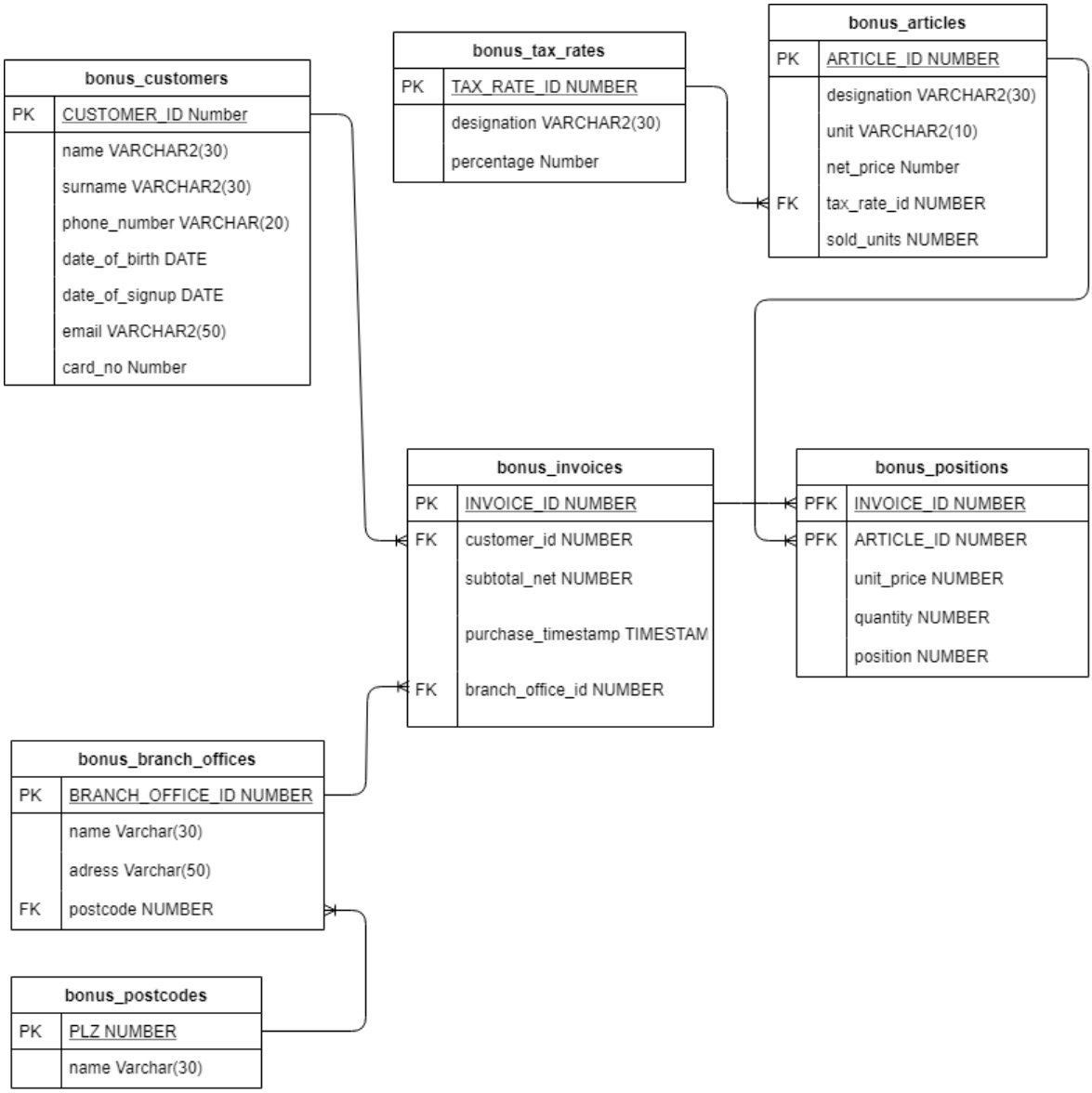
Projekt Bonusclub

Projektbeschreibung

Das Projekt Bonusclub besteht aus den beiden Projektmitgliedern Kurosh Mehdipour Moghaddam und Manuel Szecsenyi. Das Projekt soll ein vereinfachtes Kundenkartensystem sein. Das Kundeneinkaufsverhalten soll erfasst und analysiert werden. In der GUI sollen Kunden, Rechnungen, Artikel und Positionen verwaltet werden. Das Löschen von Kunden oder Rechnungen wird in der GUI aufgrund der gesetzlichen Aufbewahrungspflicht nicht möglich sein. Des Weiteren wird in der GUI das aufrufen der Statistiken möglich sein.



