Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

Juni/ Juli 2022
Trigger, Transaktionen und Normalisierung

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

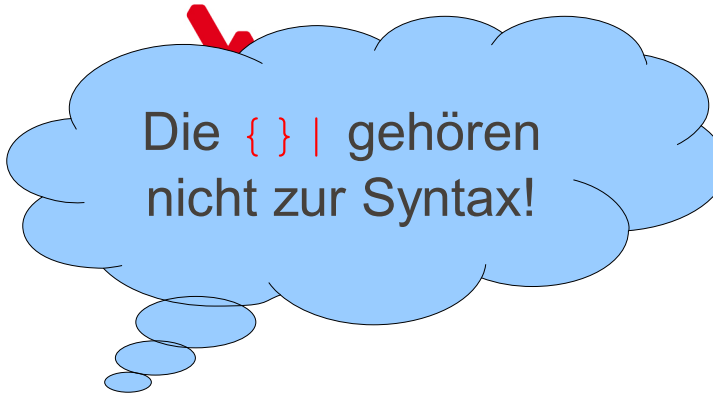
01
TRIGGER

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

TRIGGER

- Ereignis-Aktionen-Regeln
- Dieses Programm wird *Trigger* genannt
- Bei Ausführung von DB-Operation wird eine Aktion ausgeführt
- *Triggerfeuernde* Operationen sind i.d.R. Update-Operationen
- In Triggern kann sowohl auf neue als auch alte Daten zugegriffen werden (*Differenzenrelationen*)

TRIGGER




- ```
  CREATE TRIGGER
  trigger_name
     { BEFORE | AFTER }
     { INSERT | DELETE |
  UPDATE }
     ON tbl_name FOR EACH
  ROW
     trigger_stmt
  ```
- Aktivierung vor oder nach `INSERT`, `DELETE` oder `UPDATE`
- Ein Trigger gilt für eine spezielle Tabelle
- Ein Trigger wird vor oder nach jedem Einfügen eines Datensatzes ausgeführt (`FOR EACH ROW`)
  ```

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

TRIGGER

- `trigger_stmt` ist die Anweisung, die ausgeführt wird
- Mit `NEW` und `OLD` kann auf neuen bzw. alten Datensatz zugegriffen werden

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

TRIGGER

- Neue Tabelle:
- ```
  CREATE TABLE anfangsgehalt (
    genre varchar(20) NOT NULL ,
    gehalt int(5) NOT NULL,
    PRIMARY KEY (genre)
  ) ENGINE = InnoDB;
  ```
- Daten:
- ```
  INSERT INTO anfangsgehalt VALUES ('Drama', 100);
  ```

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

TRIGGER

- Weitere Tabelle:
- ```
  CREATE TABLE schauspieler (
    name varchar(20) NOT NULL,
    genre varchar(20) NOT NULL ,
    gehalt int(5) NULL,
    PRIMARY KEY (name)
  )ENGINE = InnoDB;
  ```
  ```

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

TRIGGER

- Automatische Berechnung des Anfangsgehalts (`BEFORE`-Trigger)
-
```
CREATE TRIGGER trg_schauspieler1
  BEFORE INSERT ON schauspieler
  FOR EACH ROW
  SET NEW.gehalt=
    (SELECT gehalt
       FROM anfangsgehalt
       WHERE genre=NEW.genre)
```

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

TRIGGER

- Einen Datensatz ohne Gehalt einfügen
- ```
  INSERT INTO schauspieler
    (name, genre)
    VALUES ('Hans', 'Drama')
  ```
- Gehalt wird durch Trigger ergänzt

name	genre	gehalt
Hans	Drama	100
  ```

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

TRIGGER

- Begrenze Gehaltszuwachs auf 30%
  (`BEFORE`-Trigger)
- ```
  CREATE TRIGGER trg_schauspieler2
    BEFORE UPDATE ON schauspieler
    FOR EACH ROW
    SET NEW.gehalt =
      IF(NEW.gehalt > 1.3 * OLD.gehalt,
         1.3 * OLD.gehalt,
         NEW.gehalt)
  ```
  ```

