Interreg

EUROPÄISCHE UNION

Österreich-Tschechische Republik

Europäischer Fonds für regionale Entwicklung

INFORMATIK

Datenbanken

UNIVERSITY OF APPLIED SCIENCES UPPER AUSTRIA

VŠTE

EUROPÄISCHE UNION

Inhalt

1. GRUNDKONZEPTE VON DATENBANKEN

Eine Datenbank ist eine bestimmte Menge an Informationen (Daten), meistens eine Datensatztabelle, welche auf einem Speichermedium gespeichert ist. Im weiteren Sinne enthält eine Datenbank Software-Ressourcen, welche den Zugriff bzw. die Manipulation der gespeicherten Daten erlauben.

Eine **Datenbank** ist eine Menge von Datensätzen, welche zu einem speziellen Zweck gesammelt wird. Datenbanken werden zumeist für die Speicherung umfangreicher Informationen verwendet. Datenbank-Systeme sind als Teil von Office-Paketen (zB MS Access, OpenOffice.org Basis) verfügbar. Diese Systeme sind auch als eigenständige Programme verfügbar, welche dazu eingesetzt werden, große Radio-Datenbanken zu erstellen. Beispiele hierfür sind MySQL, Oracle und andere.

Eine **Datenbank** ist eine Menge an Daten (Informationen über reale Objekte), welche auf irgendeine Art verbunden sind. Daten sind ein Ausdruck für Daten, welche zur Beschreibung eines Phänomens oder einer Eigenschaft eines beobachteten Objekts verwendet werden. Sie repräsentieren eine Form der Darstellung realer Objekte (Zeichen, Symbole, Bilder, Fakten, Ereignisse), womit sie den Zustand der Realität zu einem bestimmten Zeitpunkt wiederspiegeln.

Eine **Information** ist eine Nachricht darüber, dass ein bestimmtes Phänomen aufgetreten ist. Sie entsteht dadurch, dass Daten eine Bedeutung zugeschrieben wird und steht in einer Beziehung zum Empfänger. Die Nachricht dient dazu, über Veränderungen in der wahrgenommenen Realität zu informieren. Datenbanken begegnen uns im täglichen Leben sehr oft. Daher soll an dieser Stelle aufzeigt werden, wo Datenbanken mit beträchtlichem Ausmaß vorwiegend zum Einsatz kommen:

- Datenbanken für Terminplanungen
- Staatliche Verwaltungsdatenbank
- Informationssysteme von Banken, Schulen und Büros
- Patientenaktensysteme
- Büchereibibliotheken

1.1. Managementsysteme für Datenbanken

Ein Managementsystem für Datenbanken (Abkürzung DBMS in Anlehnung an die englische Bezeichnung database management system) ist eine Softwareausrüstung, welche sicherstellt, dass mit der Datenbank gearbeitet werden kann. Das heißt, sie erstellt eine

Schnittstelle zwischen Anwendungsprogrammen und den gespeicherten Daten. Manchmal wird dieses Konzept mit dem Konzept eines Datenbanksystems verwechselt. Allerdings handelt es sich bei einem Datenbanksystem um einen Verbund aus DBMS und Datenbank.

Entität

Dazu zählt jedes Objekt (Person, Tier, Gegenstand oder Phänomen) der realen Welt, welches im Datenmodell erfasst ist. Eine Entität muss von anderen Entitäten unterscheidbar sein und unabhängig von diesen existieren können.

Daten

Ein Ausdruck für Daten, welche verwendet werden, um eine Phänomen oder eine Eigenschaft eines beobachteten Objekts zu beschreiben. Daten werden durch Messung oder Beobachtung gewonnen und können in zusammenhängende und zugeschriebene Daten unterteilt werden. Zusammenhängende Daten beziehen sich auf eine konstante Skala, bei zugeschriebenen Daten ist dies nicht der Fall.

Information

Informationen sind Daten, welche neue Erkenntnisse bringen.

Datensätze und Attribute

Tabellenzeilen mit Attributwerten für ein Objekt (eine Entität), welche unterschiedlich sein müssen. Datensätze sind Spalten einer Tabelle, Attribute werden ihrem spezifischen Datentyp und Bereich zugeordnet, welcher der Menge zulässiger Werte eines vorgegebenen Attributs entspricht. Die Zeile wird über den Spalten der Tabelle beschnitten und dient dazu, die Daten selbst abzuspeichern.

Attribute, Felder

Eigenschaften, welche bei Entitäten im Blick behalten werden:

- Erstellen von Tabellenspalten
- Diese können verschiedene Werte erhalten
- Die Felder haben einen bestimmten Datentyp (Zahl, Text, Datum, ...)

Primärschlüssel

Dieser identifiziert einmal den Datensatz (Zeile der Tabelle). Es handelt sich um ein Attribut (Feld), welches einen eindeutigen Wert für jede Entität besitzt, zB eine Geburtsnummer, für gewöhnlich ein Hilfsfeld mit einer ID-Nummer.

Fremdschlüssel

Ein Attribut, welches der Primärschlüssel in einer anderen Tabelle ist.

Index

- Dieser stellt eine Möglichkeit dar, eine Tabelle zu sortieren.
- Die Reihenfolge der Datensätze in der Tabelle verändert sich nicht, während die Datenbank „lebt"; der Index hilft dabei, schnell Daten in der Tabelle zu finden.
- Der Index erstellt eine Hilfsdatei mit einer Tabellensortierung anhand eines spezifischen Felds.
- Einige Indizes können, abhängig von verschiedenen Attributen, zu einer Tabelle sortiert werden.
- Der Primärschlüssel ist immer ein Index.

1.2. Datenbankstruktur

Die gebräuchlichsten Datenbanken sind relationale Datenbanken. In diese werden Daten in kleineren Tabellen gespeichert, um eine minimale Redundanz (Datenredundanz) zu gewährleisten. Die Tabellen werden mithilfe von Sitzungen miteinander verbunden. Sitzungen ermitteln die Beziehungen zwischen Tabellen und legen die Zusammenschaltung einzelner Tabellen fest.

Jede dieser Tabellen sollte Daten enthalten, die sich ausschließlich auf einen Objekttyp beziehen (zB Auftragstabelle, Kunden, Preise, Waren etc.). Um eine gute Datenbank zu erstellen, ist es zunächst notwendig, die korrekte Struktur einzelner Tabellen vorzuschlagen. Diese Tabellen müssen dann durch Sitzungen verknüpft werden. Tabellen bilden die Grundlage der gesamten Datenbankstruktur. Die Grundregeln für die Tabellengestaltung lauten wie folgt:

- Jede Information sollte nur einmal in der Datenbank vorkommen
- Jede Tabelle sollte Informationen über einen Objekttyp enthalten
- Wenn Tabellen erstellt werden, sollte das Ausmaß künftig noch zu ergänzender Daten berücksichtigt werden

1.3. Datenbanktabelle

Eine Tabelle sollte Informationen über einen Objekttyp enthalten. Die Datenbanktabelle ist einer normalen Tabelle recht ähnlich. Zeilen enthalten Datensätze (über ein Objekt) und Spalten geben Datenwörter oder Felder an. Das Feld wird manchmal als eine Überschneidung von einer Zeile und einer Spalte bezeichnet, welches eine einen einzelnen Wert (ein Datenelement) enthält.

Zaměstnanci

ID_zamestn	Jméno	Příjmení	Datum naro	Pohlaví	Telefonní čí	ID_funkce
1	Tomáš	Novák	28.4.1980	muž	723 123 456	F_03
2	Josef	Koblížek	15.3.1976	muž	728 452 123	F_01
3	Petra	Maková	5.2.1985	žena	724 556 115	F_02
4	Václav	Sýkorka	30.6.1970	muž		F_06
5	Denisa	Rosolová	12.12.1956	žena		F_05
6	Michal	Aspik	3.6.1976	muž		F_04
7	Dominik	Kokeš	14.2.1981	muž		F_04
8	Tereza	Železná	25.10.1978	žena		F_02
9	Vladimíra	Mimořádná	17.11.1989	žena		F_02
10	Jakub	Pekelník	5.12.1973	muž		F_05

In der oben abgebildeten Tabelle erstellen MitarbeiterInnen Zeilen mit Datensätze, welche Informationen über individuelle MitarbeiterInnen enthalten. Die Spalten repräsentieren die Felder, wo man stets einen Datentyp (Text, Datum) sieht. Jede Spalte (jedes Datenwort) hat einen Namen, einen ausgewählten Datentyp (zB Text, Zahl, Ja/Nein, Datum und Zeit) und eine bestimmte Größe. Weitere Merkmale (Format, Standard, etc.) können ihr zugeschrieben werden. Mehr Informationen über diese Angelegenheit erfährt man in den Videoleitfäden.

1.4. Primärschlüssel

In den meisten Fällen ist es notwendig, jeden Eintrag in der Tabelle eindeutig zu identifizieren. Zu diesem Zweck wird der sogenannte Primärschlüssel herangezogen. Ein Primärschlüssel ist ein Datenbereich (Array), welcher die Absicht verfolgt, die eindeutige Indentifikation individueller Datensätze in einer Tabelle sicherzustellen. Der Primärschlüssel ist für gewöhnlich ein Einzelfeld (ein sogenannter einfacher Primärschlüssel), aber er kann auch ein aus mehreren Tabellen bestehendes Array (ein sogenannter zusammengesetzter Primärschlüssel) sein.

Es kann in jeder Tabelle nur einen Primärschlüssel geben. Mithilfe des Primärschlüssels wird schneller nach Informationen gesucht. Darüber hinaus werden Beziehungen zu anderen Tabellen hergestellt. Die Werte im Primärschlüsselfeld müssen in jedem Datensatz eindeutig sein. Der Primärschlüssel ist einer der Indizes. Für eine schnellere Rückgewinnung und Sortierung von Datensätzen entsprechend eines bestimmten Tabellenfelds, ist

es ratsam, sich der Feldindexierung (Indizes) zu bedienen. Wenn nach einer anderen Tabellenspalte als dem Primärschlüssel gesucht wird, wird diese zum Index bestimmt. Durch das Festlegen eines Index' wird die Sortierung, Suche und Bearbeitung von Werten in den Tabellen beschleunigt.

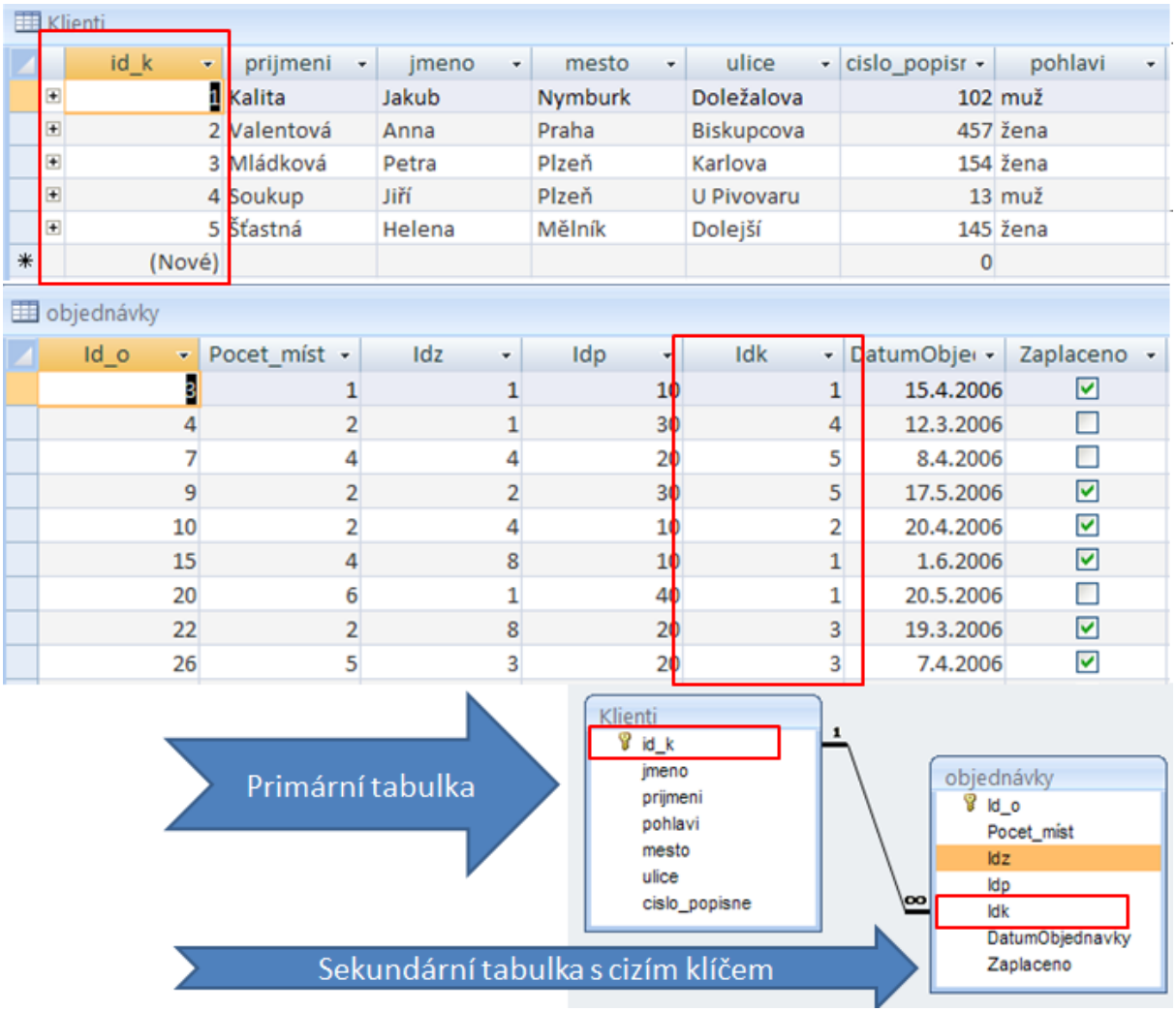
1.5. Beziehung

In einer relationalen Datenbank sind einzelne Tabellen via Sitzungen verknüpft. Der Hauptzweck der Beziehung zwischen Tabellen ist es, das Auftreten redundanter Daten einzuschränken. Daten sollten nicht wiederholt in verschiedenen Tabellen einer einzelnen Datenbank eingespeist werden. Die Beziehung basiert auf der Verbindung zwischen dem einzigartigen Feld (Primärschlüssel) einer Tabelle und dem sich darauf beziehenden Feld einer anderen Tabelle.

Um eine Beziehung zwischen zwei Tabellen herzustellen, bedarf es eines speziellen Fels in jeder Tabelle. In einer dieser Tabellen ist der Primärschlüssel zu finden. Diese Tabelle bezeichnet man gemeinhin als Primärtabelle. In der zweiten Tabelle wird ein spezielles Feld zu Sitzungszweck erstellt. Dieses Feld wird als Fremdschlüssel bezeichnet und enthält die Werte des Primärschlüssels einer anderen Tabelle. Eine Tabelle, welche einen Fremdschlüssel enthält, wird als Sekundärtabelle bezeichnet.

1.6. Primär- und Sekundärtabelle

Das Primärschlüsselfeld enthält ausschließlich eindeutige Werte. Das Fremdschlüsselfeld kann dieselben Werte enthalten. Das Primär- und das Fremdschlüsselfeld müssen dieselben Werte enthalten, um eine korrekte Sitzung zu erstellen. Die nachfolgende Grafik zeigt eine Primärtabelle, welche einzelne Kundendatensätze enthält. Der Primärschlüssel ist das erste Feld, welches die Kundennummer enthält. Diese Tabelle ist mit der Sekundärtabelle verknüpft, welche aus Datensätzen mit Kundenbestellungen besteht. Der Fremdschlüssel in der Sekundärtabelle ist das Feld, welches die Anzahl der Kunden enthält, die bereits eine Bestellung aufgegeben haben. Aus dem oben angeführten Beispiel lässt sich herauslesen, dass ein Datensatz in der Primärtabelle mit einem oder mehreren Datensätzen der Sekundärtabelle übereinstimmt. Anders gesagt: ein Kunde kann eine oder mehrere Bestellungen aufgeben. Dies entspricht dem Sitzungstyp 1: N.



1.7. Arten von Sitzungen

Es werden drei Grundarten von Sitzungen unterschieden:

Beziehungsart 1:1 (außergewöhnlich): Ein Datensatz der Primärtabelle stimmt nur mit einem Datensatz der Sekundärtabelle überein. Ein Beispiel wäre, dass eine Person nur einer Geburtsnummer zugeordnet ist.

Beziehungsart 1:N (gebräuchlichste): Ein Datensatz der Primärtabelle stimmt mit einem Datensatz oder mehreren Datensätzen in der Sekundärtabelle überein. Diese Art wurde bereits im oben angeführten Beispiel mit den Kunden und Bestellungen erklärt.

Beziehungsart M:N (sehr oft): Ein Datensatz oder mehrere Datensätze in der Primärtabelle stimmen mit einem Datensatz oder mehreren Datensätzen in der Sekundärtabelle

überein. Diese Sitzungen werden mithilfe einer Kopplungstabelle hergestellt, welche über zwei Sitzungen des Typs 1: N mit den Originaltabellen verknüpft wird.

1.8. Verweisintegrität

Die Verweisintegrität hält die Beziehungen zwischen den Tabellen aufrecht. Sie erlaubt es nicht, dass Datensätze in die Sekundärtabelle eingefügt werden, für die es keinen übereinstimmenden Datensatz in der Primärtabelle gibt. Sie überwacht auch die Veränderung von Fremdschlüsselwerten, wenn der Primärschlüssel verändert wird. Innerhalb der Verweisintegrität ist es möglich, Regeln für das Löschen von Datensätzen festzulegen.

