Warenkorbanalyse

Dokumentation Java Scoring Modul

Inhalt

[1 Einleitung 2](#_Toc359234476)

[2 Grundsätzliche Funktionsweise 3](#_Toc359234477)

[2.1 Verwendete Tabellen 3](#_Toc359234478)

[2.2 Aufbau und Verwendung des Java-Scoring-Moduls (JSM) 6](#_Toc359234479)

[2.3 Multithreading 7](#_Toc359234480)

[2.4 Weitere Dokumentation 7](#_Toc359234481)

# Einleitung

Die Warenkorbanalyse soll für die Identifikation häufig gemeinsam konsumierter Warengruppen/Produkte genutzt werden. Ziel ist es, jedem Kunden im Globus Drive Webshop eine möglichst individuell abgestimmte Leistung, in Form von Produktempfehlung, anbieten zu können und dem Kunden einen Mehrwert zu schaffen.

Dazu werden auf Seiten des DWHs Produktvorschlagsregeln extrahiert. Diese werden dann vom Web-Shop verwendet um Kunden in Abhängigkeit ihrer Warenkörbe individuelle Produktvorschläge unterbreiten zu können.

Das Scoring ist als Java-Modul implementiert und kann über eine API in den Web-Shop eingebunden werden. Das Scoring Modul wurde während des Hauptprojektes initial implementiert und wird mit dieser Dokumentation an die Implementierer des Web-Shops übergeben.

Teilnehmer des Hauptprojektes (04.06. – 11.06.2013) sind:

|  |  |  |  |
| --- | --- | --- | --- |
| **Abteilung / Firma** | **Verantwortlichkeit** | **Mitarbeiter** | **E-Mail** |
| Firma SAS | Coaching | Christoph Sieb | christoph.sieb@sas.com |
| Controlling | Datenhaltung, SAS | Desislava Fehringer | d.fehringer@globus.net |
| Marketing | Projektleitung | Andrea Linsler | a.linsler@globus.net |

Weitere Beteiligte:

|  |  |  |  |
| --- | --- | --- | --- |
| **Abteilung / Firma** | **Verantwortlichkeit** | **Mitarbeiter** | **E-Mail** |
| Fachabteilung Drive | Project Owner | Anja Weirich | a.weirich@globus.net |
| Firma UNIC | Implementierungspartner | Jens Saade | jens.saade@unic.com |

# Grundsätzliche Funktionsweise

Bei der Interaktion zwischen den Systemen auf Controlling und auf WebShop Seite findet ein Datenaustausch in beiden Richtungen statt. Allerdings ist der Rückfluss ins DWH (Tracking der Produktvorschläge) nicht Teil dieser Projektphase.

