Was sind Datenbanken?

Datenbanken sind Teil eines betriebswirtschaftlichen Informationssystems, das nach [Scheer95] als "ein System zur Aufnahme, Speicherung, Verarbeitung und Weitergabe von Informationen definiert wird.
(Script Datenbanken Prof. Dr. Johannes)

Eine Datenbank ist die elektronische Form eines Karteikastens. Es handelt sich um eine Sammlung von Daten, die aus der Sicht des Benutzers zusammengehören, z. B. eine Personaldatenbank oder eine Lagerinventardatenbank. Es gibt hierarchische, relationale, multidimensionale und Objektorientierte Datenbanken. Die Datenbank wird üblicherweise von einem Datenbankverwaltungssystem (DBMS) verwaltet. Ein DBMS zusammen mit einer oder mehreren Datenbanken nennt man Datenbanksystem (DBS).
(www.wikipedia.de)

Eine Datenbank ist eine geordnete Sammlung von Daten, die normalerweise in einer oder in mehreren zusammengehörenden Dateien gespeichert ist. Die Daten sind als Tabellen strukturiert, wobei Verweise von einer Tabelle auf eine andere möglich sind.
(Kofler, MySQL)

Vorteile / Eigenschaften von Datenbankensystemen

Datenunabhängigkeit

- unabhängig von Anwendungsprogrammen

Datensicherheit

- Datenschutzmechanismus

Prüfung der Konsistenz und Integrität

- Bedingungen (Kredit muss unter 10.000 Euro bleiben)

Synchronisation im Mehrbenutzerbetrieb

- 2 User reservieren gleichzeitig Platz

Datensicherung (Recovery)

- Wiederherstellung nach Festplatten- / DB-Crash

Zugriff per SQL

- Schnittstelle zur Datenbank

Einsatzgebiete von Datenbanken

- Unser Intranet
- Auktionsplattformen im Internet
- Buchhaltungssysteme
- Bibliotheken
- Mitarbeiterdatenbanken

Modelle und Strukturen

- **Hierarchisches Modell**
  Datendateien sind hierarchisch angeordnet
  z.B.
  Kunde, Aufträge und Artikel können hierarchisch geordnet werden
  Baumstrukturen

- **Netzwerk Modell**
  Erweiterung des hierarchischen Modells um Netzstrukturen

- **Relationales Modell**
  Unterstützt jede Form der Datenbeziehungen
  Basiert auf dem Relationenmodell der Algebra

Datenbank Architektur

Externe Datensicht

Sichtweise auf die Datenbank die sich an den Bedürfnissen des Anwenders/Benutzers orientiert (sog. Views) z.B. Telefonnummer und Name anzeigen lassen. Die restlichen Daten sind uninteressant

Konzeptionelle (logische) Datensicht

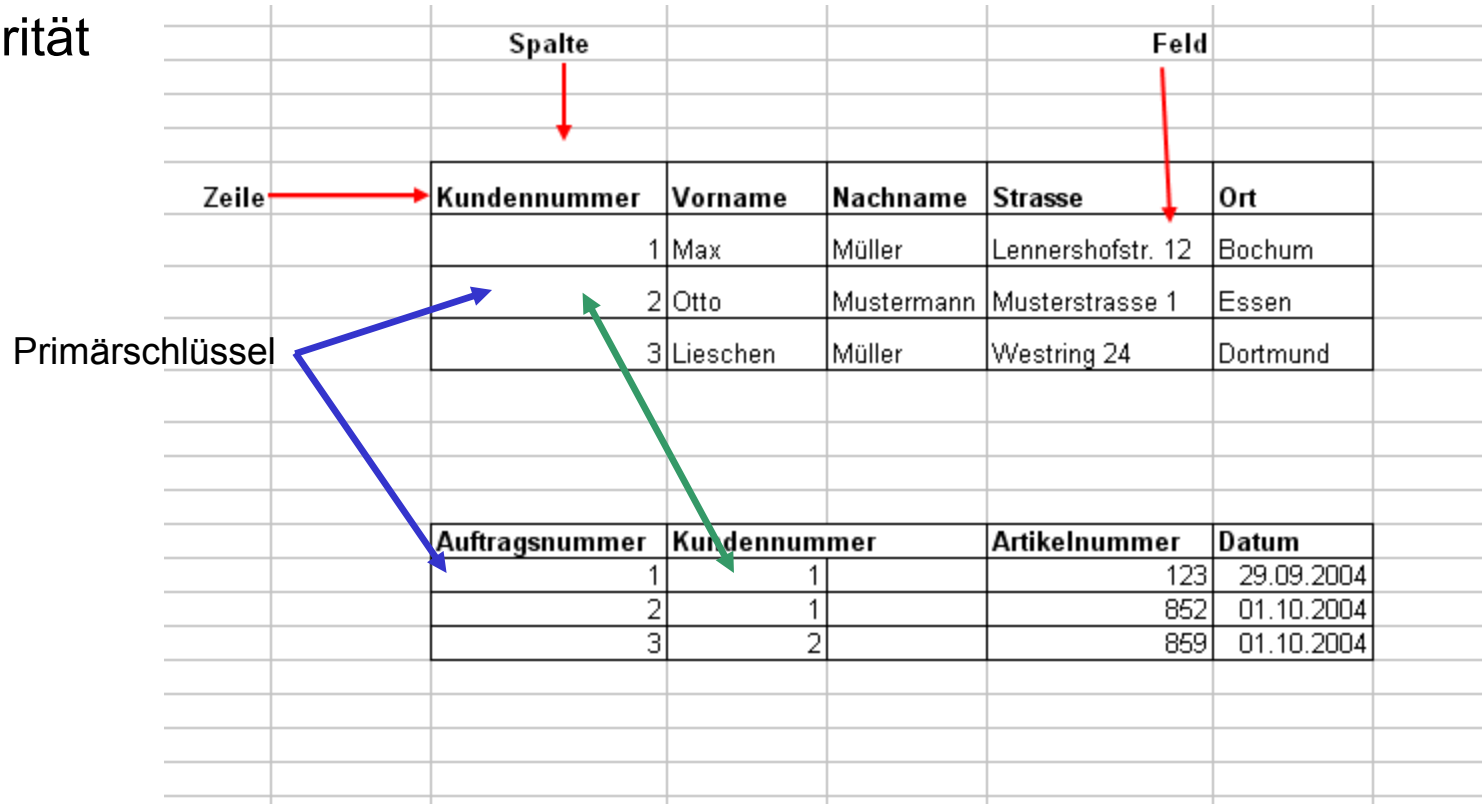
bezeichnet die Sicht der Gesamtheit der Daten in der Datenbank, sie ist einheitlich und eindeutig, beschreibt die Beziehungen der Daten untereinander in einem Schema, das den konzeptionellen Aufbau der Datenbank darstellt. (ERM)

Physische (interne) Datensicht

interne Datenorganisation

Relationales Modell - Konzept

Wurde von E.F. Codd 1970 begründet
Oracle war das erste DBMS welches auf dem relationalen Modell basierte
Basis für die Speicherung sind Tabellen
Tabellen sind durch Beziehungen in Verbindung miteinander
Primärschlüssel wird Fremdschlüssel in anderer Tabelle
Referentielle Integrität



Kundennummer	Vorname	Nachname	Strasse	Ort
1	Max	Müller	Lennershofstr. 12	Bochum
2	Otto	Mustermann	Musterstrasse 1	Essen
3	Lieschen	Müller	Westring 24	Dortmund

Auftragsnummer	Kundennummer	Artikelnummer	Datum
1	1	123	29.09.2004
2	1	852	01.10.2004
3	2	859	01.10.2004

Obsthändler Kramers erster Datenbankentwurf

Eine völlig unerfahrene Person würde eine Datenbank wie folgt planen:

Ein Obsthändler Krämer hat folgenden Datenbankentwurf erstellt:

Auftragsnummer	***Datum***	***Kunde***	***Artikelnummer***	***Bezeichnung***	***Menge***
1	01.01.04	1 Schmitt, Bonn	134	Coxorange	4 Kisten
1	01.01.04	1 Schmitt, Bonn	135	Kiwi	4 Kisten
2	01.01.04	2 Müller, Köln	140	Butterbirne	2 Kisten
2	01.01.04	2 Müller, Köln	160	Kürbis, rot	2 Stück
2	01.01.04	2 Müller, Köln	160	Kürbis, gelb	10 Stück
3	02.02.04	1 Schmitz, Bonn	103	Johannesbeeren	5 Kilo
3	02.02.04	1 Schmitz, Bonn	134	Coxorange	12 Kisten
3	02.02.04	1 Schmitz, Bonn	135	Kiwi	2 Kisten
4	02.02.04	45 Lehmann, Jülich	30	Bananen	12 Kilo
4	02.02.04	45 Lehmenn, Jülich	27	Ananas	60 Stück

Sofort erkennbare Probleme

- Viele gleiche Einträge (Redundanz) sorgen nach ein paar Monaten dafür, dass die Datenbank unnötig gross wird
- Tippfehler (letzte Zeile: Lehmenn) machen das Auffinden aller Lieferungen an Lehmann, Jülich unmöglich
- In der Spalte Kunde sind sowohl Kundennummer, Name und Ort zugleich eingetragen. Herr Krämer muß sich stets selber die Kundennummer merken und alle Daten bei jedem neuen Auftrag stets neu eingeben
- Artikelnummer und Bezeichnung könnten als Synonyme verwendet werden, sind aber hier jeweils für sich in getrennten Spalten gespeichert
- Die Farbe des Kürbis gibt eine andere Art an, die eine neue Artikelnummer erfordert
- Die Spalte Menge enthält sowohl die Stückzahl als auch die Einheit der Ware (Kiste, Stück, Kilo). Die Einheit ist aber stets an die Art der Ware gekoppelt.

Wirkliche Probleme (Anomalien)

- **Einfüge-Anomalie**
  Ein Lieferant / Kunde kann nur dann eingetragen werden, wenn er auch eine Ware liefert / kauft. Das ist bei Herrn Krämers Datenbankentwurf der Fall

- **Lösch-Anomalie**
  Wird die einzige Lieferung an einen Kunden/von einem Lieferanten gelöscht, weil diese nicht angekommen ist, oder weil dieser eine Ware nicht mehr liefert, so sind auch die Informationen über Wohnort, Anschrift … verloren. Auch dies trifft auf das Beispiel zu.

- **Änderungs-Anomalie**
  Falls ein Lieferant / Kunde umzieht, so sind mehrere Tupel (Einträge in der DB) evtl. nachträglich zu ändern, da ansonsten die Rechnungen sowohl an die alte sowie auch an die neue Anschrift adressiert werden

- **Update-Anomalie**
  Wird eine Korrektur z.B. bei einer Artikelnummer durchgeführt, so kann es passieren, daß entweder alle bisherigen Einträge ebenfalls aktualisiert werden müssen, oder daß es zu Inkonsistenzen führt. Dabei bezeichnet dann eine Artikelnummer gleich mehrere Sorten von Obst innerhalb der DB

Obsthändler Krämer II

AuftragsNr	Datum	KundenNr	Name	Ort	ArtNr	Bez	Menge

Auflösung von Kundennummer, Name und Ort
Es sind jetzt nur noch unteilbare (atomare) Werte enthalten

Aber:

- Redundanz ist noch enthalten
- Der Tippfehler ist auch noch vorhanden
- Artikelnummer ist noch synonym zur Bezeichnung
- Kürbisfarbe
- Menge enthält immer noch die Einheit

Es gehören zusammen:

ArtNr, Bezeichnung, Menge

KundenNr, Name und Ort

Obsthändler Krämer III

AuftragsNr	Datum	KundenNr	Name	Ort	ArtNr	Bez	Menge

Weitere "Verfeinerung":

AuftragsNr	Datum	KundenNr	Name	Ort

AuftragsNr	ArtNr	Bez.	Menge

Problem
Auftragsnummer und Datum ist nicht vom Kunden abhängig
Erteilt Kunde weiteren Auftrag, werden Kundendaten wieder eingetragen
Analog verhält es sich bei den Artikeln

Auftragsnummer ist nicht vom Artikel abhängig
Es ist nicht bekannt welche Menge bestellt wird

Obsthändler Krämer IV

AuftragsNr	Datum	Kunde	ArtikelNr	Bezeichnung	Menge
1	01.01.04	1 Schmitt, Bonn	134	Coxorange	4 Kisten

