7.5.3. CREATE TABLE-Syntax

7.5.3.1. Stille Spaltentyp-Änderungen

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tabelle [(create_definition,...)]
[tabellen_optionen] [select_statement]

create_definition:
  spalten_name typ [NOT NULL | NULL] [DEFAULT vorgabe_wert] [AUTO_INCREMENT]
            [PRIMARY KEY] [referenz_definition]
  oder    PRIMARY KEY (index_spalten_name,...)
  oder    KEY [index_name] (index_spalten_name,...)
  oder    INDEX [index_name] (index_spalten_name,...)
  oder    UNIQUE [INDEX] [index_name] (index_spalten_name,...)
  oder    FULLTEXT [INDEX] [index_name] (index_spalten_name,...)
  oder    [CONSTRAINT symbol] FOREIGN KEY index_name (index_spalten_name,...)
            [referenz_definition]
  oder    CHECK (ausdruck)

typ:
        TINYINT[(laenge)] [UNSIGNED] [ZEROFILL]
  oder    SMALLINT[(laenge)] [UNSIGNED] [ZEROFILL]
  oder    MEDIUMINT[(laenge)] [UNSIGNED] [ZEROFILL]
  oder    INT[(laenge)] [UNSIGNED] [ZEROFILL]
  oder    INTEGER[(laenge)] [UNSIGNED] [ZEROFILL]
  oder    BIGINT[(laenge)] [UNSIGNED] [ZEROFILL]
  oder    REAL[(laenge,dezimalstellen)] [UNSIGNED] [ZEROFILL]
  oder    DOUBLE[(laenge,dezimalstellen)] [UNSIGNED] [ZEROFILL]
  oder    FLOAT[(laenge,dezimalstellen)] [UNSIGNED] [ZEROFILL]
  oder    DECIMAL(laenge,dezimalstellen) [UNSIGNED] [ZEROFILL]
  oder    NUMERIC(laenge,dezimalstellen) [UNSIGNED] [ZEROFILL]
  oder    CHAR(laenge) [BINARY]
  oder    VARCHAR(laenge) [BINARY]
  oder    DATE, TIME, TIMESTAMP, DATETIME, TINYBLOB, BLOB, MEDIUMBLOB, LONGBLOB
  oder    TINYTEXT, TEXT, MEDIUMTEXT, LONGTEXT
  oder    ENUM(wert1,wert2,wert3,...)
  oder    SET(wert1,wert2,wert3,...)

index_spalten_name:
        spalten_name [(laenge)]

referenz_definition:
        REFERENCES tabelle [(index_spalten_name,...)]
                   [MATCH FULL | MATCH PARTIAL]
                   [ON DELETE referenz_option]
                   [ON UPDATE referenz_option]

referenz_option:
        RESTRICT | CASCADE | SET NULL | NO ACTION | SET DEFAULT

tabellen_optionen:
        TYPE = {BDB | HEAP | ISAM | InnoDB | MERGE | MRG_MYISAM | MYISAM }
or      AUTO_INCREMENT = #
or      AVG_ROW_LENGTH = #   oder   CHECKSUM = {0 | 1}
or      COMMENT = "string"
or      MAX_ROWS = #   oder   MIN_ROWS = #   oder   PACK_KEYS = {0 | 1 | DEFAULT}
or      PASSWORD = "string"
or      DELAY_KEY_WRITE = {0 | 1}
or      ROW_FORMAT= { default | dynamic | fixed | compressed }
or      RAID_TYPE= {1 | STRIPED | RAID0 } RAID_CHUNKS=#  RAID_CHUNKSIZE=#
or      UNION = (tabelle,[tabelle...])   oder   INSERT_METHOD= {NO | FIRST | LAST }
or      DATA directory="verzeichnis"   oder   INDEX directory="verzeichnis"

select_statement:
        [IGNORE | REPLACE] SELECT ...  (jedes zulässige SELECT-Statement)
```

`CREATE TABLE` erzeugt eine Tabelle mit dem angegebenen Namen in der aktuellen Datenbank.

7.5.4. `ALTER TABLE`-Syntax

```
ALTER [IGNORE] TABLE tabelle aenderungs_angabe [, aenderungs_angabe ...]

