Informations-Systeme
Informationen zur Hochregalansteuerung

Helge Janicke, Niels-Peter de Witt, Karsten Wolke

16. Januar 2002

Inhaltsverzeichnis

1 Struktur des Programms

Das Programm wurde in Java programmiert. Es benutzt eine Client-Server Architektur, um die einzelnen Ein- und Auslagerungsstationen (Clients) von der eigentlichen Jobverarbeitung (Server) zu trennen.
Die einzelnen Ein- und Auslagerungsstationen generieren Anfragen (Tasks), die an den Server (Scheduler) gestellt werden. Dieser legt die einzelnen Tasks in eine Warteschlange ab, die nach Priorität sortiert werden kann. In unserem Beispiel ist es eine einfache FIFO-Warteschlange. Jeder Job wird also in der Reihenfolge abgearbeitet, wie er zum Server kommt.
Ein Dispatcher holt sich nun immer den ersten Job aus der Warteschlange und schickt diesen zu einer passenden Logikeinheit. Die Logikeinheit berechnet nach der ausgewählten Strategie die Aktion, die ausgeführt werden soll und steuert damit den/die Treiber an.

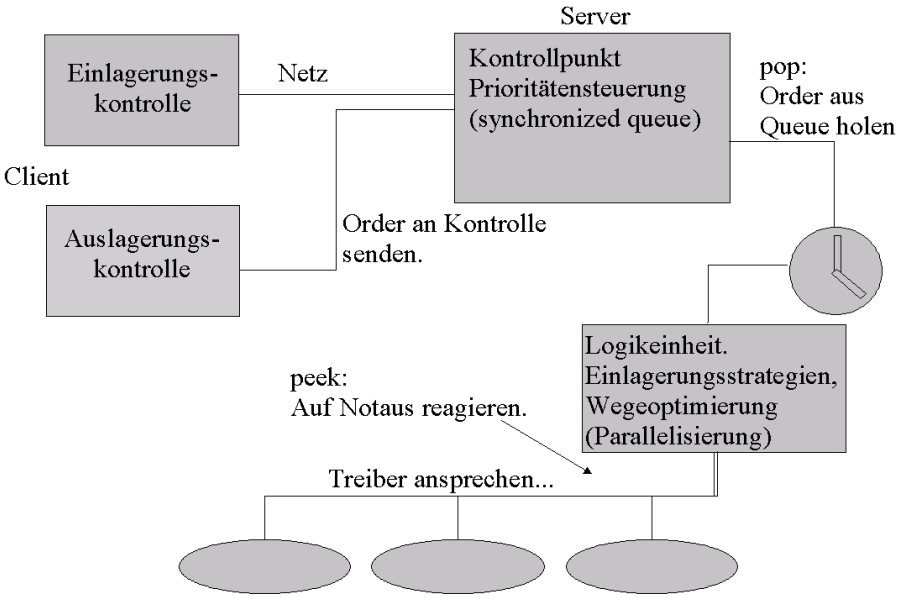

Abbildung 1: Struktur des Programms [Ablaufdiagramm2.png]

Im folgender Grafik sieht man das Klassendiagramm unseres Programms.

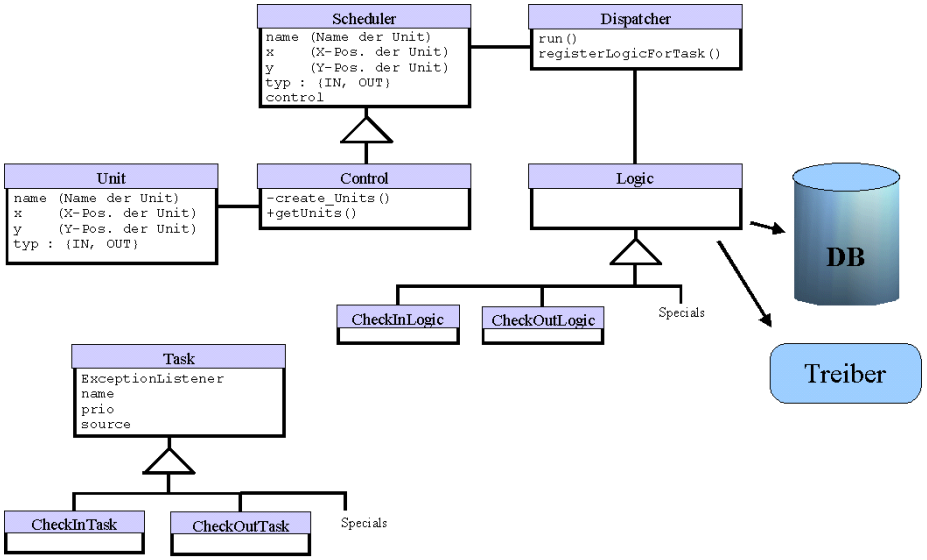

Abbildung 2: Klassendiagramm [Klassendiagramm2.png]