AuftragsNr	Datum	KundenNr	Name	Ort	ArtNr	Bez	Menge

Tabelle Kunde		1
KundenNr	Name	Ort

Tabelle Artikel	3
ArtNr	Bezeichnung

Tabelle Bestellung		3
AuftragsNr	KundenNr	Datum

Tabelle Stückliste		4
AuftragsNr	Menge	ArtNr

Verknüpfung von Tabellen

Verknüpft wird mit Hilfe des Fremdschlüssels

Master-Tabelle (Kunde) — Detail-Tabelle (Auftrag)

1 : N

Referentielle Integrität

- Keine Kundennummer in Auftrag, zu der kein Kunde existiert
- Keine Möglichkeit, einen Kunden zu löschen, der noch offene Aufträge hat

Beispiel für Master / Detailtabelle

1 Kunde hat verschiedene Aufträge
1 Auftrag gehört immer zu einem Kunden

1:N Beziehung

Mastertabelle

KdNR	Name	Strasse	Ort
1	Müller	Holzweg	Bochum
2	Mustermann	Testplatz	Dortmund
3	Meier	Lennershofstr.	Essen

Detailtabelle

AuftragNr	KdNR	ArtikelNr	Menge
1	1	123	12
2	1	456	34
3	2	567	2
4	3	678	4
5	2	987	6
6	1	876	4

Tabellen heißen auch Relationen

Tabellen und Primärschlüssel

Tabellen

- ersetzen “Karteikarten”
- Zeilen, Spalten
- Zeilen enthalten Datensätze
- Spalten enthalten Felder oder Attribute
- Schnittpunkte von Zeilen und Spalten heißen Zellen

Primärschlüssel

- identifiziert eindeutig einen Datensatz
- kann aus mehreren Spalten bestehen
- wird häufig künstlich erzeugt, z.B. Kundennummer

Tabellen können miteinander verknüpft werden, z.B. Auftrag und Kunde
Dazu enthält Aufträge z.B. das Feld Kundennr
Eine solche Spalte heißt Fremdschlüssel, da diese Spalte einen Kunden in einer “fremden” Tabelle identifiziert

Codd'sche Regeln (definieren das relationale Datenmodell)

Entitäts-Integrität:
Jede Relation weist einen Primärschlüssel auf, z. B. die Kundennummer in KUNDE.

Referentielle Integrität:
Jeder Fremdschlüssel verweist auf einen Primärschlüssel aus einer anderen Relation, z. B. die Kundennummer in AUFTRAG.

Benutzerdefinierte Integrität:
Die Zulässigkeit von Werten für ein Attribut kann eingeschränkt werden, z. B. 10000 <= PLZ <= 89999 oder FERTIGIST >= FERTIGSOLL.

Selektion:
Auswahl von bestimmten Zeilen mittels Auswahlkriterien. Aus der Tabelle wird eine Teilmenge von Datensätzen (Tupeln) ausgesucht, z. B. alle Kunden mit ORT = Bremen.

Projektion:
Auswahl von bestimmten Spalten der Tabelle. Die Ergebnismenge, d. h. das Ergebnis einer Abfrage, soll nur eine Teilmenge von Attributen (Spalten) haben, z. B. nur FIRMA und ORT.

Verbund (Join):
Mehrere Relationen können in einer Abfrage miteinander verknüpft werden, z. B. suche die Aufträge der Kunden, die in Bremen wohnen; die Daten werden dann aus den Tabellen AUFTRAG und KUNDE zusammengestellt.

SQL (**S**tructured **Q**uery **L**anguage)

DDL Data Definition Language mit
- CREATE (Anlegen von Tabellen, Sichten, Indexen, ...)
- ALTER (Ändern)
- DROP (Löschen)

DML Data Manipulation Language mit
- INSERT (Einfügen von Zeilen)
- UPDATE (Ändern)
- DELETE (Löschen)
- SELECT (Abfragen)

DCL Data Control Language mit
- GRANT (Vergabe von Zugriffsrechten)
- REVOKE (Zurücknahme von Zugriffsrechten)
- COMMIT (Abschluß einer Transaktion)
- ROLLBACK (Abbruch einer Transaktion)

Entwurf einer relationalen Datenbank

1. **Informationsbedarfs-Ermittlung und -Analyse**
   - Sammlung aller in einer Miniwelt des Unternehmens relevanten Objekte, ihrer Eigenschaften und Beziehungen.
   - Überprüfung dieser Informationen auf inhaltliche Korrektheit.
2. **Konzeptueller (logischer) Datenbankentwurf**
   - Entwurf der Tabellen und Definition der Attribute,
   - Primärschlüssel, Fremdschlüssel, Datentypen.
   - Methodik: Entity-Relationship-Model, Normalisierung.
   - Festlegung weiterer Integritäts-Regeln wie Wertebereiche u. a. sowie Konsistenzregeln.
3. **Physischer Datenbankentwurf**
   - Anlegen von Datencontainern auf den Platten des Datenbank-Computers mit Größe und Lage
4. **Externer Datenbankentwurf**
5. **Realisierung des Entwurfs in dem konkreten DBVS**

ERM Beispiel

Jeder Mitarbeiter gehört zu einer Abteilung

Abteilungen werden von Mitarbeitern geleitet

Mitarbeiter können miteinander verheiratet sein

Mitarbeiter können Gruppenleiter sein

Mitarbeiter sind:

- Manager
- Sekretäre
- Entwickler
- Techniker

Projekte benötigen Mitarbeiter

Die Teilnahme an einem Projekt setzt bestimmte Qualifikationen voraus

Starke und schwache Entities im Projektmanagementbeispiel

Entity	Attribut	Erläuterung
Mitarbeiter	MaNr	Mitarbeiter-Nummer
	Name	Name des Mitarbeiters
	Vorname	Vorname des Mitarbeiters
	PLZ	Postleitzahl
	Ort	Wohnort
	Straße	Straße
	Geb-Dat	Geburtsdatum
	Ein-Dat	Eintrittsdatum
	Gehalt	Gehalt
	Qualifikation	Qualifikation des Mitarbeiters

Starke und schwache Entities im Projektmanagementbeispiel

Abteilung	AbtNr	Abteilungs-Nummer
	AbtBez	Abteilungs-Bezeichnung
	AbtLtr	Abteilungs-Leiter
Projekt	ProjNr	Projekt-Nummer
	ProjBez	Projekt-Bezeichnung
	ProjStart	Projekt-Start
	ProjEnde	Projekt-Ende
	Status	Projekt-Status
	Qualifikation	Erforderliche Qualifikation

Tab. 2: Intuitiver Entwurf einer Projektmanagement-Datenbank

Erste Schwachstellen des Entwurfs

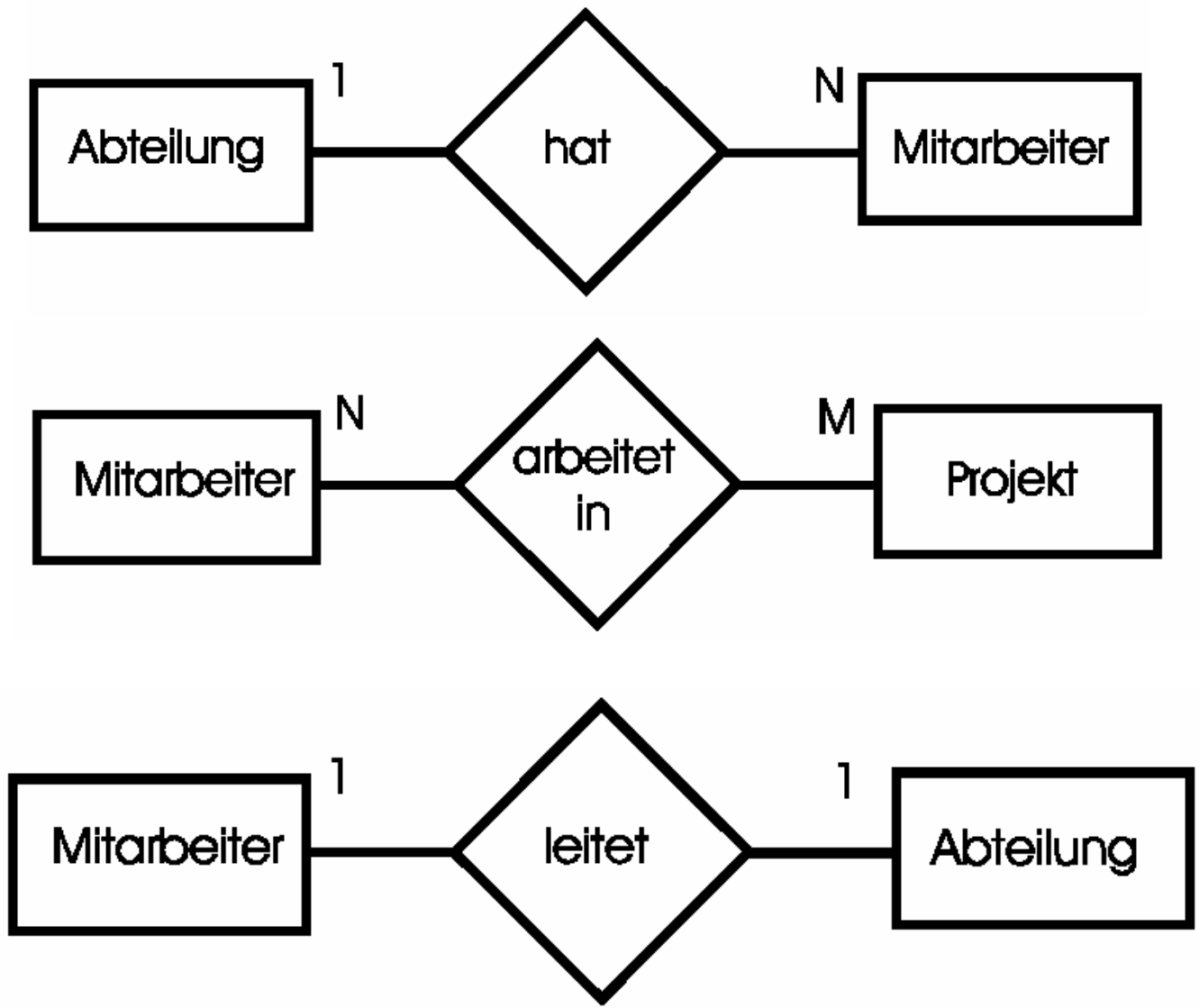
Unterschiedliche Mitarbeiter

Mitarbeiter sind:

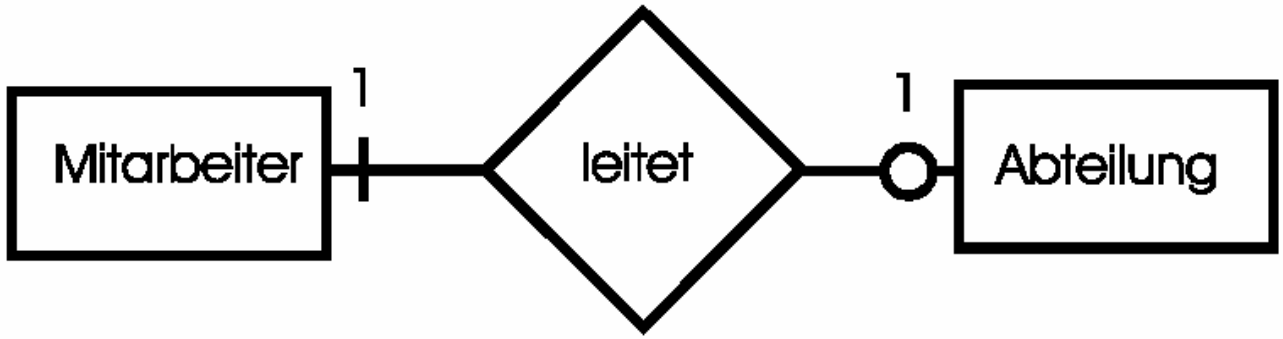
- Manager z.B. mit variablem Gehalt
- Sekretäre z.B. mit Vertretung
- Entwickler z.B. mit Heimarbeitszeitanteil
- Techniker z.B. mit Überstundenregelung

und alle mit unterschiedlichen Kostensätzen

Typ der Beziehung (Kardinalität)



Existenz



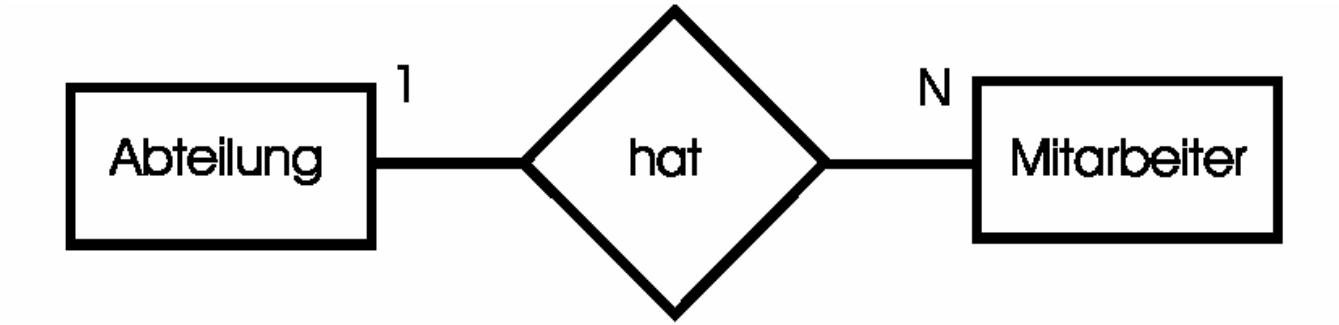
Für diese Beziehung gilt, daß ein Mitarbeiter eine Abteilung leiten **kann** (Kreis), während eine Abteilung von einem Mitarbeiter geleitet werden **muß** (Strich).

