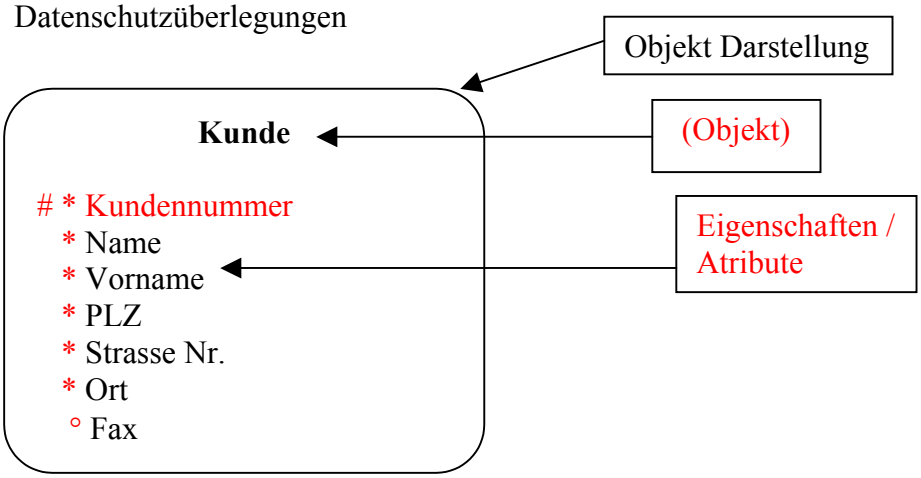
www.highsource.de Mario Kretschmer©

Datenbanken Access

Relationale Datenbanken

Datensammlung:

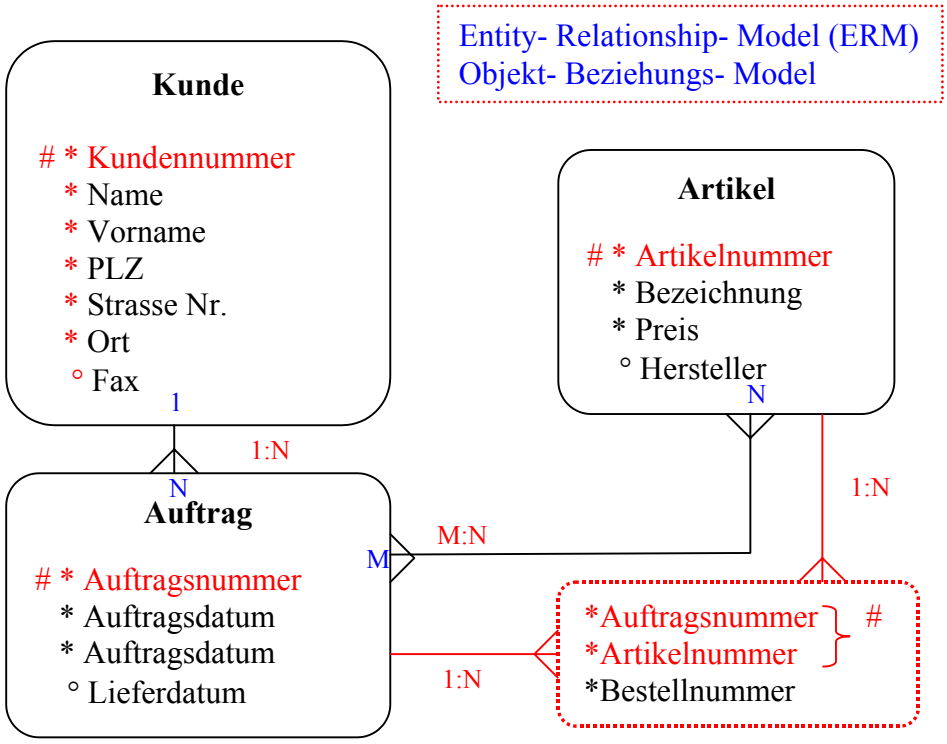
1. Datennamen
2. Datentypen
3. Größe der Zeichen oder Wertebereiche
4. Schlüsselqualität (Primärschlüssel, Fremdschlüssel)
5. Indizierungsnotwendigkeiten
6. Datenschutzüberlegungen



Überlegung:

Was brauche ich an Daten?
Wo ist mein Primärschlüssel # ?
Was ist in der Datensammlung Pflichtfelder * und Kannfelder °

Beispiel:



Übersetzungsregeln (Modell- Tabellen)

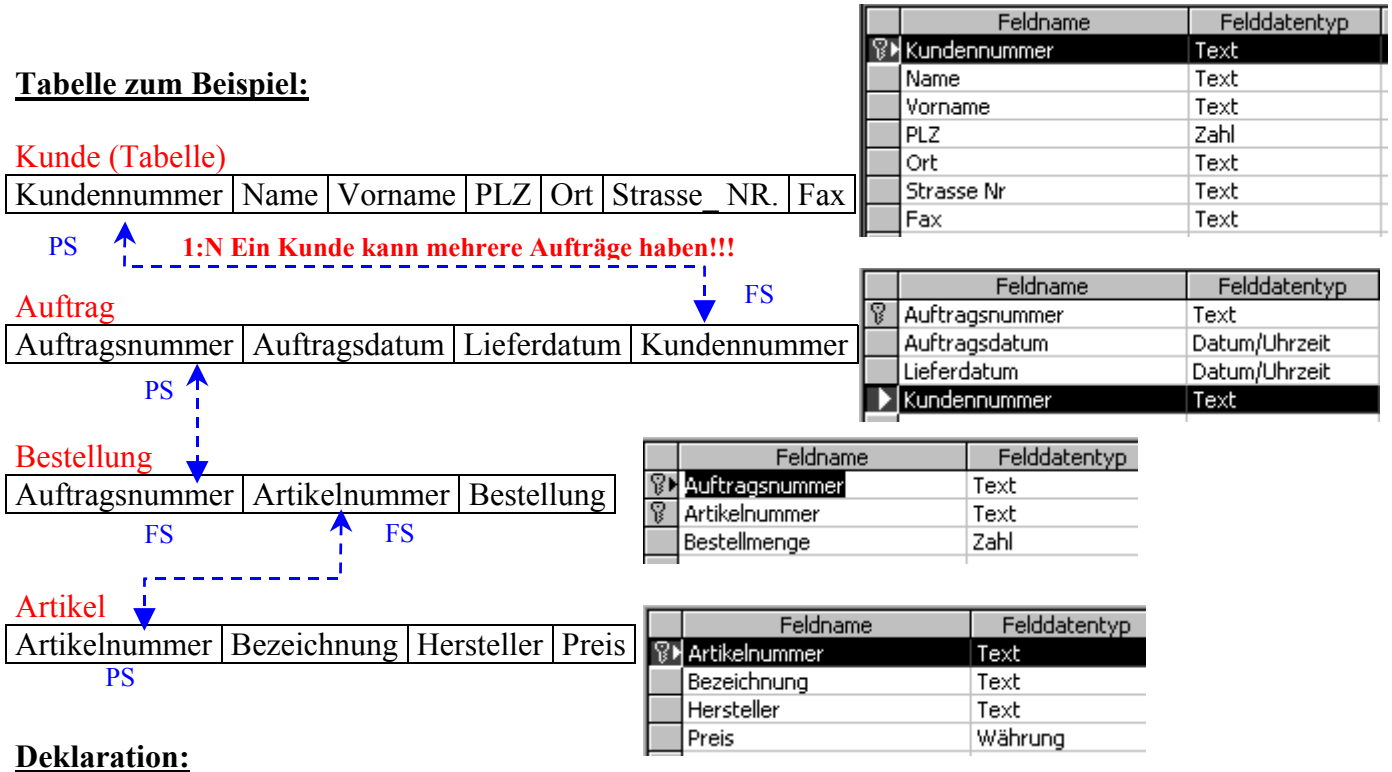
1. Jedes Objekt wird eine Tabelle (Relation)
2. Jede Eigenschaft wird zum Feld in der Tabelle
3. Übersetzung Beziehungstypen (Kardinalitätsverhältnisse)

a. 1:1 – Stehen 2 Objekte in einem Verhältnis 1:1 zueinander so ist der Primärschlüssel des einen Objektes als Fremdschlüsselattribut des anderen zu erfassen.

