

Relationale Datenbanken

Einführung in die
Programmiersprache SQL

Ausgangssituation

Betrachtet werden soll ein Bauunternehmen.

- Zu einem solchen Unternehmen gehören neben dem **Personal** auch **Kunden**, die **Aufträge** für eine **Baustelle** erteilen.
- Für die Ausarbeitung der Unterlagen werden i. a. **Architekten** von dem Bauherren (ist zugleich Kunde des Bauunternehmens) herangezogen.
- Außerdem müssen in einem solchen Betrieb eine Reihe anderer Daten (z. B. Maschinen, Rechnungen usw.) verwaltet werden.

Was sind relationale Datenbanken?

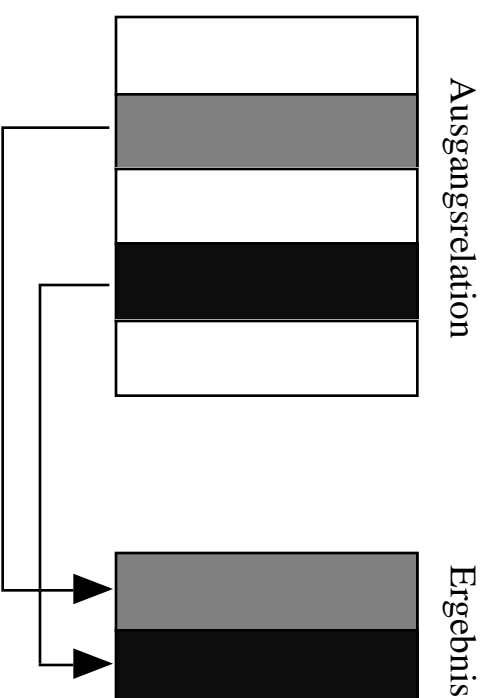
- Wollte man die oben genannten Daten alle in eine Tabelle schreiben, würde diese sehr unübersichtlich werden, vor allem würden identische Daten aber auch mehrfach in dieser Tabelle stehen.
- Diese Tatsache nennt man **Redundanz**.
- Redundanzen und damit mögliche Anomalien vermeidet man dadurch, dass die Daten in **mehrere Tabellen** verteilt werden.
- Durch sog. **Primärschlüssel** können diese Tabellen untereinander in Bezug gesetzt werden.

Operationen in relationalen Datenbanksystemen

- Im relationalen Datenmodell unterscheidet man drei Grundoperationen für Abfragen:
- Projektion
- Selektion
- Verknüpfung (engl.: Join)

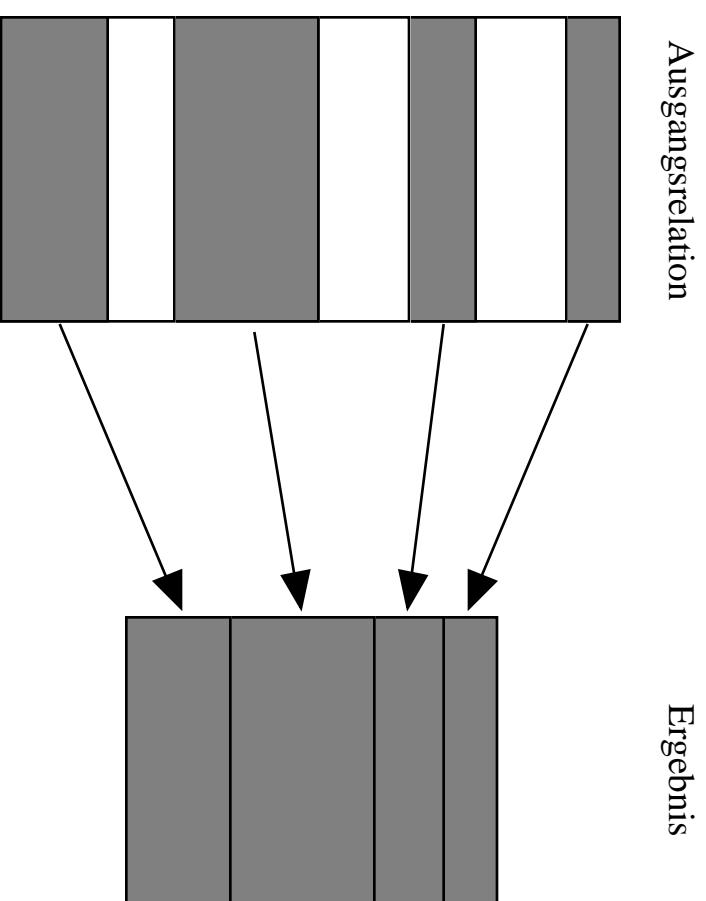
Projektion

Bei der **Projektion** werden bestimmte **Attribute** einer **Relation** ausgewählt.



