Hochschule Düsseldorf
University of Applied Sciences
Fachbereich Medien
Faculty of Media

Hochschule Düsseldorf
Fachbereich Medien
Prof. Dr.-Ing. Thomas C. Rakow
Professor für Informatik, insbesondere
Datenbanken und E-Business

thomas.rakow@hs-duesseldorf.de
dbe.medien.hs-duesseldorf.de
https://moodle.hs-duesseldorf.de/

Empfohlene Namenskonventionen und Codierungsvorgaben für SQL im Praktikum Datenbanksysteme

Autoren: Jens Lambert, Björn Salgert, Mareike Focken und Thomas C. Rakow

1. Namenskonventionen für SQL

Gemeinsame Namenskonventionen sind sinnvoll, um die Lesbarkeit von Code zu verbessern, da jeder Code häufiger gelesen als geschrieben wird. Die einheitliche Gestaltung ermöglicht es, Fehler schneller zu finden und in Entwickler-Teams leichter miteinander zu kommunizieren.

(1) Wählen Sie immer aussagekräftige und spezifische Bezeichner.

(2) Verwenden Sie immer eine gesprochene Sprache (z.B. Englisch, Deutsch, Französisch) für die Bezeichner in Ihrer Anwendung. Halten Sie sich hierbei an die Sprache für die Bezeichner in der Aufgabenstellung.

(3) Für eine bessere Lesbarkeit verwenden Sie für Bezeichner[1] durchgängig die Kleinschreibung. Bei Bezeichnern aus mehreren Worten trennen Sie diese mit Unterstrich.

Beispiel: `salary_euro`

(4) Schreiben Sie Bezeichner von Attributen immer im Singular und Bezeichner von Tabellen immer im Plural.

Beispiel: Attribut: `name`, Tabelle: `employees`

(5) Verwenden Sie immer die gleichen Bezeichner für Elemente mit der gleichen Bedeutung.

Beispiel: : Attribut: `user` oder Attribut: `operator`, aber nicht beides verwenden.

(6) Schreiben Sie ORACLE-Schlüsselwörter immer in Großbuchstaben.

(7) Verwenden Sie niemals ORACLE-Schlüsselwörter als Bezeichner. Eine Liste der ORACLE-Schlüsselwörter wird in Oracle SQL Reserved Words und Oracle PL/SQL Reserved Words oder in der Wörterbuchansicht der verwendeten Datenbank Version mit `v$reserved_words` angezeigt.

(8) Verwenden Sie Abkürzungen von Bezeichnern, um übermäßig lange Namen abzukürzen. Alle Abkürzungen müssen in einem Glossar dokumentiert werden oder zumindest allgemein bekannt sein.

Beispiel: Bahnhof: `bhf`; `course#` mit `#` als Nummerangabe

[1] Im Allgemeinen unterscheidet ORACLE bei Namen nicht zwischen Groß- und Kleinschreibung. Eine Variable mit dem Namen `personname` ist gleichbedeutend mit einer Variablen mit dem Namen `PersonName` sowie mit einer Variable mit dem Namen `PERSONNAME`. Bei Namen in doppelten Anführungszeichen (") wird zwischen Groß- und Kleinschreibung unterschieden.

EILD.nrw OER for Databases

Ein Kooperationsvorhaben empfohlen durch die: DIGITALE HOCHSCHULE NRW
Gefördert durch: Ministerium für Kultur und Wissenschaft des Landes Nordrhein-Westfalen

(9) Verwenden Sie für die Präzisierung von Bezeichnern das Muster {Präfix_}Bezeichnerinhalt{_Suffix} (siehe Tabelle unten).

(10) Die folgende Tabelle zeigt den von uns verwendeten Satz von Bezeichnern mit ihren empfohlenen Pre- und Suffixen.

Bezeichnung	**Prefix**	**Suffix**	**Beispiel**
SQL			
Tabellenbezeichner	-	-	`employees` (Plural)
Tabellenbezeichner für 1:n-, n-m-Tabellen	-	-	`employees_department`
Attributbezeichner	-	-	`job, name` (Singular)
Attributbezeichner bei mehrwertigen Typen (TYPE)	-	-	`jobs, names` (Plural)
Einwertiger Schlüssel	`<table>_`	`id`	`employee_id` mit Tabellenname im Singular
Fremdschlüssel	-	-	`employee_id` bei einwertigem Primärschlüssel gleicher Name
Primärschlüssel-Constraint	`<table>_`	`_pk`	`employees_pk` als Constraint-Bezeichner
Fremdschlüssel-Constraint	`<table>_`	`_<ref_table>_fk`	`dept_emp_fk` als Constraint-Bezeichner
Check-Constraint	`<table>_`	`_<attribute>_ck`	`employees_job_ck` als Constraint-Bezeichner
Unique-Constraint	`<table>_`	`_<attribute>_uk`	`employees_name_uk` als Constraint-Bezeichner
Index	-	`_idx`	`name_idx`
Sequence	-	`_seq`	`employees_seq`

Ein Kooperationsvorhaben empfohlen durch die:
DIGITALE HOCHSCHULE NRW
Gefördert durch:
Ministerium für Kultur und Wissenschaft des Landes Nordrhein-Westfalen

2. Codierungsvorgaben für SQL

(1) Pro Zeile eine Anweisung bzw. Schlüsselwort, wenn es nicht mit weiteren Schlüsselworten eine Einheit bildet, wie es auch in objektorientierten Programiersprachen üblich ist.

(2) Leerzeilen werden eingefügt, um mehrere Anweisungen zu trennen.

(3) Werden Zeilen zu lang und somit unübersichtlich, werden sie umgebrochen. Die folgende Zeile wird eingerückt, um zu zeigen, dass sie noch zum Konstrukt in der Zeile davor gehört. Gleichrangige folgende (neue) Zeilen werden auf einer Einzugstiefe dargestellt.
Bei verschachtelten Konstrukten werden die inneren Konstrukte um eine weitere Ebene eingerückt.

(4) Innerhalb einer Anweisung werden nur Zeilenkommentare „--“ verwendet, damit die Lesbarkeit des Codezusammenhanges erhalten bleibt.

(5) Klammern werden verwendet, wenn sie benötigt werden oder wenn sie zur Verdeutlichung eines Konstrukts hilfreich sind. Dies hilft die Reihenfolge von Auswertungen übersichtlicher zu gestalten. Klammern werden auf der Ebene geschlossen, auf der die Zeile beginnt, in der sie geöffnet wurden.

Beispiel für SQL:

```
SELECT e1.ename, e1.hiredate,
    e1.location
FROM employees e1
WHERE e1.hiredate >= (                    -- Kommentar
    SELECT e2.hiredate
    FROM employees e2
    WHERE e2.ename LIKE 'BLAKE' AND (
         e2.location = 'DALLAS' OR
         e2.location = 'HOUSTON'
    )
)
ORDER BY e1.hiredate;

SELECT *
FROM departments;
```


Ein Kooperationsvorhaben empfohlen durch die:
DIGITALE HOCHSCHULE NRW
Gefördert durch:
Ministerium für Kultur und Wissenschaft des Landes Nordrhein-Westfalen