Zusammenfassung M104

Datenmodell implementieren

2008-11-12

Emanuel Duss

Über

Autor	Emanuel Duss
Erstellt	2008-07-25
Bearbeitet	2008-11-12
Heute	2008-11-12
Bearbeitungszeit	02:33:36
Lehrjahr des Moduls	1. Lehjahr 2006/2007
Pfad	/ home/emanuel/Daten/Lehre/Zwischenprüfungen/Zusammenfassungen_von_mir/ M104/M104_Zusammenfassung.odt
CC-Lizenz	 Creative Commons Namensnennung-Keine kommerzielle Nutzung-Weitergabe unter gleichen Bedingungen 2.5 Schweiz http://creativecommons.org/licenses/by-nc-sa/2.5/ch/
Powered by	

Bearbeitungsprotokoll

Datum	Änderung(en)
2008-07-25	Erstellt
2008-07-28	Ergänzungen
2008-07-29	Mit Stichwortverzeichnis begonnen
2008-08-03	Änderungen vorgenommen, Stichwortverzeichnis fertiggestellt
2008-08-15	Letzte Änderungen
2008-08-19	Fertigstellung der Zusammenfassung

Inhaltsverzeichnis

Modulbaukasten

Modulidentifikation

Modulnummer	104
Titel	Datenmodell implementieren
Kompetenz	Ein Datenmodell (Entity Relationship Model, ERM) mit einer Datenbanksoftware implementieren.
Handlungsziele	1. Datenmodell (ERM) interpretieren und unterschiedliche Kardinalitäten der Beziehungen erkennen. 2. Entitätsmengen (Gegenstände, Personen usw.), Attribute und Beziehungen darstellen. 3. Datenmodell mit einer Datenbanksoftware implementieren. 4. Mit einer Datenbanksoftware Benutzerschnittstellen zur Erfassung, Veränderung und Auswertung von Daten in einer Datenbank erstellen. 5. Anwendung mit Testdaten überprüfen und die Ergebnisse protokollieren.
Kompetenzfeld	Data Management
Objekt	Single User Datenbank mit ca. 5 Tabellen.
Niveau	1
Voraussetzungen	Daten charakterisieren, aufbereiten und auswerten
Anzahl Lektionen	40
Anerkennung	Eidg. Fähigkeitszeugnis Informatiker/Informatikerin
Modulversion	2.0
MBK Release	R3
Harmonisiert am	04.10.2004; 28.02.2005

Handlungsnotwendige Kenntnisse

Handlungsnotwendige Kenntnisse beschreiben Wissenselemente, die das Erreichen einzelner Handlungsziele eines Moduls unterstützen. Die Beschreibung dient zur Orientierung und hat empfehlenden Charakter. Die Konkretisierung der Lernziele und des Lernwegs für den Kompetenzerwerb sind Sache der Bildungsanbieter.

Modulnummer	104
Titel	Datenmodell implementieren
Kompetenzfeld	Data Management
Modulversion	2.0
MBK Release	R3

Handlungsziel	Handlungsnotwendige Kenntnisse
1.	1. Kennt die wichtigsten Abklärungen (Systemaufgaben, Aufgabenergebnisse etc.) zur systematischen Abbildung des Ausschnitts aus der Realität, den eine Anwendung berücksichtigen muss und kann erläutern, worauf bei diesen Abklärungen zu achten ist, um vollständige Informationen für diese Anwendung zu erhalten.
2.	1. Kennt die Begriffe 'Entitätsmenge', 'Attribute' und 'Beziehungen' und kann an Beispielen erläutern welchen Beitrag diese zur strukturierten Darstellung von Informationen aus der realen Welt leisten. 2. Kennt die unterschiedlichen Beziehungstypen (Kardinalitäten, Assoziationen) von Informationen und kann anhand von Beispielen aus dem Alltag aufzeigen, welche Abhängigkeiten zwischen Informationen diesen zugeordnet sind. 3. Kann für die wichtigsten Definitionselemente eines Entity Relationship-Modells (Entität, Entitätsmenge, Attribut, Primärschlüssel, Fremdschlüssel, Assoziationen) aufzeigen, welche Sachverhalte aus der realen Welt damit beschrieben werden und wie diese in einem Entity Relationship Diagram abgebildet werden.
3.	1. Kennt die Befehle eines Datenbankprogramms zur Umsetzung eines Datenmodells in ein logisches Datenbankschema und kann aufzeigen, welche Definitionselemente (Entität, Attribut etc.) eines ERM damit umgesetzt werden.
4.	1. Kennt GUI-Komponenten und Formularfelder, die Datenbankprogramme für die Definition von Benutzerschnittstellen zur Erfassung, Veränderung sowie Auswertung von Daten zu Verfügung stellen und kann erläutern, für welche Typen von Daten diese eingesetzt werden. 2. Kennt die wichtigsten Regeln, die bei der ergonomischen Gestaltung einer Benutzeroberfläche für eine Datenbank hinsichtlich Abfolge und Bezeichnung der Informationen einzuhalten sind und kann erläutern, welche Vorteile die Befolgung dieser Regeln für den Benutzer bietet. 3. Kennt die wichtigsten Befehle einer Datenmanipulations- und Abfragesprache (DML, SQL) zur Manipulation, Selektion und Auswertung von Datenbeständen. 4. Weiss um die Schutzwürdigkeit personenbezogener Daten und kann erläutern, welche prinzipiellen Möglichkeiten ein Datenbankprogramm zur Einschränkung des Zugriffs und der Manipulationen schützenswerter Daten zur Verfügung stellt.
5.	1. Kennt die wichtigsten Testkriterien für eine Datenbank (Datenintegrität, Datensicherheit, Datenschutz, Datenkonsistenz) und kann aufzeigen, wie Testfälle für die Überprüfung dieser Kriterien aufgebaut sein müssen.

1 Datenbankgrundlagen

1.1 Einleitung

- Datenbanken spielen beim Einsatz von Computer häufig eine zentrale Rolle
- Lösten Karteikarten aus folgenden Gründen ab: Flexibilität, Schnelligkeit, Statistiken sind schnell erstellt, Erfassung über einen Bildschirm, keine Redundanzen, keine Schreibfehler, Durchsuchbar
- Für Speicherung grosser Datenmengen: Personalverwaltung, Lagerwirtschaft, Bestell- und Rechnungswesen
- Daten werden nach natürlichen Zusammenhängen gespeichert. Ausschnitt aus der realen Welt.
- Mehrere Benutzer können gleichzeitig auf einer DB arbeiten.
- DBS (Datenbanksysteme): Daten werden in einer DB zusammengefasst.

Datenstrukturen

- Unstrukturierte Daten (Fliesstext)
  - Ein Mensch muss den Text lesen, um Infos zu gewinnen.
- Schwach strukturierte Daten (Tabelle; einfache Datensätze ohne Verknüpfung)
  - Auswertung ist erstellbar.
  - Elemente: Bezeichnng; Kolonnen, Attribute; Datensatz, Record, Tupel, Entity; Feld)
- Stark strukturierte Daten (Datenbank)
  - Es existieren Beziehungen untereinander
  - Auswertung ist erstellbar
  - Auch grosse Datenbanken möglich

DB-Konzept Argumente

- Überblick ist besser
- Fehler werden vermieden
- zwingt Benutzer zum Mitarbeiten
- Entscheidungsfreiheit (Wünsche) des. Anwenders
- Schrittweise mehr Verantwortung durch EDV-Spezialist

Skalenniveaus / Messen / Statistik

- Nominalskala: Ausprägung eines Merkmals ist entweder Gleich oder Verschieden. z.B. Verheiratet oder ledig.

- Ordinalskala: Zusätzlich kann eine Rangordnung zwischen verschiedenen Merkmalen erstellt werden. z.B. Grösser / kleiner / gleich
- Intervallskala: Zusätzlich wird der Abstand der Merkmalsausprägung gekennzeichnet. z.B. 10°C und 20°C
- Verhältnisskala: Zusätzlich wird ein Verhältnis angegeben. z.B. Die Strecke AB ist länger als CD.

