1 - TINA UMLANDT, 2015

WIEDERHOLUNG

(UNTER ANDEREM)

2 - TINA UMLANDT, 2015

FR-MODELLE

3 - TINA UMLANDT, 2015

AUFGABE

ERSTELLEN SIE EIN ER-MODELL AUS FOLGENDEN RELATIONEN

```
Haus = {Strasse, Hausnummer, Stockwerk, HausId}
Stock = {StockId, Name, Etage, HausID}
Zimmer = {Name, StockId, ZimmerId, Fensteranzahl}
Flur = {FlurId, StockId, Name}
Zimmer_Flur = {ZimmerId, FlurId}
```

- `*Id` SIND DIE PRIMÄRSCHLÜSSEL JEDER RELATION
- WIRD `*Id` IN EINER ANDEREN RELATION BENUTZT, LIEGT HIER EINE FREMDSCHLÜSSELBEZIEHUNG VOR

PL-SQL

5 - TINA UMLANDT, 2015

BEISPIEL

DAS GEHALT DER MITARBEITER EINER VERKAUFSABTEILUNG SETZT SICH AUS EINEM FESTGEHALT (AKTUELL 3000 €) UND EINEM VARIABLEN ANTEIL (AKTUELL 0,987% DES MONATSUMSATZES) ZUSAMMEN. ZWEI TABELLEN SIND EINGERICHTET:

T_UMSATZ **MIT EINER SPALTE** umsatz
T_GEHALT **MIT DEN SPALTEN** umsatz **UND** gehalt

```
DECLARE
  festgehalt constant pls_integer := 3000;
  prozentualer_anteil constant number(7,5) := 0.00987;
  umsatz pls_integer;
  gehalt number(10,2);
BEGIN
  umsatz := 90000;
  gehalt := festgehalt + prozentualer_anteil * umsatz;
  INSERT INTO T_GEHALT VALUES (umsatz, gehalt);
END;
/
```

```
SELECT *
FROM T_GEHALT
```

```
 umsatz | gehalt
----------------
  90000 | 3888,30
```

```
DECLARE
   < Deklarationsteil>
BEGIN
   <Ausführungsteil>
EXCEPTION
   < Ausnahmebehandlung>
END;
```

DATENTYPEN

BOOLEAN	Wahrheitswerte, zulässig sind TRUE, FALSE oder NULL
BINARY-INTEGER/ PLS_INTEGER	Integer-Werte zwischen -2147483647 bis 2147483647
NATURAL	Integer-Werte zwischen 0 bis 2147483647
POSITIVE	Integer-Werte zwischen 1 bis 2147483647
VARCHAR2(n)	String

```
DECLARE
   meine_zahl pls_integer;
der_text VARCHAR2(50);
```

IMPLIZITER DATENTYP

13 - TINA UMLANDT, 2015

```
DECLARE
nummer personal.pnr%type;
```

INITIALISIERUNG

```
DECLARE
   nummer personal.pnr%type;
   zahl integer := 167;
   info varchar2(50) := 'hahn';

BEGIN
   SELECT PNR
   INTO nummer
   FROM PERSONAL
   WHERE NACHNAME="Euler"
END;
/
```

KONTROLLFLUSSANWEISUNGEN

IF

17 - TINA UMLANDT, 2015

```
IF Bedingung_1 THEN
    Anweisung 1;
    Anweisung 2;
    ...
ELSIF Bedingung_2 THEN
    Anweisung 3;
ELSE
    Anweisung 4;
    Anweisung 5;
END IF;
```

```
IF gehalt > 6000 THEN
    anweisung_1
ELSIF gehalt < 4000 THEN
    anweisung_2
ELSE
    anweisung_3
END IF;
```

