Übung Datenbanksysteme
Hierarchische DBMS

2.2.2004

Final-Klausur

- Termin: Freitag, 13. Februar 2004
- Uhrzeit: ab 16.15 Uhr
- Ort: Hörsaal PH HS 1 (PH 2501)

- In der Übung am 9.2.2004:
  Wiederholen von Aufgaben
- Vorschläge bis 6.2.2004 per Email an:
  bauermi@in.tum.de

Trigger

- Zwei Tabellen:
- **Temps** (place VARCHAR(20), temp Integer)
- **Extremes** (place VARCHAR(20),
  hightemp Integer, highdate Date,
  lowtemp Integer, lowdate Date)
- Event, Condition, Action (ECA-Rule)

Trigger (2a)

- CREATE TRIGGER temp_trig1
  AFTER UPDATE ON temps
  REFERENCING NEW AS newrow
  FOR EACH ROW MODE DB2SQL
  WHEN (newrow.temp >
  (SELECT hightemp FROM extremes
  WHERE place = newrow.place) OR
  (SELECT hightemp FROM extremes
  WHERE place = newrow.place) IS NULL)…

Trigger (2b)

… UPDATE extremes

SET hightemp = newrow.temp,

highdate = CURRENT DATE

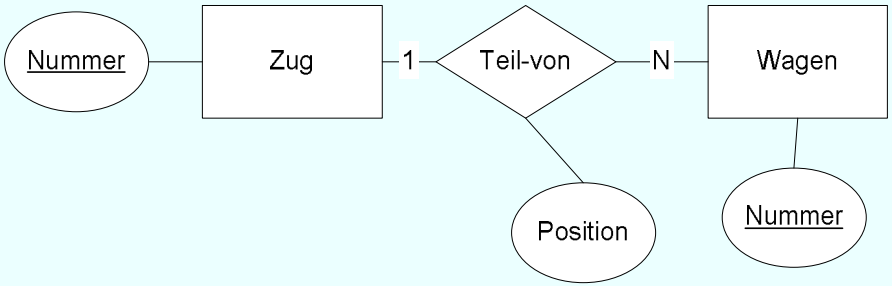
WHERE place = newrow.place;

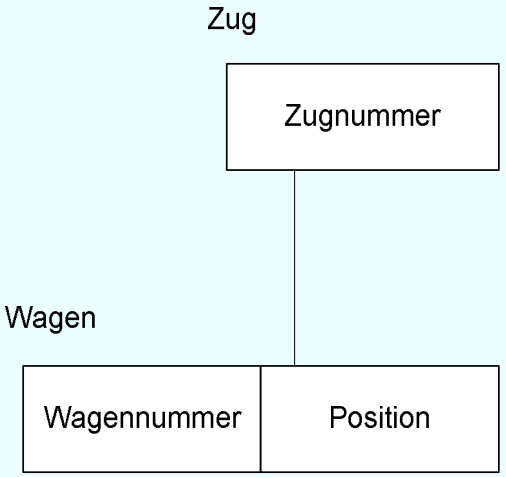
Hierarchisches Modell

- Älteres Modell als relationales Modell (ca 1960)
- In der Industrie (Banken, Versicherungen) noch weit verbreitet (IBM IMS)
- Konzept:
  - Alle Beziehungen sind Vater-Kind-Beziehung
  - Kinder werden geclustert hinter Vater gespeichert
  - Zugriff in enger Kopplung zur Programmiersprache

Faustregeln zur Umsetzung

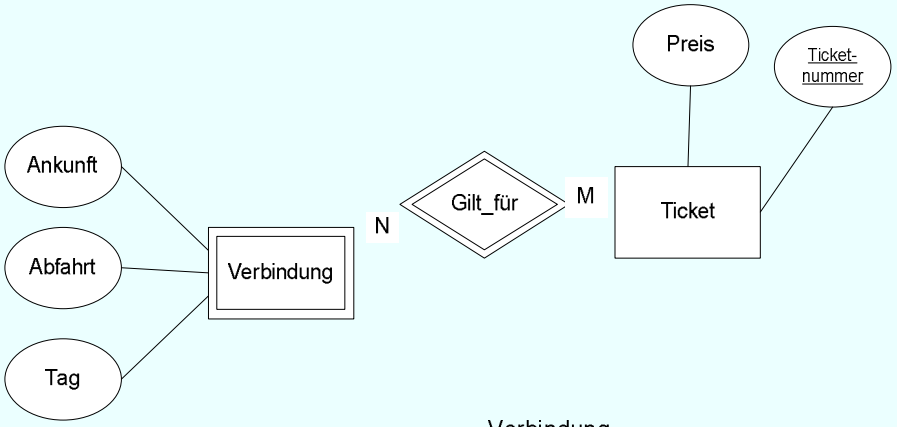
- 1-N (auch 1-1)
- Seite mit 1 wird Vaterknoten,
- Seite mit N wird Kind