	Nominal-skala	Ordinal-skala	Intervall-skala	Verhältnis-skala
Identität	+	+	+	+
Rangfolge		+	+	+
Abstände			+	+
Verhältnis				+

Datenbanksysteme

- **Generation 0: Programmeigene Datenverarbeitung**
  - 50er: Magnetbänder
  - 60er: Magnetplatten
- **Generation 1: 70er: Hierarchische DB**: baumartige Struktur, geordnete Menge
- **Generation 2: 70er: Netzwerk DB**: Zugang zu jedem anderen Knoten
- **Generation 3: 80er: Relationale DB**: Organisation mit Tabellen
- **Generation 4: 90er: Objektorientierte DB**: Unsere Welt mit ihren Eigenschaften und ihrem Verhalten nachbilden. (OODBS)
- **Generation 5: heute: Objektrelationale DB** (ORDBS). Ohne Nachteile relationaler DBS, aber mit Vorteilen der Speicherung komplexer Objekte der OODBS

> SQL soll als einheitliche Sprache beibehalten werden

Physikalische Architektur

- **Zentralisiertes DBS**: Alles auf einem Zentralrechner
- **Verteiltes DBS**: Teil-Datenbanken. Im Netzwerk verteilt.
- **Client-Server-DB**: Client stellt Anfrage und DB-Server gibt die entsprechenden Daten aus.
- **Paralleles DBS**: Es werden Datenbanksysteme verwendet, die auf Multiprozessorsystemen oder Parallelrechnern laufen. Daher ist sie leistungsfähiger.

Datenbanksysteme

Ein Datenbanksystem besteht aus:

- Datenbankmanagementsystem (DBMS)

- Datenbank (DB)

DBMS: Datenbankmanagementsystem

- Verwaltet die Datenbanken. Zentrale Steuerung.
  - Anwendungsprogramme greifen nicht direkt auf Daten zu, sondern stellen Anfragen an DBMS
  - Ermöglicht: Anlegen von Datenbanken, speichern, ändern, löschen, abfragen von Daten und die Benutzerverwaltung.
- DBMS wird durch eine Sprache angesprochen. Oft ist dies SQL (Structured Query Language)

1.2 Datenbankentwurf

Entwurfsphasen

1. Daten analysieren (Analyse der Anforderung): Was habe ich alles für Daten?
2. Datenbank-Plan zeichnen (konzeptioneller und logischer Entwurf): ERD erstellen. Genügend Zeit planen, da dies die Qualität der Datenbank beeinflusst. Wichtig ist die Vollständigkeit, Korrektheit, Lesbarkeit, Minimalität und Modifizierbarkeit.
3. Datenbank-Struktur nach Fehlern untersuchen: Redundanzen beseitigen
4. Datenbank erstellen
5. Daten in Datenbank eingeben
6. Datenbank-Lösung überprüfen: Kontrolle ob gültige Daten angenommen und ungültige abgewiesen werden.
7. Lösung dokumentieren

5 Gründe für die Verwendung eines Phasenkonzeptes

1. Besserer Überblick
2. Fehler werden vermieden
3. Zwingt Benutzer zum Mitarbeiten
4. Entschedungsfreiheit (Wünsche) des Anwenders
5. Schittweise mehr Verantwortung durch EDV-Spezialist

DB-Lebenszyklus

Das sind die Phasen, die eine Datenbank durchläuft.

- Anforderungsanalyse: Was wird gespeichert und wie wird bearbeitet?
- konzeptioneller Entwurf: Für welches DBS wird die Datenbank gemacht; ERD wird entwickelt.
- logischer Entwurf: DB-Schema wird normalisiert und Redundanzen verhindert. Festlegung des DBS.

- Entwurf der Verteilung im Netz: Erweiterungen und gegebenenfalls Änderungen werden am ERM durchgeführt.
- Physischer Entwurf/Implementierung: Definition des internen Schemas.
- Test und Validation
- Anwendung und Wartung.

3-Ebenen-Modell

- externe Ebene (Benutzerdefinierte Schichten)
- konzeptionelle Ebene (logische Gesamtschicht)
- interne Ebene (physische Beschreibung, Datenorganisation im Speicher)

1.2.1 Datenbankanalyse

- Umfang der zu speichernden Daten ermitteln (Objekte definieren) (z.B. Artikel, Kunden)
- Wesentliche Merkmale der vorhandenen Objekte definieren (z.B. Anrede, Vorname)
- Verbindungen ermitteln (z.B. Ein Kunde kauft einen oder mehrere Artikel)

Analyse

Im Rahmen der Datenanalyse sind folgende Aspekte wichtig:

- Herkunft der Daten: Woher können die benötigten Daten beschafft werden
- Form der Daten: In Welcher Form liegen die Daten vor? Können sie so verarbeitet werden?
- Qualität der Daten: Genügt die Qualität den Anforderungen?
- Vertrauenswürdigkeit der Daten: Sind die Daten vertrauenswürdig?
- Aktualität der Daten: Entspricht die Aktualität den Ansprüchen?

Auswertung

- Nach der Formatierung (z.B. Einlesung in Datenbank) werden die Daten ausgewertet. D.H. selektiert und mit anderen Daten kombiniert.
- Danach werden die Daten dargestellt.

1.3 Test und Dokumentation

Dokumentation

Zu einer Dokumentation gehört:

- Das Entity-Relationship-Diagramm (ERD)

- Die Beschreibung jeder Tabelle (mit dem Befehl SHOW FIELDS FROM…)
- Abbildung aller Bildschirmmasken
- Abbildung der Berichte
- Testprotokoll

Testing

- **Bei reinen Textfeldern** ist zu überprüfen, ob das Feld für den längsten möglichen Eintrag lang genug ist.
- **Bei Kombinationsfeldern** ist sicherzustellen, dass nur aufgelistete Werte eingegeben werden dürfen

Dateninkonsistenz

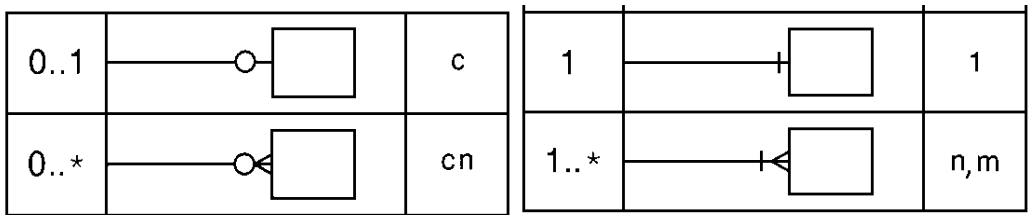
- Normalisierung des Datenbestandes
- Wahl des richtigen Datentyps bei der Definition der Tabellen
- Einbau von Kontrollmechanismen in der Bildschirmmaske
- Einsatz von grafischen Elementen in der Bildschirmmaske, die nur bestimmte Eingaben erlauben.

Bildschirmmasken

- **Elemente gruppieren**
  Bei der Platzierung darauf achten, dass Felder, die inhaltlich zusammengehören. auch zusammen erscheinen
- **Abstände und Ausrichtung**
  Abstände tragen zur besserer Lesbarkeit bei
- **Schrift**
  Serifenlose Schriften sind auf dem Bildschirm besser lesbar. GROSSBUCHSTABEN und *kursiv* eignet sich auch nicht für das menschliche Auge
- **Farben**
  zu viele Farben wirken für unser Auge ermüdend und für unser Gehirn verwirrend.
  Rot = Gefahr, Stopp, Verbot, Fehler
  Orange = Vorsicht, Achtung
  Grün = Keine Gefahr, Fluchtweg Weg frei, Sicherheit
  Menschen assoziieren Farben mit Gefühlen und Ereignissen
  ➔ grüne Fehlermeldung würde nicht bemerkt werden!!!
- **Weg des Tabulators**
  sollte nachvollziehbar sein (nicht suchen)
- **Eingabehilfen**
  Checkboxen, Optionsfelder, Kombinationsfelder

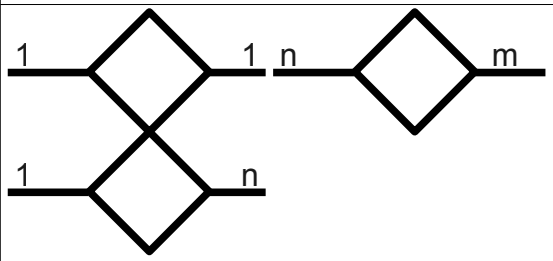
1.4 Datenbankplanung

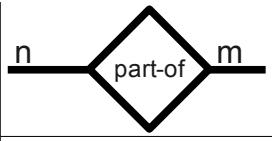
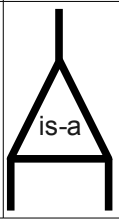
Vergleiche Glossar bei den Begriffen!

