# **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]](#footnote-1) durchgängig die Kleinschreibung. Bei Bezeichnern aus mehreren Worten trennen Sie diese mit Unterstrich.

Beispiel: salary\_euro

1. Schreiben Sie Bezeichner von Attributen immer im Singular und Bezeichner von Tabellen immer im Plural.

Beispiel: Attribut: name, Tabelle: employees

1. Verwenden Sie immer die gleichen Bezeichner für Elemente mit der gleichen Bedeutung.

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

1. Schreiben Sie ORACLE-Schlüsselwörter immer in Großbuchstaben.
2. Verwenden Sie niemals ORACLE-Schlüsselwörter als Bezeichner. Eine Liste der ORACLE-Schlüsselwörter wird in [Oracle SQL Reserved Words](https://docs.oracle.com/en/database/oracle/oracle-database/21/sqlrf/Oracle-SQL-Reserved-Words.html#GUID-55C49D1E-BE08-4C50-A9DD-8593EB925612) und [Oracle PL/SQL Reserved Words](https://docs.oracle.com/en/database/oracle/oracle-database/21/lnpls/plsql-reserved-words-keywords.html) oder in der Wörterbuchansicht der verwendeten Datenbank Version mit v$reserved\_words angezeigt.
3. 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. Verwenden Sie für die Präzisierung von Bezeichnern das Muster
 {Präfix\_}Bezeichnerinhalt{\_Suffix} (siehe Tabelle unten).
2. 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\_idbei 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 |

**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;

**Empfohlene Namenskonventionen und Codierungsvorgaben für PL/SQL im
Praktikum Datenbanksysteme**

Fortsetzung von

“Empfohlene Namenskonventionen und Codierungsvorgaben für SQL im Praktikum Datenbanksysteme”

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

**3. Namenskonventionen für PL/SQL**

Grundsätzlich basieren die Namenskonventionen für PL/SQL auf denen von SQL (Teil 1)

Darüber hinaus gilt:

1. Schreiben Sie Bezeichner von Kollektionen immer im Plural.

Beispiel: Kollektion: hobbies

1. Bezeichner für Prozeduren beschreiben immer, was die Prozedur tut.

Beispiel: calculate\_salary

1. Bezeichner für Funktionen beschreiben immer das Ergebnis, das zurückgegeben wird (eine Funktion soll keine Seiteneffekte verursachen).

Beispiel: employee\_by\_id

|  |  |  |  |
| --- | --- | --- | --- |
| **Bezeichnung** | **Prefix** | **Suffix** | **Beispiel** |
| **PL/SQL** |  |  |  |
| Lokale Variable | l\_ | - | l\_name |
| Constante | co\_ | - | co\_pi |
| Cursor | c\_ | - | c\_employees |
| Record | r\_ | - | r\_employee |
| Table | t\_ | - | t\_employees |
| Parameter | p\_ | - | p\_empno |
| IN Parameter | in\_ |  | in\_salary |
| OUT Parameter | out\_ |  | out\_salary |
| IN-OUT Parameter | io\_ |  | io\_salary |
| Exception | e\_ | - | e\_employee\_exists |
| Trigger | <table>\_ | \_trg | emp\_salary\_trg |

**4. Codierungsvorgaben für PL/SQL**

Grundsätzlich basieren die Codierungsvorgaben für PL/SQL auf denen von SQL (Teil 2)

Beispiel für PL/SQL:

PROCEDURE set\_salary(in\_employee\_id IN employees.employee\_id%TYPE) IS

 CURSOR c\_employees(p\_employee\_id IN employees.employee\_id%TYPE) IS

 SELECT last\_name, first\_name,

 salary

 FROM employees

 WHERE employee\_id = p\_employee\_id

 ORDER BY last\_name, first\_name;

 r\_employee c\_employees%ROWTYPE;

 l\_new\_salary employees.salary%TYPE;

BEGIN

 OPEN c\_employees(p\_employee\_id => in\_employee\_id);

 FETCH c\_employees INTO r\_employee;

 CLOSE c\_employees;

 new\_salary (

 in\_employee\_id => in\_employee\_id,

 out\_salary => l\_new\_salary

 );

 -- Check whether salary has changed

 IF r\_employee.salary <> l\_new\_salary THEN

 UPDATE employees

 SET salary = l\_new\_salary

 WHERE employee\_id = in\_employee\_id;

 END IF;

END set\_salary;

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. [↑](#footnote-ref-1)