1.9. Funktion der Datenbank

Professionelle Datenbanken enthalten eine große Menge wichtiger Informationen. Personen, die mit so einer Datenbank arbeiten, können in verschiedene Gruppen unterteilt werden:

- ein Datenbankspezialist plant und erstellt professionelle Datenbanken
- ein Benutzer fügt Daten ein, wartet die Daten und ruft Informationen aus der Datenbank ab
- der Datenbank-Manager gewährt Benutzern Zugang zu bestimmten Daten und ist für die Rettung der Datenbank nach einem Zusammenbruch oder schlimmen Fehler verantwortlich.

2. DATENBANKMODELLE

Ausgehend von der Sichtweise, wie Daten und Verknüpfungen zwischen diesen gespeichert wer-den, wird bei Datenbanken zwischen zwei Grundarten unterschieden:

2.1. Hierarchische Datenmodelle

Daten sind in Form einer Baumstruktur organisiert. Jeder Datensatz entspricht einem Knoten in dieser Baumstruktur, die Beziehung zwischen den Datensätzen ist vom Typ Eltern/Nachkomme. Das Aufspüren von Daten in hierarchischen Datenbanken erfordert es, dass man durch Datensätze hin zum Kindknoten, zurück zum Elternknoten oder auf die Seite hin zum nächsten Sprossknoten navigiert. Der größte Nachteil hierarchischer Layouts ist, dass das Einfügen und Löschen von Datensätzen komplexe Operationen erfordert. Auch die zeitweise unnatürliche Organisation der Daten ist zuweilen ein Nachteil.

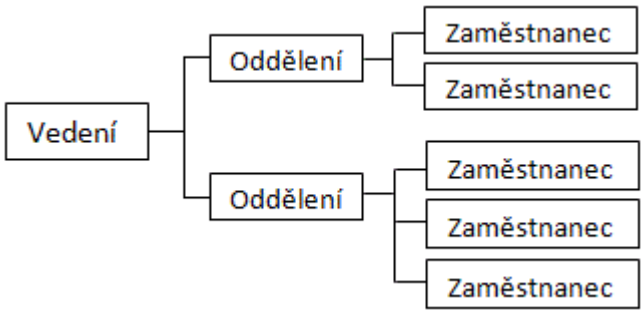


Abbildung: Hierarchisches Modell

2.2. Netzwerkdatenmodelle

Das Netzwerkdatenmodell ist im Wesentlichen eine Verallgemeinerung des hierarchischen Modells, welches dieses um Mehrfachbeziehungen (Sets) ergänzt. Diese Sets verbinden Datensätze desselben oder eines anderen Datentyps miteinander, wobei die Verbindung zu einem oder mehreren Datensätzen hergestellt wird. Der Zugriff auf verknüpfte Datensätze erfolgt unmittelbar ohne vorhergehende Suche; es gibt dafür Operationen: Ausfindig machen des Schlüsseldatensatzes, verschieben des ersten Nachwuchsknotens in das Sub-Set, verschieben des anderen Sprossknotens in das Set, Aufrücken vom Nachwuchs- zum Elternknoten in einem anderen Set. Die Nachteile einer Netzwerk-Datenbank sind deren besondere Starrheit und die Schwierigkeit, deren Struktur zu verändern.

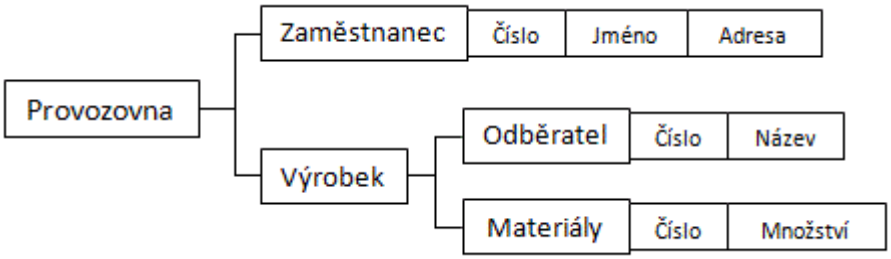


Abbildung: Netzwerkmodell

2.3. Rationales Datenmodell

Das relationale Datenbankmodell gehört zu den jüngsten und meistverwendeten Modellen. Zurzeit wird es am häufigsten vom Ministerium für Wirtschaft und Industrie. Das Model hat eine einfache Struktur, Daten sind in Tabellen organisiert, die aus Zeilen und Spalten bestehen. Alle dieser Datenbankoperationen werden in den Tabellen ausgeführt.

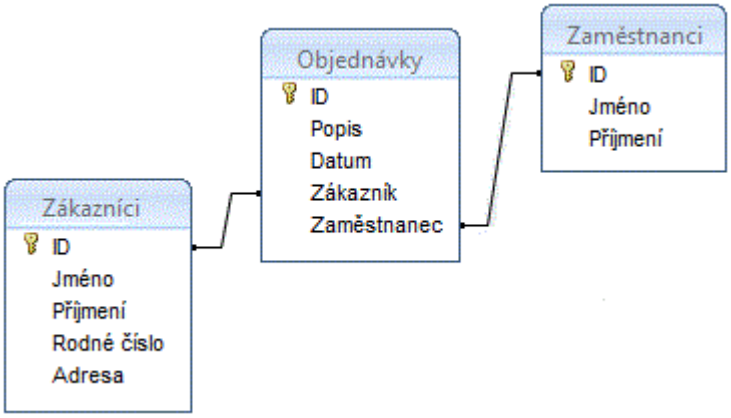


Abbildung: Relationales Modell

Eine relationale Datenbank muss die folgenden zwei Charakteristika besitzen:

- die Datenbank wird vom Benutzer als eine Menge von Sitzungen und nur als diese wahrgenommen
- die Auswahl-, Prognose- und Verbindungsoperationen sind im relationalen ROI verfügbar, ohne einen explizit vordefinierten Zugangspfad zu benötigen, um diese Operationen auszuführen.

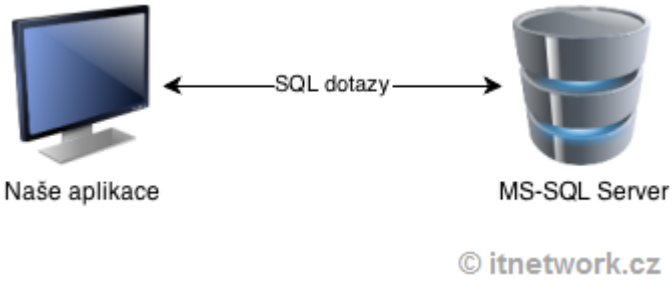
2.4. Objektdatenbanken

Neben relationalen Datenbanken gibt es auch noch die bereits erwähnten Objektdatenbanken. Diese befassen sich mit der Problematik der Objekt- und beziehungsbezogenen Inkompatibilität. Sie stellen denselben Komfort wie ORMs zur Verfügung, müssen jedoch intern keine Daten in Tabellen konvertieren, da diese als Objekte abgespeichert werden. In der Theorie gibt es keinen Leistungs-oder einen anderen Grund, warum sie nicht über kurz oder lang relationale Datenbanken ersetzen sollen. In der Praxis jedoch werden sie so gut wie kaum verwendet und man muss hoffen, dass sich das mit der Zeit ändert.

2.5. Angehängte Anwendung

Man kann die Zugriffsanwendung verwenden, wenn man Daten in Echtzeit lesen oder ändern möchte. Durch die Verwendung von DataReader, Command und Connection-Klassen werden SQL-Statements direkt an SQL geschickt, was unmittelbar zum Erhalt von Ergebnissen führt.

Diese Situation ist auf der nachfolgenden Grafik abgebildet:



2.6. Objektdatenbanken

Managementsysteme für Datenbanken (DBMS) sind derzeit die am weitesten verbreiteten Werkzeuge zur Speicherung und Manipulation von Daten kommerzieller Anwendungen. Mit ihnen lassen sich große Datenbanken relativ einfach managen, darüber hinaus bieten sie ein hohes Maß an Integrität, eine effiziente Suche und Verarbeitung von Informationen. Sie tolerieren Fehler und Ausfälle, ermöglichen einen gleichzeitigen Zugriff mehrerer Benutzer und bieten noch weitere Vorteile.

Das relationale Datenmodell – das gebräuchlichste Modell in den heutigen Datenbanksystemen – schränkt die Struktur und Beziehungen der gespeicherten Daten auf ein kleines Set von Tabellen ein, welches über einem vordefinierten Set an einfachen Datentypen

liegt. Der unbestrittene Vorteil dieses Modells ist dessen Einfachheit und – daraus resultierend – die einfache Standardisierung und Portierbarkeit. Der Nachteil hingegen ist, dass sich das reale Datenschema vom internen Datenbank-Tabellenmodell unterscheidet. Alle Beziehungen zwischen den Daten können nur in Form von Tabellen ausgedrückt werden, was in einer Anwendung mit einem etwas komplexeren Datenmodell zu einer Anzahl an Tabellen führt, die über Hilfsverknüpfungen, sogenannten Keys, miteinander verbunden sind.

Dies resultiert im Verlust von Klarheit, die Datenbank lässt sich schlechter managen und zukünftige Veränderungen im Datenmodell der Anwendung zwingen Programmierer dazu, merklich in ihre Tabellen-Darstellung einzugreifen. Relationale Datenbanken sind und werden auch weiterhin das bevorzugte Mittel sein, um kommerzielle Anwendungen zu managen, die dadurch gekennzeichnet sind, dass sie eine über eine große Menge an Daten mit einfacher Struktur verfügen. Viele Anwendungen wie z.B. Design-, Multimedia-, geografische System usw. brauchen hingegen ein Datenmodell, welches dafür sorgt, dass komplexe reale Daten besser mit deren Darstellung im Datenbanksystem korrespondieren. Besagtes Modell ist das Objektdatenmodell, welches in Datenbankensystemen auf bekannten Grundsätzen des objektorientierten Modellierens und Programmierens basiert. Darüber hinaus ist es um Techniken der Ausdauer, Darstellung von Beziehungen, Befragungen, Transaktionszugriffen usw. angereichert.

2.7. Grundlagen der Objektorientierung – Objekte und Klassen

Ein objektorientiertes Modell basiert auf dem Grundsatz, Informationen aus der realen Welt in Objekte zu zerlegen. Ein Objekt bezeichnet gemeinhin jede (sogar strukturierte) Entität, die in einem bestimmten Kontext der sie umgebenden Welt einzigartig und unabhängig identifizierbar ist. Daher hat das Objekt eine eindeutige Identität und alle zwei oder weiteren ähnlichen, baugleichen Objekte grenzen sich klar voneinander ab. Die Identität eines Objekts wird durch einen Objektidentifizierer (OID) festgestellt, welcher von dem System erstellt wird, einzigartig ist, sich während der Existenz eines Objekts nicht verändert und sowohl für den Programmierer als auch den Endverwender versteckt ist. Objekte werden durch Klassen charakterisiert. Die Klasse ist eine zusammenfassende Beschreibung des Objekts, sie ermittelt die Datenordner des Objekts und die Operationen (auch Methoden genannt), die in diesem Objekt durchgeführt werden können. Jedes Objekt ist die Instanz einer Klasse. Es ist grundsätzlich möglich, eine uneingeschränkte Anzahl von Instanzen strukturell identischer Objekte in einer Klasse zu erstellen. Als nächstes wird die Schnittstelle der Klasse gekennzeichnet.

2.8. Literale

Neben Objekten wird auch das Konzept der Literale in einem objektorientierten Modell vorgestellt. Ein Literal ist eine Datenentität eines besonderen Datentyps, welcher – anders als ein Objekt – keine eigene Identität besitzt. Literale scheinen für gewöhnlich als Objektattribute auf. Die Menge der Operationen, welche im Datentyp eines Literals durchführbar sind, ist festgelegt, sie kann nicht verändert werden. Objekte und Literale stehen in engem Zusammenhang mit dem Konzept der Variabilität (Mutabilität). Variablität wird als die Fähigkeit verstanden, Daten zu verändern und gleichzeitig deren Identität zu erhalten. In diesem Sinne sind Objekte variabel, da es möglich ist, die Werte ihrer Datenkomponenten zu verändern, während sie gleichzeitig ihre ursprüngliche Identität beibehalten. Literale hingegen sind nicht variabel.

2.9. Operation

Es gibt einige Grundarten von Operationen in Objekten Jedes Objekt hat eine oder mehrere Konstruktoren. Der Zweck eines Konstruktors ist es, ein Objekt im Zuge ihrer Erstellung mit einem Feld vorzubelegen. Darüber hinaus ist auch ein Destruktor Teil jedes Objekts. Der Destruktor wird abgerufen, wenn das Objekt unterbrochen wird. Sein Zweck ist es, das Objekt zu reinigen, bevor es entfernt wird. Eine wichtige Operation in Objekten ist das Kopieren. Hierbei werden die sogenannten seichten und tiefen Kopien unterschieden. Im Gegensatz zur seichten Kopie werden bei der tiefen Kopie nicht nur die Attribute eines Objekts kopiert, es werden auch Kopien von Objekten erstellt, auf welche das ursprüngliche Objekt verwies. Andere typische Operationen sind Methoden zum Herausfinden und Zuordnen von Attributwerten, Methoden zur Durchführung von Berechnungen und Manipulation von Objektattributen, Methoden um eine Nutzerausgabe zu produzieren usw.

Ein **Objektdatenmodell** entspricht den in einer Objektstruktur gespeicherten Daten. Es handelt sich für gewöhnlich um die Trennschicht eines Objekts zwischen dem Code und der Datenbank, wo die Daten, mit welchen die Anwendung arbeitet, nicht den Datenbankserver mit Anfragen belasten.

Objektbezogenes Datenmodell

Das objektbezogene Datenmodell ist eine klassische Datenbank, welche durch abstrakte Datentypen (ADT) erweitert wird. ADTs sind nutzerdefinierte Typen, welche aus rudimentären Datentypen einer Datenbank bestehen. Was verletzt 1NF?

- Objektmerkmale werden nun in alle größeren SARBs implementiert.
- Objekttpyen und deren Methoden werden zusammen mit Daten in der Datenbank gespeichert, damit der Programmierer keine ähnlichen Strukturen in jeder Anwendung erstellen muss.

- Der Programmierer kann auf ein Set von Objekten zugreifen, als ob es sich dabei um ein Objekt handeln würde.
- Objekte können einfach Verbindungen bilden, welchen zufolge eine Entität aus anderen Entitäten besteht (ohne, dass man dafür Verbindungen bräuchte).
- Methoden laufen auf dem Server. Es gibt keine ineffiziente Datenübermittlung über das Netzwerk.

Objektdatentypen

können folgendes enthalten:

- Daten – Attribute
- Operationen – Methoden
- ORSRBD-spezifisch ist der Umstand, dass man sich Daten sowohl aus relationaler Sicht (Datensätze, Operationen) als auch aus Objektsicht anschauen kann.

Arten von Methoden:

- Mitgliedermethoden – sie werden in einem bestimmten Objekt aufgerufen.
- Statische Methoden – sie werden in dem Datentyp aufgerufen.
- Konstruktor – ein Standard – Konstruktor wird für jeden Objekttyp festgelegt.

3. INTEGRITÄT VON DATENBANKEN

Die Integrität einer Datenbank bedeutet, dass sich die darin gespeicherten Daten konsequent an vorgegebene Regeln halten. Nur Daten, welche den vordefinierten Kriterien entsprechen, können eingegeben werden (so müssen z.B. der für die Tabellenspalte vergebene Datentyp berücksichtigt oder andere zulässige Einschränkungen berücksichtigt werden). Integrierte Einschränkungen dienen dazu, diese Integrität sicherzustellen. Es handelt sich hierbei um Werkzeuge, welche verhindern, dass falsche Daten eingefügt werden oder bestehende Datensätze verloren gehen oder beschädigt werden, wenn man mit der Datenbank arbeitet. So ist es mitunter möglich sicherzustellen, dass Daten gelöscht werden, welche bereits ihren Sinn verloren haben. Ein Beispiel hierfür wäre es, dass beim Löschen eines Benutzers auch die damit verknüpften Datensatzrestbestände in anderen Datenbanktabellen entfernt werden.

3.1. Arten von Integritätseinschränkungen

Einschränkungen der Entitätsintegrität: Verpflichtende Integritätseinschränkung um sicherzustellen, dass der Primärschlüssel einer Tabelle vollständig ist (verhindert die Speicherung von Daten, welche mit Daten in einer anderen Zeile der Tabelle identisch wären).

Einschränkungen der Domainintegrität: Diese stellen sicher, dass die Datentypen/Datendomains eingehalten werden, welche für die Spalten der Datenbanktabellen definiert wurden.

Einschränkungen der referenziellen Integrität: Diese beschäftigen sich mit den Beziehungen zwischen zwei Tabellen, in welchen ihre Zusammenhänge durch die Verbindung zwischen Primär- und Fremdschlüssel festgelegt wird.

Aktive Hinweisintegrität: Diese definiert die Aktivitäten, welche eine Datenbank auszuführen hat, wenn Regeln verletzt werden.

3.2. Berücksichtigung von Integritätseinschränkungen

Grundsätzlich gibt es drei Wege sicherzustellen, dass Integritätseinschränkungen eingehalten werden

1. Die Stelle auf dem Server der Datenbank, wo einfache Mechanismen zur Wartung von Integritätseinschränkungen gespeichert werden

- Dies ist der beste Weg, Daten zu schützen
- Es dauert allerdings länger, bis der Benutzer eine Antwort vom System erhält und kann nicht immer für ein anderes Datenbanksystem sichergestellt werden

2. Die clientseitige Speicherstelle für Schutzmechanismen

- Ist aufgrund Komforts und der Unabhängigkeit des Datenbanksystems die beste Wahl
- Der Bedarf an Kontrollmechanismen für jede Operation kann Anwendungsfehler verursachen, weshalb mehrere Anwendungen repariert werden müssen.

