

Änderungshistorie an der Fakturama Datenbank

Vorwort

Fakturama bietet in der aktuellen Version keine Möglichkeit Bestellungen bei Lieferanten zu Erzeugen oder den Wareneingang zu automatisieren, da es nicht als Warenwirtschaftssystem gedacht ist. Jegliche Bestandsänderungen müssen in der Box-Variante händisch an den Artikeln erfolgen.

Da Lagerbestandsänderungen durch Verkäufe, je nach eigener Vorgabe, erst bei Ausdruck eines Lieferscheins oder Rechnung durchgeführt werden, haben wir einen Einsprungspunkt gefunden.

Da wir schon am Anpassen sind, erschlagen wir gleich verschiedene Anforderungen

Inhaltsverzeichnis

Änderungshistorie an der Fakturama Datenbank	1
Vorwort.....	1
Änderungen an Lagerbestand oder Artikelnummer	3
FKT_PRODUCT_CHANGE.....	3
Änderungen an der Tabelle FKT_PRODUCT	3
FKT_PRODUCT_AFTER_UPDATE	4
Erzeugen einer Inventurliste	5
Die Prozedur “sp_FKT_BOOK_INVENT”	6
sp_FKT_BOOK_INVENT	6
Erfassung von Lieferantenbestellungen.....	7
Anpassungen und Ergänzungen für Lieferantenbestellungen	8
Änderungen an der Tabelle FKT_DOCUMENT	8
FKT_DOCUMENT_AFTER_INSERT	8
FKT_DOCUMENT_AFTER_UPDATE	8
Formular offener Lieferantenbestellungen.....	9
Die Prozedur “sp_FKT_BOOK_DELIVERY”	10
sp_FKT_BOOK_DELIVERY.....	10

Änderungen an Lagerbestand oder Artikelnummer

Um Änderungen an Lagerbeständen nachvollziehen zu können, wurde hierfür eine neue Tabelle **FKT_PRODUCT_CHANGE** erstellt. Diese wird durch einen Trigger, der jedoch nur bei Änderungen der Felder **ITEMNUMBER** und **QUANTITY** greift, aus der Tabelle **FKT_PRODUCT** gefüllt.

Die Tabelle **FKT_PRODUCT_CHANGE** wird mit dem folgenden SQL-Statement erzeugt:

FKT_PRODUCT_CHANGE

```
CREATE TABLE `FKT_PRODUCT_CHANGE` (  
  `ID` bigint NOT NULL AUTO_INCREMENT,  
  `PRODUCT_ID` bigint NOT NULL,  
  `MODIFIED` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP,  
  `AUTHOR` varchar(255) NOT NULL,  
  `ITEMNUMBER_OLD` varchar(255) DEFAULT NULL,  
  `ITEMNUMBER_NEW` varchar(255) DEFAULT NULL,  
  `QUANTITY_OLD` double DEFAULT NULL,  
  `QUANTITY_NEW` double DEFAULT NULL,  
  PRIMARY KEY (`ID`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

Änderungen an der Tabelle FKT_PRODUCT

An der Struktur der Tabelle wurde eine Änderungen vorgenommen, damit der Lagerstand bei einem neuen Artikel nicht NULL ist. Dies hat bei der Erfassung von Lagerzugängen zu Problemen geführt.

Die wird durch das SQL-Statement erreicht.

```
ALTER TABLE `fakturama_prod`.`FKT_PRODUCT`  
CHANGE COLUMN `QUANTITY` `QUANTITY` DOUBLE NULL DEFAULT 0;
```

Für die Nachvollziehbarkeit von Bestandsänderungen an Artikel wurde dann ein After-Update-Trigger eingefügt, der die neu erstellte Tabelle **FKT_PRODUCT_CHANGE** füllt.

FKT_PRODUCT_AFTER_UPDATE

