Informationssysteme

Skriptum zur Vorlesung

Dipl.-Ing. Paul Panhofer BSc.[1]*

1 ZID, TU Wien, Taubstummengasse 11, 1040, Wien, Austria

Abstract: Ein Informationssystem ist ein soziotechnisches System, das die Abarbeitung von Informationsnachfrage zur Aufgabe hat. Es handelt sich um ein Mensch/Aufgabe/Technik-System, das Daten produziert, beschafft, verteilt und verarbeitet.

Daneben bezeichnen Informationssysteme im allgemeineren Sinne Systeme von Informationen, die in einem wechselseitigen Zusammenhang stehen.

Die Begriffe Informationssystem und Anwendungssystem werden häufig synonym verwendet. Dabei werden Informationssysteme im engeren Sinne als computergestützte Anwendungssysteme verstanden. Es ist jedoch wichtig zu verstehen, dass ein Anwendungssystem mit Anwendungssoftware und Datenbank nur Teil eines Informationssystems sind.

MSC: paul.panhofer@gmail.com

Keywords:

Contents

*E-mail: paul.panhofer@tuwien.ac.at

.

.

1. SQL - Data Query Lanuguage

01

SQL Grundlagen

1.1. SQL Grundlagen

1.Beispiel - select Klausel

Beispielbeschreibung

- **Schwerpunkt:** SQL Grundlagen, Select Klausel
- **Komplexität:** einfach
- **Skriptum:** Seite 44

Aufgabenstellung: Select Klausel

- Geben Sie für alle Employees folgende Spalten aus:
  - First_Name, Last_Name, Salary und Email
- Sortieren Sie das Ergebnis nach Last_Name und First_Name.

Lösung: Select Klausel

```
1  -- ------------------------------------------
2  -- Loesung: Select Klausel
3  -- ------------------------------------------
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32 .
```

2.Beispiel - II Operator

Beispielbeschreibung

- **Schwerpunkt:** SQL Grundlagen, Select Klausel, II Operator, Spaltenalias
- **Komplexität:** einfach
- **Skriptum:** Seite 45

Aufgabenstellung: II Operator

- Geben Sie für alle Employees folgende Spalten aus:
  - First_Name, Last_Name, Salary und Email
- Sortieren Sie das Ergebnis nach Last_Name und First_Name.
- Das Ergebnis soll als einezelne Spalte dargestellt werden.
- Verwenden Sie den Begriff employee_data als Spaltenbezeichnung.

Lösung: II Operator

```
1  -- ------------------------------------------
2  -- Loesung: II Operator
3  -- ------------------------------------------
4
5
6
7
8
9
10
11
12
13
14 .
```

3.Beispiel - where Klausel

Beispielbeschreibung

- **Schwerpunkt:** SQL Befehl, where Klausel
- **Komplexität:** einfach
- **Skriptum:** Seite 47

Aufgabenstellung: where Klausel

- Finden Sie alle employees die im Unternehmen als IT_PROG arbeiten. Beachten Sie dass nur employees ausgegeben werden sollen die mehr als 12 000€ verdienen.
- Geben Sie für die employees first_name, last_name, job_id und salary aus.
- Sortieren Sie das Ergebnis nach first_name und last_name.

Lösung: where Klausel

```
1  -- ------------------------------------------
2  -- Loesung: where Klausel
3  -- ------------------------------------------
4
5
6
7
8
9  .
```

4.Beispiel - like Operator

Beispielbeschreibung

- **Schwerpunkt:** SQL Grundlagen, where Klausel, like Operator
- **Komplexität:** einfach
- **Skriptum:** Seite 48

Aufgabenstellung: like Operator

- Geben Sie alle employees aus deren last_name mit H beginnt aber nicht mit r endet.
- Geben Sie für die employees first_name, last_name und department_id aus.

Lösung: like Operator

```
1  -- ------------------------------------------
2  -- Loesung: like Operator
3  -- ------------------------------------------
4
5
6
7
8
9  .
```

5.Beispiel - in Operator

Beispielbeschreibung

- **Schwerpunkt:** SQL Grundlagen, where Klausel, in Operator
- **Komplexität:** einfach
- **Skriptum:** Seite 48

Aufgabenstellung: in Operator

- Finden Sie alle employess die entweder als SA_MAN, SA_REP oder als ST_CLERK arbeiten.
- Es sollen aber keine IT_PROG ausgegeben werden.
- Geben Sie den Last_name und first_name für die employees aus.

Lösung: in Operator

```
-- ------------------------------------------
-- Loesung: in Operator
-- ------------------------------------------




.
```

6.Beispiel - 3 wertige Logik

Beispielbeschreibung

- **Schwerpunkt:** 3 wertige Logik
- **Komplexität:** einfach
- **Skriptum:** Seite 49

Aufgabenstellung: 3 wertige Logik

- Werten Sie die folgenden logischen Terme aus.
- Jede der Spalten kann auch null Werte enthalten.

Lösung: 3 wertige Logik

```
-- ------------------------------------------
-- Loesung: 3 wertige Logik
-- ------------------------------------------
not(index > 100)









real = 10 and not(index = 100)









not(real >= 32 or not (index < 0 and deptno = 20))









.
```

7.Beispiel - between Operator

Beispielbeschreibung

- **Schwerpunkt:** SQL Grundlagen, where Klausel, between Operator
- **Komplexität:** einfach
- **Skriptum:** Seite 49

Aufgabenstellung: between Operator

- Finden Sie alle **Angestellten** die nicht weniger als 10000 und nicht mehr als 17000 verdienen.
- Beschränken Sie sich bei der Formulierung ihrer Query auf die **employees** Tabelle.

Lösung: between Operator

```
-- ------------------------------------------
-- Loesung: between Operator
-- ------------------------------------------









.
```

8.Beispiel - between Operator

Beispielbeschreibung

- **Schwerpunkt:** SQL Grundlagen, where Klausel, between Operator
- **Komplexität:** einfach
- **Skriptum:** Seite 49

Aufgabenstellung: between Operator

- Finden Sie alle **Angestellten** die 5 Jahre oder länger im Unternehmen arbeiten.
- Mit der **to_date** Funktion können Sie Datumswerte angeben.

Lösung: between Operator

```
-- ------------------------------------------
-- Loesung: between Operator
-- ------------------------------------------
to_date('15.05.2012', 'dd.mm.yyyy')









.
```

9.Beispiel - Order By Klausel

Beispielbeschreibung

- **Schwerpunkt:** SQL Grundlagen, where Klausel, Fetch Klausel, Order By Klausel
- **Komplexität:** einfach
- **Skriptum:** Seite 49, 50

Aufgabenstellung: Order By Klausel

- Finden Sie alle **employees** des Unternehmens
- Geben Sie den **first_name** und **last_name** der Angestellten an.
- Ordnen Sie das Ergebnis der Abfrage nach dem **first_name** absteigen und dem **last_name** aufsteigend.
- Geben Sie nur die ersten 10 Angestellten aus.

Lösung: Order by Klausel

```
-- ------------------------------------------
-- Loesung: Order By Klausel
-- ------------------------------------------
```

10.Beispiel - Fetch Klausel

Beispielbeschreibung

- **Schwerpunkt:** SQL Grundlagen, where Klausel, Fetch Klausel, Order By Klausel
- **Komplexität:** einfach
- **Skriptum:** Seite 49, 50, 51

Aufgabenstellung: Fetch Klausel

- Finden Sie alle `employees` des Unternehmens.
- Geben Sie den `first_name` und `last_name` der `employees` aus.

Lösung: Fetch Klausel

```
-- ------------------------------------------
-- Loesung: Fetch Klausel
-- ------------------------------------------
-- Geben Sie nur die ersten 10% der employees aus









-- Formulieren Sie eine Paginierung. Geben Sie die
-- Angestellten 90 - 110 aus.

.
```

1.2. Datenaggregation

1.Beispiel - Datenaggregation

Beispielbeschreibung

- **Schwerpunkt:** Datenaggregation
- **Komplexität:** einfach
- **Skriptum:** Seite 49, 50

Aufgabenstellung: Datenaggregation

- Erklären Sie den Begriff der **Datenaggregation**. Geben Sie ein Beispiel für den Vorgang der Datenaggregation.

Lösung: Datenaggregation

```
-- ------------------------------------------
-- Loesung: Datenaggregation
-- ------------------------------------------









.
```

2.Beispiel - Inner Join

Beispielbeschreibung

- **Schwerpunkt:** Datenaggregation, Inner Join
- **Komplexität:** einfach
- **Skriptum:** Seite 55, 56, 57

Aufgabenstellung: Inner Join

- Finden Sie alle `employees` die in der `IT` Abteilung arbeiten.
- Geben Sie für die Datensätzen die folgenden Spalten aus: `first_name`, `last_name`, `department_name`, `country_name`.

Lösung: Inner Join

```
1  -- ------------------------------------------
2  -- Loesung: Datenaggregation
3  -- ------------------------------------------
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34 .
```

3.Beispiel - Inner Join

Beispielbeschreibung

- **Schwerpunkt:** Datenaggregation, Inner Join
- **Komplexität:** mittel
- **Skriptum:** Seite 55, 56, 57

Aufgabenstellung: Inner Join

- Gilt die folgende **mengentheoreitsche Äquivalenz**? Begründen Sie Ihre Antwort.

$$\texttt{A} \text{ inner join } \texttt{B} \Leftrightarrow \texttt{B} \text{ inner join } \texttt{A}$$

- Mit welcher **Mengenoperation** kann der **Inner Join** verglichen werden?

