1 - TINA UMLANDT. 2015

FRAGEN ZUR PROBEKLAUSUR

WIEDERHOLUNG

3 - TINA UMLANDT, 2015

```
CREATE VIEW BOHRMASCHINEN (NR, EINKAUFSWERT, AKTUELLER_WERT)
AS
SELECT MNR, NEUWERT, ZEITWERT
FROM MASCHINE
WHERE NAME='Bohrmaschine';

SELECT *
FROM BOHRMASCHINEN WHERE AKTUELLER_WERT > 16000;
```

AUFGABEN

1. ERSTELLEN SIE EINE VIEW, DIE NUR MASCHINEN ENTHALT, DEREN ZEITWERT WENIGER ALS 60% DES NEUWERTS IST
2. ERSTELLEN SIE EINE VIEW MIT MITARBEITERN, DIE MEHR ALS 3000.- VERDIENEN

LOS GEHT'S

\o/

INTEGRITÄT

INTEGRITÄT UND ACID

SEMANTISCHE INTEGRITÄT

C (CONSISTENCY, KONSISTENZ)

PHYSISCHE INTEGRITÄT

D (DAUERHAFTIGKEIT)

A (ATOMARITÄT)

ABLAUFINTEGRITÄT

I (ISOLATION)

INTEGRITÄTSBEDINGUNGEN

BEISPIELE

UBERZIEHUNGEN DER GIROKONTEN VON STUDENTEN SIND NUR BIS 10.000 EUR GESTATTET

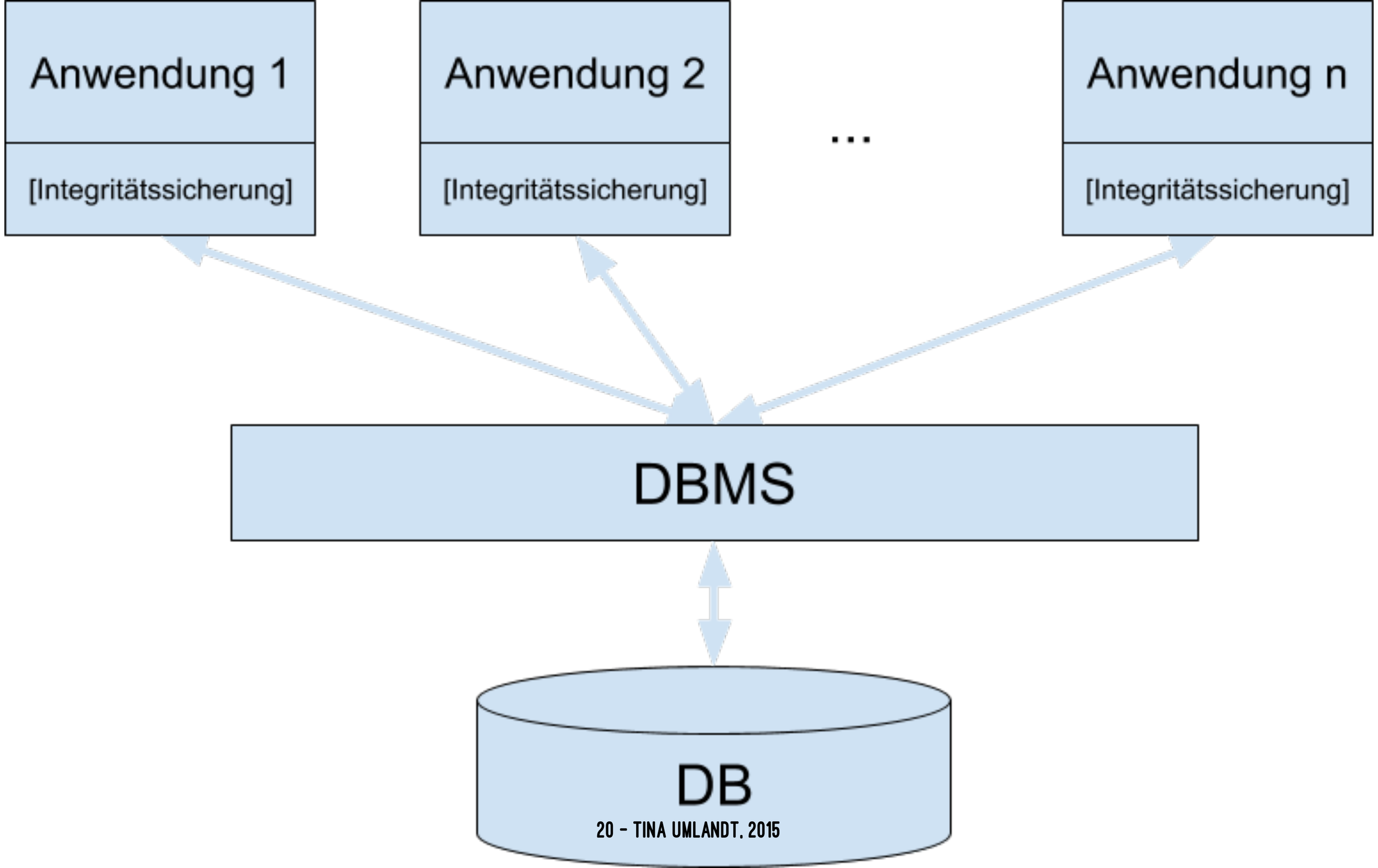
DIE SUMME ALLER KONTEN IST STETS NULL

ANDERUNGEN VON KONTEN SIND PRO TRANSAKTION AUF 10 MIO. BEGRENZT

INTEGRITÄTSSICHERUNGSKOMPONENTE

MÖGLICHKEITEN

1. PROGRAMMCODE


Anwendung 1
[Integritätssicherung]
Anwendung 2
[Integritätssicherung]
...
Anwendung n
[Integritätssicherung]
DBMS
DB