FOR - SCHLEIFE

20 - TINA UMLANDT, 2015

```
FOR Zähler IN [REVERSE] Untergrenze..Obergrenze LOOP
    Anweisungsblock;
END LOOP;
```

```
for i in reverse 1..100 LOOP
    block;
END LOOP;
```

```
DECLARE
i integer;
BEGIN
  FOR i IN 300..399 LOOP
    INSERT INTO personal_neu (pnr, datum ) VALUES (i, SYSDATE);
  END LOOP;
END;
/
```

WHILE - SCHLEIFE

```
WHILE Bedingung LOOP
    Anweisungsblock;
END LOOP;
```

NULL - **ANWEISUNG**

25 - TINA UMLANDT, 2015

```
IF i = 0 THEN
   i := i + 1;
ELSE
   NULL;
END IF;
```

AUFGABEN

1. SCHREIBEN SIE EINE ANWEISUNG, DIE ALLE NACHNAMEN IN DER KIND-TABELLE GROß SCHREIBEN

2. SCHREIBEN SIE EINE ANWEISUNG, DIE DEN ZEITWERT ALLER MASCHINEN UM 10% VERRINGERT

TRIGGER

28 - 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 OR REPLACE TRIGGER personal_gehalt
BEFORE UPDATE ON personal
FOR EACH ROW
DECLARE
   neuer_betrag number;
BEGIN
   SELECT betrag
   into neuer_betrag
   FROM gehalt
   WHERE geh_stufe=:NEW.geh_stufe;

   :NEW.geh_betrag := neuer_betrag;
END;
/
```

EXCEPTIONS WERFEN

31 - TINA UMLANDT, 2015

```
raise_application_error(
      error_number, message[, {TRUE | FALSE}]);
```

ERROR_NUMBER: EINE NEGATIVE ZAHL ZWISCHEN -20000 ... -20999

MESSAGE: STRING BIS ZU 2048 BYTES

