

*Databases*

# Exercise: SQL Basics

1. **Self-study**

 Read Chapters 3.1 – 3.3 in the book by Meier & Kaufmann (2016).

 Answer the following questions:

 What user groups are there and how do they interact with the database?

Folgende Rollen haben Bedürfnisse an relationalen Datenbanksprachen:

|  |  |
| --- | --- |
| Rolle | Arbeiten |
| Datenarchitekt/in | Struktur einheitlich beschreiben durch Tabellen und Formate. |
| Datenbankspezialist/in | Die beschriebenen Tabellen installieren, kontrollieren, überwachen und sicherstellen. Benutzerrechte vergeben. |
| Anwendungsprogrammierer/in | Tabellen abfragen, verändern oder löschen. Die Verbindung zur Softwareentwicklung. |
| Datenanalyst/in | Tabellen abfragen, auswerten oder löschen.  Brauchen die Sprache für ihre Auswertungen. |

 What is the difference between set-oriented operators and relational operators?

|  |  |  |
| --- | --- | --- |
| Set-oriented (mengenorientierte) | **Differenz** | Relational (relationenorienteren) |
| Müssen vereinigungsverträglich sein (gleiche Anzahl Merkmale und selbe Datenformate) | **vereinigungsverträglich** | Müssen NICHT vereinigungsverträglich sein. |
| Vereinigung, Durchschnitt, Differenz und *kartesisches Produkt* | **Operationen** | Projektion, Selektion, Verbund und Division von Relationen |

 What is the connection between set-oriented query languages and relational algebra?

Query Languages machen die Operatoren der Relationsalgebra einfacher verwendbar.

 How is the *selection* applied in SQL?

Mit WHERE

Bespiel:

*SELECT \*   
FROM EMPLOYEE   
WHERE City = 'Kent' AND Sub='D6';*

 How is the *projection* applied in SQL?

Mit SELECT x, y , z ….

Beispiel:

*SELECT City, Sub   
FROM EMPLOYEE   
WHERE City = 'Kent';*

 How is the *join* applied in SQL?

Ein inner join wird mit FROM und WHERE ermöglicht

Bespiel:

*SELECT E#, Name, Sub, D#*

*FROM EMPLOYEE, DEPARTMENT*

*WHERE Sub = D#;*

*// Oder*

*SELECT e.E#, e.Name, e.Sub, d.D#*

*FROM EMPLOYEE AS e*

*JOIN DEPARTMENT AS d ON e.Sub = d.D#;*

 How do you know that the property of SQL is *descriptive*?

Die Sprache SQL ist deskriptiv, denn die Ausdrücke beschreiben das gewünschte Resultat, und nicht die dafür erforderlichen Rechenschritte. Dies macht SQL sehr leserlich und verständlich im Vergleich zu nicht deskriptiven Sprachen.

Beispiel Deskriptive: «Selektiere (SELECT) das Merkmal Name aus (FROM) der Tabelle MITARBEITER, wobei (WHERE) der Wohnort Liestal ist!»

🡺

*SELECT Name FROM Mitarbeiter WHERE Wohnort = ‘Liestal’*

 What does the statement “SQL is *relationally complete”* mean?

SQL Unterstützt die mengenorienterten Operatioren Vereinigung, Differenz und karesisches Produkt sowie die relationenorienterten Operatoren Projektion und Selektion.

 What does “grouping an aggregation with GROUP BY” mean?

Gruppiert alles mit dem selbem Wert bei einem oder mehreren gewählten Merkmalen.

Beispiel:

*SELECT COUNT(E#), Wohnort FROM Mitarbeiter GROUP BY Wohnort*

* Gibt anzahl mitarbeiter pro wohnort.

# Research literature

 Read the article *SEQUEL: A Structured English Query Language* by D. Chamberlin.

 You will find it on ILIAS: 03 sequel-1974.pdf

 What was the underlying idea of SEQUEL? (see Abstract)

Sprache mit englischen Keywords die sowohl für den professionellen Programmierer als auch für den selteneren Datenbankbenutzer gedacht ist.

 What were the two reasons for introducing declarative languages? (see Introduction)

1. Vereinfachung von programmierung
2. Für “non-professionals”.

 What is the main difference between SQUARE and SEQUEL? (page 253)

Sind gleich mächtig, SQUARE verwendet jedoch mathematische Notationen.

 What are some differences between the original SEQUEL and today's SQL?

Grundsätzlich nicht.

# SQL Workbench

 Go to the homepage of the book by Meier & Kaufmann: [www.sql-nosql.org](http://www.sql-nosql.org/)

 Look at the data model: https://sql-nosql.org/de/sql-tutorial

 Log on to the MySQL Workbench: https://sql-nosql.org/workbench/

o Any user, no password

 Do tasks 1.1 - 1.15 and 2.1 - 2.12.