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

TRIGGER

- Zu hohe Gehaltssteigerung setzen:
- ```
  UPDATE schauspieler
     SET gehalt=200
     WHERE name='Hans'
  ```
- Gehalt durch Trigger angepasst:

name	genre	gehalt
Hans	Drama	130
  ```

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

TRIGGER

- Weitere Tabelle:
- ```
  CREATE TABLE gehaltaenderung (
    name varchar(20) NOT NULL ,
    aenderung int(5) NOT NULL ,
    zeit timestamp NOT NULL
  ) ENGINE = InnoDB;
  ```
  ```

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

TRIGGER

- Protokollierung der Änderung vom Gehalt (`AFTER`-Trigger)
-
```
CREATE TRIGGER
trg_schauspieler3
  AFTER UPDATE ON
schauspieler
  FOR EACH ROW
  INSERT INTO
gehaltaenderung
    (name, aenderung)
   VALUES (NEW.name,
            NEW.gehalt -
OLD.gehalt)
```

`timestamp` wird automatisch gesetzt.

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

TRIGGER

- Anpassung des Gehalts:
- ```
  UPDATE schauspieler
     SET gehalt=150
     WHERE name='Hans'
  ```
- Automatische Protokollierung:

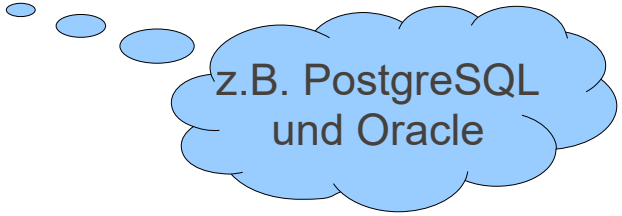
name	aenderung	zeit
Hans	20	2009-11-16 00:30:23
  ```

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

TRIGGER

- Ein Trigger kann alternativ auch nur für eine Anweisung (nicht Zeile) ausgeführt werden:
  `FOR EACH STATEMENT`

- Es kann eine Alternativanweisung angegeben werden:
  `INSTEAD OF`



Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

02
Transaktionen

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

EINFÜHRUNG

- Kapselung mehrerer SQL-Kommandos als Transaktion
- Transaktionen stellen sicher, dass Gruppen von SQL-Kommandos vollständig oder gar nicht ausgeführt werden
- Sicherstellung, dass Daten nicht gleichzeitig von anderen Benutzern verändert werden (InnoDB zeilenweise)
- → Datensystem sicherer machen

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

EINFÜHRUNG

- Beispiel Überweisung:

```
UPDATE account SET value=value-100
       WHERE no=123
```

Was ist, wenn hier
das System abstürzt?

```
UPDATE account SET value=value+100
       WHERE no=456
```

Eine Belastung eines Kontos
darf ohne Gutschrift nicht geschehen!

EINFÜHRUNG

- ACID:
  Atomicity, Consistency, Isolation, Durability
  (von InnoDB eingehalten)
  - Atomicity: Unteilbarkeit von Transaktionen (Transaktionen vollständig oder gar nicht ausführen)
  - Consistency: Am Ende einer Transaktion ist DB in konsistentem (fehlerfreien) Zustand.
    Wenn eine Transaktion zur Verletzung von Gültig-keitsregeln führt, muss sie abgebrochen werden.
  - Isolation: Mehrere gleichzeitige Transaktionen dürfen sich nicht beeinflussen bzw. stören
    - Transaktion sieht DB immer im Anfangszustand (außer Änderungen durch Transaktion)
    - Isolierung geht auf Kosten der Geschwindigkeit
  - Durability: Wenn Transaktion fertig, müssen Daten gespeichert sein!
    - Auch wenn es unmittelbar danach zum Absturz kommt
    - Schreiben von Protokolldateien, auch auf Kosten der Performance

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

TRANSAKTIONEN

- MySQL (nur InnoDB) arbeitet generell im Auto-Commit-Modus
  `SET AUTOCOMMIT=1` bzw. `0`
- Starten einer Transaktion mit `BEGIN` oder `START TRANSACTION`
- Beende mit `COMMIT` (bestätigen) oder `ROLLBACK` (widerrufen)

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

TRANSAKTIONEN

- Keine hierarchischen Transaktionen möglich
- Wird Verbindung zu Datenbank vor Transaktionsende beendet, wird Transaktion widerrufen
- Transaktionen automatisch durch `ALTER TABLE`, `CREATE TABLE`, `DROP`, `LOCK`/`UNLOCK`, `TRUNCATE` akzeptiert

TRANSAKTIONEN

BEISPIEL

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

- Tabelle:
- ```
  CREATE TABLE account (
     no INT NOT NULL,
     value INT NOT NULL,
     PRIMARY KEY (no)
  ) ENGINE = InnoDB;
  ```
  ```