Lösung: Inner Join

```
1  -- ------------------------------------------
2  -- Loesung: Datenaggregation
3  -- ------------------------------------------
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33 .
```

4.Beispiel - Inner Join

Beispielbeschreibung

- **Schwerpunkt:** Datenaggregation, Inner Join
- **Komplexität:** mittel
- **Skriptum:** Seite 55, 56, 57

Aufgabenstellung: Inner Join

- Die Tabellen `A` und `B` werden über eine **Fremdschlüsselspalte** miteinander gejoined.
- Welche **Beziehung**[1] muss zwischen `A` und B vorherschen damit das Ergebnis des **Inner Joins genausoviele** Zeilen enthält wie die Tabelle A.
- Welche **Beziehung** muss zwischen `A` und B vorherschen damit das Ergebnis des **Innner Joins weniger** Zeilen enthält als die Tabelle A.
- Welche **Beziehung** muss zwischen `A` und B vorherschen damit das Ergebnis des **Innner Joins mehr** Zeilen enthält als die Tabelle A.

Lösung: Inner Join

```
-- ------------------------------------------
-- Loesung: Datenaggregation
-- ------------
.
```

5.Beispiel - Cross Join

Beispielbeschreibung

- **Schwerpunkt:** Datenaggregation, Cross Join
- **Komplexität:** einfach
- **Skriptum:** Seite 55, 56, 57

Aufgabenstellung: Cross Join

- Gegeben ist eine Tabelle `A` mit 5 Zeilen und eine Tabelle `B` mit 10 Zeilen, wieviel Zeilen hat der **Cross Join** zwischen `A` und `B`.

Lösung: Cross Join

```
-- ------------------------------------------
-- Loesung: Cross Join
-- ------------
.
```

6.Beispiel - Cross Join

Beispielbeschreibung

- **Schwerpunkt:** Datenaggregation, Cross Join
- **Komplexität:** einfach
- **Skriptum:** Seite 55, 56, 57

Aufgabenstellung: Cross Join

- Gilt die folgende **mengentheoretische Äquvalenz**? Begründen Sie Ihre Antwort.

`A` cross join `B` $\Leftrightarrow$ `B` cross join `A`

Lösung: Cross Join

```
-- ------------------------------------------
-- Loesung: Cross Join
-- ------------
.
```

[1] *Relation*

7.Beispiel - Cross Join

Beispielbeschreibung

- **Schwerpunkt:** Datenaggregation, Cross Join
- **Komplexität:** mittel
- **Skriptum:** Seite 55, 56, 57

Aufgabenstellung: Cross Join

- Formulieren Sie die folgenden 2 Queries mit Hilfe von **cross joins**.

Lösung: Cross Join

```
1  -- ------------------------------------------
2  -- Loesung: Cross Join
3  -- ------------------------------------------
4  SELECT e.last_name,
5         e.first_name,
6         d.department_name
7  FROM employees e JOIN departements d
8  ON e.department_id = d.department_id;
9
10
11
12
13
14
15
16
17
18
19
20
21
22 SELECT e.last_name, e.first_name,
23        d.department_name,
24        c.country_name
25 FROM employees e JOIN departements d
26 ON e.department_id = d.department_id
27 JOIN locations l ON d.location_id = l.location_id
28 JOIN countries c ON l.country_id = c.country_id;
29
30
31
32
33
34
35
36
37
38
39
40
41 .
```

❑

8.Beispiel - Left Join

Beispielbeschreibung

- **Schwerpunkt:** Datenaggregation, Left Join
- **Komplexität:** einfach
- **Skriptum:** Seite 58

Aufgabenstellung: Left Join

- Geben Sie für jeden `EMPLOYEE` aus in welcher Abteilung er arbeitet.
- Für jeden Datensatz soll dazu der `FIRST_NAME`, `LAST_NAME` und `DEPARTMENT_NAME` ausgegeben werden.
- Ist ein employee keiner Abteilung zugeordnet soll die Zeichenkette `NO DEPARTMENT` ausgegeben werden.

Lösung: Left Join

```
1  -- ------------------------------------------
2  -- Loesung: Left Join
3  -- ------------------------------------------
4
5
6
7
8
9
10
11
12 .
```

❑

9.Beispiel - Left Join

Beispielbeschreibung

- **Schwerpunkt:** Datenaggregation, Left Join
- **Komplexität:** einfach
- **Skriptum:** Seite 58

Aufgabenstellung: Left Join

- Gilt die folgende **mengentheoretische Äquvalenz**? Begründen Sie Ihre Antwort.

`A` left join `B` ⇔ `B` left join `A`

- Mit welcher **Mengenoperation** kann der **Left Join** verglichen werden?

▸ **Lösung: left Join** ▾

```
-- ------------------------------------------
-- Loesung: Left Join
-- ------------------------------------------
.
```

❑

10.Beispiel - Left Join

Beispielbeschreibung ▾

- **Schwerpunkt:** Datenaggregation, Left Join
- **Komplexität:** mittel
- **Skriptum:** Seite 58

▸ **Aufgabenstellung: Left Join** ▾

- Beschreiben die folgenden Queries dieselbe Abfrage.
- Ihre Antwort muss unabhängig vom Datenbestand der Datenbank gelten.

▸ **Lösung: Left Join** ▾

```
-- ------------------------------------------
-- Loesung: Left Join
-- ------------------------------------------
SELECT e.last_name, e.first_name,
     d.department_name, c.country_name
FROM employees e LEFT JOIN departements d
  ON e.department_id = d.department_id
LEFT JOIN locations l ON d.location_id =
     l.location_id
LEFT JOIN countries c ON l.country_id =
     c.country_id;
```

```
SELECT e.last_name, e.first_name,
       d.department_name,
       c.country_name
FROM employees e
LEFT JOIN departements d ON e.department_id =
     d.department_id
JOIN locations l ON d.location_id = l.location_id
JOIN countries c ON l.country_id = c.country_id;
.
```

❑

1.3. Aggregatfunktionen

1.Beispiel - Aggregatfunktionen

Beispielbeschreibung

- **Schwerpunkt:** Aggregatfunktionen
- **Komplexität:** mittel
- **Skriptum:** Seite 70, 76

Aufgabenstellung: Aggregatfunktionen

- Geben Sie das maximale und minimale Gehalt der `employees` im Unternehmen an.
- Warum wird zur Lösung der Aufgabe keine `Group By Klausel` verwendet.
- Warum kann nicht ebenfalls der `last_name` der Angestellten ausgegeben werden?

Lösung: Aggregatfunktionen

```
1  -- ------------------------------------------
2  -- Loesung: Aggregatfunktionen
3  -- ------------------------------------------
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33 .
```

❑

2.Beispiel - Group By Klausel

Beispielbeschreibung

- **Schwerpunkt:** Aggregatfunktionen, Group By Klausel
- **Komplexität:** mittel
- **Skriptum:** Seite 70, 71, 72, 73, 74, 75

Aufgabenstellung: Group By Klausel

- Geben Sie für jedes `department` des Unternehmens das minimale und maximale Gehalt an. Geben Sie ebenfalls den `department_name` an.
-

Lösung: Group By Klausel

```
1  -- ------------------------------------------
2  -- Loesung: Group By Klausel
3  -- ------------------------------------------
4
5
6
7
8
9
10
11
12
13
14 .
```

❑

3.Beispiel - Group By Klausel

Beispielbeschreibung

- **Schwerpunkt:** Aggregatfunktionen, Group By Klausel
- **Komplexität:** einfach
- **Skriptum:** Seite 70, 71, 72, 73, 74, 75

Aufgabenstellung: Group By Klausel

- Geben Sie für jedes `department` die Anzahl der Mitarbeiter an, die dort arbeiten.
- Nach welchen Werten kann das Ergebnis sortiert werden?

▸ Lösung: Group By Klausel ▾

```
1  -- ---------------------------------------------
2  -- Loesung: Group By Klausel
3  -- ---------------------------------------------
4
5
6
7
8
9
10
11 .
```

❑

4.Beispiel - Group By Klausel

Beispielbeschreibung ▾

- **Schwerpunkt:** Aggregatfunktionen, Group By Klausel
- **Komplexität:** einfach
- **Skriptum:** Seite 70, 71, 72, 73, 74, 75

▸ Aufgabenstellung: Group By Klausel ▾

- Geben Sie für die Abteilungen, jedes Landes die Anzahl der Mitarbeiter an, die dort arbeiten.

▸ Lösung: Group By Klausel ▾

```
1  -- ---------------------------------------------
2  -- Loesung: Group By Klausel
3  -- ---------------------------------------------
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18 .
```

❑

5.Beispiel - Group By Klausel

Beispielbeschreibung ▾

- **Schwerpunkt:** Aggregatfunktionen, Group By Klausel
- **Komplexität:** einfach
- **Skriptum:** Seite 70, 71, 72, 73, 74, 75

▸ Aufgabenstellung: Group By Klausel ▾

- Beschreiben die beiden Queries dieselbe Abfrage? Erklären Sie Ihre Lösung.

▸ Lösung: Group By Klausel ▾

```
1  -- ---------------------------------------------
2  -- Loesung: Group By Klausel
3  -- ---------------------------------------------
4  SELECT location_id, department_id,
5         count(e.employee_id)
6  FROM employees e JOIN departemts d on
       e.department_id = d.department_id
7  JOIN locations l on d.location_id = l.location_id
8  GROUP BY d.department_id, l.location_id;
9
10
11 SELECT location_id, department_id,
12        count(e.employee_id)
13 FROM employees e JOIN departemts d on
       e.department_id = d.department_id
14 JOIN locations l on d.location_id = l.location_id
15 GROUP BY l.location_id, d.department_id;
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35 .
```

❑

6.Beispiel - Having Klausel

Beispielbeschreibung

- **Schwerpunkt:** Aggregatfunktionen, Having Klausel, Group By Klausel
- **Komplexität:** mittel
- **Skriptum:** Seite 75

Aufgabenstellung: Having Klausel

- Geben Sie jene Abteilungen aus, in denen mehr als 5 Mitarbeiter beschäftigt sind.
- Geben Sie den `department_name`, und die Anzahl der Mitarbeiter an.
- Alle `IT Programmierer` sollen dabei nicht berücksichtigt werden.

