- Daniel Diekmeier (544835)
- Elias Klemm (546032)
- Robert Koerber (545073)
- Robert Piwonski (544838)

Semesterprojekt im Kurs Datenbanken

Wintersemester 14/15 – Weißwurst-Datenbank

Zusammenfassung

Da wir an einem Datenbank-Modell arbeiten wollten, dass wir wenigstens ein-, zweimal aus Spaß selbst benutzen können, haben wir uns für eine Datenbank entschieden, in der wir unseren Weißwurstkonsum unserer regelmäßig stattfindenden Weißwurstessen abbilden können. Dabei sollen Weißwürste, Senfsorten, Unternehmen wie zum Beispiel Supermärkte oder Fleischer, sowie Nutzerbewertungen der Würste gespeichert werden. Wir hatten im Zuge der Entwicklung mit einigen Meinungsänderungen und technischen Problemen zu kämpfen, die wir aber zu unserer Zufriedenheit lösen konnten.

Zugriffsdaten

- Host: db.f4.htw-berlin.de
- Username: s0544835
- Passwort: Bitte bei Daniel erfragen
- Database: _s0544835__weisswurst
- Port: 3306

Funktionsumfang

Da wir die Datenbank am Ende selbst benutzen wollten, haben sich während der Entwicklungszeit einige Anforderungen unsererseits geändert, sodass der zunächst angedachte grundsätzliche Aufbau nicht vollständig umgesetzt wurde. Allerdings haben wir uns in allen Fällen, in denen eine Vorgabe von unseren Änderungen betroffen war, einen adäquaten Ersatz überlegt.

Es ist nun möglich Weißwürste, Senfsorten und Unternehmen in der Datenbank zu speichern. Außerdem kann die Kombination von Weißwurst und Senf bewertet werden.

Würste und Senfe können mit typischen Angaben (z. B. Name, Preis und Gewicht) gespeichert werden. Würste erhalten noch extra Felder zum Merken wie oft sie bereits gegessen wurden und wann zum letzten Mal. Außerdem können sie einem bestimmten Typ zugeordnet werden, der wiederum mit einer genaueren Beschreibung versehen werden kann. Dem Senf kann ein Schärfegrad in Zahlen zugeordnet werden.

Unternehmen haben ebenfalls die erwartbaren Angaben, (z. B. Ort, Straße oder Telefonnummer). Ein Unternehmen kann sowohl ein Hersteller sein, als auch ein Geschäft, in dem Produkte gekauft werden. Zu jedem Geschäft kann eine Öffnungszeit gespeichert werden, und somit können aktuell geöffnete Geschäfte in einer Sicht angezeigt werden.

Würste und Senfe können schlussendlich einem Unternehmen zugeordnet werden, dass sie entweder verkauft oder diese herstellt. Hersteller werden direkt bei der Weißwurst gespeichert, während Geschäft-Produkt-Beziehungen über Zwischentabellen verwaltet werden.

Konzeption

Angedachter Aufbau

- Eine Tabelle mit Weißwürsten:
  - Die Herkunft der Wurst, also den Schlachter, die Metzgerei, und den Markt, der diese Sorte verkauft, sollen gespeichert werden.
  - Bestimmte Besonderheiten sollen kenntlich gemacht werden: ob die Wurst frisch vom Fleischer stammt, oder im Kühlregal fertig verpackt zu finden ist, oder ob die Wurst in Naturdarm oder in künstlichen Darm gefüllt wurde.
  - Außerdem sollen weitere Bemerkungen als Freitext gespeichert werden. z. B., ob besondere Aromen besonders hervorstechen, oder wo bestimmte Verfügbarkeiten vermerkt werden.
  - Ein Bewertungssystem, das die Wurst in einem einfachen System von 1 bis 5 Sternen bewertet, was unserem subjektiven Geschmacksempfinden entspricht
- Eine Tabelle mit den entsprechenden Händlern, also Fleischereien und Supermärkten und sonstige Geschäfte
  - Adressen und die Öffnungszeiten werden dort vermerkt.
- Eine Tabelle mit Senfsorten:
  - Der vollständige Name
  - Die Art des Senfes, als Auswahl von „Süß“, „Mittelscharf“ und „Scharf“, sowie „Besondere Sorte“, mit der z. B. Frucht-Senf gekennzeichnet werden kann.
  - Wie bei der Weißwurst-Tabelle, soll das Geschäft gespeichert werden.
  - Eine Spalte für sonstige Bemerkungen als Freitext