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

TRANSAKTIONEN

- Beispiel Überweisung:

```
START TRANSACTION;

  UPDATE account SET value=value-100
         WHERE no=123;
  UPDATE account SET value=value+100
         WHERE no=456;

COMMIT;
```

TRANSAKTIONEN
BEISPIEL

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

Verbindung A

- `INSERT INTO account VALUES (1, 10);`
- `INSERT INTO account VALUES (2, 20);`
- `SELECT * FROM account;`

```
+----+-------+
| no | value |
+----+-------+
|  1 |    10 |
|  2 |    20 |
+----+-------+
```

- `START TRANSACTION;`

Verbindung B

TRANSAKTIONEN
BEISPIEL

Hochschule RheinMain
University of Applied Sciences
Wiesbaden Rüsselsheim

Verbindung A

- `UPDATE account SET value=11 WHERE no=1;`
- `SELECT * FROM account;`

```
+----+-------+
| no | value |
+----+-------+
|  1 |    11 |
|  2 |    20 |
+----+-------+
```

Verbindung B

- `SELECT * FROM account;`

```
+----+-------+
| no | value |
+----+-------+
|  1 |    10 |
|  2 |    20 |
+----+-------+
```

- `START TRANSACTION;`

TRANSAKTIONEN
BEISPIEL

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

Verbindung A

- `SELECT * FROM account;`

```
+----+-------+
| no | value |
+----+-------+
|  1 |    11 |
|  2 |    20 |
+----+-------+
```

-

- `COMMIT;`

Verbindung B

- `UPDATE account SET value=21 WHERE no=2;`
- `SELECT * FROM account;`

```
+----+-------+
| no | value |
+----+-------+
|  1 |    10 |
|  2 |    21 |
+----+-------+
```

- `UPDATE account SET value=value+3 WHERE no=1;`

TRANSAKTIONEN
BEISPIEL

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

Verbindung A

- `SELECT * FROM account;`

```
+----+-------+
| no | value |
+----+-------+
|  1 |    11 |
|  2 |    20 |
+----+-------+
```

- `SELECT * FROM account;`

```
+----+-------+
| no | value |
+----+-------+
|  1 |    11 |
|  2 |    20 |
+----+-------+
```

Verbindung B

- `SELECT * FROM account;`

```
+----+-------+
| no | value |
+----+-------+
|  1 |    14 |
|  2 |    21 |
+----+-------+
```

- `ROLLBACK;`
- `SELECT * FROM account;`

```
+----+-------+
| no | value |
+----+-------+
|  1 |    11 |
|  2 |    20 |
+----+-------+
```

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

SAVEPOINTS

- Innerhalb einer Transaktion können benannte Savepoints definiert werden: `SAVEPOINT name`
- Sobald `ROLLBACK TO SAVEPOINT name` aufgerufen wird, werden alle Kommandos bis zum Savepoint akzeptiert und alle danach widerrufen.
- Savepoints können nur innerhalb einer Transaktion genutzt werden
- Durch `COMMIT` oder `ROLLBACK` werden alle Savepoints gelöscht

SAVEPOINTS

BEISPIEL

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

- `START TRANSACTION;`
- `SELECT * FROM account;`

```
+----+-------+
| no | value |
+----+-------+
|  1 |    11 |
|  2 |    20 |
+----+-------+
```

- `UPDATE account SET value=12 WHERE no=1;`
- `SAVEPOINT one;`
- `UPDATE account SET value=21 WHERE no=2;`

SAVEPOINTS
BEISPIEL

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

- `SELECT * FROM account;`

```
+----+-------+
| no | value |
+----+-------+
|  1 |    12 |
|  2 |    21 |
+----+-------+
```