Falls unbekannt ist, ob es sich um eine kann- oder muß-Beziehung handelt, wird der Kreis bzw. der Strich weggelassen.

Zu beachten ist die Position des kann- bzw. muß-Symbols.

Grad der Beziehung - binär

binär (2-stellig)
Es sind 2 Entities an der Beziehung beteiligt.



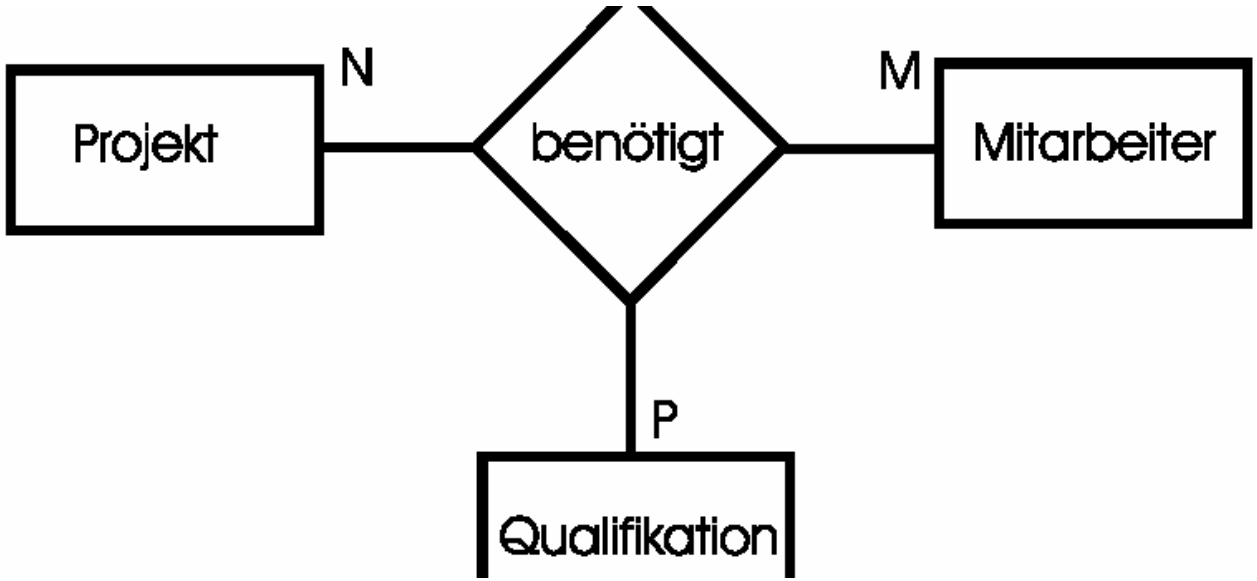
Grad der Beziehung - ternär

ternär (3-stellig)

Es sind 3 Entities an der Beziehung beteiligt.

Ternäre Beziehungen sollten, wenn möglich, in binäre Beziehungen umgewandelt werden. Beispiel:

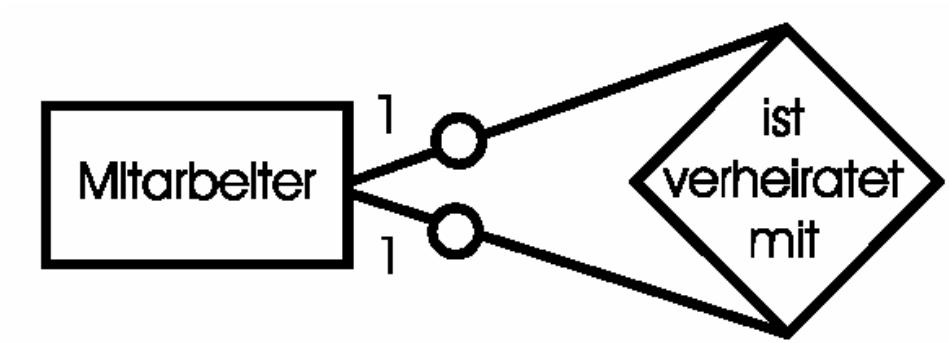
Zu einem bestimmten Projekt werden mehrere Mitarbeiter mit bestimmten Qualifikationen benötigt. Ein Mitarbeiter kann in mehreren Projekten arbeiten, braucht dazu jedoch verschiedene Qualifikationen. Eine Qualifikation kann in mehreren Projekten benötigt bzw. von mehreren Mitarbeitern erfüllt werden.



Grad der Beziehung – rekursiv binär

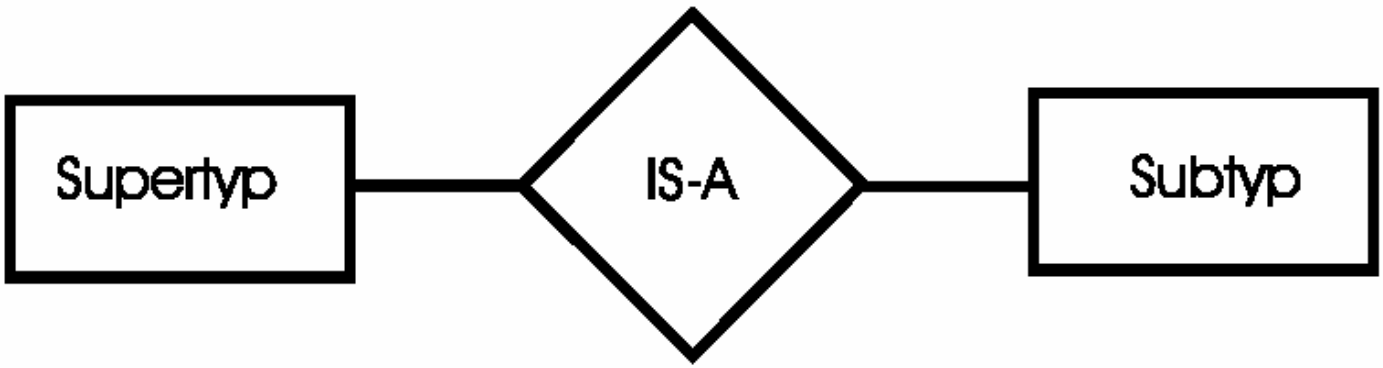
rekursiv binär

Es ist nur 1 Entity an der Beziehung beteiligt, die Beziehung ist jedoch rekursiv.



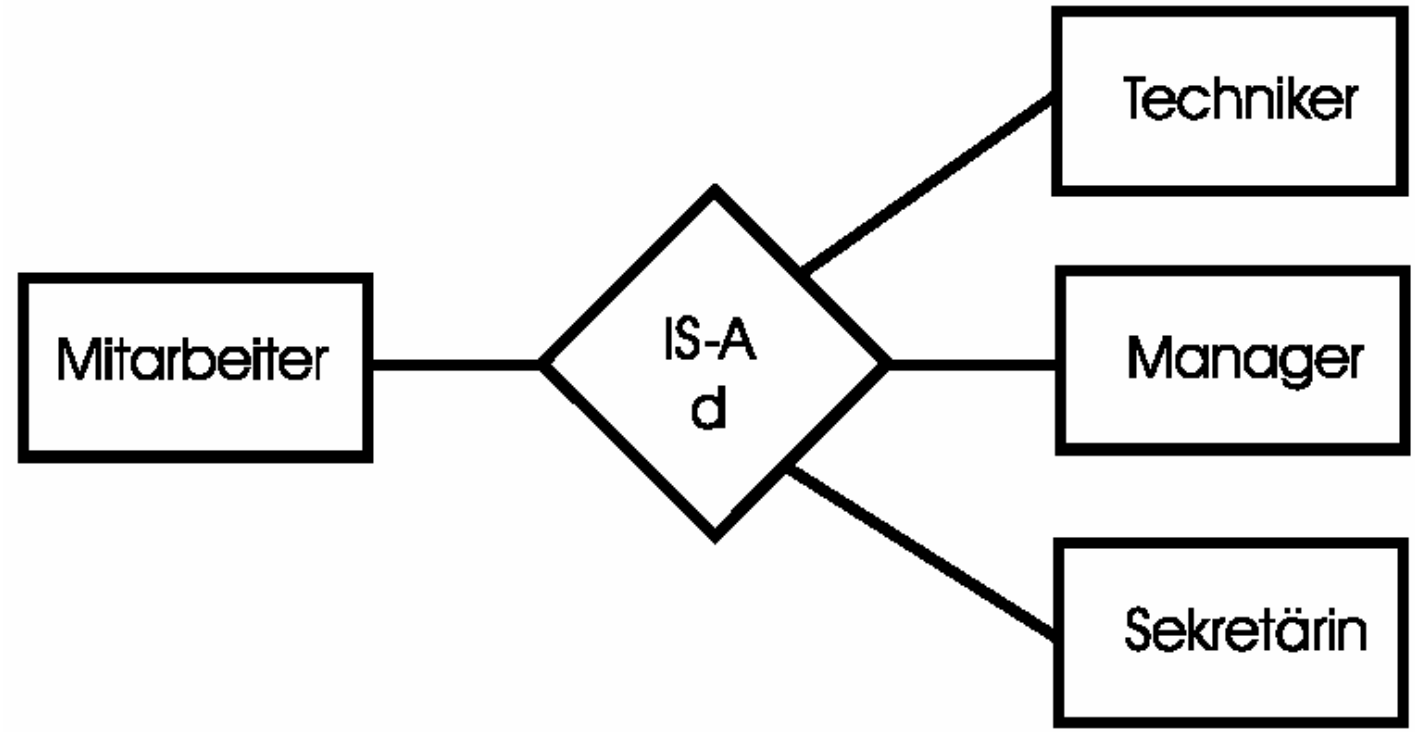
Generalisierung / Spezialisierung (IS-A)

Dieser Typ liegt vor, wenn eine Teilmenge (Subtyp) weitere Attribute gegenüber der Grundmenge (Supertyp) hat. Die so entstehenden Teilmengen können die Grundmenge total oder partiell überdecken sowie disjunkt oder nicht disjunkt sein.
In der Mengenlehre heißen zwei Mengen *A* und *B* **disjunkt** oder **elementfremd**, falls sie kein gemeinsames Element besitzen.