Selektion

Bei der **Selektion** werden bestimmte **Sätze** (Zeilen) einer Relation ausgewählt.



Theta-Join

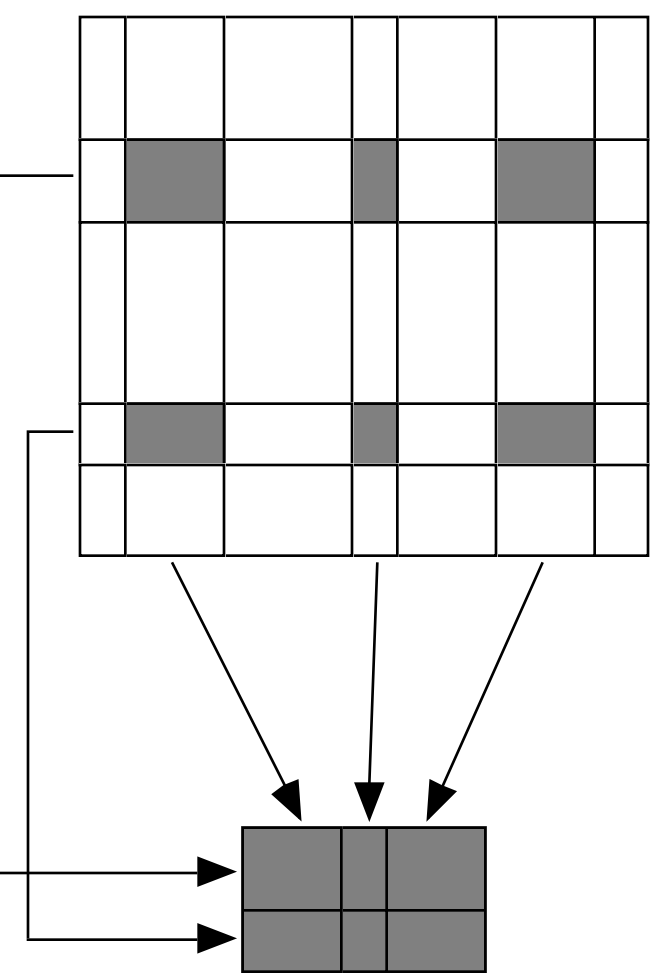
Bei der **Verknüpfung** oder **Join** werden mehrere Relationen zusammengeführt. Der *allgemeine Join* oder *Theta-Join* verknüpft jeden Satz der einen Datei mit jedem Satz der anderen Datei. Beim Theta-Join entstehen Dateien, die deutlich größer sind als die Ausgangsdateien.

Enthält die erste Datei drei Datensätze, die zweite fünf, enthält die resultierende Datei $3*5=15$ Datensätze.

Abfragen in der Praxis

Die drei Datenbank-Grundoperationen kommen in der Praxis nur selten isoliert vor. Eine übliche Datenbankabfrage ist in der Regel eine Kombination aus zwei oder drei der gezeigten Grundoperationen.