Entität	Ein einzelnes Objekt in der Datenbank. z.B. Personen, Artikel, etc. Eine Tabelle
Eintitätsmenge	Menge aller Objekte, die vom Typ her zusammengehören. Eine Tabelle.
Entitätstyp	Art der Objekte.
Attribut und Attributwert	Eingenschaften der Objekte
Domäne / Wertebereich	Menge von zulässigen Eigenschaftswerten für eine bestimmte Eigenschaft
NULL	Ein Attribut ohne Attributswert.
Primärschlüssel	Eindeutig, Laufend zuteilbar, kurz und bündig, ändert nie, nicht NULL, unterstrichen im ERD z.B. Kundennummer. Folgende Eigenschaften werden erwartet: Laufende Zuteilbarkeit (schnell zuteilbar), kürze und Prägnanz, Dauerhaftigkeit
Fremdschlüssel	Schlüssel, der die Verbindung zu einer anderen Tabelle herstellt. Assoziation = Verbindung.
Kardinalitäten	

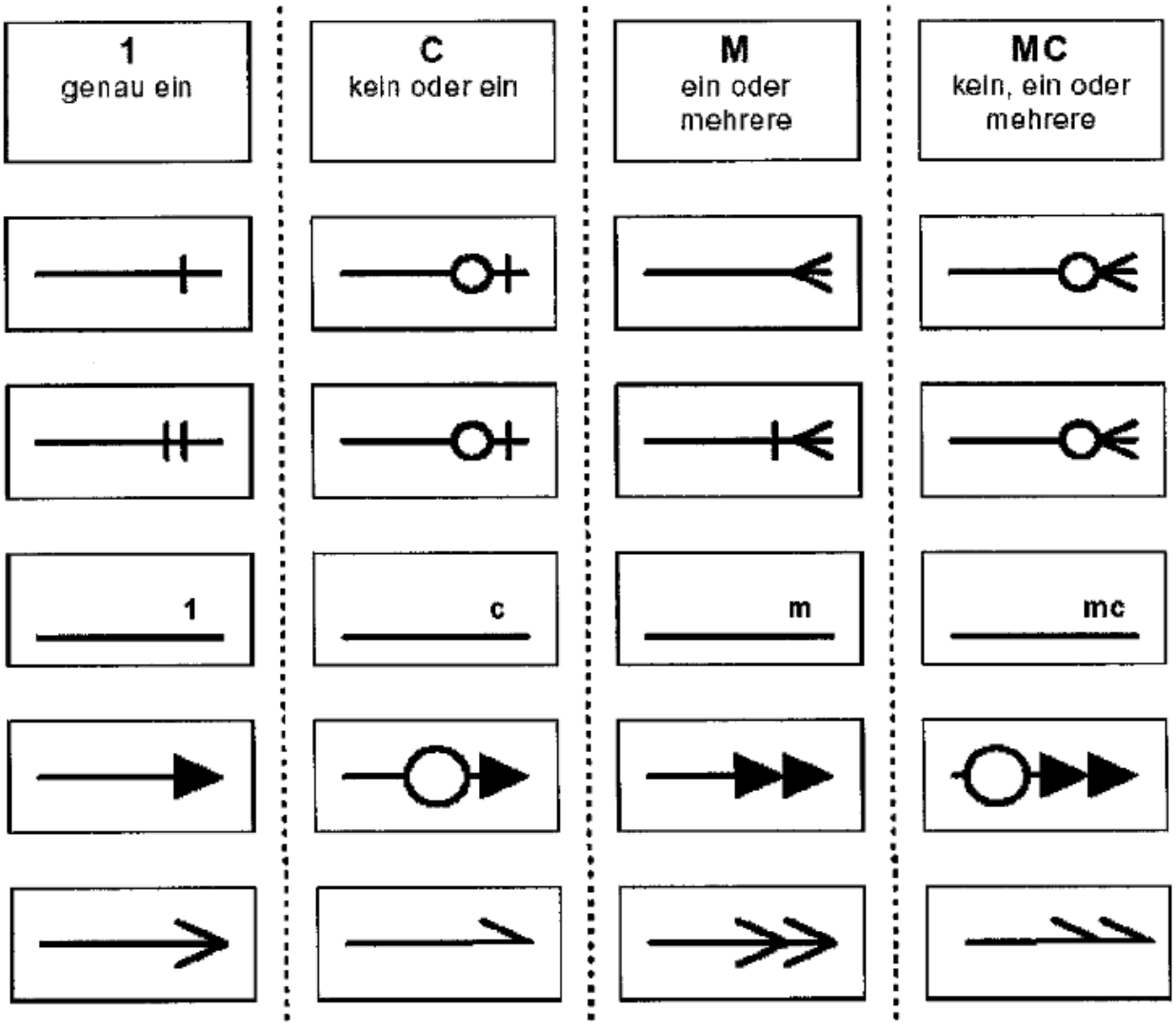
1.5 Das Entity-Relationship-Modell

1.5.1 ERD-Symbole

Entity-Typ		Dinge aus der realen Welt.
Attribut		Eigenschaften einer Entität. Wird mit Entität verbunden.
Primärschlüssel		Eindeutige Nummer einer Entität. Wird mit Entität verbunden.
Beziehungen		Ausdrückung von Abhängigkeiten In der Raute kann der Name des Beziehungstyps stehen. Beziehungen können durch Attribute beschrieben werden (z.B. „arbeitet an“ mit den Attributen Tätigkeit und Prozent) Verbindet zwei Entitäten.
Kardinalitäten	1 = genau 1 n oder m = 1 bis unendlich c = 0 oder 1 cn = 0 bis unendlich	Vergleiche oben die Darstellung bzw. unten.

Part-of-Beziehung		Aggregation. Ein Computer besteht z.B. Aus Laufwerken, Gehäuse, etc.
Is-a-Beziehung		Verallgemeinerung, Generalisierung

Gegenüberstellung der Notationsformen der Kardinalitäten



Abstraktionskonzepte

- Klassifikation
- Aggregation
- Generalisierung
- Assoziation
- Identifikation

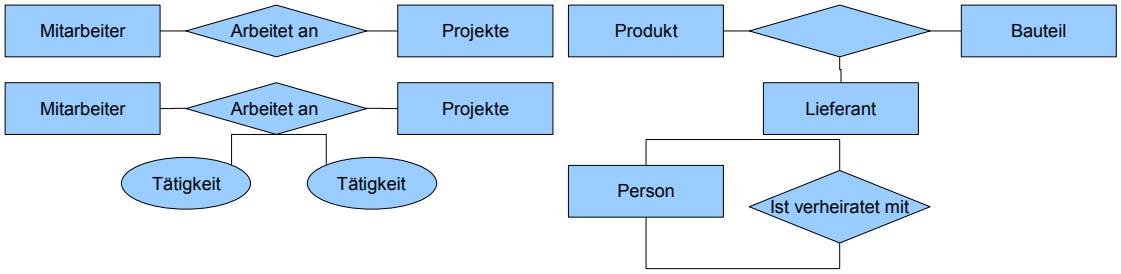
Funktionale und transitive Abhängigkeit

- Eine funktionale Abhängigkeit liegt vor, wenn es keine zwei Tupel geben kann, in denen für gleiche x-Werte verschiedene y-Werte auftreten können. Umgekehrt kann es aber für gleiche y-Werte verschiedene x-Werte geben (z.B. LieferNr → Lieferdatum).
- Eine transitive Abhängigkeit besteht, wenn ein Attribut a von einem Attribut b und ein Attribut b einem Attribut c funktional abhängig ist. In diesem Fall ist a transitiv von c abhängig (z.B. ArtikelNr → Lieferfirma → AnschriftLieferfirma).

