Die schwäbische Hausfrau

Kleine persönliche FiBu

Übersicht

1. Einführung
2. ER-Modell
3. Architektur
4. Regeln, Trigger und DB-Admin-Script
5. Transaktionen und Multi-User

Einführung

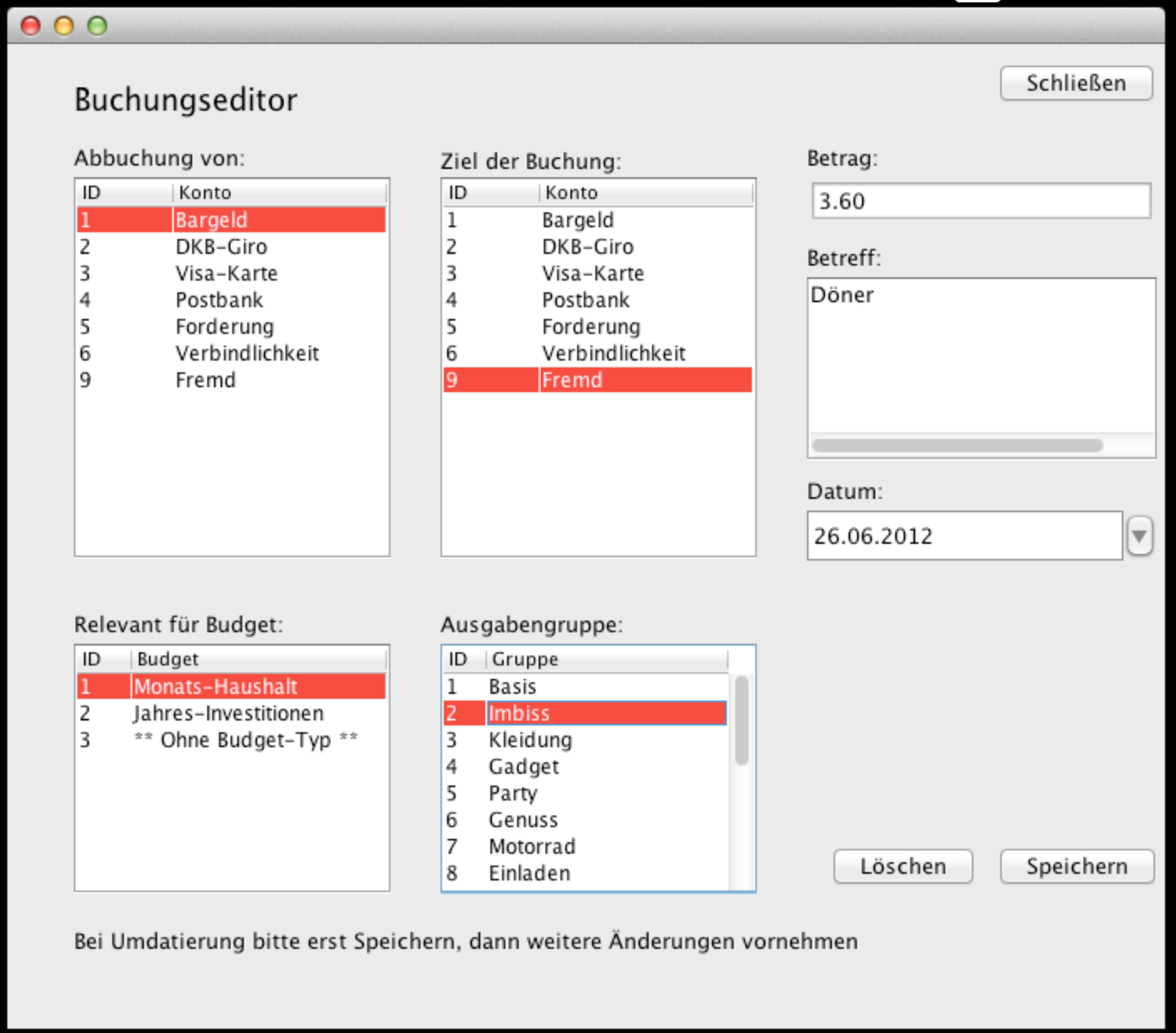
Übersicht

- Verwaltung der privaten Finanzen
- Ausgaben-Gruppen festlegen um eine Übersicht zu gewinnen, wo das Geld hinfließt
- Frei bestimmbare Budgets um die Ausgaben zu steuern

