PROJEKT DOKUMENTATION

Ausgangssituation

Die [redacted] GmbH möchte ihre interne Webanwendung um die Module Artikelverwaltung und Lieferantenverwaltung erweitern.

Diese Dokumentation beschreibt den Umgang mit Maßnahmen zur Erstellung einer Relationalen Datenbankstruktur und stellt die benötigten SQL-Befehle zur Verfügung, die im Entwurf der Ein- und Ausgabeformulare des Projekts verwendet werden sollen.

Aufbau der Datenbank

Ich sehe vor, dass in Zukunft mehr Flexibilität benötigt wird, zum Beispiel um die Rechnungs- und Zahlungsverwaltung hinzuzufügen, entschied ich mich, die Datenbank so zu normalisieren bis jede Tabelle nur noch eine Sache repräsentiert und die Spalten Informationen enthalten, die nur diese eine Sache beschreiben.

Dies ermöglicht nun eine übersichtlichere Verwaltung und Handhabung von die Daten, da sie aus verschiedenen Tabellen zusammengefügt werden können, um die gewünschten Datenergebnisse anzuzeigen und schafft gleichzeitig mehr Flexibilität für die Aufnahme weiterer Tabellen, ohne dass die bestehenden umstrukturiert (geteilt) werden müssen.

Das Vorhandensein vieler Tabellen stellt sicher, dass die Informationseinführung schnell und effizient ist, da nur die Informationen bearbeitet werden müssen, die sich auf den jeweiligen Teil des Prozesses beziehen.

Dies war besonders wichtig für die Tabelle "Artikel", da ständig Artikel ein- und ausgehen und die Tabelle inkonsistent werden würde, wenn alle Informationen zu den Artikeln in einer Tabelle stehen würden (z.B.: Preise, neue Artikel in den Katalog aufnehmen, Preise aktualisieren, oder wenn bereits gelistete Artikel ein- und ausgehen), könnte dies zu menschlich produzierten Fehlern führen.

Die Datenbank setzt sich aus den folgenden 12 Tabellen zusammen:

Schneider

Speichert die Anmeldeinformationen für die Mitarbeiter des Unternehmens.

Mitarbeiter

Hält die persönlichen Daten der Mitarbeiter des Unternehmens.

Vorgelegt von: Sihan Velázquez Silwany Abgabe: 26.05.2021

PROJEKT DOKUMENTATION

Um die Hinzufügungen und Entnahmen von Artikeln aus dem Inventar zu verwalten, wurde für jede Art von Bewegung eine Tabelle erstellt, die einen klaren Überblick darüber gibt, wer, wann und was für jede Bewegung (hinein oder hinaus) in das Inventar bewegt wurde.

Art_heraus

Hier werden die Artikel gespeichert, die aus dem Inventar entnommen werden.

Wenn ein Artikel hier eingefügt wird, wird er automatisch von der Spalte ist_menge in der Tabelle Artikel subtrahiert.

Der Trigger, der automatisch die Artikel aus der Tabelle Artikel subtrahiert:

```
CREATE TRIGGER art_after_art_heraus_insert

AFTER INSERT ON art_heraus

FOR EACH ROW

UPDATE artikel SET ist_menge = ist_menge - new.menge

WHERE art_id = new.art_id;
```

Art_hinzufuegen

Wenn ein Artikel dem Bestand hinzugefügt werden soll, geschieht dies über diese Tabelle, hier wird das Datum (automatisch eingetragen), an dem das Produkt eingefügt wurde, und die Menge festgehalten; ein Trigger fügt den Artikel automatisch dem ist_bestand in der Artikel Tabelle hinzu.

Für diesen Automatismus wurde der folgende Trigger erstellt:

```
CREATE TRIGGER art_after_art_hinzufuegen_insert

AFTER INSERT ON art_hinzufuegen

FOR EACH ROW

UPDATE artikel SET ist_menge = ist_menge + new.menge

WHERE art_id = new.art_id;
```

Artikel

Diese Tabelle speichert die Artikelbezeichnung, die Anzahl der vorhandenen Artikel (ist_menge), die Mindestanzahl der Artikel, die vorhanden sein sollten, d.h. wenn die ist_menge diese Grenze erreicht, fügt ein Trigger den Artikel in die Tabelle not_bestand(bestand in not)ein, wo er verbleibt, bis eine neue Bestellung für diesen Artikel in der Tabelle bestellung eingefügt wird.