3. Separate serverseitige Programmmodule

- In modernen Datenbanksystemen werden zu diesem Zweck so genannte Trigger implementiert. Diese entsprechen separaten Verfahren, die automatisch vor und nach Operationen laufen können, welche Daten behandeln
- Dies ermöglicht die Implementation komplexer Integritätseinschränkungen
- Der Nachteil ist hier wieder die stark eingeschränkte Möglichkeit der Migration auf ein anderes Datenbanksystem auf dem Server

Die ideale Lösung ist die Kombination der vorher genannten Möglichkeiten in Abhängigkeit von den besonderen Bedingungen. Die Überprüfung von Integritätseinschränkungen werden für gewöhnlich nach jeder Operation durchgeführt, was die Anforderungen an den Server reduziert. Es ist nicht notwendig zu protokollieren, welche Überprüfungen später durchgeführt werden müssen. Allerdings können komplexere Integritätseinschränkungen nicht immer verifiziert werden, weshalb sich die Konformität erst nach dem Abschluss der gesamten Transaktion überprüfen lässt.

3.3. Beziehungen zwischen Tabellen

Sitzungen werden verwendet, um Daten zu verbinden, die miteinander in Beziehung stehen und sich in verschiedenen Datenbanktabellen befinden. Grundsätzlich unterscheidet man zwischen vier Arten von Beziehungen:

- Es gibt keinen Zusammenhang zwischen Tabellen, weshalb keine Beziehungen definiert werden
- 1:1 (ein Datensatz stimmt nur mit einem Datensatz in einer anderen Datenbanktabelle über-ein und umgekehrt)
- 1:N (ordnet einen Datensatz mehreren Datensätzen einer anderen Tabelle zu)
- Dies ist die gebräuchlichste Art einer Sitzung, da sie sich mit vielen Situationen im echten Leben deckt
- M: N (erlaubt es, mehrere Datensätze einer Tabelle mehreren Datensätzen einer zweiten Tabelle zuzuordnen)
- Aus praktischen Gründen wird diese Beziehung zumeist realisiert, indem zwei Beziehungen der Art 1: N und 1: M kombiniert werden, welche auf eine unterstützende, so genannte Kopplungstabelle referenzieren, welche aus einer Kombination beider verwendeter Schlüssel besteht.

3.4. Normale Formen

Der Begriff Normalisierung bezeichnet das Verfahren der Vereinfachung und Optimierung der vorgeschlagenen Strukturen von Datenbanktabellen. Das Hauptziel ist es, Datenbanktabellen so zu gestalten, dass sie ein Minimum an redundanten Daten enthalten. Die Korrektheit der Struktur kann mithilfe der folgenden normalen Formen evaluiert werden.

Nullte Normale Form (0NF)

- Die Tabelle enthält mindestens eine Spalte (ein Attribut) welches mehrere Arten von Werten enthalten kann.

Erste Normale Form (1NF)

- Alle Tabellenspalten können nicht mehr weiter in Teile aufgeteilt werden, die Informationen enthalten – Elemente gleichen Atomen
- Eine Spalte enthält keine zusammengesetzten Werte.

Zweite Normale Form (2NF)

- Die Tabelle enthält nur Spalten, welche vom gesamten Schlüssel abhängen

Dritte Normale Form (3NF)

- Die Tabelle befindet sich in der dritten normalen Form, wenn es keine Abhängigkeiten zwischen Spalten gibt

Vierte Normale Form (4NF)

- Die Spalten in der Tabelle beschreiben nur einen Tatbestand oder einen Kontext

Fünfte Normale Form (5NF)

- Durch die Ergänzung einer neuen Spalte würde sich die Tabelle in mehrere Tabellen aufteilen

3.5. Datenbankintegrität

Datenbankintegrität bedeutet, dass sich die Datenbank an spezifizierte Regeln hält, konkret an Integritätseinschränkungen. Diese Integritätseinschränkungen sind Teil der Datenbankdefinition und das Managementsystem für Datenbanken ist verantwortlich für deren Einhaltung.

Feste Einschränkungen können sich entweder auf einzelne Werte beziehen, welche in die Felder einer Datenbank eingegeben werden (zB eine Note für ein Lehrfach muss zwischen 1 und 5 liegen) oder eine Bedingung für die Kombination von Werten in einigen Feldern eines Datensatzes darstellen (zB darf das Geburtsdatum nicht später sein als das Todesdatum). Eine ganzzahlige Einschränkung kann auch für eine ganze Menge an Datensätzen einer bestimmten Art gelten. So kann es zB sein, dass ein bestimmtes Feld oder eine Kombination von Feldern innerhalb der gesamten Datensatzmenge in einer Datenbank, welche einer bestimmen Art sind, einen eindeutigen Wert haben muss (ein Beispiel hierfür wäre die ID-Nummer in einem Personendatensatz).

Häufig verwendete Integritätseinschränkungen in relationalen Datenbanken beziehen sich auf die referenzielle Integrität. Es handelt sich hierbei um eine Anforderung, der zufolge ein Datensatzfeld einen Verweis auf einen anderen Datensatz irgendwo in der Datenbank enthalten muss und der referenzierte Datensatz tatsächlich existiert, sodass diese Verknüpfung nicht ins Leere führt und keine sogenannte Datenbankwaise darstellt.

Feste Einschränkungen

Es sollte sichergestellt werden, dass nur Datensätze, die Beziehungen der realen Welt entsprechen, in die Datenbank aufgenommen werden (so darf ein Altersattribut keine negativen Zahlen erhalten). Um mit solchen Fällen richtig umzugehen, wird eine Integritätseinschränkung verwendet, mit deren Hilfe eine Bandbreite von Werten festgelegt wird, die ein bestimmtes Attribut annehmen kann. Feste Einschränkungen spezifizieren, welche Einschränkungen und interne Beziehungen Datenbank-Daten einhalten müssen. Sitzungen, die den Integritätseinschränkungen entsprechen, sind erlaubt.

Relationales Datenbanksystem

- muss ein effizientes Management und effiziente Analyse gespeicherter Daten ermöglichen
- muss die folgenden drei Grundfunktionen erfüllen:
    - das Definieren von Datentypen erlauben
    - die Arbeit mit Daten ermöglichen
    - das System sollte die Mittel enthalten, um erforderliche Arbeitsgänge mit Daten auszuführen, zB Daten sortieren, filtern, Berechnungen mit Daten anstellen, Datenmanagement
- im Besonderen überprüft das System den Zugang zu den Daten, sprich wer autorisiert ist, auf die Daten zuzugreifen und wer sie aktualisieren darf
    - das System steuert die Datenteilung unter mehreren Benutzern

4. RELATIONALES DATENBANKMODELL

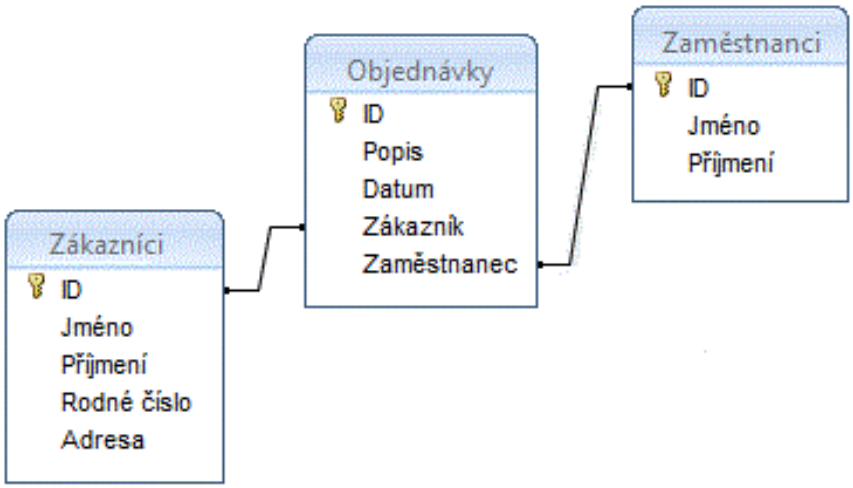


Abbildung: Relationales Modell

Eine relationale Datenbank muss die folgenden zwei Merkmale erfüllen:

- Die Datenbank wird vom Benutzer ausschließlich als eine Menge von Sitzungen wahrgenommen
- Zumindest Operationen zur Auswahl, Hochrechnung und Verknüpfung sind im relationalen ROI verfügbar, ohne dass sie explizit vordefinierte Zugangspfade benötigen, um diese Operationen auszuführen.

4.1. Zwölf Regeln gelten für das relationale Datenbankmodell

1. Informationsregel:

Jede Information in einer relationalen Datenbank wird explizit auf logischer Ebene und nur eine Weise ausgedrückt: in Form von Werten in der Tabelle.

2. Sicherheitsregel:

Auf alle Daten in der relationalen Datenbank kann man garantiert zugreifen, wenn man die Tabellennamen mit den Werten des Primärschlüssels und dem Namen der Spalte kombiniert.

3. Systematische Verarbeitung von Null-Werten:

Null-Werte werden von einem Managementsystem für Datenbanken vollständig unterstützt, um undefinierte Informationen abzubilden, unabhängig vom Datentyp.

4. Dynamischer on-line Katalog basierend auf dem relationalen Modell:

Die Beschreibung der Datenbank erfolgt auf logischer Ebene auf dieselbe Weise wie jene von Kundendaten, weshalb ein autorisierter Benutzer dieselbe relationale Sprache für seine Anfrage verwenden kann, wenn er mit den Daten arbeitet.

5. Regel zur Konfiguration von Interpretationen:

Alle Interpretationen, die theoretisch möglich sind, können auch im System konfiguriert werden.

6. Ein umfassendes Daten Sub-Wörterbuch:

Ein relationales System kann mehrere Sprachen und verschiedene Modi für Begriffsoperationen unterstützen. Allerdings muss es zumindest eine Befehlssprache mit einer ausgeklügelten Syntax geben, welche allgemeine Datendefinitionen, Interpretationsdefinitionen, Interaktivität und Programmmanipulationen, Integritätseinschränkungen, autorisierten Datenbankzugriff, Trans-aktionsbefehle usw. zulässt.

7. Fähigkeit zum Einfügen, Erstellen und Löschen von Daten:

Die Fähigkeit, relationale Regeln von sowohl standardmäßigen als auch abgeleiteten Sitzungen zu warten, bleibt nicht nur bei der Sichtung von Daten aufrecht, sondern auch bei Operationen zum Penetrieren, Ergänzen und Löschen.

8. Unabhängigkeit physischer Daten:

Anwendungsprogramme sind unabhängig von der physischen Datenstruktur.

9. Unabhängigkeit logischer Daten:

Anwendungsprogramme sind unabhängig von Veränderungen in der logischen Struktur einer Datenbankdatei.

10. Integrale Unabhängigkeit:

Feste Einschränkungen müssen von relationalen Datenbankressourcen oder deren Sprache definiert werden. Es muss möglich sein, diese im Katalog anstatt im Anwendungsprogramm zu speichern.

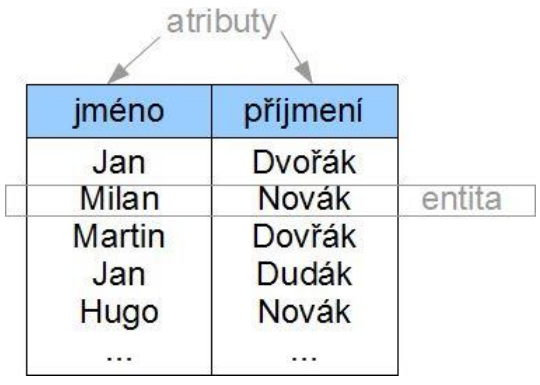
11. Unabhängigkeit der Distribution:

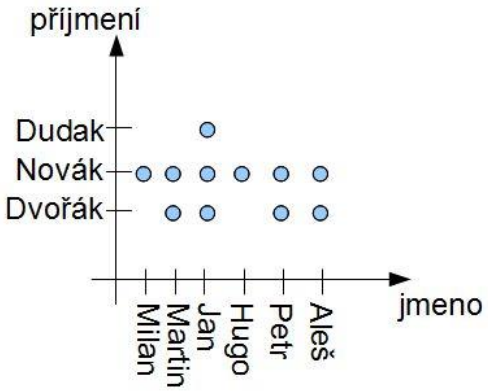
Relationale RBDs müssen in der Lage sein, auf anderen Rechnerarchitekturen aufzusetzen.

12. Datenbankzugriffsregel:

Wenn das relationale System ein niedriges Sprachniveau hat, kann dieses Niveau nicht verwendet werden, um Integritätseinschränkungen zu erstellen. In diesem Fall muss sich das System in Form einer höherstufigen relationalen Sprache ausdrücken.

4.2. Relationales Datenmodell





Ein relationales Datenmodell ist eine Möglichkeit, Daten in Tabellen zu behalten. Der Begriff relational wurde gewählt, weil die Tabelle in Form einer Sitzung definiert ist.

Die Beziehung ist im Wesentlichen eine Tabelle. Sie ist als Untermenge eines kartesianischen Produkts einer Domain definiert. Die Beziehung auf der rechten Grafik ist daher eine Untermenge des Sets von {Dudák, Novák, Dvořák} x {Milan, Martin, Jan, ..., Aleš}

Anders als bei einer mathematischen Sitzung verändert sich die Datenbank mit der Zeit (durch das Hinzufügen und Entfernen von Sitzungselementen). Neben standardmäßigen Set-Funktionen stößt man hier auf eine Auswahl-Funktion zur Zeilenauswahl und Projektion sowie zur Spaltenauswahl bei Datenbanksitzungen.

Eine **Domain** ist die Menge all jener Werte, die ein Attribut annehmen kann. Anders gesagt, die Bandbreite von Attributwerten. In der Praxis wird die Integrität einer Domain eingeschränkt (IO).

Die **Bildannahmeattributsdomain** ist die Menge aus {Dudak, Novák, Dvořák}. * Note

Ein **Attribut** ist die Eigenschaft einer Entität. Aus Sicht einer Tabelle handelt es sich um eine Spalte.

Das **relationale Schema** kann als Struktur einer Tabelle (Attribute und Domains) verstanden werden.

<u>Beispiel:</u>

Tabelle (Sitzung): ein Lehrer
Attribute: ID, Vorname, Zuname, Funktion, Büro

Domains:

- D1 – Drei Buchstaben vom Familiennamen, Anzeige von drei Zeichen. Zahlen.
- D2 – Bezeichnung des Kalenders
- D3 – Set von Familiennamen
- D4 – Set von Funktionen (Assistent, Wissenschaftler, Lehrer ...)
- D5 - A101, A102, ... A160

Relationales Schema: Lehrer (ID, Name, Adresse, Funktion, Büro)

Sitzung: Lehrer = {(nov001, lukas, novak, scientist, A135), (com123, jan, comenius, teacher, A111)

Sie ist definiert als R (A, f), wobei A einer Menge von Attributen entspricht (A1, A2, ... An) und die function f (Ai) = Di das Domainattribut zuordnet.

4.3. Relative Datenmodelleigenschaften

Nachfolgend werden die Tabelleneigenschaften einer Sitzungsdefinition aufgelistet:

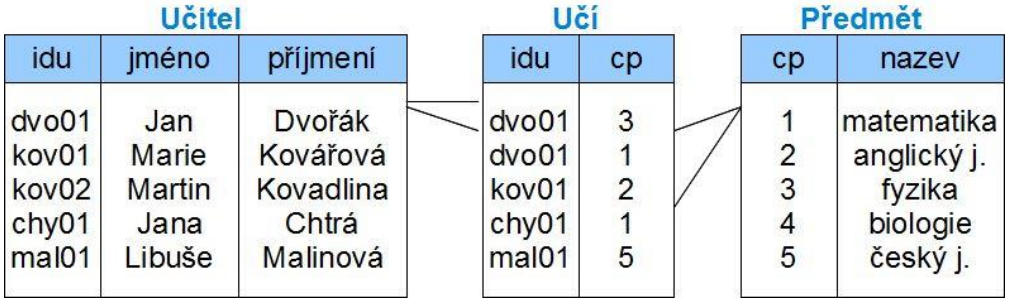
- Spaltenhomogenität (Domainelemente)
- Jeder Wert (der Wert eines Attributs in der Spalte) ist ein atomarer Eintrag
- Die Reihenfolge der Zeilen und Spalten spielt keine Rolle (es handelt sich um festgelegte Elemente/Attribute)
- Jede Zeile einer Tabelle ist eindeutig identifizierbar durch den Wert eines oder mehrerer At-tribute (Primärschlüssel)

4.4. Beziehungen im Relationalen Modell

Gemeinhin werden Verbindungen im relationalen Modell mithilfe anderer Sitzungen umgesetzt. Hierbei handelt es sich um eine sogenannte Kopplungstabelle. Diese enthält jene Sitzungsattribute (inklusive der Verbindungen), welche eindeutig deren Entitäten identifizieren – die Primärschlüssel. Wenn eine Tabelle ein Attribut enthält, welches in einer anderen Tabelle als Primärschlüssel dient, so enthält sie einen Fremdschlüssel. Die Kopplungstabelle enthält deshalb Fremdschlüssel.

In der Grafik unten sieht man die Sitzungen „Lehrer" mit Primärem ID-Schlüssel und die Sitzung mit Primärem Schlüssel cp. Damit wird die Beziehung Lehrer UČÍ ausgedrückt. Das Lehrfach wurde als die neue Sitzung „Lernen" erstellt, welche zwei Fremdschlüssel enthält (idu bezieht sich auf den Lehrer und cp bezieht sich auf die Entität des Lehrfachs). Nun ist es möglich, den Lehrer über die Kopplungs-tabelle mit jenem Lehrfach zu verbinden, das er unterrichtet.