VORTEILE

FLEXIBILITÄT DER PROGRAMMIERSPRACHE

UNABHÄNGIG VOM DBMS

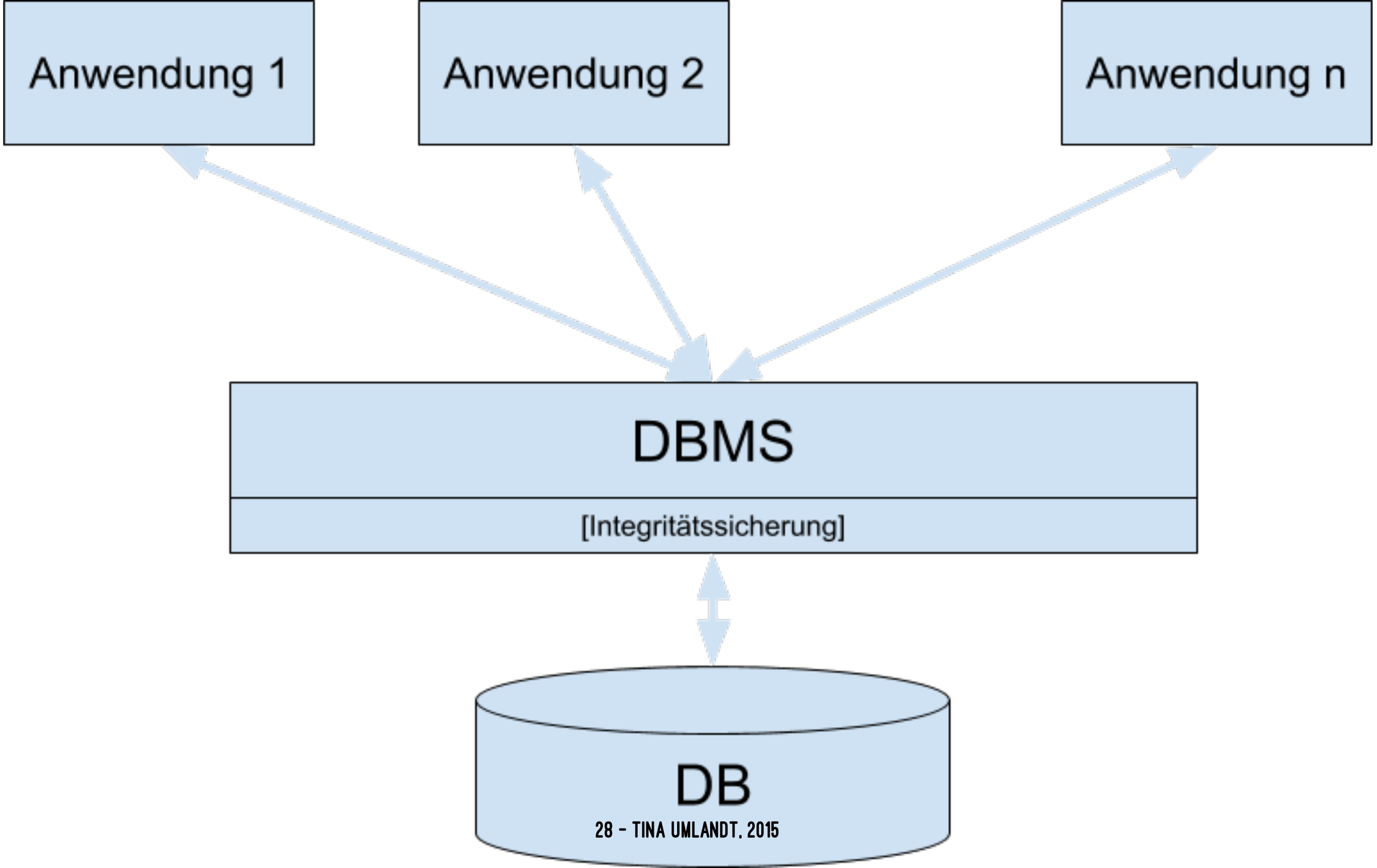
NACHTEILE

KEINE ZENTRALE KONTROLLE

REDUNDANZ

26 - TINA UMLANDT, 2015

2. DBMS


Anwendung 1
Anwendung 2
Anwendung n
DBMS
[Integritätssicherung]
DB

VORTEILE

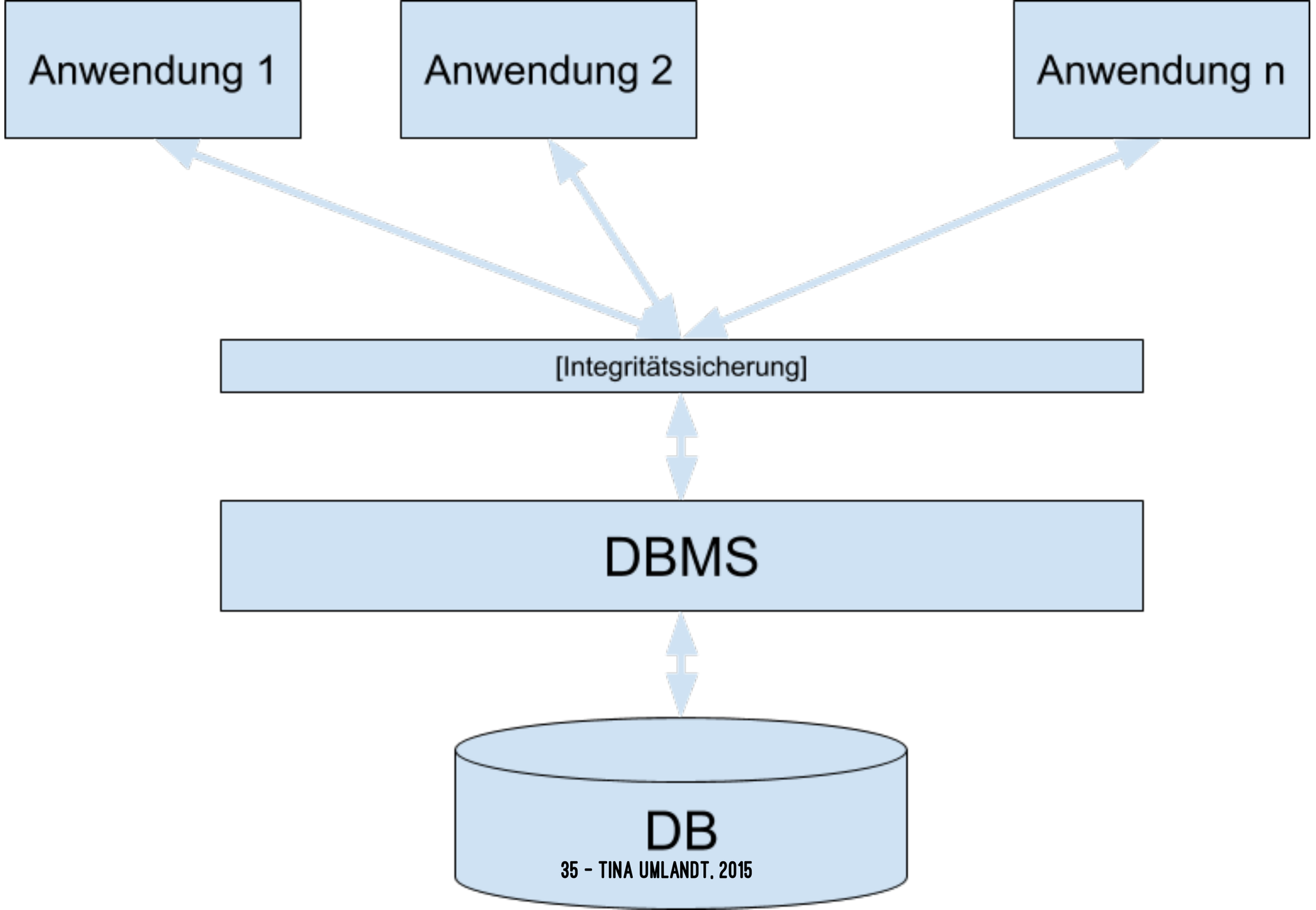
ANWENDUNGSUNABHÄNGIG

REDUNDANZFREI

NACHTEIL

ABHÄNGIG VON SPRACHMITTELN DES DBMS

3. SCHNITTSTELLE


Anwendung 1
Anwendung 2
Anwendung n
[Integritätssicherung]
DBMS
DB

VORTEILE

ANWENDUNGSUNABHÄNGIG

REDUNDANZFREI

UNABHÄNGIG VOM DBMS

NACHTEIL

GEFAHR, DASS AN KAPSELUNG „VORBEIPROGRAMMIERT" WIRD

KLASSIFIKATION

> GRANULARITÄT

> AUSDRUCKSFÄHIGKEIT DER SPRACHE

