Prof. Dr. Th. Härder
AG Datenbanken und Informationssysteme
Zi. 36/330, Tel.: 0631-205-4030
E-Mail: haerder@informatik.uni-kl.de
http://wwwdbis.informatik.uni-kl.de/

# Datenbankanwendung

Wintersemester 2000/2001

Universität Kaiserslautern
Fachbereich Informatik
Postfach 3049
67653 Kaiserslautern

**Vorlesung:**

**Ort: 48 - 208**

**Zeit: Mo., 10.00 - 11.30 Uhr**

**und**

**Ort: 48 - 210**

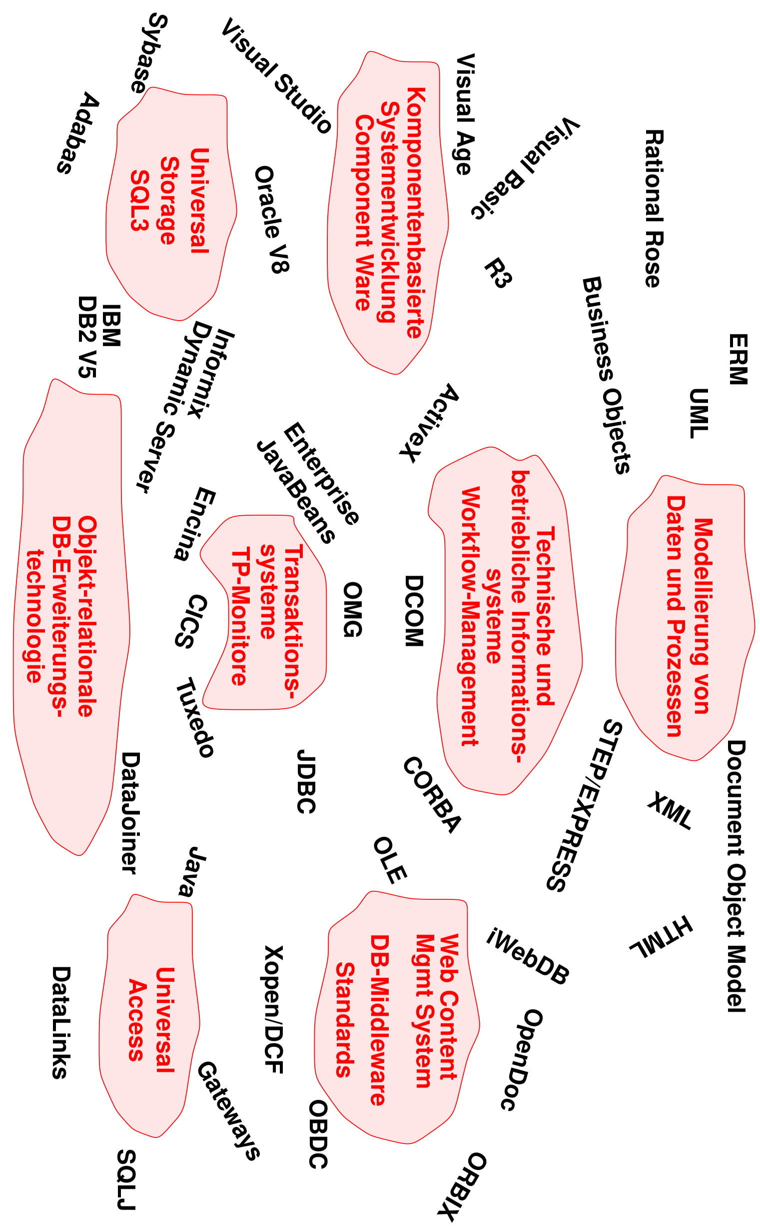
**Zeit: Mi., 11.45 - 13.15 Uhr**

**Übung:**

**Ort: 36 - 336**

**Zeit: 15.30 - 17.00 Uhr**





# Ziele

- **Vermittlung von Grundlagen- und Methodenwissen[1] zur Anwendung von Datenbanksystemen; Erwerb von Fähigkeiten und Fertigkeiten für DB-Administrator und DB-Anwendungsentwickler**
  - Entwurf, Aufbau und Wartung von Datenbanken, insbesondere auf der Basis von
    - Relationenmodell und SQL
    - objektorientierten und objekt-relationalen Datenmodellen
  - Sicherung der Abläufe von DB-Programmen
    - Transaktionsverwaltung, Synchronisation, Fehlerbehandlung
    - Semantische integrität, aktive DB-Mechanismen
    - Datenschutz und Zugriffskontrolle
  - Auswahl und Bewertung von Speicherungsstrukturen und Zugriffspfaden
  - Beschreibung und Analyse von wichtigen DB-Anwendungen

- **Voraussetzungen für Übernahme von Tätigkeiten:**
  - Entwicklung von datenbankgestützten Anwendungen
  - Nutzung von Datenbanken unter Verwendung von (interaktiven) Datenbanksprachen
  - Systemverantwortlicher für Datenbanksysteme, insbesondere Unternehmens-, Datenbank-, Anwendungs- und Datensicherungsadministrator

1.Grundlagenwissen ist hochgradig allgemeingültig und nicht von bestimmten Methoden abhängig. Die Halbwertzeit ist sehr hoch. Methodenwissen muß ständig an die aktuelle Entwicklung angepaßt werden. In der Informatik haben sich die entscheidenden Methoden alle 8-10 Jahre erheblich geändert. Werkzeugwissen ist methodenabhängig. Werkzeuge haben in der Informatik oft nur eine Lebensdauer von 2-3 Jahren.