Beispiel:
Kombination einer
Projektion und
Selektion



Datenbank-Abfragesprache

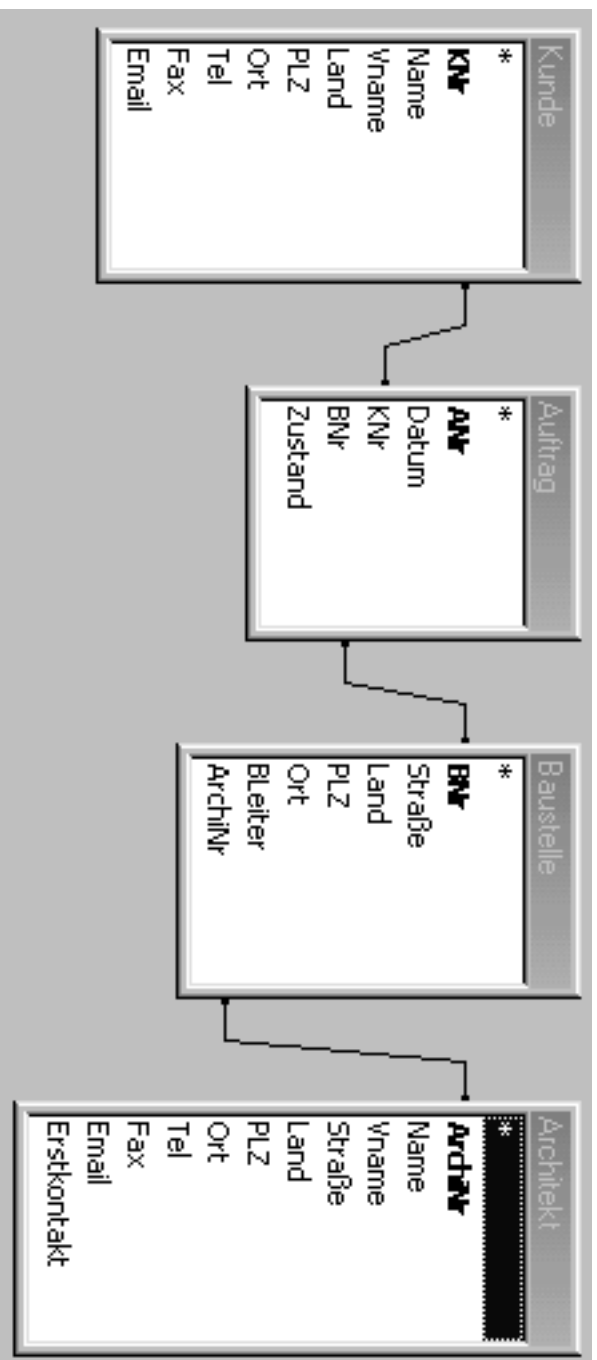
SQL

- Unterschiedliche Datenbanksysteme besitzen unterschiedliche Abfragesprachen.
- Bei einem Wechsel des Systems muss man eine neue Sprache zu lernen und die bislang verwendeten Abfragen in die neue Sprache übersetzen.
- Um unabhängig von einem speziellen Datenbanksystem zu werden und die gleichen Abfragen auch im neuen System verwenden zu können, wurden universelle, *portable* Abfragesprachen entwickelt.
- Für relationale Datenbanksysteme hat sich die portable Abfragesprache SQL (structured query language) durchgesetzt.

Generelle Struktur einer Abfrage

- **SELECT** <Namen von Attributen, mit Kommata getrennt>
FROM <Namen von Relationen, mit Kommata getrennt>
WHERE <Bedingungen>;
- In der **SELECT**-Komponente werden die Attribute, also die **Spalten** der Tabelle(n) benannt, die in der Abfrage erscheinen sollen.
- Die in die Abfrage einbezogenen **Relationen** (Tabellen) stehen in der **FROM**-Komponente.
- Die **WHERE**-Komponente beschreibt, welche **Zeilen** ausgewählt werden sollen. Sie ist optional; wenn sie weggelassen wird, werden alle Sätze ausgewählt.

Die Struktur der Datenbank



Die Pfeile zeigen die Verbindungen der Relationen untereinander an.

Projektion in SQL

- `SELECT strasse, land, plz, ort`
`FROM Baustelle;`
- Mit diesem Befehl werden die Spalten „strasse“, „land“, „plz“ und „ort“ der Tabelle „Baustelle“ ausgewählt. Es werden also *alle* bisherigen, aktuellen und projektierten Baustellen angezeigt.
- `SELECT *`
`FROM Baustelle`
- Mit diesem Befehl werden durch den Joker „*“ alle Attribute und alle Datensätze angezeigt.

Selektion

- **SELECT ***
FROM Baustelle
WHERE land = „A“
- Mit diesem Befehl werden durch den Joker „*“ alle Attribute und alle Datensätze angezeigt, bei denen das Attribut Land den Inhalt „A“ besitzt.

Kombinierte Projektion und Selektion

- `SELECT strasse, land, plz, ort`
`FROM Baustelle`
`WHERE land = 'A';`
- Man erhält die vier Spalten „strasse“, „land“, „plz“ und „ort“ angezeigt, aber nur diejenigen Sätze ausgewählt, bei denen die Bedingung „land = 'A'“ zutrifft. Außerdem:
- `SELECT strasse, plz, ort`
`FROM Baustelle`
`WHERE land = 'A';`
- Eine Variable der `WHERE`-Komponente braucht nicht unbedingt in der `SELECT`-Komponente vorzukommen.

Mögliche Operatoren

- SQL kennt wie QBE Vergleichs- und Mengenoperatoren.
- Vergleichsoperatoren sind:

=	gleich	<>	ungleich	!=	ungleich
<	kleiner	<=	kleiner gleich		
>	größer	>=	größer gleich		
!<	nicht kleiner	!>	nicht größer		
- Mengenoperatoren
IN
BETWEEN

Beispiele für Mengenoperatoren

- `SELECT strasse, land, plz, ort`
`FROM Baustelle`
`WHERE ArchiNr IN (1,2,5,16);`
- Man erhält somit alle Baustellen der Architekten 1, 2, 5 und 16 angezeigt (sofern vorhanden).
- `SELECT strasse, land, plz, ort`
`FROM Baustelle`
`WHERE ArchiNr BETWEEN 13 AND 20;`
- Es wird überprüft, ob in der Datei Baustelle die Architekten mit den Kennzahlen zwischen 13 und 20 (einschließlich) stehen.

