

SQL

SQL (offizielle Aussprache [ɛskjuːˈɛl], mitunter auch [ˈsiːkwəl]; auf Deutsch auch häufig die deutsche Aussprache der Buchstaben) ist eine Datenbanksprache zur Definition von Datenstrukturen in relationalen Datenbanken sowie zum Bearbeiten (Einfügen, Verändern, Löschen) und Abfragen von darauf basierenden Datenbeständen.

Die Sprache basiert auf der relationalen Algebra, ihre Syntax ist relativ einfach aufgebaut und semantisch an die englische Umgangssprache angelehnt. Ein gemeinsames Gremium von ISO und IEC standardisiert die Sprache unter Mitwirkung nationaler Normungsgremien wie ANSI oder DIN. Durch den Einsatz von SQL strebt man die Unabhängigkeit der Anwendungen vom eingesetzten Datenbankmanagementsystem an.

Die Bezeichnung *SQL* wird im allgemeinen Sprachgebrauch als Abkürzung für „**Structured Query Language**“ (auf Deutsch: „Strukturierte Abfrage-Sprache“) aufgefasst, obwohl sie laut Standard ein eigenständiger Name ist. Die Bezeichnung leitet sich von dem Vorgänger SEQUEL ([ˈsiːkwəl], Structured English Query Language) ab, welche mit Beteiligung von Edgar F. Codd (IBM) in den 1970er Jahren von Donald D. Chamberlin und Raymond F. Boyce entwickelt wurde. SEQUEL wurde später in SQL umbenannt, weil SEQUEL ein eingetragenes Warenzeichen der Hawker Siddeley Aircraft Company ist.^[1]

Inhaltsverzeichnis

Sprachelemente und Beispiele

- Einfache Abfrage
- Abfrage mit Spaltenauswahl (.)
- Abfrage mit eindeutigen Werten (DISTINCT)
- Abfrage mit Umbenennung (AS)
- Abfrage mit Filter (WHERE)
- Abfrage mit Filter nach Inhalt (WHERE ... LIKE ...)
- Abfrage mit Filter und Sortierung (ORDER BY)
- Abfrage mit verknüpften Tabellen (, und INNER JOIN)
- Linker äußerer Verbund (LEFT OUTER JOIN)
- Gruppierung mit Aggregat-Funktionen (GROUP BY)
- Zusammenfassung eines SELECT
- Einfügen von Datensätzen (INSERT INTO ... VALUES ...)
- Ändern von Datensätzen (UPDATE)
- Löschen von Datensätzen (DELETE)
- Zusammenfassung von INSERT, UPDATE und DELETE

Datendefinition

- Datenbanktabelle (CREATE TABLE)
- Datenbankindex (CREATE INDEX)
- Sicht (CREATE VIEW)
- Zusammenfassung von CREATE TABLE, CREATE INDEX und CREATE VIEW

Redundanz

Schlüssel

Referenzielle Integrität

SQL-Datentypen

Transaktion, Commit und Rollback

Programmieren mit SQL

Chronologie

Sprachstandard

Erweiterungen

Literatur

Weblinks

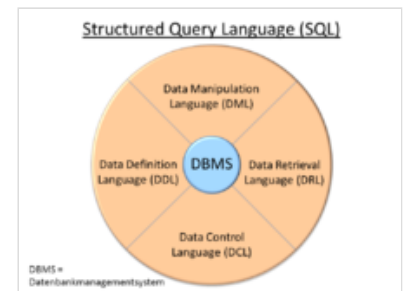
Siehe auch

Einzelnachweise

Sprachelemente und Beispiele

SQL-Befehle lassen sich in fünf Kategorien unterteilen (Zuordnung nach der Theorie der Datenbanksprachen in Klammern):

- Data Query Language (DQL) – Befehle zur Abfrage und Aufbereitung der gesuchten Informationen, wird auch als Untermenge der DML klassifiziert
- Data Manipulation Language (DML) – Befehle zur Datenmanipulation (Ändern, Einfügen, Löschen von Datensätzen) und lesendem Zugriff
- Data Definition Language (DDL) – Befehle zur Definition des Datenbankschemas (Erzeugen, Ändern, Löschen von Datenbanktabellen, Definition von Primärschlüsseln und Fremdschlüsseln)
- Data Control Language (DCL) – Befehle für die Rechteverwaltung
- Transaction Control Language (TCL) – Befehle für die Transaktionskontrolle



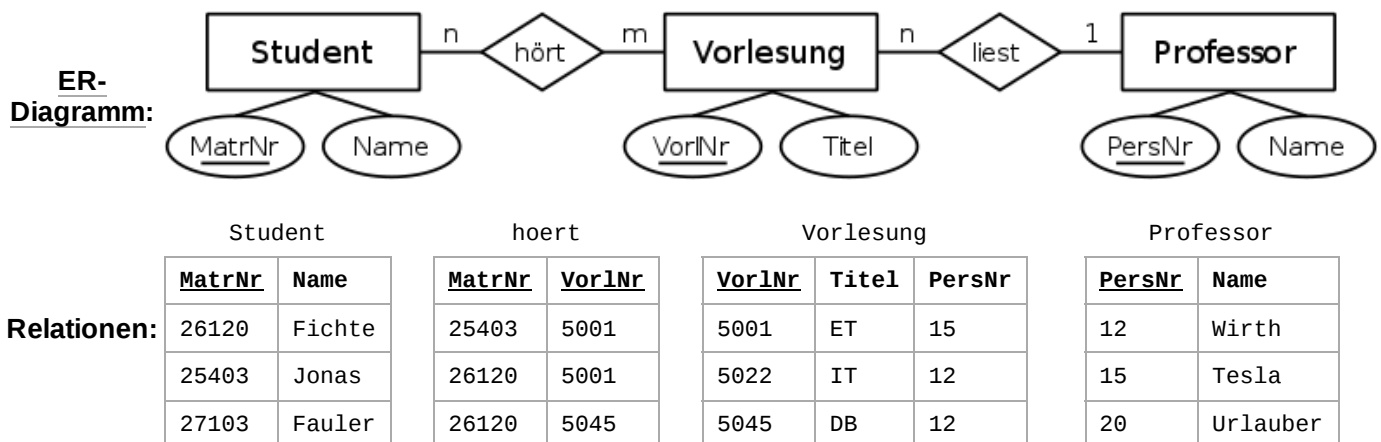
Bestandteile von SQL

Die Bezeichnung SQL bezieht sich auf das englische Wort “query” (deutsch: „Abfrage“). Mit Abfragen werden die in einer Datenbank gespeicherten Daten abgerufen, also dem Benutzer oder einer Anwendersoftware zur Verfügung gestellt.

Das Ergebnis einer Abfrage sieht wiederum aus wie eine Tabelle und kann oft auch wie eine Tabelle angezeigt, bearbeitet und weiterverwendet werden.