- `ROLLBACK TO SAVEPOINT one;`
- `SELECT * FROM account;`

```
+----+-------+
| no | value |
+----+-------+
|  1 |    12 |
|  2 |    20 |
+----+-------+
```

- `COMMIT;`

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

LITERATUR

- Kofler, Michael: MySQL 5, 3. Auflage, Addison-Wesley, 2005
- Vossen, Gottfried: Datenmodelle, Datenbank-sprachen und Datenbankmanagementsysteme, 5. Auflage, Oldenburg Wissenschaftsverlag, 2008
- Lubkowitz, M: Webseiten programmieren und gestalten, Galileo Press, 2004
- Oracle: MySQL 5.7 Reference Manual, https://dev.mysql.com/doc/refman/5.7/en/

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

03
NORMALISIERUNG

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

EINFÜHRUNG

- Sechs Normalformen für relationale Datenbanken
- U.a. von Edgar Frank Codd entwickelt
- Regeln zur Zuordnung von Attributen zu Relationen
- **Ziel:**
- Konsistenzerhöhung durch Redundanz-vermeidung
- Wir behandeln hier 1., 2., 3. NF und BCNF

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

FUNKTIONALE ABHÄNGIGKEITEN

Regeln:

- *Eine funktionale Abhängigkeit (FD) im Bezug auf zwei Attributmengen X und Y einer Relation liegt dann vor, wenn der Attributwert von X den Attributwert von Y festlegt. Y ist funktional abhängig von X:*

$$X \rightarrow Y$$

(Y muss atomar sein!)

- *Eine funktionale Abhängigkeit $\{X_1, X_2\} \rightarrow Y$ wird als volle funktionale Abhängigkeit bezeichnet, wenn Y nicht von X_1 oder X_2 abhängig ist.*
- *Ein Attribut X ist ein Schlüsselkandidat, wenn X voll funktional abhängig ist.*
  Ein Attribut X heißt Nicht-Schlüsselattribut, wenn es in keinem Schlüsselkandidaten enthalten ist.
- *Ein Attribut X ist ein Primeattribut, wenn es ein Teil von einem Schlüssel ist.*

BEISPIEL: FUNKTIONALE ABHÄNGIGKEITEN (FD)

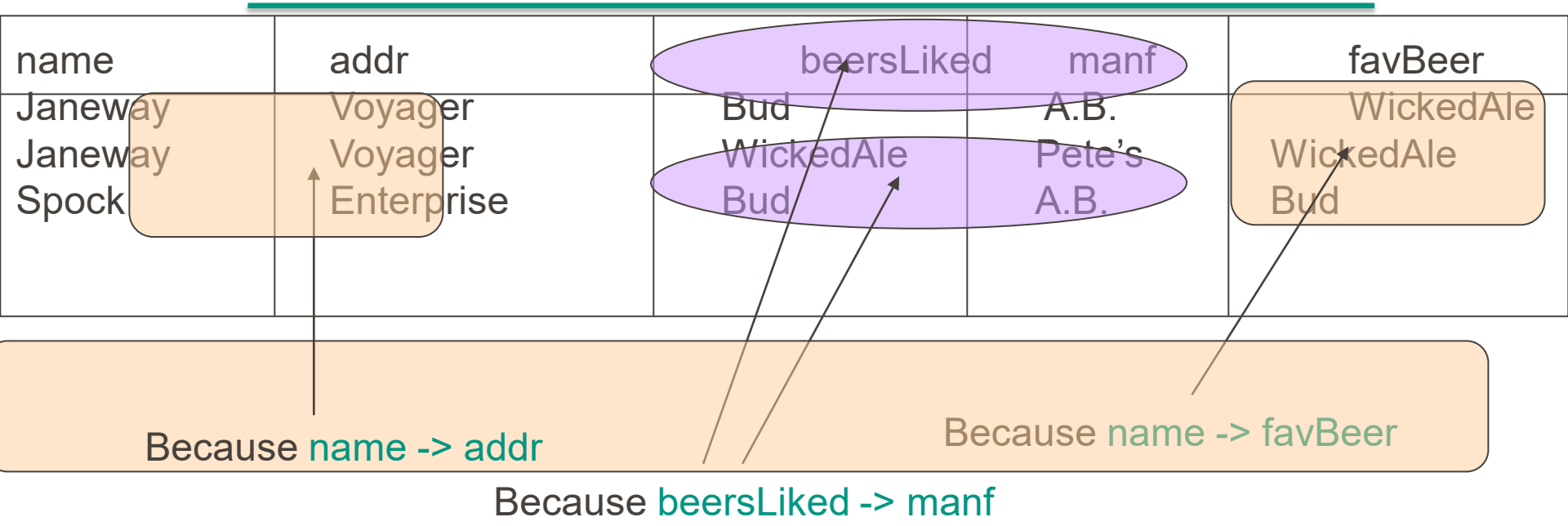
Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