```
DECLARE
   num_tables NUMBER;
BEGIN
   SELECT COUNT(*) INTO num_tables FROM USER_TABLES;
   IF num_tables < 1000 THEN
      raise_application_error(-20101, 'Mindestens 1000 Tabellen erwartet.');
   ELSE
      NULL;
   END IF;
END;
/
```

AUFGABEN

1. DER TRIGGER SOLL ALLE ANDERUNGEN AN PERSONEN, DIE DEN VORNAMEN ANDERN, ABBRECHEN.
2. DER TRIGGER SOLL EINE ERHÖHUNG UM MEHR ALS 10% AN DEN BETRÄGEN DER TABELLE GEHALT VERHINDERN.
3. DER TRIGGER SOLL EINE ERFOLGSMELDUNG IN DEN DBMS - OUTOUT SCHREIBEN, WENN EINE TABELLE ERFOLGREICH ANGELEGT WURDE.
4. DER TRIGGER SOLL EINE ZUSÄTZLICHE PRÄMIE VON 50,- EINTRAGEN, WENN DIE PRÄMIE UNTER 100,- IST. IST DIE PRÄMIE ÜBER 200,-, SOLL DER BETRAG AUF 200,- GEDECKELT WERDEN.
5. DER TRIGGER SOLL VERHINDERN, DASS PERSONEN, DIE KINDER HABEN, UNTER DIE GEHALTSKLASSE 'IT2' AKTUALISIERT WERDEN KÖNNEN.
6. DER TRIGGER SOLL VERHINDERN, DASS IN DER GEHALTSTABELLE DER BETRAG FÜR `it5` MEHR ALS DAS DOPPELTE VON `it1` WERDEN KANN.
7. DER TRIGGER SOLL VERHINDERN, DASS DIE TABELLE `Maschine` GEÄNDERT WERDEN KANN.

AUFGABEN ZU SQL

1. DER SCHLUSSEL DER ABTEILUNGEN SOLL UMBENANNT WERDEN UND DURCH EINEN NUMERISCHEN SCHLÜSSEL ERSETZT WERDEN. ERSTELLEN SIE DIE NOTWENDIGEN SQL - STATEMENTS.
2. LÖSCHEN SIE ALLE VOLLJÄHRIGEN KINDER AUS DER DATENBANK UND SCHREIBEN SIE ZUSÄTZLICH EINEN TRIGGER, DER VERHINDERT, DASS VOLLJÄHRIGE KINDER IN DER TABELLE STEHEN KÖNNEN.

LOS GEHT'S

\o/

CAP-
THEOREM

37 - TINA UMLANDT, 2015

C – KONSISTENZ

A - VERFÜGBARKEIT

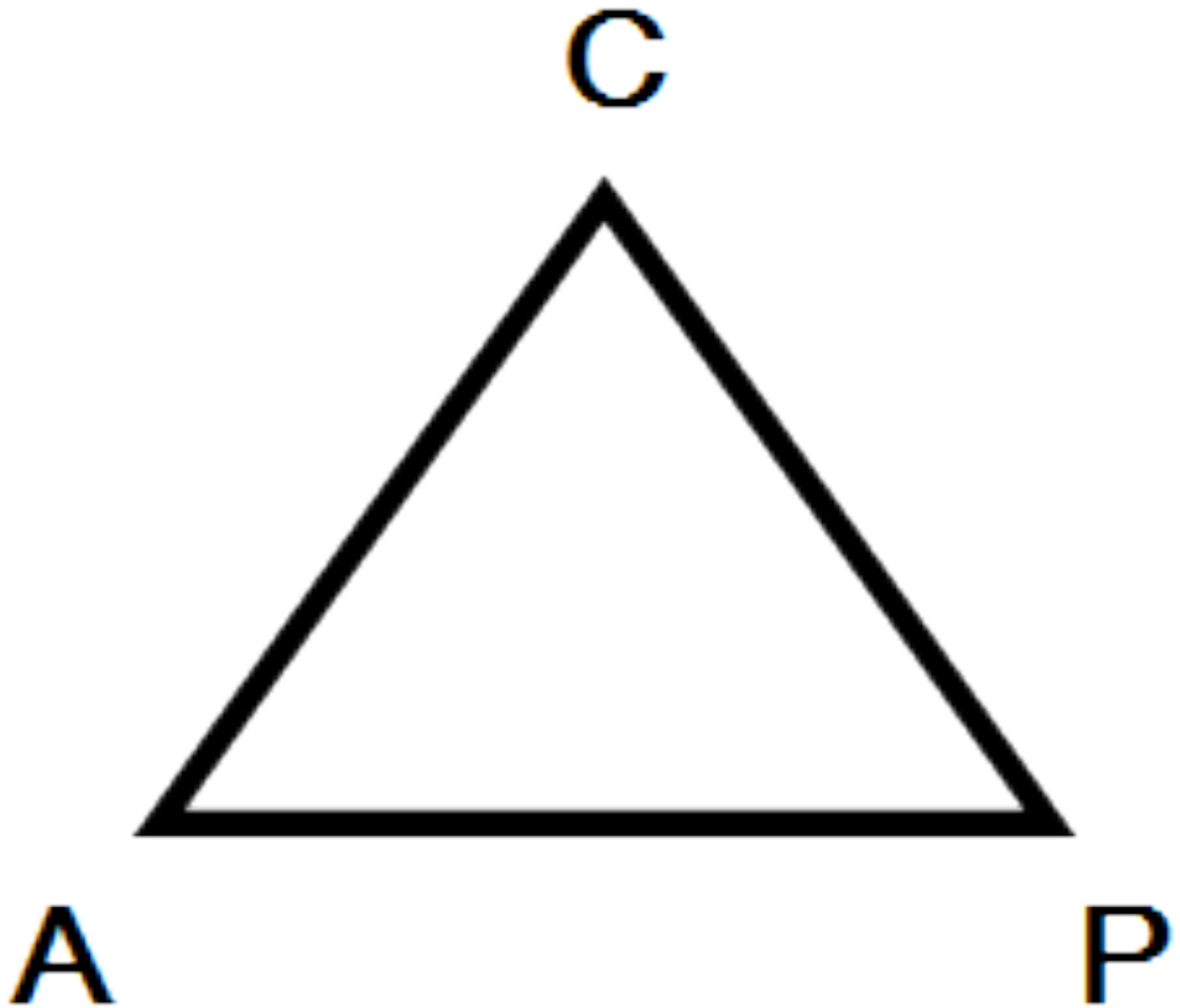
P -

PARTITIONSTOLERAN Z

40 - TINA UMLANDT. 2015

IN VERTEILTEN SYSTEMEN KÖNNEN NUR ZWEI DIESER DREI ANFORDERUNGEN GLEICHZEITIG VOLLSTÄNDIG ERFÜLLT WERDEN

– BREWER'S VERMUTUNG



42 - TINA UMLANDT, 2015

43 - TINA UMLANDT, 2015

CAP-Konfiguration	Relational	Key-Value	Spaltenorientiert	Dokumentorientiert
Consistent + Available **(CA)**	Microsoft SQL Server, Oracle, MySQL, Postgres, Sybase	—	Vertica	—
Consistent + Partition-Tolerant **(CP)**	—	Scalaris, Berkeley DB, memcached, Redis	BigTable, HBase	MongoDB
Available + Partition-Tolerant **(AP)**	—	Dynamo	Cassandra	SimpleDB, CouchDB

LÖSUNG

BASICALLY AVAILABLE. SOFT STATE. EVENTUALLY CONSISTENT (BASE)

KONSISTENZ DER VERFÜGBARKEIT UNTERORDNEN

48 - TINA UMLANDT, 2015

49 - TINA UMLANDT, 2015

NOSQL

TYPEN

51 - TINA UMLANDT, 2015

KEY-VALUE-STORE