Siehe auch: Datenbanktabelle

Die grundlegenden Befehle und Begriffe werden anhand des folgenden Beispiels erklärt:



Einfache Abfrage

```
SELECT *  
FROM Student;
```

listet alle Spalten und alle Zeilen der Tabelle `Student` auf.

Ergebnis:

MatrNr	Name
26120	Fichte
25403	Jonas
27103	Fauler

Abfrage mit Spaltenauswahl (,)

```
SELECT VorlNr,  
       Titel  
FROM Vorlesung;
```

listet die Spalten `VorlNr` und `Titel` aller Zeilen der Tabelle `Vorlesung` auf.

Ergebnis:

VorlNr	Titel
5001	ET
5022	IT
5045	DB

Abfrage mit eindeutigen Werten (DISTINCT)

```
SELECT DISTINCT MatrNr  
FROM hoert;
```

listet nur unterschiedliche Einträge der Spalte `MatrNr` aus der Tabelle `hoert` auf. Dies zeigt die Matrikelnummern aller Studenten, die mindestens eine Vorlesung hören, wobei mehrfach auftretende Matrikelnummern nur einmal ausgegeben werden.

Ergebnis:

MatrNr
25403
26120

Abfrage mit Umbenennung (AS)

```
SELECT MatrNr AS Matrikelnummer,  
       Name  
FROM Student;
```

listet die Spalten `MatrNr` und `Name` aller Zeilen der Tabelle `Student` auf. `MatrNr` wird beim Anzeigergebnis als Matrikelnummer aufgeführt.

Ergebnis:

Matrikelnummer	Name
26120	Fichte
25403	Jonas
27103	Fauler

Abfrage mit Filter (WHERE)

```
SELECT VorlNr,
       Titel
FROM Vorlesung
WHERE Titel = 'ET';
```

listet VorlNr und Titel aller derjenigen Zeilen der Tabelle Vorlesung auf, deren Titel ET ist.

Die solchermaßen strukturierte, häufig verwendete Anweisung wird nach den Anfangsbuchstaben auch als „SFW-Block“ bezeichnet.

Ergebnis:

VorlNr	Titel
5001	ET

Abfrage mit Filter nach Inhalt (WHERE ... LIKE ...)

```
SELECT Name
FROM Student
WHERE Name LIKE 'F%';
```

listet die Namen aller Studenten auf, deren Name mit F beginnt (im Beispiel: Fichte und Fauler).

LIKE kann mit verschiedenen Platzhaltern verwendet werden: _ steht für ein einzelnes beliebiges Zeichen, % steht für eine beliebige Zeichenfolge. Manche Datenbanksysteme bieten weitere solcher *Wildcard*-Zeichen an, etwa für Zeichenmengen.

Ergebnis:

Name
Fichte
Fauler

Abfrage mit Filter und Sortierung (ORDER BY)

```
SELECT Vorname,
       Name,
       StrasseNr,
       Plz,
       Ort
FROM Student
WHERE Plz = '20095'
ORDER BY Name;
```

listet Vorname, Name, StrasseNr, Plz und Ort aller Studenten aus dem angegebenen Postleitzahlbereich aufsteigend sortiert nach Name auf.

Abfrage mit verknüpften Tabellen (, und INNER JOIN)

```
SELECT Vorlesung.VorlNr,  
       Vorlesung.Titel,  
       Professor.PersNr,  
       Professor.Name  
FROM   Professor,  
       Vorlesung  
WHERE  Professor.PersNr = Vorlesung.PersNr;
```

Die Aufzählung hinter **FROM** legt die Datenquellen fest: An dieser Stelle können mit Hilfe sogenannter **JOINS** mehrere Tabellen miteinander verknüpft werden, so dass Daten aus verschiedenen Tabellen zusammengeführt und angezeigt werden.

In diesem Beispiel wird ein „innerer natürlicher Verbund“ (**NATURAL INNER JOIN**) verwendet: alle Datensätze aus den Tabellen **Professor** und **Vorlesung**, die den gleichen Wert im Feld **PersNr** haben. Professoren ohne Vorlesung und Vorlesungen ohne Professor werden damit nicht angezeigt.

Dies ist äquivalent zu:

```
SELECT Vorlesung.VorlNr,  
       Vorlesung.Titel,  
       Professor.PersNr,  
       Professor.Name  
FROM   Professor  
INNER JOIN Vorlesung  
ON     Professor.PersNr = Vorlesung.PersNr;
```

*Vorsicht: Nicht alle Implementierungen verstehen beide Schreibweisen, die Oracle-Schreibweise **FROM Professor, Vorlesung** gilt als veraltet und ist weniger verbreitet. Sie entspricht auch nicht dem ANSI-Standard und sollte deshalb vermieden werden. Aus historischen Gründen ist sie jedoch noch häufig anzutreffen.*

Tabellen können nicht nur über Schlüsselfelder, sondern über beliebige Felder miteinander verknüpft werden, wie das folgende, fachlich unsinnige Beispiel zeigt:

```
SELECT Vorlesung.Titel,  
       Professor.Name  
FROM   Professor,  
       Vorlesung  
WHERE  Professor.Name <> Vorlesung.Titel
```

Das Ergebnis enthält die Kombinationen *aller* Vorlesungen und *aller* Professoren, bei denen der Name des Professors vom Titel der Vorlesung *abweicht* – das sind einfach alle (keine Vorlesung heißt wie ein Professor):

Titel	Name
ET	Tesla
ET	Wirth
ET	Urlauber
IT	Tesla
IT	Wirth
IT	Urlauber
DB	Tesla
DB	Wirth
DB	Urlauber

Linker äußerer Verbund (LEFT OUTER JOIN)

```

SELECT Professor.PersNr,
       Professor.Name,
       Vorlesung.VorlNr,
       Vorlesung.Titel
FROM Professor
LEFT OUTER JOIN Vorlesung
ON Professor.PersNr = Vorlesung.PersNr;
```

ergibt alle Datensätze der Tabelle **Professor**, verbunden mit den Datensätzen der Tabelle **Vorlesung**, die den jeweils gleichen Wert im Feld **PersNr** haben. Professoren ohne Vorlesung sind im Ergebnis enthalten. Die Spalten aus der **Vorlesung**-Tabelle haben dann den Wert **NULL**. Vorlesungen ohne **Professor** sind nicht enthalten.

Die folgende Abfrage liefert nur diejenigen Datensätze, zu denen kein passender Datensatz im linken äußeren Verbund existiert (alle Professoren, die keine Vorlesungen halten):

```

SELECT Professor.PersNr,
       Professor.Name
FROM Professor
LEFT OUTER JOIN Vorlesung
ON Professor.PersNr = Vorlesung.PersNr
WHERE Vorlesung.PersNr IS NULL;
```

Das Gleiche kann mittels einer Unterabfrage erreicht werden:

```

SELECT Professor.PersNr,
       Professor.Name
FROM Professor
WHERE NOT EXISTS (SELECT *
                  FROM Vorlesung
                  WHERE PersNr = Professor.PersNr);
```

Gruppierung mit Aggregat-Funktionen (GROUP BY)

```

SELECT Professor.PersNr,
       Professor.Name,
       COUNT(Vorlesung.PersNr) AS Anzahl
FROM Professor
LEFT OUTER JOIN Vorlesung
ON Professor.PersNr = Vorlesung.PersNr
GROUP BY Professor.Name,
         Professor.PersNr;
```

zählt die Anzahl der Vorlesungen pro Professor mit Hilfe der Aggregat-Funktion **COUNT**.