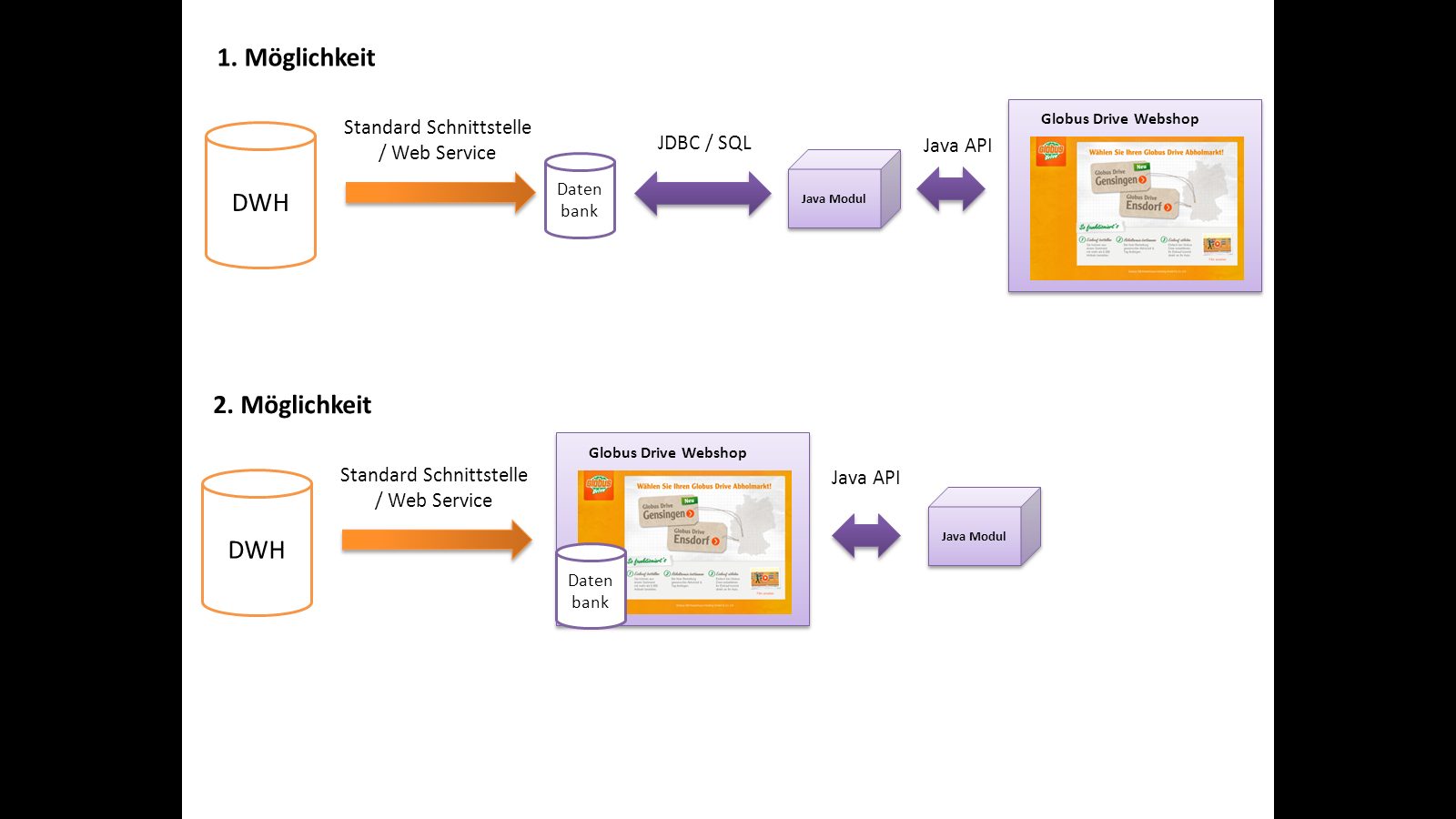


Abbildung 1: Grundsätzliche Funktionsweise des Regel-Scorings

Das Java Scoring Modul (JSM) verwendet mehrere Tabelle, die von DWH Seite zur Verfügung gestellt werden. Diese Tabellen sollten möglichst schnell an den Web-Shop angebunden sein, damit das Scoring in Echtzeit ohne große Zeitverzögerungen möglich ist.

## Verwendete Tabellen

Die Tabellen müssen in eine vom Web-Shop, per JDBC zugreifbare Datenbank geladen werden.

Die folgenden Tabellen werden vom JSM verwendet:

* WEBSHOP\_AUSSCHLUSSLISTE

Tabelle mit einer Spalte (momentan BASIS\_VARIANTE, soll aber auch Produktgruppen enthalten können und muss daher umbenannt werden). Diese Spalte enthält Produkt- und Produktgruppennummern die nicht vorgeschlagen werden sollen. Das JSM lädt diese Tabelle in den Hauptspeicher und verwendet diese Liste als Filter.

* WEBSHOP\_REGELN\_BETRIEBE

Diese Tabelle enthält die Regeln, die auf Betriebsebene extrahiert wurden. Das JSM frägt diese Tabelle via JDBC/SQL ab und verwendet dabei eine Liste mit Produktnummern.

Eine Regel sieht prinzipiell wie folgt aus:

123322536\_754W0020 MB Brötchen Konzessionäre

&

3265735007\_Banane Chiquita Stück

==> 517064007\_BIO MÖHREN 1KG Schale

Die Produkte im Warenkorb eines Kunden werden gegen den Regelkörper (linke Seite von ==>) geprüft, wenn die Regelkonsequenz (rechte Seite von ==>) nicht im Warenkorb ist, handelt es sich um einen potentiellen Produktvorschlag.

