1 - TINA UMLANDT. 2015

WIEDERHOLUNG

INSERT

```
INSERT INTO PERSONAL VALUES(167,' Krause','Gustav','it3',' d12',' dak');

INSERT INTO PERSONAL (PNR, NAME, VORNAME) VALUES (777, ‘Graf’,’Gerd’);

INSERT INTO tbl_name (a,b,c) VALUES(1,2,3),(4,5,6),(7,8,9);
```

UPDATE

```
UPDATE PERSONAL
SET KRANKENKASSE='TKK', GEH_STUFE='it5'
WHERE NAME='Graf';


UPDATE GEHALT
SET BETRAG=BETRAG * 1.07;
```

DELETE

```
DELETE FROM PERSONAL
WHERE PNR = 167;
```

DROP TABLE

```
CREATE TABLE KAFFEE_MASCHINE (
   ID INT PRIMARY KEY,
   NAME VARCHAR(20)
);

DROP TABLE KAFFEE_MASCHINE;
```

SEQUENCES

```
CREATE SEQUENCE maschine_seq START WITH 30;

INSERT INTO MASCHINE VALUES (maschine_seq.nextval, 'Diamantbohrer', 133, '01-nov-15', 34000, 34000);
```

ALTER TABLE

```
ALTER TABLE PERSONAL ADD V_NR number(4);

ALTER TABLE PERSONAL MODIFY (KRANKENKASSE char(5));
```

AUFGABEN

1. ANDERN SIE DIE TABELLE ABTEILUNG SO, DASS DER NEUE TITEL FUER "FORSCHUNG/ BILDUNG" MIT "FORSCHUNG UND ALLGEMEINE WEITERBILDUNG" EINGETRAGEN WERDEN KANN.
2. FÜGEN SIE DEN TABELLEN `Kind` UND `Praemie` EINEN KÜNSTLICHEN SCHLÜSSEL HINZU UND AKTUALISIEREN SIE DAZU DIE TABELLENEINTRÄGE.
3. ERSTELLEN SIE FÜR ALLE VORHANDENEN TABELLEN EINE SEQUENCE, DIE ZUR ERSTELLUNG IHRER PRIMÄRSCHLÜSSEL BENUTZT WERDEN KANN.
4. FÜGEN SIE NUN FÜR JEDE TABELLE, EINEN NEUEN DATENSATZ EIN, INDEM SIE IN DEM `INSERT` - STATEMENT DIE SEQUENCE BENUTZEN.

1. ÄNDERN SIE DIE TABELLE ABTEILUNG SO, DASS DER NEUE TITEL FUER "FORSCHUNG/ BILDUNG" MIT "FORSCHUNG UND ALLGEMEINE WEITERBILDUNG" EINGETRAGEN WERDEN KANN.

1. ANDERN SIE DIE TABELLE ABTEILUNG SO, DASS DER NEUE TITEL FUER "FORSCHUNG/ BILDUNG" MIT "FORSCHUNG UND ALLGEMEINE WEITERBILDUNG" EINGETRAGEN WERDEN KANN.

```
ALTER TABLE Abteilung MODIFY NAME VARCHAR(50);

UPDATE Abteilung SET NAME='Forschung und allgemeine Weiterbildung' WHERE NAME='Forschung/ Bildung';
```

2. FÜGEN SIE DEN TABELLEN `Kind` UND `Praemie` EINEN KÜNSTLICHEN SCHLÜSSEL HINZU UND AKTUALISIEREN SIE DIE SCHON BESTEHENDEN TABELLENEINTRÄGE.

2. FÜGEN SIE DEN TABELLEN `Kind` UND `Praemie` EINEN KÜNSTLICHEN SCHLÜSSEL HINZU UND AKTUALISIEREN SIE DIE SCHON BESTEHENDEN TABELLENEINTRÄGE.

```
ALTER TABLE KIND ADD KIND_ID INT;

CREATE SEQUENCE KIND_SEQ;

UPDATE KIND SET KIND_ID=KIND_SEQ.NEXTVAL;

ALTER TABLE KIND DROP CONSTRAINT SYS_C006996;

ALTER TABLE KIND ADD CONSTRAINT KIND_PK PRIMARY KEY(KIND_ID);
```

```
SQL> SELECT constraint_name, constraint_type, column_name
FROM user_constraints NATURAL JOIN user_cons_columns
WHERE table_name = 'KIND';

CONSTRAINT_NAME         C     COLUMN_NAME
----------------------- ------------------
SYS_C006996             P     PNR
SYS_C006996             P     K_NAME
SYS_C006996             P     K_VORN
SYS_C006997             R     PNR
```

```
SQL> SELECT constraint_name, constraint_type, column_name
FROM user_constraints NATURAL JOIN user_cons_columns
WHERE table_name = 'KIND';

CONSTRAINT_NAME         C     COLUMN_NAME
----------------------- ------------------
KIND_PK                 P     KIND_ID
SYS_C006997             R     PNR
```

3. Erstellen Sie für alle vorhandenen Tabellen eine Sequence, die zur Erstellung ihrer Primärschlüssel benutzt werden kann.

3 . ERSTELLEN SIE FUR ALLE VORHANDENEN TABELLEN EINE SEQUENCE. DIE ZUR ERSTELLUNG IHRER PRIMÄRSCHLÜSSEL BENUTZT WERDEN KANN.

```
CREATE SEQUENCE PERSONAL_SEQ START WITH 200;
CREATE SEQUENCE MASCHINE_SEQ START WITH 30;
CREATE SEQUENCE KIND_SEQ START WITH 10;
CREATE SEQUENCE PRAEMIE_SEQ START WITH 15;
```

4. Fügen Sie nun für jede Tabelle, einen neuen Datensatz ein, indem Sie in dem `INSERT` - Statement die Sequence benutzen.