Bemerkung: COUNT(Professor.PersNr) oder COUNT(*) wären falsch (NULL-Werte sollen nicht mitgezählt werden).

Zusammenfassung eines SELECT

Zusammengefasst kann man die wichtigsten Elemente einer SQL-SELECT-Abfrage etwa so beschreiben:

```
SELECT [DISTINCT] Auswahlliste [AS Spaltenalias]
FROM Quelle [ [AS] Tabellenalias], evtl. mit JOIN-Verknüpfungen
[WHERE Where-Klausel]
[GROUP BY ein oder mehrere Group-by-Attribute]
[HAVING Having-Klausel]
[ORDER BY ein oder mehrere Sortierungsattribute mit [ASC|DESC]];
```

Erläuterung:

- **DISTINCT:** gibt an, dass aus der Ergebnisrelation gleiche Ergebnistupel entfernt werden sollen. Es wird also jeder Datensatz nur einmal ausgegeben, auch wenn er mehrfach in der Tabelle vorkommt. Sonst liefert SQL eine Multimenge zurück.
- **Auswahlliste:** bestimmt, welche Spalten der *Quelle* auszugeben sind (* für alle) und ob Aggregatsfunktionen anzuwenden sind. Wie bei allen anderen Aufzählungen werden die einzelnen Elemente mit Komma (,) voneinander getrennt.
- **Quelle:** gibt an, wo die Daten herkommen. Es können Relationen und Sichten angegeben werden und miteinander als kartesisches Produkt oder als Verbund (JOIN, ab SQL-92) verknüpft werden. Mit der zusätzlichen Angabe eines Namens können Relationen für die Abfrage umbenannt werden (vgl. Beispiele).
- **WHERE-Klausel:** bestimmt Bedingungen, auch Filter genannt, unter denen die Daten ausgegeben werden sollen. In SQL ist hier auch die Angabe von Unterabfragen möglich, so dass SQL *streng relational vollständig* wird.
- **GROUP BY-Attribut:** legt fest, ob unterschiedliche Werte als einzelne Zeilen ausgegeben werden sollen (GROUP BY = Gruppierung) oder aber die Feldwerte der Zeilen durch Aggregationen wie Addition (SUM), Durchschnitt (AVG), Minimum (MIN), Maximum (MAX) zu einem Ergebniswert zusammengefasst werden, der sich auf die Gruppierung bezieht.
- **HAVING-Klausel:** ist wie die WHERE-Klausel, nur dass sich die angegebene Bedingung auf das Ergebnis einer Aggregationsfunktion bezieht, zum Beispiel HAVING SUM (Betrag) > 0.
- **Sortierungsattribut:** Nach ORDER BY werden Attribute angegeben, nach denen sortiert werden soll. Die Standardvoreinstellung ist ASC, das bedeutet aufsteigende Sortierung, DESC ist absteigende Sortierung.

Mengenoperatoren können auf mehrere SELECT-Abfragen angewandt werden, die gleich viele Attribute haben und bei denen die Datentypen der Attribute übereinstimmen:

- **UNION:** vereinigt die Ergebnismengen. In einigen Implementierungen werden mehrfach vorkommende Ergebnistupel wie bei DISTINCT entfernt, ohne dass UNION DISTINCT geschrieben werden muss beziehungsweise darf.
- **UNION ALL:** vereinigt die Ergebnismengen. Mehrfach vorkommende Ergebnistupel bleiben erhalten. Einige Implementierungen interpretieren aber UNION wie UNION ALL und verstehen das ALL möglicherweise nicht und geben eine Fehlermeldung aus.
- **EXCEPT:** liefert die Tupel, die in einer ersten, jedoch nicht in einer zweiten Ergebnismenge enthalten sind. Mehrfach vorkommende Ergebnistupel werden entfernt.
- **MINUS:** ein analoger Operator wie EXCEPT, der in manchen SQL-Dialekten alternativ benutzt wird.
- **INTERSECT:** liefert die Schnittmenge zweier Ergebnismengen. Mehrfach vorkommende Ergebnistupel werden entfernt.

Einfügen von Datensätzen (INSERT INTO ... VALUES ...)

```
INSERT INTO Vorlesung (VorlNr, Titel, PersNr) VALUES (1000, 'Softwareentwicklung 1', 12);
INSERT INTO Vorlesung (VorlNr, Titel, PersNr) VALUES (1600, 'Algorithmen', 12);
INSERT INTO Vorlesung (VorlNr, Titel, PersNr) VALUES (1200, 'Netzwerke 1', 20);
INSERT INTO Vorlesung (VorlNr, Titel, PersNr) VALUES (1001, 'Datenbanken', 15);
```

fügt vier Datensätze in die Tabelle **Vorlesung** ein. Die Werte müssen mit den Datentypen der Felder **VorlNr**, **Titel** und **PersNr** zusammenpassen.

```
SELECT *  
FROM Vorlesung;
```

liefert dann zum Beispiel das Ergebnis (die Reihenfolge kann auch anders sein):

VorlNr	Titel	PersNr
1001	Datenbanken	15
1000	Softwareentwicklung 1	12
1200	Netzwerke 1	20
5001	ET	12
5022	IT	12
1600	Algorithmen	12
5045	DB	15

Ändern von Datensätzen (UPDATE)

```
UPDATE Vorlesung  
SET VorlNr = VorlNr + 1000,  
    PersNr = 20  
WHERE PersNr = 15;
```

ändert alle Datensätze, für die **PersNr** den Wert **15** hat. Der Wert von **VorlNr** wird um **1000** erhöht und der Wert von **PersNr** auf **20** gesetzt.

Ergebnis eines nachfolgenden **SELECT *** ist, eventuell mit anderer Reihenfolge:

VorlNr	Titel	PersNr
1000	Softwareentwicklung 1	12
1200	Netzwerke 1	20
1600	Algorithmen	12
2001	Datenbanken	20
5001	ET	12
5022	IT	12
6045	DB	20

Löschen von Datensätzen (DELETE)

```
DELETE FROM Vorlesung  
WHERE PersNr = 12;
```

löscht alle Datensätze, für die **PersNr** den Wert **12** hat.

Ergebnis eines nachfolgenden **SELECT ***, eventuell in anderer Reihenfolge:

VorlNr	Titel	PersNr
1200	Netzwerke 1	20
2001	Datenbanken	20
6045	DB	20

Zusammenfassung von INSERT, UPDATE und DELETE

Verallgemeinert sehen die Änderungsanweisungen wie folgt aus.