Not_bestand

Soll bedeuten „Bestand in Not“. Die Artikel aus der Tabelle Artikel, die in der Spalte ist_menge den Mindestbestand (Spalte melde bestand) erreicht haben, werden per Trigger automatisch in diese Tabelle eingefügt, wo sie bis zum Einfügen des Artikels in die Tabelle Bestellung gehalten werden. Wenn eine Bestellung aufgegeben wird, wird sie automatisch aus dieser Tabelle gelöscht.

Triggers:

```
CREATE TRIGGER notbestand_after_bestellung_insert
AFTER INSERT ON bestellung
FOR EACH ROW
DELETE FROM notbestand WHERE art_id = new.art_id;
```

```
DELIMITER //

CREATE TRIGGER not_after_art_update
  AFTER UPDATE ON artikel
  For EACH ROW
  BEGIN
  IF NEW.ist_menge <= old.meldebestand THEN
  INSERT INTO notbestand (art_ID, life_id) VALUES(NEW.art_id, NEW.lief_id);
  END IF;
END; //

DELIMITER ;
```

Bestellung

Hier werden alle Informationen zu Bestellungen bei den Lieferanten registriert. Zum Beispiel, wenn eine Bestellung bei den Lieferanten aufgegeben wird, die Benutzerinformationen des Mitarbeiters und das Datum (automatisch eingetragen), an dem die Bestellung gemacht wurde.

Wenn eine Bestellung hier eingefügt wird, nimmt ein Trigger den Artikel aus der Tabelle Not_bestand.

PROJEKT DOKUMENTATION

Art_type

Für eine übersichtlichere Verwaltung der Artikel wurde eine Tabelle erstellt, in der die Kategorien, in die die Artikel einsortiert werden können und eine genauere Auswahl der Artikel ermöglichen, z.B. wenn wir sehen wollen, welche Lieferanten Getränke anbieten, oder welche Art von Chips zur Verfügung stehen.

Der Artikeltyp kann in der Artikeltabelle hinzugefügt werden.

Preis

Die Preistabelle hält die Preise fest, diese können in Stufenblöcken eingefügt werden, da die Lieferanten in der Regel je nach Bestellmenge unterschiedliche Preise anbieten.

Zur besseren Lesbarkeit ist diese Tabelle mit der Lieferantentabelle und der Artikeltabelle verknüpft. Dies ermöglicht eine detailliertere Suche, z. B. wenn wir sehen wollen, welcher der Lieferanten den besten Preis für ein Produkt anbietet.

Stat

Als Anforderung aus dem Pflichtenheft wurde eine Statistiktabelle erstellt (stat), in der die Menge der Lieferanten und Artikel angelegt wurde. Sie wird per Trigger jedes Mal automatisch aktualisiert, wenn ein Artikel in der Tabelle Artikel eingefügt oder gelöscht wird, sowie wenn ein Lieferant eingetragen oder gelöscht wird.

Für die Statistiktabelle wurden 4 Trigger gesetzt:

```
CREATE TRIGGER stat_after_artikel_insert

AFTER INSERT ON artikel

FOR EACH ROW

UPDATE stat SET artikelanzahl = artikelanzahl + 1;
```

```
CREATE TRIGGER stat_after_artikel_delete

AFTER DELETE ON artikel

FOR EACH ROW

UPDATE stat SET artikelanzahl = artikelanzahl - 1;
```

```
CREATE TRIGGER stat_after_lieferanten_insert

AFTER INSERT ON lieferanten

FOR EACH ROW
```

```
UPDATE stat SET Lieferantenanzahl = Lieferantenanzahl + 1;


CREATE TRIGGER stat_after_lieferanten_delete

AFTER DELETE ON lieferanten

FOR EACH ROW

UPDATE stat SET Lieferantenanzahl = Lieferantenanzahl - 1;
```

