Einführung in Datenbanken

Übung 3: Relationales Modell und Grundlagen der SQL Syntax

Prof. Dr. Stefan Brass
PD Dr. Alexander Hinneburg
Martin-Luther-Universität Halle-Wittenberg

Wintersemester 2020/21

http://www.informatik.uni-halle.de/~brass/db20/

Inhalt

Präsenzaufgabe: Schema für Koch-/Back-Rezepte

- Entwerfen Sie ein Schema für Rezepte (zum kochen/backen) und geben Sie es in der Kurznotation ab. Anforderungen:
  - Zu jedem Rezept muss eine Nummer, eine Bezeichnung, ein Schwierigkeitsgrad, und eine Dauer (der Zubereitung) gespeichert werden. Bezeichnungen sind nicht eindeutig.
  - Ein Rezept besteht aus mehreren Schritten. Es ist die Reihenfolge der Schritte festzuhalten, außerdem ein Text (Anweisung für den Schritt).
  - Für jeden Schritt ist zu speichern, welche Zutaten in welcher Menge benötigt werden. In einen Schritt können auch mehrere Zutaten benötigt werden (oder keine).

    Die gleiche Zutat in verschiedenen Schritten eines Rezepts ist möglich.

Präsenzaufgabe: Erste Lösung (1)

Lösung 1 (mit Fehlern):

- Rezept(<u>Nummer</u>, Bezeichnung, Schwierigkeitsgrad, Dauer)
- Schritt(<u>Position</u>, <u>Nummer</u> → Rezept, Anweisung)
- Zutat(<u>Name</u>, <u>Nummer</u> → Rezept)
- Zutat-pro-Schritt(<u>Nummer</u> → Rezept, <u>Position</u> → Schritt, <u>Name</u> → Zutat, Menge)

Frage:

- Sieht jemand Fehler/Verbesserungsmöglichkeiten?

Präsenzaufgabe: Erste Lösung (2)

Fehler I (Syntaxfehler):

- Ein Schlüssel aus zwei Attributen muss mit zwei Attributen referenziert werden.

  Anzahl (und Typen) der Fremdschlüssel-Attribute muss passen zu der Anzahl der Schlüssel-Attribute.

- Zutat-pro-Schritt(Nummer → Rezept,
  Position → Schritt,
  Name → Zutat, Menge)

- Schritt(Position, Nummer, ...)
  Zutat(Name, Nummer, ...)

Präsenzaufgabe: Erste Lösung (3)

Fehler II (Redundanz):

- Was ist der Sinn der Relation `Zutat`?
- Die Information, welche Zutat in welchem Rezept verwendet wird, ist bereits in der Relation `Zutat-pro-Schritt` enthalten.
- Vermeiden Sie die Speicherung redundanter Daten!
  - Mehr Arbeit beim Eingeben.
  - Gefahr von Inkonsistenzen.
- Sie können später aber `Zutat` als virtuelle Relation (Sicht) definieren, die aus `Zutat-pro-Schritt` berechnet wird.

Die Zusammenstellung der Zutaten pro Rezept ist ja durchaus sinnvoll.

Präsenzaufgabe: Erste Lösung (4)

Stilfrage I (Bezeichner):

- In SQL sind Bindestriche in Bezeichnern nicht zulässig: `Zutat-pro-Schritt`.
- Man kann es als „Delimited Identifier“ haben: `"Zutat-pro-Schritt"`.
- Man erleichtert sich spätere Anfragen aber, wenn man Unterstriche wählt: `Zutat_pro_Schritt`.

  In der Auflistung der Relationen erscheint es bei PostgreSQL dann allerdings ganz in Kleinbuchstaben.
- Bei der Lösung mit „Delimited Identifier“ muss man auch später in jeder Anfrage `"..."` schreiben.

Präsenzaufgabe: Erste Lösung (5)

Stilfrage II (Attribut-Reihenfolge):

- `Schritt(Position, Nummer → Rezept, Anweisung)`
- Man sollte die Attribute so anordnen, dass die größere Einheit (hier die Rezept-Nummer) vor der kleineren Einheit (die Position des Schrittes innerhalb des Rezepts) steht.

  Gewissermaßen nach der Priorität in der natürlichen Sortierreihenfolge.

- Man sollte gleiche Attribute möglichst auch immer in gleicher Reihenfolge anordnen:

```
Zutat-pro-Schritt(Nummer → Rezept,
                  Position → Schritt,
                  Name → Zutat, Menge)
```

Mindestens für die Fremdschlüssel in dieser Notation ist das zwingend.

In SQL nicht. Eventuell bei `SELECT *`, `INSERT INTO` ohne Spaltenangabe.

Präsenzaufgabe: Erste Lösung (6)

Korrigierte Fassung von Lösung 1 (optimal):

- Rezept(Nummer, Bezeichnung, Schwierigkeitsgrad, Dauer)
- Schritt(Nummer → Rezept, Position, Anweisung)
- Zutat_pro_Schritt((Nummer, Position) → Schritt, Zutat_Name, Menge)

Frage:

- Braucht man in Zutat_pro_Schritt zusätzlich den Fremdschlüssel Nummer → Rezept?