Logische Operatoren (I)

- SQL kennt die logischen Operatoren **NOT**, **AND** und **OR**. Beispiele:
- **SELECT** Anr, datum
FROM Auftrag
WHERE zustand = 'fertig' **AND** KNr = 10;
- Hier werden die Sätze selektiert, bei denen der Auftrag fertig gestellt ist und nur den Kunden mit der Nummer 10 betrifft.
- **SELECT** Anr, datum
FROM Auftrag
WHERE zustand = 'fertig' **OR** KNr = 10;
- Hier werden alle fertig gestellten Aufträge angezeigt und zusätzlich diejenigen des Kunden mit der Nummer 10

Logische Operatoren (II)

- **SELECT** Anr, datum
FROM Auftrag
WHERE NOT (zustand = 'fertig');
- Hier werden alle nicht fertig gestellten Aufträge angezeigt.
- Sollte das Merkmal „zustand“ neben „fertig“ auch die Einträge „Projekt“ und „Arbeit“ zulassen entspräche der obige SQL-Befehl dem nachstehenden:
- **SELECT** Anr, datum
FROM Auftrag
WHERE zustand = 'Projekt' **OR** zustand = 'Arbeit';

Arbeiten mit Klammern

- SQL lässt im Gegensatz zu einfachen QBE-Verfahren Klammern zu:
- `SELECT Anr, datum
FROM Auftrag
WHERE (zustand='fertig' OR KNr=10) AND BNr IN (1,16,29,32);`
- `SELECT Anr, datum
FROM Auftrag
WHERE zustand='fertig' OR (KNr=10 AND BNr IN (1,16,29,32));`
- Im ersten Fall werden alle Baustellen selektiert, die abgeschlossen sind oder vom Kunden 10 in Auftrag gegeben wurden, sie müssen aber die angegebenen Nummern besitzen.
- Im zweiten Fall werden alle abgeschlossenen Baustellen selektiert, zusätzlich werden alle Baustellen angezeigt, die vom Kunden 10 in Auftrag gegeben wurden, sie müssen aber mindestens eine der angegebenen Nummern tragen.

Funktionen

- Mit Hilfe von Funktionen können zusammenfassende Berechnungen durchgeführt werden. Zur Verfügung stehen:
- **MAX** maximaler Wert eines Attributs (einer Spalte)
- **MIN** minimaler Wert eines Attributs
- **COUNT** Anzahl der Werte in einer Spalte
- **SUM** Summe der Werte einer Spalte
- **AVG** Mittelwert der Werte einer Spalte
- **Beispiel:**

```
SELECT COUNT(ArchiNr)  
FROM Baustelle  
WHERE ArchiNr=2;
```

Gruppieren von Daten

- Gesetzt den Fall man möchte wissen, wie viele Baustellen die einzelnen Architekten betreut haben, dann müssen Gruppen gebildet werden, die einzelnen Architekten. Lösung in SQL:
- ```
SELECT ArchiNr, COUNT(BNr)
FROM Baustelle
GROUP BY ArchiNr;
```

# Weitere Sprachkonstrukte

- In der benutzten Datenbank gibt es viele Attribute vom Typ Text. Um aus solchen Merkmalen etwas herauszufiltern, benötigt man weitere Konstrukte. Besonders wichtig ist der Operator LIKE, der zusammen mit Jokern bei der Suche verwendet wird.  
Beispiel:
  - `SELECT Name, Vorname  
FROM Kunde  
WHERE name LIKE 'R%';`
  - Es werden die Kunden, deren Name mit einem R beginnt, aufgelistet.
  - `SELECT Name, Vorname  
FROM Kunde  
WHERE name LIKE '%-%';`
- Es werden die Kunden eines Bindestrichs im Nachnamen aufgelistet.

# Theta-Join und Equi-Join

- Allgemeine Konstruktion des Theta-Joins:
- `SELECT * FROM <Namen der zu verbindenden Tabellen>`
- Beispiel:  
`SELECT * FROM Kunde, Auftrag`
- Anmerkung: Der größte Teil der erzeugten Satzkombinationen wird nicht benötigt  $\Rightarrow$  man benötigt eine Bedingung, die nur zulässige Satzkombinationen erzeugt:
- `SELECT Kunde.KNr, name, vorname, datum`  
`FROM Kunde, Auftrag`  
`WHERE Kunde.KNr= Auftrag.KNr`

# Equi-Join und weitere Bedingungen

- Angenommen, man benötigt die Auftragsnummern eines bestimmten Kunden mit dem zugehörigen Auftragsdatum.

Lösung:

- ```
SELECT Kunde.KNr, name, vorname, ANr, datum
FROM Kunde, Auftrag
WHERE Kunde.KNr=Auftrag.KNr
AND name = 'Müller-Lüdenscheid'
```