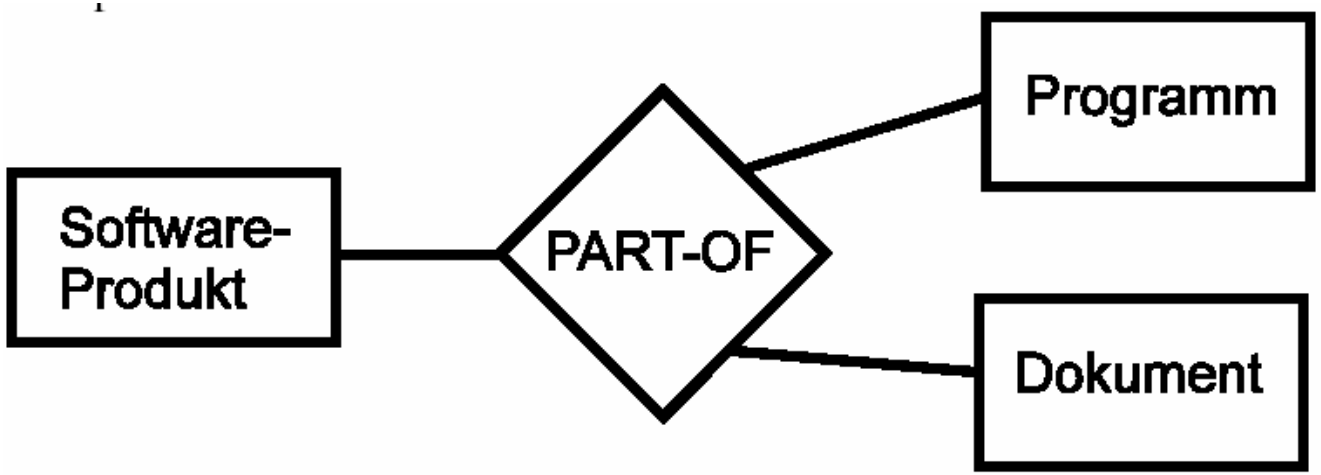
Generalisierung / Spezialisierung (IS-A)

Beispiel:
Ein Mitarbeiter kann ein Techniker, ein Manager oder eine Sekretärin sein. In allen Fällen ist er/sie zunächst ein Mitarbeiter mit den betreffenden Attributen. Zusätzlich hat jeder noch weitere Attribute, die den jeweiligen Subtyp näher beschreiben. Das d unter IS-A gibt an, daß die Teilmengen disjunkt sind.



Part-of

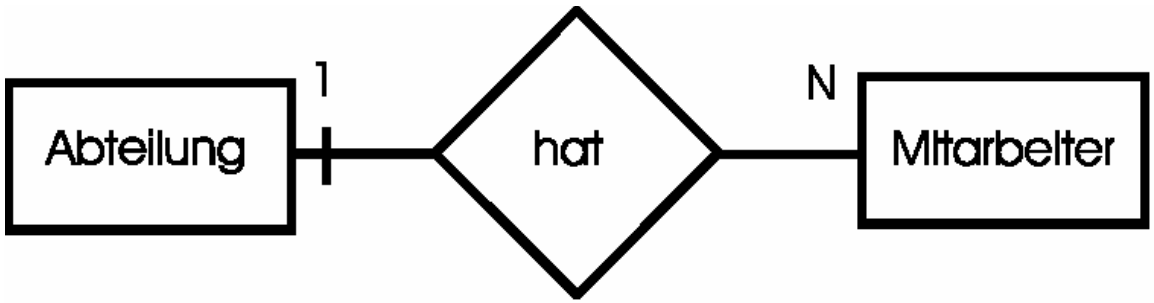
Dieser Typ liegt vor, wenn ein Entity aus mehreren eigenständigen Entities zusammengesetzt ist.



Regeln für die Ableitung von Tabellen

1:N Beziehung (obligatorisch)

Eine 1:N-Beziehung wird in zwei Tabellen abgebildet, wenn die Teilnahme der N-Entity obligatorisch (Muß-Beziehung) ist. Der Primärschlüssel der 1-Seite wird Fremdschlüssel in der anderen Tabelle.

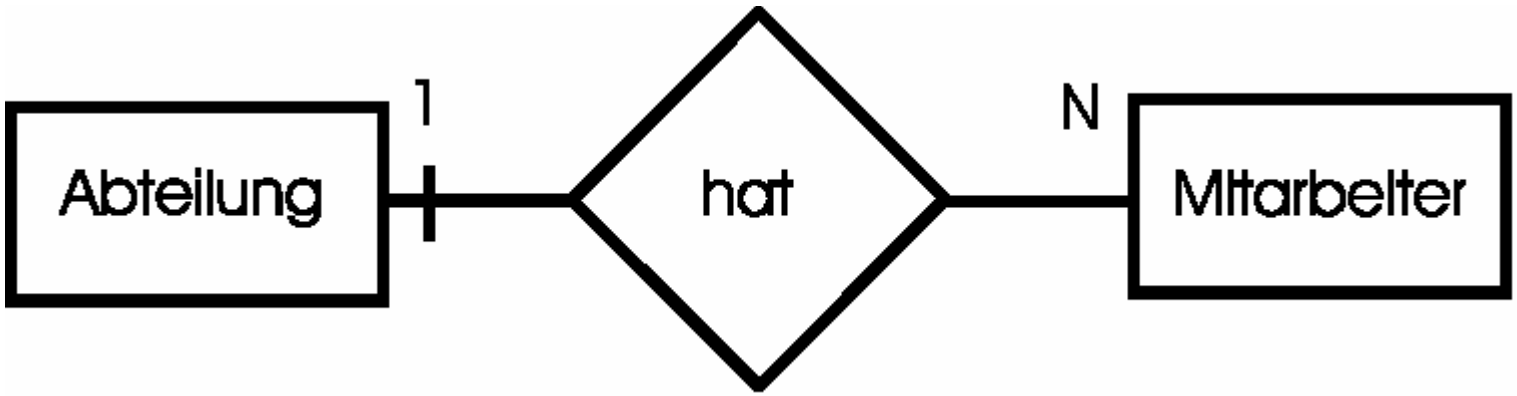


wird die AbtNr, der Primärschlüssel von Abteilung, zum Fremdschlüssel in der Tabelle Mitarbeiter.

Regeln für die Ableitung von Tabellen

1:N Beziehung (obligatorisch)

Eine 1:N-Beziehung wird in zwei Tabellen abgebildet, wenn die Teilnahme der N-Entity obligatorisch (Muß-Beziehung) ist. Der Primärschlüssel der 1-Seite wird Fremdschlüssel in der anderen Tabelle.

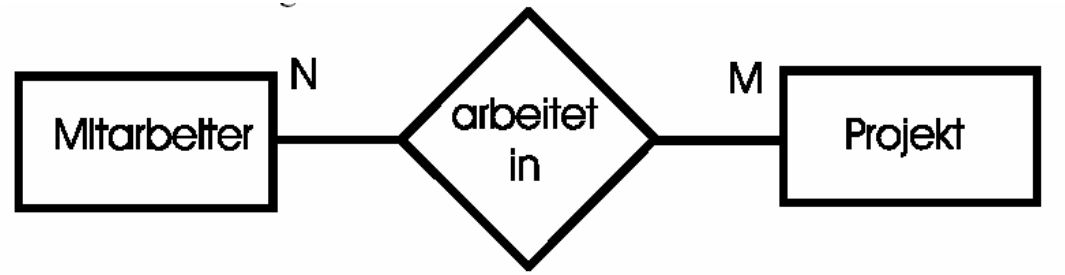


wird die AbtNr, der Primärschlüssel von Abteilung, zum Fremdschlüssel in der Tabelle Mitarbeiter.

Regeln für die Ableitung von Tabellen

M:N Beziehung

Für eine solche Beziehung wird eine zusätzliche Beziehungs-Tabelle eingeführt (also drei Tabellen). Die Primärschlüssel der Grundtabellen werden zu Fremdschlüsseln in der Beziehungstabelle und bilden dort i. a. den neuen Primärschlüssel.



Hier wird eine neue Tabelle:

ProjektMitarbeit (ProjNr, MaNr, AnzahlStunden)

angelegt.

Damit wird die N:M-Beziehung auf zwei 1:N-Beziehungen zwischen Mitarbeiter und ProjektMitarbeiter bzw. zwischen Projekt und ProjektMitarbeiter reduziert.

Regeln für die Ableitung von Tabellen

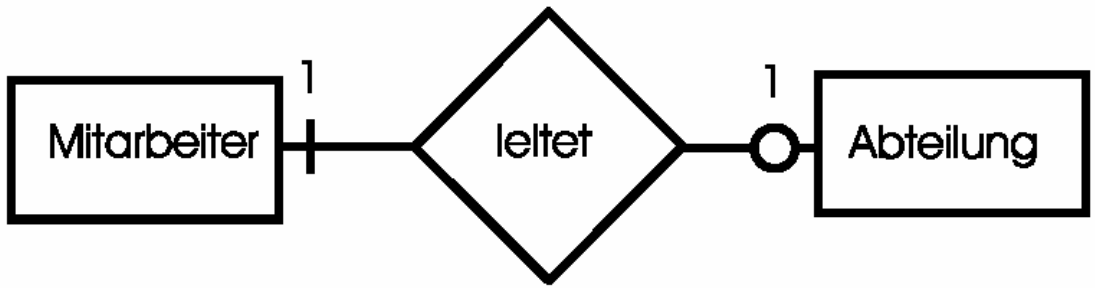
1:1 Beziehung

Eine 1:1-Beziehung wird in einer Tabelle abgebildet, wenn die Teilnahme beider Entity-Mengen obligatorisch ist. Beide Entity-Schlüssel sind Schlüsselkandidaten der Tabelle, wobei festgelegt werden muß, welcher der beiden der Primärschlüssel wird.

Eine 1:1-Beziehung wird in zwei Tabellen abgebildet, wenn die Teilnahme von nur einer der beiden Entity-Mengen obligatorisch ist. Der Primärschlüssel der obligatorischen Seite wird Fremdschlüssel in der anderen Tabelle.

Regeln für die Ableitung von Tabellen

1:1 Beziehung

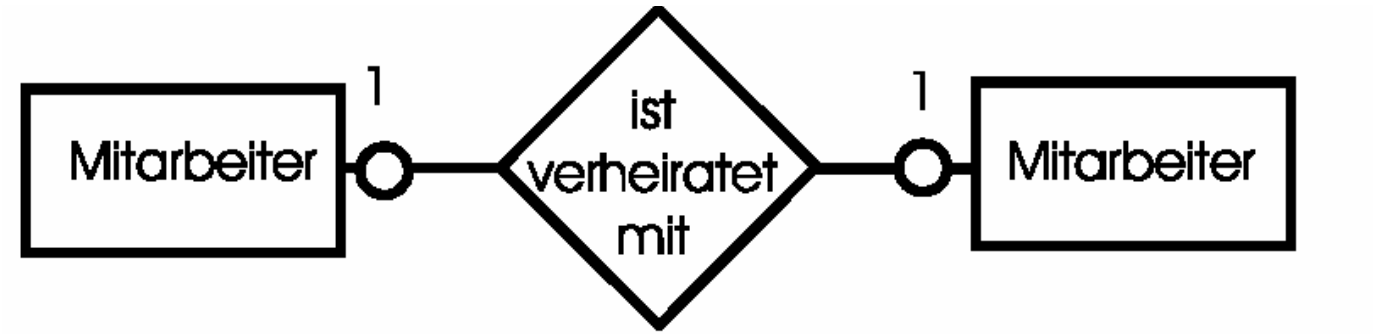


Hier werden also zwei Tabellen angelegt, wobei die MaNr aus Mitarbeiter (also die MaNr des Abteilungs-Leiters) zum Fremdschlüssel in Abteilung wird.