Dashboard


Die Schwäbische Hausfrau
Ende
Konten:
managen
ID Konto Saldo
1 Bargeld -30.49
2 DKB-Giro 173.17
3 Visa-Karte -341.75
4 Postbank 2065.00
5 Forderung 1200.00
6 Verbindlichkeit -120.00
9 Fremd -2945.93
Budgets:
managen
ID Budget Stand. Urspr.
3 ** Ohne Budg... 3663.67 0.00
6 Schoko-Budge... 548.00 700.00
Ausgabengruppen:
managen
ID Gruppe Stand.
1 Basis 187.89
2 Imbiss 0.00
3 Kleidung 0.00
4 Gadget 17.50
5 Party 3.60
6 Genuss 0.00
7 Motorrad 0.00
8 Einladen 0.00
9 Gesundheit 0.00
10 Wohnen 0.00
11 Studium 382.75
13 Infrastruktur 250.00
14 ** Einkommen ** -4027.67
16 Reisen 0.00
17 Verpflichtungen 0.00
18 Schwund 0.00
19 ** Transfer-Buchung ** -100.00
Gesamtsaldo:
Gesamt: 2945.93 €
Neue Buchung
Übersicht Buchungen
Budgeting
Aktualisieren

Einzelne Buchungen



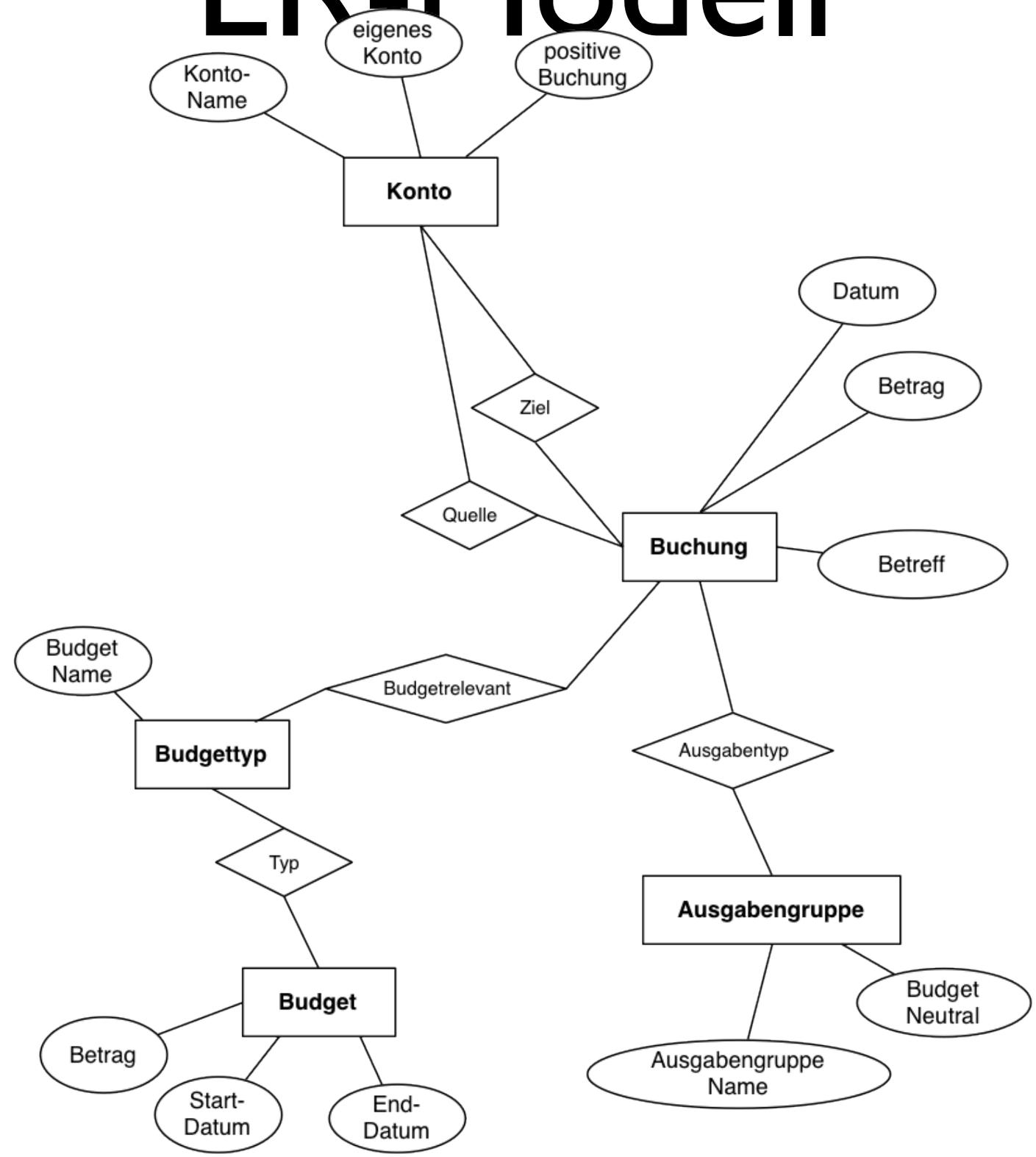
Übersicht Buchungen

ID	Datum	Betrag	Betreff	von	auf	Budget	Ausgabengrp.
6	2012-05-01	-120.00	Inventur (Initialisierung) Rasenmäher...	Verbindlichkeit	Fremd	** Ohne Budget-...	** Einkommen **
5	2012-05-01	1200.00	Inventur (Initialisierung) Felix: 1000,...	Forderung	Fremd	** Ohne Budget-...	** Einkommen **