4. FÜGEN SIE NUN FÜR JEDE TABELLE, EINEN NEUEN DATENSATZ EIN, INDEM SIE IN DEM INSERT - STATEMENT DIE SEQUENCE BENUTZEN.

```
INSERT INTO PERSONAL VALUES
    (PERSONAL_SEQ.NEXTVAL, 'Adele', 'Goldberg', 'it5', 'abt11', 'bek');
INSERT INTO MASCHINE VALUES
    (MASCHINE_SEQ.NEXTVAL, 'Dampfmaschine', 123, '01-dec-15', 120000, 90000);
INSERT INTO KIND VALUES
    (161, 'Fowler', 'Marissa', '21-nov-15', KIND_SEQ.NEXTVAL);
INSERT INTO PRAEMIE VALUES
    (173, 399, '22-jan-15', PRAEMIE_SEQ.NEXTVAL);
```

TABELLENBESCHREIBUNG

```
SQL> DESCRIBE PERSONAL;

 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------
 PNR                                       NOT NULL NUMBER(38)
 VORNAME                                            VARCHAR2(20)
 NACHNAME                                           VARCHAR2(20)
 GEH_STUFE                                          VARCHAR2(5)
 ABT_NR                                             VARCHAR2(5)
 KRANKENKASSE                                       VARCHAR2(3)
```

AUFGABEN, TEIL II

1. LEGEN SIE EINE TABELLE AN, IN DER SIE MATERIALIEN FUR DIE MASCHINEN VERWALTEN. FOLGENDE DATEN MUSSEN GESPEICHERT WERDEN KÖNNEN: NAME, ZUGEHÖRIGE MASCHINE, MENGE, LETZTES EINKAUFSDATUM
2. FÜGEN SIE DREI MATERIALIEN EIN
3. GEBEN SIE ALLE MASCHINENNAMEN AUS, FÜR DIE ES MATERIAL GIBT.
4. GEBEN SIE ALLE MITARBEITER AUS, DIE NICHT AN MASCHINEN ARBEITEN.
5. ÜBERLEGEN SIE SICH, WIE UND WO SIE SPEICHERN KÖNNEN, WIEVIEL MATERIAL WANN FÜR WELCHE MASCHINE BENUTZT WURDE. LEGEN SIE ENTSPRECHENDE TABELLEN AN ODER ÄNDERN SIE BESTEHENDE. FÜGEN SIE DAZU FÜNF DATENSÄTZE EIN.
6. GEBEN SIE EINE ÜBERSICHT AUS, WIEVIEL DIE EINZELNEN MASCHINEN AN MATERIAL VERBRAUCHT HABEN.
7. WIE MUSS DIE TABELLE IN 1) GEÄNDERT WERDEN, WENN EIN MATERIAL FÜR VERSCHIEDENE MASCHINEN BENUTZT WERDEN KANN? FORMULIEREN SIE DAS DAZUGEHÖRIGE SQL.

SCHREIBEN SIE FUR ALLE PUNKTE DAS ENTSPRECHENDE SQL. WENN NOTIG, BENUTZEN SIE SEQUENCES ANSTATT SICH PRIMÄRSCHLÜSSEL SELBER AUSZUDENKEN.