Präsenzaufgabe: Zweite Lösung (1)

- ```
  CREATE TABLE REZEPTE(
        Rezeptnummer     NUMERIC(3)  NOT NULL,
        Rezeptbezeichner VARCHAR(20) NOT NULL,
        Dauer_in_min     NUMERIC(50) NOT NULL
        PRIMARY KEY (Rezeptnummer),
        UNIQUE (Rezeptbezeichner))
  ```

  Es war die Kurznotation gefordert. Schwierigkeitsgrad fehlt.

  Bezeichner nicht eindeutig (mehrere Rezepte für das gleiche Gericht).

- Sieht jemand den Syntaxfehler?
- Was bedeutet der Typ `NUMERIC(50)`?

  Viele DBMS werden das nicht unterstützen.

- `VARCHAR(20)` für den Rezeptbezeichner ist zu kurz.

  Z.B. hat „Elisenlebkuchen mit Belegkirschen“ 33 Zeichen.
  ```

Präsenzaufgabe: Zweite Lösung (2)

- ```
  CREATE TABLE ABLAUF(
        Rezeptnummer  NUMERIC(3)    NOT NULL,
        Platzierung   NUMERIC(50)   NOT NULL,
        Anweisung     VARCHAR(1000) NOT NULL,
        FOREIGN KEY (Rezeptnummer)
                      REFERENCES REZEPTE,
        FOREIGN KEY (Platzierung)
                      REFERENCES ZUTATEN)
  ```
- Anweisungen so anordnen, dass Fremdschlüssel bereits existierende Tabellen referenzieren (`ZUTATEN` vorher).

  Hier also besser zuerst `ZUTATEN`. Bei wechselseitigen Referenzen ist das nicht möglich, dann Vorlesung „Datenbank-Programmierung“ hören (`ALTER TABLE`).
- Tabelle hat keinen Schlüssel! $\rightarrow$ nächste Folie ...
  ```

Präsenzaufgabe: Zweite Lösung (3)

- Wenn man für eine Tabelle keinen Schlüssel deklariert, sind in SQL doppelte Zeilen möglich (in allen Spalten gleich).

  In der Theorie sind Relationen Mengen, in SQL aber Multimengen (von Tupeln). Mir ist bisher kein Fall begegnet, in dem eine gespeicherte Relation mit Duplikats-Tupeln wünschenswert gewesen wäre (bei Relationen als Anfrageergebnis ist das anders, da können Duplikate im Ausnahmefall wichtig sein). Es wird handhabbarer, wenn man eine Zahl als künstlichen Schlüssel hinzu fügt. Wenn man nur Duplikate ausschließen will, kann man das mit einem Schlüssel aus allen Attributen tun.

- Im konkreten Beispiel will man für eine Rezept-Nummer und eine „Platzierung“ (Schritt-Nummer) nur einen Eintrag in der Tabelle haben:

```
PRIMARY KEY(Rezeptnummer, Platzierung)
```

Präsenzaufgabe: Zweite Lösung (4)

- ```
  CREATE TABLE ZUTATEN(
          Rezeptnummer    NUMERIC(3)      NOT NULL,
          Zutat           VARCHAR(50)     NOT NULL,
          Platzierung     NUMERIC(50)     NOT NULL,
          Menge_in_g      NUMERIC(1000),
          FOREIGN KEY (Rezeptnummer)
                          REFERENCES REZEPTE)
  ```
- `NUMERIC(1000)` ist viel zu groß.
- Nullwert für Menge?
- Schlüssel fehlt.

  Der in `ABLAUF` deklarierte `FOREIGN KEY(Platzierung) REFERENCES ZUTATEN` ist nur möglich, wenn der Schlüssel hier ausschließlich aus `Platzierung` besteht. Dann aber nicht in einem Schritt mehrere Zutaten. Der Fremdschlüssel müsste eigentlich in der anderen Richtung verweisen, von `ZUTAT` zu `ABLAUF`.
  ```

Präsenzaufgabe: Zweite Lösung (5)

Korrigiert:

- REZEPTE(Rezeptnummer, Rezeptbezeichner, Schwierigkeitsgrad, Dauer)
- ABLAUF(Rezeptnummer → REZEPTE, Platzierung, Anweisung)
- ZUTATEN((Rezeptnummer, Platzierung) → ABLAUF, Zutat, Menge_in_g)
- Datentypen (Vorschläge, es gibt Bereich sinnvoller Längen):

Rezeptnummer	NUMERIC(5)
Platzierung	NUMERIC(2)
Schwierigkeitsgrad	NUMERIC(1)
Menge_in_g	NUMERIC(4)
Rezeptbezeichner	VARCHAR(50)

Präsenzaufgabe: Zweite Lösung (6)

- ```
  CREATE TABLE REZEPTE(
        Rezeptnummer        NUMERIC(5)   NOT NULL,
        Rezeptbezeichner    VARCHAR(50)  NOT NULL,
        Schwierigkeitsgrad  NUMERIC(1)   NOT NULL,
        Dauer_in_min        NUMERIC(3)   NOT NULL,
        PRIMARY KEY (Rezeptnummer))
  ```