Coding Conventions

Es wurden keine zusätzlichen Coding Conventions erhoben. Es wird auf die der Angabe zurückgegriffen.

Tabellen und Spalten

bonus_customers

Name: bonus_customers

Inhalt: Stamminformationen der Kunden

Indizes: CUSTOMER_ID, card_no

CUSTOMER_ID

Name: Customer_ID

Datentyp: NUMBER

Constraints: NOT NULL

Inhalt: ID des Kunden

name

Name: name

Datentyp: VARCHAR2(30)

Constraints: NOT NULL

Inhalt: Der Vorname eines Kunden

surname

Name: surname

Datentyp: VARCHAR2(30)

Constraints: NOT NULL

Inhalt: Der Nachname eines Kunden

phone_number

Name: phone_number

Datentyp: VARCHAR(20)

Constraints: /

Inhalt: Die Telefonnummer eines Kunden

date_of_birth

Name: date_of_birth

Datentyp: DATE

Constraints: NOT NULL

Inhalt: Das Geburtsdatum eines Kunden

date_of_signup

Name: date_of_signup

Datentyp: DATE

Constraints: NOT NULL

Inhalt: Das Registrierdatum eines Kunden

email

Name: email

Datentyp: VARCHAR2(50)

Constraints: NOT NULL

Inhalt: Die Emailadresse eines Kunden

card_no

Name: card_no

Datentyp: NUMBER

Constraints: NOT NULL

Inhalt: Die Kartennummer eines Kunden

bonus_tax_rates

Name: bonus_tax_rates

Inhalt: Steuersätze der Artikel

Indizes: ?

TAX_RATE_ID

Name: TAX_RATE_ID

Datentyp: NUMBER

Constraints: NOT NULL

Inhalt: Steuersatzid

designation

Name: designation

Datentyp: VARCHAR2(30)

Constraints: NOT NULL

Inhalt: Bezeichnung des Steuersatzes

percentage

Name: percentage

Datentyp: NUMBER

Constraints: NOT NULL

Inhalt: Protzensatz, welcher dann später verrechnet wird

bonus_articles

Name: bonus_articles

Inhalt: Artikelstammdaten

Indizes: ?

ARTICLE_ID

Name: ARTICLE_ID

Datentyp: NUMBER

Constraints: NOT NULL

Inhalt: ID des Artikels

designation

Name: designation

Datentyp: VARCHAR2(30)

Constraints: NOT NULL

Inhalt: Bezeichnung des Artikels

unit

Name: unit

Datentyp: VARCHAR2(10)

Constraints: NOT NULL

Inhalt: Einheit des Produktes (z.B. kg)

net_price

Name: net_price

Datentyp: NUMBER

Constraints: NOT NULL

Inhalt: Nettopreis des Artikels

tax_rate_id

Name: tax_rate_id

Datentyp: NUMBER

Constraints: NOT NULL

Inhalt: ID des Steuersatzes

sold_units

Name: sold_units

Datentyp: NUMBER

Constraints: NOT NULL

Inhalt: Verkaufte Menge

bonus_positions

Name: bonus_positions

Inhalt: Position des Artikels auf der Rechnung

Indizes: ?

INVOICE_ID

Name: INVOICE_ID

Datentyp: NUMBER

Constraints: NOT NULL

Inhalt: ID der Rechnung

ARTICLE_ID

Name: ARTICLE_ID

Datentyp: NUMBER

Constraints: NOT NULL

Inhalt: ID des Artikels

unit_price

Name: unit_price

Datentyp: NUMBER

Constraints: NOT NULL

Inhalt: Stückpreis des Artikels

quantity

Name: quantity

Datentyp: NUMBER

Constraints: NOT NULL

Inhalt: Anzahl des Artikels auf der Rechnung

position

Name: position

Datentyp: NUMBER

Constraints: NOT NULL

Inhalt: Position des Artikels auf der Rechnung

bonus_invoices

Name: bonus_invoices

Inhalt: Rechnung

Indizes: ?