Lieferanten

Diese Tabelle enthält alle Informationen für die Lieferanten, einschließlich der Zeit, die für die Lieferung eines Artikels benötigt wird. Dies hilft dem Beschaffungsteam bei der Berechnung des optimalen Limits (Spalte "Meldebestand" in der Tabelle "Artikel"), das vorhanden sein sollte, um sicherzustellen, dass genug von dem Artikel übrig ist, während auf die Nachlieferung gewartet wird.

Kontakt

Diese Tabelle enthält alle Kontaktinformationen für die Lieferanten.

Authentifizierung

Zum Testen des Prototyps wurde ein Standard-Benutzerkonto mit Passwort angelegt:

Benutzer: prototyp

Kennwort: Proto%typ01

```
CREATE USER ' prototyp'@'localhost' IDENTIFIED BY 'Proto%typ01';

GRANT ALL PRIVILEGES ON schneider . * TO 'prototyp'@'localhost';
```

Die Tabelle Schneider enthält alle Informationen zu Benutzernamen und Passwörtern und ist mit der Tabelle Mitarbeiter verknüpft, die die persönlichen Daten des Mitarbeiters enthält, so dass leicht nachvollzogen werden kann, welcher Mitarbeiter welche Daten geändert oder hinzugefügt hat.

Eingabeformulare

PROJEKT DOKUMENTATION

Neue Artikel eintragen in der Artikel Tabelle

INSERT INTO artikel

(bezeichnung,ist_menge,meldebestand,max_menge,gebindemenge,lief_id,type_id)

VALUES

("Coca Cola Dose",0,5,20,1,1,1);

Eine neue Kategorie für Artikel hinzufügen (Tabelle Art_type).

INSERT INTO art_type(bezeichnung) VALUES("Getraenke"),("Snacks"),

("Alkoholische Getraenke");

Neue Lieferanten eingeben in der Lieferanten Tabelle:

INSERT INTO lieferanten (firma, adresse, telefon, email, lieferzeit)

VALUES

('drinks&snacks','Steinstr. 1 , 12345, Musterstadt',0000123456,'drinks_snacksk@muster-email',5);

Einen Kontakt hinzufügen

Um die Kontaktinformationen für den Lieferanten zu speichern, verwenden Sie den folgenden Code:

INSERT INTO kontakt (vorname,name,telefon,kontakt_email,stellung) VALUES

("Julia","Fischer",145763336,"jb_juliabraun@muster-email.de","Verkaufer");

Einen Artikel zum Inventar hinzufügen

Zum Hinzufügen eines Artikels zum Inventar, machen wir einen Eintrag in der Tabelle Art_hinzufuegen, die Spalte ist_menge in der Tabelle Artikel wird automatisch aktualisiert.

INSERT INTO art_hinzufuegen(art_id,menge) VALUES(1,12);

Artikel aus dem Inventar entnehmen

Vorgelegt von: Sihan Velázquez Silwany Abgabe: 26.05.2021

PROJEKT DOKUMENTATION

Um einen Artikel aus dem Inventar zu nehmen, machen wir einen Eintrag in der Tabelle Art_heraus, die Spalte ist_menge in der Tabelle Artikel wird automatisch aktualisiert.

INSERT INTO art_heraus (art_id,menge) VALUES(3,2);

Ausgabeformulare

Um die benötigten Produkte kaufen zu können, habe ich eine separate Tabelle (notbestand) erstellt. Diese Tabelle wird automatisch über einen Trigger aktualisiert, so dass der Mitarbeiter einfach prüfen kann, ob ein Produkt in der Artikeltabelle sein erlaubtes Limit erreicht hat.

