Inhaltsverzeichnis

1. CONSTRAINTS

Constraints können beim erstellen einer Tabelle oder später über ALTER TABLE einer Tabelle hinzugefügt werden. Sie bestimmen Regeln für die Daten in der Tabelle.

```
CREATE TABLE table_name (
column1 datatype constraint,
column2 datatype constraint,
column3 datatype constraint,
....
);
```

- NOT NULL
- UNIQUE
- PRIMARY KEY
- FOREIGN KEY
- CHECK
- DEFAULT
- INDEX

2. FOREIGN KEY

Foreign Keys werden verwendet in einer Tabelle verweist auf den Primärschlüssel einer anderen Tabelle.

3. ALTER TABLE

Ermöglicht das verändern von bereits erstellten Tabellen.

```
ALTER TABLE table_name
ADD column_name datatype;
```

Löschen einer Zeile

```
ALTER TABLE table_name
DROP COLUMN column_name;
```

Ändern des Datentyps einer Zeile

```
ALTER TABLE table_name
ALTER COLUMN column_name datatype;
```

4. AUTO INCREMENT

Erzeugt automatisch eine Nummer wenn neue Daten der Tabelle hinzugefügt werden.

```
CREATE TABLE Persons (
ID int NOT NULL AUTO_INCREMENT,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Age int,
PRIMARY KEY (ID)
);
```

5. SELECT

Ermöglicht Ausgabe von Datensätzen aus einer oder mehrere Tabellen. Zuerst werden die Spaltennamen der Datensätze angegeben, die Ausgegeben werden sollen, anschließend wird die Tabelle genannt, in der sich die Datensätze befinden.

```
SELECT
 spalte_1, spalte_2 ...
FROM
 tabelle_1
INNER | LEFT | RIGHT JOIN tabelle_2 ON Bedingungen
WHERE BEDINGUNGEN
GROUP BY spalte_1
HAVING group_Bedingungen
ORDER BY spalte_1
```

HAVING wird benutzt da bei WHERE keine aggregierenden Funktionen, also Funktionen die dazu dienen Spaltenwerte zusammenzuführen, wie z.B. AVG oder SUM genutzt werden können.

6. JOIN

6.1 Inner Join

INNER JOIN bringt Daten aus zwei Tabellen zusammen. Als erstes wird die Haupttabelle genannt, danach die Tabelle die man zur mit der Haupttabelle zusammenführen möchte. Nach dem Schlüsselwort ON wird die Bedingung genannt, unter der die Tabellen zusammengeführt werden sollen. Nur wenn

```
SELECT column_list
FROM t1
INNER JOIN t2 ON join_condition1
WHERE where_conditions;
```

6.2 Left Join

Gibt alle aus der linken Tabelle, die die zuerst genannt wird, die gesamten Inhalte, sowie alle, die ebenfalls in der Rechten Tabelle vorkommen aus. Inhalte, die nur rechts vorhanden sind, werden nicht ausgegeben.

```
SELECT column_name(s)
FROM table1
LEFT JOIN table2 ON table1.column_name = table2.column_name;
```

6.3 Right Join

Gibt alle aus der rechten Tabelle, die die als zweites genannt wird, die gesamten Inhalte, sowie alle, die ebenfalls in der linken Tabelle vorkommen aus. Inhalte, die nur links vorhanden sind, werden nicht ausgegeben.

```
SELECT column_name(s)
FROM table1
RIGHT JOIN table2 ON table1.column_name = table2.column_name;
```

6.4 Full Outer Join

Gibt alle Inhalte aus der linken und der rechten Tabelle aus, wenn es die Suchkriterien erfüllt.

```
SELECT column_name(s)
FROM table1
FULL OUTER JOIN table2 ON table1.column_name = table2.column_name;
```

7. GROUP BY

Sortiert die Ausgabe nach einem Spaltennamen.