Warum dem Lehrer kein Lehrfachattribut in der Tabelle zugewiesen wurde? Ganz einfach deshalb, weil ein Lehrer mehr als ein Lehrfach unterrichten kann. Zwischen dem Lehrer und dem Lehrfach besteht eine Beziehung M: N. Selbst wenn man „Lehrer" der Lehrfach-Attributstabelle hinzufügt, würde das nicht der Beziehung entsprechen (ein Lehrfach kann von mehreren Lehrern unterrichtet werden). Bei einer 1: N-Beziehung ist das möglich, und diese kommt auch in der Praxis vor. Die Kopplungstabelle ist nur notwendig, um die M: N-Beziehungen umzusetzen.



5. SQL GRUNDLAGEN

5.1. Einführung in SQL

SQL – Structured Query Language – ist ein allgemeines Instrument zur Manipulation, Verwaltung und Organisation von Daten, welche in einer Computerdatenbank gespeichert sind. Sie ist vor allem für Benutzer gedacht, werden aber auch auf verschiedene Arten von Anwendungsentwicklern verwendet. Sie kann an jede Umgebung angepasst werden.

Der Name dieses Werkzeugs ist kurz, aber nicht unkompliziert. SQL ist nicht nur eine Abfragesprache, sie kann auch Daten definieren, sprich eine Tabellenstruktur, Spalten einer Datentabelle (Array) mit Daten fühlen und Beziehungen zwischen Datenobjekten sowie deren Organisation definieren. Sie ermöglicht darüber hinaus eine Datenzugriffskontrolle, sprich das Gewähren und Entziehen von Zugriffsprivilegien auf verschiedenen Ebenen. Dadurch können Daten vor versehentlicher oder intendierter Zerstörung, unautorisierter Betrachtung oder Bearbeitung sowie deren Weitergabe an Dritte geschützt werden. Überdies wird eine reibungslose Funktionalität sichergestellt, wenn mehrere Benutzer gleichzeitig auf die Daten zugreifen.

SQL ist eine spezielle Programmiersprache, welche in einer geeigneten Umgebung zum Einsatz kommt – und zwar entweder benutzerfreundlich oder interaktiv zum sofortigen Erledigen von Aufgaben (meistens Anfragen). Zuweilen werden ihre Befehle auch in die Wirts-Programmsprache eingefügt. Allerdings handelt es sich bei SQL nicht um eine vollwertige Programmiersprache, zum Beispiel deshalb, weil es in den meisten Anwendungen keine Steuerungsprogrammkonstrukte gibt. Darüber hinaus fehlt es noch an anderen Elementen, die jede allgemeine Programmiersprache enthalten sollte. SQL ist daher ein standardisiertes Werkzeug, um mit relationalen Datenbanken zu arbeiten. Sie ist kein Datenbanksystem, aber ein anderweitig integrierter Teil eines Verwaltungs-systems für Datenbanken.

SQL ist vor allem eine interaktive Abfragesprache: Sie ermöglicht es, quasi in Echtzeit Antworten auf sehr komplizierte Abfragen zu bekommen. Sie ist kein prozesstechnisches Werkzeug mit einer Menge an Datenzugriffen und ist eine standardisierte Sprache. SQL ist nachvollziehbar, da es Daten in Tabellenform versteht, weshalb sie für Benutzer leicht zu verstehen ist. SQL arbeitet mit relationalen Datenbanken, in welchen dem Benutzer Daten als eine Menge verschachtelter Tabellen angezeigt werden. Jede Tabelle stellt eine Datenmenge dar, welche in Form von Zeilen (Einträgen) und Spalten (Objekte) angeordnet ist. Der Benutzer nimmt einen Datenwert als Element in einer Matrix wahr.

In der überwiegenden Mehrheit der Fälle ist das Resultat einer in SQL beschriebenen Aufgabe eine Datenmenge aus einer Tabelle oder mehreren Tabellen, eine sogenannte Ergebnistabelle, welche nicht immer das letztendliche Produkt sein muss. Sie kann als eine

Menge von Eingabedaten für die weitere Verarbeitung dienen, zB zum Drucken eines Etiketts oder zum Zeichnen eines Diagramms.

SQL kann als „gängige Sprache“ dienen, vor allem wenn sie in Netzwerken operiert, in welchen verschiedenen Datenbankprodukten verwendet werden.

SQL wird auch verwendet, um Vorschauen (Abfragen) zu erstellen. SQL Vorschauen ermöglichen es, verschiedene Ansichten von Tabellenstrukturen und Daten für verschiedene Benutzer zu erstellen. Jeder Benutzer sieht nur die Daten, die für ihn bestimmt sind. Die Daten werden von dem Benutzer wieder als einfache Tabelle gesehen, obwohl die Daten in Wahrheit von einer anderen Tabelle stammen. Die angezeigten Daten in der Ansicht sind dynamisch. Werden die Daten in den Tabellen (Datenbankdateien) hin- und herbewegt, ändern sich auch die Daten, welche die Vorschau anzeigen. Umgekehrt ist es dasselbe. Arbeitet man mit einer Aktualisierungsvorschau (eine spezielle Art der Vorschau, die nicht nur die Anzeige von Daten, sondern auch deren Ergänzung und Aktualisierung erlaubt) und verändert man dort die Daten, werden die Veränderungen in einer geeigneten Tabelle (Datenbankdatei) wiedergegeben.

Eine Schlüssel-SQL-Aussage ist ein Befehl. Jeder Befehl beginnt mit einem Schlüsselwort. Das Wort gibt an, zu welcher Aktivität der Befehl führt. Auf das Schlüsselwort folgen ein oder mehrere optionale Klauseln, welche die Natur der zu erledigenden Tätigkeit oder die Daten, mit deren Hilfe ein Befehl ausgeführt werden soll, spezifizieren.

Jeder Abschnitt beginnt mit einem Schlüsselwort, wie zB FROM oder WHERE. Einige Klauseln sind verpflichtend, andere optional. Jede SQL-Anwendung verwendet – neben den standardmäßig von den ANSI/ISO-Konventionen vorgegebenen Klauseln – ihre eigenen Klauseln. Manchmal unterscheidet sich die Funktionsweise von Standardklauseln geringfügig.

SQL-Objektnamen sind in der Regel zwischen einem und 18 Zeichen lang, das erste Zeichen muss ein Buchstabe sein und die Bezeichnung darf keine Leerzeichen oder Interpunktion enthalten. Wenn man ein Objekt mit einem Namen bezeichnet, ist es oft notwendig, diesen Namen einzuschränken. Dies ist dann der Fall, wenn es mehrere gleiche Namen gibt und nicht klar ist, auf welches Objekt sich ein Name bezieht. Eine Namenseinschränkung wird mittels `.´(Punkt)-Vermerk vorgenommen. Sehr häufig werden Einschränkungen bei Tabellenbezeichnungen eingesetzt:

```
<Owner>. <Table_name>
```

Oder wenn Spalten eingeschränkt werden, was in Datenbanksystemen gang und gäbe ist.

`<Tablename>. <Columnname>`, in der Regel sind Einschränkungen auch in SQL erlaubt.

Auf mehreren Ebenen:

```
<Owner>. <Table_name>. <Column_name>
```

5.2. SQL Abfragen

Eine Einführung in Aussagen in SQL-Datenbanken.

Einsatz von SQL-Befehlen

Zunächst einmal gilt es, standardmäßige SQL-Statements in einfachen Beispielen einzusetzen. Die so erzeugten Informationen können dann in der Programmierung von Webseiten, in Visual Basic oder Delphi verwendet werden. Die allgemeine Einführung in Datenbanken ist im Einleitungsartikel zu finden. Wenn man die Datenbankentheorie bereits versteht, kann man sich im nächsten Schritt anschauen, wie eine Datenbank in Microsoft Access erstellt wird.

Liste mit SQL-Statements

Die wichtigsten Befehle sind: SELECT, INSERT, DELETE, CREATE, FROM, WHERE und viele andere, welche in den folgenden Kapiteln vorgestellt werden. Zunächst wird jedoch eine einfache Tabelle erstellt, in welcher besagte Befehle eingesetzt werden (um klar zu machen, wozu jeder Befehl dient).

Erstellen einer Übungstabelle


zamestanci : Tabulka

ID	id_zam	jmeno	prijmeni	funkce	kancelar	plat	vek
1	1	Pavel	Velky	vyvoj	vyvojova	10000	20
2	2	Roman	Maly	vyvoj	konstrukcni	12000	25
3	3	Petr	Vezir	konstrukter	konstrukcni	15000	35
4	4	Adela	Nevečeřelova	učetni	finanční	10000	20
5	5	Martin	Šéf	učetni	finanční	18000	23
6	6	Tomáš	Bílý	ředitel	ředitelství	100000	35
7	7	Miloslava	Bílá	asistent	ředitelství	50000	16
8	8	Ferdinand	Velky	topič	ředitelství	25000	66
9	9	Kamila	Novaková	nástěnkář	ředitelství	35000	25
10	10	Laura	Benešova	sekretař	konstrukcni	25000	55
11	11	Jarmil	Pražsky	finance	zlodejovna	30000	36
12	12	Jarmila	Jungová	ekonom	zlodejovna	35000	28
13	13	Dalibor	Navratil	ekonom	zlodejovna	32000	19
14	14	Norbert	Mikulec	skladnik	sklad	2000	48
15	15	Iveta	Vezirova	konstrukter	vyvojova	18000	47
*	(číslo)	0				0	0

Záznam: 1 z 15

Um die Begriffe im SQL-Skript zu erklären empfiehlt es sich, diese in ein paar einfachen Tabellen zu testen. Sobald man sie einmal verstanden hat, lassen siche sich in einer 10.000 Zeilen umfassenden Tabelle einsetzen. Diese Tabelle kann mittels MySQL, FoxPro, Acces oder einem anderen Datenbankprogramm erstellt werden.

SELECT Aussage

Ein wichtiger Befehl. Wir verwenden unsere erstellte Tabelle und probieren die Anwendung dieses Befehls aus. Die einfachste Form der Aussage sieht wie folgt aus:

```
SELECT *
FROM employees;
```

Listet die Inhalte der gesamten Tabelle „Employees“ auf.

```
SELECT the name
FROM employees;
```

Listet die Inhalte der Spalte „Name“ aus der „Employees“-Tabelle auf.

```
SELECT DISTINCT function
FROM employees;
```

Listet die Inhalte der Spalte „Funktion“ in der „Employees“-Tabelle auf, zeigt aber nur die Objekte ohne Duplikate an (auch nur dann einmal, wenn der Entwickler 4x gelistet hat).

```
SELECT AVG(plat)
FROM zamestanci;

SELECT SUM (flat)
FROM employees;
```

Die Spalte AVG berechnet die Steigung einer vorgegebenen Spalte, die Spalte SUM berechnet die Summe der Werte einer vorgegebenen Spalte, oder es können auch in der Abfrage selbst Berechnungen durchgeführt werden.

```
SELECT SUM (PLAT + 500)
FROM employees;
WHERE
```

Wenn man 10.000 Zeilen hat, wird man die Berechnungen auf ein paar Kriterien beschränken wollen. Dies wird mithilfe von WHERE zusammen mit ein paar mathematischen Symbolen erreicht.

```
SELECT name
FROM employees
WHERE pay> 20000;
```

Listet die Namen der Mitarbeiter auf, die mehr als 20.000 Tschechische Kronen (ca. 800 Euro) verdienen.

Gebräuchliche einfache Abfragesyntax

Wählt (SELECT) die Liste von Spalten aus der WHERE-Tabelle der WHERE-Suchbegriff um nach den Sortierspalten zu ordnen (ORDER).

Die Spaltenliste ist entweder eine durch Kommas geteilte Liste von Spalten oder ein * Zeichen, um alle Spalten auszuwählen. Der Spaltenname ist entweder nur der Spaltenname oder der columname.class_name. Es kann auch ein Tabelleneintrag * verwendet werden, um alle Spalten aus der Tabelle auszuwählen. Die Tabelle kann jede der folgenden sein:

- Sitzung (echte Datenbanktabelle)
- Ergebnis einer anderen SELECT-Abfrage
- Ansicht
- Ergebnis des JOIN-Operators (virtuelle Tabelle)

Der Suchbegriff ist eine Bedingung, die jeder Datensatz erfüllen muss, welcher in den Abfrage-Ergebnissen auftaucht. Selbstverständlich muss ein Datensatz erst in der Quelltabelle existieren, bevor dieser geschrieben werden kann. Deshalb ist WHERE eine einschränkende Bedingung, es sei denn sie wird aufgelistet, sodass alle Zeilen in der Tabelle geschrieben werden. Die Column-Spalten sind eine Liste durch Kommas abgegrenzter Spalten, gemäß welcher die Ergebnistabelle sortiert wird. In der ersten Tabelle ist das primäre Sortierungskriterium, die zweite Spalte ist das sekundäre Sortierungskriterium, wenn man nicht auf Basis der ersten Spalte entscheiden kann usw.

Beispiel 1

Eingabe: Wählen Sie alle Personen mit dem Namen Radka und Tomas aus der Datenbank aus.

Lösung 1

```
SELECT name, surname, nickname FROM person
WHERE name = 'Radek' ORname = 'Tomas'
```

Man kann gebräuchliche logische Operatoren AND und OR verwenden, in den meisten Datenbanken geht stattdessen auch die Schreibweise && und ||. Es lässt sich das gebräuchliche = verwenden, um Zeichenketten (Strings) zu vergleichen, wobei es ein paar Dinge zu bedenken gilt. Abhängig von den Servereinstellungen kann die Zeichenlänge von Bedeutung sein oder nicht. Die Standard-Einstellung (und deshalb häufiger) ist, dass die Zeichenlänge egal ist. Der Akela.mendelu.cz-Server unterscheidet Groß- und Kleinschreibung. Es hängt auch von den Einstellungen des Textvergleichsserver (und der Datenbank) ab, ob Akzente beachtet werden oder nicht. Akela.mendelu.cz berücksichtigt ferner auch diakritische Zeichen, weshalb der Name exakt so geschrieben werden muss, wie er in der Datenbank gespeichert ist.

Lösung 2

SELECT name, surname, nickname FROM persons WHERE name IN ('Radek', 'Tomas')
Eine Alternative zur vorherigen Abfrage ist die Verwendung des IN-Operators. IN ist ein festgelegter Operator, welcher überprüft, ob sich ein Wert in einer spezifizierten Menge befindet. Das heißt, dass aus jeder Zeile der Wert der Namensspalte genommen und daraufhin getestet wird, ob sich dieser in der vordefinierten Menge befindet (‚Radek', ‚Tomas'). Die Menge kann entweder mittels direkter Enumerierung von Werten oder durch eine andere SQL-Abfrage definiert werden.

Beispiel 2

Eingabe: Wählen Sie alle Menschen in der Datenbank aus, die in Prag leben.

Lösung – Schritt 1

```
SELECT id_address, FROM FROM WHERE address mesto = 'Praha'
```

Im ersten Schritt werden alle Adressen eruiert, die sich in Prag befinden. Zu diesem Zweck empfiehlt es sich, das Datenbank-Schema abzurufen, welches offenbart, dass es eine Spalte id_address in der Personentabelle gibt, welche mit der Spalte id_address in der Adress-Tabelle verknüpft ist. Der Grund dafür ist folgender: Werden alle Personen ausgewählt, deren Adresse im Ergebnis der oberen Abfrage aufscheint, erhält man genau jene Personen, deren Adresse sich in Prag befindet.

Lösung – Schritt 2

```
SELECT name, receive, nickname FROM people
WHERE id_addresses IN ()

SELECT DISTINCT id_address FROM WHERE address
mesto = 'Praha' ()
```

Nun gibt es zwei SELECT-Konstruktionen in der Abfrage. Das zweite SELECT ist das die sogenannte Sub-Abfrage und es handelt sich um eine leichte Modifikation der Abfrage, welche im ersten Schritt erstellt wurde. Die erste Anpassung ist, dass sie nun nur die Spalte id_address auswählt. Dies ist angesichts der Tatsache, dass die Eltern-SQL-Abfrage nach ID_Adressen sucht, essentiell. Die Sub-Abfrage darf daher nur die ID_Adressen zurückgeben. Zusätzlich arbeitet der IN-Operator nur mit einer Skalar-Menge, weshalb eine Sub-Abfrage (zur Vorgabe einer Menge für den IN-Operator) stets nur eine Spalte zurückgeben darf. Werden mehrere Spalten in der Sub-Abfrage verwendet, wird der Datenbankserver folgende Fehlermeldung erhalten: ERROR: Sub-Abfrage hat zu viele Spalten. Die zweite Veränderung ist, dass das Schlüsselwort DISTINCT hinzugefügt wird. Dadurch wird sicher-gestellt, dass wie zurückgegebenen Werte eindeutig sind (siehe die nächsten Beispiele). In diesem besonderen Fall ist es unnötig (da die id_address immer einmalig ist). Trotzdem ist es ratsam, DISTINCT einzubauen. Einerseits handelt es sich dabei um eine allgemeine Vorbeugungsmaßnahme, andererseits tendiert der IN dazu bei Sub-Abfragen, in welchen sich die Werte wiederholen (dann ist es keine Menge), sich seltsam zu verhalten.

6. SQL – KOMPLEXERE ABFAGEN

Übungen: SQL-Abfragen mit mehreren Tabellen

6.1. Beispiel 1

Eingabe: Schreiben Sie eine SQL-Abfrage, welche aus der Datenbank alle Personen auswählt, die eine ICQ-Nummer haben. Wählen Sie den Vornamen, den Familiennamen und die ICQ-Nummer der Person aus. Sortieren Sie in aufsteigender Reihenfolge nach dem Familiennamen.