Zur Überprüfung der einfachen Liste der zu bestellenden Artikel:

SELECT * FROM notbestand;

Für eine vollständigere Ansicht mit Artikelbezeichnung und Lieferant und Preis:

SELECT notbestand.art_id, bezeichnung, ist_menge, meldebestand,datum,

preis.min_menge, preis.max_menge, preis,firma as Lieferanten

FROM notbestand

JOIN artikel as art using (art_id)

JOIN lieferanten as lief using (lief_id)

LEFT JOIN preis using(art_id)

ORDER BY preis ASC;

Auslesen der Daten zu einem Artikel mit den dazugehörigen Lieferanten anhand der Artikelnummer, nehmen wir z.B. artikel mit art_id = 1

SELECT bezeichnung, ist_menge, meldebestand, max_menge, gebindemenge,firma as lieferanten

from artikel as art

join lieferanten as lief on art.lief_id = lief.lief_id

Vorgelegt von: Sihan Velázquez Silwany Abgabe: 26.05.2021

WHERE art_id =1;

Alle Lieferanten nach Produkttyp anzeigen, z. B. alle alkoholfreien Getränke Da wir eine separate Tabelle erstellt haben, um die Kategorien der Artikel zu speichern, können wir z.B. nach allen unseren Lieferanten suchen, die alkoholfreie Getränke verkaufen:

```
SELECT bezeichnung as "Getraenke", ist_menge, meldebestand, max_menge,
gebindemenge,firma as Lieferanten
from artikel as art
join lieferanten as lief on art.lief_id = lief.lief_id
WHERE type_id =1;
```

Das Auslesen der Daten zu einem Lieferanten mit den dazugehörigen Artikeln anhand der Lieferantennummer (z.B lieferante nr. 1)

```
SELECT firma as Lieferanten, adresse, telefon,email,lieferzeit, art_id, bezeichnung
FROM lieferanten as lief
RIGHT JOIN artikel as art on art.lief_id = lief.lief_id
WHERE lief.lief_id = 1;
```

Um alle von einem Lieferanten verkauften Artikel zu prüfen:

```
SELECT  art_id, bezeichnung,firma as lieferanten
FROM artikel as Art
JOIN lieferanten as lief on art.lief_id = lief.lief_id
WHERE lief.lief_id = 1;
```

So zeigen Sie die Statistiktabelle an:

```
SELECT * FROM stat;
```

Auslesen des Passwortes des Benutzers prototyp:

SELECT pass_word FROM schneider WHERE login_user = "prototype";

Prototypentest

Um einige der Funktionalitäten der Datenbank auszuprobieren, habe ich einen kurzen Test gemacht:

Wir werden 10 Einheiten Kartoffelchips heiß hinzufügen, art_id ist 6

Die Befehle:

select* from artikel;

INSERT INTO art_hinzufuegen(art_id,menge) VALUES(6,10);

select * from art_hinzufuegen;

select* from artikel;

Das Ergebnis:

Der Artikel 6 in der Tabelle Artikel wurde um die 10 Einheiten ergänzt, die wir durch Einfügen in die Tabelle art_hinzufuegen, das Datum wurde ebenfalls automatisch hinzugefügt.

Nehmen wir nun 5 Einheiten von Artikel 1.

Die Befehle:

SELECT * FROM art_heraus;

SELECT * FROM artikel;

INSERT INTO art_heraus (art_id,menge) VALUES(1,5);

SELECT * FROM art_heraus;

select* from artikel;

Das Ergebnis:

Um die Meldebestandsgrenze auszuprobieren, nehmen wir 2 Einheiten

Die Befehle:

select * from notbestand;

select * from artikel;

INSERT INTO art_heraus (art_id,menge) VALUES(7,2);

select * from notbestand;

select * from artikel;Das Ergebnis:

Bei einer Bestellung verschwindet der nachzubestellende Artikel aus der Notbestand-Tabelle.