b. 1:N – Stehen 2 Objekte in einem Verhältnis 1:N zueinander so ist der Primärschlüssel des einen Objektes als Fremdschlüsselattribut des N: Objektes zu erfassen.

c. M:N – Stehen 2 Objekte in einem Verhältnis M:N zueinander, so entsteht eine neue Tabelle in der die Primärschlüssel, der in Beziehung stehenden Objekte erfasst werden. Sie bilden in der neuen Tabelle einen zusammengesetzten Primärschlüssel.

Tabelle zum Beispiel:



Feldname	Felddatentyp
Kundennummer	Text
Name	Text
Vorname	Text
PLZ	Zahl
Ort	Text
Strasse Nr	Text
Fax	Text

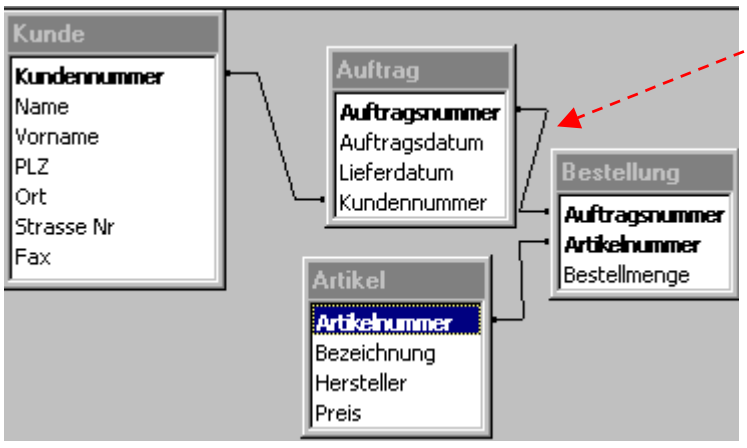
Feldname	Felddatentyp
Auftragsnummer	Text
Auftragsdatum	Datum/Uhrzeit
Lieferdatum	Datum/Uhrzeit
Kundennummer	Text

Feldname	Felddatentyp
Auftragsnummer	Text
Artikelnummer	Text
Bestellmenge	Zahl

Feldname	Felddatentyp
Artikelnummer	Text
Bezeichnung	Text
Hersteller	Text
Preis	Währung

Deklaration:

Bei der Festlegung des Felddatentyps und der Feldgröße bzw. Wertetypen ist darauf zu achten, dass das die Verbindungsfelder die gleichen Einstellungen



Referentielle Integrität= ist ein Regelsystem, das Microsoft Access verwendet, um sicherzustellen, dass Beziehungen zwischen Datensätzen in verknüpften Tabellen gültig sind und verknüpfte Daten nicht versehentlich gelöscht oder geändert werden. Aktivieren Sie dieses Kontrollkästchen, wenn Sie für diese Beziehung referentielle Integrität festlegen möchten. Aktivieren Sie es jedoch nur, wenn alle der folgenden Bedingungen vorliegen: das übereinstimmende Feld aus der Primärtabelle ist ein Primärschlüssel oder verfügt über einen eindeutigen Index, die verknüpften Felder haben denselben Datentyp und beide Tabellen sind in derselben Access-Datenbank gespeichert.

Zur Felddatentyp Einstellung

Ist die Eigenschaft des **Data Type** auf **Text** eingestellt, so geben Sie eine Zahl von 0 bis 255 ein. Die Standardeinstellung ist 50.

Ist die Eigenschaft **DataType** auf **AutoWert** eingestellt, so werden die Einstellungen der Eigenschaft **Feldgröße** als **Long Integer** oder **Replications-ID** eingestellt.

Ist die Eigenschaft **DataType** auf **Zahl** eingestellt, so werden die Einstellungen der Eigenschaft **Feldgröße** und deren Werte in folgender Weise miteinander verknüpft.

Anmerkung

Sie sollten die kleinstmögliche Einstellung der Eigenschaft **Feldgröße** verwenden, da kleinere Daten schneller verarbeitet werden können und weniger Speicherplatz belegen.

Vorsicht Wenn Sie eine große Einstellung der Eigenschaft **Feldgröße** in eine kleinere ändern und das zu ändernde Feld bereits Daten enthält, gehen möglicherweise Daten in diesem Feld verloren. Verändern Sie z.B. die Einstellung der Eigenschaft **Feldgröße** für ein Feld vom Datentyp **Text** von 255 auf 50, so gehen die Daten verloren, die mehr als 50 Zeichen umfassen.

Wenn die Daten in einem Feld vom Datentyp **Zahl** nicht mit der neuen Einstellung von **Feldgröße** übereinstimmen, werden Bruchzahlen eventuell gerundet, oder Sie erhalten einen **Null** -Wert. Wechseln Sie z.B. vom Datentyp **Single** zum Datentyp **Integer**, so werden Bruchzahlenwerte auf die nächste ganze Zahl gerundet, und Werte größer als 32.767 oder kleiner als -32.768 ergeben **Null**-Felder.

Sie können die Änderungen, die auf einer neuen Einstellung der Eigenschaft **Feldgröße** basieren, nach dem Speichern in der Tabellenentwurfsansicht nicht wieder rückgängig machen.

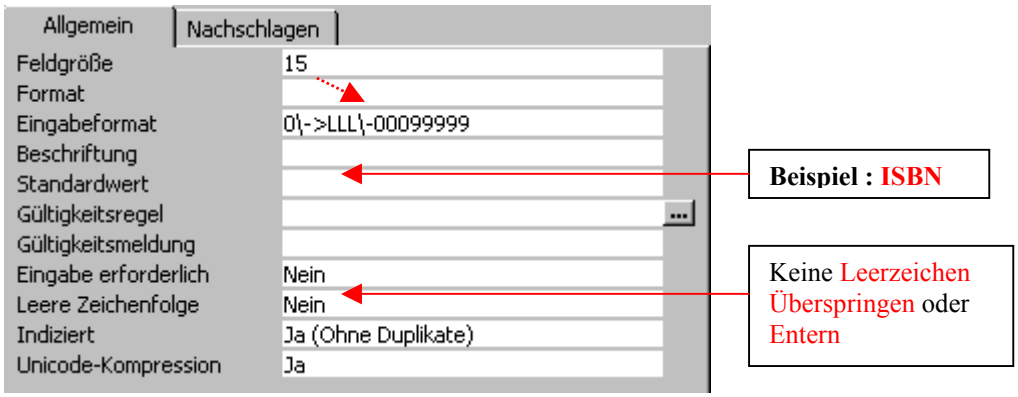
Tipp Sie können den Datentyp **Währung** verwenden, wenn Sie mehrere Berechnungen auf einem Feld durchführen möchten, das Daten mit bis zu vier Dezimalstellen enthält. Felder des Datentyps **Single** und **Double** erfordern Fließkommaberechnungen. Felder vom Datentyp **Währung** verwenden die schnellere Berechnungsmethode mit Festkomma.