INVOICE_ID

Name: INVOICE_ID

Datentyp: NUMBER

Constraints: NOT NULL

Inhalt: ID der Rechnung

customer_id

Name: customer_id

Datentyp: NUMBER

Constraints: NOT NULL

Inhalt: ID des Kunden

subtotal_net

Name: subtotal_net

Datentyp: NUMBER

Constraints: NOT NULL

Inhalt: Zwischensumme der Rechnung

purchase_timestamp

Name: purchase_timestamp

Datentyp: TIMESTAMP

Constraints: NOT NULL

Inhalt: Gespeicherte Zeit der abgeschlossenen Rechnung

branch_office_id

Name: branch_office_id

Datentyp: NUMBER

Constraints: NOT NULL

Inhalt: ID der Filiale

bonus_branch_offices

Name: bonus_branch_offices

Inhalt: Filiale wo die Artikel verkauft werden

Indizes: ?

BRANCH_OFFICE_ID

Name: BRANCH_OFFICE_ID

Datentyp: NUMBER

Constraints: NOT NULL

Inhalt: ID der Filiale

name

Name: name

Datentyp: VARCHAR(30)

Constraints: NOT NULL

Inhalt: Name der Filiale

adress

Name: adress

Datentyp: VARCHAR(50)

Constraints: NOT NULL

Inhalt: Adresse der Filiale

postcode

Name: postcode

Datentyp: NUMBER

Constraints: NOT NULL

Inhalt: Postleitzahl der Filiale

bonus_postcodes

Name: bonus_postcodes

Inhalt: Postleitzahlen der verschiedenen Filialen

Indizes: ?

PLZ

Name: PLZ

Datentyp: NUMBER

Constraints: NOT NULL

Inhalt: Postleitzahl der Filiale

name

Name: name

Datentyp: VARCHAR(30)

Constraints: NOT NULL

Inhalt: Name des Ortes

Trigger

Das Projekt beinhaltet zwei Trigger.

1. Neuberechnung der Zwischensumme
2. Aktualisieren der verkauften Stückzahl eines Produktes

Neuberechnung der Zwischensumme

Name: `tr_ar_dlu_update_subtotal`

Zweck: Die Zwischensumme einer Rechnung muss bei jedem neuen oder entfernten Artikel neu berechnet werden. Sonst ist sie inkorrekt.

Art: `After Delete, Insert, Update For Each Row`

Tabellen und Attribute: Es werden immer die Menge und Preise multipliziert. Bei einem `Insert` werden sie mit der alten Zwischensumme summiert. Bei einem `Delete` subtrahiert. Bei einem `Update` wird das alte Produkt subtrahiert und das neue addiert. Bei einem Wechsel der Rechnungs ID passiert dies in den jeweiligen Tabellen. **Attribute**: `UNIT_PRICE` und `QUANTITY` jeweils mit dem Präfix `:new`. oder `:old`. **Tabellen** die verändert werden: `BONUS_INVOICES`

Der erste Ansatz war nach jeder Änderung alle Positionen der Rechnung in der Tabelle `BONUS_POSITIONS` durchzugehen und zu addieren. Leider führte das durch den Fehler 04091 nicht zum gewünschten Erfolg. Ein Trigger kann nämlich kein `SELECT`, `DELETE` oder `UPDATE` gegen eine Tabelle ausführen, wenn in dieser der Row Trigger ist. Deshalb dann der Ansatz mit addieren und subtrahieren der Zwischensummen.

Berechnung der verkauften Artikel

Name: `tr_ar_lu_update_sold_units`

Zweck: Um die Verkauften Einheiten eines Produktes dokumentieren zu können. Änderungen bei der Artikelanzahl werden auch berücksichtigt.

Art: `After Insert or Update, for each row`

Tabellen und Attribute: Es wird immer die Anzahl der Artikel zu den Verkauften Einheiten dazugezählt. Bei einem Insert wird die Anzahl des Artikels dem derzeitigen Stand der verkauften Einheiten hinzugefügt. Bei einem Update Befehl wird zuerst geprüft ob die verkauften Einheiten über 0 liegen. Falls dies erfüllt wird, dann wird der alte Wert abgezogen und der neue Wert daraufhin dazu addiert. Wenn dies nicht der Fall ist, wird der Wert nur dazu addiert.