```
CREATE TRIGGER `FKT_PRODUCT_AFTER_UPDATE` AFTER UPDATE ON
`FKT_PRODUCT` FOR EACH ROW BEGIN
    IF OLD.ITEMNUMBER <> NEW.ITEMNUMBER THEN
        INSERT INTO FKT_PRODUCT_CHANGE (PRODUCT_ID, AUTHOR,
ITEMNUMBER_OLD, ITEMNUMBER_NEW)
VALUES (OLD.ID, SUBSTRING_INDEX(USER(), '@', 1), OLD.ITEMNUMBER,
NEW.ITEMNUMBER);
    END IF;
    IF OLD.QUANTITY <> NEW.QUANTITY THEN
        IF OLD.ITEMNUMBER <> NEW.ITEMNUMBER THEN
            INSERT INTO FKT_PRODUCT_CHANGE (PRODUCT_ID, AUTHOR,
ITEMNUMBER_NEW, QUANTITY_OLD, QUANTITY_NEW)
VALUES (OLD.ID, SUBSTRING_INDEX(USER(), '@', 1), NEW.ITEMNUMBER,
OLD.QUANTITY, NEW.QUANTITY);
        Else
            INSERT INTO FKT_PRODUCT_CHANGE (PRODUCT_ID, AUTHOR,
ITEMNUMBER_OLD, QUANTITY_OLD, QUANTITY_NEW)
VALUES (OLD.ID, SUBSTRING_INDEX(USER(), '@', 1), OLD.ITEMNUMBER,
OLD.QUANTITY, NEW.QUANTITY);
        END IF;
    END IF;
END
```

Über ein Verwaltungstool für die MySQL-Datenbank erzeugt ein „SELECT
* FROM fakturama_db.FKT_PRODUCT_CHANGE;“ folgende
beispielhafte Ansicht.

ID	PRODUCT_ID	MODIFIED	AUTHOR	ITEMNUMBER_OLD	ITEMNUMBER_NEW	QUANTITY_OLD	QUANTITY_NEW
1	1	2026-02-09 12:22:56	AndreasKrueger	80000001	NULL	1	0
2	2	2026-02-09 13:55:29	AndreasKrueger	80000002	NULL	0	20
3	2	2026-02-09 13:55:40	AndreasKrueger	80000002	NULL	20	0
4	463	2026-02-11 13:54:02	AndreasKrueger	80000351	NULL	3	1
5	463	2026-02-11 13:58:49	AndreasKrueger	80000351	NULL	1	3
6	454	2026-02-11 14:00:58	AndreasKrueger	80000342	NULL	3	4
7	454	2026-02-11 14:01:42	AndreasKrueger	80000342	NULL	4	3
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Abbildung 1 – MySQL Workbench Result

Um nicht direkt in der Datenbank arbeiten zu müssen, wurde ein PHP-Script
(**lager_change.php**) erstellt, dass genau diese Anfrage an die
Datenbank richtet und die Ergebnisse auf einer Webseite ausgibt.

Erzeugen einer Inventurliste

Leider bietet Fakturama auch keine Möglichkeit eine Liste der aktuellen Lagerbestände zu erzeugen.

Daher wurde das PHP-Script **lager_invent.php** erstellt, dass dies nachholt und eine sortierte Liste aller Artikel, deren Lagerbestand größer 0 ist ausgibt. Sortiert wird nach Warengruppe und Artikelbezeichnung.

Lagerbestand

Lageränderungen

Lieferung erfassen

Lieferhistorie

Lagerbestände für eine Inventur						
Int.Art.Nr.	Herst-Art.Nr.	Warengruppe	Bezeichnung	Stand	Bestand soll	Bestand ist
80000194		040 - Instrumente	Kompass, Armgelenk	2026-02-21	1	☐ Erfassen
80000156		070 - Taschen, Planen	MARES Cruise Carpet	2026-02-17	1	☐ Erfassen
80000351		071 - Lampen	Shustar Handlampe	2026-02-17	3	☐ Erfassen
80000350		071 - Lampen	Wurkkos WK20S	2026-02-17	4	☐ Erfassen
80000053		080 - Kleinteile	Bungee 3mm pro Meter	2026-02-17	30	☐ Erfassen
80000106		080 - Kleinteile	Bungee 4mm pro Meter	2026-02-17	20	☐ Erfassen
						☐ Erfassen

Abbildung 2 - Beispielansicht vom Lagerbestand

Über das Eingabefeld „Bestand ist“ und betätigen der Erfassen Schaltfläche, kann der Lagerbestand korrigiert werden. Hierzu wird in der Datenbank die Prozedur **sp_FKT_BOOK_DELIVERY** angesprochen.

Die Prozedur “sp_FKT_BOOK_INVENT”

Mit dem Adressieren der MySQL-Prozedur wird der Lagerbestand der Inventur verarbeitet. Der Aufruf erfolgt aus des dazugehörigen PHP-Sripts.

sp_FKT_BOOK_INVENT