Spalten:

* + LHS

Anzahl Items (Produkte / Produktgruppen) im Regelkörper (Left-Hand-Side)

* + RHS

Anzahl Items (Produkte / Produktgruppen) in der Regelkonsequenz (Right-Hand-Side)

* + COUNT

Anzahl Warenkörbe (Transaktionen) in denen alle Produkte der Regel enthalten waren

* + SUPPORT

Relativer COUNT bezogen auf alle betrachteten Warenkörbe

* + SUPLIFT

Bisher nicht verwendete Kennzahl

* + CONF

Confidenz einer Regel

* + LIFT

Lift einer Regel

* + ITEM1(-5)

Das erste Item (Produkt- oder Produktgruppe) der Regel. Eine Regel kann maximal 5 Items enthalten (4 im Regelkörper, 1 in der Regelkonsequenz)

* + LEVEL1(-5)

Das level innerhalb der Produkthierarchie des Items. Produkte auf unterster Ebene haben den Wert 1, Produkte der z.B. der Warengruppe 123322536\_754W0020 haben Wert 2.

* + RULE – Die Regel als 1 String, wie oben im Beispiel angegeben
  + BETRIEB\_ID – Die Betrieb ID (Markt) für die diese Regel gilt
  + IMPORTANCE – Sortierkriterium um die relevanz einer Regel gegenüber anderen Regel zu bestimmen.
* WEBSHOP\_REGELN\_KUNDEN

Gleich zur Tabelle WEBSHOP\_REGELN\_BETRIEBE, allerdings auf Kundenebene und daher anstatt der Spalte BETRIEBE\_ID die Spalten KD\_IDX (Kundenindex)

* WEBSHOP\_MANUELLE\_REGELN

Gleich zur Tabelle WEBSHOP\_REGELN\_BETRIEBE, allerdings gelten diese manuell erstellten Regeln für alle Kunden und alle Betriebe