- ```
  CREATE TABLE ABLAUF(
        Rezeptnummer  NUMERIC(5)     NOT NULL,
        Platzierung   VARCHAR(2)     NOT NULL,
        Anweisung     VARCHAR(1000)  NOT NULL,
        PRIMARY KEY (Rezeptnummer, Platzierung),
        FOREIGN KEY (Rezeptnummer)
                    REFERENCES REZEPTE)
  ```

Präsenzaufgabe: Zweite Lösung (7)

- ```
  CREATE TABLE ZUTATEN(
          Rezeptnummer   NUMERIC(5)  NOT NULL,
          Platzierung    VARCHAR(2)  NOT NULL,
          Zutat          VARCHAR(30) NOT NULL,
          Menge_in_g     NUMERIC(4)  NOT NULL,
          PRIMARY KEY (Rezeptnummer, Platzierung,
                       Zutat),
          FOREIGN KEY (Rezeptnummer, Platzierung)
                       REFERENCES ABLAUF)
  ```

- Eigentlich gibt es keinen Grund, warum die Tabellennamen ganz groß geschrieben sind, aber Spaltennamen groß/klein.

  PostgreSQL wandelt sowieso alles in Kleinbuchstaben um, solange man nicht `"..."` verwendet. Aber wenigstens in der Dokumentation will man es natürlich schön (und einheitlich) schreiben. Etwas Geschmackssache.
  ```

Präsenzaufgabe: Dritte Lösung

- Rezept(RID, Bezeichnung, Schwierigkeit, Dauer)
- Zutat(ZID, Name)
- Zubereitung((RID → Rezept, ZID → Zutat, Schritt, Menge, Text)
- Man hat so pro Zutat in einem Schritt einen Text, nicht pro Schritt (des Rezepts).
  Wie will man die Texte bei mehreren Zutaten ordnen?

  Z.B., um das Rezept zu drucken.
- Text zu Schritten ohne Zutat gar nicht speicherbar!
- Extra Zutaten-Tabelle: Tippfehler/Schreibvarianten würden vermutlich besser bemerkt, als wenn Zutaten-Namen direkt im Rezept. Eventuell Mengen-Einheit hier (g/Stück)?

  Ist aber auch eine Frage der Benutzerschnittstelle. Größerer Aufwand.

Präsenzaufgabe: Weitere Fragen (1)

- Kann in ABLAUF auch nur die Schrittnummer Schlüssel sein?

  ABLAUF(Schrittnummer, Anweisung, Rezeptnummer → REZEPTE)

  - Im Prinzip ja, auch so haben die Schritte eines Rezepts eine definierte Reihenfolge.
  - Aber der erste Schritt hat dann nicht die Schrittnummer 1 (höchstens für ein Rezept), sondern z.B. 1234.
  - Die Schrittnummer muss jetzt ja global eindeutig sein (in der ganzen Tabelle), nicht nur innerhalb eines Rezepts.

- Woher Notation A^* für Schlüssel?

Präsenzaufgabe: Weitere Fragen (2)

- Kann man nicht auch einfach einen künstlichen Schlüssel (fortlaufende Nummer) für jede Relation nehmen?
  - `REZEPTE(RID, Bezeichnung, Schwierigkeitsgrad, Dauer)`
  - `SCHRITTE(SID, Reihenfolge, Anweisung, RID → REZEPTE)`
  - `ZUTATEN(ZID, Zutat, Menge, SID → SCHRITTE)`
- Im Prinzip ja, aber dann werden zusätzliche `UNIQUE`-Schlüssel besonders wichtig: So kann z.B. die gleiche Zutat im gleichen Schritt mehrfach angegeben werden.
- Braucht man in `SCHRITTE` eine extra `Reihenfolge`, wenn man schon die `SID` hat? Redundant?

Präsenzaufgabe: Weitere Fragen (3)

- Geht auch dies?

  Zutaten(Zutat°, Menge,
  (SchrittNr,Nummer) → Schritt)

  - Nein, ein Primärschlüssel-Attribut, das auch Nullwerte erlaubt, ist automatisch ein Syntaxfehler.
  - Wenn ein Schritt keine Zutaten hat, braucht er auch keinen Eintrag in der Zutaten-Tabelle.

Inhalt

Beispiel zu Schlüsseln

- Was könnten hier Schlüssel sein?

R				
A	B	C	D	E
1	10	100	300	4
1	20	100	300	5
1	30	200	400	5

- Aber Vorsicht: Ein möglicher Zustand kann nur zeigen, was nicht Schlüssel sein darf.

  Man kann sich inspirieren lassen, muss aber noch prüfen, ob der Schlüssel von der Anwendung her Sinn macht (darüber ist hier nichts bekannt) und tatsächlich für alle Zustände gelten soll.

- Beachten Sie, dass interessante Schlüssel minimal bezüglich „$\subseteq$“ sein müssen.

Beispiel zu Fremdschlüsseln

- Sie haben die Punkte-Datenbank angelegt, u.a.:

STUDENTEN			
SID	VORNAME	NACHNAME	···
101	Lisa	Weiss	···
102	Michael	Grau	···
103	Daniel	Sommer	···
104	Iris	Winter	···

- Die Tabelle mit den Bewertungen enthält Fremdschlüssel:

  `BEWERTUNGEN(SID` $\rightarrow$ `STUDENTEN, ...)`

- Was würden Sie tun, um zu prüfen, ob Ihr DBMS diesen Fremdschlüssel tatsächlich überwacht?