Lösung: Having Klausel

```
1  -- ------------------------------------------
2  -- Loesung: Having Klausel
3  -- ------------------------------------------
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30 .
```

7.Beispiel - Having Klausel

Beispielbeschreibung

- **Schwerpunkt:** Aggregatfunktionen, Group By Klausel, Having Klausel
- **Komplexität:** mittel
- **Skriptum:** Seite 75

Aufgabenstellung: Having Klausel

- Finden Sie alle Länder in denen sich mehr als 3 Abteilungen befinden.
- Geben Sie `country_id` und die Anzahl der Abteilungen aus.
- Beachten Sie das `Sales` Abteilungen nicht berücksichtigt werden sollen.

Lösung: Having Klausel

```
1  -- ------------------------------------------
2  -- Loesung: Having Klausel
3  -- ------------------------------------------
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31 .
```

8.Beispiel - Group By Klausel

Beispielbeschreibung

- **Schwerpunkt:** Aggregatfunktionen, Group By Klausel
- **Komplexität:** mittel
- **Skriptum:** Seite 75

Aufgabenstellung: Having Klausel

- Geben Sie für jedes Land aus wieviele Abteilungen es dort gibt.
- Gibt es in einem Land keine Abteilungen soll 0 ausgegeben werden.

Lösung: Having Klausel

```
1  -- ------------------------------------------
2  -- Loesung: Group By Klausel
3  -- ------------------------------------------
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37 .
```

1.4. Subselect

1.Beispiel - Subselect

Beispielbeschreibung

- **Schwerpunkt:** Subselect, where Klausel, having Klausel, select Klausel
- **Komplexität:** mittel
- **Skriptum:** Seite 78, 79, 80

Aufgabenstellung: Subselect

- Geben Sie an wieviel Mitarbeiter, die Abteilung mit den meisten Mitarbeitern hat.
- Verwenden Sie das `subselect` in der `where`, `having` oder `select` Klausel.

Lösung: Subselect

```
1  -- ------------------------------------------
2  -- Loesung: Subselect
3  -- ------------------------------------------
4
5
6
7
8
9
10
11
12 .
```

2.Beispiel - Subselect

Beispielbeschreibung

- **Schwerpunkt:** Subselect, where Klausel, having Klausel, select Klausel
- **Komplexität:** mittel
- **Skriptum:** Seite 78, 79, 80

Aufgabenstellung: Subselect

- Geben Sie den Namen der Abteilung mit den meisten Mitarbeitern an.
- Verwenden Sie das `subselect` in der `where`, `having` oder `select` Klausel.

▸ Lösung: Subselect ▾

```
1  -- ------------------------------------------
2  -- Loesung: Subselect
3  -- ------------------------------------------
4
5
6
7
8
9
10
11
12 .
```

❑

3.Beispiel - Subselect

Beispielbeschreibung ▾

- **Schwerpunkt:** Subselect, where Klausel, having Klausel, select Klausel
- **Komplexität:** komplex
- **Skriptum:** Seite 78, 79, 80

▸ Aufgabenstellung: Subselect ▾

- Geben Sie den `country_name` des Landes an, indem die meisten Mitarbeiter arbeiten.
- Verwenden Sie das `subselect` in der `where`, `having` oder `select` Klausel.

▸ Lösung: Subselect ▾

```
1  -- ------------------------------------------
2  -- Loesung: Subselect
3  -- ------------------------------------------
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20 .
```

❑

4.Beispiel - Subselect

Beispielbeschreibung ▾

- **Schwerpunkt:** Subselect, where Klausel, having Klausel, select Klausel
- **Komplexität:** mittel
- **Skriptum:** Seite 78, 79, 80

▸ Aufgabenstellung: Subselect ▾

- Geben Sie für jede Abteilung den Mitarbeiter mit dem höchsten Einkommen an.
- Geben Sie für den Datensatz den `department_name`, `first_name` und `last_name` aus.
- Verwenden Sie das `subselect` in der `where`, `having` oder `select` Klausel.

▸ Lösung: Subselect ▾

```
1  -- ------------------------------------------
2  -- Loesung: Subselect
3  -- ------------------------------------------
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33 .
```

❑

5.Beispiel - Subselect, from Klausel

Beispielbeschreibung

- **Schwerpunkt:** Subselect, From Klausel
- **Komplexität:** mittel
- **Skriptum:** Seite 81

Aufgabenstellung: Subselect

- Geben Sie an wieviele Mitarbeiter, die Abteilung mit den meisten Mitarbeitern hat.
- Geben Sie für den Datensatz den `DEPARTMENT_NAME` und die Anzahl der Mitarbeiter an..
- Schreiben Sie die Query ohne die Komposition von Aggregatfunktionen.

Lösung: Subselect

```
1  -- ------------------------------------------
2  -- Loesung: Subselect
3  -- ------------------------------------------
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33 .
```

6.Beispiel - Subselect, from Klausel

Beispielbeschreibung

- **Schwerpunkt:** Subselect, From Klausel
- **Komplexität:** mittel
- **Skriptum:** Seite 81

Aufgabenstellung: Subselect

- Geben Sie an wieviele Mitarbeiter, die Abteilung mit den meisten Mitarbeitern hat.
- Geben Sie für den Datensatz den `DEPARTMENT_NAME` und die Anzahl der Mitarbeiter an..
- Schreiben Sie die Query ohne die Komposition von Aggregatfunktionen.

Lösung: Subselect

```
1  -- ------------------------------------------
2  -- Loesung: Subselect
3  -- ------------------------------------------
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33 .
```

7.Beispiel - Subselect, from Klausel

Beispielbeschreibung

- **Schwerpunkt:** Subselect, From Klausel
- **Komplexität:** mittel
- **Skriptum:** Seite 81

Aufgabenstellung: Subselect

- Geben Sie für jede Abteilung den Mitarbeiter mit dem höchsten Einkommen an.
- Geben Sie für den Datensatz `DEPARTMENT_NAME`, `FIRST_NAME`, `LAST_NAME` an.
- Verwenden Sie das Subselect in der from Klausel.

Lösung: Subselect

```
1  -- ------------------------------------------
2  -- Loesung: Subselect
3  -- ------------------------------------------
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33 .
```

❑

8.Beispiel - Subselect, from Klausel

Beispielbeschreibung

- **Schwerpunkt:** Subselect, From Klausel
- **Komplexität:** komplex
- **Skriptum:** Seite 81

Aufgabenstellung: Subselect

- Geben Sie für jedes Land die Abteilung mit den meisten Mitarbeitern an.
- Geben Sie für den Datensatz den `COUNTRY_NAME`, `DEPARTMENT_NAME` und die Anzahl der Mitarbeiter an.
- Verwenden Sie das Subselect in der from Klausel.

Lösung: Subselect

```
1  -- ------------------------------------------
2  -- Loesung: Subselect
3  -- ------------------------------------------
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32 .
```

❑

9.Beispiel - Subselect

.

Beispielbeschreibung

- **Schwerpunkt:** Subselect, SQL Engine
- **Komplexität:** mittel
- **Skriptum:** Seite 81

Aufgabenstellung: Subselect

- Beschreiben Sie in welcher Reihenfolge die Klauseln der Abfrage in Beispiel 7 ausgeführt werden.

Lösung: Subselect

```
1  -- ------------------------------------------
2  -- Loesung: Subselect
3  -- ------------------------------------------
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38 .
```

❑

1.5. Zeilenfunktionen

1.Beispiel - Datumsfunktionen

Beispielbeschreibung

- **Schwerpunkt:** Datumsfunktionen
- **Komplexität:** einfach
- **Skriptum:** Seiten 59 -64

Aufgabenstellung: Datumsfunktionen

- Geben Sie alle Mitarbeiter an die im 2.Quartal 2005 im Unternehmen beschäftigt waren.
- Geben Sie `first_name` und `last_name` aus.
- Arbeiten Sie fürs erste nur mit den Daten aus der `employees` Tabelle.

Lösung: Datumsfunktionen

```
1  -- ------------------------------------------
2  -- Loesung: Datumsfunktionen
3  -- ------------------------------------------
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33 .
```

❑

2.Beispiel - Datumsfunktionen

Beispielbeschreibung

- **Schwerpunkt:** Datumsfunktionen
- **Komplexität:** einfach
- **Skriptum:** Seiten 59 -64

Aufgabenstellung: Datumsfunktionen

- Geben Sie für alle Mitarbeiter an wieviele Tage sie im Unternehmen beschäftigt sind.
- Geben Sie `first_name`, `last_name` und die Anzahl der Tage der Beschäftigung aus. Geben Sie zusätzlich aus wieviele Jahre das sind.
- Arbeiten Sie fürs erste nur mit den Daten aus der `employees` Tabelle.

Lösung: Datumsfunktionen

```
1  -- ------------------------------------------
2  -- Loesung: Datumsfunktionen
3  -- ------------------------------------------
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35 .
```

❑

3.Beispiel - Datumsfunktionen

Beispielbeschreibung

- **Schwerpunkt:** Datumsfunktionen, union Klausel
- **Komplexität:** komplex
- **Skriptum:** Seiten 59 -64

Aufgabenstellung: Datumsfunktionen

- Geben Sie für alle Mitarbeiter an wieviele Tage sie im Unternehmen beschäftigt sind.
- Geben Sie `first_name`, `last_name` und die Anzahl der Tage der Beschäftigung aus.
- Berücksichtigen Sie ebenfalls die Daten aus der `JOB_HISTORY` Tabelle.