# ÜBERSICHT (vorl.)


## 0. Übersicht und Motivation

- Datenstrukturen und Datenbanken
- Voraussetzungen für die Vorlesung

## 1. Anforderungen und Beschreibungsmodelle

- Nachteile von Dateisystemen
- Anforderungen an DBS
- Aufbau von DBS
- Beschreibungsmodelle
  (Fünf-Schichten-Modell, Drei-Ebenen-Beschreibungsarchitektur)

## 2. Logischer DB-Entwurf

- Konzeptioneller DB-Entwurf
- Normalformenlehre (1NF, 2NF, 3NF, 4NF)
- Synthese von Relationen

## 3. Anwendungsprogrammier-Schnittstelle

- Schemaevolution
- Sichtenkonzept
- Kopplung mit einer Wirtssprache
- Übersetzung von DB-Anweisungen
- Eingebettetes / Dynamisches SQL, PSM

## 4. Transaktionsverwaltung

- Transaktionskonzept
- Ablauf von Transaktionen
- Commit-Protokolle

# ÜBERSICHT (2)

## 5. Synchronisation

- Anomalien beim Mehrbenutzerbetrieb
- Theorie der Serialisierbarkeit
- Sperrprotokolle
- Konsistenzebenen

## 6. Logging und Recovery

- Fehlermodell und Recovery-Arten
- Logging-Strategien
- Recovery-Verfahren

## 7. Integritätskontrolle und aktives Verhalten

- Semantische Integritätskontrolle
- Regelverarbeitung in DBS
- Trigger-Konzept von SQL
- Definition und Ausführung von ECA-Regeln

## 8. Datenschutz und Zugriffskontrolle

- Technische Probleme des Datenschutzes
- Zugriffskontrolle in SQL
- Sicherheitsprobleme in statistischen DBs

## 9. Lineare und hierarchische Zugriffspfade

- Externspeicherbasierte Baumstrukturen und Hash-Verfahren
- Navigationsstrukturen
- Indexierungsverfahren, verallgemeinerte Zugriffspfade



# ÜBERSICHT (3)

## 10. Mehrdimensionale Zugriffspfade

- Anforderungen und Verfahrensklassen
- Grid-File
- R-Baum

## 11. Objektorientierung und Datenbanken

- Beschränkungen klassischer Datenmodelle
- Grundkonzepte der Objektorientierung

## 12. Einführung in ein OR-Datenmodell (SQL99)

- Anforderungen, Architekturvorschläge
- Erhöhung der Anfragemächtigkeit
- Rekursion

## 13. Große Objekte

- Anforderungen und Verarbeitung mit SQL
- Lokator-Konzept
- Speicherungsstrukturen, . . .