Möglichkeit 1

Die Leichteste: berechnet, dass der Wert von id_type_contact = 1 mit dem ICQ-Typ in der contact_type-Tabelle übereinstimmt. Der JOIN-Operator arbeitet stets mit zwei Kalkulationstabellen. Die Kopplungsbedingung ist immer, dass sich Werte in ein paar Tabellen gleichen. Die join-Bedingung muss beide Tabellen enthalten, die sich im JOIN-Abschnitt befinden. Die join-Bedingung muss Spalten enthalten, welche die Tabellen verbinden. Es gibt keinen Verweis auf die Kontakte-Tabelle in der Personen-Tabelle. Die Kontakte-Tabelle ist nur ein einzelner Verweis auf die Personen-Tabelle – die Spalte id_object. INNER JOIN wählt aus der Tabelle nur jene Datensätze aus, die der Verbindungsbedingung entsprechen. Es handelt sich dabei ausschließlich um jene, welche einen Kontakt haben (aber da der Suchbegriff in der Abfrage spezifiziert wurde, ist es in diesem speziellen Fall auch möglich, LEFT und RIGHT JOIN zu verwenden, wobei es sich hierbei nur um einen übereinstimmenden Umstand handelt).

```
SELECT person.name, person.name, contact.contact FROM
     INNER JOIN contacts
          ON contacts.id_osoby = person.id_osoby
WHERE contacts.id_types_contact = '1'
ORDER BY by ASC
```

Möglichkeit 2

Identisch zur vorherigen Möglichkeit, wobei sie USING verwendet. Wenn die Verknüpfungsbedingung die Gleichwertigkeit gleichgenannter Spalten (id_id) ist, kann deren Eingabe vereinfacht werden.

```
SELECT person.name, person.name, contact.contact FROM
     Persons JOIN contacts USING (id_persons)
WHERE contacts.id_types_contact = '1'
ORDER BY by ASC
```

Möglichkeit 3

Die bessere Möglichkeit: Wählt Kontakte auf Basis des Kontaktartnamens aus, weshalb sie nicht auf irgendeinen id_type_contact Wert angewiesen ist. Der Kontaktartnamen befindet sich in der contact_type-Tabelle. Es ist wichtig zu bedenken, dass das Ergebnis verbundener Tabellen eine Tabelle ist und dass der JOIN-Operator immer mit zwei Tabellen arbeitet.

```
SELECT person.name, person.name, contact.contact FROM
     (INNER JOIN contacts
          ON contacts.id_osoby = person.id_osoby)
               INNER JOIN contact_types
                    ON contact_types.id_type_contact =
                    contacts.id_types_contact
WHERE contact_name.nazev = 'icq'
ORDER BY by ASC
```

SELECT ist darüber hinaus ein Befehl, der viele andere Worte haben kann, einige Tabellen ineinander verschränken kann usw. Aber die gesamte Grundlage der Syntax ist: SELECT, dann die Liste der Spalten, welche man erhalten oder kumulieren möchte, oder andere Funktionen FROM und der Name der Tabelle, aus welcher man Daten extrahieren möchte, WHERE und der Ausdruck mit Beschränkungen, welche für die gewünschten Zeilen gelten sollen. In diesem Fall soll die Stadt-Spalte genau den Wert „Chomutov" enthalten.

Zusammen mit der Kumulierungsfunktion sieht dies wie folgt aus: Man will herausfinden, wie viele Zeilen das sind, welche Personen aus Chomutov entsprechen:

```
SELECT COUNT (*) FROM lide WHERE mesto = ”Chomutov”;
```

Es wird eine Zeile in einer Spalte wiedergegeben, welche die Zahl 2 enthält.

Die Bedingungen können mit den boolean‘schen Operatoren AND und OR, Klammern und SQL-Funktionen viel komplizierter sein. Eine Liste dieser ist in der Dokumentation zu finden. Welche Personen aus Chomutov über 30 Jahre alt sind, findet man auf folgende Weise heraus:

```
SELECT * FROM lide WHERE mesto = ”Chomutov” AND age> 30;
```

Ein Sternchen bedeutet „alle Spalten“.

Andere Tabellen werden mithilfe von JOIN mit einer SELECT-Abfrage verknüpft. Dieses Unterfangen erfordert mehr als eine Tabelle und eine Form der Beziehung zwischen ihnen. So braucht man zB die bereits vorliegende Menschentabelle, wo jede Person eine eindeutige ID hat, sowie eine Gefahren-Tabelle mit einer Spalte clovek_id und einer Herausforderung. In den Spalten der Personen wären Zahlen die Personen, welche auf die Idee kamen, und in der Spalte der Idee wäre der Text einer Idee. Anschließend könnte man eine Abfrage erstellen, welche alle Ideen auflisten und jeder davon eine Spalte namens „Mensch“ hinzufügen würde. Aber Delaily geht über diese Vorstellung hinaus:

```
SELECT napady.napad, lide.jmeno FROM napady
LEFT JOIN lide ON lide.id = napady.clovek_id;
```

Das Ergebnis wird eine Tabelle mit Ideen sein, wo es sich bei der ersten Spalte um die Idee und bei der zweiten um die Person handeln wird, von der diese kam.

Wenn man „DELETE“ statt „SELECT (Spalten)“ eingibt, erhält man eine Abfrage zur Löschung von Zeilen. Sie funktioniert gleich wie SELECT (hier gibt es eine WHERE Bedingung), aber statt Ergebnisse zu erhalten, werden die Zeilen, welche mit der Bedingung übereinstimmen, gelöscht. Um Adam zu löschen, muss man nur folgendes eingeben:

```
DELETE FROM lide WHERE name = ”Adam”;
```

Eine der am häufigsten verwenden SQL-Abfragen ist immer noch UPDATE. Diese wird eine Zeile bearbeiten. Wenn Catherine ein Jahr älter ist, wird dies wie folgt aktualisiert:

```
UPDATE people SET age = 30 WHERE name = ”Catherine”;
```

Es können auch mathematische Ausdrücke und Verweise auf bestehende Spaltenwerte verwendet werden. In diesem Fall lässt sich die Spalte Alter wie folgt um die Zahl eins erhöhen:

```
UPDATE people SET age = age + 1 WHERE name = ”Catherine”;
```

Das sind soweit die am häufigsten verwendeten SQL-Abfragen (Klauseln). Individuelle, komplexere und präzisere Abfragen können mehr Wörter enthalten. Um diese zu verstehen ist es jedoch notwendig, sich intensiver mit den individuellen Unterlagen zu beschäftigen oder nach einem spezifischen Problem zu suchen. Verwendet man z.B. die Kumulierungsfunktion in SELECT, erhält man das Ergebnis der Kumulierung der gesamten Tabelle. Möchte man hingegen die Durschnitts-Temperatur der letzten Jahre ermittelt und hat dafür die Monatstemperaturen zur Verfügung, ist es notwendig, GROUP BY zu verwenden. Wenn man bei Google „GROUP BY YEAR DATETIME MYSQL“ eingibt, findet man heraus, dass man DATETIME YEAR benötigt:

```
GROUP BY YEAR (record_date)
```

Neben GROUP BY kann man auch ORDER BY auswählen, wo man spezifiziert, nach welchen Spalten die Ergebnisse sortiert werden sollen und ob dies ab- oder aufsteigend geschehen soll. Manchmal ist die Satzstellung wichtig. Für SELECT findet man diese in den Unterlagen und man sieht überdies, dass man zuerst GROUP BY und dann ORDER BY verwenden muss.

7. GRAPH-DATENBANKEN

Ein Graph ist eine Datenstruktur, die aus Ecken und Kanten besteht. Die Graph-Datenbank ist ein Datenspeicher und -verarbeitungssystem in Form eines Graphen. In vielen Fällen werden Domänen sehr natürlich modelliert: So enthalten solche Domains menschliche Beziehungen, Gene und Proteine, mobile Netzwerke, Distributionsnetzwerke verschiedener Art, neuronale Netzwerke oder ökologische Netzwerke, welche die Interaktionen zwischen Organismen erfassen.

Graphische Datenbanken enthalten oft verschiedene Systeme, welche Daten in Graphen verwalten. Ausgehend von dieser Definition wäre es theoretisch möglich, sich relationale Datenbanken anzusehen, bzw. deren Überbau wie bei Graphen-Datenbanken. Relationale Datenbanken hingegen erlauben keine effektive Speicherung und Abfrage von Graphendaten.

7.1. Rationale Datenbanken

Der ansteigende Pfad in einer relationalen Datenbank erfordert JOINS und ist deshalb ineffizient für tiefe Arbeitsgänge. Ein JOIN, welcher den Index verwendet, erfordert eine logarithmische Zeit O (log n), ohne dass der Index O ist (n log n).
I
So können zB menschliche Beziehungen mithilfe von zwei Tabellen modelliert werden: Man und Beziehung. Für jeden Durchlauf durch die Beziehung zwischen zwei Personen ist in der Regel ein JOIN notwendig. Traditionelle relationale Datenbanken sind für JOIN-Einheiten optimiert. Da es etliche JOINs gibt, bekommen relationale Datenbanken trotz der Verwendung von Indizes Probleme.

7.2. True Chart Datenbanken

Als echte Graph-Datenbanken werden zum Zweck dieses Artikels jene Datenbanken bezeichnet, die nur eine konstante Zeit O (1) für den Durchlauf des Graphen. True Chart Datenbanken speichern Daten, wie andere NoSQL-Datenbanken auch, in einer denormalisierten (bereits „fusionierten“) Form. Für echte Graph-Datenbanken sind sie deshalb besonders wichtig für das Schreiben, während der Sendeabruf minderwertiger ist.

Ein Beispiel für eine True Chart Datenbank ist Neo4j, welche seit mehr als einem Jahr entwickelt wird und weit genug ausgereift ist, um Produkte zu entwickeln. Ein weiteres Beispiel ist das neuere OrientDB System. Beispiele für Datenbanken, welche Graphendaten speichern und abrufen können, aber nicht notwendigerweise effizient Diagramme durchstöbern, sind die meisten Dreiergruppen (Sesame, Jena, Virtuoso) oder FlockDB (Twitter

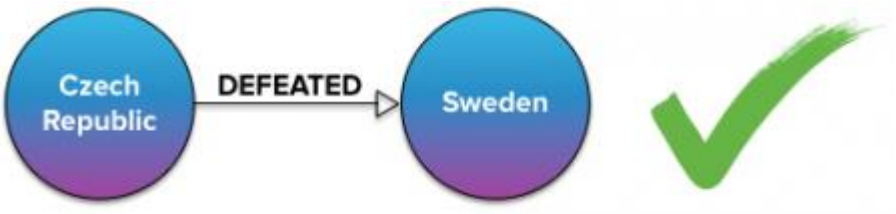
Wie bei den meisten NoSQL-Datenbanken gibt es hier keine einfache Abfragesprache für Graphen-Datenbanken. Allerdings implementieren viele Chart Datenbanken die Gremlin Chart Crawling Sprache, welche in Zusammenarbeit mit den Neo4j-Autoren entwickelt wurde.

Der Übergang von der relationalen Welt in die Welt der Graphen erfordert eine Veränderung im Denken sowie bei der Betrachtung von Daten. Obwohl Graphen viel häufiger intuitiver sind als Kalkulationstabellen, unterlaufen Menschen beim Modellieren solcher Diagramme ständig Fehler. In diesem Artikel werden die häufigsten Fehler thematisiert, ebenso wie die Modellierung von wechselseitigen Beziehungen. Zum Schluss wird auch ein echtes Beispiel gezeigt, welches dazu dient, die Reihe fortzusetzen.

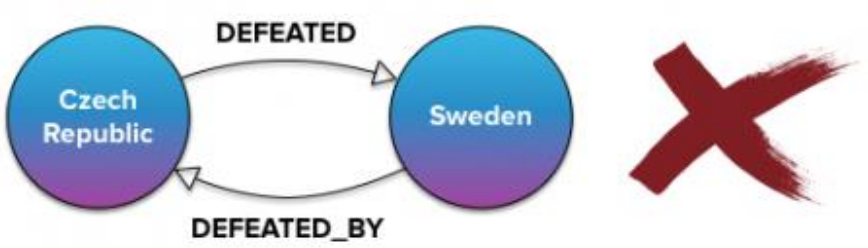
7.3. Einseitige Beziehungen

Beziehungen in Neo4j müssen von einer Art sein, welche der Beziehung eine semantische Bedeutung gibt und die auch eine festgelegte Richtung hat. Dadurch wird der Sinn unter Entitäten ausgedrückt. Anders gesagt ist die Beziehung mehrdeutig, wenn man nicht die Richtung der Beziehung festlegt.

So zeigt das folgende Diagramm zB an, dass die Tschechische Republik Schweden bei einem Eishockey-Spiel besiegt (DEFEATED) hat. Wäre die Richtung der Beziehung umgekehrt, wäre Schweden viel glücklicher. Es gibt keine Möglichkeit zu erfahren, wer der Gewinner ist, weshalb die Beziehung ergebnislos ist.



Man bemerke, dass die Beziehung in gegensätzlicher Richtung existiert, wie es unten im nächsten Graphen gezeigt wird. Hierbei handelt es sich um einen typischen Fall. Noch einmal wird dies anhand eines anderen Beispiels erklärt: Pulp Fiction wurde von Quentin Tarantino inszeniert (DIRECTED), was gleichbedeutend damit ist, dass Quentin Tarantino der Regisseur des Films Pulp Fiction ist (IS_DIRECTOR_OF). Dies kann in zahlreichen Paarungen resultieren.

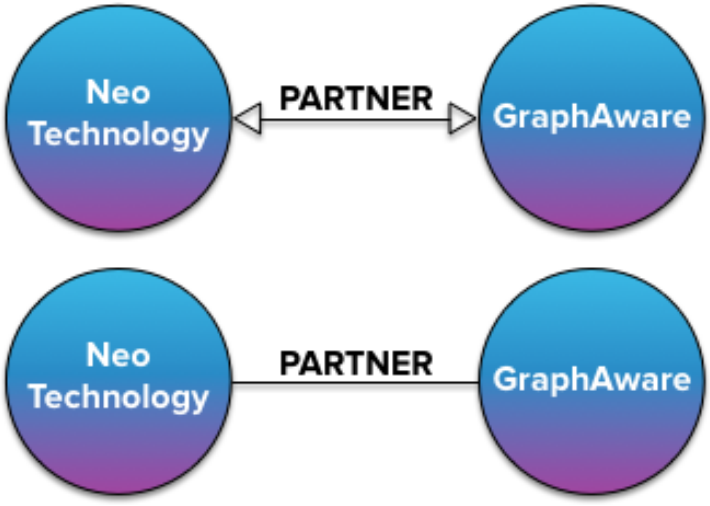


Dies ist ein Fehler, der vielen unterläuft, wenn sie ein Neo4j-Diagramm modellieren und in zwei Richtungen gehende Beziehungen definieren. Da eine Beziehung die zweite ausdrückt (symmetrische Beziehung), ist dies sowohl hinsichtlich des benötigten Platzes als auch des Graphen-Durchlaufs unwirtschaftlich. Neo4j erlaubt es einem, die Beziehung in beide Richtungen nachzuvollziehen, sprich auch in die gegengesetzte Richtung der Kanten. Dazu kommt, dass aufgrund der Art und Weise, wie Neo4j Daten speichert, die Geschwindigkeit des Durchlaufs nicht von ihrer Richtung abhängt.

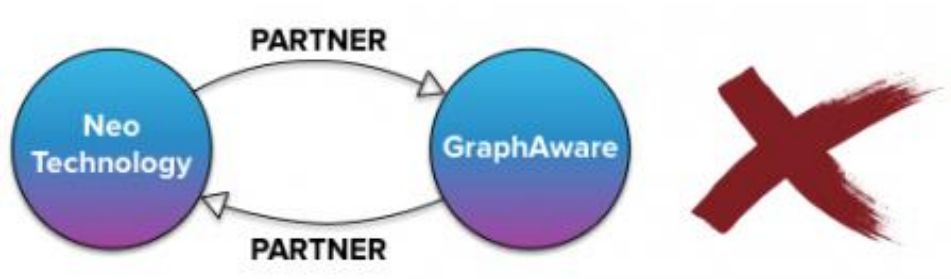
7.4. Zweiseitige Beziehungen

Einige Beziehungen sind natürlicherweise zweiseitig. Ein klassisches Beispiel ist Facebook oder eine Freundschaftsbeziehung. Die Beziehung beruht auf Gegenseitigkeit: wenn man mit jemandem befreundet ist, so ist auch dieser jemand (vielleicht) mit einem selbst befreundet. Abhängig davon, aus welcher Warte man das Diagramm-Modell betrachtet, kann man sagen, dass es entweder wechselseitig oder nicht ausgerichtet ist.

Ein Beispiel: GraphAware und NeoTechnology sind Partnerunternehmen. Da es sich um eine Beziehung handelt, kann diese entweder als zweiseitige oder nicht ausgerichtete Beziehung modelliert werden.



Dies ist jedoch in Neo4j nicht möglich: Hier muss jede Beziehung einen Ausgangs- und einen Abschlussknoten haben. Anfänger greifen häufig auf das folgende Modell zurück, welches genau dasselbe Problem hat wie das zuvor genannte Eishockey-Beispiel: Redundanz.



Die Neo4j-API erlaubt es Entwicklern, die Ausrichtung der Beziehung bei Bedarf komplett zu ignorieren, wenn Abfragen geschrieben werden. In Cypher könnte die Suche nach allen Partnerunter-nehmen von NeoTechnology zB wie folgt aussehen:

```
MATCH (neo) - [: PARTNER] - (partner)
```

Das Ergebnis wäre dasselbe, wenn man die Ergebnisse dieser zwei unterschiedlichen Abfragen zusammenführen würde:

```
MATCH (neo) - [: PARTNER] -> (partner) and MATCH (neo)
<- [: PARTNER]
```

Die richtige (oder zumindest effizienteste) Möglichkeit, Partnerschaftsbeziehungen zu modellieren ist daher, eine einzelne PARTNER-Beziehung zu verwenden, die beliebig ausgerichtet sein kann.

