



# WIEDERHOLUNG

(UNTER ANDEREM)

# ER-MODELLE

# 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

# 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/<br>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

```
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

```
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

```
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

```
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

```
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

```
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

**C – KONSISTENZ**

# A – VERFÜGBARKEIT

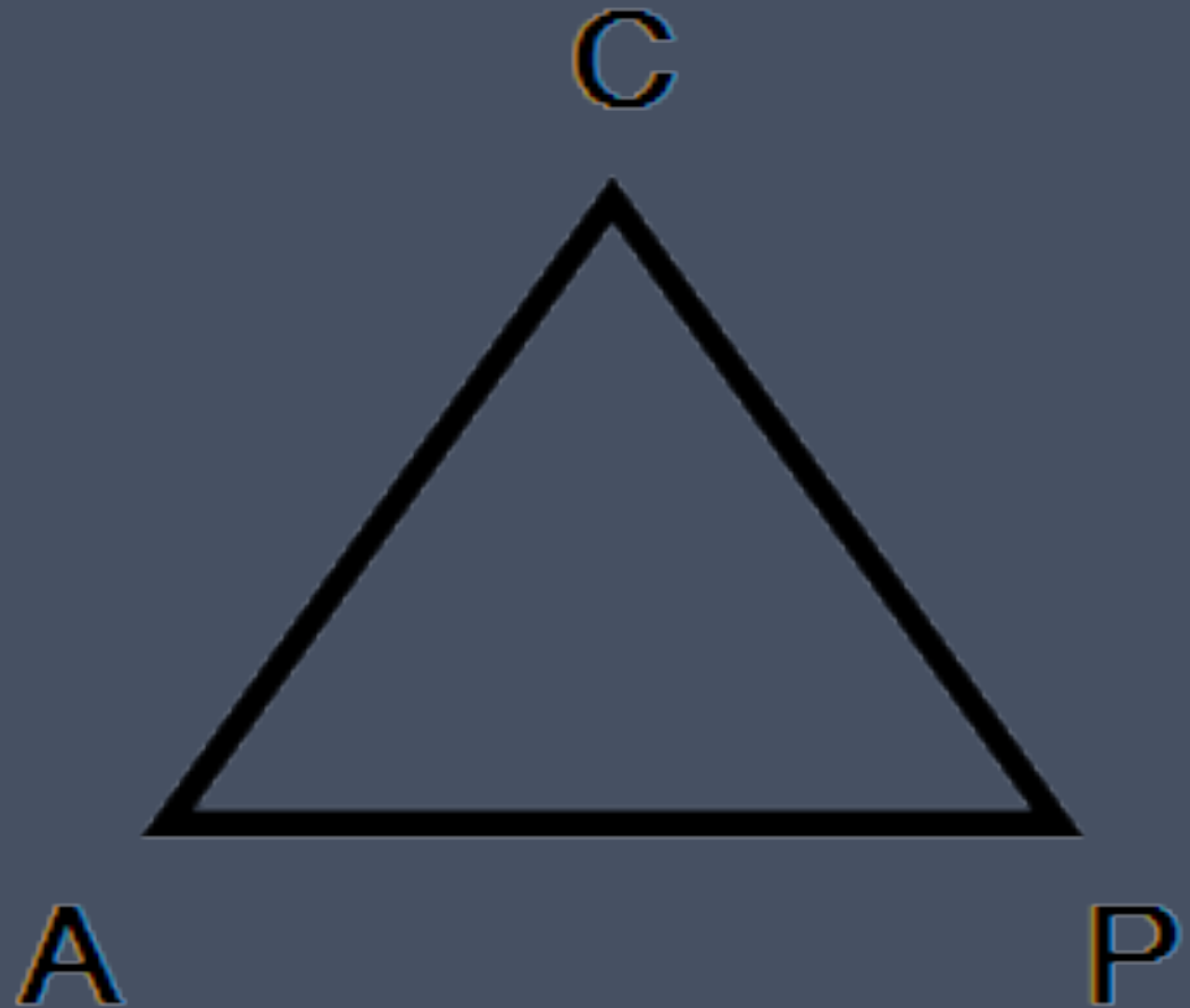
P -

PARTITIONSTOLERAN  
Z



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

– BREWER'S VERMUTUNG



AP

CA

CP

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

# LÖSUNG

**BASICALLY AVAILABLE, SOFT  
STATE, EVENTUALLY  
CONSISTENT (BASE)**

# KONSISTENZ DER VERFÜGBARKEIT UNTERORDNEN





# NOSQL

# TYPEN

# 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)**

**DATENTYP IST NICHT  
VORGEGEBEN**

**QUERIES SIND AUF SCHLUSSEL  
BEGRENZT**

# EINFACHE SKALIERBARKEIT

# **COLUMN STORE (SPALTENORIENTIERTE DB)**

EIGENSCHAFTEN

**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;
```

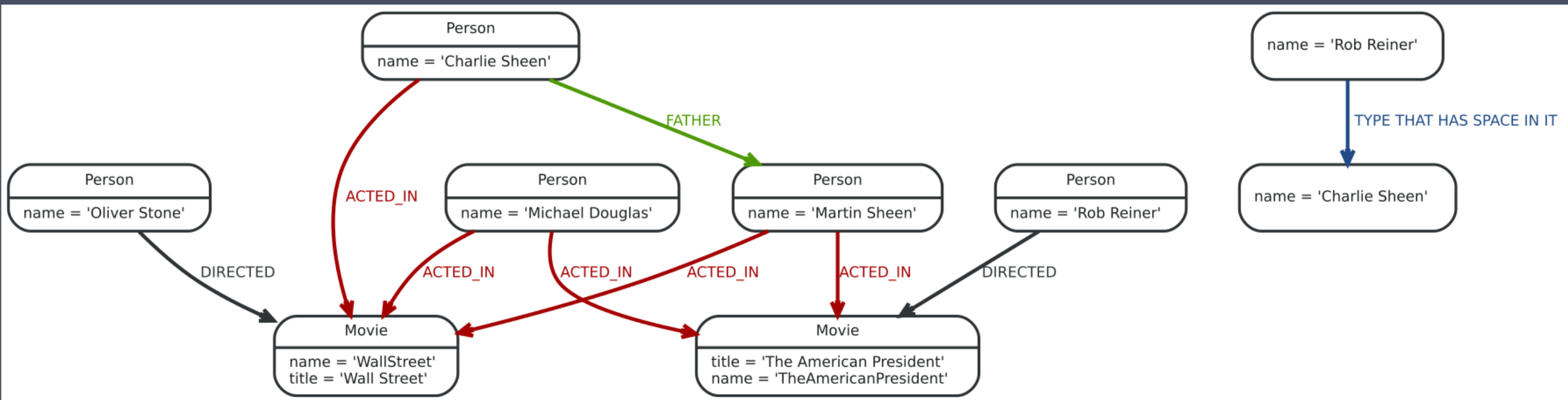
# KOMPRESSION DER DATEN

# GRAPH DATABASES

EIGENSCHAFTEN

**DATEN = KNOTEN**  
**BEZIEHUNGEN DER DATEN**  
**ZUEINANDER = KANTEN**

# KEINE STANDARDISIERTE ABFRAGESPRACHE





```
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

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

**ES KANN AUCH NACH INHALTEN  
GESUCHT WERDEN**

# ANFRAGESPRACHE ABHÄNGIG VOM DATENFORMAT





# NACHSTES (UND LETZTES) MAL

- > DATENSCHUTZ
- > LETZTE FRAGEN ZUR KLAUSUR



DAS WAR'S  
FÜR HEUTE

#scnr