```
CREATE PROCEDURE `sp_FKT_BOOK_INVENT` (  
  IN p_item_id BIGINT, -- DOCUMENTIDEM.ID -> Datensatz ID  
  IN p_item_quantity DOUBLE -- DOCUMENTIDEM.QUANTITY die geliefert  
  wurde  
)  
BEGIN  
  DECLARE v_DB_User VARCHAR(255); -- Datenbank-User  
  DECLARE v_Art_QA DOUBLE; -- FKT_PRODUCT.QUANTITY  
  DECLARE v_Art_IM VARCHAR(255); -- FKT_PRODUCT.ITEMNUMBER  
  
  -- SELECT SUBSTRING_INDEX(USER(), '@', 1) INTO v_DB_User;  
  SET v_DB_User = 'Inventur'; -- Fest auf Inventur gesetzt  
  
  -- Lese aktuell gesetzten Bestand des Artikels  
  SELECT QUANTITY, ITEMNUMBER INTO v_Art_QA, v_Art_IM FROM  
  FKT_PRODUCT WHERE ID = p_item_id;  
  
  -- Schreibe Inventur-Bestand des Artikels  
  UPDATE FKT_PRODUCT  
  SET  
    MODIFIED = curdate(),  
    MODIFIEDBY = v_DB_User,  
    QUANTITY = p_item_quantity  
  WHERE ID = p_item_id;  
  
  -- Manuelles loggen der Veränderung in FKT_PRODUCT_CHANGE  
  INSERT INTO FKT_PRODUCT_CHANGE (PRODUCT_ID, AUTHOR,  
  ITEMNUMBER_OLD, QUANTITY_OLD, QUANTITY_NEW)  
  VALUES (p_item_id, v_DB_User, v_Art_IM, v_Art_QA,  
  p_item_quantity);  
END
```

Erfassung von Lieferantenbestellungen

Der hauptsächliche Grund, sich mit den Interna von Fakturama zu beschäftigen, ist die fehlende Unterstützung der Lagerverwaltung.

Da wir die Funktion **Auftragsbestätigung** nicht benötigen; wir betreiben keinen Web-Shop; haben diese entsprechend angepasst. Statt Kundendaten werden Lieferantendaten (STRG + Mausklick) aus den Personendaten gezogen und die gewünschten Artikel hinzugefügt. Fakturama erzeugt daraus einen neuen Datensätze in der Tabelle **FKT_DOCUMENT**, der die Dokumentenreferenz ist.

Für den Typ Auftragsbestätigung erhält das Feld **DTYPE** den Inhalt „**Confirmation**“. Die ausgewählten Artikel landen, entsprechend der Programmlogik, in der Tabelle **FKT_DOCUMENTITEM**.

Über die Druckfunktion des dazu angepassten Templates wird die Bestellung an den Lieferanten zu Papier gebracht. Damit sind alle erforderlichen Schritte in Fakturama erfolgt.



GDTA Sports-4u - Triq l'Arcipriet Gamri Camilleri - GRB1043 Gharb

Polaris MARCOM GmbH
Polaris MARCOM GmbH
Carl-Zeiss-Str. 3
52477 Alsdorf

Purchase Order		Page 1 of 1
PO Number:	PO-20260209-001	
Date:	09.02.2026	
Customer Number:	47110815	
Payment:		
Please quote the Order number in all correspondence.		

Pos.	QTY	Art.Nr.	Description
1	1		Z-Valve, 1 Port

This Purchase Order is subject to the Buyer's Standard Terms and Conditions of Purchase, which are available at www.gdta-sports-4u.com/generalterms or attached hereto. Any different or additional terms proposed by the Seller are hereby rejected.

Abbildung 3 - Muster einer Bestellanforderung

Damit die nun erfasste Bestellung, nach Wareneingang, verarbeitet werden kann, sind einige Anpassungen an Datenbank sowie eine Schnittstelle zur Verarbeitung erforderlich.

Anpassungen und Ergänzungen für Lieferantenbestellungen

Änderungen an der Tabelle **FKT_DOCUMENT**

Um mit Fakturama erstellten Datensätze als „Bestellungen bei Lieferanten“ nutzen zu können, sind einige Kunstgriffe erforderlich. Für die vorbereitende Erfassung einer Lieferanforderung wird beim Speichern des Auftragsstyps **Auftragsbestätigung** nach den zugehörigen Artikeln gesucht. Über den Trigger **FKT_DOCUMENT_AFTER_INSERT** wird in den Bestellpositionen in der Tabelle **FKT_DOCUMENTITEM** das Feld **DESCRIPTION** geleert. Hier wird später die Lieferscheinnummer eingetragen.

Damit dies auch nach Aktualisierung möglich ist, gibt es einen zweiten Trigger (**FKT_DOCUMENT_AFTER_UPDATE**) der dies auch nach Änderungen erledigt, sofern noch kein Eintrag im Feld **DESCRIPTION** existiert.

FKT_DOCUMENT_AFTER_INSERT