7.5. Zusammenfassung

Beziehungen in Neo4j können gleich schnell in beide Richtungen durchlaufen werden. Darüber hinaus kann die Ausrichtung völlig ignoriert werden. Aus diesem Grund ist es nicht notwendig, zwei verschiedene Beziehungen zwischen Entitäten zu erstellen, wenn die Kanten der gegengesetzten Richtung dieselbe Bedeutung haben.

7.6. Grafische Datenbankmodellierungsmethodologie

Im folgenden Artikel werden die Gestaltung, Implementierung und Überprüfung einer Neo4j Graphen-Datenbank für das Projekt „BestPartyToday“ gezeigt (hierbei handelt es sich um eine weltweite Teilliste, welche detaillierte Statistiken über Events anbietet). Aktuell verwendet die MySQL-Datenbank dutzende Millionen Datensätze, weshalb man auch darauf warten muss, dass Daten aus einem relationalen Datenbankspeicher in einen Graphen übertragen werden. Im heutigen Arbeitspapier wird allerdings nur die Graphen-Datenbank von Tabellen und Attributen modelliert, die aus der aktuellen Struktur der MySQL-Datenbank ausgewählt wurden.

Das konzeptionelle Design der Graphen-Datenbank wird auch als Whiteboard-freundlich bezeichnet. Um schnell alle Ideen und Erkenntnisse wiederzugeben, die im Zuge des Gestaltungs-Brainstormings gewonnen wurden, bedarf es nur eines Flipcharts. Das daraus resultierende Layout kann sehr leicht anderen Projektgruppen (Verkaufsabeilung, Kontomanager, Vertriebsmannschaft etc.) präsentiert werden, welche keine technische Ausbildung haben. Es ist nicht länger ein Problem für sie, das Design zu verstehen, da sich das Layout einfach an das reale Leben anpassen lässt.

7.7. Implementation des konzeptionellen Modells

Das Top 5-Diagramm der aktuellen relationalen Datenbank, repräsentiert durch Symbole, welche die Knoten und Richtungspfeile für Beziehungen darstellen, wird im schlussendlichen Graphendiagramm-Design des BestPartyToday-Projekts verwendet. Anhand der Entitäten (Knoten und Beziehungen) wird der Graph durchstöbert. Dabei werden die benötigten Daten gewonnen, welche anschließend verarbeitet und dem Benutzer der Anwendung angezeigt werden. Ihre Bedeutung ist in der nachfolgenden Tabelle beschrieben. Ein Überblick über die Diagramm-Modell-Entitäten.

8. NOSQL-DATENBANKEN

NoSQL-Datenbanken sind das Thema, dem sich diese Arbeit widmen wird. Zunächst einmal muss erläutert werden, was eine NoSQL-Datenbank ist. Im nächsten Kapitel geht es dann um das CAP-Theorem. Dieses erklärt, dass einer der Gründe dafür, warum es so viele NoSQL-Produkte gibt, jener ist, dass jedes davon bestimmte Eigenschaften nur auf Kosten von anderen zur Verfügung stellen kann. Aus diesem Grund muss sich der Benutzer entscheiden, welche Kombination von Eigenschaften für ihn die wichtigste ist. Auf dieser Grundlage fällt dann die Entscheidung für ein Produkt.

Hinzu kommt, dass sich die Arbeit damit auf die Skalierbarkeit fokussiert, sprich die Fähigkeit, die Leistung des Datenbankservers zu steigern. Dies gelingt entweder durch die Verbesserung des Servers mittels effizienterer Hardware oder durch Aufteilung der Datenbank auf mehrere Server. Ähnlich wichtig wie die Skalierbarkeit ist auch das Teilen, eine Methode, mit der Datenbanken in mehrere Teile aufsplittet werden, die eigenständig arbeiten können und schneller sind als eine große Datenbank. Als nächstes geht es um die bekanntesten NoSQL-Produkte, wobei drei Vertreter davon herausgegriffen werden, auf die näher eingegangen wird.

8.1. Beschreibung

Wie bereits in der Einleitung erwähnt, befassen sich NoSQL-Datenbanken nicht mit der relationalen Art der Datensteuerung, sondern einer anderen. Sie basierend nicht primär auf Tabellen und verwenden kein SQL, um mit Daten zu arbeiten. Ihre Domäne ist eine hohe Suchoptimierung auf Kosten geringer Funktionalität, welche oft auf einfache Datenspeicherung begrenzt ist. Diese Mängel im Vergleich zu SQL werden durch Skalierbarkeit und Leistung in bestimmten Datenmodellen wettgemacht. NoSQL ist für die Speicherung großer Datenmengen geeignet, die nicht verknüpft werden müssen. Strukturierte Daten sind hingegen erlaubt.

NoSQL-Datenbanken sind in der Lage – mit Ausnahme von Konstanz – sämtliche ACID-Kriterien (siehe Kapitel 9.1) bei der Ausführung von Arbeitsvorgängen vollumfänglich zu unterstützen. In manchen Situationen erhält man aufgrund der Missachtung von ACID eine bessere Zugänglichkeit sowie bessere Skalierbarkeit. Dieser Zugang ohne „starke Konstanz“ ist für NoSQL geeignet, welche ihn auch oft anwendet. NoSQL hat eine dezentralisierte Architektur, welche fehlertolerant ist. Einige Daten können sich auf mehreren Servern befinden, weshalb der Ausfall eines Servers toleriert werden kann. Diese Datenbanken skalieren für gewöhnlich horizontal und veralten große Datenmengen, wobei Leistung wichtiger ist als Konstanz.

NoSQL ist buchstäblich eine Kombination aus zwei Wörtern: No und SQL. Ihre Technologie steht jener von SQL entgegen. Die Abkürzung mag etwas verwirrend sein und es gibt kei-

nen Konsens unter den Menschen darüber, was es bedeutet. Die am weitesten verbreitete Auffassung ist jedoch, dass es sich um ein Akronym für „Not only SQL“ handelt. Was auch immer genau diese Abkürzung bedeutet, NoSQL ist mittlerweile ein Überbegriff für eine eigene Art von Datenbanken, welche nicht zu den bekannten RDBMS (relationalen Datenbank-Managementsysteme) gehören und gemeinhin mit großen Datenmengen assoziiert werden.

8.2. Was ist NoSQL?

Zunächst einmal ist es wichtig zu betonen, was NoSQL ist. Es ist definitiv KEIN SQL, weshalb es dazu tendiert, relationale Datenbanken abzulehnen. Die Abkürzung NoSQL bedeutet „Not Only SQL“, was so viel heißt wie dass es durchaus eine Alternative zu SQL gibt, die in bestimmten Fällen besser geeignet ist.

Bei größeren Anwendungen sieht man, dass für einige Daten relationale Datenbanken, für andere NoSQL-Datenbanken und für andere Daten wiederum andere NoSQL-Datenbanken geeignet sind. Dieser pragmatische Ansatz, mehrere Datenbanken in einem Projekt zu vermischen, wird of als Polyglott-Beharren bezeichnet.

NoSQL-Datenbanken entstehen (und entwickeln sich) als Lösungsansatz aus einem echten Problem heraus. Sie stellen eine durchaus praktikable Lösung dar und sind nicht nur ein dubioses Konstrukt, welches sich realitätsferne Akademiker in ihrem Kämmerchen zusammengesponnen haben. NoSQL-Datenbanken werden gemeinhin im Kontext von Projekten geboren, die mit großen Datenmengen arbeiten müssen: Facebook (Cassandra), Google (BigTable), Amazon (Dynamo), LinkedIN (Voldemort)

Die Verwendung einer NoSQL-Datenbank kann auch sinnvoll sein, wenn weniger Daten verfügbar sind (was eher die Domäne relationaler Datenbanken ist), da einige NoSQL-Datenbanken ein Datenmodell anbieten, das für gewisse Anwendungen natürlicher ist. Aber zurück zur einleitenden Frage: Eine NoSQL-Datenbank ist eine Datenerhaltungssoftware, welche eine Alternative zur klassischen relationalen Datenbank darstellt.

9. TRANSAKTIONEN

Eine Transaktion ist eine logische Arbeitseinheit, die aus einem oder mehreren SQL-Statements besteht, welche atomar sind, was die die Erholung von SQL-Transaktionsfehlern anbelangt. Sie beginnt mit dem SQL-Bereitstellungsbefehl (SELECT, INSERT). Veränderungen, welche durch einen Arbeitsvorgang durchgeführt werden, sind für andere in Wettbewerb stehende Arbeitsvorgänge unsichtbar, es sei denn, die Transaktion endet nicht.

Ein Arbeitsvorgang kann auf vier verschiedene Arten enden:

- COMMIT beendet den Arbeitsvorgang erfolgreich und die Veränderungen werden permanent gespeichert.
- ROLLBACK unterbricht die Transaktion und alle Veränderungen werden aufgehoben, die Datenbank wird auf den Stand vor der Transaktion zurückgesetzt.
- Innerhalb des Programms: Endet das Programm erfolgreich, Arbeitsvorgang.
- Innerhalb des Programms: Endet das Programm mit einem Fehler, wird die SQL-Transaktion abgebrochen.

Das SQL-Datenbank-Zugriffsmanagement definiert zwei Befehle, um auf Tabellen zuzugreifen:

- GRANT und REVOKE. Der Sicherheitsmechanismus basiert auf Autorisierungsidentifikatoren – Inhaberverhältnis – Privilegien

9.1. Lösung: Transaktionen

Transaktionen können als die Reihenfolge von Datenbank-Operationen verstanden werden, welche die folgenden ACID-Kriterien erfüllen.

1. **Atomarität (Atomicity):** Es werden entweder alle Arbeitsvorgänge innerhalb einer Transaktion ausgeführt oder keine.

2. **Konstanz (Consistency):** Die Datenbank befindet sich sowohl vor als auch nach einem Arbeitsvorgang in einem gleichbleibenden Zustand. Alle Integrationseintschränkungen müssen erfüllt werden.

3. **Isolation:** Einzelne Transaktionen werden isoliert. Veränderungen, die während der Ausführung eines Arbeitsvorgangs gemacht werden, sind nicht in anderen Transaktionen sichtbar.

4. **Beständigkeit (Durability):** Nachdem der Arbeitsvorgang erfolgreich ausgeführt wurde, werden die Daten gespeichert und können nicht verloren gehen.

9.2. Arbeitsvorgänge in SQL

In SQL stehen die folgenden Befehle zur Verfügung, um Arbeitsvorgänge zu erledigen:

BEGIN	startet eine Transaktion ... befiehlt innerhalb einer Transaktion etwas ...
COMMIT	speichert und beendet die Transaktion
BEGIN	... befiehlt innerhalb einer Transaktion etwas ...
ROLLBACK	kehrt Veränderungen an einer Transaktion um und unterbindet diese

Isolationsstufen

ACID-Anforderungen sind ziemlich stark und zeitaufwendig. Deshalb können diese Bedingungen durch sogenannte Isolationsstufen reduziert werden. Es gibt vier Stufen (von der schwächsten bis hin zur stärksten):

1. READ UNCOMMITTED

Es ist einem Arbeitsvorgang möglich, die Daten zu lesen, welche von einer anderen Transaktion aufgezeichnet wurden, ohne dass besagte andere Transaktion mit COMMIT abgeschlossen wird. Das Lesen solcher Daten wird als „schmutziges Lesen" (dirty read) bezeichnet. Daten, welche auf diese Weise gelesen werden, können inkonsistent sein (Zeit läuft von oben nach unten ab):

2. READ COMMITTED

Auf dieser Stufe der Isolation kann es nicht mehr zu „schmutzigem Lesen" kommen. Die Daten, die hier ausgelesen werden, sind das Ergebnis einer erfolgreich mit COMMIT beendeten Transaktion. Was hingegen auftreten kann, ist ein reproduzierbarer Lesevorgang. Werden Daten in einer Transaktion mehrfach ausgelesen, kann es vorkommen, dass die wiederholt durchgeführten Lesevorgänge nicht immer zum selben Ergebnis führen. Es ist möglich, erfolgreich eine Transaktion durchzuführen, die zwischen den Transaktionsdatenauslesevorgängen einige Daten aufgezeichnet, bearbeitet oder gelöscht hat.

3. REPEATABLE READ

Auf dieser Stufe wird sichergestellt, dass es keine nicht reproduzierbare Lesevorgänge mehr gibt. Die Daten, die einmal ausgelesen wurden, können sich nicht verändern, wenn sie erneut gelesen werden. Allerdings kann ein Phantom-Lesevorgang auftreten. Werden Daten in einer Transaktion mehrfach ausgelesen, kann es vorkommen, dass die Ergeb-

Lesevorgangs neue Zeilen enthalten, die in den vorangegangenen Lesevorgängen nicht aufscheinen. Besagte Transaktionsdaten lesen erfolgreich eine simultan durchgeführte Transaktion aus, welche neue Daten produziert hat.

4. SERIALIZABLE

Auf dieser Stufe kann es zu keinem der oben beschriebenen Phänomene kommen, da zwangsweise die ACID-Kriterien eingehalten werden.

10. VERFAHREN UND FUNKTIONEN, TRIGGER UND SEQUENZEN

Wodurch unterscheiden sich Funktionen, Methoden und Verfahren?

Alle drei (Funktionen, Methoden und Verfahren) sind ziemlich ähnlich: Es handelt sich um eine DEFINIERTE Reihenfolge von Befehlen, den Teil eines Programms, der wiederholt von anderen Teilen des Programms abgerufen werden kann.

10.1. Formelle Ansicht

Aus einem formellen Blickwinkel folgt man der Nomenklatur der vorgegebenen Programmiersprache.

- Funktion ist ein Begriff funktionalen Programmierens, welcher in Sprachen wie JavaScript oder Haskell auftritt.
- Wenn man von Methoden spricht, bezieht man sich auf Funktionen des Objekt-Orientierten Programmierens (OOP), wo die Datenstrukturen und funktionale Codes mit Objekten verknüpft sind. Man findet diese Methoden in Java, Smalltalk und anderen Sprachen.
- Verfahren ist ein Begriff, den man aus verfahrenstechnischen Sprachen kennt. Man findet sie z.B. in Pascal, aber auch in ein paar Datenbanksystemen, sogenannten gespeicherten Verfahren.

Man kann Funktionen, Methoden und Verfahren jedoch auch aus einem physikalischen Blickwinkel (gemäß ihrer Charakteristika und Bedeutung) betrachten und von der Terminologie einer speziellen Programmiersprache abweichen.

Funktion
Funktionen in der Programmierung sind jenen in der Mathematik sehr ähnlich. Eine Funktion hat ein bestimmtes Definitionsfeld (Art der Eingabeparameter) und eine bestimmte Bandbreite an Werten (Art der wiederzugebenden Werte).

Beispiel eines einfachen JavaScript-Merkmals:

```
Function sum (a, b) {return a + b};
```

Methode

In Objektsprachen wie Java gibt es Methoden anstelle von Funktionen, was jedoch nicht deren Eingabe verhindert.

```
Public static int sum (int a, int b) {return a + b};
```

Obwohl *sum ()* formell eine Methode (öffentlich und statisch) ist, handelt es sich Wahrheit um eine Funktion und die Klasse hier erfüllt nur die Rolle eines Namensraums (zusammen mit dem Namen des Pakets). Es wird jedoch keine Instanz geschaffen (oder nicht). Es handelt sich um eine Gestaltungsmusterbibliothek-Klasse (Design Pattern Library Class).

In Java findet man solche Funktionen, z.B. in der java.lang.Math-Klasse. Die folgende Funktion gibt die größerer von zwei Zahlen zurück:

```
Int x = java.lang.Math.max (a, b);
```

Verfahren

Das Verfahren ist ein spezieller Anwendungsfall einer Funktion: Es hat keinen Rückgabewert und muss auch nicht zwangsläufig Eingabeparameter haben. Verfahren werden oft bei der Stapelverarbeitung verwendet, zB wird jede Stunde ein Verfahren abgerufen, welches die in der Datenbank akkumulierten Bestellungen verarbeitet und diese an ein anderes System weiterleitet.

Nachfolgendes Beispiel zeigt ein sehr einfaches PostgreSQL-Verfahren, welches die Werte der Spalten A und B in nicht adressierten Zeilen summiert und das Ergebnis in der Spalte c speichert:

```
CREATE OR REPLACE FUNCTION add ()
RETURNS void AS
$ BODY $
UPDATE table of summaries
SET c = a + b
WHERE c is NULL
$ BODY $
LANGUAGE sql VOLATILE;
```

Dieses Verfahren ist in SQL geschrieben. Darüber hinaus kann es in PLPGSQL geschrieben werden, welches die Entwicklung äußerst komplexer Verfahren ermöglicht. Alternativ kann auch die Sprache C oder eine andere verwendet werden.

Das oben gezeigte Verfahren (formell Funktion) hat keine Eingabeparameter oder Rückgabewerte, es ist einfach eine Reihenfolge von Sprachbefehlen, welche definiert und gespeichert wurden.

Es ist interessant, dass Verfahren Parameter haben können: Diese gibt es nicht nur im Kontext der klassischen Eingabe, sondern auch der Ausgabe (oder beiden). Das Ausgabeparameter-Verfahren ist ähnlich zur Funktion, obwohl es keinen klassischen Rückgabewert hat. So können zB Daten an einen Prozessor weitergegeben werden, welcher diese modifiziert.