LOS GEHT'S

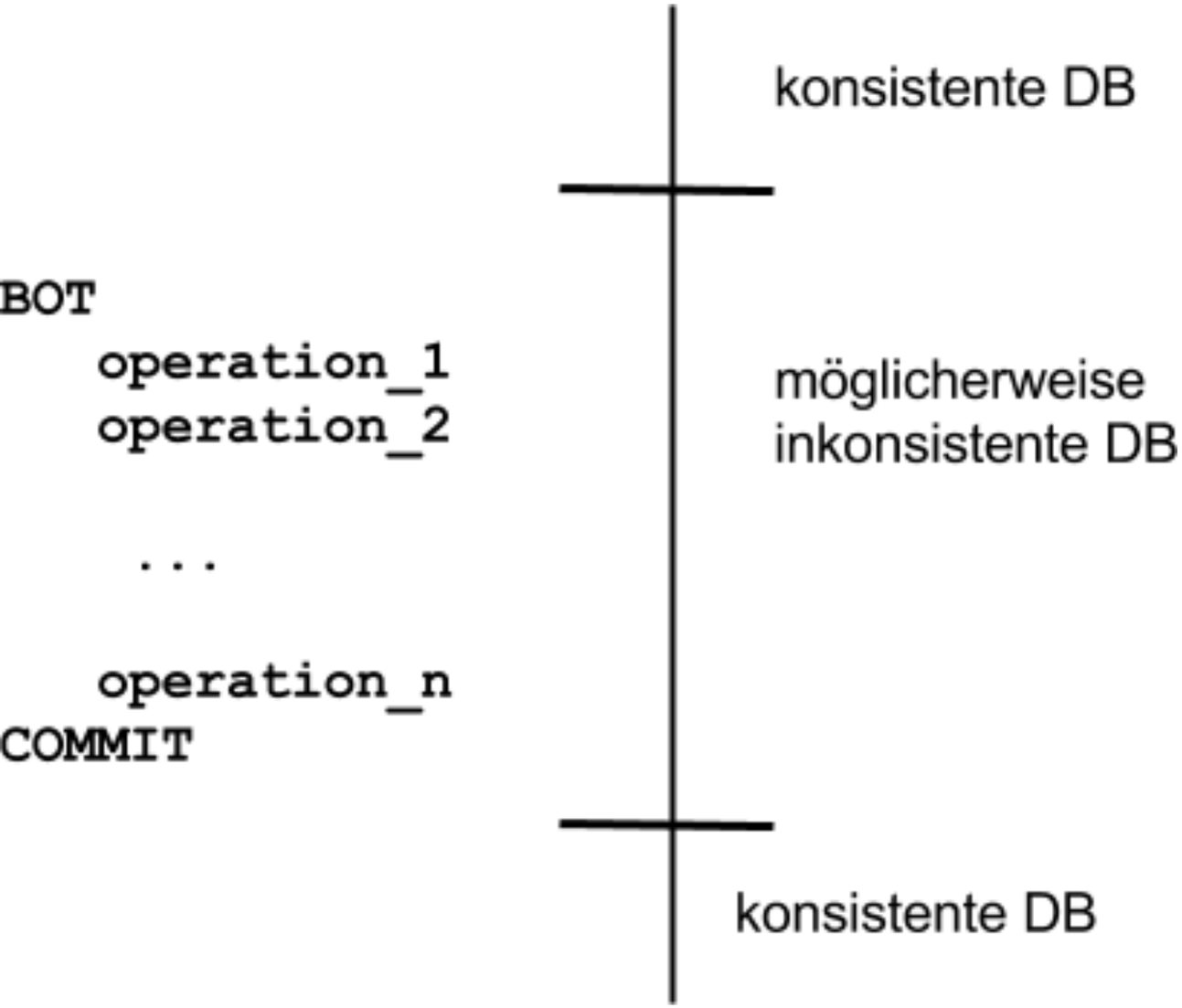
\o/

TRANSAKTION

23 - TINA UMLANDT, 2015

EINE FOLGE VON DATENBANKOPERATIONEN, DIE INSBESONDERE BEZÜGLICH DER DATENBANKKONSISTENZ ALS LOGISCHE EINHEIT (ATOMAR) ANGESEHEN WIRD.

-- TRANSAKTION



ABLAUF EINER TRANSAKTION

BEISPIEL:

- REISEBUCHUNG: HOTELZIMMER UND FLUGBUCHUNG
  - NEUE MASCHINE UND NEUER MITARBEITER

WARUM?

- DATENBANK MUSS VOR UND NACH EINER TRANSAKTION IM ZULÄSSIGEN ZUSTAND SEIN!
- WEITERHIN UNTERSTÜTZT EIN DBS HÄUFIG VIELE BENUTZER, DIE DANN VIELE TRANSAKTIONEN GLEICHZEITIG STARTEN
- PARALLELE AUSFÜHRUNG VON TRANSAKTIONEN OHNE „MERKWÜRDIGE" EFFEKTE
- FEHLER IM SYSTEM (STROMAUSFALL, PLATTENFEHLER, ABSTURZ DER SOFTWARE)

ACID

28 - TINA UMLANDT, 2015

A WIE ATOMICITY

TRANSAKTIONEN WERDEN GANZ ODER GAR NICHT AUSGEFUHRT

```
BOT
  Buche das Hotelzimmer
  Buche den Flug
COMMIT

BOT
  Buche das Hotelzimmer
  IF Hotelzimmer nicht moeglich THEN ROLLBACK
  Buche den Flug
COMMIT
```