## 14. Erweiterbares Typsystem

- Umbenannte Typen
- Benutzerdefinierte Datentypen/Funktionen
- Typhierarchien, Subtypen, . . .
- . . .

# LITERATURLISTE

*Bernstein, P.A., Hadzilacos, V., Goodman, N.:* Concurrency Control and Recovery in Database Systems, Addison-Wesley Publ. Comp., 1987.

*Chamberlin, D.:* DB2 Universal Database, Addison-Wesley Publ. Comp., 1999

*Date, C.J.:* An Introduction to Database Systems, Addison-Wesley Publ. Comp., Reading, Mass., 7th Edition, 2000

*Dreßler, H.:* Datenstrukturentwurf – Vom Faktenchaos zur Anwenderdatenbank, Oldenbourg-Verlag, München, 1995

*Guluzan, P., Pelzer, T.:* SQL-99 Complete, Really, R&D Books Miller Freeman, Inc., Lawrence, Kansas, 1999

*Härder, T., Rahm, E.:* Datenbankensysteme – Konzepte und Techniken der Implementierung, Springer-Verlag, Berlin, 1999 (Es sind Hörerscheine erhältlich)

*Heuer, A., Saake, G.:* Datenbanken – Konzepte und Sprachen, 2. Auflage, Int. Thompson Publ. Comp., 2000

*Kemper, A., Eickler, A.:* Datenbanksysteme – Eine Einführung, 3. Auflage, Oldenbourg-Verlag, 1999

*Ramakrishnan, R.:* Database Management Systems, McGraw-Hill, Boston, 1998.

*Saake, G., Heuer, A.:* Datenbanken – Implementierungstechniken, Int. Thompson Publ. Comp., 1999

*Vossen, G.:* Datenbankmodelle, Datenbanksprachen und Datenbankmanagementsysteme, 4. Auflage, Oldenbourg-Verlag, München, 2000

*Wedekind, H.:* Kaufmännische Datenbanken, BI-Wissenschaftsverlag Mannheim, 1993

**ZEITSCHRIFTEN:**

*TODS* Transactions on Database Systems, ACM Publikation (vierteljährlich)

*Information Systems* Pergamon Press (6-mal jährlich)

*The VLDB Journal* (vierteljährlich)

*Informatik - Forschung und Entwicklung* (vierteljährlich)

**TAGUNGSBÄNDE:**

*SIGMOD* Tagungsband, jährliche Konferenz der ACM Special Interest Group on Management of Data

*VLDB* Tagungsband, jährliche Konferenz „Very Large Data Bases“

*IEEE* Tagungsband, jährliche Konferenz „Int. Conf. on Data Engineering“

*GI* Tagungsbände der Tagungen der Gesellschaft für Informatik, Tagungen innerhalb des Fachausschusses 2.5 Informationssysteme

und viele weitere Konferenzreihen



# Datenstrukturen ?

- **Bisher bekannte Datenstrukturen**
  - Felder (Reihungen, Arrays, .... )
  - Verbunde (Tupel, Sätze, Records, .... )
  - Listen
  - Graphen
  - Bäume

  → bisher im **Hauptspeicher**, d. h. Bestand nur für die Dauer einer Programmausführung („transiente“ Daten)