1.5.2 Erstellen eines ERDs

Grundkonzept

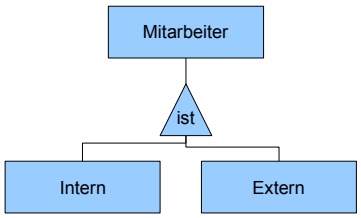
1. Die verschiedenen Entitätstypen (Klassen, Objekte) werden herausgesucht und in ein Rechteck geschrieben.
2. Zuweisen von Eigenschaften (Informationen, Attribute).
3. Ein Primärschlüssel **muss** für jedes Objekt existieren.
4. Objekte mittels Beziehungstypen verknüpfen. Dazu verwendet man die Raute und schreibt einen Text in die Raute.
   Beispiele:



5. Kardinalitäten der Beziehungstypen kennzeichnen.
   (1 = genau 1; n oder m = 1 bis unendlich; c = 0 oder 1; cn = 0 bis unendlich)
   Bei folgendem Beispiel fehlen die Eingenschaften der Objekte. Es geht nur darum, die Kardinalitäten zu zeigen:

Erweiterung des ERMs

- Generalisierung: Objektorientierte Ansätze der Vererbung fliessen ein: man sucht Untertypen. Der Obertyp vererbt den mehreren Untertypen seine Eigenschaften. Diese Generalisierung wird durch ein Dreieck dargestellt:



- Aggregation: Wird durch eine Raute in einem Rechteck dargestellt. In der Schule kam dies nie zum Zug.

1.6 Umgang mit mehrfach vorhandenen Daten

1.6.1 Eliminieren von Redundanzen

- Eine Redundanz ist das mehrfache Speichern von Informationen.
- Wir dürfen ein bestimmtes Attribut nur einmal in der DB verwenden. Wir müssen z.B. Nicht bei der Personen-Tabelle die Abt_Nr. Und die Abteilung hinschreiben.
- Wir erstellen eine zweite Tabelle für die Abteilungen und verweisen bei der Person mit der Abt_Nr auf die jeweilige Abteilungsnummer.
- Wenn man Redundanzen hat, ist die Datenbank inkonsistent.
- Anomalien beachten
- Datenbank in die 3. Normalform bringen

2 Das Relationale Datenmodell

2.1 Begriffe

2.1.1 Definition Relation

Eine Relation ist definiert durch...

- Einen eindeutigen Namen wie z.B. Kunde
- Mehrere Attribute (Spalten)
- Keine bis beliebig viele Tupel (Tabellenzeilen oder Datensätze)
- Einen einzigen Wert pro Attribut in einem Tupel (Tabellenzeile)
- Einen Primärschlüssel, bestehend aus einem oder mehreren Attributen, welche den jeweiligen Datenbankeintrag eindeutig definiert und dessen Wert sich während der Existenz dieses Datensatzes nie ändert.

2.1.2 M:N-Beziehungen

- Es muss immer eine zusätzliche Relation erstellt werden.

2.1.3 Relationale Beziehung

SQL ist eine relationale Sprache, mit welcher die Beziehungen zwischen einzelnen Objekten genau definiert werden kann. Sprachen, die mit solchen relationalen Datenbanken arbeiten, müssen in der Lage sein, folgende Operationen auszuführen:

- Anlegen von neuen Relationen
- Verändern von Relationen
- Löschen von Relationen
- Erzeugen von Relationen aus vorhandenen Relationen mit ausgewählten Tupeln und ausgewählten Attributen

2.1.4 Relationenalgebra

Mengenoperationen, die auf eine Menge von Tupeln angewendet werden. Dabei entstehen Ergebnisrelationen, welche nicht in der DB gespeichert werden.

2.1.5 Relationenkalkül

Der Relationenkalkül beschreibt eine Menge von auszuwählenden Tupeln durch prädikatenlogische Ausdrücke, welche entsprechende Ergebnisrelationen nicht in die Datenbank speichert. Der Relationenkalkül ist somit eine deklarative, mengenorientierte Anfragesprache, die auf der Prädikatenlogik erster Stufe basiert.

2.2 Normalisierung

2.2.1 Anomalien

Widersprüchliche Information (Inkonsistenzen)

- **Änderungsanomalie**: Wenn man den Preis ändert wird er nur an einem Ort geändert
- **Löschanomalie**: Wenn man eine Person löscht, geht die Abteilung ebenfalls verloren
- **Einfügeanomalie**: Beim Einfügen einer Person schreibt man die Abteilung aus versehen falsch. So entsteht eine Abteilung mehr.

2.2.2 Normalformen

Bei relationalen Datenbanken werden bei der Normalisierung folgende Ziele erreicht:

Beheben von Anomalien (d.h. Probleme beim ändern, einfügen, löschen von Datensätzen), vermeiden von Redundanzen und übersichtlicher und einfacher Aufbau der Relationen, einfache Datenpflege.

Erste Normalform

- Zweidimensional
- jeder Datensatz kommt nur einmal vor
- Erstellen von Tabellen
- nur Daten die zu einem Objekt der realen Welt gehören
- Attribut kommt nur einmal vor
- nur 1 Wert pro Attribut

Zweite Normalform

- schafft sinnvolle Teiltabellen
- wobei ein Attribut zum Schlüssel der Relation bestimmt wird
- Die Angabe des Schlüssels ist für jede Relation obligatorisch.
- In der 2. Normalform ist jedes Nichtschlüsselattribut voll funktional vom Schlüssel abhängig, nicht von Teilen des Schlüssels und auch von keinem anderen Nichtschlüsselattribut.
- Zudem ist die Erste Normalform erfüllt

Dritte Normalform

- in der 3. Normalform wird die transitive Abhängigkeit aller Nichtschlüsselattribute von Schlüsselattributen, d.h. die Abhängigkeit über ein anderes Attribut, beseitigt
- Auch hier sind die erste Normalform und die zweite Normalform erfüllt.

2.2.3 Normalformen (zweite Erklärung)

1. NF	Eliminieren sich wiederholender Gruppen in einzelnen Tabellen. Erstellen einer separaten Tabelle für jeden Satz von Daten, die in einer Beziehung zueinander stehen. Kennzeichnen jedes Satzes mit zueinander in Beziehung stehenden Daten mit einem Primärschlüssel. Verwenden Sie nicht mehrere Felder in einer einzelnen Tabelle, um gleichartige Daten zu speichern.
2. NF	Erstellen separater Tabellen für Sätze von Werten, die auf mehrere Datensätze zutreffen. Herstellen einer Beziehung zwischen diesen Tabellen mit einem Fremdschlüssel. Datensätze sollten nur vom Primärschlüssel einer Tabelle abhängig sein (falls erforderlich, ein zusammengesetzter Schlüssel).
3. NF	Eliminieren von Feldern, die nicht vom Schlüssel abhängig sind. Werte in einem Datensatz, die nicht Teil des Schlüssels dieses Datensatzes sind, gehören nicht in die Tabelle. Im Allgemeinen sollten Sie immer dann, wenn der Inhalt einer Gruppe von Feldern auf mehr als einen Datensatz in der Tabelle zutreffen kann, diese Felder in einer separaten Tabelle zusammenfassen. Beispielsweise kann in einer Tabelle "Personalbeschaffung" der Name und die Adresse der Universität eines Kandidaten enthalten sein. Sie benötigen jedoch eine vollständige Liste von Universitäten für Serienbriefe. Wenn Universitätsdaten in der Tabelle "Kandidaten" gespeichert sind, besteht keine Möglichkeit, Universitäten aufzulisten, ohne dass aktuelle Kandidaten vorhanden sind. Erstellen Sie eine separate Tabelle "Universitäten" und verknüpfen Sie sie über einen Universitätscode-Schlüssel mit der Tabelle "Kandidaten".

http://support.microsoft.com/kb/100139/de

2.2.4 Normalformen: Beispiel

Quelle: Wikipedia (http://de.wikipedia.org/wiki/Normalisierung_(Datenbank))

Keine Normalform

- Das Feld Album beinhaltet die Attributwertebereiche Interpret und Albumtitel.
- Das Feld Titelliste enthält eine Menge von Titeln.

CD_Lieder

CD_ID	Album	Titelliste
4711	Anastacia - Not That Kind	{1. Not That Kind, 2. I'm Outta Love, 3. Cowboys & Kisses}
4712	Pink Floyd - Wish You Were Here	{1. Shine On You Crazy Diamond}

1. Normalform

- Das Feld Album wird in die Felder Albumtitel und Interpret gespalten.
- Das Feld Titelliste wird in die Felder Track und Titel gespalten sowie auf mehrere Datensätze aufgeteilt.