Diese Verfahren können auch in Java simuliert werden. Hier ist ein Beispiel für eine Struktur:

```
Public class Person {public String name;
Public String Surname; Public String fullname};
```

Ein Verfahren:

```
Public class StoredProcedures {public void reportNameName
(Person o) {o.celeName = o.name + "" + oName}};
```

Die Daten (sprich die Person) werden an das Verfahren weitergegeben, außerdem werden die erforderlichen Anpassungen vorgenommen. Wenn man diesen Stil allerdings in Java programmiert, macht man wahrscheinlich etwas falsch und macht sich bereit für die Hauptvorteile von PPE.

<u>Merke:</u>

Vorsicht vor Werttransfers! Wenn man innerhalb des Java-Verfahrens eine neue Person einer Variable zuordnet, wird diese Veränderung nicht im Verfahren abgebildet. Die ursprüngliche Person bleibt unangetastet. Man kann jedoch mit den Attributen der ursprünglichen Person arbeiten – in diesem Fall dem Code des Verfahrens und dem Code der die Instanz derselben Person „sieht".

Methode

Methoden sind Teil der Klasse und (wenn sie nicht statisch sind) eng verwandt mit der vorgegebenen Klasseninstanz (Objekt). Deshalb sind sie keine Funktionen, welche mit globalen Variablen und Daten arbeiten, aber sie haben Zugriff auf die Variablen einer gegebenen Instanz (in diesem Fall Individuen).

Das vorherige Beispiel kann „objektiver" wie folgt überschrieben werden:

```
Public class Person {public String name; Public
String Surname; Public String getCeléName ()
{return name + ”” + surname}};
```

Es ist keine gespeicherte Verfahrensklasse notwendig, um die benötigte Funktionalität näher an die Daten heranzurücken (zur Personenklasse). Man muss an dieser Stelle anmerken, dass sich nicht einmal der errechnete Wert des ganzen Namens speichern lässt – sprich er wird nur dann errechnet, wenn ihn jemand benötigt, z.B. durch den Abruf der *getNameName () Methode.*

Was ist ein Trigger?

Ein Trigger ist ein Verfahren, das automatisch ausgelöst wird, wenn etwas passiert. Er lässt sich sowohl für DML (Modifikations-) als auch für DDL (Erstellungs-) Operationen festlegen. Überdies lässt sich festlegen, ob er vor (BEFORE) oder nach (AFTER) einer Operation ausgelöst werden soll. Es ist interessant, dass der Trigger als Tabelle bezeichnet (aber nicht vorgeschlagen) werden kann und dass eine beliebige Anzahl an Triggern für dieselbe Tabelle und dasselbe Ereignis definiert werden kann.

11. ANALYTISCHE WERKZEUGE - OLAP

11.1. OLAP – Informationen unter Kontrolle

Welche Kunden bringen 80% des Profits? Wie hat sich der Erfolg, Kunden zu halten, nach dem Loyalitäts-Programm verändert? Wie wirken sich verschiedene Business-Events (Kampagnen, Einführung eines neuen Produkts, Veränderungen in Unternehmensprozessen usw.) und der Markt (Wettbewerberaktivitäten, Veränderungen der wirtschaftlichen Rahmenbedingungen usw.) auf die Verkaufsfluktuation aus? Wie sieht der Vergleich mit der letzten Periode aus? Diese und ähnliche Fragen werden von jedem realistischen Manager gestellt. Jeden Tag versucht er, das Verhalten des Unternehmens auf Basis der Analyse ihm zur Verfügung stehender Daten zu verstehen. In gut laufenden Unternehmen haben ausgewählte Abteilungen Analytiker, welche nur diese Aufgabe übernehmen und nur begründete Empfehlungen an Manager abgeben. Alle diese Personen brauchen eine fertige Infrastruktur und Werkzeuge, welche es ihnen ermöglichen, ihre wichtige Aufgabe zu erfüllen.

11.2. Was die Arbeit erleichtert und wie

Der Weg zu erfolgreicher Datenanalyse beginnt mit deren Vorbereitung. Der erste wichtige Schritt ist Daten aus Unternehmenssystemen in eine geeignete Analyse- und Berichterstattungsform zu bringen. Bereits während dieser Vorbereitung ist es notwendig, sicherzustellen, dass die verschiedenen Daten und Datenqualitäten korrekt angeglichen werden. Es muss jede Verknüpfung zum Quellensystem zuverlässig dokumentiert werden, sodass die Ergebnisse immer glaubwürdig und verifizierbar sind – um Fehler zu beheben, welche in den ursprünglichen Daten auftreten können. Die daraus resultierten Daten werden in einer relationalen Datenbank gespeichert, wo sie im für operationale Analysen notwendigen Umfang verfügbar sind. Sofort verwendbare Daten setzt man ein, um detaillierte Berichte über die Leistungsfähigkeit eines Unternehmens und einfachere Analysen basierend auf detaillierten Daten zu erstellen. Für das folgende Beispiel allerdings benötigt man eine andere Art der Information: Eine landesweit aktive Firma verkauft verschiedene Produktkategorien. Der Manager unterteilt individuelle Branchen in Regionen. Die verkaufte Produktkategorie hat drei Ebenen: Kategorien, Unterkategorien, spezifisches Produkt. Die Firma hat eine logische Definition, der zufolge Verkäufe durch eine Planung geregelt sind. Die echte Einhaltung des Plans wird durch die Summe aller Produkte und aller Filialen gewährleistet.

11.3. Wie erhält man diese Informationen?

Zu diesem Zweck werden fortgeschrittene Analysen als Multidimensional bezeichnet. Der Manager kann messbare Geschäftsparameter in Form von Zusammenhängen, Blickwinkeln und auf verschieden detaillierten Ebenen verfolgen. Eine solche Analyse erfordert OLAP-(on-line analytical processing) Technologie, um sicherzustellen, dass Daten auf jene Art und Weise gespeichert und vorberechnet werden, dass spätere Abfragen für eine sinnvolle Dauer währen. Die OLAP-Technologie erlaubt es, mit Daten auf einer zusammenfassenden Ebene zu arbeiten, ein Problem oder ein Interessensgebiet zu identifizieren und die nächste Detailstufe zu erreichen, die im Zusammenhang der Entscheidungsfindung von Relevanz ist. Ein typisches Gebiet, in der die Vorteile von OLAP sichtbar werden, ist in der Verkaufsanalyse.

Abbildung 1: OLAP in der Oracle Business Intelligence – Discoverer

Mithilfe von OLAP-Werkzeugen kann ein Manager kontrollieren, wie sich die Realität im Widerspruch zur Planung entwickelt. Werden die Erwartungen nicht erfüllt, zeigt die Trendkurve diese Information rechtzeitig an, wodurch verhindert wird, dass das Geschäftsjahr mit einem Fiasko endet. Aufsummierte Zahlen lassen sich leicht herunterbrechen, sodass man sich eine Zusammenfassung der Vorgaben für jede Region anschauen kann. Eine Analyse kann so z.B. zeigen, dass das Problem nur eine Region betrifft und die anderen Regionen voll auf Plan sind. Durch weiteres Hineinzoomen gelangt der Manager auf die Filialebene, auf welcher sich identifizieren lässt, welche Filialen in der problematischen Region deutlich hinter den Erwartungen liegen. Das System erlaubt ein Aufteilen

der Zahlen bis hin zur Kategorien- und Subkategorienebene, bei Einzelprodukten können die Zahlen jedoch nur kumuliert und nicht nach Verkaufserfolg pro Standort ausgewiesen werden.

Basierend auf diesen Informationen ist es dann möglich, korrektive Maßnahmen zu entwickeln und umzusetzen, um die Situation zu verbessern. Das Beispiel zeigt, dass OLAP-Daten tatsächlich in einer Baumstruktur gespeichert werden, sprich in einem Würfel. Auf jeder Tiefenstufe dieses Baums werden Werte für den Unterbaum berechnet. So eine hierarchische Struktur lässt sich auf Grundlage der meisten Unternehmensdaten festlegen. Diese trägt wesentlich zu einer Vereinfachung der Daten bei und erleichtert die Analyse, wenn die Daten ausgewertet werden. Der Preis für diese Vereinfachung sind die Zeit und der Platz, welche zur Berechnung und zur Speicherung des Baums benötigt werden. Wenn der der Würfel allerdings gefüllt ist, kann er adäquat sehr komplizierte Abfragen beantworten. Dies betrifft vor allem das Vergleichen von Zeitfenstern, welche durch komplexe Bedingungen eingeschränkt sind.

Ein Beispiel für eine typische, komplizierte Anfrage ist: „Welche zehn Kunden aus den 20% unserer am wenigsten profitablen Bezirke sind im Vergleich zum letzten Jahr für den größten Umsatzanstieg in unserem Unternehmen verantwortlich?“ Das Ergebnis kann als Anlass dafür genutzt werden, diesen Kunden besondere Aufmerksamkeit zu widmen. Mit OLAP ist es möglich, Daten unter verschiedenen Gesichtspunkten – vielen Dimensionen – abzufragen und diese Gesichtspunkte mit beliebigen Kennziffern zu versehen. Wie bereits gezeigt wurde, ist es nicht ratsam, Daten für diese Art von Abfragen in relationaler Form abzuspeichern.

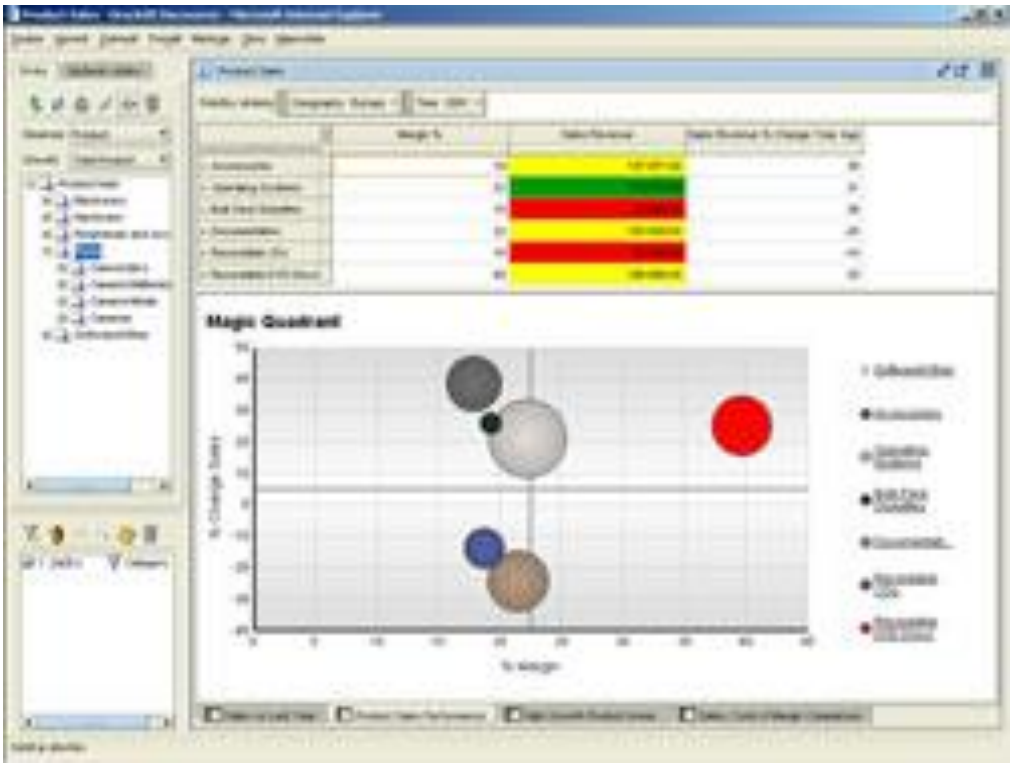


Abbildung 2: Verkaufsanalyse in der Oracle Business Intelligence – Discoverer

11.4. Analytische Instrumente

Bis jetzt wurden Wege und Technologien der Datenspeicherung beschrieben. Die Arbeit mit diesen – sprich die Durchführung von Analysen – erfordert geeignete Werkzeuge. Die Auswahl der richtigen analytischen Werkzeuge ist der Schlüssel zur effektiven Sammlung von Informationen. Es ist dabei besonders wichtig, die Expertise von Analytikern zu berücksichtigen, welche bereits mit den ihnen zur Verfügung stehenden Ressourcen arbeiten. Sehr häufig werden in diesem Kontext Werkzeuge aus der Microsoft Office-Familie verwendet. Aus diesem Grund ist es wichtig, dass die ausgewählten Lösungen im Zusammenspiel mit den bisher genutzten Werkzeugen funktionieren. Dies führt zu geringeren Kosten, wenn bei der Arbeit mit Informationen ein effizienterer Weg eingeschlagen wird. Gleichzeitig müssen der sichere Zugang zu Informationen, die automatisierte Verteilung von Berichten, die qualitativ hochwertige, grafische aufbereitete Datenausgaben für Präsentationen sowie bei Bedarf die Integration in das Unternehmensintranet sichergestellt werden.

Die Hauptakteure sind

Bei der ganzen Flut an Technologie darf man nicht auf die Menschen vergessen. Ein paar der wertvollsten Mitarbeiter, die ein Unternehmen gewinnen kann, sind erfahrene Analytiker. Nur solche erfahrenen Analytiker sind in der Lage, aus den Unmengen an Daten das herauszulesen, worauf es ankommt und zielgerichtete Empfehlungen für das Management zu formulieren.

11.5. Der OLAP-Würfel

Der OLAP-Würfel ist eine Methode, Daten zu organisieren, welche über eine zweidimensionale Tabellenanordnung hinausgehen, weshalb jede Datendimension auf einer Achse des Würfels gespeichert wird. Damit werden ein paar der Einschränkungen relationaler Datenbanken überwunden.

Das Anordnen von Daten in Würfelvektoren ermöglicht den Zugriff darauf aus verschiedenen Perspektiven (Dimensionen), aber auf dieselbe Weise. Die ausgeführte Systemleistung macht es schwierig, viele RDBMS-Tabellen miteinander zu kombinieren. Allerdings erlaubt die physische Datenspeicherung in Würfeln kein schnelles Bearbeiten, weshalb der gesamte Würfel wieder aufbereitet werden muss. Der Würfel besteht aus Werten, welche in Dimensionen kategorisiert sind. Die Struktur wird mithilfe relationaler Tabellen in ein Sterndiagramm oder Schneeflocken-Schema implementiert. Es handelt sich in der Regel um eine Eltern-Kind-Struktur, wo die Eltern-Elemente die Verdichtung des Ursprungs darstellen und sie selbst in ihren Eltern-Elementen kumuliert werden können.

11.6. Grundlegende Datenwürfeloperationen

Da sie aus analytischer Sicht notwendig sind, werden die nachfolgenden Datenwürfeloperationen durchgeführt, um die Verarbeitung und Projektion von Daten sowie deren Verständnis zu erleichtern:

Slice a Cube: Begrenzung einer oder mehrerer Dimensionen auf eine Untermenge eines Elements.

Cubming: Begrenzung einer oder mehrerer Dimensionen auf eine Untermenge zweier oder mehrerer Elemente.

Roll up and drill down: Nach oben und nach unten durch die Datenhierarchie navigieren.

Pivot: Rotieren des Würfels, um Datenbeziehungen aus verschiedenen Blickwinkeln zu betrachten.

Aggregation: Verdichtung gemäß der Beziehungen, welche von Formeln vorgegeben wurden.

12. SPEZIFIKA VON DATENBANKSYSTEMEN – TECHNOLOGIEN FÜR DEN ZUGRIFF AUF DATENBANKEN – GEOGRAPHISCHE INFORMATIONSSYSTEME

Eine Datenbank ist ein Dateisystem mit einer festgelegten Datensatzstruktur. Diese Dateien sind mithilfe von Schlüsseln miteinander verbunden. Im weiteren Sinne enthält eine Datenbank Software-Ressourcen, welche die Manipulation von und den Zugriff auf gespeicherte Daten ermöglichen. Diese Software wird in Tschechiens technischer Literatur als Datenbank-Managementsystem (DBMS) bezeichnet. Für gewöhnlich wird die Bezeichnung Datenbank – abhängig vom jeweiligen Kontext – entweder mit gespeicherten Daten oder mit der Software (DBMS) gleichgesetzt.

12.1. Überblick über den Daten Zugriff in der ASP.NET-Technologie

Visual Studio 2010
Web-Anwendungen greifen für gewöhnlich auf Datenquellen zu, um dort dynamische Daten zu speichern bzw. Diese von dort abzurufen. Man kann einen Datenzugriffscode schreiben, indem man die Klassen im System.Data Namensraum (in der Regel als ADO.NET bezeichnet) und im System.Xml Namensraum verwendet. Dieser Zugang war in den Vorgänger-Versionen von ASP.NET üblich.

Darüber hinaus erlaubt es die ASP.NET-Technologie, Datenverknüpfungen deklarierend zu erklären. In den meisten gebräuchlichen Szenarios ist kein Code erforderlich. Dies können die folgenden sein:

- Auswahl und Anzeige von Daten
- Sortieren und Paginieren von Daten sowie deren Aufnahme in den Cache-Speicher
- Aktualisieren, Einfügen und Löschen von Daten
- Datenfilterung mithilfe von Laufzeit-Parametern
- Erstellen von Master-Detail-Szenarios mithilfe von Parametern

ASP.NET enthält einige Arten von Server-Steuerungen, welche Teil des deklarativen Datenbindungs-Modells sind. Dazu gehören Datenquellensteuerungen, Datenbindungssteuerungen und eine erweiterte Abfragesteuerung. Diese Steuerungen bewältigen die grundlegenden Aufgaben, welche von einem unausgereiften Seitenmodell benötigt werden, um Daten auf einer ASP.NET-Webseite anzuzeigen und zu aktualisieren. Steuerungen

ermöglichen es, der Webseite eine Datenbindungs-funktionalität hinzuzufügen, ohne die Lebenszyklusdetails der Seite verstehen zu müssen.