SQL UND TRANSAKTIONEN

- `ROLLBACK` ZUM ABBRECHEN DER TRANSAKTION
- `COMMIT` ZUM ERFOLGREICHEN BEENDEN DER TRANSAKTION

C WIE CONSISTENCY

TRANSAKTIONEN UBERFUHREN DIE DATENBANK VON EINEM KONSISTENTEN ZUSTAND IN EINEN ANDEREN

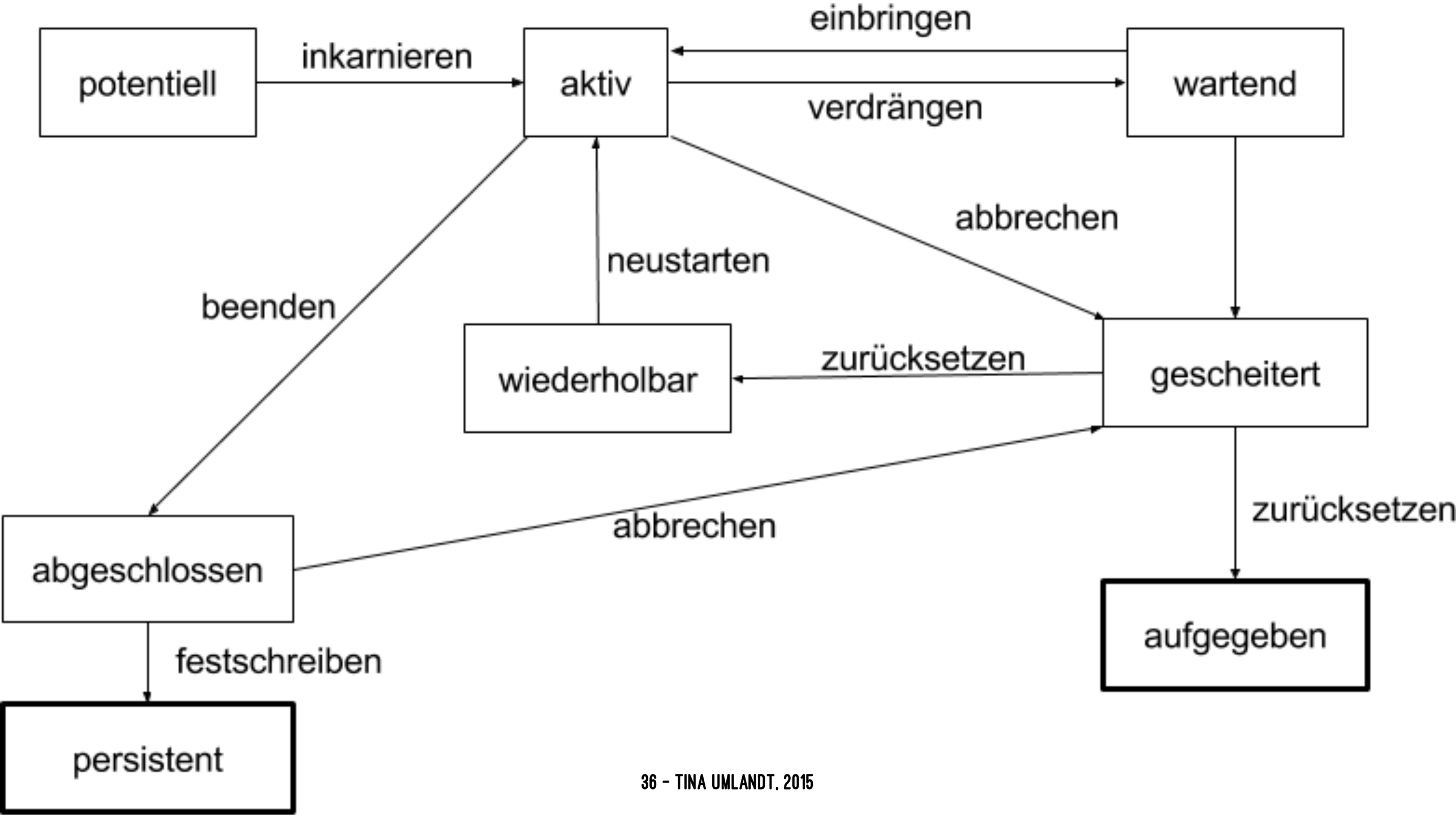
WIE ISOLATION

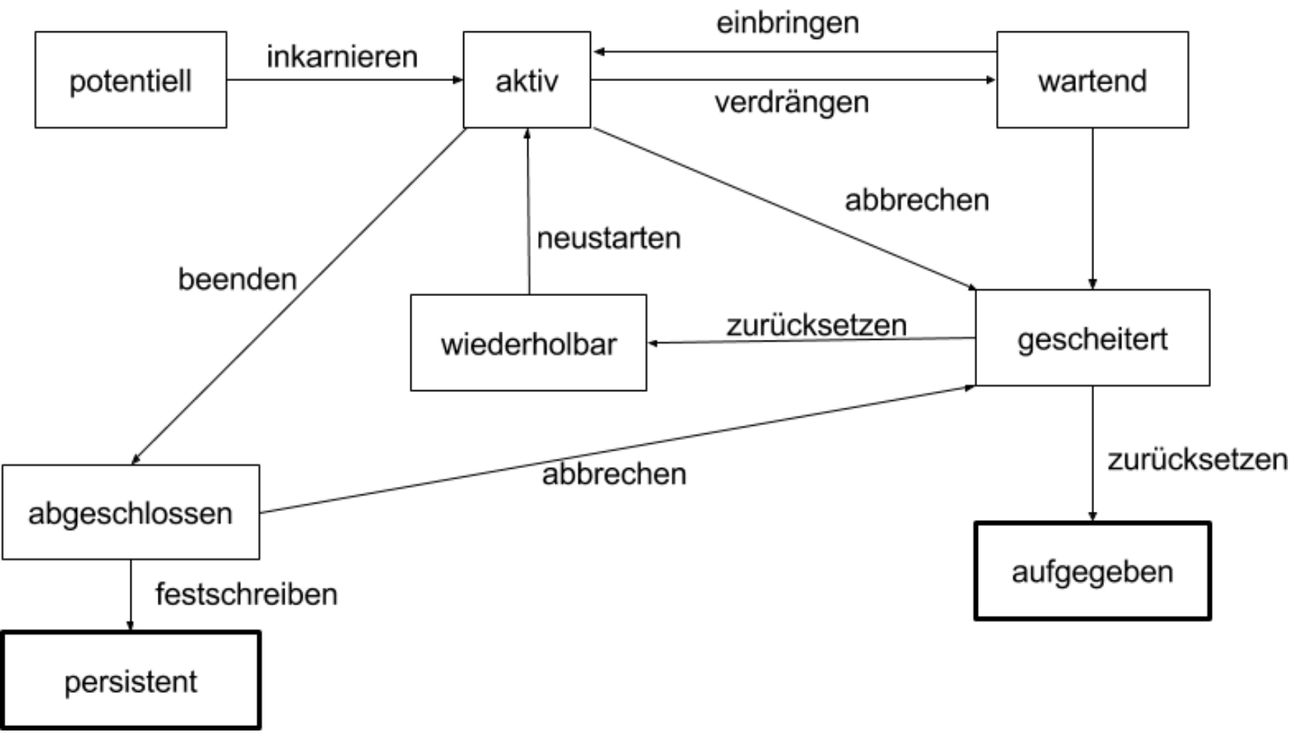
TRANSAKTIONEN WERDEN SO AUSFUHRT, ALS WENN SIE ALLEINE AUF DER DATENBANK OPERIEREN WÜRDEN