CD_Lieder

CD_ID	Albumtitel	Interpret	*Track*	Titel
4711	Not That Kind	Anastacia	1	Not That Kind
4711	Not That Kind	Anastacia	2	I'm Outta Love
4711	Not That Kind	Anastacia	3	Cowboys & Kisses
4712	Wish You Were Here	Pink Floyd	1	Shine On You Crazy Diamond

2. Normalform

- Der Primärschlüssel der Relation ist aus den Feldern *CD_ID* und *Track* zusammengesetzt. Die Felder *Albumtitel* und *Interpret* sind zwar vom Feld *CD_ID* abhängig, nicht aber vom Feld *Track*.

CD

CD_ID	Albumtitel	Interpret
4711	Not That Kind	Anastacia
4712	Wish You Were Here	Pink Floyd

Lieder

CD_ID	*Track*	Titel
4711	1	Not That Kind
4711	2	I'm Outta Love
4711	3	Cowboys & Kisses
4712	1	Shine On You Crazy Diamond

3. Normalform

- Die Tabelle Lieder enthält schliesslich nur noch Felder, die voll funktional von *CD_ID* und *Track* abhängen, liegt also auch in der 2. Normalform vor. Mit Hilfe dieser verlustfreien Zerlegung sind auch die genannten Redundanzen der Daten beseitigt.

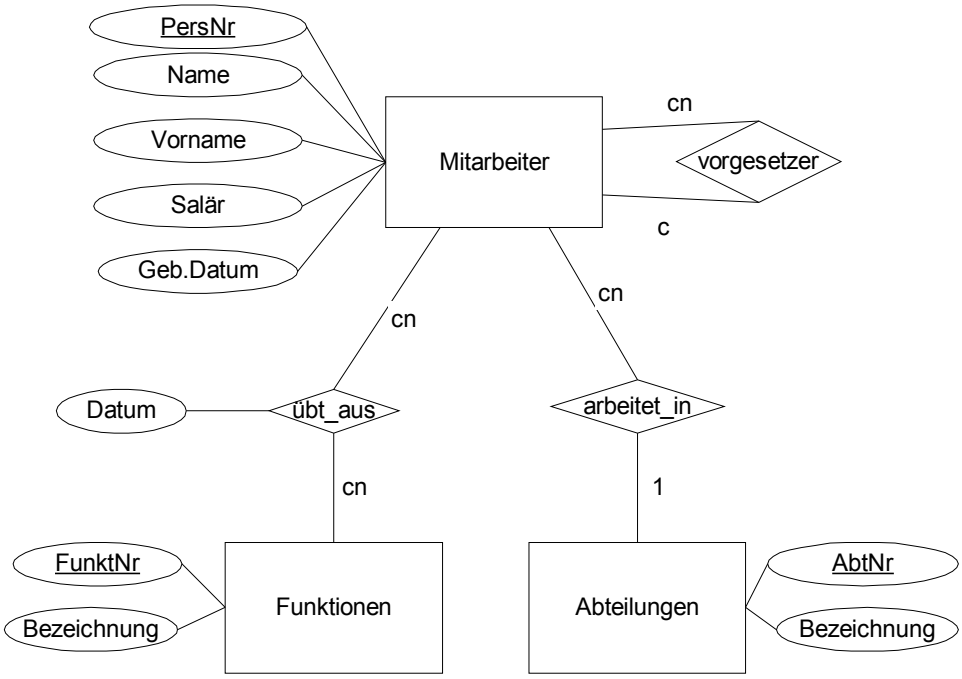
CD

CD_ID	Albumtitel	Interpret
4711	Not That Kind	Anastacia
4713	Freak of Nature	Anastacia
4712	Wish You Were Here	Pink Floyd

Künstler

Interpret	Jahr der Gründung
Anastacia	1999
Pink Floyd	1964

2.3 ERD in das relationale Datenmodell umwandeln



Schritt1 “Entitätsmengen = Relationen“

- Entitätsmenge Mitarbeiter → Neue Relation MITARBEITER
- Entitätsmenge Funktionen → Neue Relation FUNKTIONEN
- Entitätsmenge Abteilungen → Neue Relation ABTEILUNGEN

```
MITARBEITER (PersNr, Name, Vorname, Salär, GebDatum)
FUNKTIONEN (FunktNr, Bezeichnung)
ABTEILUNGEN (AbtNr, Bezeichnung)
```

Schritt2 “ Beziehungen die mind. auf einer Seite nicht bis unendlich gehen (1 oder C oder 0..2)“:

- Beziehung arbeitet_in
- Beziehung vorgesetzter

```
MITARBEITER (PersNr, Name, Vorname, Salär, GebDatum, fk_Abteilung,
fk_Vorgesetzter)
FUNKTIONEN (FunktNr, Bezeichnung)
ABTEILUNGEN (AbtNr, Bezeichnung)
```

Schritt3 “Beziehungen die auf beiden Seiten bis unendlich gehen (m oder n):

- Beziehung übt_aus

```
MITARBEITER (PersNr, Name, Vorname, Salär, GebDatum, fk_Abteilung,
fk_Vorgesetzter)
FUNKTIONEN (FunktNr, Bezeichnung)
ABTEILUNGEN (AbtNr, Bezeichnung)
UEBT_AUS (fk_PersNr, fk_FunktNr, Datum)
```

3 Schlüsselfelder und Indizes

3.1 Schlüsselfelder

- Jeder Datensatz braucht einen Primärschlüssel.
- Der Schlüsselfeld ist entweder ein Datenfeld oder eine Kombination von Datenfeldern.
- Schlüssel braucht man um Beziehungen zwischen Tabellen zu erstellen.
- Von jedem Schlüssel wird ein Index angelegt.

3.1.1 Primärschlüssel (Primary Key)

- Muss beim Erstellen der Tabelle erstellt werden.
- Nützlich, wenn er sich selbst erhöht.
- (Der Primärschlüssel darf keine NULL-Werte enthalten.) ???

3.1.2 Sekundärschlüssel (UNIQUE)

- Sekundärschlüssel können aus beliebig vielen Datenfeldern bestehen.
- Die Sekundärschlüssel dürfen auch keine NULL-Werte enthalten.

3.1.3 Fremdschlüssel (Foreign Key)

- Mit dem Fremdschlüssel verweist man auf eine andere Tabelle.
- Im Fremdschlüssel steht der Primärschlüssel von der anderen Tabelle.

3.2 Indizes

- Für jeden Schlüssel wird ein Index angelegt.
- Ein Index wird allgemein für ein bestimmtes Datenfeld oder auch mehrere Datenfelder angelegt, nach denen häufig sortiert oder gesucht wird.
- Es können mehrere Indizes definiert werden.
- Bei vielen Datensätzen oder grossen Tabellen, kann man merklich schneller durchsuchen.
- Wenn man etwas ändert, löscht oder hinzufügt, werden alle Indizes aktualisiet, was viel Zeit in Anspruch nehmen kann.
    - Deshalb erst am Schluss den Index erstellen (bei umfangreichen Tabellen)

4 Datenbankprogrammierung mit MySQL

Zum Sprachumfang von SQL gehören vier Befehlsgruppen:

- **DDL** (Data Definition Language): Erstellen von Datenbanken, Tabellen (Relationen) und Indizes
- **DML** (Data Manipulation Language)**:** Anlegen, Ändern und Löschen von Datensätzen
- **DQL** (Data Query Language): Abfragen von Daten
- **DCL** (Data Control Language): Anlegen von Benutzern und Vergabe von Zugriffsrechten

4.1 Datentypen

SMALLINT	2 Byte; -32768 bis + 32767
INTEGER	4 Byte; -2147483648 bis + 2147483647
FLOAT	4 Byte; 7 signifikante Stellen
DOUBLE PRECISION	8 Byte; 15 signifikante Stellen
NUMERIC(Präzision, Skalierung) **DECIMAL**(Präzision, Skalierung)	Präzision = Gesamtzahl der signifikanten Stellen; Skalierung = Anzahl Nachkommastellen NUMERIC: exakte Anzahl der signifikanten Stellen; DECIMAL: minimale Anzahl der signifikanten Stellen
DATE	8 Byte; 1.1.100 bis 11.12.5941. Wenn nichts angegeben wird auch automatisch die Zeig gespeichert [tt.mm.jj; tt-mmm-jj; mm-tt-jj]
CHAR(Länge)	Textinformationen, Max. Länge wird übergeben (1..255)
VARCHAR(Länge)	Länge = 1..255; Tatsächliche Länge in einem Byte, Zusatzbyte für Länge des Textes.
BLOB	Sehr grosse Datenmengen. Auch Binäre (Videos, Bilder, ect.)
TEXT	Wie BLOB jedoch unabhängig von Gross- und Kleinschreibung
Datentypumwandlung	**CAST** (Wert **AS** Datentyp) NUMERIC → CHAR, VARCHAR, DATE CHAR, VARCHAR → NUMMERIC DATE → CHAR, VARCHAR, DATE BLOB, TEXT → nicht möglich
Domänen	Benutzerdefinierter Datentyp mit der zusätzlichen Möglichkeit, Gültigkeitsprüfungen und Standardwerte festzulegen. Werden vom MySQL Server nicht unterstützt.