```
CREATE TRIGGER `FKT_DOCUMENT_AFTER_INSERT` AFTER INSERT ON
`FKT_DOCUMENT` FOR EACH ROW BEGIN
    UPDATE FKT_DOCUMENTITEM
    SET DESCRIPTION = ''
    WHERE FK_DOCUMENT = NEW.ID AND NEW.DTYPE = 'Confirmation';
END
```

FKT_DOCUMENT_AFTER_UPDATE

```
CREATE TRIGGER `FKT_DOCUMENT_AFTER_UPDATE` AFTER UPDATE ON
`FKT_DOCUMENT` FOR EACH ROW BEGIN
    DECLARE v_doc_status BIT(1);
    SELECT DELETED INTO v_doc_status
    FROM FKT_DOCUMENT
    WHERE ID = NEW.ID;
    IF v_doc_status=0 THEN
        UPDATE FKT_DOCUMENTITEM
        SET DESCRIPTION = NULL,
            MODIFIED = NULL
        WHERE FK_DOCUMENT = NEW.ID AND NEW.DTYPE = 'Confirmation';
    END IF;
END
```

An der Struktur der Tabelle sind keine Änderungen vorgenommen worden.

Formular offener Lieferantenbestellungen

Ebenfalls durch ein PHP-Script wird eine Webseite erzeugt, die die offenen Bestellungen bei Lieferanten ausgibt. Das Einlesen der Information für das Webformular erfolgt über normale SQL-Abfragen, deren Ergebnisse optisch aufbereitet werden.

In der Übersicht findet sich zu jeder nicht vollständig erledigten Bestellung eine Kopfzeile, mit ID der Bestellanforderung (PO-YYYYMMDD-XXX), sowie Zeilen der jeweiligen Artikelpositionen. Am Ende der Artikelzeile gibt je ein Feld zur Eingabe der Liefermenge sowie der Lieferscheinnummer, die über die Schaltfläche **Erfassen** bestätigt werden kann.

Lieferschein erfassen, Wareneingang Buchen						
PO Nummer	Bestelldatum	Lieferant	Art.Nr	Bezeichnung	Menge	Delivered mit ..
PO-20260209-001	2026-02-09	Polaris MARCOM GmbH, Polaris MARCOM GmbH				
		->	80000147	Z-Valve, 1 Port	2	Lieferschein ID ... Erfassen
		->	80000147	Z-Valve, 1 Port	1	Lieferschein ID ... Erfassen
		->	80000147	Z-Valve, 1 Port	3	Lieferschein ID ... Erfassen
		->	80000147	Z-Valve, 1 Port	3	Lieferschein ID ... Erfassen

Abbildung 4 - Formular zur Lieferschein Verarbeitung.

Nach einer Betätigung des Schalters Erfassen erfolgt eine Prüfung der Parameter, in der die Lieferschein ID eventuell mit einem führenden „LS-“ ergänzt wird, bevor die eindeutige ID des Datensatzes, die Lieferscheinnummer und die zu buchende Menge an die Datenbank zurückgegeben werden.

Die Rückgabe erfolgt durch den Aufruf an die MySQL-Prozedur sp_FKT_BOOK_DELIVERY, die diese drei Werte erwartet.

Die Prozedur “sp_FKT_BOOK_DELIVERY”

Mit dem Adressieren der MySQL-Prozedur werden verschiedene Arbeitsschritte angestoßen. Erforderliche Werte werden aus dem Datensatz der Artikelposition, zur weiteren Nutzung, ausgelesen. Weicht die Erfasste Menge von der Bestellten ab, wird der ursprüngliche Datensatz kopiert und mit der nun noch offenen Anzahl Artikel der Bestellung hinzugefügt. Danach wird die, als empfangen quittierte, Bestellposition und deren Menge im ursprünglichen Datensatz um die Lieferscheinnummer, das aktuelle Datum und den Datenbanknutzer ergänzt. Als Datenbanknutzer wird der eigens dafür angelegte „lieferschein“ identifiziert.

Als nächster Schritte erfolgt die automatische Aktualisierung der Lagerbestände, ohne in Fakturama manuell in den Artikel eingreifen zu müssen.

In der darauffolgenden Prüfung, ob alle Positionen einer Bestellung erfasst sind, erfolgt nach positivem Ergebnis eine Aktualisierung in der Tabelle **FKT_DOCUMENT**. Eine Änderung ist dann, ohne direkt in die Datenbank einzugreifen, nicht mehr möglich, da der Rahmendatensatz zu der Anforderung in **FKT_DOCUMENT** mit einer Markierung versehen wurde, der den Datensatz in Fakturama ausblendet.

Analog zum Modul Erfassung gibt es eine reine Funktion zur Anzeige erfolgter Lieferungen, die als Historie oder Logbuch zu verstehen ist.

sp_FKT_BOOK_DELIVERY

