SQL für Anfänger

SELECT-Abfragen verstehen und formulieren mit Oracle SQL und SQL Server

```
1  SELECT e.employee_id,
2         e.first_name,
3         e.last_name,
4         d.department_name
5  FROM employees e
6  JOIN departments d
7    ON e.department_id = d.department_id
8  WHERE e.salary > 3000;
9  ORDER BY e.last_name;
```

```
1  SELECT d.department_name,
2       COUNT(*) AS mitarbeiteranzahl
3  FROM employees e
4  JOIN departments d
5    ON e.department_id = d.department_id
6  GROUP BY d.department_name
7  HAVING COUNT(*) > 5
8  ORDER BY mitarbeiteranzahl DESC;
```

Hartmut Kruger

SQL für Anfänger

SELECT-Abfragen verstehen und formulieren
mit Oracle SQL und SQL Server

Hartmut Kruger

Impressum

Bibliografische Information der Deutschen Nationalbibliothek: Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über dnb.dnb.de abrufbar.

Herstellung und Verlag:
BoD · Books on Demand GmbH, Überseering 33, 22297 Hamburg

ISBN: 978-3-6963-3019-4

Die richtige Frage macht aus Daten eine Antwort.

SQL hilft, diese Frage präzise zu formulieren.

Vorwort

SQL ist eine der wichtigsten Sprachen für die Arbeit mit Daten. Wer Daten nicht nur ansehen, sondern gezielt auswerten möchte, braucht ein Verständnis für Tabellen, Spalten, Filter, Verbindungen und Gruppierungen.

Dieses Buch richtet sich an Einsteigerinnen und Einsteiger. Es konzentriert sich bewusst auf SELECT-Abfragen, weil diese in der täglichen Auswertung die wichtigste Rolle spielen. Später werden auch INSERT, UPDATE und DELETE erklärt, aber immer mit besonderem Blick auf Sicherheit und Kontrolle.

Die Beispiele verwenden eine kleine kaufmännische Beispieldatenbank mit Kunden, Artikeln, Aufträgen, Rechnungen und Zahlungen. Dadurch lassen sich typische Fragen aus der Praxis nachvollziehbar üben: Welche Kunden haben Umsatz? Welche Rechnungen sind offen? Welche Artikel wurden verkauft? Welche Daten fehlen?

Die Beispiele sind so formuliert, dass sie die Unterschiede zwischen Oracle SQL und SQL Server sichtbar machen, ohne den Einstieg unnötig zu erschweren.

Hinweise zur Arbeit mit diesem Buch

- SQL-Schlüsselwörter werden in Großbuchstaben geschrieben, damit die Struktur der Abfrage gut sichtbar ist.
- Tabellen- und Spaltennamen werden klein geschrieben, zum Beispiel kunden, rechnungen und bruttobetrag.
- Jedes Kapitel erklärt die SQL-Bausteine, die in seinen Beispielen vorkommen.
- Wenn ein Beispiel Oracle SQL und SQL Server unterschiedlich betrifft, werden beide Schreibweisen gezeigt.
- Die Beispieldatenbank kann mit den Skripten in Anhang D und Anhang E aufgebaut werden.
- Vor UPDATE und DELETE sollte immer eine SELECT-Prüfung erfolgen.

Begleitdateien zum Buch

Zu diesem Buch stehen Begleitskripte zur Beispieldatenbank bereit. Über den folgenden Link oder den QR-Code können Sie die SQL-Skripte für SQL Server und Oracle herunterladen.

Downloadseite:

https://hartmut-kruger.de/de/sql/

Hinweis:

Bitte führen Sie die SQL-Skripte nur in einer geeigneten Testumgebung aus und prüfen Sie sie vor der Ausführung. Die Skripte dienen zur Erstellung der Beispieldatenbank

QR-Code zur Downloadseite

Inhaltsverzeichnis

Teil I – Grundlagen von SQL und Datenbanken

1 Was ist SQL?

SQL ist die Sprache, mit der relationale Datenbanken abgefragt werden. Dieses Kapitel führt SQL als Lesesprache ein und beschränkt sich bewusst auf SELECT und FROM.

