SQL Cheat Sheet

Montag, 30. Oktober 2023 20:31

Initialisierung

CREATE DATABASE *database-name*;

weitere Optionen (Bsp Rechte) möglich

CREATE TABLE *table-name* (*field-name type, field-name type, ...*);

Types:

CHARACTER(*m*) *and* CHARACTER VARYING(*m*) *for fixed and variable length strings of (maximum) length m,*
BIT(*m*) *and* BIT VARYING(*m*) *for fixed and variable length bit strings of (maximum) length m,*
NUMERIC, DECIMAL, INTEGER, *and* SMALLINT *for fixed point and integer numbers,*
FLOAT, REAL, *and* DOUBLE PRECISION *for floating point numbers,*
DATE, TIME, *and* TIMESTAMP *for points in time, or*
INTERVAL *for ranges of time.*

gegen leere Einträge:

field-name type NOT NULL
field-name type DEFAULT value

Bearbeiten

INSERT INTO *table-name* (*field-name, ...*) **VALUES** (*value, ...*);

(value_1_1, ...), (value_2_1, ...), (value_3_1, ...); möglich

UPDATE *table* **SET** *field-name* = *value,...* **WHERE** *condition*;

DELETE FROM *table-name* **WHERE** *condition*;

Auslesen

SELECT *field-name, ...* **FROM** *table-name* **WHERE** *condition*;

Gleichheit: field-name = value
Vergleichsoperatoren: <, >, <=, >=
Logik: AND OR NOT ( )
leer: IS NULL
Muster in strings: field-name LIKE string
→ schreibe Platzhalter _ oder % für beliebig lange Platzhalter
→ Bsp. '1 2 3 4 5 6 7 8 9 10' LIKE '_2%8_10%'

alles auslesen: SELECT * FROM table-name;

SELECT *aggregate, ...*;

aggregate <=> Funktion: MIN(field-name), MAX(...)

COUNT(field_name): Reihen mit Einträgen zählen
FUNKTION(DISTINCT field_name): nur einzigartige Einträge

SELECT *field-name|aggregate, ...* **GROUP BY** *field-name,...*;

aggregate auf Gruppen von Einträge mit gleichen field_name,...

Einschränkungen vor dem Gruppieren: WHERE...

Einschränkungen nach dem Gruppieren: HAVING...

SELECT ... ORDER BY *field-name,...*;

Ausgabe sortieren nach field_name,...
DESC; / ASC; am Ende für ab/aufsteigend

Keys

ALTER TABLE *table*
ADD CONSTRAINT UNIQUE (*field-name*,...);

Einzigartigkeit in dem Feld (der Kombination der Felder) erzwingen

ALTER TABLE *table* ADD PRIMARY KEY (*field-name*,...);

erzwingt Einzigartigkeit, Einträge ≠ NULL
Feld kann als Key (Verlinkung) verwendet werden

ALTER TABLE *left-table* ADD FOREIGN KEY (*field-name*,...)
REFERENCES right-table;

fügt left-table Spalten hinzu in denen Einträge aus right-table mit "primary keys" verlinkt werden können

Joins

SELECT ...
FROM *left-table* INNER JOIN *right-table* ON *condition*;

condition: left-table.foreign_key = right-table.primary_key

fügt Zeilen von left-table mit den verlinkten von right-table zusammen

Zeilen ohne mit "foreign_key=NULL" in left-table und nicht verlinkte Zeilen in right-table werden ignoriert

SELECT ...
FROM *left-table* LEFT|RIGHT|FULL OUTER JOIN *right-table*
ON *condition*;

condition: left-table.foreign_key = right-table.primary_key

fügt Zeilen von left-table mit den verlinkten von right-table zusammen

LEFT OUTER JOIN:
nicht verlinkte Zeilen in right-table werden ignoriert, alle Zeilen aus left-table werden angezeigt

RIGHT OUTER JOIN:
Zeilen ohne mit "foreign_key=NULL" in left-table werden ignoriert, alle Zeilen aus right-table werden angezeigt

FULL OUTER JOIN:
alle Zeilen werden angezeigt