Regeln für die Ableitung von Tabellen

1:1 Beziehung

Eine 1:1-Beziehung kann in drei Tabellen abgebildet werden, wenn die Teilnahme beider Entity-Mengen optional ist. Die Beziehung wird in einer zusätzlichen Tabelle dargestellt, die nur die tatsächlichen Beziehungsausprägungen enthält. Der Primärschlüssel der dritten Tabelle setzt sich i. a. aus den Primärschlüsseln der beiden Grundtabellen zusammen, die dann auch Fremdschlüssel sind.
In folgendem Beispiel



sollte eine zusätzliche Tabelle

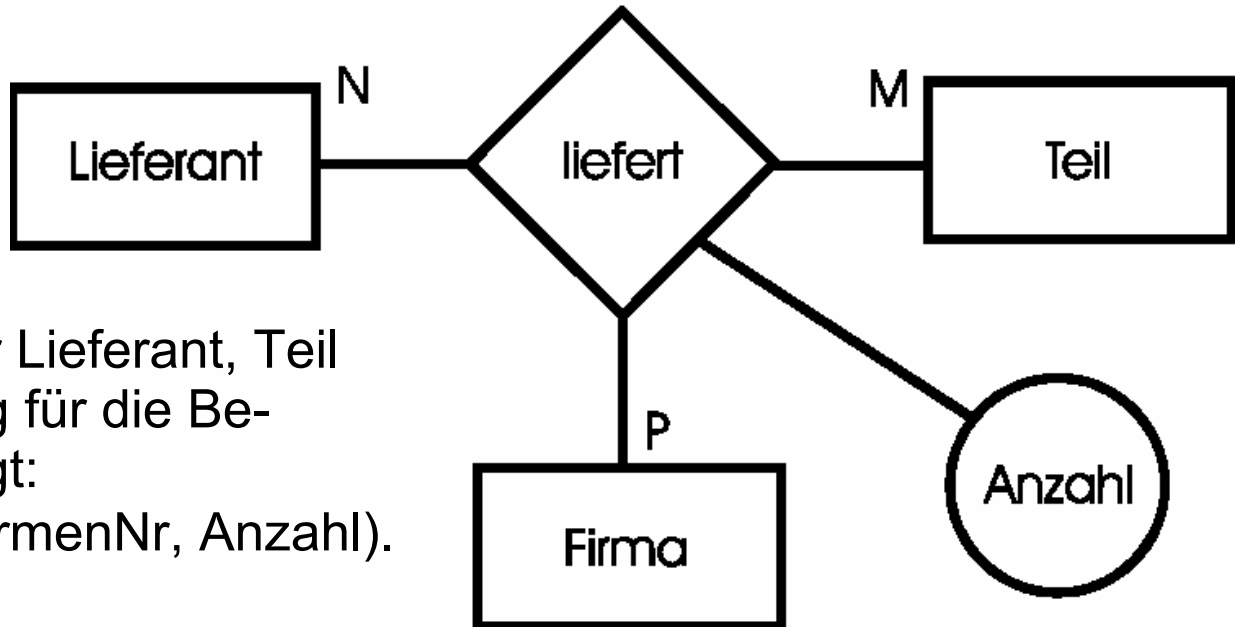
MitarbeiterEhe (MaNr1, MaNr2, verheiratet_seit)

angelegt werden, da eine solche Ehe relativ selten ist. Diese Tabelle wäre dann die „dritte“ Tabelle, da die Mitarbeiter-Tabelle durch die rekursive Beziehung quasi doppelt gezählt werden kann.

Regeln für die Ableitung von Tabellen

Ternäre Beziehung

Bei ternären Beziehungen werden insgesamt vier Tabellen angelegt. Zu den Grundtabellen kommt noch eine Beziehungstabelle hinzu, die sämtliche Primärschlüssel der Grundtabellen als Fremdschlüssel enthält.



Hier wird neben den Tabellen für Lieferant, Teil und Firma eine Tabelle Lieferung für die Beziehung „liefert“ wie folgt angelegt:
Lieferung (LieferantNr, TeilNr, FirmenNr, Anzahl).

Regeln für die Ableitung von Tabellen

IS-A Beziehung

Zusätzlich zur Tabelle für die Grundmenge wird eine weitere Tabelle für jede Teilmenge angelegt, die denselben Primärschlüssel wie die Grundmenge hat. Im Beispiel mit Mitarbeiter IS-A Techniker werden folgende Tabellen definiert:

Mitarbeiter (MaNr, Name, Vorname, ...)
Techniker (MaNr, Überstunden)

Regeln für die Ableitung von Tabellen

Part-Of Beziehung

Bei solchen Beziehungen wird je eine Tabelle für den Super- und den Subtyp angelegt. Die Beziehung selbst wird nur dokumentiert, nicht jedoch per Fremdschlüssel dargestellt.

Regeln für die Ableitung von Tabellen

Kann – Muß Beziehung

Abhängig davon, ob es sich um eine muß- oder kann-Beziehung handelt, muß die Fremdschlüsselspalte einen Wert haben oder nicht.

Regeln für die Ableitung von Tabellen

Rekursive Beziehungen

Im Fall der Stückliste (Teil ist Oberteil zu Teil mit Zusatzattribut Anzahl; N:M) werden zwei Tabellen angelegt, die wie folgt aussehen können:

Teile (TeileNr, Bezeichnung, Gewicht, ...)
Struktur (OberTeileNr, UnterTeileNr, Anzahl)

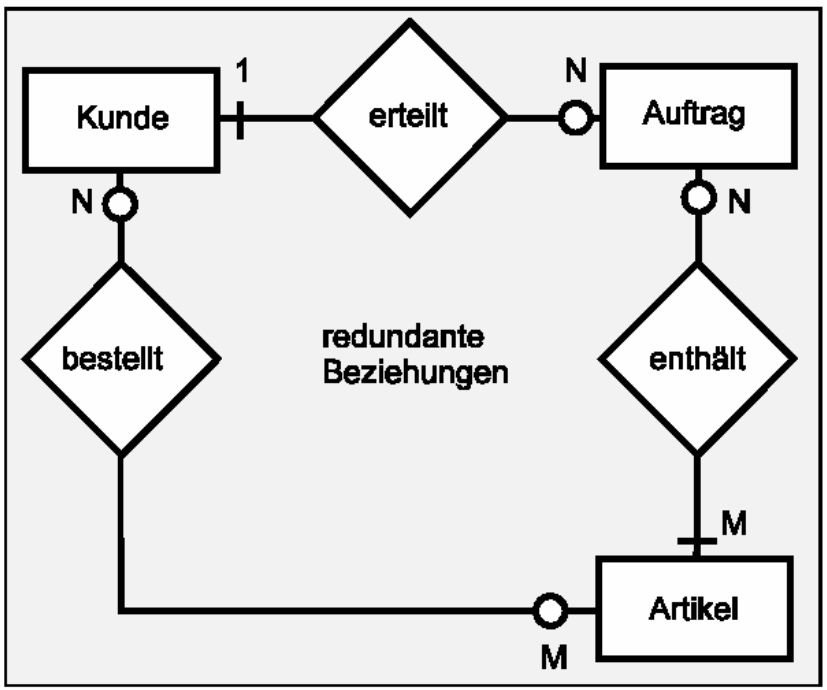
Bei der Gruppenleiter-Beziehung (Mitarbeiter leitet Mitarbeiter; 1:N) kann folgende Tabelle angelegt werden:

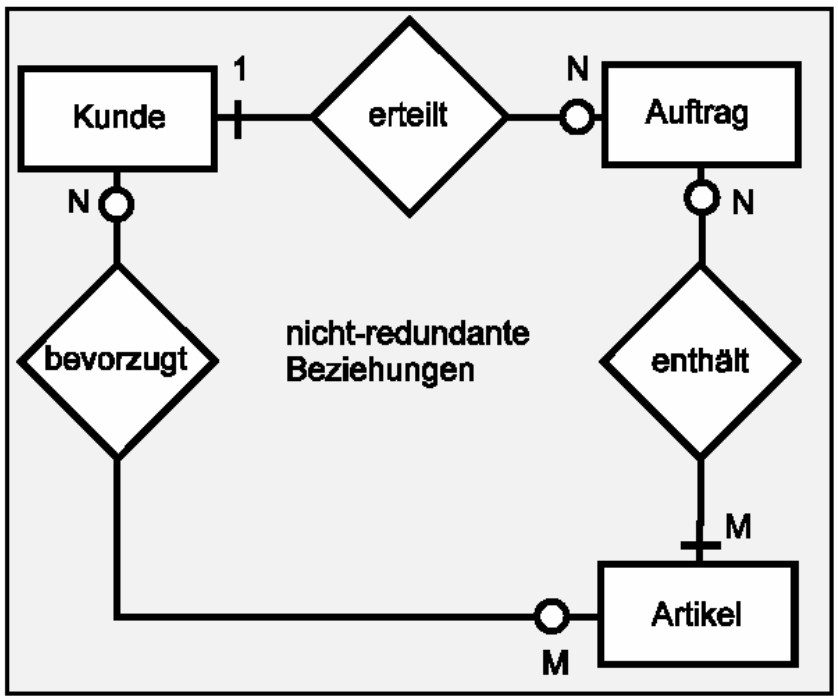
Mitarbeiter (MaNr, Name, Beruf, AbtNr, Gehalt, MaNrGrpLeiter)
MaNrGrpLeiter ist die MaNr des jeweiligen Gruppenleiters

Regeln für die Ableitung von Tabellen

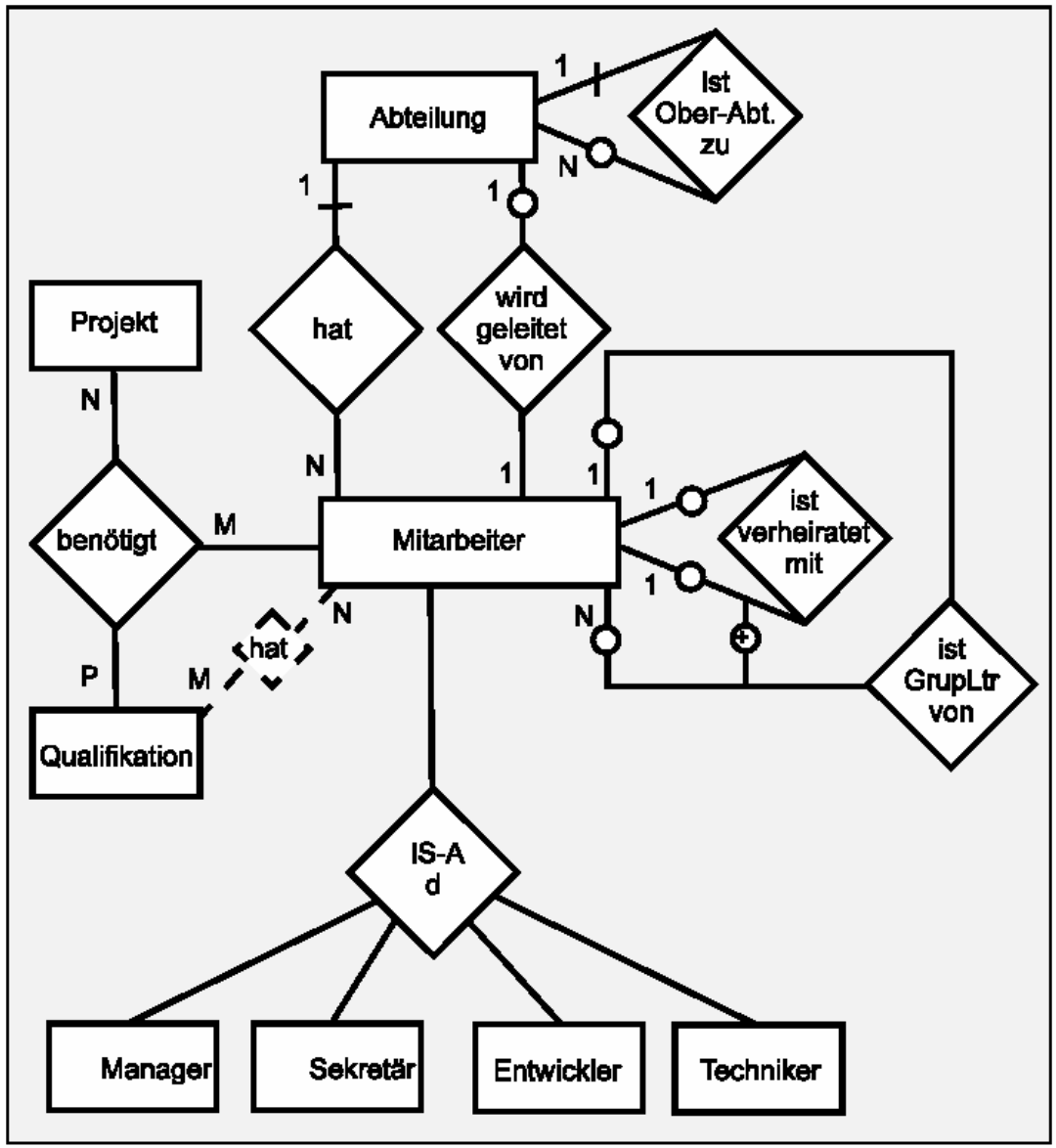
Redundante Beziehungen

Redundante Beziehungen dürfen nicht in Tabellen umgesetzt werden, da sie sonst zu Datenredundanzen führen können. Sie werden dann besser erkannt, wenn man sämtliche Beziehungen in einem einzigen Beziehungsgraphen darstellt. Überall dort, wo ein geschlossener Beziehungskreis vorliegt, muß auf Redundanz untersucht werden.





