Typ: mdb. heruntergeladen werden. Damit können alle in der Übung vorhanden Anfragen ausprobiert werden, um deren Unterschiede festzustellen.

Datenbank mit bereits erfassten Anfragen (www.gitta.info/RelQueryLang/de/multimedia/SQL-SelfAss2Tut.mdb)Grösse: 220 KB. Typ: mdb.

Alle SELECT-Anfragen der Übung sind in in dieser Access-Datenbank (www.gitta.info/RelQueryLang/de/multimedia/SQL-SelfAss2Tut.mdb)Grösse: 220 KB. Typ: mdb. bereits erfasst. Die UPDATE-Befehle sind nicht vorhanden, da sonst die Daten bei jeder Ausführung der Befehle neu eingegeben werden müssten

4.6 Zusammenfassung

SQL ist eine standardisierte Sprache zur Kommunikation mit relationalen und objektrelationalen Datenbanken. Mittels SQL können Datenbankschemata erzeugt und abgeändert werden. Tabellen und Abhängigkeiten können beispielsweise mit dem `CREATE TABLE` -Befehl erzeugt und mit dem `DROP TABLE` -Befehl gelöscht werden. Dieser Teil der SQL-Sprache wird auch als Datendefinitionssprache (DDL) bezeichnet. Im täglichen Umgang mit Datenbanken viel wichtiger sind die SQL-Befehle zur Datenabfrage und -manipulation (DML). Damit können Anfragen an die Datenbank gestellt und Tupel in Tabellen mutiert werden (Befehle `INSERT, DELETE` und `UPDATE`). Ausserdem können mittels SQL Zugriffsrechte auf die Datenbank vergeben werden.

4.7 Literaturempfehlungen

- ELMASRI, R.; NAVATHE, S.B., 1994 *Fundamentals of Database Systems* [**?**, Einführung ins Thema Datenbanken und SQL, auf Englisch]
- GULUTZAN, P.; PELZER, T., 1999 *SQL-99 Complete, Really* [**?**, An Example-Based Refernce Manual of the New Standard]