INSERT-Anweisung:

```
INSERT INTO Quelle [(Auswahlliste)]
VALUES (Werteliste) | SELECT <Auswahlkriterien>;
```

UPDATE-Anweisung:

```
UPDATE Quelle SET Zuweisungsliste
[FROM From-Klausel]
[WHERE Auswahlbedingung];
```

DELETE-Anweisung:

```
DELETE FROM Quelle
[WHERE Auswahlbedingung];
```

Datendefinition

Datenbanktabelle (CREATE TABLE)

Die Datenbanktabelle Vorlesung kann mit der folgenden Anweisung erzeugt werden:

```
CREATE TABLE Vorlesung (VorlNr INT NOT NULL PRIMARY KEY, Titel VARCHAR NOT NULL, PersNr NOT NULL, FOREIGN KEY
(PersNr) REFERENCES Professor (PersNr));
```

In keinem der Felder VorlNr, Titel, PersNr ist der Wert NULL erlaubt. Der Fremdschlüssel PersNr referenziert den Primärschlüssel PersNr der Tabelle Professor. Damit wird sichergestellt, dass in die Tabelle Vorlesung nur Datensätze eingefügt werden können, für die der Wert von PersNr in der Tabelle Professor als Primärschlüssel vorkommt (siehe referenzielle Integrität).

Datenbankindex (CREATE INDEX)

Mit der Anweisung

```
CREATE INDEX VorlesungIndex
ON Vorlesung (PersNr, Titel);
```

kann ein Datenbankindex für die Tabelle Vorlesung definiert werden, der vom Datenbanksystem bei der Ausführung von Abfragen zur Beschleunigung verwendet werden kann. Ob das sinnvoll ist, entscheidet das Datenbanksystem eigenständig durch komplexe Auswertungen und Analysen für jede Abfrage erneut.

Sicht (CREATE VIEW)

Eine Sicht ist im Wesentlichen ein Alias für eine Datenbankabfrage. Sie kann wie eine Datenbanktabelle verwendet werden. Die Anweisung

```
CREATE VIEW Vorlesungssicht AS
SELECT Vorlesung.VorlNr,
       Vorlesung.Titel,
       Professor.PersNr,
       Professor.Name
FROM Professor
INNER JOIN Vorlesung
ON Professor.PersNr = Vorlesung.PersNr;
```

speichert die definierte Abfrage als Sicht. Die Abfrage

```
SELECT Titel,
       Name
FROM Vorlesungssicht
WHERE VorlNr < 5000;
```

verwendet diese Sicht und könnte zum Beispiel folgendes Ergebnis liefern:

Titel	Name
Softwareentwicklung 1	Wirth
Netzwerke 1	Urlauber
Algorithmen	Wirth
Datenbanken	Urlauber

Zusammenfassung von CREATE TABLE, CREATE INDEX und CREATE VIEW

Zusammengefasst sind die wichtigsten Elemente der Definition einer Datenbanktabelle, eines Datenbankindex oder einer Sicht wie folgt anzugeben:

```
CREATE TABLE Tabellennamen (Attributdefinition [PRIMARY KEY]) [, FOREIGN KEY (Attributliste) REFERENCES
Tabellennamen (Attributliste)];
DROP TABLE Tabellennamen;
ALTER TABLE Tabellennamen (Attributdefinition [PRIMARY KEY]) [, FOREIGN KEY (Attributliste) REFERENCES
Tabellennamen (Attributliste)];

CREATE INDEX Indexname ON Tabellennamen (Attributliste);
DROP INDEX Indexname;

CREATE VIEW Sichtname [(Attributliste)] AS SELECT <Auswahlkriterien>;
DROP VIEW Sichtname;
```

Redundanz

→ Hauptartikel: Normalisierung (Datenbank)

Ein Grundsatz des Datenbankdesigns ist, dass in einer Datenbank keine Redundanzen auftreten sollen. Dies bedeutet, dass jede Information, also zum Beispiel eine Adresse, nur genau einmal gespeichert wird.

Beispiel: in der Teilnehmerliste einer Vorlesung werden die Adressen nicht erneut erfasst, sondern nur indirekt über die Matrikelnummer. Um dennoch eine Teilnehmerliste mit Adressen zu erstellen, erfolgt eine SELECT-Abfrage, in der die Teilnehmertabelle mit der Studententabelle verknüpft wird (siehe oben: JOIN).

In manchen Fällen ist die Performance einer Datenbank besser, wenn sie nicht (vollständig) normalisiert wird. In diesem Falle werden in der Praxis oft Redundanzen bewusst in Kauf genommen, um zeitaufwändige und komplexe Joins zu verkürzen und so die Geschwindigkeit der Abfragen zu erhöhen. Man spricht auch von einer Denormalisierung einer Datenbank. Wann (und ob überhaupt) eine Denormalisierung sinnvoll ist, ist umstritten und hängt von den Umständen ab.

Schlüssel

→ Hauptartikel: Schlüssel (Datenbank)

Während die Informationen auf viele Tabellen verteilt werden müssen, um Redundanzen zu vermeiden, sind Schlüssel das Mittel, um diese verstreuten Informationen miteinander zu verknüpfen.

So hat in der Regel jeder Datensatz eine eindeutige Nummer oder ein anderes eindeutiges Feld, um ihn zu identifizieren. Diese Identifikationen werden als Schlüssel bezeichnet.

Wenn dieser Datensatz in anderen Zusammenhängen benötigt wird, wird lediglich sein Schlüssel angegeben. So werden bei der Erfassung von Vorlesungsteilnehmern nicht deren Namen und Adressen, sondern nur deren jeweilige Matrikelnummer erfasst, aus der sich alle weiteren Personalien ergeben.

So kann es sein, dass manche Datensätze nur aus Schlüssel (meist Zahlen) bestehen, die erst in Verbindung mit Verknüpfungen verständlich werden. Der eigene Schlüssel des Datensatzes wird dabei als Primärschlüssel bezeichnet. Andere Schlüssel im Datensatz, die auf die Primärschlüssel anderer Tabellen verweisen, werden als Fremdschlüssel bezeichnet.

Schlüssel können auch aus einer Kombination mehrerer Angaben bestehen. Zum Beispiel können die Teilnehmer einer Vorlesung durch die eindeutige Kombination von Vorlesungsnummer und Studentenummer identifiziert werden, so dass die doppelte Anmeldung eines Studenten zu einer Vorlesung ausgeschlossen ist.

Referenzielle Integrität

→ Hauptartikel: Referentielle Integrität

Referenzielle Integrität bedeutet, dass Datensätze, die von anderen Datensätzen verwendet werden, in der Datenbank auch vollständig vorhanden sind.

Im obigen Beispiel bedeutet dies, dass in der Teilnehmertabelle nur Matrikel-Nummern gespeichert sind, die es in der Studenten-Tabelle auch tatsächlich gibt.