* WEBSHOP\_REGELN\_HIERARCHIE

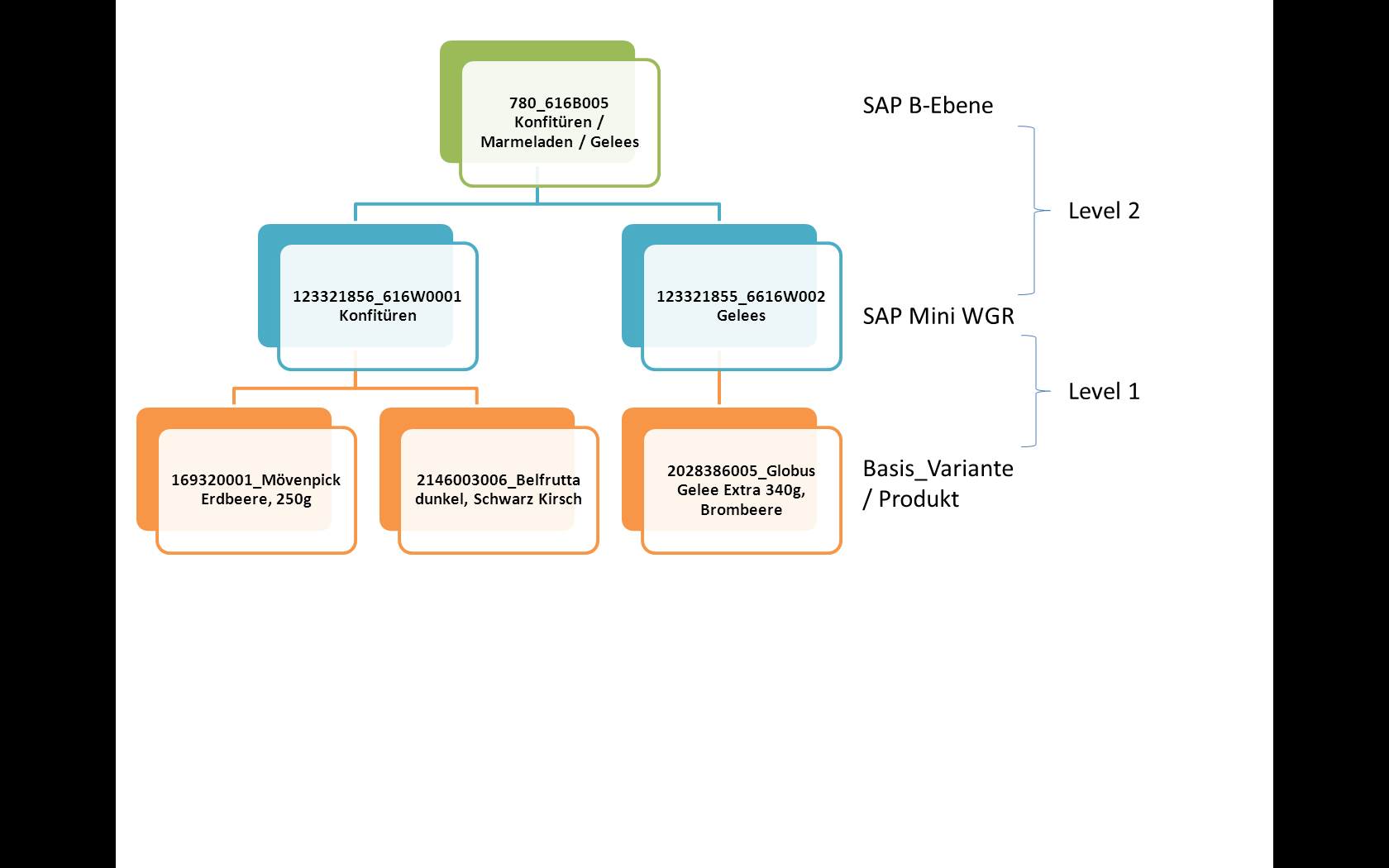


Abbildung 2: Ausschnitt der Produkthierarchie

Regeln können nicht nur Produkte, sondern auch Produktgruppen höherer Hierarchielevel enthalten. Die Hierarchietabelle ist notwendig, damit Warenkörbe mit konkreten Produkten um die entsprechenden Produktgruppen erweitert werden können. Andernfalls würde man keine Regeln finden, die Produktgruppen enthalten

* + KIND

Kind ID (Entweder Produkt- oder Produktgruppe)

* + KIND\_BEZ

Kind Bezeichner

* + KIND\_ID\_BEZ

Kind ID + Bezeichner

* + ELTERN

Eltern Id zum Kind

* + ELTERN\_BEZ

Eltern Bezeichner

* + ELTERN\_ID\_BEZ

Eltern Id + Bezeichner

* + EBENE

Ebene dieses Mappings. Z.B. ist ein konkretes Produkt als Kind und die nächsthörere Produktgruppe als Eltern als Ebene 1 angegeben. Eine Produktgruppe auf Ebene Mini-Warengruppe als Kind und die B-Warengruppe als Eltern ist mit Eben 2 angegeben

* + IST\_BLATT

Zur schnelleren Verarbeitung ist hier angegeben, ob es sich um ein Blatt handelt.

* WEBSHOP\_MAN\_STANDARD\_MAPPINGS

Mannuell erstellte mappings zwischen Produktgruppen und konkreten Produkten. Dies ist notwendig, da aus fachlicher Sicht immer konkrete Produkte vorgeschlagen werden sollen. Regeln enthalten oft aber auch Produktgruppen in der Konsequenz. Diese werden über die Mappingtabelle aufgelöst. Diese Tabelle wird vom JSM ebenfalls im Hauptspeicher gehalten und als Hash-Map für einen schnellen Zugriff verwaltet. Enthält die Spalten:

* + SAP\_WGR\_NR

Enthält die Warengruppe (egal welcher Hierarchieebene)

* + BASIS\_VARIANTE

Enthält die Produktnummer (Basisvariante) die als Mapping verwendet werden soll

* WEBSHOP\_STANDARD\_MAPPINGS\_BETRIEBE

Mappings auf Betriebsebene. Diese sind nicht manuell erstellt, sondern werden automatisiert auf Basis der gekauften Produkte pro Betrieb extrahiert. Momentan noch nicht verfügbar, aber im JSM bereits berücksichtigt.

* WEBSHOP\_STANDARD\_MAPPINGS\_KUNDEN

Siehe oben.