  In MySQL wurden lange Fremdschlüssel im `CREATE TABLE` akzeptiert (aus Kompatibilitätsgründen), aber nicht überwacht.

Inhalt

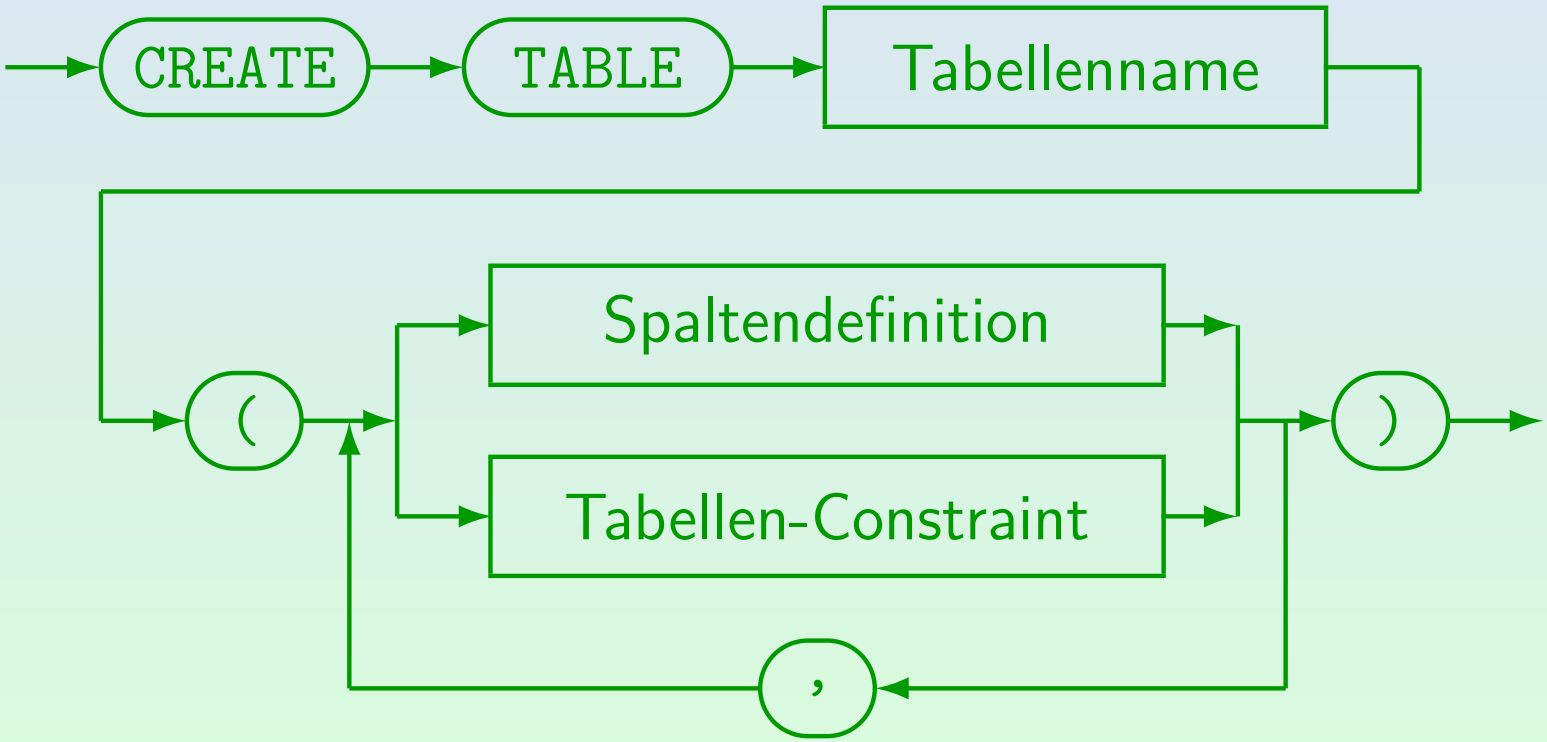
CREATE TABLE: Syntax (1)

- Ziel ist es im Folgenden, die Notation für Syntax-Diagramme zu verstehen, und damit umgehen zu können.
- Dies geschieht anhand von Syntax-Diagrammen für das `CREATE TABLE`-Statement.
- Diese stammen aus der Vorlesung „DB-Programmierung“, und enthalten viele Konstrukte, die wir in der aktuellen Vorlesung nicht besprechen und nicht brauchen.
- Wenn Sie sich die SQL-Referenz zu einem DBMS anschauen, oder den SQL-Standard, sind da noch viel mehr Möglichkeiten, als in beiden Vorlesungen besprochen werden.

  Insofern ist es normal, dass man sich seinen Weg durch Diagramme bahnen muss, und nur die Teile beachten, die einen interessieren.