3	2012-05-01	-134.00	Inventur (Initialisierung)	Visa-Karte	Fremd	** Ohne Budget-...	** Einkommen **
4	2012-05-01	2340.00	Inventur (Initialisierung)	Postbank	Fremd	** Ohne Budget-...	** Einkommen **
2	2012-05-01	423.17	Inventur (Initialisierung)	DKB-Giro	Fremd	** Ohne Budget-...	** Einkommen **
1	2012-05-01	78.50	Inventur (Initialisierung)	Bargeld	Fremd	** Ohne Budget-...	** Einkommen **
12	2012-06-01	250.00	Externe Festplatte	Fremd	DKB-Giro	Jahres-Investitio...	Infrastruktur
32	2012-06-01	35.89	Aldi-Einkauf	Fremd	Bargeld	Jahres-Investitio...	Basis
9	2012-06-19	18.75	Amazon: Buch Java Swing	Fremd	Visa-K...	Monats-Haushalt	Studium
10	2012-06-26	100.00	Bargeld abgehoben	Bargeld	Visa-K...	Jahres-Investitio...	** Transfer-Buc...
13	2012-06-26	17.50	Notizbuch in Leder	Fremd	Bargeld	Monats-Haushalt	Gadget
7	2012-06-26	3.60	Döner 42	Fremd	Bargeld	Jahres-Investitio...	Party
17	2013-01-03	275.00	Studiengebühren Sommersemester 2...	Fremd	Postbank	** Ohne Budget-...	Studium
18	2013-01-03	89.00	Bestellung JavaWorld-Abo	Fremd	Visa-K...	** Ohne Budget-...	Studium
29	2013-01-14	150.00	Milka-Festival in Süddeutschland	Fremd	Bargeld	Schoko-Budget...	Basis
30	2013-01-14	2.00	Nippon	Fremd	Bargeld	Schoko-Budget...	Basis