Diese wichtige Funktionalität kann (und sollte) bereits von der Datenbank überwacht werden, so dass zum Beispiel

- nur vorhandene Matrikelnummern in die Teilnehmertabelle eingetragen werden können,
- der Versuch, den Datensatz eines Studenten, der schon eine Vorlesung belegt hat, zu *löschen*, entweder verhindert wird (Fehlermeldung) oder der Datensatz auch gleich aus der Teilnehmertabelle entfernt wird (Löschweitergabe) und
- der Versuch, die Matrikelnummer eines Studenten, der schon eine Vorlesung belegt hat, zu *ändern*, entweder verhindert wird (Fehlermeldung) oder der Eintrag in der Teilnehmertabelle gleich mitgeändert wird (Aktualisierungsweitergabe).

Widersprüchlichkeit von Daten wird allgemein als Dateninkonsistenz bezeichnet. Diese besteht, wenn Daten bspw. die Integritätsbedingungen (z. B. Constraints oder Fremdschlüsselbeziehungen) nicht erfüllen.

Ursachen für Dateninkonsistenzen können Fehler bei der Analyse des Datenmodells, fehlende Normalisierung des ERM oder Fehler in der Programmierung sein.

Zum letzteren gehören die Lost-Update-Phänomene sowie die Verarbeitung von zwischenzeitlich veralteten Zwischenergebnissen. Dies tritt vor allem bei der Online-Verarbeitung auf, da dem Nutzer angezeigte Werte nicht in einer Transaktion gekapselt werden können.

Beispiel:
Transaktion A liest Wert x
Transaktion B verringert Wert x um 10
Transaktion A erhöht den gespeicherten Wert von x um eins und schreibt zurück
Ergebnis $x' = x + 1$
Die Änderung von B ist verloren gegangen

SQL-Datentypen

→ Hauptartikel: Datentypen

In den oben vorgestellten Befehlen `CREATE TABLE` und `ALTER TABLE` wird bei der Definition jeder Spalte angegeben, welchen Datentyp die Werte dieser Spalte annehmen können. Dazu liefert SQL eine ganze Reihe standardisierter Datentypen mit. Die einzelnen DBMS-Hersteller haben diese Liste jedoch um eine Unzahl weiterer Datentypen erweitert. Die wichtigsten Standarddatentypen sind:

INTEGER

Ganze Zahl (positiv oder negativ), wobei je nach Zahl der verwendeten Bits Bezeichnungen wie `SMALLINT`, `TINYINT` oder `BIGINT` verwendet werden. Die jeweiligen Grenzen und die verwendete Terminologie sind vom Datenbanksystem definiert.

NUMERIC (n, m) oder DECIMAL (n, m)

Festkommazahl (positiv oder negativ) mit insgesamt maximal n Stellen, davon m Nachkommastellen. Wegen der hier erfolgenden Speicherung als Dezimalzahl ist eine besonders für Geldbeträge notwendige Genauigkeit gegeben.

FLOAT (m)

Gleitkommazahl (positiv oder negativ) mit maximal m Nachkommastellen.

REAL

Gleitkommazahl (positiv oder negativ). Die Genauigkeit für diesen Datentyp ist jeweils vom Datenbanksystem definiert.

DOUBLE oder DOUBLE PRECISION

Gleitkommazahl (positiv oder negativ). Die Genauigkeit für diesen Datentyp ist jeweils vom Datenbanksystem definiert.

FLOAT und DOUBLE

sind für technisch-wissenschaftliche Werte geeignet und umfassen auch die Exponentialdarstellung. Wegen der Speicherung im Binärformat sind sie aber für Geldbeträge nicht geeignet, weil sich beispielsweise der Wert 0,10 € (entspricht 10 Cent) nicht exakt abbilden lässt.

CHARACTER (n) oder CHAR (n)

Zeichenkette Text mit n Zeichen.

VARCHAR (n) oder CHARACTER VARYING (n)

Zeichenkette (also Text) von variabler Länge, aber maximal n druckbaren und/oder nicht druckbaren Zeichen. Die Variante `VARCHAR2` ist für Oracle spezifisch, ohne dass sie sich tatsächlich unterscheidet.

TEXT

Zeichenkette (zumindest theoretisch) beliebiger Länge. In manchen Systemen synonym zu `CLOB`.

DATE

Datum (ohne Zeitangabe)

TIME

Zeitangabe (evtl. inklusive Zeitzone)

TIMESTAMP

Zeitstempel (umfasst Datum und Uhrzeit; evtl. inklusive Zeitzone), meistens mit Millisekundenauflösung, teilweise auch mikrosekundengenau

BOOLEAN

Boolesche Variable (kann die Werte `true`(wahr) oder `false` (falsch) oder `NULL` (unbekannt) annehmen). Dieser Datentyp ist laut SQL:2003 optional und nicht alle DBMS stellen diesen Datentyp bereit.

BLOB (n) oder BINARY LARGE OBJECT (n)

Binärdaten von maximal n Bytes Länge.

CLOB (n) oder CHARACTER LARGE OBJECT (n)

Zeichenketten mit maximal n Zeichen Länge.

Wenn es die Tabellendefinition erlaubt, können Attribute auch den Wert `NULL` annehmen, wenn kein Wert bekannt ist oder aus anderen Gründen kein Wert gespeichert werden soll. Der `NULL`-Wert ist von allen anderen möglichen Werten des Datentyps verschieden.

Transaktion, Commit und Rollback

→ *Hauptartikel: Transaktion (Informatik)*

Eine Transaktion bezeichnet eine Menge von Datenbankänderungen, die zusammen ausgeführt werden (müssen). So ist beispielsweise die Buchung (als Transaktion) eines Geldbetrags durch zwei atomare Datenbankoperationen „Abbuchung des Geldbetrages von Konto A“ und „Buchung des Geldbetrages auf Konto B“ gekennzeichnet. Kann die vollständige Abarbeitung der elementaren Datenbankoperationen der Transaktion nicht durchgeführt werden (z. B. aufgrund eines Fehlers), müssen alle durchgeführten Änderungen an dem Datenbestand auf den Ausgangszustand zurückgesetzt werden.

Der Vorgang, der alle Änderungen einer Transaktion zurücksetzt, wird als Rollback bezeichnet. Der Begriff Commit bezeichnet das Ausführen einer Transaktion. Transaktionen sind eine Möglichkeit, die Konsistenz des Datenbestandes zu sichern. Im Beispiel der doppelten Kontenführung wird durch das Verhindern von ungültigen Teilbuchungen eine ausgeglichene Kontobilanz gewährleistet.