```
CREATE PROCEDURE `sp_FKT_BOOK_DELIVERY` (
  IN p_item_id BIGINT, -- DOCUMENTIDEM.ID -> Datensatz ID
  IN p_item_quantity DOUBLE, -- DOCUMENTIDEM.QUANTITY die geliefert
  wurde
  IN p_item_ls_number longtext -- DOCUMENTIDEM.DESCRPTION ->
  Lieferscheinnummer
)
BEGIN
  DECLARE v_Art DO BIGINT; --
  DOCUMENTIDEM.FK_DOCUMENT -> FK_DOCUMENT.ID
  DECLARE v_Art IM VARCHAR(255); --
  DOCUMENTIDEM.ITEMNUMBER -> FK_PRODUCT.ITEMNUMBER
  DECLARE v_Art ID BIGINT; --
  DOCUMENTIDEM.FK_PRODUCT -> FK_PRODUCT.ID
  DECLARE v_Art QA DOUBLE; --
  DOCUMENTIDEM.QUANTITY -> FK_PRODUCT.QUANTITY
  DECLARE v_Art QC DOUBLE; --
  DOCUMENTIDEM.QUANTITY aus DOCUMENTIDEM.QUANTITY minus
  p_item_quantity
```

```

        DECLARE v_ls_pos INT; --                Anzahl der
        Datensätze die zu FK_DOCUMENT.ID gehören

        DECLARE v_ls_set INT; --                Anzahl der
        Datensätze die zu FK_DOCUMENT.ID gehören und einen "LS-" Eintrag
        haben

        DECLARE v_DB_User VARCHAR(255); --        Angemeldeter SQL
        Datenbank-User

--      SELECT SUBSTRING_INDEX(USER(), '@', 1) INTO v_DB_User;
      SET v_DB_User = Lagereingang; -- Fest auf Lagereingang gesetzt

-- 1. Abfrage des Artikel-Datensatzes aus FKT_DOCUMENTITEM
SELECT
    FK_DOCUMENT,
    ITEMNUMBER,
    FK_PRODUCT,
    QUANTITY
INTO
    v_Art_DO,
    v_Art_IM,
    v_Art_ID,
    v_Art_QA
FROM FKT_DOCUMENTITEM
WHERE ID = p_item_id;

set v_Art_QC = v_Art_QA - p_item_quantity;