Die Tabellen der Senfsorten und der Weißwürste sollen derart miteinander verknüpft werden, dass auch eine Bewertung gespeichert wird, die das Zusammenspiel von Senf und Wurst speichert

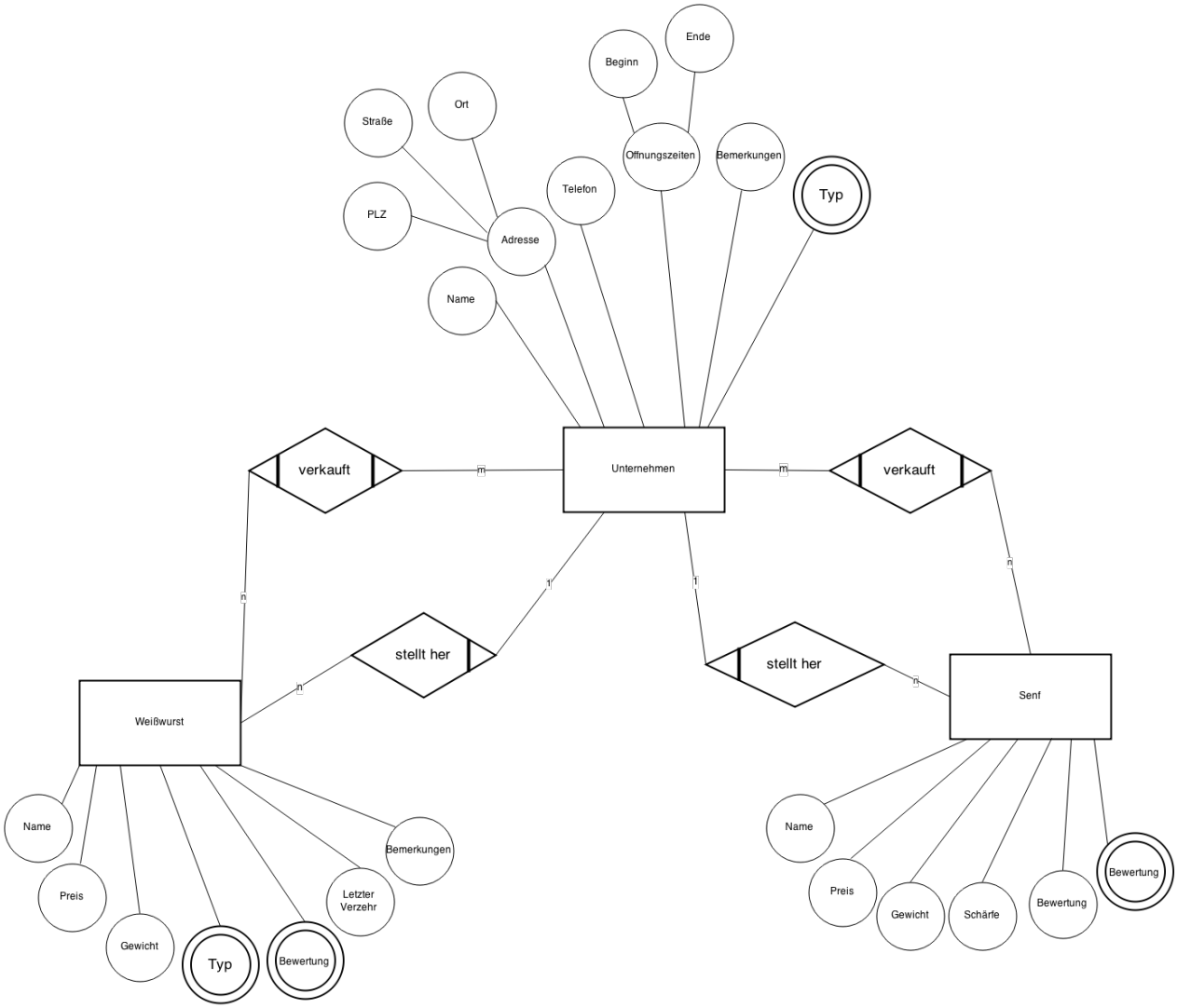
Zu guter Letzt sollen die Tabellen für Senf und Weißwürste mit den Geschäften verknüpft werden, in denen man sie kaufen kann.