	Einstellung	Beschreibung	Dezimale Genauigkeit	Speichergröße
Ganzzahl	Byte	Speichert Zahlen von 0 bis 255 (keine Bruchzahlen).	Keine	1 Byte
	Dezimal	Speichert Zahlen von -10^38 -1 bis 10^38 -1 (.adp) Speichert Zahlen von -10^28 -1 bis 10^28 -1 (.mdb)	28	12 Byte
	Integer	Speichert Zahlen von -32.768 bis 32.767 (keine Bruchzahlen).	Keine	2 Bytes
	Long Integer	(Voreinstellung) Speichert Zahlen von -2.147.483.648 bis 2.147.483.647 (keine Bruchzahlen).	Keine	4 Bytes
Dezimalzahl	Single	Speichert Zahlen von -3.402823E38 bis -1.401298E-45 für negative Werte, und von 1.401298E-45 bis 3.402823E38 für positive Werte.	7	4 Bytes
	Double (Doppelte Genauigkeit)	Speichert Zahlen von -1.79769313486231E308 bis -4.94065645841247E-324 für negative Werte, und von 1.79769313486231E308 bis 4.94065645841247E-324 für positive Werte.	15	8 Bytes
	Replikations-ID	GUID (Globally Unique Identifier) Ein 16 Byte großes Feld in einer Microsoft Access-Datenbank als eindeutige Kennung für eine Replikation. GUIDs werden zum Identifizieren von Replikaten, Replikatgruppen, Tabellen, Datensätzen und anderen Objekten verwendet. Anmerkung In einer Access-Datenbank werden GUIDs als Replikations-IDs bezeichnet.	N/A	16 Bytes

Eingabeformat Einstellungen:

Zeichen	Bedeutung
0	Ziffern von 0-9 Eingabe Erforderlich, aber + und – nicht erlaubt
9	Ziffern, Leerzeichen erlaubt Eingabe optional, + und – nicht erlaubt
#	Ziffern, Leerzeichen erlaubt Eingabe optional, + und – sind erlaubt
L	Buchstaben erlaubt, Eingabe erforderlich
?	Buchstaben erlaubt, Eingabe optional
A	Buchstabe oder Ziffer möglich, Eingabe erforderlich
a	Buchstabe oder Ziffer möglich, Eingabe optional
&	Beliebige Zeichen, Eingabe erforderlich
C	Beliebige Zeichen, Eingabe optional
<	Umwandlung der Nachfolgezeichen in klein Buchstaben
>	Umwandlung der Nachfolgezeichen in groß Buchstaben

Beispiel:
Kundennummer: 1-KYA-345.....
Eingabeformat: 0->LLL-000999999 ;0 NICHT SPEICHERN oder ;1 SPEICHERN
Standarteinstellung ist nicht Speichern!!!



Suchkriterien:

Namen (Text)	" Müller "
Vollzeit (Ja/Nein)	ja o. nein im Hintergrund fragt Access aber Ungleichnull oder null-1 oder 0
Titel	IST NULL oder IST NICHT NULL / hat einen Titel oder hat einen Titel
Datum	# Datum #

Teil Suche:

*	M* beliebig viele Zeichen
?	M???? soviel Zeichen
[]	Eckige Klammern immer für 1 Zeichen [W,L] bedeutet W oder L
-	heißt bis [W-L]
WIE "M*"	Suche was so aussieht WIE (TEXT)
WIE "12.??.??"	Datum suche aber keine # sonder hier wird wie bei (TEXT) gesucht!

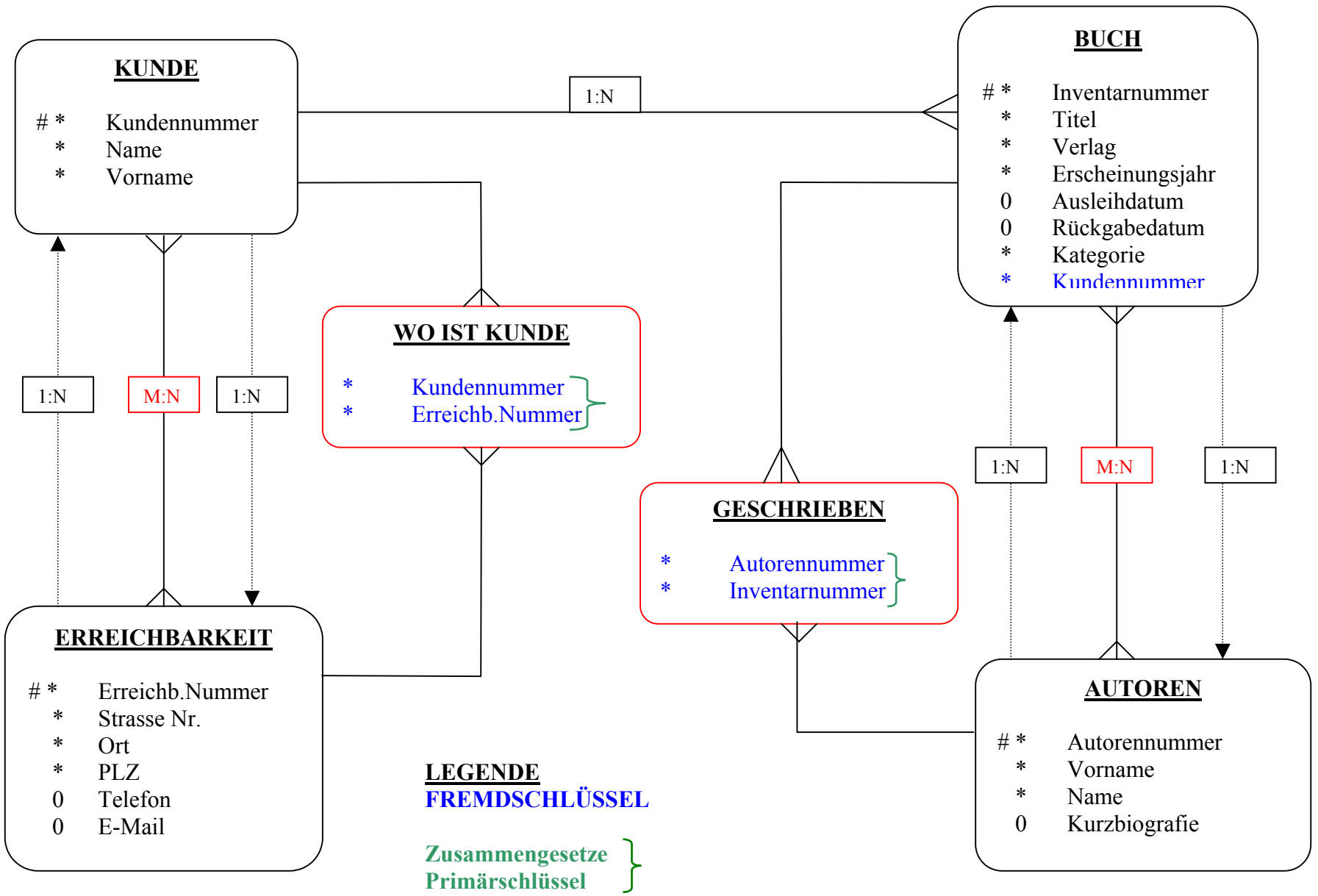
Mustersuche: mit < , >, <=, >=, <> aber kein (=) da dieses schon automatisch gemacht wird nur Zahlen keinen Text!!! (Numerische Felder wie Zahlen, Währung, Datum
Logische Operatoren: UND,ODER - bei Access auch ZWISCHEN