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
ORDER BY column_name(s);
```

8. LIKE

Ermöglicht Ausgabe aller Datensätze, die ein Bestimmtes Muster aufweisen.

```
SELECT
employeeNumber, lastName, firstName
FROM
employees
WHERE
firstName LIKE 'a%';
```

9. HAVING

Wird bei der Verwendung von WHERE benötigt, da diese nicht bei aggregierenden Funktionen, wie z.B SUM genutzt werden kann.

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
HAVING condition
ORDER BY column_name(s);
```

10. VIEWS

Ein VIEW ist eine Virtuelle Tabelle basierend auf dem Ergebnissen einer SELECT abfrage auf eine oder mehrere Tabellen aus einer Datenbank. Es enthält Zeilen und Spalten wie eine Tabelle.

```
CREATE VIEW view_name AS
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

11. ANOMALIEN

11.1 Einfügeanomalie

Bei einem fehlerhaften oder inkorrekten Datenbankendesign kann es bei der Einfügeanomalie passieren, dass Daten gar nicht in die Datenbank übernomen werden wenn nicht zu allen Attributen (Den Spaltenüberschriften) Werte vorliegen. Das Einfügen von Datensätzen ohne den Primärschlüssel (oder Teile davon) ist nicht möglich.

11.2 Änderungsanomalie

Bei der Änderungsanomalie werden gleiche Attribute eines Datensatzes in einer Transaktion nicht automatisch geändert. So entsteht eine Inkonsistenz der Daten.

11.3 Löschanomalie

Zu einer Löschanomalie kann es kommen, wenn durch das löschen eines Datensatzes mehr Informationen gelöscht werden als gewünscht. Dies kann geschehen, wenn ein Datensatz mehrere unabhängige Informationen enthält.

12. NORMALFORMEN

12.1 Erste Normalform

Jedes Attribut muss atomar vorliegen. Dass bedeutet, dass die Daten nicht weiter aufgespalten werden, beispielsweise wenn der Name in Vor- und Nachname aufgeteilt wurde.

12.2 Zweite Normalform

Erste Normalform muss vorliegen. Ebenfalls müssen nicht Schlüsselattribute (Die, die nicht Teil des Primärschlüssels sind) von allen ganzen Schlüsseln abhängig. Voraussetzungen für die zweite Normalform zusammengefasst:

- In der ersetn Normalform
- ist von einem Schlüsselkandidaten abhängig, oder
- ist nicht von einer echten Teilmenge eines Schlüsselkandidaten abhängig.

12.3 Dritte Normalform

Erste und zweite Normalform müssen vorliegen. Kein Nichtschlüsselattribut darf von einem Schlüsselattribut abhängig sein.

13. MENGENOPERATIONEN

13.1 Union

Verbindet die Ergebnisse von zwei oder mehr SELECT Abfragen. Dabei muss man beachten, dass die SELECT Statements die selbe anzahl an Spalten aufweisen, sowie dass sie dieselben Datentypen beinhalten sowie in derselben Reihenfolge aufgelistet werden.

```
SELECT column_name(s) FROM table1
UNION
SELECT column_name(s) FROM table2;
```

13.2 Intersect

Das Ergebnis enthält die Schnittmenge der Teilmengen, d.h. die Datensätze müssen in allen Tabellen der Abfrage vorhanden sein.

```
Beispiel: Artikel, die sowohl eingekauft als auch verkauft wurden:
select distinct artikel_nr
from bestell_pos p

INTERSECT

select distinct artikel_nr
from auftrag_pos;
```

13.3 minus

Die Ergebnismenge enthält die Differenz der beiden Tabellen. Ein Datensatz wird nur dann als Ergebnis ausgegeben, wenn er in der ersten Tabelle vorhanden ist, aber nicht in der zweiten.

```
Beispiel: Artikel, die zwar eingekauft, nicht aber verkauft wurden:
select distinct artikel_nr
from bestell_pos p

MINUS

select distinct artikel_nr
from auftrag_pos;
```