Lösung: Datumsfunktionen

```
1  -- ------------------------------------------
2  -- Loesung: Datumsfunktionen
3  -- ------------------------------------------
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35 .
```

4.Beispiel - Textfunktionen

Beispielbeschreibung

- **Schwerpunkt:** Textfunktionen, fetch Klausel
- **Komplexität:** einfach
- **Skriptum:** Seiten 59 -64

Aufgabenstellung: Textfunktionen

- Finden Sie alle Projekte `C_PROJECTS` die in ihrer Beschreibung (`DESCRIPTION`) den Ausdruck `Method` enthalten.
- Geben Sie für die Projekte jeweils den Titel und die Beschreibung aus.
- Ein Projekt soll dabei ausgewählt werden, unabhängig davon wie der `Method` Ausdruck geschrieben wird. z.B.: METHOD, metHOD, usw.
- Geben Sie nur 20% der Ergebnisse zurück.

Lösung: Textfunktionen

```
1  -- ------------------------------------------
2  -- Loesung: Textfunktionen
3  -- ------------------------------------------
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30 .
```

5.Beispiel - Textfunktionen

Beispielbeschreibung

- **Schwerpunkt:** Textfunktionen, fetch Klausel
- **Komplexität:** einfach
- **Skriptum:** Seiten 59 -64

Aufgabenstellung: Textfunktionen

- Für einen internen Report sollen die Daten der Angestellten angepasst werden.
- Geben Sie für alle Angestellten den `LAST_NAME` und `FIRST_NAME` aus.
- Für die email Adresse des Angestellten soll der erste Buchstabe des Vornamens getrennt durch einen Punkt gefolgt vom Nachnamen angegeben werden.
- Alle Buchstaben der email sollen klein geschrieben sein.
- Implementieren Sie eine Paginierung für die Abfrage.

Lösung: Textfunktionen

```
1  -- ---------------------------------------------
2  -- Loesung: Textfunktionen
3  -- ---------------------------------------------
4  --z.b.: Max Musterman -> m.musterman@oracle.com
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27 .
```

6.Beispiel - Numerische Funktionen

Beispielbeschreibung

- **Schwerpunkt:** Numerische Funktionen
- **Komplexität:** einfach
- **Skriptum:** Seiten 59 -64

Aufgabenstellung: Textfunktionen

- Geben Sie alle Angestellten den `LAST_NAME`, `FIRST_NAME` und das `SALARY` aus.
- Die Angabe des Gehalts soll dabei formatiert sein.
- Implementieren Sie eine Paginierung für die Abfrage.

Lösung: Numerische Funktionen

```
1  -- ---------------------------------------------
2  -- Loesung: Numerische Funktionen
3  -- ---------------------------------------------
4  z.B.: 10000 -> 10,000.00
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36 .
```

2. Entwicklung relationaler Datenbanken

01

SQL Grundlagen

2.1. Normalformen

1.Beispiel - Normalformen

Beispielbeschreibung

- **Schwerpunkt:** Relationale Datenmodell, Normalformen
- **Komplexität:** einfach
- **Skriptum:** -

Aufgabenstellung: Normalformen

- Beschreiben sie die ersten 3 Normalformen der relationalen Modellierung

Lösung: Normalformen

```
1  -- ----------------------------------------
2  -- Loesung: Normalformen
3  -- ----------------------------------------
4  1.Normalform
5
6
7
8
9
10
11
12
13
14
15 2.Normalform
16
17
18
19
20
21
22
23
24
25
26
27 3.Normalform
28
29
30
31
32
33
34
35
36
37
38 .
```

OID	Name	Abteilung	ProjektNr	ProjektName	ProjektZeit
101	Panhofer Paul	1:Physik	11, 12	A, B	60, 40
102	Lauer Christian	2:Chemie	13	C	100
103	Bauer Iris	2:Chemie	11, 12, 13	A, B, C	20, 50, 30
104	Kauer Alexander	1:Physik	11, 13	A, C	80, 20

Abbildung 1. **2.Beispiel - 0te Normalform**

2.Beispiel - Normalformen

Beispielbeschreibung

- **Schwerpunkt:** Relationale Datenmodell, Normalformen
- **Komplexität:** einfach
- **Skriptum:** -

Aufgabenstellung: Normalformen

- Entwickeln Sie aus der angegebenen Tabelle die 1te Normalform

Lösung: Normalformen

```
1  -- ------------------------------------------
2  -- Loesung: Normalformen
3  -- ------------------------------------------
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23 .
```

❑

3.Beispiel - Normalformen

Beispielbeschreibung

- **Schwerpunkt:** Relationale Datenmodell, Normalformen
- **Komplexität:** einfach
- **Skriptum:** -

Aufgabenstellung: Normalformen

- Entwickeln Sie aus der angegeben Tabelle die 2te und 3te Normalform.

Lösung: Normalformen

```
1  -- ------------------------------------------
2  -- Loesung: Normalformen
3  -- ------------------------------------------
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23 .
```

❑

Name	Artikel	Adresse	Newsletter	Supplier	Supplier Phone	Preis
Smith Alan	Xbox One	35 Palm Str., Miami	Xbox News	Microsoft	8982398	250
Banks Roger	Playstation 4	47 Campus Rd, Boston	PlayStation News	Sony	2363432	300
Wilson Evan	Xbox One	28 Rock Av., Denver	Xbox News	Microsoft	8982398	450
Smith Alan	PlayStation 4	47 Campus Rd, Boston	PlayStation News	Sony	2363432	300

Abbildung 2. **4.Beispiel - 0te Normalform**

4.Beispiel - Normalformen

Beispielbeschreibung

- **Schwerpunkt:** Relationale Datenmodell, Normalformen
- **Komplexität:** einfach
- **Skriptum:** -

Aufgabenstellung: Normalformen

- Entwickeln Sie aus der angegebenen Tabelle die 1te Normalform

Lösung: Normalformen

```
1  -- ------------------------------------------
2  -- Loesung: Normalformen
3  -- ------------------------------------------
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23 .
```

5.Beispiel - Normalformen

Beispielbeschreibung

- **Schwerpunkt:** Relationale Datenmodell, Normalformen
- **Komplexität:** einfach
- **Skriptum:** -

Aufgabenstellung: Normalformen

- Entwickeln Sie aus der angegeben Tabelle die 2te und 3te Normalform.

Lösung: Normalformen

```
1  -- ------------------------------------------
2  -- Loesung: Normalformen
3  -- ------------------------------------------
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23 .
```

2.2. Relationale Modellierung

1.Beispiel - Vererbung

Beispielbeschreibung

- **Schwerpunkt:** Relationale Modellierung, Vererbung
- **Komplexität:** einfach
- **Skriptum:** -

Aufgabenstellung: Auflösen der Vererbung

- Beschreiben Sie die unterschiedlichen Vorgehensweisen um Vererbung im relationalen Modell aufzulösen.
- Welche Unterschiede bestehen in den Ergebnissen. Beschreiben Sie die Vorteile und Nachteile der beiden Strategien.

Lösung: Vererbung

```
1  -- ------------------------------------------
2  -- Loesung: Relationale Modellierung
3  -- ------------------------------------------
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31 .
```

❑

2.Beispiel - Vererbung

Beispielbeschreibung

- **Schwerpunkt:** Relationale Modellierung, Vererbung
- **Komplexität:** mittel
- **Skriptum:** -

Aufgabenstellung: Auflösen der Vererbung

- Die folgende Informationsstruktur wurde an Sie weitergegeben um daraus das relationale Modell zu entwickeln.
- Überlegen Sie welche der Vererbungsstrategien Sie verwenden sollten. Begründen Sie Ihre Wahl.

Erklärung: Informationstruktur

- **Person:** Eine Person ist eindeutig durch eine **ID**. Für eine Person wird zusätzlich gespeichert ein **Vorname** (Varchar2(20) - not null), ein **Nachname** (Varchar2(20) - not null) und eine **OUID** (Varchar2(20) - not null, unique).

  Bei Personen unterscheidet man nach Studenten, Assistenen und Professoren. Für Studenten wird zusätzlich eine **Matrikelnr** (Varchar2(8) - not null, unique) und das Datum der **Inskribierung** (Date - not null) gespeichert.

- **Forschungsschwerpunkt:** Eine Forschungsschwerpunkt ist eindeutig durch eine **ID**. Für einen Forschungsschwerpunkt wird eine eindeutige **Bezeichnung** (Varchar2(20) - not null, unique) gespeichert. Ein Forschungsschwerpunkt kann mehreren Disziplinen (Mathematik, Informatik, Chemie, Physik, Architektur) zugeordnet werden.

  Professoren und Assisteneten forschen in mehreren Forschungsschwerpunkten. Ein Forschungsschwerpunkt kann von mehreren Professoren oder Assistenten erforscht werden.

- **Vorlesung:** Eine Vorlesung ist eindeutig identifiziert durch eine **ID**. Für eine Vorlesung wird eine eindeutige **Bezeichnung** (Varchar2(30) - not null, unique) gespeichert. Für eine Vorlesung wird zusätzlich die **Dauer** (Number - not null, @Min(0), @Max(10)) gespeichert.

  Eine Vorlesung wird von einem Professor abgehalten unterstützt von einem oder mehreren Assistenten. Ein Professor kann mehrere Vorlesungen abhal-

ten. Selbiges gilt für Assistenten. Vorlesungen werden von Studenten besucht. Ein Student kann mehrere Vorlesungen besuchen.