```
shop.settings.vat=19

shop.country="de_DE"
```

BEISPIEL: REDIS

```
redis> ping
PONG
redis> set foo bar
OK
redis> get foo
"bar"
```

```
redis> incr mycounter
(integer) 1
redis> incr mycounter
(integer) 2
redis> get mycounter
"2"
```

EIGENSCHAFTEN

STRUKTUR DER DATEN IST NICHT VORGEGEBEN (SCHEMALOS)

57 - TINA UMLANDT, 2015

DATENTYP IST NICHT VORGEGEBEN

QUERIES SIND AUF SCHLUSSEL BEGRENZT

59 - TINA UMLANDT, 2015

EINFACHE SKALIERBARKEIT

COLUMN STORE
(SPALTENORIENTIERTE DB)

EIGENSCHAFTEN

62 - TINA UMLANDT, 2015

INHALTE SPALTENWEISE STATT ZEILENWEISE ABSPEICHERT

```
001:10,Smith,Joe,40000;
002:12,Jones,Mary,50000;
003:11,Johnson,Cathy,44000;
004:22,Jones,Bob,55000;
```

```
10:001,12:002,11:003,22:004;
Smith:001,Jones:002,Johnson:003,Jones:004;
Joe:001,Mary:002,Cathy:003,Bob:004;
40000:001,50000:002,44000:003,55000:004;
```

PERFORMANCE-VORTEIL BEI DATENAGGREGATIONEN