4.2 Kommentare

Kommentare	/* Hier */ oder so -- Das ist ein Kommentar

4.3 Datenbanken verwalten

Datenbank erstellen	`CREATE DATABASE [IF NOT EXISTS] datenbankname;`
Datenbank benutzen	`USE datenbankname;`
Datenbanken anzeigen	`SHOW DATABASES;`
Datenbank löschen	`DROP DATABASE [IF EXISTS] datenbankname;`
Aktuelle DB löschen	`DROP DATABASE;`

4.4 DDL: Tabellen erstellen und verwalten

- Daten können nur in Tabellen gespeichert werden.
- Beim zugreifen auf eine Tabelle muss immer der Name angegeben werden.

4.4.1 Tabellen erstellen

Tabelle erstellen	`CREATE TABLE tabellenname`
	`(datenfeld1 datentyp1 [DEFAULT standardwert | NULL | NOT NULL] [AUTO_INCREMENT],`
	`…`
	`Datenfeld_n datentyp_n [DEFAULT standardwert | NULL | NOT NULL] [AUTO_INCREMENT],`
	
	`CONSTRAINT constraintname1 CHECK (gültikeitsprüfung),`
	`…`
	`PRIMARY KEY (datenfeldname));`

`DEFAULT`	Der Standardwert eines Feldes, z.B. `preis FLOAT DEFAULT(1.5)`
`NULL`	Standardmässig hat das Datenfeld keinen Wert (MySQL)
`NOT NULL`	Bei der Eingabe muss unbedingt ein Wert eingegeben werden.
`AUTO_INCREMENT`	Es wird beim Anlegen eines Tupels automatisch +1 gerechnet. Für Primärschlüssel geeignet. (MySQL)
`CHECK`	`FeldN DatentypN CHECK (FeldN BETWEEN 100 AND 200)` oder so:
`CONSTRAINT`	`CONSTRAINT constraintname CHECK(FeldN > 100)`
Gültigkeitsprüfungen	Filter:
Datensätze Prüfen	`Name NOT LIKE „%Ä%“` `Abteilung IN („Einkauf“, „Verkauf“)`
Berechnen (nicht MySQL)	`Wert COMPUTED BY (stueck * preis)`

Primärschlüssel
`PRIMARY KEY`
	`FeldN DatentypN PRIMARY KEY [NOT NULL] [AUTO_INCREMENT]`
	oder:
	`[CONSTRAINT pk_name] PRIMARY KEY(FeldN [, FeldN2])`

Sekundärschlüssel `UNIQUE` Eindeutigkeit	`[CONSTRAINT uq_name] UNIQUE [Schlüsselfeldname] (Name, Vorname)`

Fremdschlüssel
 	
`FOREIGN KEY`
	`[CONSTRAINT fk_fremdschl] FOREIGN KEY(fk_PersNR)`
	`REFERENCES Tabelle (PersNr)`
	`[ON UPDATE referenzoption]`
	`[ON DELETE referenzoption]`
	
	Referenzoptionen
	Ausführung wird abgebrochen
	`NO ACTION`
	
	Alle Datensätze in anderen Tabellen die auf diesen Schlüssel verweisen werden auch geändert bzw. gelöscht:
	`CASCADE`
	
	Alle referenzierten Datenfelder werden auf den Defaultwert zurückgesetzt.
	`SET DEFAULT`
	
	Alle referenzierten Datenfelder werden auf NULL gesetzt.
	`SET NULL`

Indizes	`CREATE [ASC | DESC] INDEX indexname ON tabellenname (datenfeldname);` ASC DESC nicht in MySQL,

4.4.2 Tabellen verwalten

Tabellen anzeigen	`SHOW TABLES [FROM datenbankname] [LIKE „muster"];`
	z.B.
	`SHOW TABLES FROM kap_05 LIKE „%a%";`

Tabelle ändern	`ALTER TABLE tabellenname`
	
	`[ADD datenfelddefinition]`
	`[ADD indexdefinition]`
	`[ADD CONSTRAINT constraintname CHECK (gültigkeitsprüfung)]`
	`[DROP objektname]`

Tabelle löschen	`DROP tabellenname;`
Primärschlüssel löschen	`ALTER TABLE tabellenname DROP PRIMARY KEY;`
Sekundärschlüssel hinzufügen	`ALTER TABLE tabellenname ADD UNIQUE [schlüsselname] (datenfeld1, datenfeld2);`
Sekundärschlüssel löschen	`ALTER TABLE tabellenname DROP INDEX sclüsselname;`
Fremdschlüssel hinzufügen	`ALTER TABLE tabellenname ADD FOREIGN KEY(fk_PersNR)` `REFERENCES Tabelle (PersNr)`
Indizes anzeigen	`SHOW INDEX;` /*Nicht in MySQL */
Indizes löschen	`DROP INDEX indexname [ON tabellenname];`

4.5 DML: Daten einfügen, aktualisieren und löschen

4.5.1 Daten einfügen

Daten einfügen	`INSERT INTO tabellenname (feld1, ..., feldX)` `VALUES (wert1, ..., wertX);`
Default, NULL, NOT NULL	Der DEFAULT-Wert wird automatisch geschrieben. Bei NOT NULL wird eine Fehlermeldung ausgegeben, wenn man nichts eingibt. NULL = Keinen Wert

Mehrere Datensätze einfügen	`INSERT INTO tabellenname (feld1, ..., feldX)`
	`SELECT [ * | datenfelder] FROM tabellenname`
	`[WHERE bedingung];`
	Zum Beispiel:
	`INSERT INTO schuepfheim (name, vname, str, plz, ort, alt)`
	`SELECT name, vname, str, plz, ort, alt FROM personen`
	`WHERE ort = „Schuepfheim“;`

alle Datensätze anzeigen	`SELECT * FROM tabellenname;`

4.5.2 Datensätze ändern

Datensätze ändern	`UPDATE tabellename` `SET feld1 = wert1, ..., feldX = wertX [WHERE bedingung];`

4.5.3 Datensätze löschen

Datensätze löschen	`DELETE FROM tabellenname [WHERE bedingung];`

4.6 DQL: Einfache Datenabfragen

Daten abfragen	`SELECT [DISTINCT] *|(Datenfelder AS Spaltenüberschrift)`
	`FROM Tabellenname`
	`[WHERE Bedingung]`
	`[GROUP BY [HAVING ]]`
	`[ORDER BY [ASC|DESC]]`
	`[LIMIT [Start, ] Anzahl]`

`DISTINCT`	Doppelte Datensätze vermeiden
Berechnungen	`SELCET preis, stueck, preis*stueck AS Lagerwert` `FROM tabellenname`
Konstante Überschriften	`SELECT 'Inhalt' as 'Überschrift', vname AS Vorname` `FROM Tabellenname`
Beispiel	`SELECT name AS Familienname, vname as Vorname` `FROM tabelle`