## Aufbau und Verwendung des Java-Scoring-Moduls (JSM)

Das JSM besteht momentan aus 6 Klassen, die im package **scoring** liegen. Desweiteren wird im package **test** die Klasse **RuleScorerTester**, die die exemplarische Verwendung des Rule Scorers beschreibt.

Die Hauptklasse ist **RuleScorer**. Diese wird vom Web Shop instanziiert und kann dann zum scoren von Produktwarenkörben verwendet werden. Die Parameter werden der Übersichtlichkeit halber als Parameterobjekt **RuleScorerParameters** übergeben. Dazu wird die Parameterklasse instanziiert und dann über die setter Methoden mit den gewünschten Parametern befüllt. Folgende Parameter stehen zur Verfügung.

* **private** Connection m\_con;

JDBC Connection über die auf die oben erwähnten Tabellen zugegriffen werden kann.

* **private** String m\_hierarchyCacheFile ;

Das Lesen der Produkthierarchie dauert einige Sekunden bis Minuten. Da dies nur einmal erfolgen must (bzw. beim Nachladen der Hierarchie) gibt es hier prinzipiell keine Probleme. Das Cache File wird hauptsächlich während der Entwicklung verwendet um schnellere Entwicklungszyklen zu ermöglichen. Kann NULL bleiben.

* **private** String m\_driveHierarchyTable;

Name der Hierarchietabelle. Standard: WEBSHOP\_REGELN\_HIERARCHIE

* **private** String m\_childColumn;

Name der zu verwendenden Kindspalte der Hierarchietabelle. Standard: **Kind**

* **private** String m\_parentColumn;

Name der zu verwendenden Elternspalte der Hierarchietabelle. Standard: **Eltern**

* **private** String m\_ruleTableManual;

Name der zu verwendenden Regeltabelle für manuelle Regeln

* **private** String m\_ruleTableCustomer;

Name der zu verwendenden Regeltabelle für Kundenregeln

* **private** String m\_ruleTableMarket;

Name der zu verwendenden Regeltabelle für marktbezogene Regeln

* **private** String m\_blackListTable;

Name der zu verwendenden Tabelle mit den Produktausschlüssen

* **private** String m\_manualMappingTable;

Name der zu verwendenden Mappingtabelle für manuelle Mappings

* **private** String m\_customerMappingTable;

Name der zu verwendenden Mappingtabelle für Kunden Mappings

* **private** String m\_marketMappingTable;

Name der zu verwendenden Mappingtabelle für marktbezogene Mappings

Der RuleScorer stellt die Methode score() zur Verfügung. Dieser wird eine ArrayList<String> mit den Produktnummern (Basisvarianten) übergeben, die momentan im Warenkorb liegen. Außerdem muss die Kundennummer und die ID des momentan verwendeten DRIVE Marktes übergeben werden.

Zurückgeliefert wird eine ArrayList<Rule> mit Regelobjekten. Diese enthalten in Sortierreihenfolge die Empfehlungen (Recommendations) auf Produktebene (Basisvariante) und können vom Web-Shop angezeigt werden. Siehe Beispielcode.

## Multithreading

Der RuleScorer muss von einem Thread instanziiert und initialisiert werden (siehe auch Beispielcode). Nach erfolgreicher Initialisierung (Methode init()) wird der Scorer intern zum Scoren freigegeben. Wird bereits ein Scoringversuch durchgeführt, bevor die Initialisierung durch einen Thread erfolgreich beendet wurde, wird eine Exception geworfen.

Der Scorer kann nur einmal initialisiert werden. Die Methode init() ist **synchronized** um parallele Zugriffe mehrerer Threads auszuschließen.

Ein RuleScorer Objekt kann innerhalb mehrerer Threads gleichzeitig zum scoren verwendet werden. Alle Membervariablen werden während eines Scoringaufrufs nur Lesend zugegriffen.

Ausnahme sind reload-Methoden (Blacklist/Ausschlussliste und Hierarchie). Diese sind wie die init() Methode **synchronized.** Die Reload-Methoden bauen die Blacklist bzw. die Hierarchie erst vollständig auf und ersetzen dann die vom Scorer verwendete Referenz. Daher wird die Überganszeit minimal gehalten.

## Weitere Dokumentation

Die detaillierte weitere Dokumentation ist direkt im Code verfügbar.