1.1

SELECT \* FROM `movies`;

1.2

SELECT username FROM `user`;

1.3

22

SELECT COUNT(name)

FROM `category`;

1.4

7292

SELECT COUNT(DISTINCT lastName) FROM `crew`;

1.5

Yes

SELECT title FROM `movies` WHERE title like 'a beautiful mind';

1.6

SELECT name FROM ‘award’

ORDER BY name DESC

1.7

3

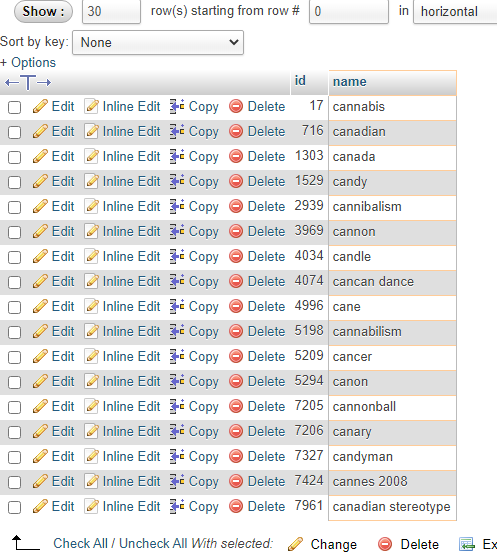
SELECT title, budget FROM `movies` WHERE budget > 280000000

1.8

ChuckNorris70

SELECT \* FROM `user` WHERE username LIKE '%norris%'

1.9

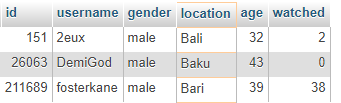


17

SELECT \* FROM `keywords` WHERE name LIKE 'can%';

1.10

SELECT \* FROM `user` WHERE location LIKE 'Ba\_\_'



1.11

SELECT \* FROM `user` WHERE age < 12 AND watched > 800

124

Alter kann 0 sein!

1.12

SELECT \* FROM `user` WHERE age BETWEEN 1 AND 11 AND watched > 800

1.13

SELECT \* FROM `user` WHERE age > 0 AND age < 12 AND watched > 800

1.14

SELECT COUNT( imdbRating) AS Anzahl FROM `movies` WHERE imdbRating = 7 OR imdbRating = 8

1’691 titles

1.15

SELECT title, imdbRating, year FROM `movies` WHERE (imdbRating > 8 OR metascore > 80) AND year > 2012

20

2.1

SELECT title, code FROM `playsInCountry`, `country`, `movies` WHERE c\_id = country.id AND m\_id = movies.id

2.2

SELECT title, firstName, lastName FROM `movies` , `crew` WHERE f\_id = movies.id

2.3

SELECT title, firstName, lastName

FROM `movies` , `crew`

WHERE f\_id = movies.id

ORDER BY title

2.4

SELECT title, username

FROM `hasWatched` , `user` , `movies`

WHERE user.username LIKE 'SwissMarco'

AND user.id = u\_id

AND m\_id = movies.id

2.5

Nach definition von “Success” abhängig.

//Mit check ob gewonnen

SELECT title, rank

FROM `hasRank` , `movies` , `awardRank`

WHERE m\_id = movies.id

AND r\_id = awardRank.id

AND awardRank.id =1

ORDER BY rank

LIMIT 5

//ohni gewonnen check

SELECT title, rank

FROM `hasRank` , `movies`

WHERE m\_id = movies.id

ORDER BY rank

LIMIT 5

2.6

SELECT DISTINCT title , movies.year

FROM `movies`, `keywords`, `hasKeyword`

WHERE

k\_id = 19 AND m\_id = movies.id AND movies.year > 2000

2.7

SELECT AVG( movies.year ) , category.name

FROM `movies` , `category` , `hasCategory`

WHERE c\_id = category.id

AND m\_id = movies.id

GROUP BY category.name

2.8

19122600 sekunden

SELECT SUM(movies.duration)

FROM `movies`

2.9

SELECT title AS Film , category.name as Kategorie

FROM `movies` , `category` , `hasCategory`

WHERE category.name LIKE 'Horror'

AND c\_id = category.id

AND m\_id = movies.id

2.10

SELECT name

FROM `featureCategory`

UNION

SELECT name

FROM `category`

2.11

SELECT title AS Film , category.name as Kategorie, imdbRating as Rating

FROM `movies` , `category` , `hasCategory`

WHERE (category.name LIKE 'Action' OR category.name LIKE 'Comedy' )

AND c\_id = category.id

AND m\_id = movies.id

AND imdbRating > 8;

2.12

SELECT DISTINCT title

FROM `award` , `movies` , `awardRank` , `hasAward`

WHERE hasAward.m\_id = movies.id

AND hasAward.a\_id = award.id

AND hasAward.ar\_id = awardRank.id

AND awardRank.id <3

AND award.name LIKE 'Golden Globes'

# Relevance to the project

Siehe Demo.