Bei dem Trigger wurden die **Tabellen**: `BONUS_ARTICLES` und `BONUS_POSITIONS` verwendet.

Attribute: `QUANTITY` und `SOLD_UNITS`

Packages

pa_bonus_articles

Entwickler: Kurosh Mehdipour

Beschreibung: Ermöglicht Statistiken der Artikel einzusehen

sp_add_article

Fügt ein Artikel hinzu.

```
PROCEDURE sp_add_article(v_designation_in IN VARCHAR2(30),
             v_unit_in IN VARCHAR2(10),
             n_net_price_in IN NUMBER,
             n_tax_rate_id_in IN NUMBER,
             n_sold_units_in IN NUMBER);
```

Parameter

`v_designation_in IN VARCHAR2(30)` - Bezeichnung des Artikels

`v_unit_in IN VARCHAR2(10)` - Einheit des Produkts

`n_net_price_in IN NUMBER` – Nettopreis des Produktes

`n_tax_rate_id_in IN NUMBER` – Steuersatzid

`n_sold_units_in IN NUMBER` - Verkaufte Einheiten

Rückgabewert

Kein Rückgabewert.

f_get_article_id_rc

Nimmt die ArtikelID entgegen und gibt den Artikel mit dieser ArtikelID aus.

```
function f_get_article_id_rc(n_article_id_in IN NUMBER) RETURN Sys_Refcursor;
```

Parameter

`n_article_id_in IN NUMBER` – ArtikelID

Rückgabewert

`Sys_Refcursor` – Der gewünschte Artikel wird zurückgegeben.

f_get_articles_rc

Gibt allen Artikeln aus.

```
function f_get_articles_rc return Sys_Refcursor;
```

Parameter

Keine Parameter.

Rückgabewert

`Sys_Refcursor` – Alle Artikel werden zurückgegeben.

sp_update_article

Diese Prozedur ermöglicht es die Artikeldaten zu verändern.

```
PROCEDURE sp_update_article(n_article_id_in IN number,
              v_designation_in IN VARCHAR2(30),
              v_unit_in In VARCHAR2(10),
              n_net_price_in IN NUMBER,
              n_sol_units_in IN NUMBER);
```

Parameter

`n_article_id_in IN NUMBER` – ID des Artikels

`v_designation_in IN VARCHAR2(30)` - Bezeichnung des Artikels

`v_unit_in In VARCHAR2(10)` - Einheit des Artikels

`n_net_price_in IN NUMBER` – Nettopreis des Artikels

`n_sol_units_in IN NUMBER` – Verkaufte Einheiten des Artikels

Rückgabewert

Kein Rückgabewert

---

pa_bonus_customers

Package name: `pa_bonus_customer`

Entwickler: Manuel Szecsenyi

Beschreibung: Ermöglicht das Aufrufen von Kundendaten.

sp_add_customer

Fügt einen neuen Kunden hinzu.

```
procedure sp_add_customer(v_name_in IN VARCHAR,
                          v_surname_in IN VARCHAR,
                          v_phone_number IN VARCHAR,
                          d_date_of_birth_in DATE,
                          v_email_in VARCHAR,
                          n_card_no_in Number);
```

Parameter

`v_name_in IN VARCHAR` - Der Vorname des Kunden

`v_surname_in IN VARCHAR` - Der Nachname des Kunden

`v_phone_number IN VARCHAR` - Die Telefefonnummer des Kunden

`d_date_of_birth_in DATE` - Das Geburtsdatum des Kunden

`v_email_in VARCHAR` - Die Email-Adresse des Kunden

`n_card_no_in Number` - Die Kartennummer des Kunden

Rückgabewert

Die Prozedur gibt nichts zurück.

sp_update_customer

Aktualisiert die Daten eines Kunden in der Datenbank

```
procedure sp_update_customer( n_customer_id_in IN NUMBER,
                        v_name_in IN VARCHAR,
                        v_surname_in IN VARCHAR,
                        v_phone_number IN VARCHAR,
                        d_date_of_birth_in DATE,
                        v_email_in VARCHAR,
                        n_card_no_in Number)
```

Parameter

`n_customer_id_in IN NUMBER` - Die ID der Person. Wird zur Identifizierung verwendet.