Bedingungen
`WHERE`
`BETWEEN`
`IN`
`LIKE`
`IS NULL`
`<>`
`NOT IN`
	`SELECT * FROM Tabellenname WHERE...`
	Vergleichen: preis < 100 OR (name = „Meier“ AND Preis >500);
	<>, !=
	Bereichsprüfung: preis BETWEEN 10 AND 100
	Elementprüfung: abteilung in („Einkauf“, „Verkauf“) OR PersNr IN (100, 5);
	Mustervergleich: name LIKE „M%“ OR name LIKE „___ller“
	Falls im Muster eines der Zeichen % oder _ vorkommt, dann bitte so machen: name LIKE „Abt_%“ ESCAPE „
	Nullwertprüfung: preis IS NULL
	Logische Operationen: name <> „Meier“ OR name <> „Mayer“
	AND OR NOT
	Nur Attribute und keine Aliasse verwenden (AS-Schlüsselworte)
	Verschachtelt: where PersNr NOT IN (select fk_PersNr From Reinigt);
	(Person, die keinen Raum putzt)

Gruppieren
`GROUP`
 	
`HAVING`
	`SELECT ort, COUNT(name) FROM Tabellenname GROUP BY ort HAVING COUNT(name) > 1;`
	Die Datensätze werden ermittelt und nach dem Ort gruppiert. Die Aggregatfunktion COUNT ermittelt die Anzahl der Namen. Der Name des Feldes, auf welche die Aggregatfunktion angewendet werden soll, folgt in den Klammern.
	Having: Einschränken: GROUP BY name HAVING COUNT(name) > 1;
	= Alle, die mehr als 1 Mitarbeiter haben.

Sortieren
`ORDER BY`
`ASC | DESC`
	`SELECT * from klassenliste ORDER BY geschlecht, name, vorname DESC`
	ASC = Aufsteigend; **DESC** = Absteigend

4.7 Funktionen

In Abfragen mit Funktionen kann man keine Datenfelder miteinbeziehen. Möglich ist dies aber, wenn man das Datenfeld mit GROUP BY gruppiert:

```
SELECT ort AS Wohnort, COUNT(name) FROM tabelle GROUP BY ort
```

Aggregatfunktionen: Funktionen, die der statistischen Auswertung der Werte eines Datenfeldes oder einer Gruppe innerhalb einer Abfrage dienen.

4.7.1 Standard-Funktionen

Standard-Funktionen sind mit quasi allen SQL-DBS möglich. Es dürfen nur Attribute und keine konstanten Werte verwendet werden.

`COUNT()`	Liefert die Anzahl der Werte in der Ergebnismenge einer SELECT-Abfrage bzw. einer Gruppierung. `SELECT PersNR, COUNT(projekt) FROM tabelle GROUP BY PersNR.`
`COUNT(DISTINCT)`	Liefert die Anzahl der unterschiedlichen Werte in einer Abfrage oder Gruppierung. `SELECT COUNT(distinct plz) FROM tabelle`
`AVG()`	Liefert der Durchschnittswert zurück. `SELECT AVG(stueck) FROM tabelle`
`MIN()` `MAX()`	Liefert den kleinsten / grössten Wert eines Datenfeldes der Abfrage oder Gruppierung. `SELECT MIN(preis) FROM tabelle`

`SUM()`	Liefert die Summe der Werte eines Datenfeldes in der Abfrage oder Gruppierung.
	`SELECT SUM(preis) FROM tabelle WHERE stueck>10`
	`SELECT SUM(Besucher) FROM tabelle GROUP BY Stil`

Bedingungen `HAVING` Nur bei GROUP BY	`SELECT proj_id, count(ma_id) FROM tabelle GROUP BY proj_id` `HAVING COUNT(ma_id)>2;`

4.7.2 Nicht standardisierte Funktionen

Bei diesen Abfragen verwendet man auch wieder den SELECT-Befehl. Man kann Attribute von Tabellen verwenden, konstante Werte (Zahlen und Strings) oder Ergebnisse anderer Funktionen (verschachteln).

Man kann diese Funktionen auch mit dem WHERE-Befehl benutzen: ... where lower(name) = „meyer“;.

Die Funktionen können auch in der VALUES-Klausel der INSERT-Anweisung verwendet werden: INSERT INTO tabelle VALUES(rand(), ...);

`ABS(zahl)`	Liefert den Betrag (Abstand zu Null)
`CEILING(zahl)`	Rundet den Wert auf die nächstgrössere Zahl
`FLOOR(zahl)`	Rundet den Wert auf die nächstkleinere Zahl
`ROUND(zahl)`	Rundet die Zahl nach der üblichen mathematischen Regel
`ROUND(zahl,stellen)`	Rundet die Zahl nach der üblichen mathematischen Regel auf die angegebenen Anzahl Dezimalstellen
`LOG(zahl)`	Ermittelt den natürlichen Logarithmus
`MOD(zahl1,zahl2)`	Liefert den Rest der Division von zahl1 durch zahl2
`PI()`	liefert den Wert von PI zurück
`RAND()`	Zufällige Zahl zwischen 0 und 1 round(rand() * 1000) + 1 = zufällige Zahl zwischen 1 und 1000
`SIGN(zahl)`	Ermittelt das Vorzeichen einer Zahl und liefert daraufhin den Wert -1 bzw. 0 oder 1
`SIN(zahl); COS(zahl); TAN(zahl)`	Berechnet die Winkelfunktionen.
`SQRT(zahl)`	Ermittelt die Quradrahtwurzel der Zahl
`ASCII(string)`	liefert den ASCII-Wert vom ersten Zeichen der Zeichenkette
`CHAR(wert)`	Liefert das Zeichen vom angegebenen ASCII-Wert
`LENGTH(string)`	gibt die Länge der Zeichenkette aus
`LOWER(); UPPER()`	Wandelt die angegebene Zeichenkette in Gross- bzw. Kleinbuchstaben um
`LTRIM(string)` `RTRIM(string)`	Entfernt alle führenden bzw nachfolgenden Leerzeichen der Zeichenkette
`SUBSTRING(string, anfang, länge)` `SUBSTR(string, anfang, ende)`	Liefert ein Teilstück der angegebenen Zeichenkette, definiert durch die Anfangsposition und Länge bzw. Endposition.

4.8 Datenabfragen über mehrere Relationen

- Da die Daten über mehrere Relationen verteilt sind (mit Hilfe von Fremdschlüssel), muss man spezielle Abfragen machen, damit man alle Daten in einer Tabelle ausgeben kann.

4.8.1 Einfaches Verknüpfen von Tabellen

- Folgende Form von Verknüpfen von Tabellen ist veraltet.

Einfache Abfrage	`SELECT tabelle1.nr, tabelle1.name, tabelle2.* FROM tabelle1, tabelle2, WHERE tabelle1.nr = tabelle2.fk_PersNr;`
Verkürzte Fassung (mit AS)	`SELECT m.Vorname, m.Name, m.Abt_Nr, a.* FROM t_mitarbeiter AS m, t_abteilung AS a WHERE m.Abt_Nr = a.id;`

4.8.2 Mit Joins verknüpfen

Inner-Join / Equi-Join

Dieser Join verbindet Datensätze aus zwei oder mehreren Tabellen alle Datensätze, zu denen auf beiden Seiten ein entsprechender Datensatz existiert. Der Fremdschlüssel entspricht dabei dem Primärschlüssel.

```
SELECT t_ma.name, t_ma.abtnr, t_abt.* FROM t_ma INNER JOIN t_abt ON t_ma.abnr =
t_abt.id [INNER JOIN ...]
```

Left-Outer-Join

Wie der Inner-Join, es wird die linke Tabelle (vom Schlüsselwort Join aus) genommen, und mit der rechte ergänzt. So kann es vorkommen, dass bei der rechten Tabelle viele Nullwerte vorkommen.

```
SELECT t_ma.name, t_ma.abtnr, t_abt.* FROM t_ma LEFT JOIN t_abt ON t_ma.abnr =
t_abt.id [LEFT JOIN ...]
```

Right-Outer-Join

Wie der Inner-Join, es wird die rechte Tabelle (vom Schlüsselwort Join aus) genommen, und mit der linken ergänzt. So kann es vorkommen, dass bei der linken Tabelle viele Nullwerte vorkommen.

```
SELECT t_ma.name, t_ma.abtnr, t_abt.* FROM t_ma RIGHT JOIN t_abt ON t_ma.abnr =
t_abt.id [RIGHT JOIN ...]
```