```
SELECT SUM(Gehalt) FROM PERSONAL_GEHALT;
UPDATE tabelle SET Gehalt = Gehalt * 1.03;
```

KOMPRESSION DER DATEN

67 - TINA UMLANDT, 2015

GRAPH DATABASES

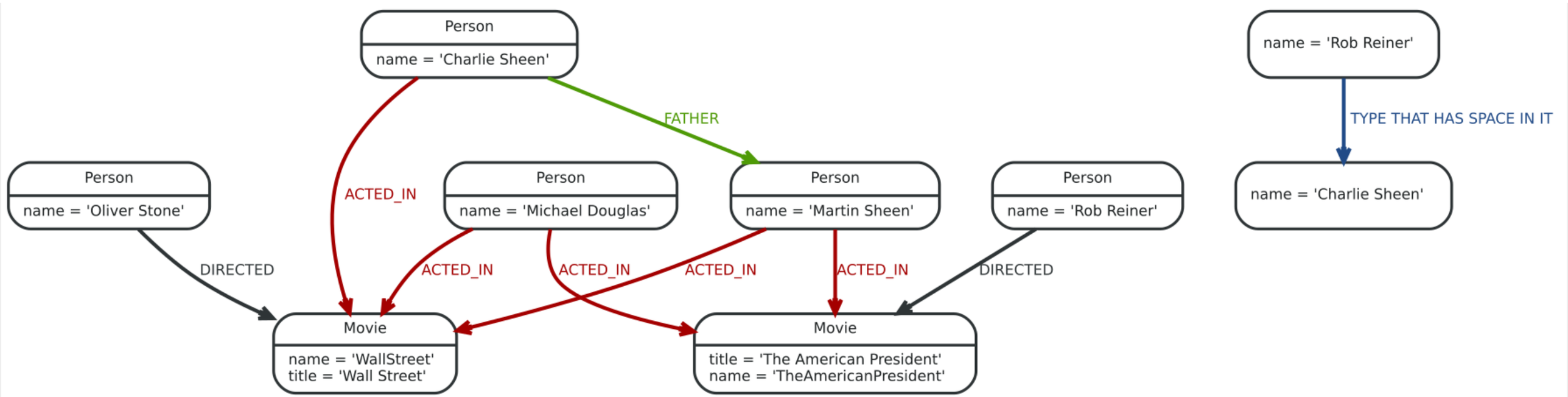
EIGENSCHAFTEN

69 - TINA UMLANDT, 2015

DATEN = KNOTEN
BEZIEHUNGEN DER DATEN
ZUEINANDER = KANTEN

KEINE STANDARDISIERTE ABFRAGESPRACHE

71 - TINA UMLANDT, 2015


Person
name = 'Charlie Sheen'
FATHER
name = 'Rob Reiner'
TYPE THAT HAS SPACE IN IT
Person
name = 'Oliver Stone'
ACTED_IN
Person
name = 'Michael Douglas'
Person
name = 'Martin Sheen'
Person
name = 'Rob Reiner'
name = 'Charlie Sheen'
DIRECTED
ACTED_IN
ACTED_IN
ACTED_IN
ACTED_IN
DIRECTED
Movie
name = 'WallStreet'
title = 'Wall Street'
Movie
title = 'The American President'
name = 'TheAmericanPresident'

```
MATCH (movie:Movie)
RETURN movie
```

```
movie
-----------------------------------------------------------------------
Node[3]{title:"The American President",name:"TheAmericanPresident"}
-----------------------------------------------------------------------
Node[4]{name:"WallStreet",title:"Wall Street"}
-----------------------------------------------------------------------
2 rows
```

DOKUMENTEN-ORIENTIERT

BEISPIEL: COUCHDB-DOKUMENT

```
{
    "_id" : "00a271787f89c0ef2e10e88a0c0001f4"
    "_rev": "5509377776",
    "name": "Peter Lustig",
     "address": {
         "street": "Teststr.",
         "city": "Hamburg"
     }
}
```

EIGENSCHAFTEN

77 - TINA UMLANDT, 2015

DATEN SIND DOKUMENTE, DIE NACH EINEM BESTIMMTEN DATENFORMAT (XML, JSON, ETC) STRUKTURIERT SIND

ES KANN AUCH NACH INHALTEN GESUCHT WERDEN

79 - TINA UMLANDT, 2015

ANFRAGESPRACHE ABHANGIG VOM DATENFORMAT

80 - TINA UMLANDT, 2015

81 - TINA UMLANDT. 2015

NACHSTES (UND LETZTES) MAL

- DATENSCHUTZ
- LETZTE FRAGEN ZUR KLAUSUR

DAS WAR'S FÜR HEUTE

#scnr