Beispielanwendungen

- Wir möchten herausfinden, welcher Senf besonders gut zu einer bestimmten Wurstsorte passt.
- Wir möchten uns nur Senfsorten und Würste anzeigen lassen, die beim gleichen Geschäft zu kaufen sind
- Beim Einkauf auf den letzten Drücker wollen wir herausfinden, welche Geschäfte am Samstag nach 19:00 Uhr offen haben.
- Für die wahren Zuzler kommt natürlich nur echter Darm in Frage, also sollen alle Suchen auf Würste nur in Naturdarm beschränkt werden können.

Konzeptuelles Schema

Als grobe Richtlinie haben wir uns aus dem angedachten Aufbau ein Entity-Relationship-Modell (ERM) gezeichnet. Jedoch wurde das Modell immer wieder kleineren und größeren Änderungen unterworfen (Beispiele unter *Änderungen während der Modellierungsphase*), sodass das ERM vom physikalischen Modell abweicht.



Konzeptuelles Schema

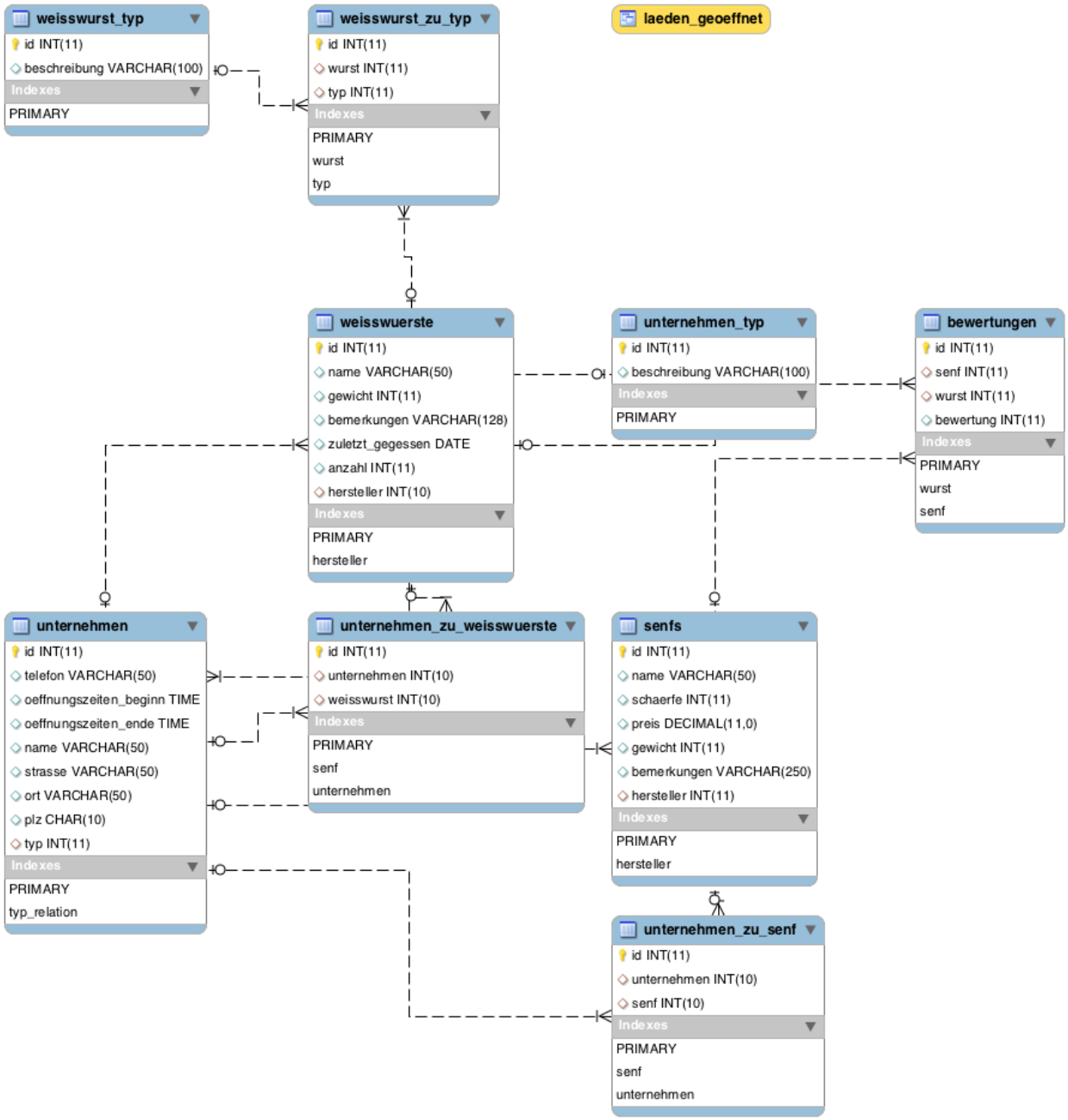
Durchführung

Technische Voraussetzungen

- Laufende MySQL-Installation
- Software zur Verwaltung:
  - MySQL-Befehlszeilenprogramm
  - MySQL Workbench
  - Sequel Pro
  - PHPmyAdmin
- Richtige Berechtigungen auf der Datenbank