- **Prüfung:** Eine Prüfung ist eindeutig identifiziert durch eine **ID**. Für Prüfungen wird das **Datum** (DATE - NOT NULL) gespeichert an dem sie abgehalten werden. Eine Prüfung wird von einem Studenten für eine Vorlesung bei einem Professor oder Assistenten abgelegt.

❑

3.Beispiel - Relationale Modellierung

Beispielbeschreibung

- **Schwerpunkt:** RELATIONALE MODELLIERUNG
- **Komplexität:** MITTEL
- **Skriptum:** -

Aufgabenstellung: Relationale Modellierung

- DIE FOLGENDE INFORMATIONSSTRUKTUR WURDE AN SIE WEITERGEGEBEN UM DARAUS DAS RELATIONALE MODELL ZU ENTWICKELN.
- ÜBERLEGEN SIE WELCHE DATENTYPEN UND CONSTRAINTS FÜR DIE EINZELNEN SPALTEN SINNVOLL SIND.

Erklärung: Informationstruktur

Ein Hersteller von Skiliften möchte die in seinen Produkten verwendete Software aktualisieren. Dies inkludiert den Entwurf einer neuen Datenbank. Es sind keine NULL-Werte erlaubt, und Redundanzen sollen vermieden werden.

- **Lift:** Ein Lift ist eindeutig durch eine **ID**. Jeder Lift wird durch seinen Namen identifiziert (NAME), und besteht aus zwei bis fünf Stationen.
- **Station:** Eine Station ist eindeutig durch eine **ID**. Zu jeder Station wird die Seehoöhe vermerkt (SEEHOOEHE). Außerdem sind Stationen innerhalb eines Lifts durch ihren Namen (NAME) eindeutig identifizierbar – der selbe Stationsname (wie z.B. "Bergstation" oder "Talstation") kann aber in verschiedenen Liften verwendet werden. Es soll möglich sein zu speichern, welche Stationen eines Liftes jeweils direkt verbunden sind, wie weit die Strecke zwischen diesen Stationen ist (LAENGE), und wie schnell der Lift zwischen diesen Stationen im Normalfall fährt (GESCHW).

  Zum Beispiel: Besteht ein Lift aus "Talstation", "Mittelstation", und "Bergstation", so ist die "Talstation" mit der "Mittelstation" verbunden, und die "Mittelstation" mit der "Bergstation".
- **Schranke:** In jeder Station wird der Zutritt zum Lift mit mindestens einer Schranke geregelt. Jede Schranke hat eine Nummer (NR) die innerhalb der Station eindeutig ist, sowie einen Typ (TYP).
- **Liftkarte:** Fur jede Liftkarte ist die Kombination aus Kartennummer (NUMMER), Vorname (VNAME) und Nachname (NNAME) des Besitzers eindeutig. Jede Verwendung einer bestimmten Karte ist durch den Zeitpunkt der Verwendung (UHRZEIT) eindeutig identifizierbar, wobei natürlich zwei verschiedene Karten zum selben Zeitpunkt verwendet werden können. Zu jeder Verwendung einer Karte soll außerdem die eindeutige Schranke gespeichert werden, an welcher die Karte verwendet wurde.
- **Fehler:** Ein Fehler wird eindeutig identifiziert durch eine id (ID). Bei der Verwendung einer Karte können mehrere Fehler auftreten. Jeder solcher Fehler ist (im Rahmen einer bestimmten Verwendung) durch den Fehlercode (FEHLERCODE) eindeutig bestimmbar.

❑

4.Beispiel - Relationale Modellierung

Beispielbeschreibung

- **Schwerpunkt:** RELATIONALE MODELLIERUNG
- **Komplexität:** MITTEL
- **Skriptum:** -

Aufgabenstellung: Relationale Modellierung

- DIE FOLGENDE INFORMATIONSSTRUKTUR WURDE AN SIE WEITERGEGEBEN UM DARAUS DAS RELATIONALE MODELL ZU ENTWICKELN.
- ÜBERLEGEN SIE WELCHE DATENTYPEN UND CONSTRAINTS FÜR DIE EINZELNEN SPALTEN SINNVOLL SIND.

Erklärung: Informationstruktur

Bei einem punktgenauen, präzise zielgerichteten Mitschnitt des gesamten Internetverkehrs zur Terrorismusbekämpfung wurden zufällig Informationen über die IT Systeme des Osterhasen aufgefangen. Ein befreundeter

Reporter hat diese auf Wikileaks entdeckt und bittet Sie nun um Hilfe bei deren Aufarbeitung.

- **Versteck:** Ein Versteck ist eindeutig durch eine **ID**. Jedes Versteck hat eine eindeutige Verstecknummer (VNR). Außerdem wird natürlich der Ort des Verstecks(ORT), sowie seine Kapazität (KAPAZITAET) gespeichert. Osternester sind eine besondere Art von Verstecken, für welche ein Name (NAME) sowie deren Qualität (QUALITAET) vermerkt wird. Jedes Osternest gehört einer oder zwei Personen.
- **Person:** Eine Person ist eindeutig durch eine **ID**. Personen können durch Ihren Namen (NAME) gemeinsam mit dem Geburtsdatum (GEBDAT) eindeutig identifziert werden. Zusätzlich ist zu jeder Person eine Anmerkung (ANMERKUNG) notiert (dieses Feld enthält Informationen wie Nahrungsmittelallergien, . . .). Jeder Person können beliebig viele Osternester gehören, nicht jeder Person gehört jedoch auch ein Osternest.
- **Osternest:** Eine Osternest ist eindeutig durch eine **ID**. Jedes Osternest wird darüber hinaus von mindestens einer Person gebaut. An einem Osternest können jedoch nicht mehr als 10 Personen mitbauen. Jede Person kann an einer beliebigen Zahl von Osternestern mitbauen. Verstecke die der Osterhase in seiner Datenbank verwaltet werden für ihn von Scouts entdeckt. Scouts sind Personen, fur die zusätzlich ein login (LOGIN) und Passwort (PW) (fur das Online-Meldesystem des Osterhasen) gespeichert werden. Jedes Versteck wird von mindestens einem Scout entdeckt, und es gibt auch Scouts die bislang noch kein Versteck entdeckt haben.
- **Geschenk:** Eine Geschenk ist eindeutig durch eine **ID**. Ein Geschenk hat einen eindeutigen Barcode (BARCODE). Weiters besitzt jedes Geschenk eine ebenfalls eindeuti- ge Inventarnummer (INR). Jedes Geschenk liegt in genau einem Versteck, und ist für genau eine Person bestimmt - wobei auch jede Person mindestens ein Geschenk erhält. In jedem Versteck können eine beliebige Anzahl von Geschenken liegen (wobei es auch möglich ist dass ein Versteck leer bleibt).
- **Widmung:** Eine Widmung ist eindeutig durch eine Zu manchen Geschenken sind Widmungen vermerkt, wobei eine Widmung aus einem Text (TEXT) und einem Absender (VON) besteht. Der Absender dient dazu eine Widmung innerhalb eines Geschenkes eindeutig zu iden- tifizieren. Zu jedem Geschenk kann es maximal 2 Widmungen geben.

❑

5.Beispiel - Relationale Modellierung

Beispielbeschreibung

- **Schwerpunkt:** Relationale Modellierung
- **Komplexität:** mittel
- **Skriptum:** -

Aufgabenstellung: Relationale Modellierung

- Die folgende Informationsstruktur wurde an Sie weitergegeben um daraus das relationale Modell zu entwickeln.
- Überlegen Sie welche Datentypen und Constraints für die einzelnen Spalten sinnvoll sind.

Erklärung: Informationstruktur

Als Sie am Ende des Semsters mit einigen KollegInnen zusammensitzen beschließen Sie, dass eine kleine Applikation, welche einen Überblick über die unterschiedlichen Algorithmen zur Lösung verschiedener Probleme bietet, eine praktische Sache wäre. Zu Ihrer großen Freude erhalten Sie die Aufgabe, die nötige Datenbank zu entwerfen.

- **Problem:** Ein Problem ist eindeutig durch eine **ID**. Ein Problem besitzt einen eindeutigen Namen (NAME). Zu jedem Problem soll seine Beschreibung (BESCHREI- BUNG) sowie eine Referenz auf die wichtigste Literatur/Unterlagen zu dem Problem (REF) gespeichert werden. Außerdem soll fur jedes Problem vermerkt werden, in welche anderen Probleme es überführt werden kann, sowie die Laufzeit (LAUFZEIT) fur solch eine Übersetzung.
- **Algorithmus:** Ein Algorithmus ist eindeutig durch eine **ID**. Jeder Algorithmus ist eindeutig identifiziert durch das Problem das er löst gemeinsam mit seiner Bezeichnung (BEZ). Zusätzlich soll eine Beschreibung des Algorithmus in Pseudocode (PSEUDOCODE), sowie seine Laufzeit (LAUFZEIT) in der Datenbank vermerkt werden. Für den Fall, dass jemand eine Implementierung für einen der Algorithmen erstellt hat, sollen auch diese in der Datenbank verwaltet werden.
- **Implementierung:** Eine Implementierung ist eindeutig durch eine **ID**. Eine Implementierung eines Algorithmus ist wiederum eindeutig identifiziert durch den Algorithmus zusammen mit einem Spitznamen (SNAME) und der Versionsnummer (VERSION).

Jede Implementierung ist in mindestens einer Programmiersprache erstellt.