Lernziele

- verstehen, dass SQL Fragen an Tabellen formuliert
- SELECT als Auswahl der anzuzeigenden Spalten erkennen
- FROM als Angabe der Tabelle verstehen
- erkennen, dass eine Abfrage ein Ergebnis in Zeilen und Spalten liefert

Fachlicher Einstieg

Am Anfang geht es nicht um schwierige Technik, sondern um eine einfache Idee: In einer Tabelle liegen Daten, und SQL beschreibt, welche davon angezeigt werden sollen. Eine SELECT-Abfrage verändert keine Daten. Sie liest Daten und stellt sie als Ergebnis dar.

Grundmuster

```
SELECT
    spalte1,
    spalte2
FROM tabelle;
```

Erklärung des Grundmusters

SELECT nennt die Spalten, die im Ergebnis erscheinen sollen. FROM nennt die Tabelle, aus der diese Spalten gelesen werden. Mehr enthält dieses erste Grundmuster noch nicht: keine Sortierung, kein Filter, keine Verbindung zu anderen Tabellen.

Praxisbeispiele

Beispiel 1.1: Zwei Kundenspalten lesen

Die Abfrage zeigt Kundennummer und Kundenname aus der Tabelle kunden.

```
SELECT
    kundennr,
    kundenname
FROM kunden;
```

Das Ergebnis hat eine Zeile je Kunde, weil direkt aus der Kundentabelle gelesen wird.

Beispiel 1.2: Artikeldaten lesen

Dasselbe Muster kann auf eine andere Tabelle angewendet werden.

```
SELECT
    artikelnr,
    artikelname
FROM artikel;
```

Die Abfrage liest Stammdaten zu Artikeln. Auch hier wird noch nichts gefiltert oder berechnet.

Beispiel 1.3: Rechnungen betrachten

Eine Rechnungstabelle enthält Bewegungsdaten. Auch sie kann mit SELECT betrachtet werden.

```
SELECT
    rechnungsnr,
    kundennr,
    bruttobetrag
FROM rechnungen;
```

Die Spalte kundennr zeigt nur die Nummer des Kunden. Der Kundenname steht in einer anderen Tabelle und wird erst später mit JOIN erklärt.

Beispiel 1.4: Zahlungen ansehen

Auch Zahlungen sind normale Tabellenzeilen und können gelesen werden.

```
SELECT
    zahlung_id,
    rechnungsnr,
    betrag
FROM zahlungen;
```

Damit wird sichtbar, dass SQL nicht nur Stammdaten, sondern auch Vorgänge wie Zahlungen lesen kann.

Typische Fehler und Stolperstellen

- SELECT mit INSERT, UPDATE oder DELETE verwechseln: SELECT liest nur.
- Zu früh mehrere Tabellen verbinden wollen, bevor eine einzelne Tabelle verstanden ist.
- Erwarten, dass ohne ORDER BY eine fachlich garantierte Reihenfolge entsteht. Sortierung kommt später.

Zusammenfassung

Das erste SQL-Muster besteht aus SELECT und FROM. Damit kann eine Tabelle gelesen werden. Die Ergebnisebene entspricht der Tabelle, aus der gelesen wird: bei kunden eine Zeile je Kunde, bei rechnungen eine Zeile je Rechnung.

2 Tabellen, Zeilen, Spalten und Schlüssel

Daten in relationalen Datenbanken liegen in Tabellen. Dieses Kapitel erklärt, was Tabellen, Zeilen, Spalten und Schlüssel bedeuten.

Lernziele

- Tabellen als fachliche Sammlungen verstehen
- Zeilen als einzelne Datensätze erkennen
- Spalten als Eigenschaften eines Datensatzes verstehen
- Primärschlüssel als eindeutige Kennung erkennen
- Fremdschlüssel als Verweis auf eine andere Tabelle einordnen

Fachlicher Einstieg

Eine Tabelle ist wie eine geordnete Liste von gleichartigen Dingen. Die Tabelle kunden enthält Kunden. Die Tabelle rechnungen enthält Rechnungen. Jede Zeile beschreibt einen konkreten Datensatz, jede Spalte eine Eigenschaft dieses Datensatzes.

Grundmuster

```
SELECT
    schluesselspalte,
    beschreibungsspalte
FROM tabelle;
```

Erklärung des Grundmusters

Das Grundmuster zeigt bewusst Schlüsselspalte und Beschreibungsspalte zusammen. Die Schlüsselspalte identifiziert den Datensatz technisch eindeutig. Die Beschreibungsspalte macht ihn für Menschen lesbar.