- **hier nun neue Aspekte: Nutzung von Hintergrundspeicher und neuen Operationen**
  - **Persistenz:** Werte bleiben über Programmende, Sitzungsende, Betriebssystem-Uptime, Rechnereinschaltung, .... hinaus erhalten
  - andere Arten des Zugriffs: Lese- und Schreiboperationen, in vorgegebenen Einheiten von Blöcken und Sätzen

    → Strukturen und zugehörige Algorithmen (Suchen, Sortieren), die im Hauptspeicher optimal sind, sind es auf Sekundärspeicher nicht unbedingt!
  - gezielter, wertabhängiger Zugriff auch auf sehr große Datenmengen
  - umfangreiche Attributwerte (z. B. Bilder, Texte)

## Datenmodelle ?

- **Mengen von Konstruktoren**

  zur (abstrakten) Erzeugung von Datenstrukturen
  mit darauf definierten Operatoren

  z. B. Tabellen (Relationen): Mengen von gleichartig strukturierten Tupeln

```
CREATE TABLE STUDENT
        (MATRIKELNUMMER       INTEGER,
        NACHNAME              VARCHAR(40),
        FBNUMMER              INTEGER,
        GEBURTSDATUM          CHAR(8),
        .... )

CREATE TABLE FACHBEREICH
        (FBNUMMER             INTEGER,
        FBNAME                VARCHAR(20),
        DEKAN                 PROF,
        .... )
```

  nicht nur ein einzelner Satz, sondern Menge

- **Wichtige Rolle von Beziehungen**

  in einigen Datenmodellen auch sehr spezielle Arten von **Beziehungen** zwischen Sätzen und/oder Tupeln: Hierarchien, Zusammensetzungen, funktionale Zuordnungen u.v.a.m.

- **Modellbegriff**

  vorgegebene Menge von (sprachlichen) Ausdrucksmitteln, mit denen Diskursbereich beschrieben (erfaßt) werden muß

  → auch Programmiersprachen und Betriebssysteme haben ein „Datenmodell“



## Datenbanken ?

- **große Datenmengen:**
  - auch, aber **nicht immer** entscheidend

- **Datenunabhängigkeit (der Anwendungen):**
  - Benutzung der Daten, ohne Details der systemtechnischen Realisierung zu kennen (abstraktes „Datenmodell“, z. B. Tabellen)
  - einfache Handhabung der Daten, mächtige Auswertungsoperationen

- **Offenheit der Daten für neue Anwendungen (Anwendungsneutralität der Speicherung)**
  - symmetrische Organisationsformen (keine Bevorzugung einer Verarbeitungs- und Auswertungsrichtung)
  - explizite Darstellung der Annahmen (nicht in den Anwendungsprogrammen verstecken)
  - Redundanzfreiheit aus der Sicht der Anwendung:
    keine wiederholte Speicherung in unterschiedlicher Form
    für verschiedene Anwendungen
  - Konsistenzüberwachung durch das Datenbanksystem

## Datenbanken ? (2)

- **Transaktionskonzept mit Garantie von ACID-Eigenschaften**
  - Atomizität (*atomicity*)
  - Konsistenz (*consistency*)
  - Isolation (*isolated execution*)
  - Dauerhaftigkeit (*durability*)

- **Ausfallsicherheit**
  - Aufzeichnung redundanter Daten im Normalbetrieb
  - Replikation von Datenstrukturen
  - automatische Reparatur der Datenbestände nach Programm-, System- und Gerätefehlern
    - Rückgängigmachen unvollständiger Transaktionen, so daß sie wiederholt werden können
    - Wiederherstellen der Ergebnisse vollständiger Transaktionen, so daß sie nicht wiederholt werden müssen