- **Implementierung:** Eine Implementierung ist eindeutig identifiziert durch ihren Namen (NAME) gemeinsam mit einem Zusatz zum Namen (NZusatz). Fur deklarative Programmiersprachen soll daruber hinaus gespeichert werden, in welcher LVA es mehr Informationen daruber gibt (LVA).
- **Person:** Abschließend soll in der Datenbank gespei chert werden, welche Personen die Implementierungen erstellt haben, und bei wem man evtl. mehr Informationen zu einem Problem erhalten kann: Fur jede Person soll sowohl die Matri- kelnummer (MATRNR) als auch eine E-mail Adresse (EMAIL) gespeichert werden, wobei nur die Email-Adresse eine Person eindeutig identifiziert. Zu jeder Implementierung soll nun vermerkt werden, welche Person(en) sie erstellt hat (haben). Dabei gilt es zu beachten, dass jede Implementierung von mindestens einer Person entwickelt wurde. Zusatzlich wird zu jedem Problem genau eine Person als ExpertIn zugewiesen.

❑

2.3. Datenbankartefakte

1.Beispiel - Datenbankartefakte

Beispielbeschreibung

- **Schwerpunkt:** Constraints, Index, View
- **Komplexität:** mittel
- **Skriptum:** -

Aufgabenstellung: Datenbankartefakte

- Programmieren sie für Beispiel 2.2.2 die erforderlichen Datenbankconstraints[2].
- Überlegen Sie welche **Indexe** für welche Felder Sinn machen und programmieren Sie diese. Begründen Sie Ihre Wahl.
- Schreiben Sie anschliessen die folgenden Views.
- Überlegen Sie welche Felder in der View angegeben werden sollen.

Aufgabenstellung: Views

```
1  -- ------------------------------------------
2  -- Loesung: Datenbankartefakte programmieren
3  -- ------------------------------------------
4  -- Geben Sie fuer einen Studenten eine Liste mit
5  -- allen Vorlesungen an die er besucht.
6
7
8  -- Geben Sie fuer einen Studenten eine Liste mit
9  -- allen Pruefungen an die er abgelegt hat. Geben
10 -- Sie die Namen der Vorlesungen, Professoren und
11 -- die Note an
12
13
14 -- Geben Sie fuer einen Professor eine Liste aller
15 -- Vorlesungen an die er leitet
16
17
18 -- Geben Sie fuer einen Professor eine Liste aller
19 -- Forschungsschwerpunkte an die er erforscht
20
21
22
23
24 .
```

❑

[2] *Schreiben Sie alle erforderlichen Constraints von Hand, ausgenommen der Schlüssel und Fremdschlüsselrelationen*

2.Beispiel - Datenbankartefakte

Beispielbeschreibung

- **Schwerpunkt:** Constraints, Index, View
- **Komplexität:** mittel
- **Skriptum:** -

Aufgabenstellung: Datenbankartefakte

- Programmieren sie für Beispiel 2.2.3 die erforderlichen Datenbankconstraints[3].
- Überlegen Sie welche **Indexe** für welche Felder Sinn machen und programmieren Sie diese. Begründen Sie Ihre Wahl.
- Schreiben Sie anschliessen die folgenden Views.
- Überlegen Sie welche Felder in der View angegeben werden sollen.

Aufgabenstellung: Views

```
1  -- ------------------------------------------
2  -- Loesung: Datenbankartefakte programmieren
3  -- ------------------------------------------
4  -- Geben Sie fuer eine bestimmte Liftkarte
5  -- alle Lifte an die der Fahrgast besucht hat
6
7  -- Geben sie fuer jede Schranke eines bestimmten
8  -- Liftes die Gesamtanzahl von Fahrten an.
9  -- Zusaetzlich soll die Anzahl unterschiedlicher
10 -- Gaeste angegben werden.
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25 .
```

❑

[3] *Schreiben Sie alle erforderlichen Constraints von Hand, ausgenommen der Schlüssel und Fremdschlüsselrelationen*

3.Beispiel - Userverwaltung

Beispielbeschreibung

- **Schwerpunkt:** User, Privileges
- **Komplexität:** einfach
- **Skriptum:** -

Aufgabenstellung: Userverwaltung

- Programmieren sie für das Beispiel 2.3.1 eine **Userverwaltung**.
- Legen Sie für die folgenden Benutzer die richtigen Privilegien fest.

Aufgabenstellung: Views

```
1  -- ------------------------------------------
2  -- Loesung: Userverwaltung
3  -- ------------------------------------------
4  -- User: university-admin  Psw: admin-l393n6
5
6  -- der university-admin soll fuer die university
7  -- datenbank alle moeglichen Privilegien besitzen
8
9
10 -- User: uniapp-client  Psw: appclient-h390
11
12 -- der uniapp-client soll in der university
13 -- Datenbank das Recht haben in alle Tabellen
14 -- Daten zu schreiben, zu lesen bzw. zu loeschen
15
16
17 -- User: uni-client  Psw: client-sjw
18
19 -- Der uni-client Benutzer darf nur die Views des
20 -- Systems aufrufen. uni-client darf sonst keine
21 -- Rechte haben.
22
23
24
25
26
27
28
29
30
31
32
33
34 .
```

❑

3. NoSQL - Informationssysteme

01

SQL Grundlagen

3.1. NoSQL Informationssysteme

1.Beispiel - Sql vs. NoSql System

Beispielbeschreibung

- **Schwerpunkt:** Sql Informationssysteme, NoSQL Informationssysteme
- **Komplexität:** mittel
- **Skriptum:** Seite 166 - 169

Aufgabenstellung: Thorieklausur

- Geben Sie 3 konkrete **Beispielanwendungen** für SQL Datenbanken an. Argumentieren Sie warum Sie für die angegebenen Szenarien ein SQL System einsetzen würden.
- Geben Sie nun 3 konkrete **Beispielanwendungen** für NoSQL Datenbanken an. Argumentieren Sie warum Sie für die angegbenen Szenarien ein NoSQL System einsetzen würden.

2.Beispiel - CAP Theorem

Beispielbeschreibung

- **Schwerpunkt:** CAP Theorem
- **Komplexität:** mittel
- **Skriptum:** Seite 170 - 173

Aufgabenstellung: Thorieklausur

- Geben Sie jeweils ein Beispiel für ein CP, AP und CA System. Erklären Sie warum die jeweiligen Informationssystem der entsprechenden Kategorie zugeordnet werden.
- Verwenden Sie keine Beispiele aus dem Vorlesungsskriptum.

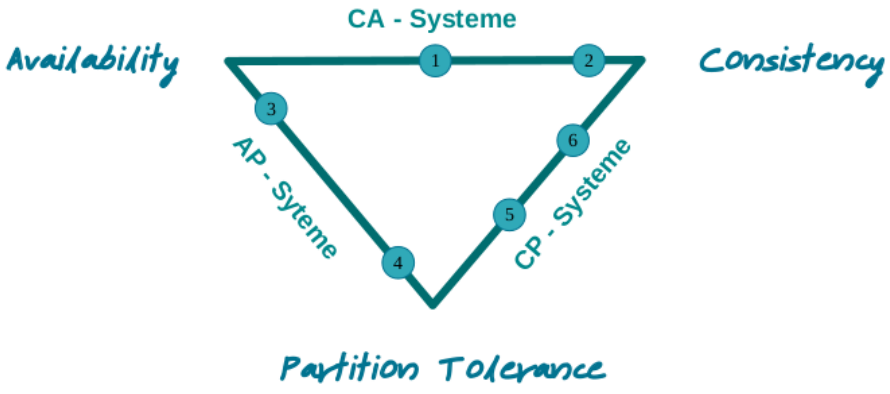


Abbildung 3. **Informationssysteme**

3.Beispiel - CAP Theorem

Beispielbeschreibung

- **Schwerpunkt:** CAP Theorem
- **Komplexität:** komplex
- **Skriptum:** Seite 170 - 173

Aufgabenstellung: Thorieklausur

- Beschreiben Sie die Eigenschaften jedes der in der CAP Theorem Grafik eingezeichneten Informationssystems.
- Finden Sie für jedes eingezeichnete Informationssystem eine konkrete Ausprägung.

> Zur einfacheren Recherche wird eine Liste von einschlägigen Informationssystemen angegeben.

Auflistung: Informationssysteme

- MySQL, Postgre SQL, Oracle Database
- Apache Cassandra, Google Big Table, MongoDB
- Redis, Apache Spark, Apache Hadoop
- Apache HBase, Apache Pig, Couchbase Server
- existDB

3.2. Konsistenzmodelle

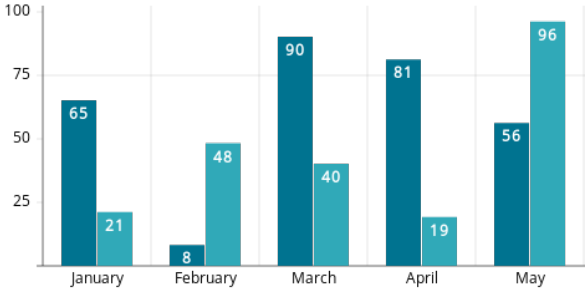
1.Beispiel - Transaktion

Beispielbeschreibung

- **Schwerpunkt:** Transaktion
- **Komplexität:** einfache
- **Skriptum:** Seite 174 - 175

Aufgabenstellung: Thorieklausur

- Geben Sie ein Beispiel aus dem Alltag bei dem in Informationssystemen Transaktionen verwendet werden.
- Erklären Sie warum in dem von IHnen beschriebenem Anwendungsfall das Fehlen eines Transaktionsmodels zu einem Fehler führen würde.
- Zeichnen Sie ein Anwendungsfalldiagramm für Ihre Szenario.