`v_name_in IN VARCHAR` - Der Vorname des Kunden

`v_surname_in IN VARCHAR` - Der Nachname des Kunden

`v_phone_number IN VARCHAR` - Die Telefefonnummer des Kunden

`d_date_of_birth_in DATE` - Das Geburtsdatum des Kunden

`v_email_in VARCHAR` - Die Email-Adresse des Kunden

`n_card_no_in Number` - Die Kartennummer des Kunden

Rückgabewert

Die Prozedur gibt nichts zurück.

f_get_customer_turnover_n

Nimmt die KundenID entgegen und gibt den Gesamtumsatz den der Kunde in den Geschäften hinterlassen hat zurück.

```
function f_get_customer_turnover_n(n_CUSTOMER_ID_in IN NUMBER) RETURN NUMBER;
```

Parameter

`n_CUSTOMER_ID_in IN NUMBER` - Die ID des Kunden

Rückgabewert

`NUMBER` - Der Umsatz des Kunden. Bei einem Fehler wird `-1` zurück gegeben.

f_get_customers_rc

Gibt alle Kunden zurück.

```
function f_get_customers_rc RETURN SYS_REFCURSOR
```

Parameter

Keine Parameter.

Rückgabewert

`SYS_REFCURSOR` - Alle Kunden

f_get_customers_rc

Gibt einen Kunden zurück mit der angegeben ID.

```
function f_get_customers_rc(n_customer_id_in NUMBER) RETURN SYS_REFCURSOR
```

Parameter

`n_customer_id_in NUMBER` - Die KundenID

Rückgabewert

`SYS_REFCURSOR` - Kunde mit der angegebenen ID

pa_bonus_branch

Package name: `pa_bonus_branch`

Entwickler: Manuel Szecsenyi

Beschreibung: Ermöglicht das Aufrufen von Filialdaten.

f_get_branches

Gibt alle Filialen zurück.

```
function f_get_branches RETURN SYS_REFCURSOR
```

Parameter

Keine Parameter.

Rückgabewert

`SYS_REFCURSOR` - Alle abgespeicherten Filialen.

f_get_branch

Gibt eine die Filiale zurück mit der angegebenen ID.

```
function f_get_branch(n_branch_id_in IN NUMBER) RETURN SYS_REFCURSOR
```

Parameter

`n_branch_id_in NUMBER` - Eine Filialid

Rückgabewert

`SYS_REFCURSOR` - Die Filiale mit der angegebenen ID.

---

pa_bonus_invoices

Package name: `pa_bonus_invoices`

Entwickler: Manuel Szecsenyi

Beschreibung: Hinzufügen und aktualisieren von Rechnungen ist mit diesem Package möglich. Es können auch Artikel hinzugefügt und gelöscht werden. Aber Rechnungsdaten können abgerufen werden.

sp_add_invoice

Fügt eine neue Rechnung hinzu. Das System ermittelt selbständig die nächste Rechnungsnummer und speichert den aktuellen Zeitstempel ab. Die Rechnungen sind leer und haben Anfangs einen Nettobetrag von Null (0€).

```
PROCEDURE sp_add_invoice(
        n_customer_id_in IN NUMBER,
        n_branch_office_id_in IN NUMBER,
        )
```

Parameter

`n_customer_id_in IN NUMBER` - Der Kunde zu dem die Rechnung abgespeichert werden soll.

`n_branch_office_id_in` - Die Filial-ID in der der Einkauf getätigt wurde.

Rückgabewert

Kein Rückgabewert

sp_update_invoice

Aktualisiert die wichtigsten Metainformationen einer Rechnung. Die Rechnungs-ID wird nicht verändert.

```
PROCEDURE sp_update_invoice(
        n_invoice_id_in IN NUMBER,
        n_customer_id_in IN NUMBER,
        d_purchase_timestamp_in IN TIMESTAMP,
        n_branch_office_id_in IN NUMBER
        )
```

Parameter

`n_invoice_id_in` - Die Rechnungs-ID der Rechnung welche geändert werden soll. Die ID kann nicht verändert werden.

`n_customer_id_in` - Der Kunde zu dem die Rechnung gehört.

`n_purchase_timestamp_in` - Der Timestamp des Einkaufs.

`n_branch_office_id_in` - Die Filial-ID in der der Einkauf getätigt wurde.