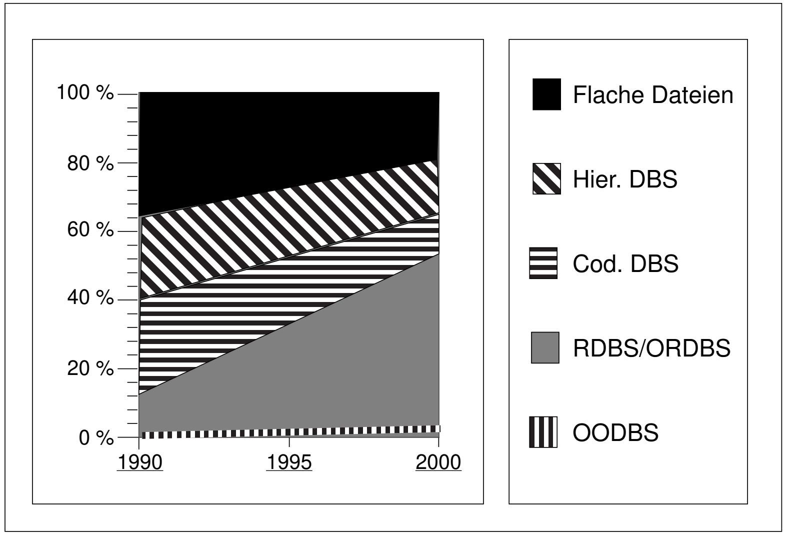
- **Mehrbenutzerbetrieb**
  - gleichzeitiger (zeitlich eng verzahnter) Zugriff verschiedener Anwendungen und Benutzer auf gemeinsame Daten
  - Synchronisation, d. h. Vermeidung von Fehlern in der wechselseitigen Beeinflussung
  - Kooperation über gemeinsame Daten mit hohem Aktualitätsgrad



## Verteilung von DBS und Dateien

- **Es gibt verschiedenartige Datenmodelle und die sie realisierenden DBS**
  - relational und objekt-relational (RDBS/ORDBS)
  - hierarchisch (DBS nach dem Hierarchiemodell)
  - netzwerkartig (DBS nach dem Codasyl-Standard)
  - objektorientiert (OODBS)

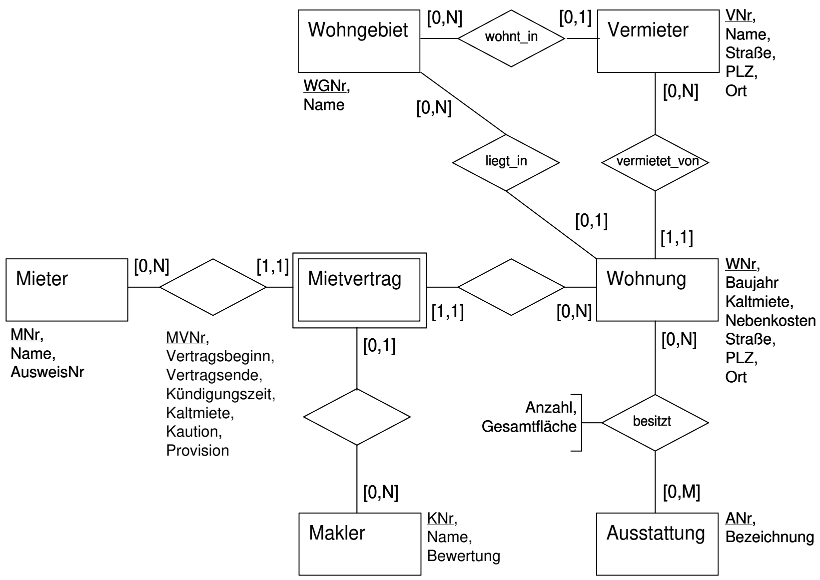
| | 1990 | 1995 | 2000 |
|---|---|---|---|
| OODBS | 1 % | 2 % | 3 % |
| RDBS/ORDBS | 9 % | 25 % | 50 % |
| Cod. DBS | 30 % | 22 % | 12 % |
| Hier. DBS | 25 % | 20 % | 15 % |
| Flache Dateien | 35 % | 31 % | 20 % |
| Gesamt | 100 % | 100 % | 100 % |