2.Beispiel - Konsistenzmodelle

Beispielbeschreibung

- **Schwerpunkt:** Konsistenzmodelle
- **Komplexität:** mittel
- **Skriptum:** Seite 174

Aufgabenstellung: Thorieklausur

- Finden Sie jeweils 3 Anwendungsfälle in denen **Eventual Consistency** für die Verarbeitung von Daten in einer Anwendung ausreichen ist.
- Finden Sie jeweils 3 Anwendungsfälle in denen für die Verarbeitung von Daten in einer Anwending **Strict Consistency** zwingend notwendig ist.

3.Beispiel - Sperren

Beispielbeschreibung

- **Schwerpunkt:** Konsistenzmodelle
- **Komplexität:** mittel
- **Skriptum:** Seite 174

Aufgabenstellung: Thorieklausur

- Für die Implementierung des **Isolationskriteriums** werden in Informationssystemen Sperren eingesetzt. Überlegen Sie den effizienten Einsatz von Sperren bei Schreib- bzw. Lesezugriffen auf Daten in der Datenbank.
- Geben Sie eine graphische Darstellung Ihrer Überlegungen.

4.Beispiel - BASE Transaktionen

Beispielbeschreibung

- **Schwerpunkt:** Konsistenzmodelle
- **Komplexität:** einfach
- **Skriptum:** Seite 176

Aufgabenstellung: Thorieklausur

- Erklären Sie den Unterschied zwischen der Eigenschaft des **Soft States** und des **Eventually Consistent**.

5.Beispiel - MVCC Verfahren

Beispielbeschreibung

- **Schwerpunkt:** Konsistenzmodelle
- **Komplexität:** einfach
- **Skriptum:** Seite 176 - 177

Aufgabenstellung: Thorieklausur

- Geben Sie eine graphische Darstellung des **MVCC Verfahrens**.
- Stellen Sie die Interaktion zwischen folgenden Entitäten dar:
  - Der zu verändernde Datensatz im Speicher.
  - Der zu verändernde Datensatz in der Datenbank.
  - Die Anwendung die den Datensatz ändert.
  - Die Datenbank die den Datensatz speichert.

.

.

4. MongoDB

01

Mongo

4.1. Modellierung

1.Beispiel) Modellierung

Beispielbeschreibung

- **Schwerpunkt:** Dokumentorientierte Modellierung, MongoDB
- **Komplexität:** mittel
- **Skriptum:** Seite 182 - 192

Aufgabenstellung: Projektmodellierung

- Entwickeln Sie für die folgenden Entitäten die Collections für MongoDB.
- Definieren Sie für die **Collections** jeweils ein **Schema**.
- Beachten Sie bei der **Modellierung** die Prinzipien der Dokumentorientierten Modellierung.

Codebeispiel: Entitätenbeschreibung

```
1  //------------------------------------------------
2  //  Project
3  //------------------------------------------------
4  @Data
5  @Table(name="projects")
6  public class Project implements Serializable{
7
8     @Size(min=3, max = 50)
9     private String title;
10
11    private EProjectType type;
12
13    private EProjectState state;
14
15    @Size(min=0, max=4000)
16    private String description;
17
18    private Date projectBegin;
19
20    private Boolean isFWFSponsered;
21
22    private Boolean isFFGSponsered;
23
24    private Boolean isEUSponsered;
25
26    private Boolean isSmallProject;
27
28 }
29
30 public enum EProjectType{
31     REQUEST_FUNDING_PROJECT,
32     RESEARCH_FUNDING_PROJECT,
33     MANAGEMENT_PROJECT
34 }
```

```java
//-----------------------------------------------
//  Subproject
//-----------------------------------------------
@Data
@Table(name="subprojects")
public class Subproject implements Serializable{

    @Size(min=3, max=100)
    private String title;

    @Size(min=0, max=4000)
    private String description;

    @Min(0)
    @Max(100)
    private Integer appliedResearch;

    @Min(0)
    @Max(100)
    private Integer theoreticalResearch;

    @Min(0)
    @Max(100)
    private Integer focusResearch;

}

//-----------------------------------------------
//  Facility
//-----------------------------------------------
@Data
@Table(name="facilities")
public class Facility implements Serializable{

    @Size(min=3, max=100)
    private String name;

    private String code;

}
//-----------------------------------------------
//  Debitor
//-----------------------------------------------
@Data
@Table(name="debitors")
public class Debitor implements Serializable{

   @Size(min=5, max=100)
   private String name;

   private String description;

}

//-----------------------------------------------
//  EProjectState
//-----------------------------------------------
public enum EProjectState {
    CREATED,
    IN_APPROVEMENT,
    APPROVED
}

//-----------------------------------------------
//  Bedingungen
//-----------------------------------------------
1.)Ein Project Dokument soll auch
Informationen uerber die Subprojekte enthalten
aus denen es besteht.

2.) In einem Project sollen gleichzeitig die
Daten der Geldgeber zusammen mit dem Geldbetrag
vermerkt werden den der entsprechende Geldgeber
ueberwiesen hat.

3.) In einem Subproject Dokument sollen auch
Daten bezueglich des umgebenden Projects enthalten
sein.

4.) Gleichzeitig sollen im Subproject Dokument
Daten zu dem Institut (Facility) gespeichert
werden an dem das Subprojekt durchgefuehrt
wird.

5.) Fuer ein Institut muessen auch die Projekte
gespeichert werden an denen das Institut arbeitet.

6.) Fuer Geldgeber wird gespeichert welche Projekte
Sie finanzieren zusammen mit dem Betrag und dem
Tag der Ueberweisung.
```

❑

2.Beispiel) Dokumente einfügen

Beispielbeschreibung

- **Schwerpunkt:** insertOne, insertMany
- **Komplexität:** einfach
- **Skriptum:** 195

Aufgabenstellung: Dokumente einfügen

- Fügen Sie mindestens 5 **Project** Dokumente in die **projects** Collection.
- Fügen Sie mindestens 5 **Subproject** Dokumente in die **subprojects** Collection.
- Fügen Sie mindestens 5 **Facility** Dokumente in die **facilities** Collection.
- Fügen Sie mindestens 5 **Debitor** Dokumente in die **debitors** Collection.

❑

3.Beispiel) Dokumente bearbeiten

Beispielbeschreibung

- **Schwerpunkt:** updateOne, updateMany
- **Komplexität:** einfach
- **Skriptum:** 195 - 199

Aufgabenstellung: Dokumente bearbeiten

- Bearbeiten Sie die in der Datenbank vorhandenen Dokumente.
- Stellen Sie sicher dass die Daten nach der Bearbeitung konsistent sind.

Codebeispiel: Dokumente bearbeiten

```
//------------------------------------------------
//  1.Aufgabe
//------------------------------------------------
1.) Fuegen Sie allen project Dokumenten die
folgenden Felder hinzu.

*)projectEnd date  defaultValue: Date('Jan 01,
    2020')
*)rating    int    defaultValue: 5
*)partners  array  item: {name: "TU Wien"}

//------------------------------------------------
//  2.Aufgabe
//------------------------------------------------
2.) Jedes REQUEST_FUNDING_PROJECT muss nun als
von der EU gefoerdert markiert werden.

//------------------------------------------------
//  3.Aufgabe
//------------------------------------------------
3.) Beim Anlegen der project Collection ist ein
Fehler unterlaufen. Benennen Sie das rating Feld
um in projectRating.

//------------------------------------------------
//  4.Aufgabe
//------------------------------------------------
4.) Fuegen Sie dem partners die folgenden Elemente
hinzu.

{name : "HTL Krems"},
{name : "TU Graz"}

Entfernen Sie anschliessend den HTL Krems Eintrag.
```

4.2. Abfragen

1.Beispiel) Query Kriterien

Beispielbeschreibung

- **Schwerpunkt:** Query Abfragen, Cursor Methoden
- **Komplexität:** mittel
- **Skriptum:** 264 - 270

Aufgabenstellung: MongoDB DQL

- Schreiben Sie die folgenden MongoDB Queries für die project Collections.

Codebeispiel: MongoDB Queries

```
//------------------------------------------------
//  1.Aufgabe
//------------------------------------------------
1.) Finden Sie alle project Dokumente in der
projects collection.

*) Geben Sie nur die ersten 5 Project Dokumente
aus.
*) Sortieren Sie die projecte nach dem titel
aufsteigend
*) Geben Sie fuer die project Dokumente jeweils
den Titel, den Type und den Projektzustand aus.

//------------------------------------------------sor
//  2.Aufgabe
//------------------------------------------------
2.) Finden Sie alle REQUEST_FUNDING_PROJECTe die
sich im Zustand APPROVED befinden.

*) Geben Sie nur die ersten 5 Project Dokumente
aus.
*) Sortieren Sie die projecte nach dem titel
aufsteigend
*) Geben Sie fuer die project Dokumente jeweils
den Titel, den Type und den Projektzustand aus.

//------------------------------------------------
//  3.Aufgabe
//------------------------------------------------
3.) Finden alle Subprojecte deren appliedResearch
oder theoreticalResearch oder focusResearch einen
Wert ueber 60 vorweisen.

*) Geben Sie nur die ersten 3 Subproject Dokumente
aus.
*) Sortieren Sie die subprojecte nach dem titel
aufsteigend
*) Geben Sie fuer die subproject Dokumente jeweils
den Titel und die Forschungsschwerpunkte aus.
```

2.Beispiel) Query Kriterien

Beispielbeschreibung

- **Schwerpunkt:** Query Abfragen, Cursor Methoden
- **Komplexität:** mittel
- **Skriptum:** 264 - 270

Aufgabenstellung: MongoDB DQL

- Schreiben Sie die folgenden MongoDB Queries für die project Collections.