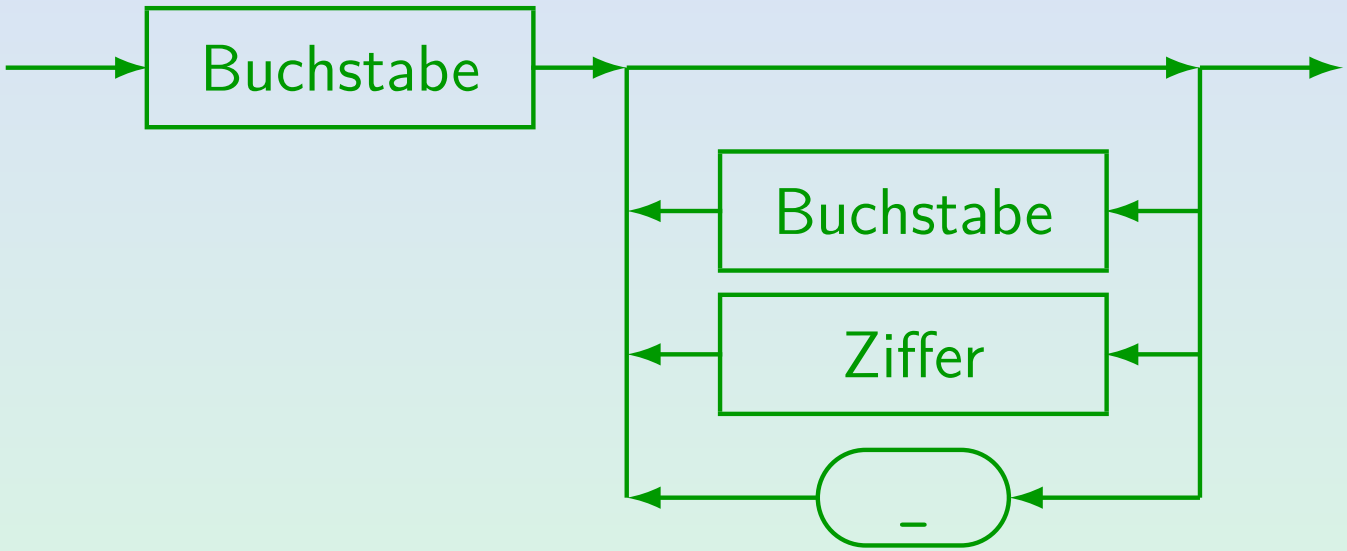
CREATE TABLE: Syntax (2)

CREATE TABLE-Statement:



CREATE TABLE: Syntax (3)

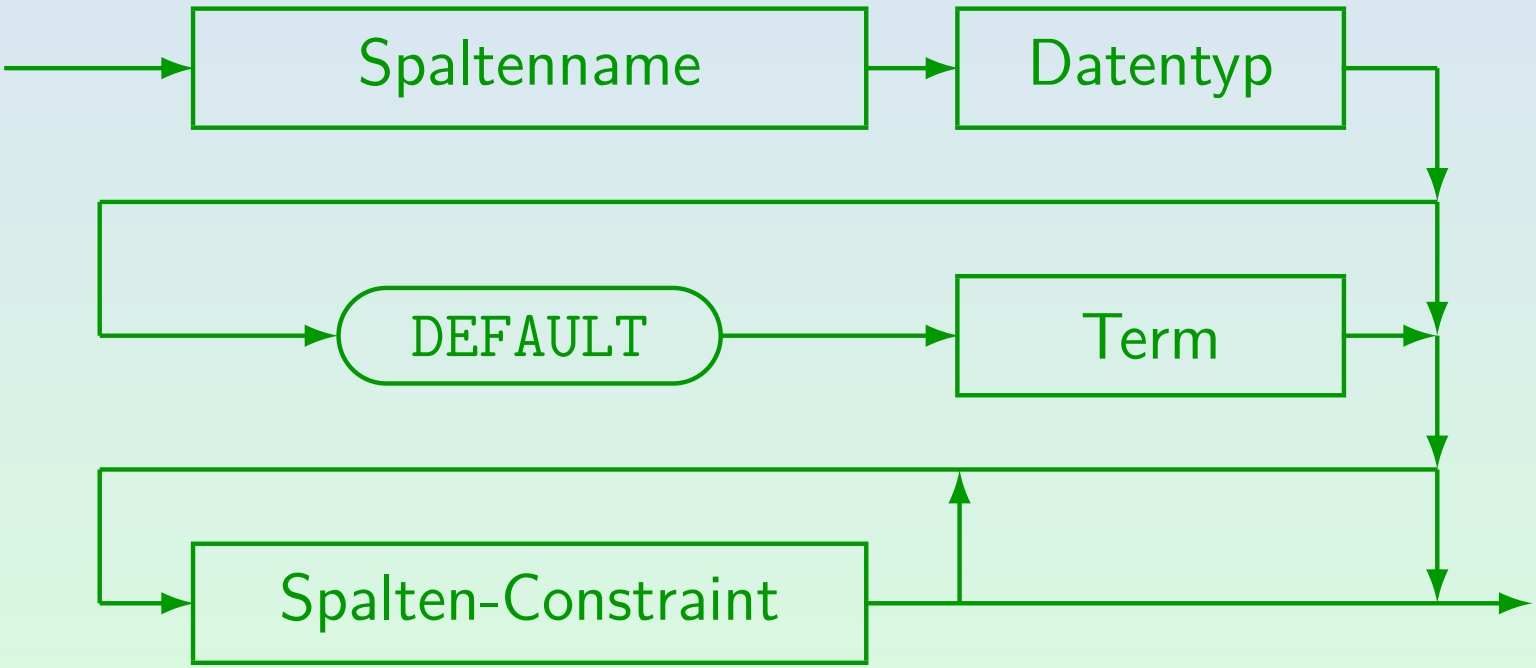
Tabellenname/Spaltenname/Bezeichner:



- Dies ist etwas vereinfacht, SQL hat ja zusätzlich die „Delimited Identifier“, die auch als Tabellenname oder Spaltenname genutzt werden können.
- Diagramme für Buchstabe und Ziffer: Siehe Vorlesung, Kap. 5.