Datenbanken erlauben es zum Teil, bestimmte Befehle außerhalb einer Transaktion auszuführen. Darunter fällt insbesondere das Laden von Daten in Tabellen oder das Exportieren von Daten mittels Utilities. Manche DBMS erlauben das temporäre Abschalten der Transaktionslogik sowie einiger Kontrollen zur Erhöhung der Verarbeitungsgeschwindigkeit. Dies muss allerdings meist durch einen expliziten Befehl erzwungen werden, um ein versehentliches Ändern von Daten außerhalb einer Transaktion zu vermeiden. Solche Änderungen können, falls eine Datenbankwiederherstellung erforderlich ist, zu schweren Problemen oder gar Datenverlusten führen. Eine Transaktion wird mit der SQL-Anweisung COMMIT beendet. Alle Änderungen der Transaktion werden persistent gemacht, und das DBMS stellt durch geeignete (interne) Mittel (z. B. Logging) sicher, dass diese Änderungen nicht verloren gehen.

Mit dem Befehl ROLLBACK wird eine Transaktion ebenfalls beendet, es werden jedoch alle Änderungen seit Beginn der Transaktion rückgängig gemacht. Das heißt, der Zustand des Systems (in Bezug auf die Änderungen der Transaktion) ist der gleiche wie vor der Transaktion.

Programmieren mit SQL

Programmierschnittstelle

Das ursprüngliche SQL war keine Turing-vollständige Programmiersprache, es ermöglichte also nicht die Realisierung von beliebigen Computerprogrammen. Mittlerweile lässt es sich mit anderen Programmiersprachen kombinieren, um eine Programmierung im engeren Sinne zu ermöglichen. Hierfür gibt es unterschiedliche Techniken.

- Mit Embedded SQL können SQL-Anweisungen im Quelltext eines Programms, typischerweise in C, C++, COBOL, Ada, Pascal o. Ä. geschrieben, eingebettet werden. Während der Programmvorbereitung übersetzt ein Precompiler die SQL-Befehle in Funktionsaufrufe. Embedded SQL ist Teil des ANSI-SQL-Standards. Beispiele für Implementierungen: SQLJ für Java, Pro*C für C, C++, ADO und ADO.NET.
- Herkömmliche Programmierschnittstellen erlauben die direkte Übergabe von SQL-Befehlen an Datenbanksysteme über Funktionsaufrufe. Beispiele: ODBC, JDBC, ADO.
- Persistenz-Frameworks wie etwa Hibernate oder iBATIS abstrahieren vom Datenbankzugriff und ermöglichen objektorientierte Verarbeitung von relationalen Datenbanken in einer objektorientierten Programmiersprache (z. B. Java oder C#).
- Mit dem Teil 4 SQL/PSM des Standards werden Konstrukte wie IF-Blöcke und Schleifen bereitgestellt. Er wird in den Datenbanksystemen in unterschiedlicher Ausprägung und mit Hersteller-spezifischen Erweiterungen implementiert, z. B. PL/SQL in Oracle oder Transact-SQL im MS SQL Server.

Statisches und dynamisches SQL

Unabhängig von der verwendeten Programmiertechnik wird zwischen *statischem* und *dynamischem* SQL unterschieden.

- Bei *statischem SQL* ist die SQL-Anweisung dem Datenbanksystem zum Zeitpunkt der Programmübersetzung bekannt und festgelegt (z. B. wenn die Abfrage eines Kontos vorformuliert ist und zur Laufzeit nur die Kontonummer eingesetzt wird).
- Bei *dynamischem SQL* ist die SQL-Anweisung dem Datenbanksystem erst zum Zeitpunkt der Programmausführung bekannt (z. B. weil der Benutzer die komplette Abfrage eingibt). So sind z. B. alle SQL-Anweisungen, die mittels SQL/CLI oder JDBC ausgeführt werden grundsätzlich dynamisch. Ausgeführt werden dynamische SQL-Anweisungen im Allgemeinen mit `execute immediate` (*SQL-String*).

Bei dynamischem SQL *muss* das Datenbanksystem die SQL-Anweisung zur Laufzeit des Programms interpretieren und den Zugriffspfad optimieren. Da dieser sogenannte Parse-Vorgang Zeit in Anspruch nimmt, puffern viele Datenbanksysteme die bereits geparsen SQL-Anweisungen, um mehrfaches Parsen gleicher Abfragen zu vermeiden. Bei statischem SQL *kann* schon bei der Übersetzung der Programme bzw. beim Binden der SQL-Anweisungen an eine Datenbank (sogenanntes *Bind* der SQL-Befehle) der optimale Zugriffsweg bestimmt werden. Damit sind kürzestmögliche Laufzeiten der

Anwendungsprogramme möglich, allerdings muss der Zugriffsweg aller betroffenen Programme neu bestimmt werden, wenn sich Voraussetzungen (z. B. Statistiken) ändern (*Rebind*). Die *Bind*-Phase ist heute vor allem im Großrechner-Umfeld bekannt, die meisten Datenbanksysteme optimieren hingegen zur Laufzeit.

Chronologie

- etwa 1975: *SEQUEL* = *Structured English Query Language*, der Vorläufer von *SQL*, wird für das Projekt System R von IBM entwickelt.
- 1979: *SQL* gelangt mit *Oracle V2* erstmals durch *Relational Software Inc.* auf den Markt.
- 1986: *SQL1* wird von *ANSI* als Standard verabschiedet.
- 1987: *SQL1* wird von der Internationalen Organisation für Normung (ISO) als Standard verabschiedet und 1989 nochmals überarbeitet.
- 1992: Der Standard *SQL2* oder *SQL-92* wird von der ISO verabschiedet.
- 1999: *SQL3* oder *SQL:1999* wird verabschiedet. Im Rahmen dieser Überarbeitung werden weitere wichtige Features (wie etwa Trigger oder rekursive Abfragen) hinzugefügt.
- 2003: *SQL:2003*. Als neue Features werden aufgenommen SQL/XML, Window functions, Sequences.
- 2006: *SQL/XML:2006*. Erweiterungen für SQL/XML^[2].
- 2008: *SQL:2008* bzw. ISO/IEC 9075:2008. Als neue Features werden aufgenommen *INSTEAD OF-Trigger*, *TRUNCATE*-Statement und *FETCH* Klausel.
- 2011: *SQL:2011* bzw. ISO/IEC 9075:2011. Als neue Features werden aufgenommen „Zeitbezogene Daten“ (*PERIOD FOR*). Es gibt Erweiterungen für Window functions und die *FETCH* Klausel.
- 2016: *SQL:2016* bzw. ISO/IEC 9075:2016. Als neue Features werden aufgenommen *JSON* und „row pattern matching“.
- 2019: *SQL/MDA:2019*. Erweiterungen für einen Datentyp „mehrdimensionales Feld“.

Sprachstandard

Ziel der Standardisierung ist es, Anwendungsprogramme so erstellen zu können, dass sie vom verwendeten Datenbanksystem unabhängig sind. Heutige Datenbanksysteme implementieren mehr oder weniger große Teile des Sprachstandards. Darüber hinaus stellen sie oftmals herstellerspezifische Erweiterungen bereit, die nicht dem Standard-Sprachumfang entsprechen. In der Vor-SQL-Zeit strebte man die Portabilität von Anwendungen über die kompatible Schnittstelle an.