13. DATENBANKOBJEKTE

Der Begriff „Datenbank“ wird oft vereinfacht und als Synonym für ein Datenbanksystem (Datenbank-Maschine) oder ein Datenbank-Managementsystem gebraucht. Sie enthält nicht nur Kalkulations-tabellen, welche nur eine von vielen sogenannten Datenbank-Objekten (manchmal auch Datenbank-Entitäten) sind. Erweiterte Datenbankensysteme enthalten:

- **Ansichten oder Aussagen:** SQL-Befehle, die im Datenbanksystem bestimmt und gespeichert sind. Man kann diese (durch Anwendung des SELECT-Statements) auch in anderen Tabellen aufrufen.

- I**ndizes für jede Tabelle:** Schlüssel werden oberhalb einzelner Tabellenspalten definiert (ein Schlüssel kann mehrere enthalten). Sie haben die Funktion, doppelte Einträge in Spalten der Datensätze zu behalten, welche über LUT-Tabellen (Nachschlagetabellen) für die Volltext-Suche definiert wurden.

- **Trigger:** Mechanismus für einzelne Zeilen einer Tabelle (oder die Tabelle selbst), welcher nach dem Verändern, Löschen oder Hinzufügen einer Zeile, dem Löschen einer Tabelle und einer vorprogrammierten Maßnahme abgerufen wird (zB zur Überprüfung der Datenintegrität).

- **Benutzerdefinierte Verfahren und Funktionen:** Einige Datenbankmaschinen unterstützen die Speicherung bestimmter Code-Teile, welche eine bestimmte Reihenfolge von Befehlen (Verfahren) in einer Datenbank unter Verwendung vorgegebener Tabellen ausführen oder Ergebnisse zurückgeben (Benutzerfunktionen). Sie können Parameter haben, die sich im Wesentlichen in Eingabe (IN), Ausgabe (OUT) sowie Ein- und Ausgabe (INOUT) gliedern.

- **Ereignisse, auch („cash“):** Hierbei handelt es sich um Verfahren, welche an einem bestimmten (vom Benutzer vorgegebenen) Datum zu einer bestimmten Zeit bzw. wiederholt inner-halb einer festgelegten Periode ausgelöst werden. Sie dienen zur Wartung, temporären Datenschmierung oder zur Überprüfung der Verweisintegrität.

- **Formulare:** Einige Datenbanksysteme wie Microsoft Acces erlauben es Benutzern, Eingabe-formulare zu erstellen, die optisch zur Eingabe einladen. So kann ein Benutzer zB das Layout jedes Eingabefelds einer vorgegebenen Tabelle, Etikette usw. festlegen.

- **Berichte:** Ähnlich wie Berichtsformulare erlauben sie es den Benutzern, das Layout für die Felder vorgegebener Tabellen festzulegen, welchen aktuelle Werte hinzugefügt werden. Sie werden zur Datenausgabe (Druckversion, Präsentation oder

einfache Ansicht) verwendet. Aufstellungen können ergänzt werden, zB durch Filter, welche nur die gewünschten Datensätze herausfiltern.

- **Benutzerbefugnisse:** Bessere Datenbanksystemen stellen in der Regel Optionen zur gestaffelten Einschränkung individueller Zugriffsrechte auf Datenbankobjekte zur Verfügung. Es gibt dutzende Möglichkeiten detaillierter Einschränkungen, welche bis hin zur Festlegung einzelner Befehlsarten geht, die ein Benutzer verwenden darf.

- **Partitionierung:** Eine Möglichkeit, die Daten in einer Tabelle auf mehrere Festplatten aufzuteilen und dadurch die Last zu reduzieren, die sie zu tragen hat.

- **Prozesse:** Datenbankmaschinen können einen Überblick über die Prozesse geben, welche deren Dienste gerade verwenden.

- **Variable Einstellungen:** Typischerweise dutzende Variablen, die sich neu gestalten lassen und dadurch das Verhalten und die Leistung der Datenbankmaschine selbst beeinflussen.

- **Kollation:** MySQL bietet fortgeschrittene Optionen zum Erstellen einiger Dutzend Zeichen-Sets und Vergleiche, welche sich unter dem Sammelbegriff Kollation zusammenfassen las-sen. Solche Kollationen können auf der Ebene einzelner Textspalten, ganzer Tabellen oder ganzer Datenbanken eingerichtet werden (mit kaskadenförmiger Vererbung). Eine Kollation beeinflusst auch die Sortierung, zB der Wert utf8_czech_ci stellt sicher, dass korrekt nach dem Tschechischen Alphabet (sprich inklusive aller diakritischer Zeichen und CH) sortiert wird.

- **Visuelles E-R Schema:** (im MySQL INFORMATION.SCHEMA). Visuelle Darstellung von Beziehungen (s) voneinander abhängiger Felder (Fremdschlüssel) zwischen Tabellen.

13.1. Grundlegende Konzepte der Datenbanktechnologie

Einführung die Datenbanktechnologien

Mit dem Aufkommen der Microcomputer und ihrem Einzug in das tägliche Leben scheint ein gesteigertes Interesse an rechnergenerierten Daten einhergegangen zu sein. Dieses bezieht sich nicht nur auf Berechnungen, sondern auch auf die Produktion von Wissen. Diese Erklärung lässt sich sehr gut nachvollziehen, wenn man bedenkt, dass sogar die anspruchsvollsten Computerspiele mit der Zeit die Spieler ermüden und dass Programmieren im Wesentlichen ein Weg ist, die Lösungsart für ein System aus zwei Gleichungen mit zwei Unbekannten zu berechnen.

Jemand möchte sich gerne Informationen organisieren bzw. nach Informationen suchen, die er häufig verwendet und während der Verwendung verändert. Geeignete Mittel zur Speicherung solcher Informationen in Datenform vorausgesetzt, wäre dieser jemand gewillt, im Rahmen einiger Programmabende zu versuchen, relevante Daten zu produzieren, zu kaufen oder mit einem anderen vergleichbaren Datenenthusiasten auszutauschen. Ein Beispiel könnte eine Bibliothek, ein Büro, eine Flugticketbearbeitung, ein Stadtmuseum, ein Krankenhaus oder ein Geschäft sein.

Das fertige Objekt der beschriebenen Art kann dann als Informationssystem (IS) bezeichnet werden. Dieses IS ist eine Menge an Elementen in den wechselseitigen Beziehungen zwischen Information und Verfahren (Informationsverfahren), welche Daten verarbeiten und das Kommunizieren von Informationen zwischen Elementen sicherstellen. Aus Sicht eines Benutzers sollte das IS Konstruktionsmerkmale haben, welche die Manipulation von Daten (Dateneingabe, Abfrageformulierung, Einsatz von Anwendungsprogrammen) einfacher macht und gleichzeitig sicherstellt, dass die zur Verfügung gestellten Funktionen stark und schnell genug sind. Ein flexibles „leeres“ IS, welches heutzutage einem Benutzer bereitgestellt wird, kann sehr schnell sein, aber über einen ausreichen-den Wissensstand verfügen.

Das Ziel wird es sein, die sogenannte Datenbankverarbeitungstechnologie zu demonstrieren. Eine Datenbanktechnologie entspricht einer Menge von Konzepten, Ressourcen und Methoden, die verwendet werden, um Informationssysteme (IS) zu erschaffen. Auf der Grundstufe lässt sich die IS-Architektur mit der Datenbank wie folgt abbilden: Daten sind in einer Datenbank (DB) organisiert und werden von einem Programmpaket, welches Datenbank-Managementsystem (DBMS) genannt wird, verwaltet. Die Datenbank und das Datenbank-Managementsystem bilden zusammen das sogenannte Datenbanksystem (DBS). In Übereinstimmung mit der Terminologie lässt es sich auch wie folgt schreiben: DBS = DB + SBBD. Das IS verwendet die vom DBS bereitgestellten Daten entweder direkt oder verarbeitet diese mithilfe von Anwendungsprogrammen.

13.2. Geografische Informationssysteme

Ein geografisches Informationssystem (GIS) erlaubt die Speicherung, Verwaltung und Analyse von Gebietsdaten.

Die meisten realen Objekte und Phänomene treten auf irgendeiner Stelle der Erdoberfläche auf (z.B. Bäume, Häuser, Flüsse etc.) oder haben eine Beziehung zu einem bestimmten Platz auf der Erdober-fläche (ein Staatsbürger hat einen permanenten Wohnsitz, das Produkt wurde in einer bestimmten Fabrik hergestellt). Gleichzeitig treten diese Objekte zusammen mit anderen Objekten in diesem Raum auf, wo sie miteinander interagieren. Aus diesem Grund ist die Kenntnis eines Orts und der räumlichen Beziehungen zwischen Objekten sehr wichtig und kann eine wichtige Rolle im Hinblick auf zahlreiche menschliche

Aktivitäten spielen. Diese können von der Gestaltung eines Atomkraftwerks bis hin zur Gestaltung eines geschäftlichen Netzwerks und dessen Erfolgsauswertung reichen.

In der Praxis bedeutet dies, dass diese beiden Informationen, sprich die tatsächlichen Daten des Objekts und dessen Standort, gleichzeitig auf dem Computer gespeichert werden müssen. Diese Art von Daten bezeichnet man als geografische (oder räumliche) Daten. Ein Computersystem, welches das Speichern und Verwenden solcher Daten ermöglicht, nennt man geografisches Informationssystem.

Grundlegende Komponenten eines geografischen Informationssystems

Vít Voženílek beschreibt diese in seinem Buch Geographical Information System I. The Concept, History, Basic Components wie folgt:

- Hardware
- Software
- Daten
- Bedienungspersonal

Um Systeme effizient bedienen zu können, ist deren Balance essentiell. Die Hardware, oder auch technische Ausrüstung, bildet die technische Grundlage geografischer Informationssysteme. Die Soft-ware ist eine Menge von Programmen, welche alle Systemoperationen ausführen. Die Daten sind das Schlüsselelement jedes geografischen Informationssystems. GIS ist aus der Perspektive organisa-tioneller Strukturen ein wahrhaftiges System. Seine Tätigkeit ist die Sammlung von Aktivitäten, welche die Funktionen des Systems zur Verfügung stellen.

13.3. Geografische Daten

Dimensionen geografischer Objekte

Die grundlegende Unterteilung geografischer Objekte erfolgt nach der Anzahl der Dimensionen. Reale Objekte auf der Erdoberfläche sind immer dreidimensional. Um in das GIS übertragen werden zu können, müssen sie zuvor jedoch auf ein erforderliches Maß abstrahiert werden.

- 0D Geo-Objekte: dimensionslose Objekte, Punkte werden nur durch ihren Standort definiert. Ein Beispiel hierfür ist eine Bushaltestelle in einem GIS, welches Transport modelliert oder der GSM-Transmitter des GIS eines Mobilfunkanbieters, welcher Signalabdeckungen modelliert.

- 1D Geo-Objekte: eindimensionales Objekt, mittels Strichen abgebildete Streckenabschnitte (Kanten, Linien), begrenzte Länge und keine Fläche. Beispiele hierfür sind Straßen und Flüsse.

- 2D Geo-Objekte: zweidimensional, Polygone, mit begrenztem Umfang, begrenzter Oberfläche.

- 3D Geo-Objekte: dreidimensionale Objekte, geometrischer Körper. In einem GIS werden sie nur in Ausnahmefällen verwendet. Die dritte Dimension wird im GIS am häufigsten mithilfe eines sogenannten Terrainmodels (DMT, DEM) an der Oberfläche modelliert. Hierzu wird es mit topologischen Oberflächen (2.5D) verknüpft.

Die Grundlage für jedes GIS bilden geografische Daten. Geografische Daten sind Informationen über die Erdoberfläche und die darauf befindlichen Objekte. Diese Daten können in Form einer bestimmten Informationsschicht dargestellt werden, welche auch Thema genannt wird. Jedes Thema repräsentiert ein bestimmtes Element (zB Straßen, Seen, Städte, etc.) Geografische Daten können in einem GIS mittels der folgenden zwei Basismodelle organisiert werden.

13.4. Datensammlung

Das Erhalten von Daten und das Verschieben von Informationen in das System erfordert eine Menge Zeit. Es gibt eine Anzahl von Methoden, die verwendet werden, um GIS-Daten einzugeben, welche in digitaler Form gespeichert sind. Ausgangsdaten, welche auf Papier ausgedruckt oder als PET-Film vorliegen, können digitalisiert oder eingescannt werden, um digitale Daten zu produzieren. Der Digitalisierer produziert Vektordaten als Punkte, Linien und Polygone. Auch das Zerlegen von Karten in das Format eines Rasters, der weiterverarbeitet werden könnte, ist geeignet, um Vektordaten zu produzieren.
Gesammelte Daten erhalten dank einer Koordinationsgeometrie genannten Methode direkten Zugang zum GIS. Standorte des Globalen Navigationssatellitensystem (GNSS) können auch gesammelt und anschließend in das GIS importiert werden. Aktuell gibt es einen Trend in der Daten-sammlung, der es Benutzern erlaubt, Computer im Feld zu benutzen, welche die die Möglichkeit bieten, Daten in Echtzeit über eine Internetverbindung oder sogar ohne Internetverbindung zu bearbeiten. Dadurch besteht kein Bedarf mehr daran, Bürodaten zu importieren und zu aktualisieren, nach diese in Feldarbeit gesammelt worden sind.

Dies schließt die Fähigkeit ein, Standorte miteinzubeziehen, welche mittels eines Laserentfernungs-messers gewonnen wurden. Neue Technologien ermöglichen es Benutzern, sowohl Karten zu erstellen als auch Feldanalysen durchzuführen, Projekte effizienter zu gestalten und Karten genauer zu zeichnen. Aus der Ferne erhobene Daten spielen auch eine wichtige Rolle bei der Datensammlung und werden mittels Sensoren gesammelt, die mit einer Plattform verbunden sind. Diese Sensoren können Kameras, digitale Scanner

und LIDAR sein, während es sich bei den Plattformen zumeist um Luftfahrzeuge und Satelliten handelt. Mitte der 1990er-Jahre machten in England Hybrid-Ballone, welche Helikites genannte wurden, das erste Mal Werbung für den Einsatz kompakter Luftkameras als Geoinformationssysteme. Flugzeugsoftware mit einer Messpräzision von 0,4 mm wurde verwendet, um Fotografien mit Bodenmessungen zu verknüpfen. Helikites sind günstig und sammeln mehr adäquate Daten als Flugzeuge. Ein Produkt der jüngsten Entwicklungen sind unbemannte Miniaturdronen. So wurde z.B. die Aeryon Scout Drone eingesetzt, um in zwölf Minuten ein 20 Hektar großes Grundstück mit einer Musterdichte von 2,54 cm zu kartographieren.

13.5. Mehrdimensionaler Würfel

Die elementare Baueinheit in einer mehrdimensionalen Datenbank sind Würfel (Würfel, Daten-würfel, mehrdimensionale Würfel, Hyperwürfel). Ein Würfel in einer mehrdimensionalen Datenbank besteht aus einer Menge von Dimensionen und Messgrößen.

Die Dimensionen eines Würfels sind die Kategorien, anhand welcher Daten kumuliert und analysiert werden sollen. Dimensionen ergeben sich aus relationalen Datenbanktabellen. Typische Dimensionen in multidimensionalen Datenbanken sind Zeit, Standort und Produkt. Dimensionen können aus einer Anzahl an Ebenen bestehen, welche die Daten weiter aufbereiten.

Bei Würfelmessgrößen handelt es sich um die quantitativen Daten, die es es zu analysieren gilt. Dimensionen leiten sich aus relationalen Datenbanktabellen ab. Typische Maßzahlen sind Verkäufe, Kosten, Preise, aber fast jede quantitative Zahl kann eine Messgröße eines multidimensionalen Würfels sein.

13.6. Daten in der OLAP-Datenbank speichern

Die Quellendaten aus der relationalen Datenbank können in einer dieser drei grundlegenden Methoden gespeichert werden:

Datenschranken enthalten kumulierte Daten, welche die Grundlage für multidimensionale Analysen bilden. Umfangreiche multidimensionale Anwendungen (zB informetry) enthalten Daten, welche sich auf verschiedene Kontexte beziehen, aber einige Dimensionen können geteilt werden. Es ist daher eher verständlich, einen einzelnen Datenwürfel für einen Kontext zu erstellen als einen Datenwürfel zu verwenden, welcher alle Kontexte umfasst. Dieser Datenwürfel besteht nur aus jenen dimensionalen Attributen, welche all ihre numerischen Attribute miteinander teilen. Das bedeutet, dass dimensionale Attribute die Grundlage eines Datenwürfels bilden. Wenn man Informationen aus mehreren Datenschranken herausbekommen möchte, können diese Attribute auf der Ebene einer oder mehrerer gemeinsamer Dimensionen verknüpft sein. Die Navigation zwischen Datenwürfeln nimmt keine Rücksicht auf den Benutzer.

Dimensionstabellen beschreiben spezifische Dimensionseigenschaften. Für jedes dimensionale Attribut gibt es maximal eine Dimensionstabelle. Anders als das ursprüngliche Sternmodell, kommen die Dimensionswürfel in diesem Modell ohne Dimensionstabellen aus. Auf schematischer Ebene besteht eine Dimensionstabelle aus Attributsnamen und auf der Beispielebene basiert es auf den Werten dieser Attribute.

In Informationen müssen kumulierte Daten oft auf Basis komplexer Kriterien analysiert werden, welche von den Eigenschaften der Dimensionen abhängen. Um diese Kriterien spezifizieren zu können, sollten die Informationen in mehreren Dimensionstabellen verwendet werden. Im Anschluss werden semantische Beziehungen zwischen Dimensionstabellen erstellt, welche auf gemeinsamen Werten einiger Attribute in den Dimensionstabellen basieren.