Physisches Modell

Das abschließende physische Modell haben wir mit Hilfe der MySQL Workbench erzeugt. Es zeigt alle angelegten Tabellen und deren Verbindungen. Es werden einige Veränderungen und Erweiterungen zum zuvor entworfenen Entity-Relationship-Modell deutlich.



Physisches Modell

Prozeduren

Beispiel `laden_geoeffnet`

Um herauszufinden, ob ein bestimmtes Geschäft noch zur aktuellen Uhrzeit offen hat, haben wir uns eine Prozedur programmiert, die uns eine entsprechende Meldung mit der verbleibenden Öffnungszeit ausgibt.

```
CREATE PROCEDURE `laden_geoeffnet` (in unternehmen_id integer)
BEGIN
    DECLARE o_ende TIME;
    DECLARE o_anfang TIME;
    DECLARE unternehmen_name char(40);
    SELECT name INTO unternehmen_name FROM unternehmen WHERE id = unternehmen_id;
    IF is_open(unternehmen_id) THEN
        SELECT oeffnungszeiten_ende INTO o_ende FROM unternehmen WHERE id = unternehmen_id;
        SELECT CONCAT('Das Unternehmen ', unternehmen_name, ' hat noch ', TIMEDIFF(o_ende, CURTIME()), ' Stu
    ELSE
        SELECT oeffnungszeiten_beginn INTO o_anfang FROM unternehmen WHERE id = unternehmen_id;
        SELECT CONCAT('Das Unternehmen ', unternehmen_name, ' ist geschlossen und öffnet um ', o_anfang);
```

```
    END IF;
END;
```

Prozedur wird mit einer Unternehmens-ID aufgerufen. Zunächst wird über die Prozedur `is_open` (näher Beschrieben im folgenden Abschnitt *Funktionen*) geprüft, ob das Unternehmen offen hat. Hat es offen, wird der Name und die Schließzeit des Unternehmens abgefragt. In der Meldung wird die Schließzeit mit der aktuellen Uhrzeit verrechnet, um eine verbleibende Öffnungsdauer anzugeben. Hat es geschlossen, wird die Öffnungszeit mit der aktuellen Uhrzeit verrechnet um die verbleibende Zeit auszurechnen, bis das Unternehmen wieder öffnet.

Funktionen

Beispiel `is_open`

Diese Funktion gibt Antwort auf die Frage, ob ein Unternehmen zurzeit offen hat.

```
CREATE FUNCTION `is_open` (id_unternehmen INTEGER) RETURNS tinyint(1)
BEGIN
    DECLARE output BOOLEAN;
    DECLARE time_begin TIME;
    DECLARE time_ende TIME;
    SELECT oeffnungszeiten_beginn INTO time_begin FROM unternehmen WHERE id = id_unternehmen;
    SELECT oeffnungszeiten_ende INTO time_ende FROM unternehmen WHERE id = id_unternehmen;
    SET output = false;
    IF time_begin < now() AND time_ende > now() THEN
        SET output = true;
    END IF;
    RETURN output;
END;
```

Diese Funktion wird mit einer Unternehmens-ID aufgerufen und gibt einen boolschen Rückgabewert. Von diesem Unternehmen werden die Öffnungs- und Schließzeiten abgefragt. Der Rückgabewert wird zunächst auf `false` gesetzt. Liegt aber die Öffnungszeit vor und die Schließzeit nach der aktuellen Uhrzeit – was bedeutet, dass das Geschäft gerade offen ist – wird der Rückgabewert `true` gesetzt. Schließlich wird der entsprechende Wert zurückgegeben.