Full-Join

Wie der Inner-Join. Es werden alle Datensätze der linken und alle Datensätze der rechten Tabelle angezeigt. Daher kann es vorkommen, dass es leere Datenfelder von der rechten und von der linken Tabelle anzeigt.

```
SELECT t_ma.name, t_ma.abtnr, t_abt.* FROM t_ma FULL JOIN t_abt ON t_ma.abnr =
t_abt.id [FULL JOIN ...]
```

Natural-Join

Dies ist ein Inner-Join, bei dem doppelte Datensätze vermieden werden. Es gibt jedoch keine spezifische Syntax für diesen Join. Daher greifen wir zum Schlüsselwort DISTINCT.

```
SELECT DISTINCT t_ma.name, t_ma.abtnr, t_abt.* FROM t_ma INNER JOIN t_abt ON
t_ma.abnr = t_abt.id [INNER JOIN ...]
```

Theta-Join

Wie der Inner-Join / Equi-Join, es wird jedoch in der ON-Klausel nicht auf Gleichheit, sondern nach einer anderen logischen Operation geprüft. In diesem Beispiel ist es: t_ma.abnr > t_abt.id.

```
SELECT t_ma.name, t_ma.abtnr, t_abt.* FROM t_ma INNER JOIN t_abt ON t_ma.abnr >
t_abt.id [INNER JOIN ...]
```

Self-Join

Die Tabelle macht mit sich selbst eine Verbindung. Beim Self-Join muss man das Schlüsselwort AS benutzen, damit man die selbe Tabelle mehrmals angeben kann.

```
SELECT a1.name, a2.name, a1.ort FROM t_abt AS a1 INNER JOIN t_abt AS a2 ON
a1.ort = a2.ort WHERE a1.id <> a2.id
```

Cross-Join

Das ist das kartesische Produkt oder auch vollständiges Kreuzprodukt genannt. Es wird jede Kombination von den Datensätzen von den Tabellen gemacht. D.h. bei Tabellen von 75 und 648 gibt das 48.600 Zeilen!

```
SELECT t_ma.name, t_ma.abtnr, t_abt.* FROM t_ma CROSS JOIN t_abt;
```

4.8.3 Zwei Tabellen vereinigen

Vereinigungsmenge

Es werden nach Datensätzen gesucht, welche entweder in tabelle1 oder in tabelle2 vorkommen. (z.B. Bei tabelle1 nur Personen von Schüpfheim und in tabelle2 nur Personen von Entlebuch). Diese werden dann vereinigt.

```
SELECT name, ort FROM tabelle1 WHERE ort = 'schüpfheim' UNION SELECT name, ort
FROM tabelle2;
```

Schnittmenge

Aus zwei Tabellen werden nach zwei gleichen Datenfeldern gesucht:

```
SELECT name, ort FROM tabelle1 INTERSECT SELECT name, ort FROM tabelle2;
```

Differenzmenge

Es werden nach Datenfeldern gesucht, welche nur in der tabelle1 vorhanden sind.

```
SELECT name, ort FROM tabelle1 MINUS SELECT name, ort FROM tabelle2;
```

4.9 Sichten

- Eine Sicht erstellt quasi eine „virtuelle“ Tabelle.
- Die Sichten kann man wie Tabellen verwenden.
- Es lassen sich Berechtigungen erteilen.
- Sichten sind Ereignismengen von Abfragen.
- Die Struktur der Tabelle wird durch die Sichten nicht verändert.
- Man kann mehrere Tabellen verknüpfen.
- GROUP BY, HAVING, ORDER BY und UNION dürfen nicht verwendet werden.

Sicht erstellen	`CREATE VIEW sichtname [(datenfeldertitel)] AS SELECT datenfelder FROM tabelle;`
Sicht löschen	`DROP VIEW sichtname;`
z.B. Datenabfrage	`SELECT * FROM sichtname; → ?????????`
z.B. Datenänderung	`UPDATE sichtname SET datenfeldname = 'wert' WHERE bedingung;`
z.B. Daten einfügen	`INSERT INTO sichtname (datenfelder) VALUES (werte);`

- Mit der Option WITH CHECK OPTION nach der WHERE-Klausel erfolgt eine Überprüfung der geänderten Datensätze.

4.10 Cursor

- Für die schrittweise Abfragung von Datensätzen werden Datenpuffer angelegt.
- Es können nur Abfragen gemacht werden.
- Der Cursor verweist auf eine bestimmte Stelle in diesem Datenpuffer.
- Beim Verarbeiten erfolgt ein sequenzielles Lesen aus der Datei: Datei öffnen, Datensatz für Datensat lesen, Datei schliessen.
- Es wird Datensatz für Datensatz gelesen. Jede Abfrage liefert den nächsten Datensatz zurück.

Cursor erstellen	`DECLARE cursorname CURSOR FOR SELECT datenfelder FROM tabelle WHERE bedingung;`
Cursor öffnen	`OPEN cursorname;`
Daten lesen ?????	`FETCH cursorname INTO | USING :hostvariable_1, :hostvariable_n` Mit INTO bzw USING wird das Speichern der Werte der einzelnen Datenfelder in entsprechende Variabeln des ausführenden Programms gesteuert. Diese Variabeln müssen vorher definiert werden und einen passenden Datentyp besitzen.
Cursor schliessen	`CLOSE cursorname;`

5 Glossar

Attribut	Eingenschaften von einem Objekt, Spalte, Datenfeld, Column, Field
Beziehung	Relationship
Column	Siehe Spalte,
Dateien	Siehe Daten (z.B. Musik, Filme)
Daten	Nachrichten, die maschinell verarbeitet und gespeichert werden können.
Daten / Nachrichten	Daten, die maschinell gespeichert und verarbeitet werden.
Transaktion	Eine abfolge von Befehlen, die als Ganzes betrachtet wird. Wird die Transaktion fehlerhaft ausgeführt, wird die Datenbank in den Zustand vor der Transaktion zurückversetzt.
Datenbank	Enthält Daten und Metadaten
Datensatz	Auch Tupel, Zeile einer Spalte: Enthält Attribute wie z.B. Den Namen oder die Adresse.
DBMS	DatenBankManagementSystem; Software zur Verwaltung von Datenbanken
DBS	DatenBankSoftware
Entität	Zeile, Datensatz, Tupel, Row
Entitätsmenge	Menge aller Objekte, die vom Typ her zusammengehören.(Alle Karteikarten von den Kunden) Tabelle, Relation
Entity	Siehe Objekt.
Feld	Siehe Datenfeld.
Field	Siehe Datenfeld.
Information	Hat einen Neuigkeitswert und ist somit nützlich. Daten stellen für sich alleine noch keine Informationen dar, erst der Zusammenhang macht aus ihnen Informationen (Bsp. Die Daten 1 und 8 werden im Zusammenhang mit einem Haus zu einer Hausnummer).
Information	Mehr als Nur Daten. Daten in Bezug zu Etwas. Information = nützliche Nachricht
Kolonnen	Siehe Spalte.
Metadaten	Daten, welche die Struktur der Daten innerhalb der Datenbank beschreiben.d
Metainformationen	Siehe Metadaten
Nachricht	Eine Nachricht ist das Wissen über Zustände und Ereignisse der Realität und beinhaltet je nach den Bedürfnissen des Empfängers Informationen und Redundanzen.
NULL	Ein Attribut ohne Attributswert.
Objekt	Eine Entität.
Record	Siehe Tupel.
Redundanz	Mehrfach vorhanden, keinen Neuigkeitswert --> nutzlos
Referentielle Integrität	Beziehungen zwischen Objekten
Schwach strukturierte Daten	Tabelle
Spalte	Datenfeld
SQL	Structured Query Language

Strukturierte Daten	Datenbank
Tabellen	Schwach strukturierte Daten
Tupel	Siehe Datensatz.
Unstrukturierte Daten	Reiner Text
Wertebereich	Siehe Domäne.
Zeile	Siehe Datensatz.
Domäne	Liste von zugelassener Werte.

Begriff (ERD)	Synonym DE	Synonym EN
Entitätsmenge	Tabelle, Relation	Table, Relation
Entität	Zeile, Tupel,Datensatz	Row
Attribut, Eigenschaft	Spalte, Datenfeld	Column, Field
Beziehung	Beziehung	Relationship

6 Gute Links

-

Stichwortverzeichnis