Codebeispiel: MongoDB Queries

```
1  //---------------------------------------------
2  //  1.Aufgabe
3  //---------------------------------------------
4  1.) Finden Sie alle subproject Dokumente aus die
5  vom Institut fuer Softwareentwicklung, durchgefuert
6  werden.
7
8  *) Sortieren Sie die subprojecte nach dem titel
9
10 //---------------------------------------------
11 //  2.Aufgabe
12 //---------------------------------------------
13 2.) Finden Sie alle subproject Dokumente deren
14 Forschungsschwertpunkte in Summe 100 ergeben.
15
16 *) Sortieren Sie die subprojecte nach dem titel
17 aufsteigend.
18
19 //---------------------------------------------
20 //  3.Aufgabe
21 //---------------------------------------------
22 3.) Finden Sie alle debitoren die mehr als 2
23 Projekte finanzieren
24
25 *) Sortieren Sie die debitoren nach dem Namen
26
27 //---------------------------------------------
28 //  4.Aufgabe
29 //---------------------------------------------
30 4.) Finden Sie alle projecte die mit mehr als
31 700000 finanziert sind.
32
33 *) Sortieren Sie die projecte nach dem titel.
```

3.Beispiel) Query Kriterien

Beispielbeschreibung

- **Schwerpunkt:** Query Abfragen, Cursor Methoden
- **Komplexität:** mittel
- **Skriptum:** 195 - 270

Aufgabenstellung: MongoDB DQL

- Schreiben Sie die folgenden MongoDB Queries.

Codebeispiel: MongoDB Queries

```
1  //---------------------------------------------
2  //  1.Aufgabe
3  //---------------------------------------------
4  1.)Finden Sie alle Projekte die weder
5  REQUEST_FUNDING_PROJECTS noch
6  RESEARCH_FUNDING_PROJECTS
7  sind.
8
9  *) Fuer die Projekte soll nur der Titel und der
10    Projekttyp angegeben werden.
11 *) Sortieren Sie das Ergebnis nach dem titel
12    absteigend
13
14 //---------------------------------------------
15 //  2.Aufgabe
16 //---------------------------------------------
17 2.) Finden Sie alle Projekte die weder
18 REQUEST_FUNDING_PROJECTS noch
19 RESEARCH_FUNDING_PROJECTS
20 sind. Die Projekte muessen 1 Subprojekt haben.
21  Zustzlich muessen die Projekt die TU Wien als
22 Partner haben.
23
24 *) Fuer die Projekte soll nur der Titel und der
25    Projekttyp angegeben werden.
26 *) Sortieren Sie das Ergebnis nach dem titel
27    absteigend
28
29 //---------------------------------------------
30 //  3.Aufgabe
31 //---------------------------------------------
32 3.) Finden Sie alle Projekte die ein Rating
33 zwischen 3 und 5 haben. Die Projekte mussen
34 als Partner sowohl die TU Wien als auch die
35 TU Graz haben.
```

4.Beispiel) Query Kriterien

Beispielbeschreibung

- **Schwerpunkt:** Query Abfragen, Cursor Methoden
- **Komplexität:** komplex
- **Skriptum:** 195 - 270

Aufgabenstellung: MongoDB DQL

- Schreiben Sie die folgenden MongoDB Queries.

Codebeispiel: MongoDB Queries

```
//---------------------------------------------
//  1.Aufgabe
//---------------------------------------------
Fuegen Sie allen Subprojekten die Forschungspunkt
ueber 80 haben die Felder - marked : true - und
das Objekt - funding : {amount : NumberLong(10000)}
hinzu. Folgende Felder sollen folgende Felder haben

theoreticalResearch : 100
focusResearch : 0
appliedResearch : 0
//---------------------------------------------
//  2.Aufgabe
//---------------------------------------------
Fuer jedes Subprojekt soll ein project Dokument
angelegt werden, dass die _id des zugehoerigen
Projekts und seinen Titel enthaelt.


//---------------------------------------------
//  3.Aufgabe
//---------------------------------------------
Fuegen Sie jedem Subproject ein Objekt funding
hinzu. Das Objekt enthaelt ein Feld amount. Der
Wert von amount berechnet sich folgendermassen:

Berechnen Sie die Projektfoerderung in dem Sie
die einzelnen Foerderungen aufsummieren.

Verteilen Sie die Projektfoerderung nun
gleichermassen auf alle Subprojekte.
```

4.3. Aggregation von Daten

1.Beispiel) Aggregationsmethoden

Beispielbeschreibung

- **Schwerpunkt:** Aggregationsmethoden
- **Komplexität:** einfach
- **Skriptum:** 264 - 270

Aufgabenstellung: MongoDB DQL

- Schreiben Sie die folgenden MongoDB Queries.

Codebeispiel: MongoDB Queries

```
//---------------------------------------------
//  1.Aufgabe
//---------------------------------------------
1.Beispiel) Welche unterschiedlichen projectTypen
sind in den Dokumenten der project Collection
enthalten.

//---------------------------------------------
//  2.Aufgabe
//---------------------------------------------
2.Beispie) Geben Sie an von welchem Projekttyp
wieviele Dokumente gespeichert sind.

//---------------------------------------------
//  3.Aufgabe
//---------------------------------------------
3.Beispie) Geben Sie alle debitoren an die Projekte
finanziell unterstuetzen.
.
```

2.Beispiel) Aggregationspipeline

Beispielbeschreibung

- **Schwerpunkt:** Aggregationspipeline
- **Komplexität:** mittel
- **Skriptum:** 264 - 270

Aufgabenstellung: MongoDB DQL

- Verwenden Sie zur Lösung der folgenden Beispiele das Aggregationsframework.

Codebeispiel: MongoDB Queries

```
//------------------------------------------------
//  1.Aufgabe
//------------------------------------------------
1.Beispiel) Fuer alle REQUEST_FUNDING_PROJECTS
sollen folgende Aenderungen durchgefuehrt werden.

*)Fuegen Sie ein Feld projectFunding hinzu das
den gesamten Foerderungsbetrag speichert.

*)Fuegen Sie ein Feld subprojectCount hinzu, das
die Anzahl der Subprojekte speichert.

*)Das Feld subprojects sollen nur mehr die Titel
der Subprojekte speichern.

*)Die Projekte sollen folgende Felder beinhalten:
_id, title, projectFunding, subprojectCount
subprojects

*)Beruecksichtigen Sie nur Projekte die ein
Rating zwischen 2 und 5 haben. Das Projekt
darf kein Kleinprojekt sein.

*)Sortieren Sie die Projekte nach dem Titel
absteigend. Limitieren Sie die Ausgabe auf
maximal 5 Projekte.

*)Speichern Sie ihr Ergebnis in der Collection
projectreport

*) projekt Beispiel
{
  _id   : ...
  title : ...
  projectFunding   : ...
  subprojectCount : ...
  subprojects : [
     "title", "title"
  ]
}
```

```
//------------------------------------------------
//  2.Aufgabe
//------------------------------------------------
2.Beispiel) Fuegen Sie allen Subprojekten die
Forschungspunkt ueber 80 haben die Felder -
marked : true - und
das Objekt - funding : {amount : NumberLong(10000)}
hinzu.

*) Folgende Felder sollen mit den folgenden Werten
   initialisiert werden:

   theoreticalResearch : 100
   focusResearch : 0
   appliedResearch : 0

*) Beispiel:
   {
      _id: ...
      title : ..
      theoreticalResearch : 100,
      focusResearch     : 0,
      appliedResearch   : 0,
      marked : true,
      funding : {
         amount : NumberLong(10000)
      },
      project : {
         _id :  ...
         title : ...
      }
   }

*) Speichern Sie Ihr Ergebnis in einer Collection
   subprojectreport


//------------------------------------------------
//  3.Aufgabe
//------------------------------------------------
3.Beispiel) Fuer die Debitoren in der Datenbank
soll ein Report erstellt werden.

*) Geben Sie fuer jeden Debitor die folgenden
Daten an

{
   _id  : ...,
   name : ...,
   fundedProjects : ...,
   fundedAmount  : ...,
   projects : [
      "title1", "title2", ...
   ]
}

*) Achten Sie darauf dass die Titel im projects
   Array absteigend sortiert sein sollen
*) Speichern Sie Ihr Ergebnis in einer Collection
   debitorreport
```

```
//-----------------------------------------------
//  4.Aufgabe
//-----------------------------------------------
4.Beispiel) Fuer die Facilities in der Datenbank
soll ein Report erstellt werden.

*) Geben Sie fuer jede Facility die folgenden
Daten an

{
   _id : ... ,
   name : ...,
   code  : ...,
   isFWFSponsered : ...
   isFFGSponsered : ...
   isEUSponsered : ...
   isSmallProject : ...
   projectCount :
   subprojectscount : ...
}

*) Es duerfen nur jene Subprojekte angegeben
   werden die von der entsprechenden
   facility umgesetzt werden

*) Speichern Sie die Foerderungsstatistik in
   den Feldern isFWFSponsered, isFFGSponsered
   usw.
   Die Felder sollen die Anzahl der Projekte
   angegen die die entsprechende Foerderung
   bekommen.

*) Speichern Sie das Ergebnis in einer Collection
   facilityreport

//-----------------------------------------------
//  5.Aufgabe
//-----------------------------------------------
5.Beispiel) Finden Sie das Projekt mit der
hoechsten Foerderung.

*) Management Projekte sollen nicht beruecksichtigt
werden.
*) Geben Sie _id, title, projectType, projectState
und die Foerdersumme aus.

//-----------------------------------------------
//  6.Aufgabe
//-----------------------------------------------
6.Beispiel) Finden Sie den Debitor der die meisten
Projekte sponsert.

*) Geben Sie den namen des Debitors und die Anzahl
der Projekte die durch ihn gefrdert werden aus.
Es soll ebenfalls die Titel der Projekte in einem
Array gespeichert werden.



.
```

❑