Drinkers(name, addr, beersLiked, manf, favBeer)

- Anzunehmende FDs sind
  1. name -> addr favBeer
     - Anmerkung: ist das Gleiche wie name -> addr und name -> favBeer.
  2. beersLiked -> manf

BEISPIEL: MÖGLICHE DATEN

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim



Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

SCHLÜSSEL EINER RELATION

- *K* ist ein Superschlüssel der Relation R wenn

  K alle Attribute von *R funktionale determiniert*.

- *K* ist ein Schlüssel der Relation R wenn

  K ein Superschlüssel ist, jedoch keine Teilmenge von K ein Superschlüssel ist

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

BEISPIEL: SUPERSCHLÜSSEL

Drinkers(name, addr, beersLiked, manf, favBeer)

- {name, beersLiked} ist ein Superschlüssel, da sie zusammen alle Attribte von Drinkers determinieren.
  - name -> addr favBeer
  - beersLiked -> manf

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

BEISPIEL: SCHLÜSSEL

- {name, beersLiked} is ein Schlüssel weil weder {name} oder {beersLiked} ein Superschlüssel sind.
  - name !-> manf; beersLiked !-> addr.
- Es gibt keine anderen Schlüssel, aber viele mögliche Superschlüssel, diese wären:
  - Jeder Obermenge von {name, beersLiked}.

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

WOHER KOMMEN DIE SCHLÜSSEL?

1. Init K=Alle Attribute von A
2. Behaupten sie, dass *K ein Schlüssel ist*.
3. Überprüfen Sie ob, *K -> A* für alle Attribute in *A wahr ist.*
   - *Wenn K wahr ist, überprüfen Sie ob K ein Superschlüssel ist. Dazu erzeugen Sie alle möglichen Teilmengen und führen Schritt 2-3 wieder aus. Wenn K kein Superschlüssel ist, dann haben Sie einen möglichen Schlüssel gefunden. Wenn K ein Superschlüsse ist, überprüfen Sie die Teilmengen von K mit den Schritten 2-3*
   - *Wenn nicht, ist K ein Schlüssel.*

BEISPIEL EINES SCHLECHTEN DESIGNS

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

Drinkers(name, addr, beersLiked, manf, favBeer)

name	addr	beersLiked	manf	favBeer
Janeway	Voyager	Bud	A.B.	WickedAle
Janeway	???	WickedAle	Pete's	???
Spock	Enterprise	Bud	???	Bud

Daten sind redundant, jedes ??? kann aus den FDs name -> addr favBeer und beersLiked -> manf hergeleitet werden.

•Update anomaly: wenn Janeway zur *Intrepid transferiert wird*, warden wir wirklich alle Tupel updaten?

• Deletion anomaly: Wenn niemand Bud mehr mag, verlieren wir alle Informationen darüber.