CREATE TABLE: Syntax (4)

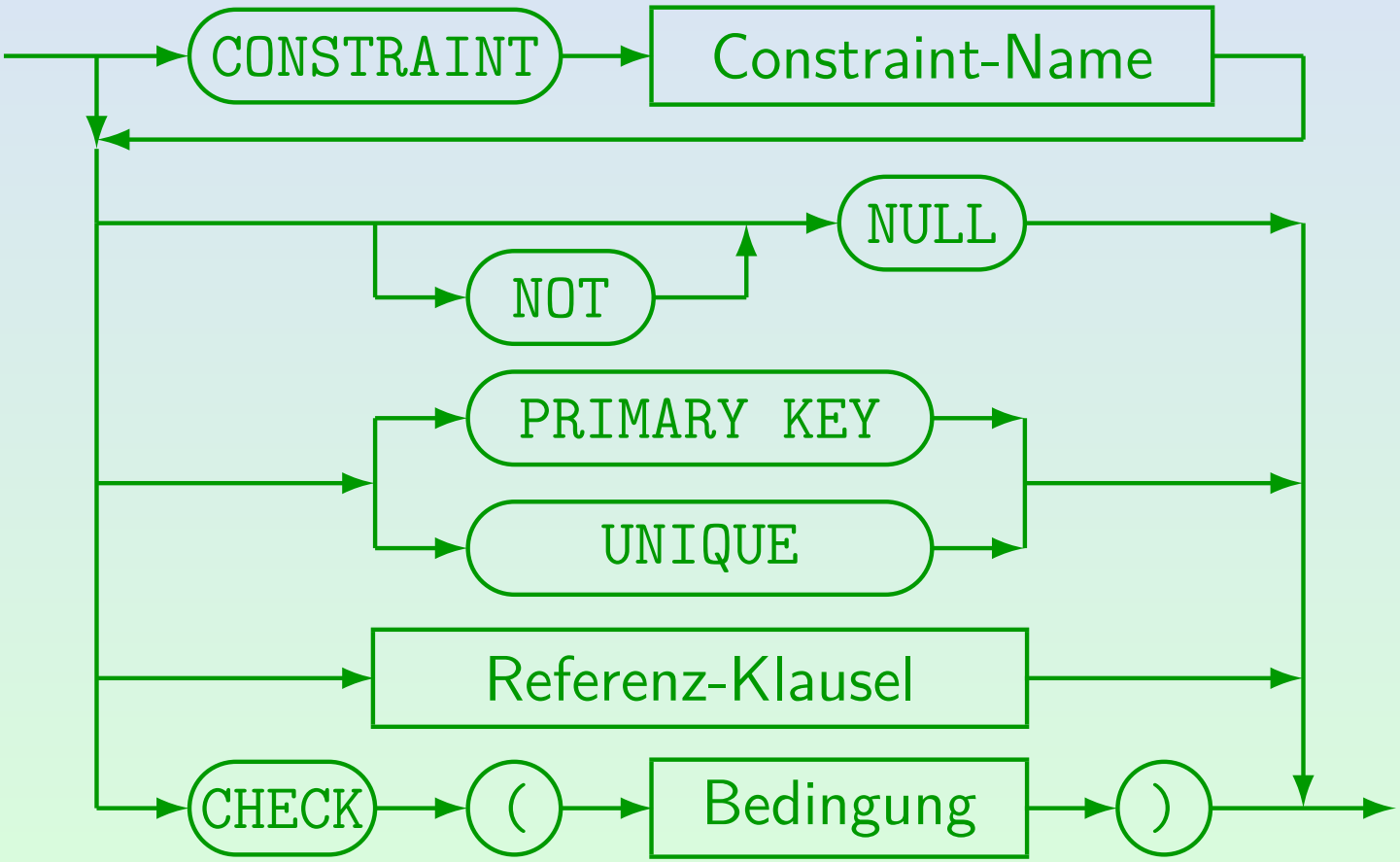
Spaltendefinition:



- Hinweis: `NOT NULL` ist ein Spaltenconstraint.