Vollständiges ERM der Projektaufgabe



Tabellenstruktur der Projektaufgabe

Tabelle	Primärschlüssel	Attribute
Abteilung	AbtNr	Bezeichnung, Aufgabe, ü-bergeordnAbtNr (1), AbtLtr
Mitarbeiter	MaNr	Name, Vorname, AbtNr
Qualifikation	Qualifikation	
Projekt	ProjNr	ProjBez, Start, Ende, Status
ProjektMitarbeit	ProjNr, MaNr,	Summe_Stunden
	Qualifikation	
MaEhen	Mann, Frau (2)	verheiratet_seit, An-zahl_Kinder
Gruppe	MaNr, GrpLtr (3)	seit_wann
Manager	MaNr	Manager_seit, var_Gehalt, ...
Techniker	MaNr	Spezialgebiet, Q_Stufe, ...
MaQualifikation	MaNr, Qualifik.	Q-Stufe

Normalisierung

Normalisierung = Bottom-Up Methode
ERM = Top-Down Methode

Ziele der Normalisierung
- Beseitigung von Redundanzen
- Beseitigung von Anomalien (Lösch-, Einfüge-, Updateanomalie)

Vorgehensweise:
- Auslagerung von Attributen in kleinere Einheiten
- Auflösung von Abhängigkeiten
- ggf. Streichung von Attributen

Normalformen
- Regeln für die Normalisierung
- Wir behandeln 1 – 4 Normalform

1. Normalform

Eine Tabelle ist in der 1. Normalform (1NF), wenn sie sich als lineare Tabelle darstellen läßt, d. h. es gibt keine Tabelle in der Tabelle. Jedes Feld ist ein einfacher Wert. Dies kann immer durch Auslagerung von unterstrukturierten Feldern zu anderen/eigenständigen Tabellen erreicht werden. Der Bezug auf diese neue Tabelle wird durch Fremdschlüssel hergestellt.

AuftragsNr	Datum	KundenNr	Name	Ort	ArtNr	Bez	Menge

Auftragsnummer ist der Primärschlüssel
Jede Spalte enthält atomare Werte

2. Normalform

Eine Tabelle ist in 2NF, wenn sie in 1NF ist und jedes Nicht-Schlüsselattribut von allen Schlüsselattributen abhängig ist. Mit Schlüssel ist hier der Primärschlüssel gemeint. Dies wird erreicht, indem die nicht abhängigen Nicht-Schlüsselattribute in eigene Tabellen ausgelagert werden.

Die 1. Normalform wird aufgeteilt nach Objekten, z.B. hier Kunde und Artikel
Vollständig funktional abhängig bedeutet, dass ein Nicht-Schlüsselattribut vom kompletten (falls zusammengesetzt) Primärschlüssel abhängig sein muss
Jedes Nicht-Schlüsselfeld muss über den Primärschlüssel identifizierbar sein.

Die zweite Normalform betrifft grundsätzlich nur zusammengesetzte Primär-schlüssel

3. Normalform

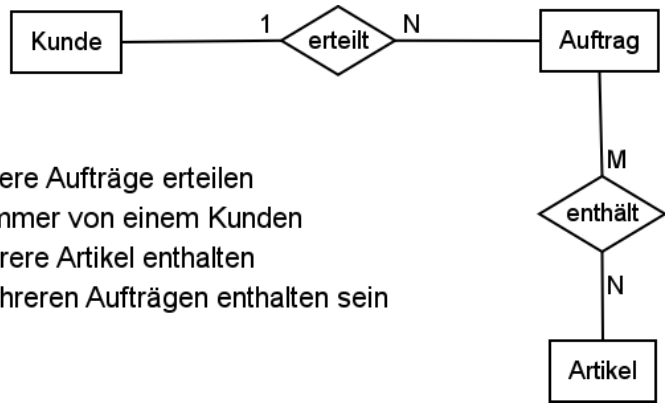
Eine Tabelle ist in 3NF, wenn sie in 2NF ist und jedes Nicht-Schlüsselattribut von allen Schlüsselattributen nicht transitiv abhängig ist.
Kurz: Die Nicht-Schlüsselfelder müssen unabhängig voneinander sein (Mit Schlüssel ist wieder der Primärschlüssel gemeint).
Die 3NF wird wieder durch Auslagerung erreicht.

Tabelle Kunde		
KundenNr	Name	Ort

Tabelle Artikel	
ArtNr	Bezeichnung

Tabelle Bestellung		
AuftragsNr	KundenNr	Datum

Tabelle Stückliste		
AuftragsNr	Menge	ArtNr



1 Kunde kann mehrere Aufträge erteilen
1 Auftrag stammt immer von einem Kunden
1 Auftrag kann mehrere Artikel enthalten
1 Artikel kann in mehreren Aufträgen enthalten sein

4. Normalform

Eine Tabelle ist in 4NF, wenn sie in 3NF ist und keine aus der Datenbank abgeleiteten Felder, z. B. durch Berechnung, enthält. Dabei muß auch untersucht werden, ob sich ein Feld nicht aus Feldern anderer Tabellen ableiten läßt.

Datenbanksprache SQL

SQL ist eine standardisierte Sprache (ANSI) zur Manipulation und zum Zugriff auf Datenbanken.

Mit Hilfe von SQL-Statements werden Daten geändert sowie ausgelesen.
Mit Hilfe von SQL können alle gängigen DBMS manipuliert werden, z.B.
MS Access, DB2, Informix, MS SQL Server, Oracle, Sybase, MySQL usw.

Es gibt verschiedene Standards, z.B. SQL 89, SQL 92, SQL99 und bald auch SQL 2003

Auch haben die DBMS eigene Erweiterungen des Standards, aber die Grundfunktionen werden immer unterstützt.

Datenbanksprache SQL

SQL steht für Structured Query Language

SQL erlaubt den Zugriff auf eine Datenbank

SQL ist ein standardisierte (ANSI) Computersprache

SQL kann Abfragen ausführen

SQL kann Daten aus einer Datenbank holen

SQL kann neue Datensätze einfügen

SQL kann Datensätze löschen

SQL kann Daten ändern

SQL ist leicht zu lernen

Select

Stelle alle Kunden dar

```
select * from Kunde;
```

ALL	Alle Werte, auch mehrfach vorkommende, sollen angezeigt werden
DISTINCT	Es sollen nur verschiedene Werte angezeigt werden
ASC	Sortierreihenfolge aufsteigend (Default- Einstellung)
DESC	Sortierreihenfolge absteigend

Suche Kundenkürzel und Firmenname aller Kunden, sortiert nach Firmenname

```
SELECT KD_NR, FIRMA FROM KUNDE ORDER BY FIRMA;
```

Suche die unterschiedlichen Postleitzahlen aller Kunden

```
SELECT DISTINCT PLZ FROM KUNDE;
```

Select – Kommando (Fortsetzung)

Suche Ort und Firmenname der Kunden, sortiert nach Ort, falls Ort doppelt, dann zusätzlich sortiert nach Firmenname

```
SELECT   ORT, FIRMA
FROM     KUNDE
ORDER BY ORT, FIRMA
```

Suche alle Kunden mit Kundennummer > 105

```
SELECT   KD_NR, FIRMA, LAND, ORT
FROM     KUNDE
WHERE    KD_NR > 105
```

Suche alle Kunden mit Kundennummer > 105 und Land = D

```
SELECT   KD_NR, FIRMA, LAND, ORT
FROM     KUNDE
WHERE    KD_NR > 105 and LAND = "D"
```

Vergleichsoperatoren

= gleich
< kleiner als
> grösser als
<= kleiner gleich
=> grösser gleich
<>ungleich

– alphanumerische Konstanten sind mit " " oder ' ' anzugeben
– Gross- und Kleinbuchstaben sind unterschiedlich
– ANSI- und ASCII-Vergleich liefern evtl. unterschiedliche Werte

ANSI und ASCII sind zwei verschiedene Zeichensätze für reine Textdateien

American **S**tandard **C**ode for **I**nformation **I**nterchange

Verknüpfung von Vergleichsoperatoren

Mehrere Vergleichsoperatoren können mittels logischer Operatoren miteinander verknüpft werden

Operator	Beschreibung
NOT	Verneinung der Bedingung
AND	Logisches UND
OR	Logisches ODER
XOR	Logische Antivalenz (dt: Entweder A oder B)
EQV	Logische Äquivalenz (zeichenweiser Bitvergleich)
IMP	Logische Implikation (Wenn-Dann-Beziehung)

In einem Vergleich können mathematische Ausdrücke mit *, /, +, -, \(Div), ^(Potenz) und Mod verwendet werden.

& - Operator verknüpft Zeichenketten
"Hans"& "" & "Hugo" ergibt "Hans Hugo"

SELECT – Kommando (Fortsetzung)

Suche die Aufträge mit KDNR = 106 und Auftragsstatus-Nr = 3 oder 4 (erledigt oder Ablage):

```
SELECT    KD_NR, A_NR, AUFTRAG_TEXT, STDSATZ_NR
FROM      AUFTRAG
WHERE     KD_NR = 106
AND       (STDSATZNR = 3 OR STDSATZNR = 4);
```

Suche die Aufträge mit Fertigsoll bis zum 31.12.97:

```
SELECT    KD_NR, A_NR, AUFTRAGTEXT, STDSATZ_NR, FERTIGSOLL
FROM      AUFTRAG
WHERE     FERTIGSOLL <= #12/31/97#;
```

Hinweis:

In SQL-Anweisungen in Access Basic müssen US-/englische Datums-formate verwendet werden. Im QBE-Entwurfsbereich von Access können jedoch die internationalen Datumsformate verwendet werden, z. B. <= #31.12.1997# (wobei # von Access automatisch ergänzt wird).

SELECT mit BETWEEN, IN und LIKE

Mit dem Operator BETWEEN Wert1 AND Wert2

kann folgende Bedingung

```
WHERE FERTIGIST >= #01/01/97# AND FERTIGIST <= #12/31/97#;
```

auch wie folgt spezifiziert werden:

```
WHERE FERTIGIST BETWEEN #01/01/97# AND #12/31/97#;
```

Mit dem Operator IN (Wert1, Wert2,,,) kann eine Liste von Werten zum Vergleich benutzt werden (entspricht einem mehrfachen OR).

Mit dem Operator LIKE für alphanumerische Zeichenketten kann nach Zeichenmustern gesucht werden. Dabei können Platzhalterzeichen verwendet werden.

Platzhalterzeichen

Zeichen	Steht für
? bzw. _	Ein beliebiges einzelnes Zeichen * bzw. % Null oder mehrere beliebige Zeichen
#	Eine beliebige einzelne Ziffer (0-9)
[ZeichListe]	Ein beliebiges Zeichen aus der ZeichListe, z. B. "[AD]*": alle Werte, die mit A oder D beginnen. "[A-D]*": alle Werte, die mit A, B, C oder D beginnen.
[!ZeichListe]	Ein beliebiges Zeichen außerhalb von ZeichListe

Um herauszufinden, ob in einer Spalte kein Wert eingetragen ist, muß

`WHERE` Spaltenname **`IS NULL`**

formuliert werden.

Die Verneinung von **`BETWEEN`**, **`LIKE`** und **`IN`** wird durch ein vorangestelltes **`NOT`** realisiert (z. B. Spaltenname **`NOT LIKE`** Wert); für **`IS NULL`** lautet die Verneinung dagegen **`IS NOT NULL`**

Platzhalterzeichen (Beispiele)

Suche die Aufträge der Kunden 106 und 110

```
SELECT      KD_NR, A_NR, AUFTRAG_TEXT, STUNDENSATZ,
            FERTIGIST
FROM        AUFTRAG
WHERE       KD_NR IN (106,110)
ORDER BY    KD_NR;
```

Suche die Aufträge, die im Auftragstext die Zeichenkette "DIAG" enthalten

```
SELECT    KD_NR, A_NR, AUFTRAG_TEXT, STUNDENSATZ, FERTIGIST
FROM      AUFTRAG
WHERE     AUFTRAGTEXT LIKE "*DIAG*";
```

Suche die Aufträge, die noch nicht fertig sind

```
SELECT    KD_NR, A_NR, AUFTRAG_TEXT, STUNDENSATZ, FERTIGIST
FROM      AUFTRAG
WHERE     FERTIGIST IS NULL;
```

SELECT mit Funktionen, Ausdrücken, virtuellen Spalten

Bei der Formulierung eines Queries können statistische Werte abgefragt werden. Dazu stehen folgende Funktionen zur Verfügung, die sich jeweils auf die Ergebnismenge der Abfrage (bzw. der Gruppe, siehe unten) beziehen. Im Fall einer Abfrage darf dann jedoch nur eine Spalte selektiert werden.