# Voraussetzungen

- **Beherrschung von**
  - Informationsmodellen (erweitertes ER-Modell)
  - Relationenmodell und Relationenalgebra
  - SQL-92 als Standardsprache

- **Beispiel eines ER-Schemas**






# Voraussetzungen (2)

- **Zugehöriges SQL-Schema**

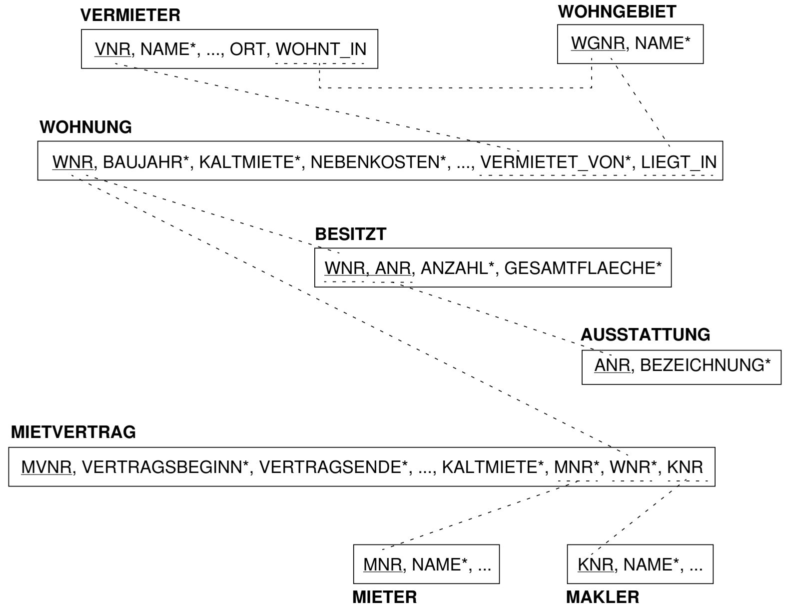
```
CREATE TABLE WOHNGEBIET (
       WGNR           WOHNGEBIETSNUMMER   PRIMARY KEY,
       NAME           CHAR(30)            NOT NULL);

CREATE TABLE VERMIETER (
       VNR            VERMIETERNUMMER     PRIMARY KEY,
       NAME           CHAR(30)            NOT NULL,
       STRASSE        CHAR(30),
       PLZ            POSTLEITZAHL,
       ORT            CHAR(30),
       WOHNT_IN       WOHNGEBIETSNUMMER,
       FOREIGN KEY (WOHNT_IN) REFERENCES WOHNGEBIET (WGNR));

CREATE TABLE WOHNUNG (
       WNR            WOHNUNGSNUMMER      PRIMARY KEY,
       BAUJAHR        INT                 DEFAULT 1800 NOT NULL
                                          CHECK (BAUJAHR >= 1800),
       KALTMIETE      INT                 NOT NULL,
       NEBENKOSTEN    INT                 NOT NULL,
       STRASSE        CHAR(30),
       PLZ            POSTLEITZAHL,
       ORT            CHAR(30),
       LIEGT_IN       WOHNGEBIETSNUMMER,
       VERMIETET_VON  VERMIETERNUMMER     NOT NULL,
       FOREIGN KEY (LIEGT_IN) REFERENCES WOHNGEBIET (WGNR),
       FOREIGN KEY (VERMIETET_VON) REFERENCES VERMIETER (VNR));

CREATE TABLE AUSSTATTUNG (
       ANR            AUSSTATTUNGSNUMMER PRIMARY KEY,
       BEZEICHNUNG    CHAR(30)            NOT NULL);
/* Zur Ausstattung gehören z. B. Schlafzimmer, Bad, Balkon, usw. */
```