ANLAGE

SQL-ANWEISUNGEN

```
CREATE DATABASE schneider;
USE schneider;

CREATE TABLE mitarbeiter(ma_id INT PRIMARY KEY AUTO_INCREMENT,
vorname varchar (20), name VARCHAR (20));

CREATE TABLE schneider (login_user VARCHAR(10) PRIMARY KEY, gin
ma_id int (11),
pass_word  varchar (20),
FOREIGN KEY (ma_id) REFERENCES mitarbeiter(ma_id));

CREATE TABLE lieferanten (lief_id  INT  PRIMARY KEY AUTO_INCREMENT,
firma VARCHAR(100),
adresse VARCHAR (100),
telefon INT,
email VARCHAR (100),
lieferzeit int);
```

```
CREATE TABLE art_type (
type_id INT AUTO_INCREMENT PRIMARY KEY,
bezeichnung VARCHAR (100));

CREATE TABLE ARTIKEL (art_id INT PRIMARY KEY AUTO_INCREMENT,
bezeichnung VARCHAR(100),
ist_menge INT,
meldebestand INT,
max_menge INT,
gebindemenge INT,
lief_id INT,
type_id INT,
FOREIGN KEY (lief_id) REFERENCES lieferanten(lief_id),
FOREIGN KEY (type_id) REFERENCES art_type(type_id));

CREATE TABLE Bestellung (bestellung_id INT PRIMARY KEY AUTO_INCREMENT,
login_user VARCHAR(10),
art_id INT,
bestell_menge INT,
datum DATE DEFAULT (CURRENT_DATE),
FOREIGN KEY (login_user) REFERENCES schneider(login_user),
FOREIGN KEY(art_id) REFERENCES artikel(art_id));

CREATE TABLE kontakt (kontakt_id INT PRIMARY KEY AUTO_INCREMENT,
lief_id INT,
vorname VARCHAR (20),
name VARCHAR (50),
telefon INT,
kontakt_email VARCHAR(100),
stellung VARCHAR(50),
FOREIGN KEY (lief_id) REFERENCES lieferanten(lief_id));
```

```
CREATE TABLE preis(
id int PRIMARY KEY AUTO_INCREMENT,
lief_id INT,
art_id INT,
min_menge INT,
max_mengge INT,
preis DECIMAL(8,2),
FOREIGN KEY (lief_id) REFERENCES lieferanten(lief_id),
FOREIGN KEY (art_id) REFERENCES artikel(art_id));

 CREATE TABLE stat(Artikelanzahl INT, Lieferantenanzahl INT);

CREATE TABLE art_hinzufuegen(
id INT  PRIMARY KEY AUTO_INCREMENT,
datum DATE DEFAULT (CURRENT_DATE),
art_id INT,
menge INT,
FOREIGN KEY (art_id) REFERENCES artikel(art_id));

CREATE TABLE notbestand (Art_id INT,Lief_id INT);

CREATE TABLE art_heraus
(id INT PRIMARY KEY AUTO_INCREMENT,
art_id INT,
menge INT,
datum DATE DEFAULT (CURRENT_DATE),
FOREIGN KEY(art_id) REFERENCES artikel(art_id));

CREATE TRIGGER stat_after_artikel_insert
AFTER INSERT ON artikel
```

```
FOR EACH ROW
UPDATE stat SET artikelanzahl = artikelanzahl + 1;

CREATE TRIGGER stat_after_artikel_delete
AFTER DELETE ON artikel
FOR EACH ROW
UPDATE stat SET artikelanzahl = artikelanzahl - 1;

CREATE TRIGGER stat_after_lieferanten_insert
AFTER INSERT ON lieferanten
FOR EACH ROW
UPDATE stat SET Lieferantenanzahl = Lieferantenanzahl + 1;

CREATE TRIGGER stat_after_lieferanten_delete
AFTER DELETE ON lieferanten
FOR EACH ROW
UPDATE stat SET Lieferantenanzahl = Lieferantenanzahl - 1;

INSERT INTO stat VALUES(0,0);
INSERT INTO mitarbeiter (ma_id, vorname, name) VALUES (1,"Klaus", "Mueller"),
(2,"Beatrice","Richter"),(3,"Kurt", "Helmig"),(4,"Helga", "Schneider");

INSERT INTO schneider (login_user, ma_id, pass_word) VALUES
("prototype",1,"Proto%typ01"),i
("richter_B_002",2,"Beisp_iel%2;"),
("Helmig_K_003",3,"Beisp_iel%3;"),
("Schneider_H_004",4,"Beisp_iel%4;");

INSERT INTO lieferanten (lief_id, firma, adresse, telefon,email, lieferzeit) VALUES
(1,'Metromark','Steinstr. 1 , 12345, Musterstadt',00001234,'metromark@muster-email',5),
(2,'Extramarkt','Hauptstr. 1 , 12347, Musterstadt',00001235,'varmetromark@muster-email',5),
```