D WIE DURABILITY

TRANSAKTIONEN SIND NACH ERFOLGREICHEM ABSCHLUSS DAUERHAFT UND GEHEN AUCH DURCH FEHLER NICHT MEHR VERLOREN

ZUSTANDSÜBERGÄNGE EINER TRANSAKTION

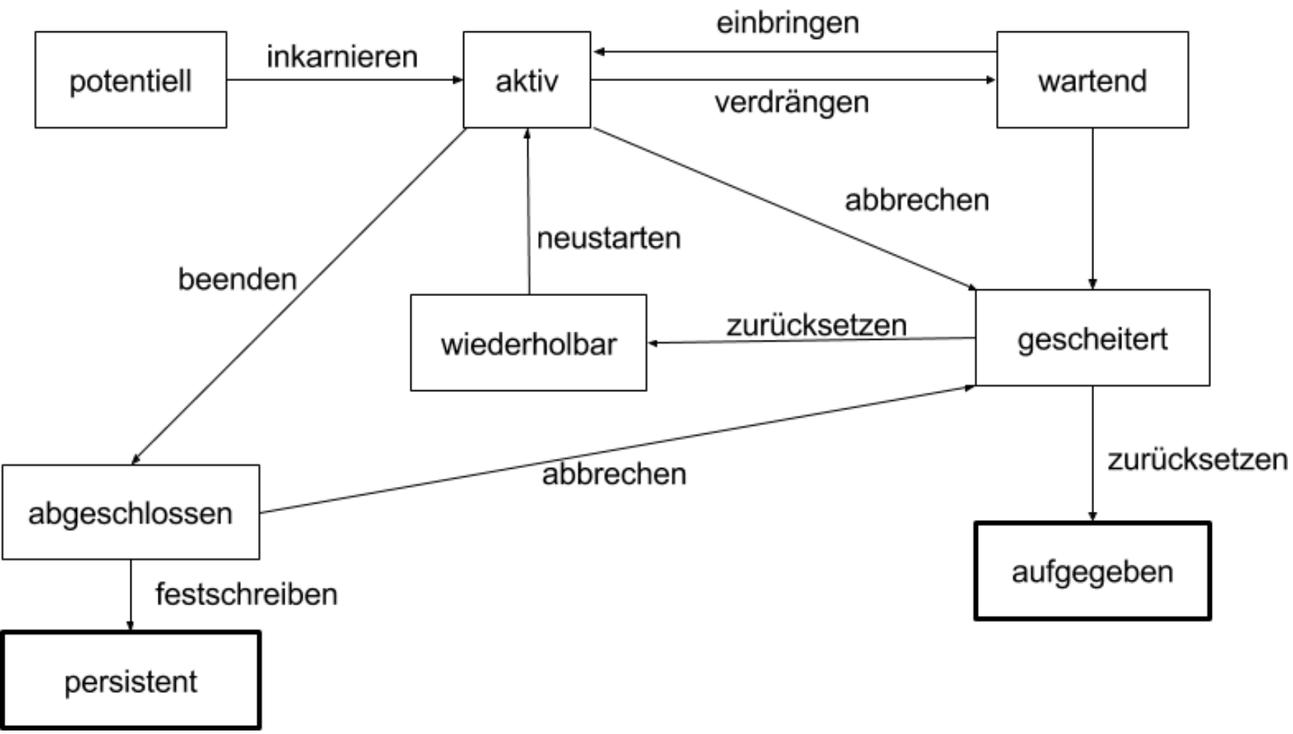

potentiell
inkarnieren
aktiv
einbringen
wartend
verdrängen
abbrechen
neustarten
beenden
wiederholbar
zurücksetzen
gescheitert
zurücksetzen
abbrechen
abgeschlossen
aufgegeben
festschreiben
persistent



POTENTIELL: TA IST CODIERT UND WARTET DARAUF AKTIV ZU WERDEN.