# Voraussetzungen (3)

```
CREATE TABLE BESITZT (
        WNR             WOHNUNGSNUMMER,
        ANR             AUSSTATTUNGSNUMMER,
        ANZAHL          INT                   DEFAULT 1 NOT NULL,
        GESAMTFLAECHE   INT                   NOT NULL, /* In m^2 */
        PRIMARY KEY (WNR, ANR),
        FOREIGN KEY (WNR) REFERENCES WOHNUNG (WNR),
        FOREIGN KEY (ANR) REFERENCES AUSSTATTUNG (ANR));

CREATE TABLE MAKLER (
        KNR             MAKLERNUMMER          PRIMARY KEY,
        NAME            CHAR(30)              NOT NULL,
        BEWERTUNG       BEWERTUNGSSKALA);

CREATE TABLE MIETER (
        MNR             MIETERNUMMER          PRIMARY KEY,
        NAME            CHAR(30)              NOT NULL,
        AUSWEISNR       CHAR(30)              NOT NULL);

CREATE TABLE MIETVERTRAG (
        MVNR            MIETVERTRAGSNUMMER    PRIMARY KEY,
        VERTRAGSBEGINN  DATE                  NOT NULL,
        VERTRAGSENDE    DATE                  NOT NULL,
        KUENDIGUNGSZEIT INT                   DEFAULT 90, /* In Tagen */
        KALTMIETE       INT                   NOT NULL,
        KAUTION         INT                   DEFAULT 3 NOT NULL,
        PROVISION       INT                   DEFAULT 2 NOT NULL,
        /* Werte fuer Kaution und Provision seien n-Mal Kaltmiete */
        MNR             MIETERNUMMER          NOT NULL,
        WNR             WOHNUNGSNUMMER        NOT NULL,
        KNR             MAKLERNUMMER          DEFAULT NULL,
        FOREIGN KEY (MNR) REFERENCES MIETER (MNR) ON DELETE CASCADE,
        FOREIGN KEY (WNR) REFERENCES WOHNUNG (WNR) ON DELETE CASCADE,
        FOREIGN KEY (KNR) REFERENCES MAKLER (KNR)
                                      ON DELETE SET NULL);
```



# Voraussetzungen (4)




* = NOT NULL

# Voraussetzungen (5)

- **Anfragen in SQL**

1. Lösche alle Vermieter, die keine einzige Wohnung vermieten und deren Wohngebiet unbekannt ist.

```
DELETE   FROM VERMIETER V
WHERE    NOT EXISTS
         (SELECT  *
          FROM    WOHNUNG W
          WHERE   W.VERMIETET_VON = V.VNR
         )
         AND      V.WOHNT_IN IS NULL
```

2. Erhöhe die Kaltmiete von Wohnungen, die später als 1990 gebaut wurden und im Uni-Wohngebiet in KL liegen, jeweils um 20% der zugehörigen Nebenkosten.

```
UPDATE   WOHNUNG W
SET      W.KALTMIETE = W.KALTMIETE + W.NEBENKOSTEN * 0.2
WHERE    W.BAUJAHR > 1990
AND      W.ORT = 'KL'
AND      EXISTS
         (SELECT  *
          FROM    WOHNGEBIET WG
          WHERE   WG.WGNR = W.LIEGT_IN
          AND     WG.NAME = 'Uni-Wohngebiet'
         )
```

3. Reduziere die Nebenkosten aller Wohnungen jeweils um 10%, die mindestens 2 Schlafzimmer oder eine Schlafzimmergesamtfläche von < 50 $m^2$ besitzen und deren Kaltmiete niedriger als ihre zugehörigen Nebenkosten ist.

```
UPDATE   WOHNUNG W
SET      W.NEBENKOSTEN = W.NEBENKOSTEN * 0.9
WHERE    W.KALTMIETE < W.NEBENKOSTEN
AND      EXISTS
         (SELECT  *
          FROM    BESITZT B, AUSSTATTUNG A
          WHERE   B.WNR = W.WNR
          AND     B.ANR = A.ANR
          AND     A.BEZEICHNUNG = 'Schlafzimmer'
          AND     (B.ANZAHL >= 2 OR B.GESAMTFLAECHE < 50)
         )
```