Praxisbeispiele

Beispiel 2.1: Kunden mit Primärschlüssel ansehen

kundennr ist die eindeutige Nummer eines Kunden.

```
SELECT
    kundennr,
    kundenname
FROM kunden;
```

Die Kundennummer ist der technische Zugriffspunkt. Der Kundenname ist die fachliche Bezeichnung.

Beispiel 2.2: Rechnungen mit Fremdschlüssel ansehen

In rechnungen steht kundennr ebenfalls, aber hier ist sie ein Verweis auf kunden.

```
SELECT
    rechnungsnr,
    kundennr,
    bruttobetrag
FROM rechnungen;
```

rechnungsnr identifiziert die Rechnung. kundennr zeigt, zu welchem Kunden die Rechnung gehört.

Beispiel 2.3: Artikel mit Artikelschlüssel

Auch Artikel besitzen eine eindeutige Schlüsselspalte.

```
SELECT
    artikelnr,
    artikelname,
    kategorie
FROM artikel;
```

Die Artikelnr ist technisch eindeutig. Artikelname und Kategorie beschreiben den Artikel.

Beispiel 2.4: Zahlungen mit Bezug zur Rechnung

Zahlungen haben eine eigene ID und verweisen zusätzlich auf eine Rechnung.

```
SELECT
    zahlung_id,
    rechnungsnr,
    betrag
FROM zahlungen;
```

zahlung_id identifiziert die Zahlung. rechnungsnr zeigt, zu welcher Rechnung diese Zahlung gehört.

Typische Fehler und Stolperstellen

- Schlüsselnummern als fachliche Messwerte behandeln. Eine Kundennummer ist keine Zahl zum Addieren.
- Fremdschlüssel mit Primärschlüsseln verwechseln. Dieselbe Spalte kundennr hat je nach Tabelle eine andere Rolle.
- Namen als sichere Verbindung ansehen. Tabellen werden später über Schlüssel verbunden, nicht über Namen.

Zusammenfassung

Tabellen bestehen aus Zeilen und Spalten. Primärschlüssel identifizieren Zeilen eindeutig. Fremdschlüssel zeigen Beziehungen zu anderen Tabellen. Dieses Wissen ist die Grundlage für spätere JOINs.

3 Datenbanken, Tabellen und Beziehungen verstehen

Die Beispieldatenbank wird hier fachlich eingeordnet: Welche Tabellen gibt es, welche Aufgaben haben sie und warum sind Beziehungen zwischen Tabellen notwendig?

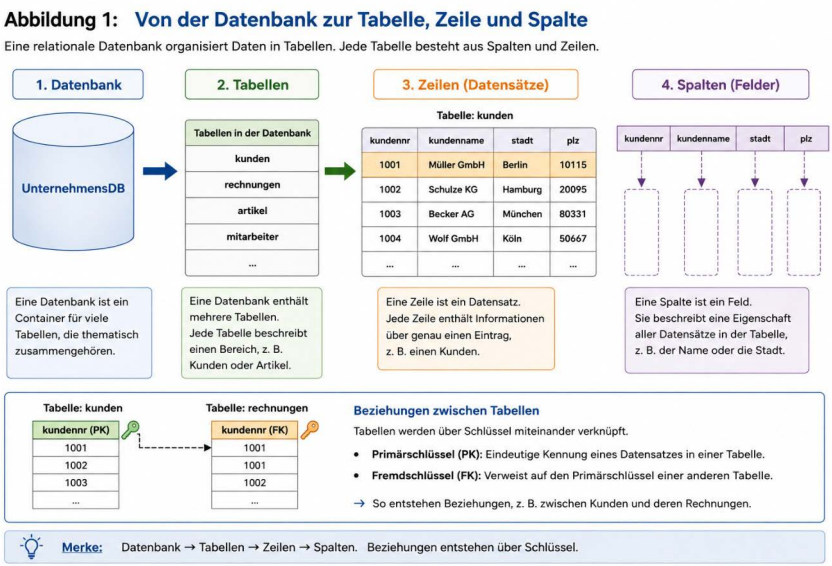

Abbildung 1: Von der Datenbank zur Tabelle, Zeile und Spalte

Lernziele

- Stammdaten und Bewegungsdaten unterscheiden
- Beziehungen zwischen Tabellen erkennen
- typische Beziehungswege der Beispieldatenbank lesen
- verstehen, warum Informationen auf mehrere Tabellen verteilt sind

Fachlicher Einstieg

In einer Datenbank steht nicht alles in einer einzigen großen Tabelle. Kundendaten, Rechnungen, Zahlungen und Artikel werden getrennt

gespeichert. Dadurch werden Wiederholungen reduziert und Beziehungen klarer.