> ANZAHL DER BETRACHTETEN DATENBANKZUSTÄNDE

> ZEITPUNKT DER ÜBERPRÜFUNG

> REAKTION

BEISPIEL FUR KLASSIFIKATION VON INTEGRITÄTSBEDINGUNGEN

UBERZIEHUNGEN DER GIROKONTEN VON STUDENTEN SIND NUR BIS 10.000 EUR GESTATTET

GRANULARITAT: TUPEL
AUSDRUCKSFÄHIGKEIT: SQL
EIN DB-ZUSTAND
ZEITPUNKT: IMMER
REAKTION: ABBRECHEN

DIE SUMME ALLER KONTEN IST STETS NULL.

GRANULARITÄT: RELATION
AUSDRUCKSFÄHIGKEIT: SQL
EIN DB-ZUSTAND
ZEITPUNKT: TRANSAKTIONSENDE
REAKTION: ABBRECHEN

ANDERUNGEN VON KONTEN SIND PRO TRANSAKTION AUF 10 MIO. BEGRENZT

GRANULARITAT: TUPEL
AUSDRUCKSFÄHIGKEIT: SQL
ZWEI DB-ZUSTÄNDE
ZEITPUNKT: TRANSAKTIONSENDE
REAKTION: ABBRECHEN

FORMULIERUNG
VON INTEGRITÄTSBEDINGUNGEN

INHÄRENTE INTEGRITÄTSBEDINGUNGEN

> TYPINTEGRITÄT

> SCHLÜSSELINTEGRITÄT

> REFERENTIELLE INTEGRITÄT

EXPLIZITE INTEGRITÄTSBEDINGUNGEN

> NULL / NOT NULL

> PRIMARY KEY

- FOREIGN KEY (<Attribute>)
  REFERENCES <tabelle>
  (<Attribute>)

› CHECK (Suchbedingung)

```
CREATE TABLE divisions
   (div_no    NUMBER  CONSTRAINT check_divno
              CHECK (div_no BETWEEN 10 AND 99)
              DISABLE,
    div_name  VARCHAR2(9)  CONSTRAINT check_divname
              CHECK (div_name = UPPER(div_name))
              DISABLE,
    office    VARCHAR2(10)  CONSTRAINT check_office
              CHECK (office IN ('DALLAS','BOSTON',
              'PARIS','TOKYO'))
              DISABLE);
```

ORACLE[1]