COUNT	ermittelt die Anzahl der Treffer inkl. der NULL-Werte
SUM	ermittelt die Summe der numerischen Werte
AVG	ermittelt den Durchschnitt der numerischen Werte
MAX	ermittelt den maximalen Wert aller Werte
MIN	ermittelt den minimalen Wert aller Werte
StdAbw	ermittelt die Standard-Abweichung in der Stichprobe
StdAbwG	dito in der Grundmenge
Varianz	ermittelt die Varianz in der Stichprobe
VarianzG	dito in der Grundmenge

SELECT mit Funktionen und Ausdrücken

Beispiele:

Wieviele Kunden sind in der Datenbank?

```
SELECT COUNT(*) FROM KUNDE;
```

Welches ist die größte Kundennummer?

```
SELECT MAX(KD_NR) FROM KUNDE;
```

Durchschnitt der Kundennummern ;-)

```
SELECT AVG(KD_NR) FROM KUNDE;
```

Gesamtsumme Aufträge

```
SELECT SUM(AUFTRAGSWERT) FROM AUFTRAG;
```

SELECT mit virtuellen Spalten (AS)

Bei vielen Datenbanksystemen können bei der Ausgabe eines interaktiven SQL-Kommandos die Spaltenüberschriften mit AS geändert werden (Aliasnamen für Spalten), z. B.

Ermittle Auftragskürzel, Kundenkürzel und Auftragsdatum der Aufträge:

```
SELECT   A_NR AS Auftragsnummer, KD_NR AS Kundennummer,
         A_DATUM AS Auftragsdatum
FROM     AUFTRAG;
```

Es können auch zusätzliche Spalten (Virtuelle Spalten) in der Ergebnismenge auftreten, die nicht in der Tabelle enthalten sind; sie werden durch Berechnung o. ä. gebildet, z. B.

Ermittle die Auftragswerte der Aufträge:

```
SELECT A_NR, ANZ*SATZ AS Auftragswert FROM AUFTRAG;
```

Ermittle die Mehrwertsteuerbeträge der Aufträge:

```
SELECT   A_NR, KD_NR, ANZ*SATZ*MWSATZ AS Mehrwertsteuer
FROM     AUFTRAG;
```

SELECT mit virtuellen Spalten (AS) II

Zwei oder mehrere Spalten können zu einer (virtuellen) Ergebnisspalte zusammengefaßt werden:

```
SELECT    PLZ & " " & ORT AS Wohnort
FROM      KUNDE;
```

```
SELECT    A_NR, FERTIGIST, FERTIGSOLL,
          FERTIGIST - FERTIGSOLL AS Verspätung (in Tagen)
FROM      AUFTRAG;
```

AKURZ	FERTIGIST	FERTIGSOLL	Verspätung
109	01.05.96	01.05.96	0
111		30.11.98	
112	13.04.97	28.02.97	44
113	04.02.98	15.12.97	51
114		12.07.98	
115	02.09.97	30.07.97	34
116			

SELECT mit virtuellen Spalten (AS) III

```
SELECT      VORNAME, NACHNAME, PNR AS Personalnun
            mer
FROM        PERSON
WHERE       PNR < 10
ORDER BY    NACHNAME;
```

VORNAME	NACHNAME	Personalnummer
Karl-Heinz	Adam	3
Werner	Assmann	7
Karl	Auffahrt	2
Ewald	Beck	4
Klaus	Bettag	1
	Bomhoff	9
	Henning	8
	Kummer	6
Adolf	Schultz	5

SELECT mit virtuellen Spalten (AS) IV

```
SELECT      KD_NR, A_NR, TEXT, ANZ*SATZ AS Netto,
            ANZ*SATZ*(1+MWSATZ) AS Brutto
FROM        AUFTRAG
ORDER BY    ANZ*SATZ;
```

sql122a : Auswahlabfrage

KD_NR	A_NR	TEXT	Netto	Brutto
110	116	Einführung	800	800
109	109	Serienbrief-Einweisung	1.500	1.740
110	113	Kunden-Verwaltung	5.600	5.600
106	115	Datenbank-Entwurf DIAG	6.500	7.540
101	114	ORACLE-Rezension	9.000	9.630
110	112	Offertenverwaltung	9.600	9.600
100	111	Mitarbeiterschulung	18.000	20.880

Mit Format (AANZ*ASATZ,"#.###") AS Netto wird eine zusätzliche Formatangabe zur Darstellung des Tausenderpunktes benutzt.

SELECT mit virtuellen Spalten (AS) V

```
SELECT    A_NR, FERTIGIST,
          FERTIGSOLL,
          FERTIGISTFERTIGSOLL AS [Sollzeit der Fertigstellung
          in Tagen],
          Date() AS heute,
          Date()-FERTIGIST AS [Tage seit Fertigstellung]
FROM      AUFTRAG
ORDER BY  Date()-FERTIGIST;
```

AKURZ	Fert.IST	Fert.SOLL	Sollzeit Fer-tigstell.	heute	Tage seit Fertig-stellung
116				17.09.97	
114		12.07.98		17.09.97	
111		30.11.98		17.09.97	
113	04.02.98	15.12.97	51	17.09.97	-140
115	02.09.97	30.07.97	34	17.09.97	15
112	13.04.97	28.02.97	44	17.09.97	157
109	01.05.96	01.05.96	0	17.09.97	504

SELECT mit virtuellen Spalten (AS) VI

```
SELECT    VORNAME & " " & NACHNAME AS Name, PNR
FROM      PERSON
WHERE     P_NR < 10
ORDER BY  NACHNAME;
```

Name	PNR
Karl-Heinz Adam	3
Werner Assmann	7
Karl Auffahrt	2
Ewald Beck	4
Klaus Bettag	1
Bomhoff	9
Henning	8
Kummer	6
Adolf Schultz	5

Gruppierungen

Mit GROUP BY wird eine Gruppenbildung bei SELECT durchgeführt. Gruppenbildungen werden i. a. vorgenommen, um Werte wie Summe, Maximum o.ä. für die ganze Gruppe zu berechnen und dann in einer Zeile für die Gruppe auszugeben.

Betrachten wir dazu ein Beispiel:
Ermittle die Anzahl der Aufträge mit verschiedenen Mehrwertsteuersätzen. Die Menge der Aufträge zu einem MWST-Satz bildet dabei jeweils eine Gruppe:

```
SELECT      MWSATZ, COUNT(*) As Häufigkeit
FROM        AUFTRAG
GROUP BY    MWSATZ;
```

MWSATZ	Häufigkeit
0	3
0,07	1
0,16	3

Die Gruppenspalte MWSATZ muß als Suchspalte hinter SELECT angegeben werden. Es wird automatisch nach der Gruppenspalte sortiert. Als Ergebnisspalten (hinter SELECT) sind nur Gruppenspalten, arithmetische Funktionen und virtuelle Spalten zugelassen.

Gruppierungen II

Man kann auch mehrere Spalten als Gruppenspalten definieren:

```
SELECT    MWSATZ, LSATZ, Count(*) AS Anzahl
FROM      LEISTUNG
GROUP BY  MWSATZ, LSATZ;
```

MWSATZ	LSATZ	Anzahl
0	20	1
0	1.200	3
0	1.400	1
0,07	125	1
0,07	300	2
0,16	10	1
0,16	75	1
0,16	125	4
0,16	1.500	2

Durch die zusätzliche Gruppenspalte LSATZ wird das Ergebnis erweitert; es werden auch mehr Zeilen angezeigt. Das liegt daran, daß jetzt jeweils eine Gruppe bezüglich der Kombination MWSATZ und LSATZ gebildet wird.

Gruppierungen III

Wichtig:
Hinter GROUP BY müssen mindestens dieselben Tabellenspalten wie hinter SELECT aufgeführt sein!
Sortiert wird zunächst nach der 1. Gruppenspalte, dann nach der 2. usw.

Mit HAVING wird eine Selektion hinsichtlich einer Gruppeneigenschaft vorgenommen. Dabei sind nur arithmetische Funktionen zugelassen.
Als Beispiel wollen wir wissen, wieviele Kontaktpersonen pro Kunde existieren; dabei interessieren uns nur Ergebnisse mit Anzahl > 2:

```
SELECT      KD_NR, COUNT(*) As Anzahl
FROM        KONTAKT
GROUP BY    KD_NR
HAVING COUNT(*) > 2;
```

KKURZ	Anzahl
100	3
101	4
107	3
109	3

Gruppierungen IV

Welche kumulierten, geplanten Umsätze (Summe aus AANZ*ASATZ) sind pro Kunde vorhanden?
Hier wird die Gruppe durch den jeweiligen Kunden (KKURZ in AUFTRAG) gebildet; die Anzahl der Aufträge des jeweiligen Kunden soll auch angezeigt werden:

```
SELECT     KD_NR, SUM(AANZ*ASATZ) As Planumsatz,
           COUNT(A_NR) As Auftragsanzahl
FROM       AUFTRAG
GROUP BY   KD_NR;
```

KKURZ	Planumsatz	Auftragsanzahl
100	18.000	1
101	9.000	1
106	6.500	1
109	1.500	1
110	16.000	3

Gruppierungen V

Von welchem Kunden liegt mehr als 1 Auftrag vor?

```
SELECT      KD_NR, COUNT(*) As Anzahl Aufträge
FROM        AUFTRAG
GROUP BY    KD_NR
HAVING COUNT(*) > 1;
```

KKURZ	Anzahl Aufträge
110	3

Gruppierungen VI

Es sollen die pro MWSt-Satz kumulierten Leistungsergebnisse (Summe LANZ*LSATZ aus LEISTUNG) derjenigen Leistungen ermittelt werden, deren MWSATZ ungleich 0 ist.
Da in der HAVING-Klausel immer eine Funktion (hier MIN) angegeben werden muß, kann die Abfrage wie folgt formuliert werden
(mit WHERE MWSATZ > 0 würde dasselbe Ergebnis erzielt werden):

```
SELECT      MWSATZ,
            SUM(LANZ*LSATZ) As Nettoumsatz,
            COUNT(MWSATZ) As Anzahl
FROM        LEISTUNG
GROUP BY    MWSATZ
HAVING      MIN(MWSATZ) > 0;
```

MWSATZ	Nettoumsatz	Anzahl
0,07	9.125	3
0,16	25.175	8

Daten einfügen, ändern und löschen I

INSERT INTO Tabellenname [(Spaltenname1,,,)] Quelle;

Quelle kann dabei eine Liste von Werten sein, die mit VALUES eingeleitet wird, oder es können Werte aus anderen Tabellen sein, die über ein SELECT aus diesen Tabellen ermittelt werden.

```
INSERT INTO KUNDE
(KD_NR, FIRMA, ZUSATZ, STRASSE, LAND, PLZ, ORT, TELEFON, INFO)
VALUES
(113, "Stadt", "Auto", "Weg. 4", "D", "28283", "Bremen", "0421-
667788", Verwaltung");
```

Da hier alle Spalten betroffen sind, kann man auch einfach schreiben:

```
INSERT INTO KUNDE VALUES
(113, "Stadt", "Auto", "Weg. 4", "D", "28283", "Bremen", "0421-
667788", "Verwaltung");
```

Beachten:
Zeichenketten sind in "", Reine Zahlen ohne ""

Daten einfügen, ändern und löschen II

Daten löschen:

DELETE
FROM Tabellenname
[WHERE Suchbedingung];

Falls die WHERE-Klausel nicht angegeben ist, werden sämtliche Zeilen in der Tabelle gelöscht.

```
DELETE FROM KUNDE;

DELETE FROM KUNDE WHERE KD_NR = 113;

DELETE FROM KUNDE WHERE NAME LIKE "M*";

DELETE FROM KUNDE WHERE ORT = „Bochum“;
```

Daten einfügen, ändern und löschen III

Daten verändern

UPDATE Tabellenname
SET Spaltenname = Feldwert,,,
[WHERE Suchbedingung];

Bei den folgenden Änderungen soll der Leistungssatz (LSATZ) der noch nicht abgerechneten Leistungen (RNR Is Null) um 10% erhöht werden; dann sollen die Aufträge 113 und 115 in den Status Ablage (4) übergehen:

```
UPDATE    LEISTUNG
SET       SATZ = SATZ*1.10
WHERE     R_NR Is Null;

UPDATE    AUFTRAG
SET       Status_NR = 4
WHERE     A_NR IN (113, 115);

UPDATE ADRESSBUCH SET NAME ="Meier" where NAME ="Meyer";
```

Daten in mehreren Tabellen suchen

Gleichheitsverknüpfung	Equijoin
Reflexionsverknüpfung	Selfjoin
Inklusionsverknüpfung	Outerjoin