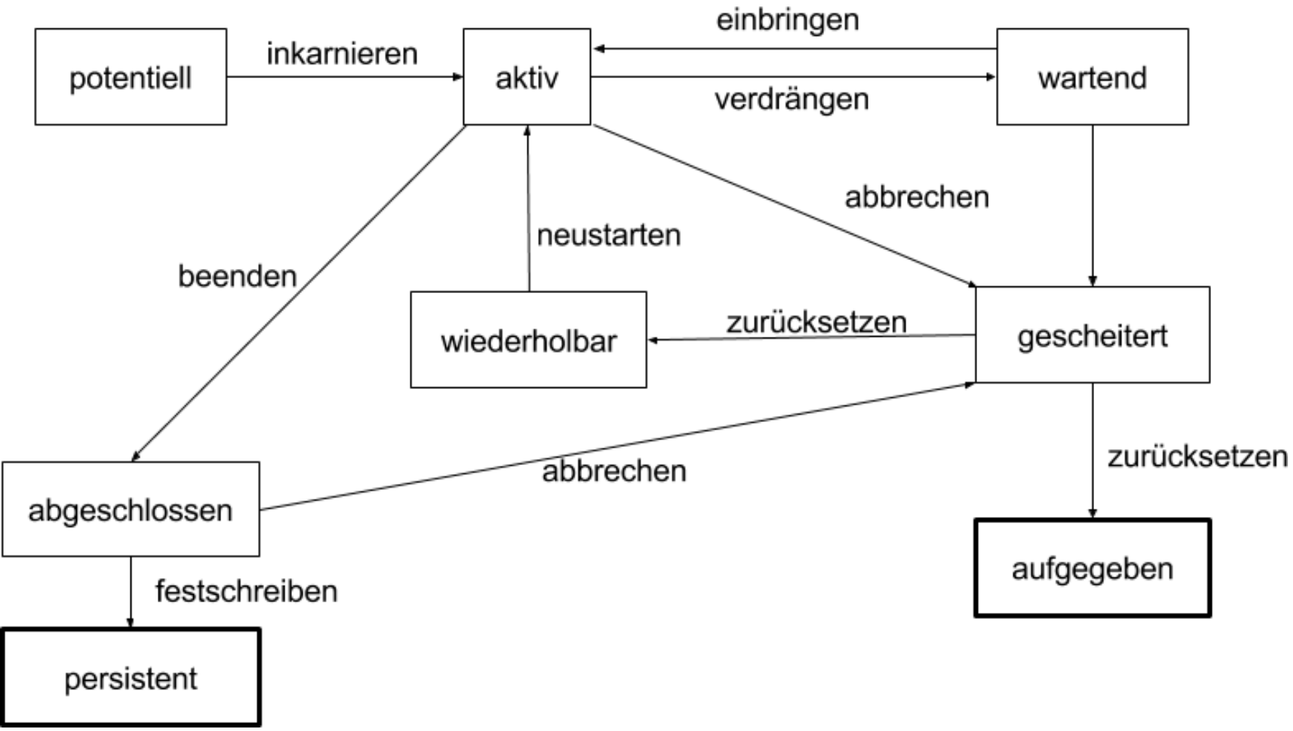
AKTIV: DIE AKTIVEN TAS KONKURRIEREN UM BETRIEBSMITTEL.

WARTEND: BEI ÜBERLAST KÖNNEN AKTIVE TAS VERDRÄNGT WERDEN. NACH DER ÜBERLAST WERDEN SIE WIEDER EINGEBRACHT.



ABGESCHLOSSEN: DURCH `commit` WIRD AKTIVE TA BEENDET. VORM SCHREIBEN MÜSSEN ABER EVENTUELLE KONSISTENZBEDINGUNGEN GEPRÜFT WERDEN.

PERSISTENT: ABGESCHLOSSENE TAS WERDEN DURCH FESTSCHREIBEN DAUERHAFT GESCHRIEBEN. TA IST PERSISTENT. DIES IST EINER VON ZWEI MÖGLICHEN ENDZUSTÄNDEN.



GESCHEITERT: TA KANN AUFGRUND VIELER EREIGNISSE SCHEITERN (BEISPIEL: NUTZER BENUTZT SELBST `abort` ODER SYSTEMFEHLER TRETEN AUF). ODER KONSISTENZVERLETZUNGEN WERDEN FESTGESTELLT.

WIEDERHOLBAR: WIRKUNG ZURÜCKSETZEN UND NOCHMAL AKTIVIEREN.

AUFGEGEBEN: GESCHEITERTE TA, DIE AUFGEGEBEN WIRD. WIRKUNG WIRD ZURÜCKGESETZT.

TRANSAKTIONSVERWALTUNG

- KONSISTENZSICHERUNG
- RECOVERY
- MEHRBENUTZERSYNCHRONISATION

RECOVERY UND FEHLERFALL

BEI TRANSAKTIONSFEHLERN GARANTIERT DAS TRANSAKTIONSKONZEPT, DASS ENTWEDER ALLE BEFEHLE DER TRANSAKTION DURCHGEFÜHRT WERDEN ODER KEINER