Edit | Löschen | Update | Schließen

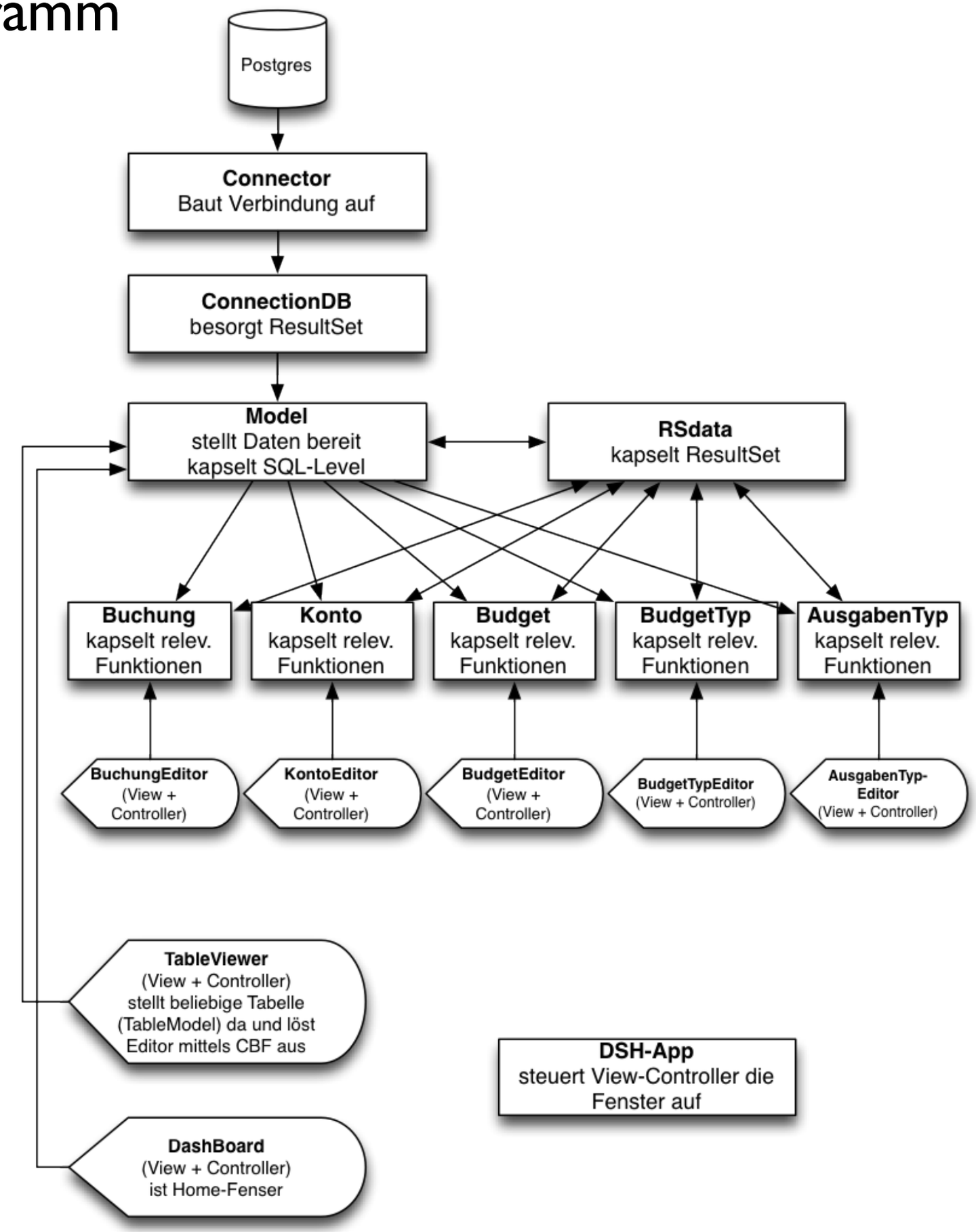
ER-Modell

ER-Modell

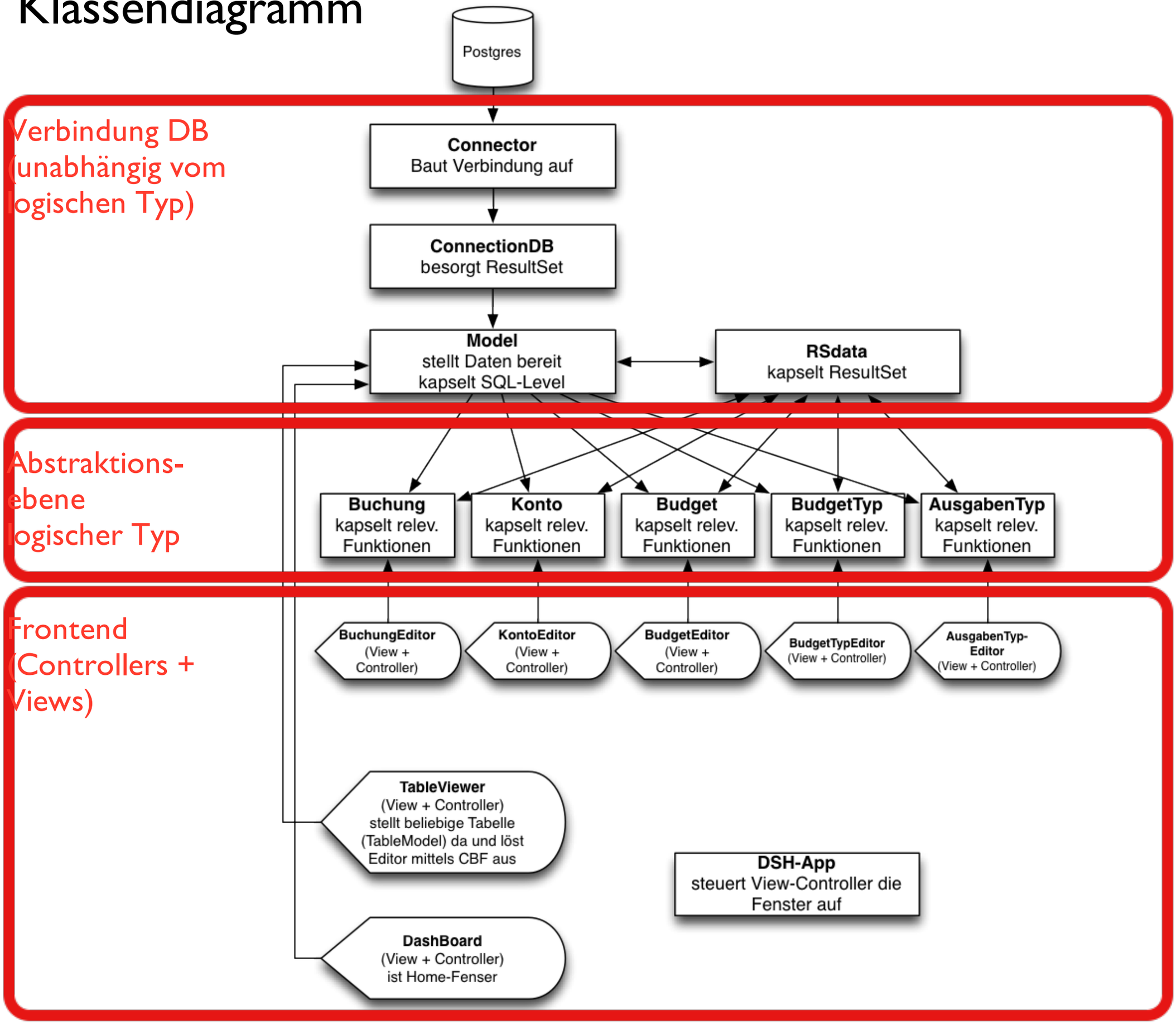

Konto-
Name
eigenes
Konto
positive
Buchung
Konto
Datum
Betrag
Ziel
Quelle
Buchung
Betreff
Budget
Name
Budgetrelevant
Budgettyp
Ausgabentyp
Typ
Ausgabengruppe
Budget
Neutral
Budget
Betrag
Start-
Datum
End-
Datum
Ausgabengruppe
Name

Architektur

Klassendiagramm



Klassendiagramm



Technik

- Technologie Stack
  - Java Swing
  - Java 6
  - JDBC 4
  - Postgres 9

Regeln, Trigger und DB-Admin-Script

Vorzeichenbestimmung bei Saldierung

```
-- ============== Ermitteln ob absolut gesehen positiver oder negativer Buchungswert
create or replace function absoluteSigning (integer, boolean, numeric (8,2))
-- Parameter: Konto_id, Konto war Ziel? (Ziel=true, Quelle=false), Betrag
returns numeric (8,2)
as $$
declare
newSignedMoney numeric (8,2):=$3;
positivKonto boolean:='false';
begin
Select konto.positiv into positivKonto from Konto where konto.konto_id=$1;
if (not positivKonto) then
    newSignedMoney:='0'-newSignedMoney; -- Wenn nicht positiv Konto Vorzeichen umdrehen
end if;
if (not $2) then
    newSignedMoney:='0'-newSignedMoney; -- Wenn nicht Ziel dann Vorzeichen umdrehen
end if;

return newSignedMoney;
end;
$$
LANGUAGE 'plpgsql';
```