Grundmuster

```
SELECT
    schluesselspalte,
    verweis_spalte,
    beschreibung
FROM tabelle;
```

Erklärung des Grundmusters

Das Muster zeigt eine Tabelle mit eigener Schlüsselspalte und möglicher Verweisspalte. Solche Verweisspalten sind die Grundlage, um Tabellen später sinnvoll zu verbinden.

Praxisbeispiele

Beispiel 3.1: Kunden als Stammdaten

Kunden sind Stammdaten. Sie beschreiben Geschäftspartner.

```
SELECT
    kundennr,
    kundenname,
    ort
FROM kunden;
```

Diese Daten ändern sich normalerweise seltener als Rechnungen oder Zahlungen.

Beispiel 3.2: Rechnungen als Bewegungsdaten

Rechnungen sind Vorgänge. Sie verweisen auf Kunden.

```
SELECT
    rechnungsnr,
    kundennr,
    rechnungsdatum,
    bruttobetrag
FROM rechnungen;
```

Die Beziehung zum Kunden ist über kundennr sichtbar.

Beispiel 3.3: Zahlungen als weitere Vorgänge

Zahlungen verweisen auf Rechnungen.

```
SELECT
    zahlung_id,
    rechnungsnr,
    zahlungsdatum,
    betrag
FROM zahlungen;
```

Damit lässt sich später prüfen, welche Rechnungen bezahlt wurden.

Beispiel 3.4: Artikel als Stammdaten

Artikel beschreiben Produkte oder Leistungen.

```
SELECT
    artikelnr,
    artikelname,
    kategorie
FROM artikel;
```

Bestellte Mengen stehen nicht in artikel, sondern später in auftragspositionen.

Typische Fehler und Stolperstellen

- Erwarten, dass jede gewünschte Information in einer einzigen Tabelle steht.
- Beziehungen nur über gleiche Namen vermuten, statt Schlüsselspalten zu prüfen.
- Stammdaten und Bewegungsdaten vermischen: Kunden beschreiben Personen oder Firmen, Rechnungen beschreiben Vorgänge.

Zusammenfassung

Die Beispieldatenbank enthält mehrere fachliche Bereiche. Kunden, Artikel, Mitarbeiter und Abteilungen sind Stammdaten. Aufträge, Positionen, Rechnungen und Zahlungen sind Bewegungsdaten. Beziehungen entstehen über Schlüsselspalten.

4 Oracle SQL und SQL Server als Arbeitsumgebung

Die Beispiele laufen grundsätzlich in Oracle SQL und SQL Server. An dieser Stelle werden die wichtigsten Unterschiede in Arbeitsumgebung, Datumswerten, Datentypen und Funktionen eingeordnet.

Lernziele

- Oracle SQL und SQL Server als zwei SQL-Dialekte einordnen
- gemeinsame SELECT-Grundstruktur erkennen
- Datumsschreibweisen unterscheiden
- wissen, warum manche Funktionen unterschiedlich heißen

Fachlicher Einstieg

Die SQL-Grundidee ist in Oracle SQL und SQL Server gleich. Unterschiede entstehen vor allem bei Datumswerten, Datentypen und einzelnen Funktionen. Deshalb zeigt das Buch wichtige Varianten getrennt.

Grundmuster

```
-- SQL Server
SELECT
    spalte
FROM tabelle;

-- Oracle SQL
SELECT
    spalte
FROM tabelle;
```

Erklärung des Grundmusters

Die einfache SELECT-Struktur ist gleich. Unterschiede werden erst wichtig, wenn Datumsliterale, aktuelle Datumsfunktionen, Textfunktionen oder spezielle Begrenzungen verwendet werden.

Praxisbeispiele

Beispiel 4.1: Gleiche SELECT-Abfrage in beiden Systemen

Diese Abfrage ist in Oracle SQL und SQL Server gleich.

```
SELECT
    kundennr,
    kundenname
FROM kunden;
```

SELECT und FROM gehören zum gemeinsamen SQL-Grundbestand.