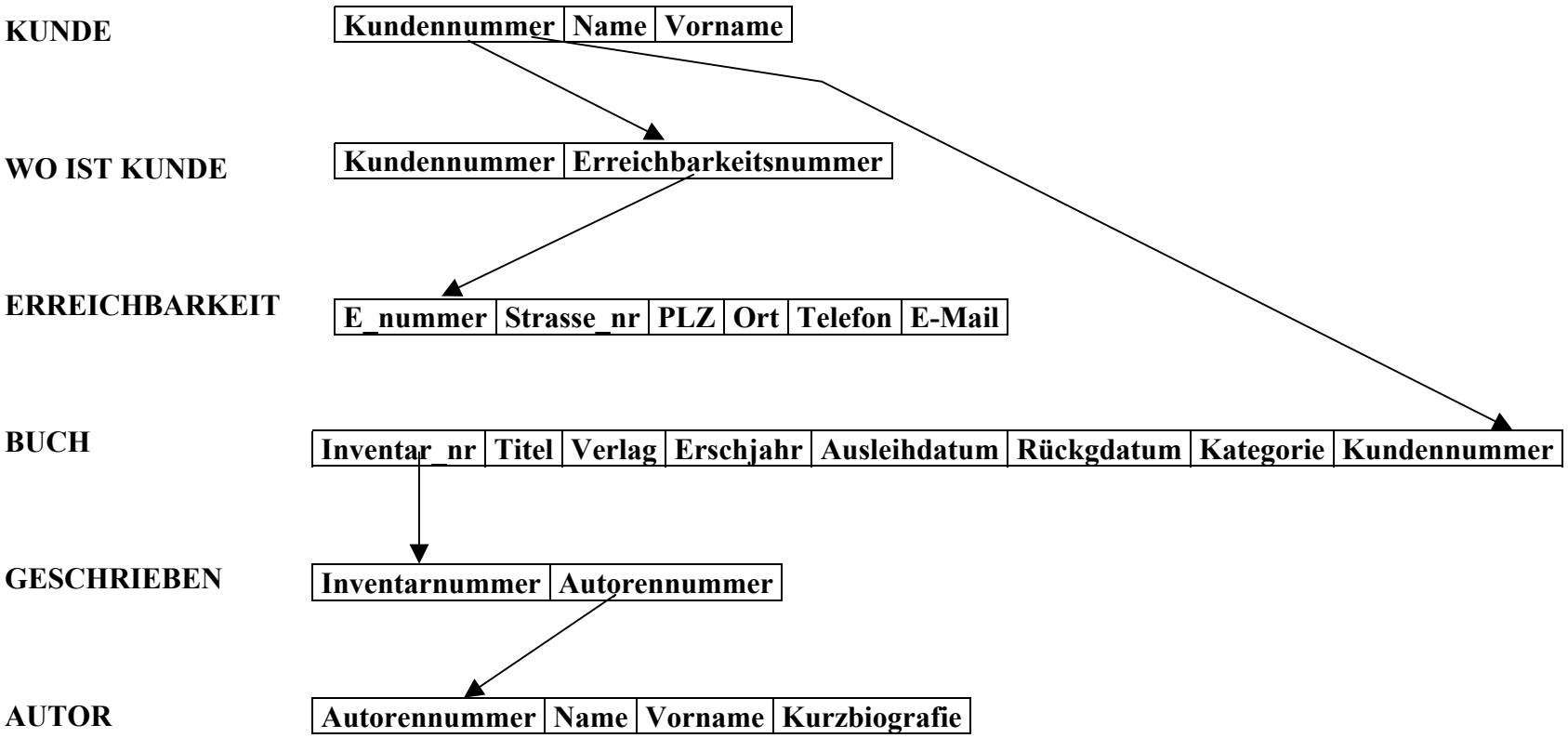
Berechnungen in der Abfrage:

Mann schreibt einfach in ein Abfragefeld wie dieses heißen soll z.B. **LOHNSTEUER** dahinter **:** um diesen Namen zu Trennen. Nun folgt das Bezugsfeld hier wäre es **[GEHALT]** (die Klammern nicht vergessen)!!! Jetzt wollen wir 50% Lohnsteuer berechnen lassen, also ***0,5**. **Jetzt einfach die** Abfrage starten!

Feld: **Lohnsteuer:[Gehalt]*0,5**

Übung Buchladen




KUNDE
Kundennummer
Name
Vorname
WO IST KUNDE
Kundennummer
Erreichbarkeitsnummer
ERREICHBARKEIT
E_nummer
Strasse_nr
PLZ
Ort
Telefon
E-Mail
BUCH
Inventar_nr
Titel
Verlag
Erschjahr
Ausleihdatum
Rückgdatum
Kategorie
Kundennummer
GESCHRIEBEN
Inventarnummer
Autorennummer
AUTOR
Autorennummer
Name
Vorname
Kurzbiografie

Normalisierung:

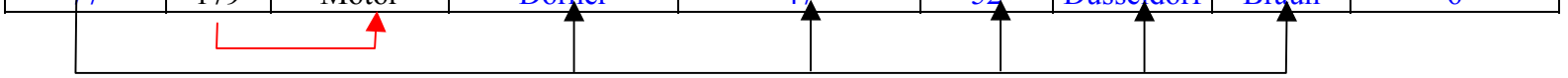
Ziel:
Redundante (sich wiederholende Werte) Werte zu vermeiden

- Speicherplatz sparen
- Änderungsaufwand senken, da Daten nicht in verschiedenen Tabellen geändert werden müssen
- Und damit Gewährleistung der Konsistenz der Daten (Übereinstimmung)

Beispiel: Ausgangstabelle aber ohne die blauen Einträge

Primärschlüssel

PERS_NR	FBNR	FABNAME	MECH_NAME	MECH_ALTER	FIL_NR	FIL_SITZ	LEITER	LEISTWERT
21	113	Karosserie	Adams	55	52	Düsseldorf	Braun	3
35	113	Karosserie	Becker	32	44	Köln	Grün	5
35	179	Motor	Becher	32	44	Köln	Grün	1
35	204	Transport	Becker	32	44	Köln	Grün	6
50	179	Motor	Conradi	40	44	Köln	Grün	2
77	148	Reifen	Dorner	47	52	Düsseldorf	Braun	6
77	179	Motor	Dorner	47	52	Düsseldorf	Braun	6



1. Normalform

- Primärschlüssel oder Teile davon keine Nullwerte daher Auffüllen wie hier in blau
- Keine Wiederholgruppen wie Becker, Becker, Becker oder Grün, Grün, Grün

2. Normalform → →

- Keine Abhängigkeiten von Feldern von Teilen eines zusammengesetzten Pimärschlüssel`s (nur wenn es einen zusammengesetzten Primärschlüssel gibt wie in diesem Beispiel)

Überführen einer Tabelle **aus der ersten in** die **zweite Normalform**:

1. Ermitteln der Abhängigkeiten von Feldern von Teilen des zusammengesetzten Primärschlüssels.
2. Auslagern der abhängigen Teile in eine neue Tabelle Schlüsselteile stellen in neuen Tabellen die Primärschlüssel da.