Der Standard besteht insgesamt aus zehn einzelnen Publikationen:^[3]

- ISO/IEC 9075-1:2016 Part 1: Framework (SQL/Framework)
- ISO/IEC 9075-2:2016 Part 2: Foundation (SQL/Foundation)
- ISO/IEC 9075-3:2016 Part 3: Call-Level Interface (SQL/CLI)
- ISO/IEC 9075-4:2016 Part 4: Persistent stored modules (SQL/PSM)
- ISO/IEC 9075-9:2016 Part 9: Management of External Data (SQL/MED)
- ISO/IEC 9075-10:2016 Part 10: Object language bindings (SQL/OLB)
- ISO/IEC 9075-11:2016 Part 11: Information and definition schemas (SQL/Schemata)
- ISO/IEC 9075-13:2016 Part 13: SQL Routines and types using the Java TM programming language (SQL/JRT)
- ISO/IEC 9075-14:2016 Part 14: XML-Related Specifications (SQL/XML)
- ISO/IEC 9075-15:2019 Part 15: Multi-dimensional arrays (SQL/MDA)

Ein weiterer Teil befindet sich derzeit (2019) in Entwicklung:

- ISO/IEC 9075-16:20xx Part 16: Property Graph Queries (SQL/PGQ)

Der Standard wird durch sechs, ebenfalls standardisierte *SQL multimedia and application packages* ergänzt:

- ISO/IEC 13249-1:2016 Part 1: Framework
- ISO/IEC 13249-2:2003 Part 2: Full-Text
- ISO/IEC 13249-3:2016 Part 3: Spatial
- ISO/IEC 13249-5:2003 Part 5: Still image

- ISO/IEC 13249-6:2006 Part 6: Data mining
- ISO/IEC 13249-7:2013 Part 7: History

Weiterhin existieren eine Reihe Technical Reports, die eine Einführung zu den einzelnen Themen bieten.

- ISO/IEC TR 19075-1:2011 Part 1: XQuery Regular Expression Support in SQL Download (https://standards.iso.org/ittf/PubliclyAvailableStandards/c041528_ISO_IEC_TR_19075-1_2011.zip)
- ISO/IEC TR 19075-2:2015 Part 2: SQL Support for Time-Related Information Download (https://standards.iso.org/ittf/PubliclyAvailableStandards/c060394_ISO_IEC_TR_19075-2_2015.zip)
- ISO/IEC TR 19075-3:2015 Part 3: SQL Embedded in Programs using the JavaTM programming language Download (https://standards.iso.org/ittf/PubliclyAvailableStandards/c065141_ISO_IEC_TR_19075-3_2015.zip)
- ISO/IEC TR 19075-4:2015 Part 4: SQL with Routines and types using the JavaTM programming language Download (https://standards.iso.org/ittf/PubliclyAvailableStandards/c065142_ISO_IEC_TR_19075-4_2015.zip)
- ISO/IEC TR 19075-5:2016 Part 5: Row Pattern Recognition in SQL Download (https://standards.iso.org/ittf/PubliclyAvailableStandards/c065143_ISO_IEC_TR_19075-5_2016.zip)
- ISO/IEC TR 19075-6:2017 Part 6: SQL support for JavaScript Object Notation (JSON) Download (https://standards.iso.org/ittf/PubliclyAvailableStandards/c067367_ISO_IEC_TR_19075-6_2017.zip)
- ISO/IEC TR 19075-7:2017 Part 7: Polymorphic table functions in SQL Download (https://standards.iso.org/ittf/PubliclyAvailableStandards/c069776_ISO_IEC_TR_19075-7_2017.zip)
- ISO/IEC TR 19075-8:2019 Part 8: Multi-dimensional arrays (SQL/MDA) (wurde zurückgezogen^[4])

Ein weiterer Teil befindet sich derzeit (2019) in Entwicklung:

- ISO/IEC TR 19075-9:20xx Part 9: Online Analytic Processing (OLAP) capabilities

Der offizielle Standard ist nicht frei verfügbar, jedoch existiert ein Zip-Archiv mit einer Arbeitsversion von 2008.^[5] Die Technical Reports sind kostenlos von ISO erhältlich.

Erweiterungen

Die beiden ersten Teile des SQL Standards *SQL/Framework* und *SQL/Foundation* legen die Kernfunktionalitäten fest. In den weiteren Teilen werden spezifische Aspekte der Sprache definiert.

- Teil 4: Bei *SQL/PSM* handelt es sich um die Erweiterung um prozedurale Konstrukte. Sie ermöglichen unter anderem das Programmieren von Schleifen (FOR, WHILE, REPEAT UNTIL, LOOP), Cursors, Exception-Handling, Triggern und eigenen Funktionen. Oracle implementiert diese Funktionalität unter dem Namen *PL/SQL*, DB2 verwendet den Begriff *SQL/PL*, PostgreSQL nennt es *PL/pgSQL*.
- Teil 14: *SQL/XML* ermöglicht es, XML-Dokumente in SQL-Datenbanken zu speichern, mit XPath in SQL/XML:2003 und XQuery ab SQL/XML:2006 abzufragen und relationale Datenbankinhalte als XML zu exportieren. Um die ursprünglichen Arbeiten an diesem Teil des Standards zu beschleunigen, hatte sich im Jahr 2000 eine informelle Arbeitsgruppe gebildet (IBM, Oracle, ...), die unter dem Namen *The SQLX Group* und unter der Bezeichnung *SQLX* die Kernfunktionalitäten festlegte. Deren Arbeit ist in den jetzigen Standard eingeflossen.


Als Ergänzung zum SQL-Standard existiert mit *ISO/IEC 13249: SQL multimedia and application packages* eine Norm, die für die Anwendungsfälle *Text*, *Geografische Daten*, *Bilder*, *Data mining* und *Metadaten* spezialisierte Schnittstellen in SQL Syntax festlegt.

Literatur

- Donald D. Chamberlin, Raymond F. Boyce: *SEQUEL: A Structured English Query Language*. In: *SIGMOD Workshop*. Vol. 1 1974, S. 249–264.
- Donald D. Chamberlin, Morton M. Astrahan, Kapali P. Eswaran, Patricia P. Griffiths, Raymond A. Lorie, James W. Mehl, Phyllis Reisner, Bradford W. Wade: *SEQUEL 2: A Unified Approach to Data Definition, Manipulation, and Control*. In: *IBM Journal of Research and Development*. 20(6) 1976, S. 560–575.
- Günter Matthiessen, Michael Unterstein: *Relationale Datenbanken und SQL in Theorie und Praxis* Springer Vieweg, ISBN 978-3-642-28985-9.
- Edwin Schicker: *Datenbanken und SQL – Eine praxisorientierte Einführung*. Teubner, ISBN 3-519-02991-X.