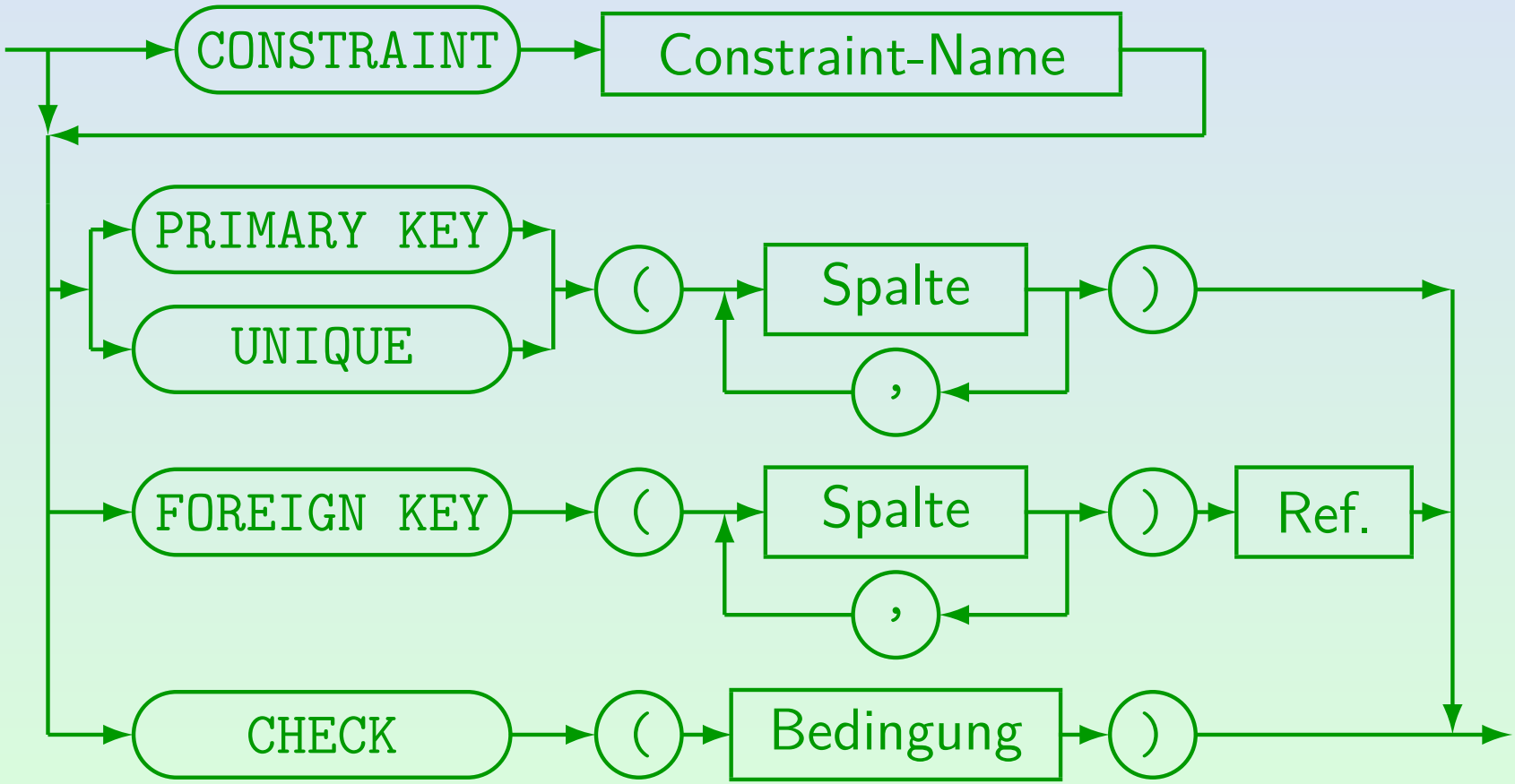
CREATE TABLE: Syntax (5)

Spalten-Constraint:



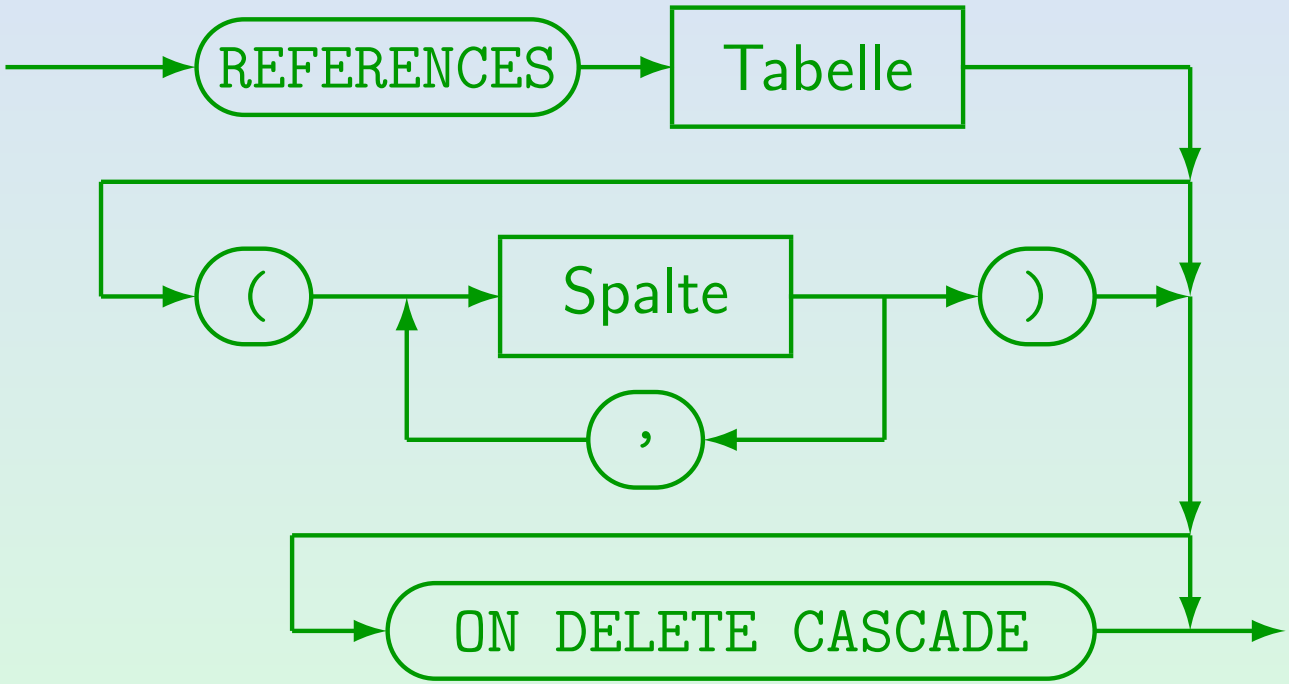
CREATE TABLE: Syntax (6)