PERS_NR	FBNR	LEISTWERT
21	113	3
35	113	5
35	179	1
35	204	6
50	179	2
77	148	6
77	179	6

FBNR	FABNAME
113	Karosserie
179	Motor
204	Transport
148	Reifen

Bleibt als Fremdschlüssel in dieser Tabelle

PERS_NR	MECH_NAME	MECH_ALTER	FIL_NR	FIL_SITZ	LEITER
21	Adams	55	52	Düsseldorf	Braun
35	Becker	32	44	Köln	Grün
50	Conradi	40	44	Köln	Grün
77	Dorner	47	52	Düsseldorf	Braun

3. Normalform

- Wenn es keine Abhängigkeiten von Nicht Schlüsselfeldern gibt (transitive Abhängigkeiten)
- **wie hier grün dargestellt**

FIL_NR	FIL_SITZ	LEITER
52	Düsseldorf	Braun
44	Köln	Grün

Überführung in die **dritte Normalform**:

1. Auslagern der abhängigen Felder in eine Tabelle.
2. Der Primärschlüssel der neuen Tabelle verbleibt als Fremdschlüssel in der alten Tabelle.

SQL (Structured Query Language) = Strukturierte Abfrage

SELECT wählt die Felder aus die bearbeitet werden sollen oder * für alle Felder aus allen Tabellen

SELECT_[feld1],[feld2],[feld3].....
SELECT_*

FROM sagt aus welcher Tabelle oder Tabellen die Felder stammen sollen!

FROM_tabelle1,tabelle2.....

z.B. **SELECT_tabelle1.[feld1],tabelle2.[feld2],.....**
FROM_tabelle1,tabelle2.....

WHERE sagt aus welchem Datensatz Kriterium
WHERE_Bedingung

Bedingungen: bestehen aus FELDNAME_VERGLEICHSOPERATOR(=,>,<)_Suchwert(Datum oder Name)

Feldname:

1. **[]** wenn Leerzeichen im Namen
2. Bei Abfragen in mehreren Tabellen muss der Feldname eindeutig einer Tabelle zugeordnet werden können. Tritt der Feldname in mehreren Tabellen in der FROM- Klausel auf ist die Schreibweise **tabelle.[feld]** erforderlich.
3. Fehler in der Schreibweise des Feldnamens führen zum Fehler "Parametereingabe"

Suchwerte

1. **bei Text "Suchwert"** in Gänsefüße setzen
2. **bei Datum #MM/TT/JJ#** Jahreszahlen können nur bis 26 so eingegeben werden 2026 oder 1926
3. **bei Zahlen 1500.00** sind Punkte zu setzen keine Kommas
4. **bei Logischen Feldern** (JA/NEIN) mit **= yes, no**
5. **bei IST_NULL oder IST_NICHT_NULL - IS_NULL** oder **IS_NOT_NULL**
6. **bei Mustern für Text und Datumsangaben LIKE_"Muster"** Wildcarts: **, ?, [x,y], [x-z]**

a. LIKE nur bei Access
b. MATCHES bei Oracle

i. **z.B.** LIKE_"[M,W*er] –um die zu finden die entweder mit M oder W anfangen und mit er enden!

ii. **Datum** suchen mit **LIKE** ist zu beachten, dass das Datum wie Text geschrieben wird **"15.01.85"** aber **nicht 1985** da nur 2 Stellen gespeichert werden

iii. **Datum suche** alle die am **01** oder **15** eingestellt worden sind **"[0,1][1,5].*"**

7. **BETWEEN** erster_Wert **AND** zweiter_Wert

a. Beispiel: BETWEEN_3000_AND_5000 also zwischen 3000 und 5000 inklusive (3000 und 5000)

Verknüpfung mehrer Bedingungen:

SELECT.....
FROM.....
WHERE_Bedingung1_logischer Operator_Bedingung
AND
OR
NOT

Regel der Logischen Operatoren: **dienen zur Verknüpfung von Bedingungen**

- **NOT** vor **AND** und **AND** vor **OR**
- Bei Klammerung **( )** haben diese Vorrang vor allen Operatoren

Beispiel:
SELECT_Nachname,Gehalt,Abteilung,Initial
FROM verkäufer
WHERE Gehalt > 3000 AND Gehalt < 6000 OR Abteilung = "Verkauf" AND Initial = "F"

WHERE (Gehalt > 3000 AND Gehalt < 6000 OR Abteilung = "Verkauf") AND Initial = "F"

Beispiel:

Suchen Sie alle Leute die keinen Kommentar in der Abteilung Verwaltung arbeiten männlich sind und nach dem 30.11.64 eingestellt worden sind!

SELECT Nachname,Gehalt,Abteilung,Initial,Titel,Komment
FROM verkäufer
WHERE komment is null AND Abteilung = "Verwaltung" AND initial = "H" AND einst_dat > #11/30/64#

SELECT bezieht sich auf die Spaltenauswahl und werden per Komma getrennt

SELECT_[spalte1],[spalte2] oder * heißt alle auswählen
FROM_tabelle
WHERE_bedingung besteht aus 3 Teilen
[spalte] **vergleichsoperator** wonach man sucht **suchwert**

BETWEEN.....AND = suche zwischen da und da

LIKE bei der suche mit Jockerzeichen bei Text aber nicht wenn ich den Suchbegriff kenne
* beliebig
? ein Zeichen
[ ] eines der enthaltenen Zeichen
[!] genau dieses Zeichen

Ergänzung zur SELECT –Klausel

SELECT_DISTINCT_([spalte]) => ermittelt die unterschiedlichen Werte der angegebenen Spalte
Tritt meist bei der Verknüpfung von mehreren Tabellen auf
z.B. wenn mehrmals der gleiche Ort bei der Abfrage auftauchen würde wird er nur einmal angezeigt!