- Oliver Bartosch, Markus Throll: *Einstieg in SQL*. Galileo Press, ISBN 3-89842-497-9.
- Daniel Warner, Günter Leitenbauer: *SQL*. Franzis, ISBN 3-7723-7527-8.
- H. Faeskorn-Woyke, B. Bertelsmeier, P. Riemer, E. Bauer: *Datenbanksysteme, Theorie und Praxis mit SQL2003, Oracle und MySQL*. Pearson-Studium, ISBN 978-3-8273-7266-6.
- Jörg Fritze, Jürgen Marsch: *Erfolgreiche Datenbank Anwendung mit SQL3. Praxisorientierte Anleitung – effizienter Einsatz – inklusive SQL-Tuning*. Vieweg Verlag, ISBN 3-528-55210-7.
- Can Türker: *SQL 1999 & SQL 2003*. Dpunkt Verlag, ISBN 3-89864-219-4.
- Gregor Kuhlmann, Friedrich Müllmerstadt: *SQL*. Rowohlt, ISBN 3-499-61245-3.
- Michael J. Hernandez, John L. Viescas: *Go To SQL*. Addison-Wesley, ISBN 3-8273-1772-X.
- A. Kemper, A. Eickler: *Datenbanksysteme – Eine Einführung*. Oldenbourg, ISBN 3-486-25053-1.
- Marcus Throll, Oliver Bartosch: *Einstieg in SQL 2008*. 2. Auflage. Galileo Computing, ISBN 978-3-8362-1039-3 inklusive Übungssoftware SQL-Teacher
- Marco Skulschus: *SQL und relationale Datenbanken* Comelio Medien, ISBN 978-3-939701-11-8.
- Michael Wagner: *SQL/XML:2006 – Evaluierung der Standardkonformität ausgewählter Datenbanksysteme* 1. Auflage. Diplomica Verlag, ISBN 3-8366-9609-6.
- Christian F. G. Schendera: *SQL mit SAS. Band 1: PROC SQL für Einsteiger*. Oldenbourg Wissenschaftsverlag, München 2011, ISBN 978-3-486-59840-7.
- Christian F. G. Schendera: *SQL mit SAS. Band 2: Fortgeschrittenes PROC SQL*. Oldenbourg Wissenschaftsverlag, München 2012, ISBN 978-3-486-59836-0.
- C. J. Date with Hugh Darwen: *A Guide to the SQL standard: a users guide to the standard database language SQL, 4th ed.*, Addison-Wesley, USA 1997, ISBN 978-0-201-96426-4
- Jim Melton: *Advanced SQL:1999: Understanding Object-Relational and Other Advanced Features, 1st ed.*, Morgan Kaufmann, USA, 2002, ISBN 978-1558606777.

Weblinks

 **Wikibooks: SQL** – Lern- und Lehrmaterialien

- Erklärvideos zu SQL (<https://www.youtube.com/playlist?list=PLC4UZxBVGKte0o6iUssqQVmxhryrOqPXi>), Big Data Analytics Group, Uni Saarland
- Linkkatalog zum Thema SQL (<https://curlie.org/World/Deutsch/Computer/Programmieren/Sprachen/SQL/>) bei *curlie.org* (ehemals DMOZ)
- SQL-Grundlagen (<http://www.torsten-horn.de/techdocs/sql.htm>) – Einführung mit Beispielen und Vergleich diverser Datenbanken
- Merkblatt SQL (<https://esb-dev.github.io/mat/sql-merkblatt.pdf>)
- The 1995 SQL Reunion: People, Projects, and Politics (http://www.mcjones.org/System_R/SQL_Reunion_95/) – zur frühen Geschichte von SQL (englisch)
- Frei verfügbare SQL-Standard-Dokumente (<http://www.wiscorp.com/SQLStandards.html>), z. B. SQL:2003 und SQL:2008 (englisch)
- SQL-Tutorial (<http://www.1keydata.com/de/sql/>)
- GNU SQLTutor (<http://sqltutor.fsv.cvut.cz/cgi-bin/sqltutor>)
- SQLcoach (<https://sqlcoach.informatik.hs-kl.de/sqlcoach/>) – Freies Üben von SQL
- Interaktiver SQL-Trainer (Anmeldung notwendig) (<https://edb2.gm.th-koeln.de/apps/sqltrainer/>)
- Blogbeitrag zu SQL-Befehlen und deren Einsatz in PHP (<https://blog.homepage-webhilfe.de/Artikel/mysql-tutorial-fuer-anfaenger/>)
- SQL Tuning (<http://sql-tuning.com/>)
- Transact-SQL Reference (Database Engine) (<https://docs.microsoft.com/en-us/sql/t-sql/language-reference?view=sql-server-ver15>)

Siehe auch

- [Data Manipulation Language](#)
- [Data Definition Language](#)
- [Data Control Language](#)
- [Transaction Control Language](#)
- [SQL-Injection](#)

- [SchemaSQL](#)
- [Continuous Query Language](#)
- [Liste der Datenbankmanagementsysteme](#)
- [Bereichsabfrage](#)

Einzelnachweise

1. *Diskussion über System R und zur Namensänderung von SEQUEL zu SQL* (http://www.mcjones.org/System_R/SQL_Reunion_95/sqlr95-System.html)
2. Michael Wagner: *SQL/XML:2006 – Evaluierung der Standardkonformität ausgewählter Datenbanksysteme*. Diplomica Verlag, 2010, ISBN 3-8366-9609-6, S. 100.
3. *ISO/IEC 9075 und 13249*. (<https://www.iso.org/committee/45342/x/catalogue/p/1/u/0/w/0/d/0>) International Organization for Standardization, abgerufen am 20. September 2018 (englisch).
4. 14:00-17:00: *ISO/IEC TR 19075-8:2019*. (<https://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/06/97/69777.html>) Abgerufen am 24. August 2022 (englisch).
5. Arbeitsversion des Standards von 2008 (<http://www.wiscorp.com/sql200n.zip>) (ZIP; 12,7 MB)

Normdaten (Sachbegriff): GND: 4134010-3 LCCN: sh86006628
--

Abgerufen von „<https://de.wikipedia.org/w/index.php?title=SQL&oldid=239068065>“

Diese Seite wurde zuletzt am 13. November 2023 um 08:06 Uhr bearbeitet.

Der Text ist unter der Lizenz „Creative-Commons Namensnennung – Weitergabe unter gleichen Bedingungen“ verfügbar; Informationen zu den Urhebern und zum Lizenzstatus eingebundener Mediendateien (etwa Bilder oder Videos) können im Regelfall durch Anklicken dieser abgerufen werden. Möglicherweise unterliegen die Inhalte jeweils zusätzlichen Bedingungen. Durch die Nutzung dieser Website erklären Sie sich mit den Nutzungsbedingungen und der Datenschutzrichtlinie einverstanden.
 Wikipedia® ist eine eingetragene Marke der Wikimedia Foundation Inc.