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

BOYCE-CODD NORMAL FORM

- Eine Relation R ist in *BCNF* wenn immer *X ->Y* ist immer eine nicht triviale Funktionale Abhängigkeit welche in R gilt und *X* ein Superschlüssel ist.
  - *Nicht trivial* bedeudet, Y ist keine Teilmenge von X
  - Ein Superschlüssel ist eine Obermenge von einem Schlüssel

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

BEISPIEL

Drinkers(name, addr, beersLiked, manf, favBeer)

FD's: name->addr favBeer, beersLiked->manf

- Schlüssel ist {name, beersLiked}.
- In jeder FD ist die linke Seite kein Superschlüssel.
- Damit verletzt jede der FDs die BCNF Bedingungen und damit ist *Drinkers* nicht in BCNF

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

ANDERES BEISPIEL

Beers(name, manf, manfAddr)

FD's: name->manf, manf->manfAddr

- Schlüssel ist {name} .
- name->manf verletzt nicht BCNF, aber manf->manfAddr tut es.
- Damit ist Beers auch nicht in BCNF

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

BEISPIEL -- DRINKERS

- Die Aufsplittung der Relation *Drinkers in*:
  1. Drinkers1(name, addr, favBeer)
  2. Drinkers3(beersLiked, manf)
  3. Drinkers4(name, beersLiked)

  Ist in BCNF
- Anmerkung:
  - *Drinkers1* beinhaltet alle Informationen über die Drinker
  - *Drinkers3* beinhaltet alle Informationen über das Bier
  - *Drinkers4* beinhaltet die Beziehung zwischen Drinker und Bier

- Die Aufsplittung erfolgt duch einen Algorithmus, welcher nicht in dieser Vorlesung behandelt wird.

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

NORMALFORMEN

Regel:

- *Eine Relation befindet sich in erster Normalform (1NF), wenn sie ausschließlich **atomare Attribute** enthält.*
- *Eine Relation ist in der zweiten Normalform (2NF), wenn die **1NF** vorliegt und wenn jedes Nichtschlüsselattribut von einem Schlüssel **voll funktional abhängig** ist.*
- *Eine Relation befindet sich in dritter Normalform (3NF), wenn die **2NF** vorliegt und **kein Nichtschlüsselattribut transitiv von einem Kandidatenschlüssel abhängt**.*

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

3NF

- *Ein Attribut X ist ein Primeattribut, wenn es ein Teil von einem Schlüssel ist.*
- *X ->A* verletzt 3NF nur wenn X nicht ein Superschlüssel ist und A kein Primeattribut.

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

BEISPIEL: 3NF UND BCNF

- Gegeben folgende Funktionalen Abhängigkeiten der Relation R(A, B, C):

  AB ->C und *C ->B*

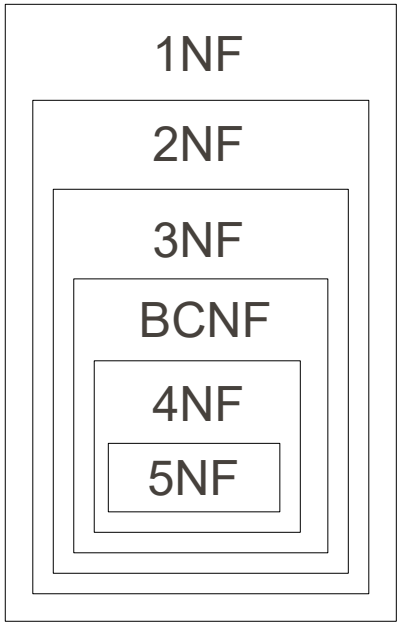
- Damit haben wir die folgenden möglichen Schlüssel *AB* und *AC*.

- Damit sind *A*, *B*, und *C* Primeattribute
- Auch wenn *C ->B* BCNF verletzt, es verletzt nicht 3NF.

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

WEITERE NORMALFORMEN

- Zusätzliche Verfeinerungen
  - Vierte Normalform (4NF)
  - Fünfte Normalform (5NF)
- Abhängigkeitserhaltend bis 3NF
- Verlustlos in allen NF
- Normalformen bauen aufeinander auf



Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

ZUSAMMENFASSUNG

- Normalisierung bewirkt eine Reduzierung der Redundanzen
- In der Praxis meist BCNF und 3NF genutzt
- Ein gutes ER-Modell ergibt ein Schema in 3NF bzw. BCNF

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

LITERATUR

- Vossen, Gottfried: Datenmodelle, Datenbank-sprachen und Datenbankmanagementsysteme, 5. Auflage, Oldenburg Wissenschaftsverlag, 2008
- Thomas Kudraß: Taschenbuch Datenbanken, Hanser, 2007