DISTINCTROW => für Zeilen (gleiche Datensätze nicht anzeigen)

Aggregate:

SELECT COUNT (*) oder COUNT ([spalte]) => ermittelt die Anzahl der gefundenen Werte einer Spalte

COUNT	Zählt
SUM_([spalte])	Summe
MIN_	Minimum
MAX_	Maximum
Avg_	Durchschnitt
TOP_nummer	ersten „nummer x“ Datensätze
TOP_zahl_PERCENT	„zahl“ in Prozent Datensätze vom Gesamtbestand

Beispiel:
SELECT_COUNT (DISTINCT([Kundennummer]))
FROM Auftrag

Zählt alle unterschiedlichen Kundennummern

Spaltenüberschrift erstellen zu einem jeweiligen Abfrage

SELECT_COUNT (DISTINCT([Kundennummer])_AS_[Auftragskunden]
⇨ Überschrift heißt Auftragskunden

Rechnen:
SELECT_[Menge]*[Preis]_AS_Lagerwert => Berechne Menge mal Preis diese Spalte soll Lagerwert heißen

SELECT_[spalte1]_math.operator_[spalte2]
*
/
+
-
^ Potenz

- Bei mehreren Rechenoperationen sind Klammern ( ) zur Änderung der Vorrangsregel möglich

Verknüpfung von Tabellen

Microsoft spezifisch:
SELECT…….
FROM_tabelle1_INNER_JOIN_tabelle2_ON_tabelle1.verbindungsfeld = tabelle2.verbindungsfeld

Andere SQL Derivate:
SELECT.....
FROM_tabell1,tabelle2,tabelle3
WHERE_tabelle1.verbfeld=tabelle2.verbfeld_AND_tabelle2.verbfeld=tabelle3.verbfeld

Ergänzung zu WHERE

Rechnen in der WHERE Klausel berechnet das Suchkriterium

Vergleiche: Einkaufspreis soll doppelt so groß sein wie Verkaufspreis

WHERE: [EKP]*2=[VKP]

IN - (alle die enthalten sein sollen)

SELECT.....
FROM_tabell1
WHERE_tabelle1.verbfeld IN (Kundennr1001,Kundennr1007……)

Ersetz hier die OR Schreibweise:

WHERE kunden1.vnum = 1001 or kunden1.vnum = 1007 or kunden1.vnum = 1004

Übung: zur Datenbank Versand 2000

BLATT 1

1.
```
SELECT Knum
From Aufträge1
where adatum = #10/03/90#
```

2.
```
SELECT Knum,Anum
From Aufträge1
where Knum = 2003
```

3.
```
SELECT adatum,kname
From Aufträge1,Kunden1
where kunden1.knum = aufträge1.knum and aufträge1.adatum = #10/06/90#
```

4.
```
SELECT adatum,kname,vname
From Verkäufer1,Kunden1,Aufträge1
where verkäufer1.vnum = kunden1.vnum and kunden1.knum = aufträge1.knum and
        aufträge1.adatum =  #10/03/90#
```

5.
```
SELECT anum,betrag
From Aufträge1
where aufträge1.betrag  between 1500 and  5000
```

6.
```
SELECT *
From Verkäufer1
where verkäufer1.vname like "*[L,i]*"
```

7.
```
SELECT *
From aufträge1
where aufträge1.anum = 3003 or aufträge1.anum = 3006 or aufträge1.anum=  3009
```

8.
```
SELECT aufträge1.knum,adatum,aufträge1.vnum
From Aufträge1
where aufträge1.adatum = #10/03/90#   and aufträge1.vnum = 1007
```

BLATT 2

1. und 2.
```
SELECT vnum as Verkäufernummer,vname as Verkäufername,provision
From verkäufer1
```

3.
```
SELECT Anum,Betrag,Adatum
From Aufträge1
```

4.
```
SELECT knum,kname,ort,kennzahl,vnum
From kunden1
WHERE vnum = 1002 and ort = "essen"
```

5.
```
SELECT kennzahl,Kname,ort
From kunden1
WHERE ort = "essen"
```

6.
```
SELECT kennzahl,Kname,ort,Knum,Vnum
From kunden1
WHERE ort <> "essen" and kennzahl = 200
```

7.
```
SELECT kennzahl,Kname,ort,Knum,Vnum
From kunden1
WHERE ort <> "essen" and kennzahl <> 200
```

8.
```
SELECT Vname,Ort,Provision
From Verkäufer1
WHERE ort = "Hamburg" and Provision = 0.12
```

9.
```
SELECT KNAME, ORT,knum,vnum
FROM  Kunden1
WHERE  kunden1.vnum = 1001 or  kunden1.vnum = 1007 or  kunden1.vnum = 1004
```

10.
```
SELECT KNAME, ORT,knum,kennzahl
FROM  Kunden1
WHERE  kennzahl is null
```

11.
```
SELECT KNAME, ORT,knum,kennzahl
FROM  Kunden1
WHERE  kname like "[A,G]*s"
```

12.
SELECT vnum,vname,ort,provision,umsatz
FROM verkäufer1
WHERE Provision is not null

13.
SELECT vnum,vname,ort,provision,umsatz
FROM verkäufer1
WHERE (vname LIKE "H*" or vname LIKE "*r") and ort = "Berlin"

Weiteres zur SELECT Klausel

SUM ,COUNT, MIN, MAX;

5.-SELECT knum,**AGGREGATE** die sich nicht auf die GROUP_BY Klausel stehenden Spalte beziehen
1.-FROM .....
2.-WHERE .....
3.-GROUP_BY_knum
⇨ (Da Group By die mit der gleichen z.B. Kundennummer, Beträge usw. in einer Temporären Liste ablegt kann er die nun zusammengefassten Werte nicht einzeln wieder ausgeben. Er kann nur sagen z.B. wie viele einträge in der jeweiligen Tabelle enthalten sind (Aggregatfunktionen)!!!

4.-HAVING => auswahl der Gruppen - sucht aus den bestimmten Gruppen oder auf die Aggregate die aus der GROUP_BY Klausel) HAVING funktioniert nur mit Verbindung von Group BY

6.-ORDER_BY => Sortierung von Spalten z.B. ORDER_BY_spalte1_**DESC** oder **ASC**,spalte2
ASC- Aufsteigend (standart)
DESC- Absteigend

Übung zu Datenbank Verband

1.
SELECT spielernr,verbandsnr,name
FROM Spieler
WHERE verbandsnr = 7060

2.
SELECT spielernr,geburtsjahr,name,geschlecht
FROM Spieler
WHERE [geburtsjahr] +17 = beitrittsjahr

3.
SELECT spielernr,geburtsjahr,name,geschlecht
FROM Spieler
WHERE geschlecht = "M" and geburtsjahr > 1970

4.

```
SELECT spielernr,geburtsjahr,name,geschlecht,ort,PLZ,strasse
FROM Spieler
WHERE ort = "Krefeld" or ort = "Meerbusch"
```

5.
```
SELECT spielernr,geburtsjahr,name,geschlecht,ort,PLZ,strasse
FROM Spieler
WHERE ort = "Krefeld" or ort = "Ratingen" or ort = "neuss" or ort = "Hilden"
```

6.
```
SELECT spielernr,geburtsjahr,name,geschlecht,ort,PLZ,strasse
FROM Spieler
WHERE geburtsjahr = 1962 or geburtsjahr = 1963 or geburtsjahr = 1970
```

7.
```
SELECT spielernr,name
FROM Spieler
WHERE name Like "*[e]?"
```

8.
```
SELECT spielernr,name,titel
FROM Spieler
WHERE titel <> "dr."
```

Übung Datenbank Versand

Agregatfunktionen:

a)
```
SELECT count (*) as Auftragszahl
FROM aufträge1
WHERE adatum = #10/03/1990#
```

Mathematische Ausdrücke in Select-Klausel

a)
```
SELECT anum,Verkäufer1.vnum,betrag, umsatz,[umsatz]*0.12 as Proumsatz
FROM Verkäufer1,Aufträge1
WHERE Verkäufer1.vnum = Aufträge1.vnum
```

ÜBUNG zum Übungsblatt SELEGRU2.doc Datenbank VERBAND2000

1.
```
SELECT spieler.name,team.teamnr
FROM spieler,team
where spieler.spielernr = team.spielernr
```