Beispiel 4.2: SQL Server mit ISO-Datum

SQL Server kann ein ISO-Datum in Hochkommas verwenden.

```
SELECT
    rechnungsnr,
    rechnungsdatum
FROM rechnungen
WHERE rechnungsdatum = '2024-01-15';
```

WHERE wird erst später ausführlich erklärt. Hier geht es nur um die Datumsschreibweise in SQL Server.

Beispiel 4.3: Oracle SQL mit DATE-Literal

Oracle SQL verwendet für ein klares Datumsliteral DATE.

```
SELECT
    rechnungsnr,
    rechnungsdatum
FROM rechnungen
WHERE rechnungsdatum = DATE '2024-01-15';
```

Diese Schreibweise vermeidet Missverständnisse mit länderspezifischen Datumsformaten.

Beispiel 4.4: Unterschiedliche Datentypnamen

Beim Erstellen von Tabellen heißen Datentypen unterschiedlich.

```
-- Oracle SQL: VARCHAR2, NUMBER
-- SQL Server: NVARCHAR, INT, DECIMAL
```

Die fachliche Bedeutung kann gleich sein, obwohl der Datentypname unterschiedlich ist.

Typische Fehler und Stolperstellen

- Oracle-Funktionen und Funktionen von SQL Server mischen, zum Beispiel GETDATE() in Oracle SQL.
- DATE '2024-01-01' in SQL Server verwenden, obwohl diese Schreibweise für Oracle SQL gedacht ist.
- Annehmen, dass alle Beispiele ohne Anpassung in beiden Systemen laufen.

Zusammenfassung

Oracle SQL und SQL Server folgen derselben SQL-Denkweise, haben aber unterschiedliche Dialekte. Bei einfachen SELECT-Abfragen ist vieles gleich. Bei Datum, Textfunktionen und Datentypen muss bewusst unterschieden werden.

5 Erste SELECT-Abfragen

Nach den Grundlagen werden nun einfache SELECT-Abfragen bewusst geschrieben und gelesen.

Lernziele

- eine einzelne Spalte auswählen
- mehrere Spalten auswählen
- SELECT-Listen lesbar schreiben
- erkennen, dass die ausgewählten Spalten die Ergebnisbreite bestimmen

Fachlicher Einstieg

Eine SELECT-Abfrage beginnt meistens mit der Frage: Welche Spalten brauche ich wirklich? Für Anfänger ist es sinnvoll, bewusst nur wenige Spalten auszuwählen und das Ergebnis zu verstehen.

Grundmuster

```
SELECT
    spalte1,
    spalte2,
    spalte3
FROM tabelle;
```

Erklärung des Grundmusters

Die Spaltenliste bestimmt, welche Spalten im Ergebnis erscheinen. Die Anzahl der Ergebniszeilen wird dadurch nicht verändert. Sie hängt zunächst von der Tabelle ab, aus der gelesen wird.

Praxisbeispiele

Beispiel 5.1: Eine Spalte auswählen

Nur die Kundennamen werden angezeigt.

```
SELECT
    kundenname
FROM kunden;
```

Das Ergebnis enthält weiterhin eine Zeile je Kunde, aber nur eine Spalte.

Beispiel 5.2: Mehrere Kundenspalten auswählen

Mehrere Spalten werden durch Kommas getrennt.

```
SELECT
    kundennr,
    kundenname,
    ort
FROM kunden;
```

Die Spalten werden in der Reihenfolge angezeigt, in der sie in SELECT stehen.

Beispiel 5.3: Rechnungsdaten bewusst auswählen

Für eine Rechnungsliste sind Rechnungsnummer, Datum und Betrag wichtig.

```
SELECT
    rechnungsnr,
    rechnungsdatum,
    bruttobetrag
FROM rechnungen;
```

Weitere Spalten wie status können ergänzt werden, wenn sie für die Frage benötigt werden.

Beispiel 5.4: Kontaktdaten anzeigen

Die Abfrage zeigt Kontaktspalten aus kunden.

```
SELECT
    kundennr,
    kundenname,
    telefon,
    email
FROM kunden;
```

Fehlende Werte erscheinen als NULL. NULL wird in Kapitel 14 genauer erklärt.

Typische Fehler und Stolperstellen

- Komma zwischen Spalten vergessen.
- Ein Komma nach der letzten Spalte setzen.
- SELECT * als fertige Berichtslösung verwenden, statt die benötigten Spalten bewusst zu wählen.
- Spalten aus einer anderen Tabelle erwarten, obwohl nur eine Tabelle in FROM steht.