Tabellen-Constraint:



CREATE TABLE: Syntax (7)

Referenz-Klausel (Ref.):

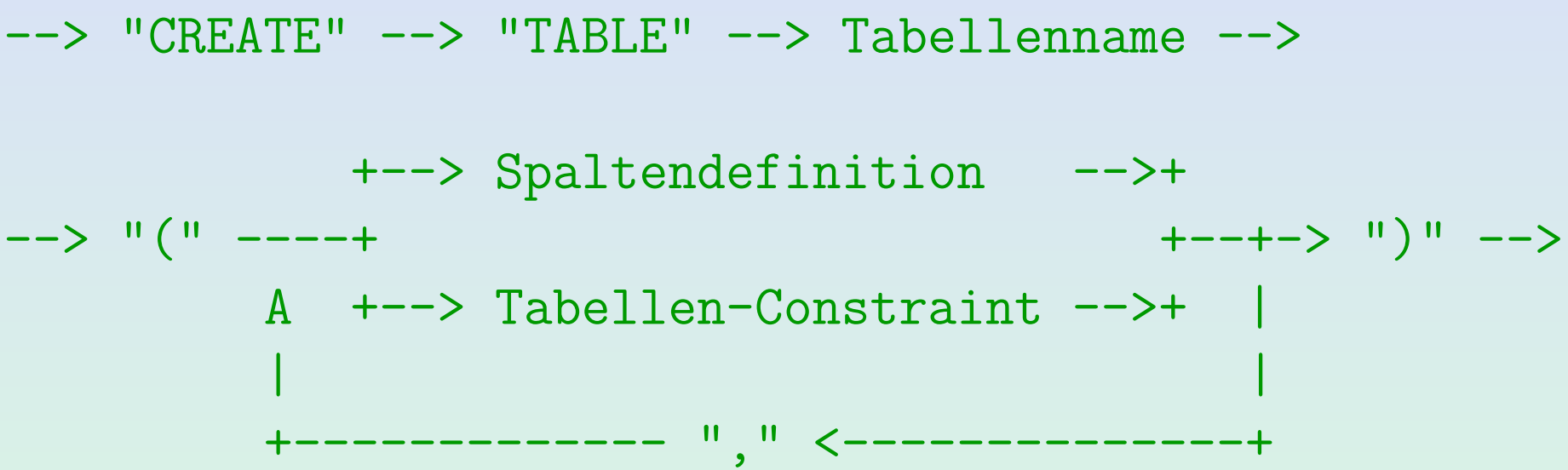


- Hinweis: Wenn man den Primärschlüssel referenziert, braucht man keine Spalten nach der Tabelle anzugeben.

  Nur bei Alternativ-Schlüsseln (`UNIQUE`).

Syntax-Diagramme als ASCII-Text

CREATE TABLE-Statement:



- Mehrere Teildiagramme übereinander sind ok, sie müssen dann alle nacheinander durchlaufen werden.
- Terminalsymbole (die sonst im Oval stehen würden) werden mit "..." markiert, Nichtterminalsymbole (syntaktische Kategorien, Rechteck) ohne "..." geschrieben.

Inhalt

Präsenzaufgabe: Syntaxdiagramme

- Zeichnen Sie Syntaxdiagramme für eine ganz einfache Kommandosprache eines Textadventure-Spiels.
- Typische Kommandos bestehen aus einem Verb und einem Objekt, z.B. “`take lamp`”.

  Verben: “`take`”, “`drop`”, “`examine`”, “`use`”.

  Objekte: “`lamp`”, “`sword`”, “`rope`”.
- Man kann optional einen Artikel verwenden: „`take the lamp`“.
- Ein Verb kann auf mehrere Objekte angewendet werden, durch „and“ getrennt: “`take the lamp and the rope`”.

  Dies ist gleichbedeutend mit: “`take lamp`”, “`take rope`”.
- Sie müssen als Gruppen aus den „Breakout-Rooms“ abgeben.

  Geben Sie als PDF oder JPG ab, oder auch ASCII (.txt).