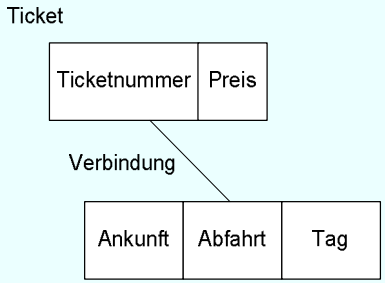


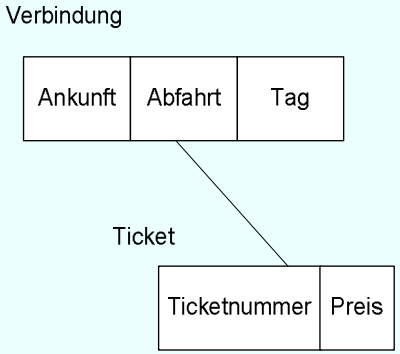


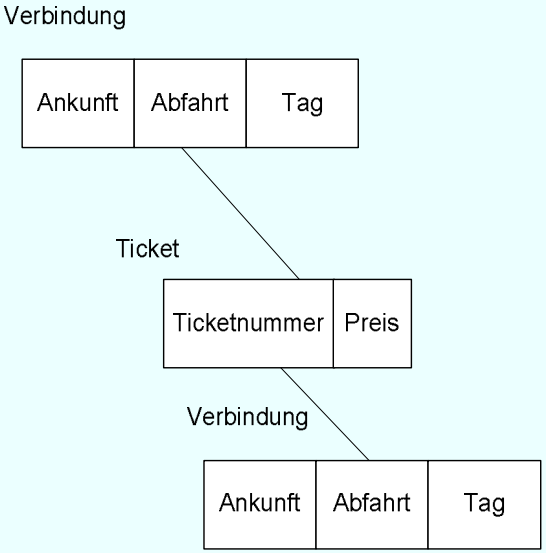
Problemstellung bei N-M

- Wer ist Vater, wer ist Kind:



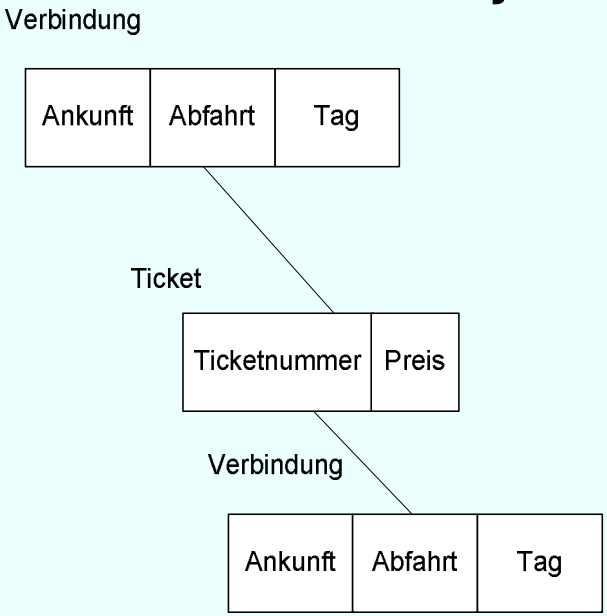


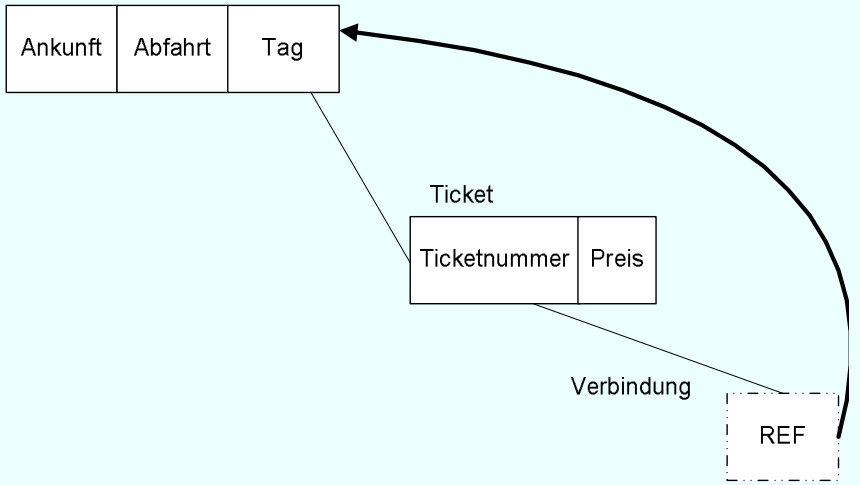




Vereinfachung durch Referenzen

- Problem der strikten Hierarchie:
  Hohe Redundanzen
- Vermeidung der Redundanzen durch Verweis auf bestehendes Objekt





Algorithmus zur Umsetzung

- Bisher nach Faustregeln und mit Nachbearbeitung
- Algorithmus vereinfacht Verfahren
- Vorveränderung:
- Jede N-M durch eine 1-M- und eine N-1-Beziehung ersetzen

Algorithmus (1)

Hauptprogramm: wähle Entität n mit vielen auf n zeigenden Links, ohne Links weg von n zu nicht markierten Knoten:

```
begin       <lösche alle Markierungen>
  while ¬   <alle Knoten markiert> do
                    wähle unmarkiertes n;
                    build-tree (n)
            od
end
```

Algorithmus (2)

```
procedure build-tree (n)
   begin     <markiere n>;
       for each Kante m   ->   n in E/R Schema  do
              begin        <n wird Vaterknoten,
                           ziehe von n zu m,        ;
                    if ¬ <markiert m> then
                           build-tree (m)
                    else m Referenzknoten
                    fi
       end
   end
```

Zugriff auf hierarchische DBMS

Prozeduraufrufe aus Wirtssprache
(jeweils mit Parameter für Prädikate)

Syntax nicht einheitlich, Semantik schon

- GET UNIQUE ≈ **GU**
  Holt ersten Datensatz
- GET NEXT ≈ **GN**
  Holt folgende Datensätze nach **GU**
- GET NEXT WITHIN PARENT ≈ **GNP**
  Holt ersten oder folgende Datensätze unterhalb des aktuellen Datensatzes
- DBSTATUS: Status der letzten Operation, 0 wenn erfolgreich

Aufgabe 3a)

Ermitteln Sie alle Wagen, die an vierter Position in einem Zug sind.

- Idee:
  - Schleife über alle Züge
  - Suche einen Wagen mit Position 4 im Zug (es gibt maximal einen !)
  - Wenn gefunden, ins Ergebnis
  - Wenn nicht, weiter mit nächstem Zug

Aufgabe 3a) (2)

```
Vector ergebnis;
GU ZUG
While(DBSTATUS == 0) {
   Wagen w = GNP WAGEN (Position == 4)
   If (DBSTATUS == 0) {
        Ergebnis.add(w)
   }
   GN ZUG
}
```

Aufgabe 3b)

Ermitteln Sie alle Züge, die Raucherplätze am Fenster in der ersten Klasse haben

Idee:

- Schleife über alle Züge
- Schleife über alle Wagen – Abbruch, sobald ein Wagen Plätze nach den Kriterien enthält
- Suche nach einem Platz, der die Kriterien erfüllt
- Wenn gefunden, Ende der Wagenschleife, ins Ergebnis einfügen
- Wenn nicht, weiter mit nächstem Wagen

Aufgabe 3b (2)

```
Vector ergebnis;
Boolean found = false;
Zug z = GU ZUG
WHILE (DBSTATUS == 0) {
    found = false;
    GNP WAGEN
    WHILE (DBSTATUS == 0 && !found) {
        GNP Platz (Raucher == True, Klasse == 1, Fenster == True)
        IF (DBSTATUS == 0)  {
            found = true;
        }
        GNP WAGEN
     }
     if (found) {
            ergebnis.add(z)
    }
    z = GN Zug
}
```

Aufgabe 3b (3)

Zum Vergleich SQL:

```
SELECT z.Zugnummer
FROM Zug z, Wagen w, Platz p
WHERE z.Zugnummer = w.Zugnummer
AND w.Wagennummer = p.Wagennummer
AND p.Raucher = 't'
AND p.Fenster = 't'
AND p.Klasse = 1
```

Vorteile – Nachteile ??