(3,'gunstigmarkt','muellerstr. 1 , 12387, Musterstadt',00001236,'gunstigmarkt@muster-email',5),

(4,'rapidmarkt','muellerstr.1, 12387, Musterstadt', 00001237,'gunstigmarkt@muster-email',8);

INSERT INTO art_type(bezeichnung) VALUES("Getraenke"),("Snacks"),("Alkoholische Getraenke");

INSERT INTO artikel (art_id, bezeichnung, ist_menge, meldebestand, max_menge, gebindemenge, lief_id, type_id)

VALUES(1,"Coca Cola 330 ml Dose",480,240,1200,24,1,1),

(2,"Coca Cola 330 ml Dose",480,240,1200,24,2,1),

(3,"Tasty Bier 330ml Dose",101,100,200,24,1,3),

(4,"Tropical Sun Cassava Chips 0,08 K, 23", 230,24,240,24,4,2),

(5,"High Five Bier",250,240,1200,24,3,3),

(6, "Potato Chips hot",14,10,20,20,1,2),

(7,"Pure drops Wasser- Still, 500ml",27,25,50,12,3,1);

INSERT INTO art_heraus (art_id,menge) VALUES(3,2);

INSERT INTO bestellung (art_id,bestell_menge)VALUES(3,10);

INSERT INTO kontakt (lief_id, vorname, name, telefon, kontakt_email, stellung)

VALUES

(1,"Hans","Schmidt",00012456643,"schmidt_hans@muster-email.de","Verkaufer"),

(2, "Martin","Koch",01455555526,"koch_martin@muster-email.de","Verkaufer"),

(3, "Andreas","Bauer",454782526,"abauer@muter-email.de","Verkaufer"),

(4, "Julia","Braun",145763336,"jb_juliabraun@muster-email.de","Verkaufer");

INSERT INTO lieferanten (lief_id, firma, adresse, telefon,email, lieferzeit)

VALUES

(005,'drinks&snacks','Steinstr. 1 , 12345, Musterstadt',0000123456,'drinks_snacksk@muster-email',5);

```
INSERT INTO bestellung
(login_user,art_id,bestell_menge) VALUES("Helmig_K_0",5,2);

INSERT INTO preis(lief_id,art_id,min_menge,max_menge,preis) VALUES(1,1,1,10,16.50),
(2,2,1,10,15.50),(5,6,1,10,8.50),(1,1,11,20,15),(1,1,21,30,14.50),
(2,2,21,30,13.50),(3,7,1,5,5),(3,7,6,10,4.50);

CREATE TRIGGER art_after_art_heraus_insert
AFTER INSERT ON art_heraus
FOR EACH ROW
UPDATE artikel SET ist_menge = ist_menge - new.menge
WHERE art_id = new.art_id;

CREATE TRIGGER art_after_art_hinzufuegen_insert
AFTER INSERT ON art_hinzufuegen
FOR EACH ROW
UPDATE artikel SET ist_menge = ist_menge + new.menge
WHERE art_id = new.art_id;

CREATE TRIGGER notbestand_after_bestellung_insert
AFTER INSERT ON bestellung
FOR EACH ROW
DELETE FROM notbestand WHERE art_id = new.art_id;

DELIMITER //

CREATE TRIGGER not_after_art_update
   AFTER UPDATE ON artikel
   For EACH ROW
   BEGIN
   IF NEW.ist_menge <= old.meldebestand THEN
```

```
   INSERT INTO notbestand (art_ID) VALUES(NEW.art_id);
   END IF;
END; //

DELIMITER ;
```