2.
```
SELECT spielernr,teamnr
FROM wettkämpfe
Group By teamnr,spielernr
HAVING sum  (gewonnen) > sum (verloren)
```

3.
```
SELECT beitrittsjahr,spielernr
FROM spieler
Where beitrittsjahr in (1970,1975,1979)
```

4.
```
SELECT beitrittsjahr,spielernr,name
FROM spieler
Where name like "B*"
```

5.
```
SELECT beitrittsjahr,spielernr,name
FROM spieler
Where name like "*n"
```

10.
```
SELECT Strafen.Spielernr, Avg(Strafen.BETRAG) AS [Mittelwert von BETRAG]
FROM Strafen
GROUP BY Strafen.Spielernr
HAVING (((Avg(Strafen.BETRAG))<50));
```

```
SELECT Strafen.Spielernr,SUM (betrag)
FROM Strafen
GROUP BY Strafen.Spielernr
HAVING count (spielernr) >= 2
```

11.
```
SELECT ort,count (spielernr) as Soviele
FROM spieler
GROUP BY ort
HAVING ort = "Düsseldorf"
```

Aktionsabfragen

UPDATE_tabelle(die geändert werden soll) - Änderung vorhandener Daten
SET_feld = wert
WHERE_bedingung

Beispiel:
UPDATE_strafen
SET_betrag = 100 **=>** **betrag = betrag * 1.1**
WHERE_spielernr =27

Ändere in der Tabelle Strafen den Betrag auf 100 für den Spieler 27 !!!
Rechnen in der SET Anweisung: Rechne 10 % zu den 100.

Verknüpfen von DATEN Textverkettung mit &

Beipsiel:

SET telnr = "030-" & [telnr]

DELETE **=>** **löschen von Datensetzen**
FROM_tabelle
WHERE_bedingung

Erstellen von Tabellen

CREATE_TABLE_tabellenname_(feldname_felddatentyp(Text)_ NOT NULL(keine Nullwerte zulassen) ,PRIMARY_KEY, ...

INDEX- Definition (OHNE Dublikate oder MIT Dublikate einfach UNIQUE weglassen)

CREATE_UNIQUE_INDEX_indexname_ON_tabellenname_(feldname_DESC,feldname2)

Beispiel:
CREATE_UNIQUE_INDEX_indspieler_ON spieler (spielernr)

Ohne Dublikat ist Primärschlüssel
Mit Dublikate sind Fremdschlüssel

Felddatentypen:

ACCESS	MS JET SQL	ANSI
TEXT	TEXT CHARACTER CHAR	CHARCTER
MEMO	LONGTEXT LONGCHARACTER MEMO	VARYING
ZAHLEN		
INTEGER	SMALLINT SHORT	SMALLINT
LONG INTEGER	INTEGER INT LONG	INTEGER
SINGLE	REAL SINGLE	REAL
DOUBLE	FLOAT NUMBER DOUBLE	FLOAT
BYTE	BYTE	/
Währung	CURRENCY MONEY	/
DATUM	DATE DATETIME	DATE DATETIME TIME
JA / NEIN	BIT YESNO LOGICAL BOOLEAN	BIT
AUTOWERT	COUNTER AUTOINCREMENT	

Beispiel: zum erstellen von TABELLE mit Primärschlüssel und Felddatentypen!

CREATE_TABLE_produkt_(pnr_smallint_Primary_KEY,
Zunr_integer_not_null,
bez_char(20)_not_null,
vpreis_money_not_null,
zupreis_money);

CREATE_TABLE_zulieferer_(zunr_integer_PRIMARY_Key,
zufirma_char(50)_not_null,
zutel_char(25));

CREATE_UNIQUE_INDEX_indzulieferer_ON_zulieferer (zunr);

CREATE_UNIQUE_INDEX_indprodukt_ON product (pnr),

CREATE_INDEX_find_product_ON_product (zunr);

INSERT_INTO_product (pnr,zunr,bez,vpreis,zupreis)
VALUES (10,1001,“SKI“,1050.00,850.00);

INSERT_INTO_tabellenname (feld1,feld2,feld,3…..)
VALUES (wert1,wert2,wert3.....) => z.B. “TEXT“,#MM/TT/JJ#, YES / NO, 0.01, NULL = nichts

DROP_INDEX_indexname_ON_tabellenname

Beispiel:
DROP_Index_indposition_On_position

Übung Datenbank Liefer Erstellen

TABELLE KUNDE:

```
Create table kunde (knr smallint Primary key not null,
KFirma text (30) not null,
Status smallint not null,
Kort text (30) not null)
```

Primärschlüssel setzen für KNR:

```
Create Unique Index ind_Kunde ON Kunde (KNR)
```

TABELLE AUFTRAG:

```
Create Table Auftrag (Anr smallint Primary Key not null,
Knr smallint not null,
ADatum Date not null,
LDatum Date)
```

Primärschlüssel setzen für ANR:

```
Create Unique Index  ind_Auftrag ON Auftrag (ANR)
```

Fremdschlüssel setzen KNR:

```
Create Index ind_Auftrag On Auftrag (KNR)
```

TABELLE POSITION:

```
Create Table Position (ANR smallint not null,
PNR smallint not null,
PMenge smallint not null,Primary Key (ANR,PNR))
```

Zusammengesetzten Primärschlüssel setzen für ANR und PNR

```
Create Unique Index ind_Position ON Position (ANR,PNR)
```

Fremdschlüssel setzen ANR:

```
Create Index ind_PositionFS On Position  (ANR)
```

Fremdschlüssel setzen PNR:

```
Create Index ind_PositionFS2 On Position  (PNR)
```

TABELLE PRODUKT:

Create Table Produkt (PNR smallint Primary Key,
Zunr smallint,
Bez Text (30) not null,
VPreis Money not null,
ZuPreis Money not null)

Primärschlüssel setzen für PNR:

Create Unique Index ind_Produkt ON Produkt (PNR)

Fremdschlüssel setzen für Zunr:

Create Index ind_ProduktFS On Produkt (Zunr)

TABELLE ZULIEFERER:

Create Table Zulieferer (Zunr smallint Primary Key,
ZuFirma Text (30) not null,
ZuOrt Text (30) not null,
ZuTel Text (30) not null,
ZuFax Text (30) not null)

Primärschlüssel setzen für Zunr:

Create Unique Index ind_Zulieferer ON Zulieferer (Zunr)

TABELLE LAGER:

Create Table Lager (PRN smallint not null,
LOrt Text (30) not null,
LMenge smallint not null)

Zusammengesetzten Primärschlüssel setzen für PRN und Lort:

Create Unique Index ind_Lager On Lager (PRN,LOrt)

Fremdschlüssel setzen für PRN:

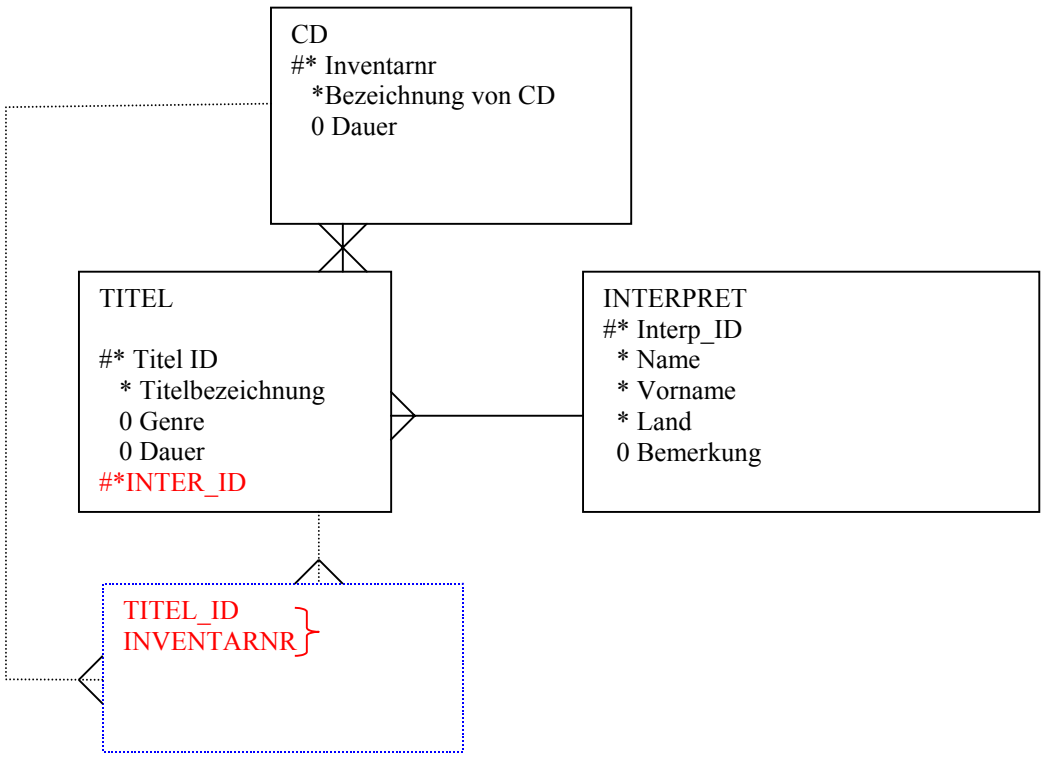
Create Index ind_LagerFS On Lager (PRN)

ZUR Klausur

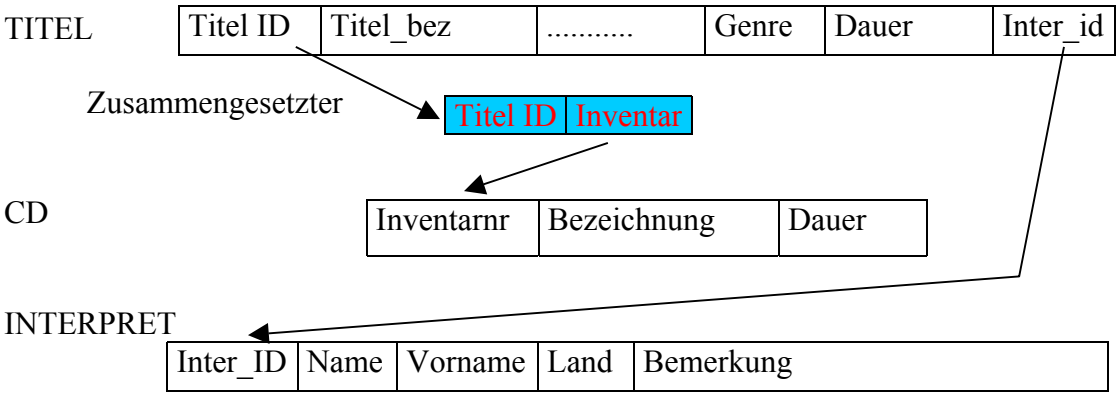
Objekt Beziehungsmodel

Daten die man hat zu Ordnen also bestimmten Objekten zuzuordnen

Es ist darauf zu achten das keine Zusammengesetzten Atribute verwendet werden



Beziehungstypen: 1 Regel jedes Objekt eine Tabelle



Jetzt Tabelle erstellen in SQL:

CREATE_TABLE_titel_(titelid_smallint_PRIMARY_KEY,
Titelbez_text(40)_not_null,
Genre_text(15),
Dauer_date,
Interid_smalint_not_null)

CREATE_TABLE_titleCD_(titelid_smallint_not_null,
Inventnr_smallint_not_null)

CREATE_TABLE_CD_(Inventarnr_smallint_Primary_KEY,
Bezeichnung_text(40)_not_null,
dauer_datetime_not_null)

CREATE_TABLE_InterID_(Name_text(30) Primary_KEY,
Vorname_text(20)_not_null
Land_text(30),
Bemerkung memo)

Nun die Index zuweisen:

CREATE_UNIQUE_INDEX_titel-ind_ON_titel_(titelid);

CREATE_INDEX_titel-indFS_ON_titel_(interid);

CREATE_UNIQUE_INDEX_titelcd-ind_ON titel_cd (titelid,Inventarnr);

CREATE_INDEX_titelcd-indFS2_ON _titel_-cd (Inventarnr);

CREATE_UNIQUE_INDEX_InterID-ind_ON_Interpret (Inter-ID);

Thema 2:

SELECT werden Spalten für die Anzeige ausgewählt!!!

SELECT_feld1_feld2.....
Oder alle
SELECT_*

Rechnen:
SELECT_feld1*feld2
SELECT_(feld1+feld2)*0.5

Agregate: für ganze Tabellen!!!

SELECT_sum(gehalt)
FROM_verkäufer

Bei SUM warden die Spalten zusammengefasst

Personalnummer	Name	Gehalt
1,2,3,4,5,....	Müller,Meier....	4000,5000,6000
	COUNT (Name)	SUM (Gehalt)

Agregate mit GROUP BY betreffen die zugruppierenden Felder nicht wie oben alle
SELECT_SUM
FROM Verkaufer
GROUP_BY_Abteilung

Persnr	Abteilung	Initial
	Finanzen Verkauf Verwaltung	

In Verbindung mit GROUP BY kann noch HAVING genutzt werden (wer hat) woanders nicht!!!

Abarbeitung des PC:
SELECT 6
FROM 1
WHERE 2
GROUP BY 3
HAVING 4
ORDER BY 5

SELECT => muss vorhanden sein
FROM => muss vorhanden sein
WHERE => kann vorhanden sein => und was da noch sein kann

WHERE feldname vergleichsoperator Suchwert

=	"Suchtext"
<	#MM/TT/JJ#
>	Zahl 1000 oder 0.53
<=	YES od. NO
>=	
<>	

bei Text mit Jokerzeichen (?* wird statt = LIKE (also wie)
Beispiel: WHERE_feldname_LIKE_"M*"

Ist Null oder Ist nicht Null
Beispiel: WHERE_feldname_IS_NULL od. IS_NOT_NULL

BETWEEN_wert1_AND_wert2
Beispiel: WHERE_feldname_BETWEEN_wert1_AND_wert2

IN => kann man anstatt mehrerer OR Verknüpfung
Beispiel: WHERE_feldname_IN (“Berlin“,“München“,“Frankfurt“)

Verknüpfen von Tabellen:

Achtung: darauf achten das immer die Verbindungsfelder also Primär und Fremdschlüssel miteinander Verbunden werden!!!

SELECT
FROM_Tabelle1,Tabelle2
WHERE Tabelle1.feldname = Tabelle2.Feldname

Sortierung von Feldern

ORDER_BY_feld1_feld2_feld3 DESC od. ASC Absteigend od. Aufsteigend