-- 2.1 Prüfe ob gefieferte Anzahl kleiner bestellter Anzahl ist
IF v_Art_QA > 1 AND p_item_quantity < v_Art_QA THEN
    -- Order ist größer 1 geliefert aber kleiner als Order

    -- 2.1.1 Dupliziere den Datensatz
    INSERT INTO FKT_DOCUMENTITEM
        (DATEADDED, DELETED, DESCRIPTION, GTIN, ITEMNUMBER,
        ITEMREBATE, ITEMTYPE, MODIFIED, MODIFIEDBY, NAME, NOVAT, OPTIONAL,
        PICTURE, POSNR, PRICE, QUANTITY, QUANTITYUNIT, VALIDFROM, VALIDTO,
        VESTINGPERIODEND, VESTINGPERIODSTART, WEIGHT, FK_VAT, FK_PRODUCT,
        FK_DOCUMENT, SUPPLIERITEMNUMBER)
    SELECT
        DATEADDED, DELETED, DESCRIPTION, GTIN, ITEMNUMBER,
        ITEMREBATE, ITEMTYPE, MODIFIED, MODIFIEDBY, NAME, NOVAT, OPTIONAL,
        PICTURE, POSNR, PRICE, v_Art_QC, QUANTITYUNIT, VALIDFROM, VALIDTO,
        VESTINGPERIODEND, VESTINGPERIODSTART, WEIGHT, FK_VAT, FK_PRODUCT,
        FK_DOCUMENT, SUPPLIERITEMNUMBER
    FROM FKT_DOCUMENTITEM WHERE id = p_item_id;

    -- 2.1.2 Aktualisiere ursprünglichen Datensatz als geliefert
    mit berichtigter Liefermenge
    UPDATE FKT_DOCUMENTITEM
    SET
        DESCRIPTION = p_item_ls_number,
        MODIFIED = curdate(),
        MODIFIEDBY = v_DB_User,
        QUANTITY = p_item_quantity
    WHERE ID = p_item_id;

    -- 2.1.3 und den Lagerbestand aktualisieren

```

```

UPDATE FKT_PRODUCT
SET
    QUANTITY=QUANTITY + p_item_quantity
WHERE ID=v_Art_ID;

-- 2.2 Prüfe ob gelieferte Anzahl gleich bestellter Anzahl ist
ELSEIF p_item_quantity = v_Art_QA THEN
    -- geliefert ist gleich der Order

    -- 2.1.2 Aktualisiere Position als geliefert
    UPDATE FKT_DOCUMENTITEM
    SET
        DESCRIPTION = p_item_ls_number,
        MODIFIED = curdate(),
        MODIFIEDBY = v_DB_User
    WHERE ID = p_item_id;

    -- 2.2.3 und den Lagerbestand aktualisieren
    UPDATE FKT_PRODUCT
    SET
        QUANTITY=QUANTITY + p_item_quantity
    WHERE ID=v_Art_ID;
END IF;

-- 3. Logikprüfung: Zählen der Positionen für das zugehörige
DOCUMENT
    -- Anzahl der Datensätze die zu FK_DOCUMENT.ID gehören
    SELECT COUNT(*) INTO v_ls_pos
    FROM FKT_DOCUMENTITEM
    WHERE FK_DOCUMENT = v_Art_DO;

    -- Anzahl der Datensätze die zu FK_DOCUMENT.ID gehören und einen
    "LS-" Eintrag haben
    SELECT COUNT(*) INTO v_ls_set
    FROM FKT_DOCUMENTITEM
    WHERE DESCRIPTION LIKE 'LS-%' AND FK_DOCUMENT = v_Art_DO;

    -- 4. Wenn alle Positionen den Status 'LS-' haben, Dokument auf
    'Deleted' setzen
    IF v_ls_pos > 0 AND v_ls_pos = v_ls_set THEN
        UPDATE FKT_DOCUMENT
        SET DELETED = 1, MODIFIED = curdate(), MODIFIEDBY
        ='Wareneingang'
        WHERE DTYPE = 'Confirmation' AND ID = v_Art_DO;
    END IF;
END

```