# Voraussetzungen (6)

- **Anfragen in SQL**

4. Finde alle zur Zeit vermieteten Wohnungen, deren Kaltmiete größer ist als die Gesamtsumme der vom (einzelnen) Mieter zu zahlenden Kaltmieten.
(Eine Wohnung kann mehrere gleichzeitig gültige Mietverträge haben (z. B. WG))

```
SELECT   W.*
FROM     WOHNUNG W
WHERE    W.KALTMIETE >
         (SELECT  SUM(MV.KALTMIETE)
          FROM    MIETVERTRAG MV
          WHERE   MV.WNR = W.WNR
          AND     MV.VERTRAGSBEGINN <= TODAY
          AND     MV.VERTRAGSENDE >= TODAY
         )
```

5. Finde alle Wohnungen mit ihrer Gesamtfläche, die ein Bad und wenigstens ein Ausstattungsmerkmal mit der Gesamtfläche von 30 $m^2$ oder größer besitzen.

```
SELECT   B.WNR, SUM(B.GESAMTFLAECHE)
FROM     BESITZT B
WHERE    EXISTS
         (SELECT  *
          FROM    AUSSTATTUNG A
          WHERE   B.ANR = A.ANR
          AND     A.BEZEICHNUNG = 'Bad'
         )
GROUP BY B.WNR
HAVING   MAX(B.GESAMTFLAECHE) >= 30
```

6. Finde alle Mieter, die zur Zeit mehr als eine Wohnung im Uni-Wohngebiet mieten.

```
SELECT   M.*
FROM     MIETER M
WHERE    1 <
         (SELECT  COUNT(*)
          FROM    MIETVERTRAG MV, WOHNUNG W, WOHNGEBIET WG
          WHERE   MV.MNR = M.MNR
          AND     MV.WNR = W.WNR
          AND     W.LIEGT_IN = WG.WGNR
          AND     WG.NAME = 'Uni-Wohngebiet'
          AND     MV.VERTRAGSBEGINN <= TODAY
          AND     MV.VERTRAGSENDE >= TODAY
         )
```

## Voraussetzungen (7)

- **Anfragen in SQL**

7. Finde alle Wohnungen, die 3 Schlafzimmer und 2 Bäder besitzen.

   In SQL:

```
SELECT   W.*
FROM     WOHNUNG W
WHERE    EXISTS
         (SELECT  *
          FROM    BESITZT B, AUSSTATTUNG A
          WHERE   B.WNR = W.WNR
          AND     B.ANR = A.ANR
          AND     A.BEZEICHNUNG = 'Schlafzimmer'
          AND     B.ANZAHL = 3
         )
AND      EXISTS
         (SELECT  *
          FROM    BESITZT B, AUSSTATTUNG A
          WHERE   B.WNR = W.WNR
          AND     B.ANR = A.ANR
          AND     A.BEZEICHNUNG = 'Bad'
          AND     B.ANZAHL = 2
         )
```

Wie läßt sich diese Anfrage ohne EXISTS-Prädikate ausdrücken?

8. Finde alle Wohnungen, die alle Ausstattungsmerkmale haben.

   In SQL:

```
SELECT   W.*
FROM     WOHNUNG W
WHERE    NOT EXISTS
         (SELECT  *
          FROM    AUSSTATTUNG A
          WHERE   NOT EXISTS
                  (SELECT  *
                   FROM    BESITZT B,
                   WHERE   B.WNR = W.WNR
                   AND     B.ANR = A.ANR
                  )
         )
```

Alternative Formulierung: Finde alle Wohnungen, so daß es kein Ausstattungsmerkmal gibt, das die Wohnung nicht besitzt.



## The Big Picture