TRANSAKTIONSFEHLER

SYSTEMFEHLER

45 - TINA UMLANDT, 2015

MEDIENFEHLER

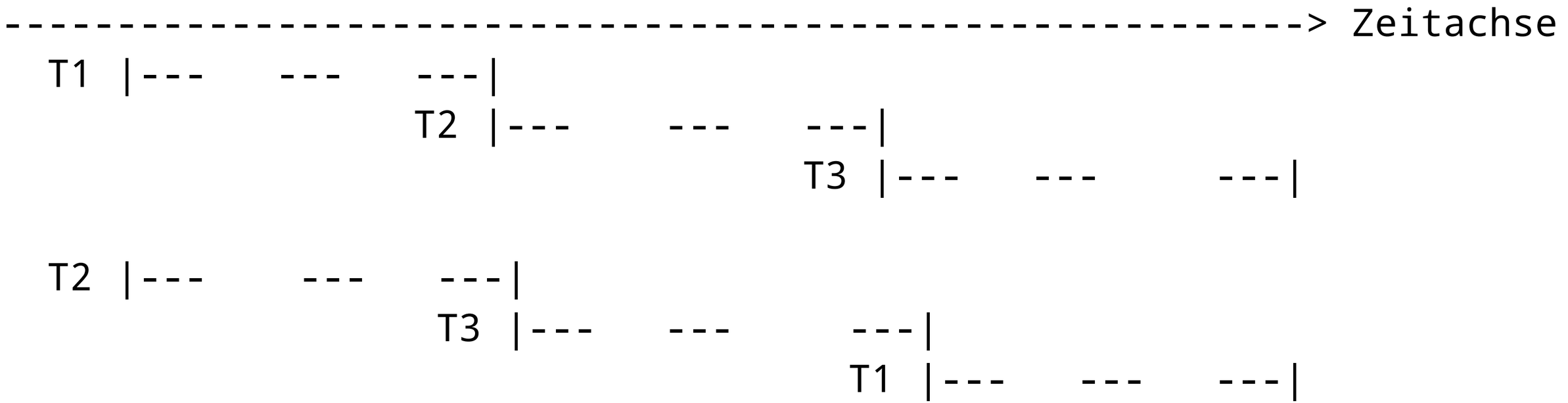
LOGBUCH ZUR SICHERUNG

SAVEPOINTS

MEHRBENUTZER-FÄHIGKEIT

SERIELLE AUSFÜHRUNG

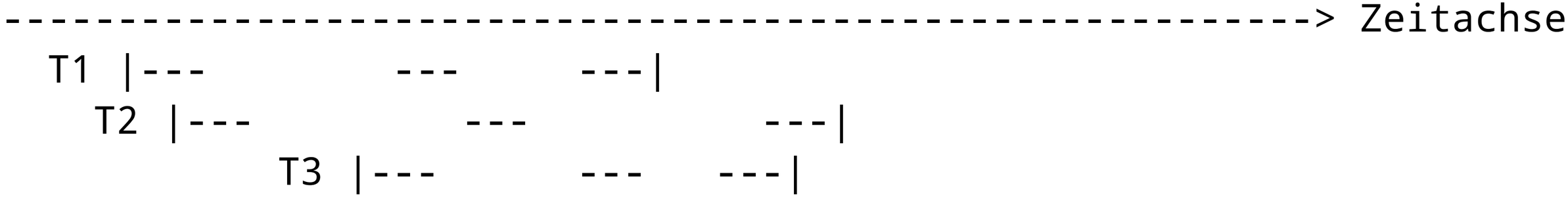
SERIELLE AUSFUHRUNG



T1, T2, T3 T1, T3, T2
T2, T3, T1 T2, T1, T3
T3, T1, T2 T3, T2, T1

PARALLELE AUSFÜHRUNG

MEHRBENUTZERBETRIEB (PARALLEL)



ANOMALIEN BEI UNKONTROLLIERTEM MEHRBENUTZERBETRIEB

54 - TINA UMLANDT, 2015

PROBLEM

LOST UPDATE

Zeit	Transaktion 1	Transaktion 2
1	Lese Betrag	-
2	-	Lese Betrag
3	Erhöhe Betrag um 1	-
4	-	Erhöhe Betrag um 1
5	Commit	-
6	-	Commit

```
Zeit    Transaktion 1          Transaktion 2
-------------------------------------------
1       Lese Betrag            -
2       -                      Lese Betrag
3       Erhöhe Betrag um 2 -                        -> geht verloren
4       -                      Erhöhe Betrag um 1
5       Commit                 -
6       -                      Commit
```

AUFGABE: BEISPIEL MIT UNSEREN TABELLEN FINDEN

PROBLEM

UNCOMMITTED DEPENDENCY (DIRTY READ)

Zeit	Transaktion 1	Transaktion 2
1	Lese Betrag	-
2	Erhöhe Betrag um 1	-
3	-	Lese Betrag
4	Rollback	-

```
Zeit    Transaktion 1          Transaktion 2
-------------------------------------------
1       Lese Betrag            -
2       Erhöhe Betrag um 1     -
3       -                      Lese Betrag    -> nicht lösbarer Konflikt
4       Rollback               -
```

AUFGABE: BEISPIEL MIT UNSEREN TABELLEN FINDEN

PROBLEM

INCONSISTENCY READ (NONREPEATABLE READ)

Zeit	Transaktion 1	Transaktion 2
1	Lese Haben_Konto	-
2	-	Ändere Soll_Konto
3	-	Ändere Haben_Konto
4	-	Commit
5	Lese Soll_Konto	-
6	Commit	-

```
Zeit    Transaktion 1       Transaktion 2
-------------------------------------------
1       Lese Haben_Konto    -
2       -                   Ändere Soll_Konto    -> Verletzung Isolation
3       -                   Ändere Haben_Konto
4       -                   Commit
5       Lese Soll_Konto     -
6       Commit              -
```

AUFGABE: BEISPIEL MIT UNSEREN TABELLEN FINDEN

PROBLEM

PHANTOMDATEN

Zeit	Transaktion 1	Transaktion 2
1	Lese Kontostand	-
2	-	Ändere Kontostand
4	-	Commit
5	Lese Kontostand	-
6	Commit	-

```
Zeit    Transaktion 1       Transaktion 2
-----------------------------------------
1       Lese Kontostand     -
2       -                   Ändere Kontostand   -> Veränderung des Datenbestandes
4       -                   Commit
5       Lese Kontostand     -
6       Commit              -
```

AUFGABE: BEISPIEL MIT UNSEREN TABELLEN FINDEN

LÖSUNG: LOCKS

READ LOCKS (SHARED)

WRITE LOCKS (EXCLUSIVE)

ZWEI-PHASEN-SPERRPROTOKOLL (2PL)

PROBLEM

LOST UPDATE

Zeit	Transaktion 1	Transaktion 2
1	Lese Betrag	-
2	-	Lese Betrag
3	Erhöhe Betrag um 1	-
4	-	Erhöhe Betrag um 1
5	Commit	-
6	-	Commit