[1] HTTPS://DOCS.ORACLE.COM/CD/B19306_01/SERVER.102/B14200/CLAUSES002.HTM#I1002719

TRIGGER

63 - TINA UMLANDT. 2015

```
CREATE OR REPLACE TRIGGER Print_maschine_changes
  BEFORE DELETE OR INSERT OR UPDATE ON maschine
  FOR EACH ROW
WHEN (new.ZEITWERT > 0)
DECLARE
    zeitwert_diff number;
BEGIN
    zeitwert_diff := :new.ZEITWERT  - :old.ZEITWERT;
    dbms_output.put_line('Alter Zeitwert: ' || :old.ZEITWERT);
    dbms_output.put_line('Neuer Zeitwert: ' || :new.ZEITWERT);
    dbms_output.put_line(' -> Differenz: ' || zeitwert_diff);
END;
/
```

EREIGNISSE

65 - TINA UMLANDT, 2015

INSERT, UPDATE, DELETE

CREATE, ALTER, DROP

```
CREATE OR REPLACE TRIGGER drop_trigger
    BEFORE DROP ON KIND
    BEGIN
        RAISE_APPLICATION_ERROR (
            num => -20000,
            msg => 'Cannot drop KIND table!!11elf');
    END;
/
```

DATENBANKINTERNE EREIGNISSE

ZEITPUNKTE

70 - TINA UMLANDT, 2015

before / after

instead of

EIGENSCHAFTEN

```
referencing (new|old) as
```

LEVEL

75 - TINA UMLANDT, 2015

```
for each (statement|row)
```

```
CREATE TRIGGER gehalt_neu_upd_bef
BEFORE update on GEHALT
for each row
when (new.BETRAG / old.BETRAG > 1.1 ) BEGIN
    insert into GEHALT_ALT
    values (:old.GEH_STUFE, :old.BETRAG);
END;
```

REIHENFOLGE

78 - TINA UMLANDT, 2015

1. AUSFÜHRUNG VON BEFORE-ANWEISUNGSTRIGGER

2. AUSFÜHRUNG VON BEFORE-ZEILENTRIGGER

3 . AUSFUHRUNG DES EIGENTLICHEN SQL-STATEMENTS

4. ABARBEITUNG DER AFTER-ZEILENTRIGGER FÜR JEDE ZEILE

DIE PUNKTE 2. BIS 4. WIEDERHOLEN SICH JETZT SO OFT WIE ZEILEN BETROFFEN SIND

5. AUSFÜHRUNG DER AFTER-ANWEISUNGSTRIGGER

SYNTAX

85 - TINA UMLANDT, 2015

```
CREATE [OR REPLACE] TRIGGER [user.]triggername
{BEFORE |AFTER| INSTEAD OF}
{INSERT | UPDATE [OF column [, column] ... ] | DELETE}
[OR {INSERT | UPDATE [OF column [, column] ... ] | DELETE} ]
ON [user. ] {TABLE | VIEW}
[FOR EACH {ROW | STATEMENT}]
[WHEN Bedingung]
Anweisungsblock
```

```
CREATE TRIGGER gehalt_upd_ins
BEFORE insert OR update on GEHALT
for each row
BEGIN
  IF INSERTING THEN
    insert into GEHALT_ALT
    values (:new.GEH_STUFE, :new.BETRAG);
  ELSE -- not inserting, then we are updating BETRAG
    insert into GEHALT_ALT
    values (:old.GEH_STUFE, :old.BETRAG);
  END IF;
END;
/
```

MEHR BEISPIELE

```
CREATE TABLE verlauf (
datum   date ,
pnr number(4),
pnr_neu number(4)
);

CREATE OR REPLACE TRIGGER verlauf_tr
BEFORE UPDATE ON personal
FOR EACH ROW
BEGIN
  INSERT INTO verlauf (datum, pnr, pnr_neu)
  VALUES (sysdate, :old.pnr, :new.pnr);
END;
/
```

```
SELECT *
FROM verlauf;

UPDATE PERSONAL
SET PNR=500 WHERE PNR=177;

SELECT *
FROM verlauf;
```

AUFGABEN

1. SCHREIBEN SIE EINEN TRIGGER, DER BEI EINER ANDERUNG DES ZEITWERTS DER MASCHINEN DEN VERLUST IN PROZENT ANGIBT
2. WENN EINEM MITARBEITER EIN KIND HINZUGFÜGT WIRD, SOLL SIE AUTOMATISCH EINE PRÄMIE IN HÖHE VON 300,- BEKOMMEN.
3.
4.
5.

DAS WAR'S FÜR HEUTE