aenderungs_angabe:
        ADD [COLUMN] create_definition [FIRST | AFTER spalten_name]
  oder    ADD [COLUMN] (create_definition, create_definition,...)
  oder    ADD INDEX [index_name] (index_spalten_name,...)
  oder    ADD PRIMARY KEY (index_spalten_name,...)
  oder    ADD UNIQUE [index_name] (index_spalten_name,...)
  oder    ADD FULLTEXT [index_name] (index_spalten_name,...)
  or    ADD [CONSTRAINT symbol] FOREIGN KEY index_name (index_spalten_name,...)
            [referenz_definition]
  oder    ALTER [COLUMN] spalten_name {SET DEFAULT literal | DROP DEFAULT}
  oder    CHANGE [COLUMN] alter_spalten_name create_definition
  oder    MODIFY [COLUMN] create_definition
  oder    DROP [COLUMN] spalten_name
  oder    DROP PRIMARY KEY
  oder    DROP INDEX index_name
  oder    DISABLE KEYS
  oder    ENABLE KEYS
  oder    RENAME [TO] neue_tabelle
  oder    ORDER BY spalte
  oder    tabellen_optionen
```

Mit `ALTER TABLE` können Sie die Struktur einer bestehenden Tabelle ändern. Sie können beispielsweise Spalten hinzufügen oder löschen, Indexe erzeugen oder löschen, den Typ bestehender Spalten ändern oder Spalten oder die Tabelle selbst umbenennen. Sie können auch den Kommentar für die Tabelle und den Typ der Tabelle ändern.

7.5.6. `DROP TABLE`-Syntax

```
DROP TABLE [IF EXISTS] tabelle [, tabelle,...] [RESTRICT | CASCADE]
```

`DROP TABLE` entfernt eine oder mehrere Tabellen. Alle Tabellendaten und die Tabellendefinition werden *zerstört*, seien Sie daher **vorsichtig** mit diesem Befehl!

Ab MySQL-Version 3.22 können Sie die Schlüsselwörter `IF EXISTS` benutzen, um Fehler zu vermeiden, die auftreten, wenn Tabellen nicht existieren.

7.5.7. `CREATE INDEX`-Syntax

```
CREATE [UNIQUE|FULLTEXT] INDEX index_name ON tabelle (spalten_name[(laenge)],... )
```

Das `CREATE INDEX`-Statement macht vor MySQL-Version 3.22 nichts. Ab Version 3.22 ist `CREATE INDEX` auf ein `ALTER TABLE`-Statement gemappt, um Indexe zu erzeugen.

Normalerweise erzeugen Sie alle Indexe auf eine Tabelle zur Zeit, wo die Tabelle selbst mit `CREATE TABLE` erzeugt wird.

7.4.1. SELECT-Syntax


```
SELECT [STRAIGHT_JOIN] [SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
       [HIGH_PRIORITY]
       [DISTINCT | DISTINCTROW | ALL]
    select_ausdruck,...
    [INTO {OUTFILE | DUMPFILE} 'datei' export_optionen]
    [FROM tabellenreferenz
        [WHERE where_definition]
        [GROUP BY {positive_ganzzahl | spalten_name | formel} [ASC | DESC], ...]
        [HAVING where_definition]
        [ORDER BY {positive_ganzzahl | spalten_name | formel} [ASC | DESC] ,...]
        [LIMIT [offset,] zeilen]
        [PROCEDURE prozedur_name]
        [FOR UPDATE | LOCK IN SHARE MODE]]
```

`SELECT` wird benutzt, um ausgewählte Zeilen aus einer oder mehreren Tabellen abzurufen. `select_ausdruck` gibt die Spalten an, die Sie abrufen wollen. `SELECT` kann auch benutzt werden, um Zeilen ohne Bezug zu irgend einer Tabelle abzurufen.

7.4.1.1. JOIN-Syntax

MySQL unterstützt folgende `JOIN`-Syntaxen für `SELECT`-Statements:

```
tabellen_verweis, tabellen_verweis
tabellen_verweis [CROSS] JOIN tabellen_verweis
tabellen_verweis INNER JOIN tabellen_verweis join_bedingung
tabellen_verweis STRAIGHT_JOIN tabellen_verweis
tabellen_verweis LEFT [OUTER] JOIN tabellen_verweis join_bedingung
tabellen_verweis LEFT [OUTER] JOIN tabellen_verweis
tabellen_verweis NATURAL [LEFT [OUTER]] JOIN tabellen_verweis
{ oder tabellen_verweis LEFT OUTER JOIN tabellen_verweis ON bedingungs_ausdruck }
tabellen_verweis RIGHT [OUTER] JOIN tabellen_verweis join_bedingung
tabellen_verweis RIGHT [OUTER] JOIN tabellen_verweis
tabellen_verweis NATURAL [RIGHT [OUTER]] JOIN tabellen_verweis
```

Wobei `tabellen_verweis` definiert ist als:

```
tabelle [[AS] alias] [USE INDEX (schluessel_liste)] [IGNORE INDEX
(schluessel_liste)]
```

Und `join_bedingung` definiert ist als:

```
ON bedingungs_ausdruck |
USING (spalten_liste)
```

Sie sollten nie irgend welche Bedingungen im `ON`-Teil haben, die dazu benutzt werden, um die Zeilen, die im Ergebnissatz auftauchen, zu beschränken. Wenn Sie so etwas tun wollen, müssen Sie das in der `WHERE`-Klausel tun.

7.4.2. INSERT-Syntax

```
    INSERT [LOW_PRIORITY | DELAYED] [IGNORE]
        [INTO] tabelle [(spalten_name,...)]
        VALUES (ausdruck,...),(...),...
oder  INSERT [LOW_PRIORITY | DELAYED] [IGNORE]
        [INTO] tabelle [(spalten_name,...)]
        SELECT ...
oder  INSERT [LOW_PRIORITY | DELAYED] [IGNORE]
        [INTO] tabelle
        SET spalten_name=ausdruck, spalten_name=ausdruck, ...
```

`INSERT` fügt neue Zeilen in eine bestehende Tabelle ein. Die `INSERT ... VALUES`-Form des Statements fügt Zeilen basierend auf explizit angegebenen Werten ein. Die `INSERT ... SELECT`-Form fügt Zeilen ein, die aus einer oder mehreren anderen Tabellen ausgewählt wurden.

7.4.5. UPDATE-Syntax

```
UPDATE [LOW_PRIORITY] [IGNORE] tabelle
    SET spalten_name1=ausdruck1, [spalten_name2=ausdruck2, ...]
    [WHERE where_definition]
    [LIMIT #]
```

`UPDATE` aktualisiert Spalten in bestehenden Tabellenzeilen mit neuen Werten. Die `SET`-Klausel gibt an, welche Spalten geändert werden sollen und welche Werte ihnen zugewiesen werden. Die `WHERE`-Klausel legt - falls angegeben - fest, welche Zeilen aktualisiert werden sollen. Ansonsten werden alle Zeile aktualisiert. Wenn die `ORDER BY`-Klausel angegeben ist, werden die Zeilen in der angegebenen Reihenfolge aktualisiert.

Wenn Sie das Schlüsselwort `IGNORE` angeben, bricht das UPDATE-Statement nicht ab, selbst wenn während der Aktualisierung Fehler wegen doppelter Schlüsseleinträge auftreten. Zeilen, die Konflikte verursachen würden, werden nicht aktualisiert.

7.4.6. DELETE-Syntax

```
DELETE [LOW_PRIORITY | QUICK] FROM tabelle
    [WHERE where_definition]
    [ORDER BY ...]
    [LIMIT zeilen]

oder

DELETE [LOW_PRIORITY | QUICK] tabelle[.*] [tabelle[.*] ...] FROM
tabellenverweis [WHERE where_definition]
```

`DELETE` löscht Zeilen aus `tabelle`, die mit der in `where_definition` angegebenen Bedingung übereinstimmen, und gibt die Anzahl der gelöschten Datensätze zurück.

7.3.6. Funktionen zur Benutzung bei GROUP BY-Klauseln

Wenn Sie in einem Statement eine Gruppierungsfunktion benutzen, die keine GROUP BY-Klausel enthält, ist das gleichbedeutend mit der Gruppierung aller Zeilen.

- COUNT(ausdruck)

  Gibt die Anzahl der Zeilen mit Nicht-NULL-Werten zurück, die durch ein SELECT-Statement abgerufen werden:

```
mysql> select student.student_name,COUNT(*)
           from student,kurs
           where student.student_id=kurs.student_id
           GROUP BY student_name;
```

  COUNT(*) ist insofern anders, als es die Anzahl der abgerufenen Zeilen zurückgibt, egal ob sie NULL-Werte enthalten oder nicht.

  COUNT(*) ist darauf optimiert, das Ergebnis sehr schnell zurückzugeben, wenn es mittels eines SELECT von einer Tabelle abruft, wenn keine weiteren Spalten abgerufen werden und es keine WHERE-Klausel gibt. Beispiel:

```
mysql> select COUNT(*) from student;
```

- COUNT(DISTINCT ausdruck,[ausdruck...])

  Gibt die Anzahl unterschiedlicher Nich-NULL-Werte zurück:

```
mysql> select COUNT(DISTINCT ergebnisse) from student;
```

  Bei MySQL erhalten Sie die Anzahl unterschiedlicher Ausdruckskombinationen, die nicht NULL enthalten, indem Sie eine Liste von Ausdrücken angeben. In ANSI-SQL müssten Sie eine Verkettung aller Ausdrücke innerhalb von CODE(DISTINCT ..) angeben.

- AVG(ausdruck)

  Gibt den Durchschnittswert von ausdruck zurück:

```
mysql> select student_name, AVG(test_ergebnis)
           from student
           GROUP BY student_name;
```

- MIN(ausdruck),MAX(ausdruck)

  Gibt den kleinsten oder größten Wert von ausdruck zurück. MIN() und MAX() können Zeichenketten-Argumente aufnehmen und geben in solchen Fällen den kleinsten oder größten Zeichenketten- Wert zurück. See Abschnitt 6.4.3, „Wie MySQL Indexe benutzt“.

```
mysql> select student_name, MIN(test_ergebnis), MAX(test_ergebnis)
           from student
           GROUP BY student_name;
```

- `SUM(ausdruck)`

  Gibt die Summe von `ausdruck` zurück. Beachten Sie, dass der Rückgabewert NULL ist, wenn die Ergebnismenge keine Zeilen hat!

- `STD(ausdruck)`, `STDDEV(ausdruck)`

  Gibt die Standardabweichung von `ausdruck` zurück. Das ist eine Erweiterung zu ANSI-SQL. Die `STDDEV()`-Form dieser Funktion wird aus Gründen der Oracle-Kompatibilität zur Verfügung gestellt.

- `BIT_OR(ausdruck)`

  Gibt das bitweise `OR` aller Bits in `ausdruck` zurück. Die Berechnung wird mit 64-Bit-(`BIGINT`)-Genauigkeit durchgeführt.

- `BIT_AND(ausdruck)`

  Gibt das bitweise `AND` aller Bits in `ausdruck` zurück. Die Berechnung wird mit 64-Bit-(`BIGINT`)-Genauigkeit durchgeführt.

MySQL hat die Benutzung von `GROUP BY` erweitert. Sie können Spalten oder Berechnungen im `SELECT`-Ausdruck angeben, die nicht im `GROUP BY`-Teil erscheinen. Das steht für *jeden möglichen Wert für diese Gruppe*. Das können Sie benutzen, um bessere Performance zu erzielen, indem Sie Sortieren und Gruppieren unnötiger Bestandteile vermeiden. Zum Beispiel müssen Sie in folgender Anfrage nicht nach `kunde.name` gruppieren:

```
mysql> select bestellung.kunde_id,kunde.name,max(zahlungen)
       from bestellung,kunde
       where bestellung.kunde_id = kunde.kunde_id
       GROUP BY bestellung.kunde_id;
```

In ANSI-SQL müssten Sie der `GROUP BY`-Klausel `kunde.name` hinzufügen. In MySQL ist der Name überflüßig, solange Sie nicht im ANSI-Modus fahren.

Benutzen Sie dieses Feature nicht, wenn die Spalten, die Sie im `GROUP BY`-Teil auslassen, in der Gruppe nicht eindeutig sind! Sonst erhalten Sie unvorhersagbare Ergebnisse.

In einigen Fällen können Sie `MIN()` und `MAX()` benutzen, um einen bestimmten Spaltenwert zu erhalten, selbst wenn er nicht eindeutig ist. Folgendes gibt den Wert von `spalte` aus der Zeile zurück, die den kleinsten Wert in der `sortierung`-Spalte enthält:

```
substr(MIN(concat(rpad(sortierung,6,' '),spalte)),7)
```