LÖSUNG

LOST UPDATE

Zeit	Transaktion 1	Transaktion 2
1	Lese Betrag (Lese-Sperre erteilt)	-
2	-	Lese Betrag (Lese-Sperre erteilt)
3	Erhöhe Betrag um 1 (wartet auf Schreib-Sperre)	-
4	-	Erhöhe Betrag um 1 (wartet auf Schreib-Sperre)
5	wartet	wartet
6	wartet	wartet

```
Zeit    Transaktion 1                          Transaktion 2
-------------------------------------------------------------
1       Lese Betrag                            -
            (Lese-Sperre erteilt)
2       -                                      Lese Betrag
                                                   (Lese-Sperre erteilt)
3       Erhöhe Betrag um 1                    -
            (wartet auf Schreib-Sperre)
4       -                                      Erhöhe Betrag um 1
                                                   (wartet auf Schreib-Sperre)
5       wartet                                 wartet                          -> Deadlock
6       wartet                                 wartet                          -> Deadlock
```

DEADLOCK

SERIALISIERBARKEIT

VEREINIGUNG DER VORTEILE DER SERIELLEN VERARBEITUNG - ISOLATION - UND DES MEHRBENUTZERBETRIEBS - ERHÖHTER DURCHSATZ

DEFINITION:
EINE PARALLELE AUSFÜHRUNG MEHRERER TRANSAKTIONEN HEIßT SERIALISIERBAR, WENN IHR EFFEKT IDENTISCH MIT DEM EFFEKT EINER (BELIEBIG GEWÄHLTEN) SERIELLEN AUSFÜHRUNG DIESER TRANSAKTIONEN IST.

READ LOCKS (SHARED)

WRITE LOCKS (EXCLUSIVE)

ZWEI-PHASEN-SPERRPROTOKOLL (2PL)

1. JEDES OBJEKT, DAS VON EINER TA BENUTZT WERDEN SOLL, MUSS VORHER ENTSPRECHEND GELOCKT WERDEN

2. EINE TA FORDERT EIN LOCK, DAS SIE SCHON HAT, NICHT ERNEUT AN

3. EINE TA MUSS DIE LOCKS ANDERER TA BEACHTEN. KANN EIN LOCK NICHT GEWAEHRT WERDEN, WIRD GEWARTET.

4. ES GIBT ZWEI PHASEN:

4.A WACHSTUMSPHASE

4.B. SCHRUMPFUNGSPHASE

5. BEI TRANSAKTIONSENDE MUSS DIE TA ALLE IHRE LOCKS ABGEBEN.

SPEZIALFÄLLE DES 2PL

KONSERVATIVE 2PL

89 - TINA UMLANDT, 2015

AM ANFANG WERDEN ALLE LOCKS AUF EINMAL GESETZT

VORTEILE

- KEINE DEADLOCKS
- FREIGABE GESPERRTER OBJEKTE VOR ENDE DER TRANSAKTION

NACHTEILE

- HOHER VERLUST AN PARALLELITAT
- ES MUSS BEKANNT SEIN, WELCHE LOCKS BENÖTIGT WERDEN

STRIKTE 2PL

93 - TINA UMLANDT. 2015

ALLE GESETZTEN WRITE-LOCKS WERDEN AM ENDE DER TRANSAKTION GELÖST

VORTEILE

- VERHINDERT SCHNEEBALLEFFEKT VON KASKADIERENDEN SICH BEEINFLUSSENDEN TAS

NACHTEILE

- LOCKS WERDEN LANGER GEHALTEN ALS NOTIG
- WARTEZEIT VON BLOCKIERTEN TA WIRD ERHÖHT

ISOLATION LEVEL

```
SET TRANSACTION <Isolation_level>
```

- SERIALISABLE
- REPEATABLE READ
- READ COMMITTED
- READ UNCOMMITTED

READ UNCOMMITTED

- LESEOPERATIONEN IGNORIEREN LOCKS
- MEISTE DBMS HABEN EINE EBENE, DIE MINDESTENS LOST UPDATES VERHINDERT

READ COMMITTED

- SETZT SCHREIBSPERREN AUF OBJEKTEN, DIE VERANDERT WERDEN SOLLEN, LESESPERREN ABER NUR KURZ

REPEATABLE READ

- SICHER GESTELLT, DASS WIEDERHOLTES LESEN DIESELBEN ERGEBNISSE HABEN. LOCKS SORGEN DAFÜR, DASS BIS AUF PHANTOM READS KEINE ANOMALIEN AUFTRETEN KÖNNEN

SERIALISABLE

- HOCHSTE ISOLATIONSEBENE
- WIRKUNG PARALLER TAS IST DIE GLEICHE WIE SERIELL
- KANN ZU ABBRÜCHEN KOMMEN

ISOLATION LEVEL
UND
ANOMALIEN

103 - TINA UMLANDT. 2015

Isolation Level	Dirty Read	Nonrepeatable Reads	phantom Reads	Lost updates
Read uncommitted	YES	YES	YES	YES
Read committed	no	YES	YES	no
Repeatable Read	no	no	YES	no
Serialisable	no	no	no	no

ZUGRIFFSMODUS EINER TRANSAKTION

read only

read write

SYNTAX

```
SET TRANSACTION READ ONLY
SET TRANSACTION READ WRITE
```

JDBC

```
try {
    con.setAutoCommit(false);
    // Hier jetzt Änderungsoperationen
    con.commit();
} catch (final SQLException e) {
    con.rollback();
}
```

DAS WAR'S FÜR HEUTE

#scnr