Rückgabewert

Kein Rückgabewert

sp_add_article

Fügt einen Artikel mit der angegebenen Stückzahl zu einer Rechnung hinzu. Es wird der im Artikel hinterlegte Preis verrechnet.

```
PROCEDURE sp_add_article(
        n_invoice_id_in IN NUMBER,
        n_article_id_in  IN NUMBER,
        n_quantity_in  IN NUMBER
        )
```

Parameter

`n_invoice_id_in` - Die Rechnungs-ID der Rechnung.

`n_article_id_in` - Die Artikel-ID eines Artikels

`n_quantity_in` - Wie viel Stück des Produkte soll verrechnet werden?

Rückgabewert

Kein Rückgabewert.

sp_delete_article

Der angegebene Artikel auf der Rechnung wird entfernt. Die Rechnungspositionen werden nachgerückt.

```
PROCEDURE sp_delete_article(
        n_invoice_id_in IN NUMBER,
        n_article_id_in  IN NUMBER
        )
```

Parameter

`n_invoice_id_in` - Die Rechnungs-ID der Rechnung.

`n_article_id_in` - Die Artikel-ID eines Artikels

Rückgabewert

Kein Rückgabewert.

f_get_invoices_rc

Gibt alle Rechnungen zurück.

```
function f_get_invoices_rc RETURN SYS_REFCURSOR
```

Parameter

Keine Parameter.

Rückgabewert

`rc_cursor_out` - Alle vorhandenen Rechnungen

f_get_invoice_rc

Gibt die Metadaten einer Rechnung zurück.

```
function f_get_invoice_rc(n_invoice_id_in in NUMBER) RETURN SYS_REFCURSOR
```

Parameter

`n_invoice_id_in` - Die Rechnungs-ID der Rechnung.

Rückgabewert

`` `rc_cursor_out `` - Eine Rechnung mit der angegebenen ID

f_get_invoice_articles_rc

Gibt alle Positionen einer Rechnung zurück

```
function f_get_invoice_articles_rc(n_invoice_id_in in NUMBER) RETURN
SYS_REFCURSOR
```

Parameter

`n_invoice_id_in` - Die Rechnungs-ID der Rechnung.

Rückgabewert

`rc_cursor_out` - Alle vorhandenen Positionen

---

pa_bonus_stats

Package name: `pa_bonus_stats`

Entwickler: Manuel Szecsenyi

Beschreibung: Mit diesem Package können verschiedene Statistken ausgelesen und zurückgesetzt werden.

f_get_most_bought_article_rt

Gibt den Artikel zurück mit dem größten `sold_units` Attribut. Zählt **nicht** die Häufigkeit eines Artikels in den Positionen. Diese Statistik kann mit dem zurücksetzen der verkauften Einheiten eines Produktes verändert werden.

```
function f_get_most_bought_article_rt RETURN BONUS_ARTICLES%ROWTYPE;
```

Parameter

Keine Parameter

Rückgabewert

`rt_article_row_out` - Ein Eintrag aus der Tabelle Artikel.

f_get_customer_with_highest_turnover_n

Gibt den Kunden mit dem größten Umsatz zurück.

```
function f_get_customer_with_highest_turnover_rt RETURN NUMBER;
```

Parameter

Diese Funktion nimmt keine Parameter entgegen.

Rückgabewert

`n_customer_id_out NUMBER` - Die Id des Kunden mit dem größten Umsatz.

ps_reset_sold_units

```
PROCEDURE ps_reset_sold_units(
        n_article_id_in IN NUMBER
        )
```

Parameter

`n_article_id_in` - Der Artikel welcher zurückgesetzt werden soll.

Rückgabewert

Kein Rückgabewert.

Description

Setzt die `sold_units` des Artikels auf null (0).

Views

bonus_customer_statistic

Die View `bonus_customer_statistic` zeigt die Anzahl der Einkäufe und den gesamten Umsatz des Kunden in Netto.