Sicht

Beispiel `laeden_geoeffnet`

Ein Anwendungsfall, der sehr häufig auftritt, ist, alle Läden zu finden, die aktuell geöffnet sind. Dazu erstellten wir eine Sicht wie folgt:

```
CREATE VIEW laeden_geoeffnet AS
    SELECT unternehmen.name AS name
    FROM (unternehmen
        LEFT JOIN unternehmen_typ ON (unternehmen.typ = unternehmen_typ.id)
    ) WHERE (
        (is_open(unternehmen.id) AND
        (unternehmen_typ.beschreibung <> 'Hersteller')
    );
```

In dieser View wird wiederum auf die Funktion `is_open` zurückgegriffen, wobei als zusätzliche Bedingung keine Unternehmen vom Typ “Hersteller” gefunden werden sollen, da explizit nach Verkaufs-Geschäften gefragt ist. Die Tabelle `unternehmen_typ` enthält nur fünf Einträge wobei die vier anderen Beschreibungen neben “Hersteller” die erforderten

Verkaufs-Geschäfte bezeichnen.

Änderungen während der Modellierungsphase

- Die Abstufungen bezüglich dem Schärfegrad von Süß, Mittelscharf, Scharf und Besondere Sorte wurde zu einer numerischen (zwischen 1 und 100) geändert. Dadurch entsteht die Möglichkeit zwei ähnlich scharfe Senfe miteinander zu vergleichen.
- Die Bewertung der Würste und Senfe haben wir zu einer kombinierten Bewertung zusammengelegt, da unter Umständen auch der beste Senf nicht zu bestimmten Wursttypen passt.
- Wir haben uns dazu entschlossen eine eigene Tabelle für die Weiswurst Typen zu definieren, da diese immer wieder vorkommen und die Eindeutigkeit verbessert werden kann.
- Aus Gründen der Ähnlichkeit haben wir die Fleischer und Verkäufer in die Tabelle `Unternehmen` gebündelt. Somit sind diese zentral verwaltbar. Das führte allerdings zu dem Problem, dass eine Wurst von einem Unternehmen hergestellt und von einem weiteren verkauft wird. Daher definieren wir eine Tabelle `Unternehmenstyp` um eine Unterscheidung festzuhalten.

Probleme

- Ursprünglich hatten wir uns für eine Datenbank mit MySQL entschieden, damit wir diese zentral auf einem HTW-eigenen Server speichern und alle Gruppenteilnehmer darauf zugreifen können. Jedoch mussten wir bei einem unserer Treffen Verbindungsabbrüche / Probleme mit dem Aufbau feststellen. Dies ist auf die Limitierung von einzelnen Verbindungen auf die Datenbank zurückzuführen (`Mysql Error 1203`). Bei späteren Treffen trat dieses Problem glücklicherweise nicht mehr auf, was vermuten lässt, dass es sich um ein temporäres Problem im HTW-Rechenzentrum handelte.
- In der Funktion `is_open` werden von dem selben Unternehmen sowohl `oeffnungszeiten_beginn` als auch `oeffnungszeiten_ende` einzeln abgerufen und in Variablen gespeichert, was nahelegt, diese Abfragen lieber in einer Select-Anweisung abzuhandeln. Wir haben einige Zeit darauf angewendet, das umzusetzen, schließlich aber aufgegeben und uns entschieden, dass es nicht allzu schlimm ist, vorrübergehend zwei getrennte Abfragen zu haben.
- Insbesondere, wenn bereits Foreign Key Constraints definiert sind, ist es manchmal ein bisschen schwierig, die Spalten einer Tabelle nachträglich zu verändern – Immerhin sollen ja stets alle Regeln eingehalten werden.
- Beim Arbeiten mit Foreign Keys auf ID-Spalten mussten wir darauf achten, dass die Integer alle `unsigned` sind.

Erweiterungsmöglichkeiten

- Die Tatsache, dass Geschäfte je nach Wochentag unterschiedliche Öffnungszeiten haben können, sollte in einer in einer detailierteren Tabelle reflektiert werden, in der die jeweiligen Zeiten der einzelnen Tage vermerkt werden.
- Es sollte eine Prozedur für ein Weißwurstessen geben, in der die Weißwurst angegeben wird und damit automatisch der dazugehörige Zähler `anzahl` um eins hochgezählt wird, sowie das aktuelle Datum in das Feld `zuletzt_gegessen` eingetragen.
- Die Bewertungen sollten ausgelagert werden, damit jeder von uns seine eigenen Bewertungen anlegen kann.
- Eine große Erweiterung wäre ein Terminkalender, in dem die Daten der nächsten anstehenden Weißwurstfrühstücke eingetragen werden.

Quellen

- http://dev.mysql.com/doc/refman/5.6/en/index.html
- http://downloads.mysql.com/docs/workbench-en.pdf
- http://www.haendlmaier.de/