Gleichheitsverknüpfung (Equijoin)

Es sollen alle Aufträge, die geplant sind oder für die es einen Vertrag gibt (also ASTNR = 1 oder 2), inkl. der Firmennamen der Kunden, angezeigt werden.
Das Problem besteht darin, daß die Auftragsdaten in der Tabelle AUFTRAG, die Firmennamen jedoch in der Tabelle KUNDE stehen.

```
SELECT    AUFTRAG.AUFTRAG_NR,
          AUFTRAG.KD_NR,
          KUNDE.FIRMA,
          AUFTRAG.STATUS_NR
FROM      AUFTRAG, KUNDE
WHERE     AUFTRAG.KD_NR = KUNDE.KD_NR
AND       AUFTRAG.STATUS_NR IN (1, 2);
```

Tabellenname.Spaltenname
In der FROM-Klausel werden die beteiligten Tabellennamen angegeben.
Die Bedingung WHERE AUFTRAG.KKURZ = KUNDE.KKURZ ist die Verbund-bedingung.
Mit ihr wird die Verbindung zwischen den beiden Tabellen hergestellt
AUFTRAG - Fremdschlüssel
KUNDE - Primärschlüssel

Gleichheitsverknüpfung (Equijoin) II

ACCESS:
FROM KUNDE INNER JOIN AUFTRAG ON KUNDE.KKURZ = AUFTRAG.KKURZ

Um kürzere SQL-Kommandos zu erzeugen, können Aliasnamen für die Tabellennamen verwendet werden. Der Aliasname wird in der FROM-Klausel angegeben:
FROM Tabellenname1 As Alias1, Tabellenname2 As Alias2
Er kann dann auch zur Qualifizierung der Attributnamen benutzt werden.

```
SELECT    BOOKING.A_NR,
          BOOKING.KD_NR,
          CUSTOMER.FIRMA,
          BOOKING.STATUS_NR
FROM      AUFTRAG AS BOOKING, KUNDE AS CUSTOMER
WHERE     BOOKING.KD_NR = CUSTOMER.KD_NR
AND       BOOKING.STATUS_NR IN (1, 2);
```

sqlv15 : Auswahlabfrage

A_NR	KD_NR	FIRMA	Status_NR
111	100	Versicherungspartne	1
114	101	Computer Courier	2
116	110	Bacher Elektronik A	2

Gleichheitsverknüpfung (Equijoin) III

Equijoin über 3 Tabellen:

```
SELECT    BOOKING.A_NR,
          BOOKING.KD_NR,
          CUSTOMER.FIRMA,
          BOOKING.STATUS_NR,
          AB.STATUS_TEXT
FROM      AUFTRAG AS BOOKING, KUNDE AS CUSTOMER, AUFSTAT AS AB
WHERE     BOOKING.KD_NR = CUSTOMER.KD_NR
AND       BOOKING.STATUS_NR = AB.STATUS_NR
AND       AB.STATUS_NR IN (1, 2);
```

In der obigen Abfrage müssen 2 Verbundbedingungen spezifiziert werden:
KD_NR für den Verbund zwischen den Tabellen AUFTRAG und KUNDE,
und STATUS_NR für den Verbund zwischen AUFTRAG und AUFSTAT.

sqlv17 : Auswahlabfrage

A_NR	KD_NR	FIRMA	Status_nr	status_text
111	100	Versicherungspartner	1	geplant
114	101	Computer Courier	2	Vertrag
116	110	Bacher Elektronik AG	2	Vertrag

Gleichheitsverknüpfung (Equijoin) IV

Welcher Kunde hat einen Vertrag?
Ausgabe des Firmennamens und des Auftragsstatustextes.

```
SELECT    K.FIRMA,
          A1.Status_txt AS Auftragsstatus
FROM      AUFTRAG AS A, KUNDE AS K, AUFSTAT AS A1
WHERE     A.KD_NR = K.KD_NR
AND       A.Status_nr = A1.status_nr
AND       A1.Status_txt = 'Vertrag';
```

FIRMA	Auftragsstatus
Computer Courier	Vertrag
Bacher Elektronik	Vertrag

Reflexionsverknüpfung

Bei einem Join kann auch eine Tabelle mit sich selbst verknüpft werden(Reflexionsverknüpfung).
Gegeben sei folgende Tabelle ANG, die die Angestellten einer Abteilung umfassen soll:

AngNr	Name	Beruf	AbtNr	VorgesNr	Gehalt
374	KÄMP	SEKR	3	401	3.800
112	MANG	PROGR	1	205	4.300
400	KLEF	SEKR	3	198	3.800
205	WIND	ORGAN	1		5.100
307	MILL	PROGR	2	205	5.400
117	SEEL	ING	3	401	4.100
198	FELD	KAUFM	3	401	4.100
401	OTTO	KAUFM	3	205	4.700

Die VorgesNr ist die AngNr derjenigen Person, die Vorgesetzter der gerade betrachteten Person ist. Wir wollen nun wissen, welcher Angestellte mehr als sein Vorgesetzter verdient. Wir bilden dazu einen Verbund der Tabelle ANG mit sich selbst. Dabei stellen wir uns vor, die Tabelle ANG läge 2x identisch vor. Wir bilden nun einen Verbund zwischen den beiden Kopien von ANG.

Reflexionsverknüpfung II

ANGESTELLTE

AngNr	Name	Beruf	AbtNr	VorgesNr	Gehalt
374	KÄMP	SEKR	3	401	3.800
112	MANG	PROGR	1	205	4.300
400	KLEF	SEKR	3	198	3.800
205	WIND	ORGAN	1		5.100
307	MILL	PROGR	2	205	5.400
117	SEEL	ING	3	401	4.100
198	FELD	KAUFM	3	401	4.100
401	OTTO	KAUFM	3	205	4.700

VORGES

AngNr	Name	Beruf	AbtNr	VorgesNr	Gehalt
374	KÄMP	SEKR	3	401	3.800
112	MANG	PROGR	1	205	4.300
400	KLEF	SEKR	3	198	3.800
205	WIND	ORGAN	1		5.100
307	MILL	PROGR	2	205	5.400
117	SEEL	ING	3	401	4.100
198	FELD	KAUFM	3	401	4.100
401	OTTO	KAUFM	3	205	4.700

...welcher Angestellte mehr als sein Vorgesetzter verdient:

```
SELECT   ANGESTELLTE.Name AS Angestellter,
         ANGESTELLTE. Gehalt AS A-Gehalt,
         VORGES.Name AS Vorgesetzter,
         VORGES.Gehalt AS V-Gehalt
FROM     ANGESTELLTE, ANGESTELLTE AS VORGES
WHERE    ANGESTELLTE.VorgesNr = VORGES.AngNr
AND      ANGESTELLTE.Gehalt > VORGES.Gehalt;
```

Angestellter	A-Gehalt	Vorgesetzter	V-Gehalt
MILL	5.400	WIND	5.100

Inklusionsverknüpfung

Neben der bisher betrachteten Gleichheitsverknüpfung (Innerjoin) gibt es noch die Inklusionsverknüpfung (Outerjoin).
Eine Gleichheitsverknüpfung erfaßt nur Zeilen, bei denen mindestens eine Zeile mit gleichem Attributwert bei der Gleichsetzung der Verbundspalten gefunden wird.
Die Inklusionsverknüpfung behandelt auch Tabellenwerte, für die kein Vergleichswert (in der anderen Tabelle) existiert, z. B. Kunden, die keinen Auftrag haben.

Man unterscheidet zwischen einer linksseitigen (left) und einer rechtsseitigen (right) Inklusionsverknüpfung. Links bzw. Rechts bezieht sich auf die Anordnung Master - Detailtabelle

Beispiel für eine Inklusionsverknüpfung:

```
SELECT    nachname AS Sachbearbeiter,
          präsent AS Präsent
FROM      Sachbearbeiter
LEFT JOIN   SA-Präsente ON Sachbearbeiter.nr = SA-Präsente.nr;
```

Es werden alle Sachbearbeiter angezeigt, auch diejenigen, die keine Präsente bekommen haben.

SELECT mit Subquery mit einem Vergleichswert

Die Grundstruktur eines SELECT mit Subquery mit einem Vergleichswert sieht wie folgt aus:

```
SELECT      Spalte1,,,
FROM        Tabelle1
WHERE       Spaltex = (   SELECT      Spaltex
                          FROM        Tabelle2
                          WHERE
                          );
```

Dabei wird zunächst die Unterabfrage abgearbeitet.
Mit dem daraus ermittelten Wert (nur 1 Wert erlaubt!) wird dann die Hauptabfrage bearbeitet; also Abarbeitung von unten nach oben.

Beispiel: Suche alle Firmen, die denselben Firmensitz wie Kunde 101 haben.

```
SELECT    FIRMA, ORT
FROM      KUNDE
WHERE     ORT = (SELECT ORT FROM KUNDE WHERE KD_NR = 101);
```

FIRMA	ORT
Versicherungspartner	Bremen
Computer Courier	Bremen
Deutsche Unterhaltung	Bremen
JUFI GmbH	Bremen

SELECT mit Subquery mit einem Vergleichswert II

Es gelten folgende Einschränkungen für solche SELECTs:

- in der Unterabfrage darf nur eine Spalte im SELECT angegeben werden,
- die Selektion in der Unterabfrage darf nur einen Wert liefern;
  (für mehr als 1 Wert siehe nächstes Kapitel)
- DISTINCT ist in der Haupt- und Nebentabelle erlaubt
- ORDER BY ist in der Nebentabelle nicht erlaubt

Weiteres Beispiel:
Welche Leistung ist zuletzt abgeliefert worden?

```
SELECT   LNR, LTEXT
FROM     LEISTUNG
WHERE    LDATUM = (SELECT MAX(LDATUM)  FROM LEISTUNG);
```

LNR	LTEXT
16	Rezension SQL*FORMS

SELECT mit Subquery mit einem Vergleichswert III

Welcher Kunde hat Aufträge mit einem Auftragswert, der über dem Durchschnitt aller Aufträge liegt?

```
SELECT   KD_NR, ANZ*SATZ As Auftragswert
FROM     AUFTRAG
WHERE    ANZ*SATZ > (SELECT AVG(ANZ*SATZ) FROM AUFTRAG);
```

KKURZ	Auftragswert
100	18.000
110	9.600
101	9.000

SELECT mit Subquery mit einem Vergleichswert IV

Welcher Kunde hat den Auftrag mit dem höchsten Wert?
(Anzeige der Firma, der PLZ und der Kundennummer):

```
SELECT    FIRMA, PLZ, KD_NR
FROM      KUNDE
WHERE     KD_NR = (SELECT    KD_NR
                   FROM      AUFTRAG
                   WHERE     (SATZ*ANZ) = (SELECT MAX(SATZ*ANZ)
                                           FROM    AUFTRAG
                                           )
                  );
```

FIRMA	PLZ	KKURZ
Versicherungspartner	28719	100

SELECT mit Subquery mit mehr als einem Vergleichswert

Um mit mehr als einem Vergleichswert aus einer Nebentabelle arbeiten zu können, muß eine weitere Form des SELECT mit Subquery benutzt werden.
Hierbei sind die Schlüsselwörter ANY, ALL, IN und EXISTS im Vergleichsoperator zulässig, wobei auch NOT IN und NOT EXISTS zugelassen sind.
Die allgemeine Form für solche SELECTS lautet (nur die WHEREBedingung):

WHERE Spaltenname Vergleichsoperator [ANY | ALL] (Unterabfrage)

oder

WHERE Spaltenname IN (Unterabfrage)

oder

WHERE Spaltenname EXISTS (Unterabfrage)

Mit dem IN-Operator (entspricht einem = ANY) in der Hauptabfrage werden alle Zeilen gesucht, bei denen der betreffende Spaltenwert einem der Ergebniswerte der Unterabfrage entspricht.

SELECT mit Subquery mit mehr als einem Vergleichswert II

Welche Aufträge haben die Bremer Kunden gegeben?
Anzeige der Kundennummer, des Auftragskürzels und des Auftragstextes.

```
SELECT    KD_NR, A_NR, TEXT
FROM      AUFTRAG
WHERE     KD_NR IN
          (SELECT KD_NR FROM KUNDE WHERE ORT = 'Bremen');
```

KKURZ	AKURZ	ATEXT
109	109	Serienbrief-Einweisung
100	111	Mitarbeiterschulung
101	114	ORACLE-Rezension