Name: `bonus_customer_statistic`

Inhalt:

- Kunden Vor und Nachname
- Kartennummer
- Anzahl der Rechnungen
- Anzahl Nettoumsatz

Basisrelationen: `bonus_customers , bonus_invoices`

bonus_customer_statistics

Name Artikelstatistik

Inhalt:

- ArtikelID
- Bezeichnung
- Verkaufte Mengen
- Gebrachten Nettoumsatz
- Meistverkauft in PLZ

Basisrelationen: `bonus_articles, bonus_positions, bonus_invoices, bonus_branch_office, bonus_postcodes`

Um die View zu erstellen wurden zuerst kleinere Selects geformt. Um die Anzahl der verkauften Einheiten einzusehen wird nur auf die Artikeltabelle geschaut, der Artikelumsatz wird in den Positionen nachgeschlagen da dort der beim Verkauf aktuell Preis hinterlegt ist. Zuletzt mithilfe der `STATS_MODE` Funktion und der Gruppierung nach Artikel ID gelesen in welchen Bezirk das Produkt am häufigsten verkauft wurde. Ein `Select` umschließt die Subselects und formt die endgültige View.

Indizes

Auf welche Attribute Indizes gelegt wurde ist unter "Tabellen und Spalten" ersichtlich.

GUI

Technik

Für die GUI wurde das PHP Framework Laravel gewählt. Die Funktionalität bietet Laravel, die Gestaltung wird vom Framework Bootstrap übernommen.

Damit auf die Oracle Datenbank zugegriffen werden kann musste die PHP-Erweiterung OCI8 installiert werden. Um das Framework voll nutzen zu können wurde ein Driver für Oracle verwenden der hier zu finden ist (https://github.com/yajra/laravel-oci8). Mit diesem war es nun möglich einfacher auf die Datenbank zuzugreifen.

Es wird nun mehr auf die Interaktion mit der Datenbank eingegangen und weniger darauf wie das Framework und der Seitenaufbau funktioniert.

Starten der GUI

Um die GUI zu verwenden muss PHP in der Kommandozeile aufrufbar und die Oracle OCI8 installiert sein. Zum Starten navigiert man in den Hauptfolder und gibt folgendes in der Kommandozeile ein:

```
php artisan serve
```

Nach kurzem warten erscheint der Link unter dem die Webapp gestartet werden kann.

Die Verbindung zu Datenbank ist bereits eingestellt.

Struktur

Wichtig für das Projekt waren folgende Files:

- `routes/web.php` - Hier sind alle verfügbaren Routen gelistet. Diese Verweisen beim Aufruf auf den jeweiligen Controller
- `app/Http/Controllers` - Hier sind alle Controllers gelistet mit den Funktionen der Webapp. Ein Controller gibt eine Ansicht (View) zurück. Diese finden sich unter
- `resources/views/` - Hier sind alle verfügbaren Views aufgelistet.

Beispiel für die Interaktion:

Im folgenden Beispiel wird das Package `pa_bonus_customers` mit der Prozedur `sp_add_customer` näher vorgestellt. Die Daten für die Prozedur kommen mittels POST an den Controller der folgenden Code ausführt:

```
public function add(){

        $data = request();

        $procedureName = 'pa_bonus_customers.sp_add_customer';

        $bindings = [
            'v_name_in' => $data["name"],
            'v_surname_in' => $data["surname"],
            'v_phone_number' => $data["phone_number"],
            'd_date_of_birth_in' => $data["date_of_birth"],
            'v_email_in' => $data["email"],
            'n_card_no_in' => 1
        ];

        $result = DB::executeProcedure($procedureName, $bindings);

        return view("customer.add", ['db_result' => $result]);
    }
```

Mit der `function request()` werden die Daten in ein Array übergeben. Wir sichern uns den Namen der Prozedur vorab in eine Variable um den Code übersichtlicher zu gestalten. Danach bauen wir unsere Bindings auf. Wir verwenden hier die gleichen Namen wie die Prozedur `IN` Variablen. Als Kartennummer haben wir uns zu Demonstrationszwecken für die 1 entschieden. Es wäre möglich unterschiedliche Kartennummer herauszugeben. Im nächsten Schritt wird der Prozedurname und die Bindings der Funktion `executeProcedure()` übergeben. Sie gibt einen Boolean Wert zurück. Wahr wenn die Prozedur erfolgreich durchgeführt wurde. Zu guter Letzt zeigt der Controller die View `add.customer` an und gibt ein Array als Parameter mit (notwendig). Der User bekommt also nach seinem Klick auf “Abschicken” die gleiche Seite angezeigt, allerdings mit einem leeren Formular und mit einer Erfolgsmeldung über das Abspeichern des Kunden.