Budget-Ermittlung

```
select
    (aktiv_budget.bd_amount+sum(absoluteSigning(buchung.ziel,'true',buchung.betrag)))as sum
into
    budget_rest
from
    buchung
join -- Budget-Tabellen zusammenführen + zeitlich relevantes Budget ausgesucht
    (
        select
            budget_typ.budget_typ_id as bd_typ_id,
            budget_typ.budget_typ_name as bd_typ_name,
            budget.budget_betrag as bd_amount,
            budget.start_date as start,
            budget.end_date as end
        from
            budget,
            budget_typ
        where
            budget.typ=budget_typ.budget_typ_id -- joint budget-tabellen
            and budget.start_date <=$2 -- Budget im Zeitrahmen?
            and budget.end_date >=$2 -- Budget im Zeitrahmen?
            and budget_typ.budget_typ_id=$1 -- nur gesuchtes Budget
     ) as aktiv_budget
    on (buchung.budget_relevant=aktiv_budget.bd_typ_id)  -- joint: nur aktive Budgets
where
    buchung.datum >=aktiv_budget.start    -- Buchung im Zeitrahmen?
    and buchung.datum <=aktiv_budget.end    -- Buchung im Zeitrahmen?
Group by
    aktiv_budget.bd_typ_id,
    aktiv_budget.bd_typ_name,
    aktiv_budget.bd_amount;
```

Trigger-Function: Block Delete

```
-- ================ Trigger-Function: Stop delete of non-deletable rows
CREATE OR REPLACE FUNCTION stop_delete_row () RETURNS trigger

AS $$
DECLARE
BEGIN
    if (old.deletable=false) then
        raise exception 'This row cannot be deleted';
    end if;
return old;
END; $$
language 'plpgsql';
```

2x Trigger: Block Delete (Tables: budget, ausgabengruppe)

```
-- ================ Trigger: Stop delete of nobudget-row
create trigger trigger_stop_delete_nobudget_row
before
    delete
on
    budget
for each row execute procedure
    stop_delete_row();

-- ================ Trigger: Stop delete of sonstiges-ausgabengruppe
create trigger trigger_stop_delete_sonstiges_row
before
    delete
on
    ausgabengruppe
for each row execute procedure
    stop_delete_row();
```

DB-Admin-Script: Doubletten I

```
BEGIN
create temp table doubles_removed (id integer);
for first_pick in (
    select
        *
    from
        buchung
    )
Loop
    -- raise notice 'first_pick: %', first_pick;
    for doubles_found in (
        select
            *
        from
            buchung
        where
            datum=first_pick.datum and
            betrag=first_pick.betrag and
            betreff=first_pick.betreff and
            ziel=first_pick.ziel and
            quelle=first_pick.quelle and
            budget_relevant=first_pick.budget_relevant and
            ausgaben_typ=first_pick.ausgaben_typ and
            COALESCE(repeat,'0')=COALESCE(first_pick.repeat,'0') and
            buchung_id!=first_pick.buchung_id
        )
    Loop
```

DB-Admin-Script: Doubletten 2 entfernen

```
Loop
    select count (id) from doubles_removed
        where doubles_removed.id=first_pick.buchung_id
        into doubles_counter;
    if (doubles_counter=0) then
        raise notice 'doubles_found: %', doubles_found;
        insert into backup_doubles_buchung (buchung_id, datum, betrag, betreff, ziel,
            values (
                doubles_found.buchung_id,
                doubles_found.datum,
                doubles_found.betrag,
                doubles_found.betreff,
                doubles_found.ziel,
                doubles_found.quelle,
                doubles_found.budget_relevant,
                doubles_found.ausgaben_typ,
                doubles_found.repeat
            );
        delete from buchung where buchung_id=doubles_found.buchung_id;
        insert into doubles_removed (id) values (doubles_found.buchung_id);
    end if;
end loop;
```

Transaktionen und Multi-User

Transaktion

- Werden Daten zum Editieren in den Client geladen, dann werden die entsprechenden Rows in der DB geblockt (autocommit deaktivieren)

- Mit dem Speichern wird dann ein commit () ausgeführt

```
if (blockDB) {
    this.getConnection().setAutoCommit(false);
}
```

Multi-User

- Die Transaktion verhindert, dass zwei User die selbe Zeile editieren.
- Es verhindert ab keine indirekten Inkonsistenzen
  (es können z.B. referenzierte Daten verändert werden)

Danke!