Die Klasse Unit repräsentiert unsere Ein- und Auslagerungsstationen (die einzelnen Clients). Von jede dieser Stationen aus können einzelne Tasks zum Ein- und Auslagern von Waren an den Server geschickt werden.
Die Klasse Control ist die Umsetzung eines Schedulers, der alle erzeugten Tasks in die Warteschlange einfügt. Desweiteren erzeugt die Klasse Control alle Units aus den Informationen der Datenbank.
Der Dispatcher ist ein eigener Thread, der in regelmäßigen Abständen eine Task aus der Warteschlange holt und die an eine für den Task registrierte Logik-Klasse zur Verarbeitung weitergibt. Diese Logik-Klassen greifen auf die Datenbank zu, um ihre Aufgabe zu erfüllen. Außerdem wird von ihnen der/die Treiber angesteuert.

2 Data-Dictionary

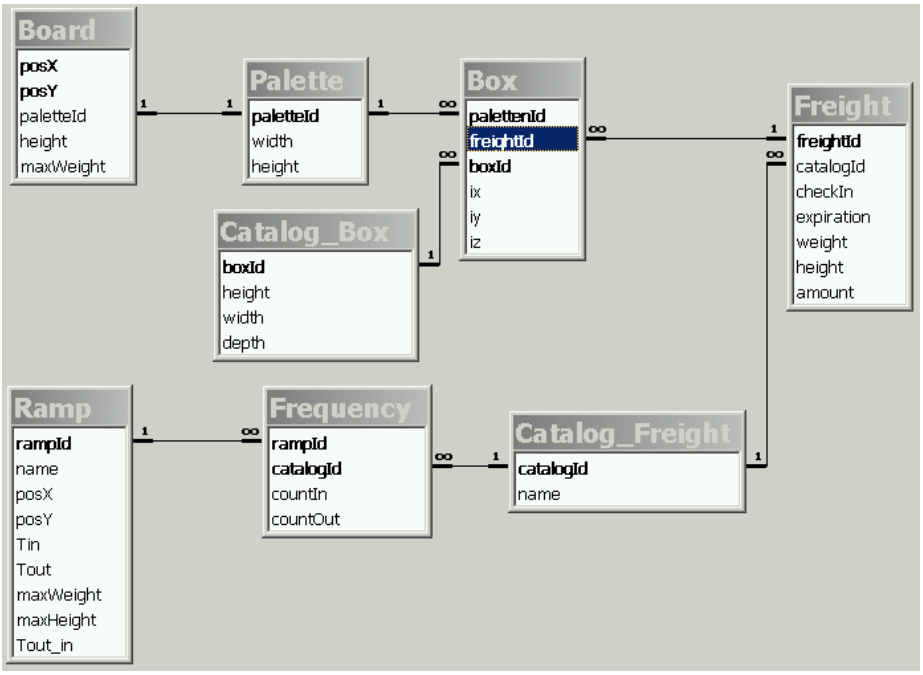

Abbildung 3: Relationship-Model [accessRM.png]

2.1 Das Data-Dictionary mit Beziehungen

```
Entity-Typ Board
        Attribute:              posX, posY, paletteId, height, maxWeight
        Primärschlüssel:        posX, posY

Entity-Typ Palette
        Attribute:              paletteId, width, height
        Primärschlüssel:        palleteId

Entity-Typ Box
        Attribute:              paletteId, freightId, boxId, ix, iy, iz
        Primärschlüssel:        palleteIdm freightId, boxId

Entity-Typ Catalog_Box
        Attribute:              boxId, height, width, depth
        Primärschlüssel:        boxId

Entity-Typ Freight
        Attribute:              freightId, catalogId, checkIn, expiration
                                weight, height, amount
        Primärschlüssel:        freightId
```

```
Entity-Typ Catalog_Freight
        Attribute:              catalogId, name
        Primärschlüssel:        catalogId

Entity-Typ Frequency
        Attribute:              rampId, catalogId, countIn, countOut
        Primärschlüssel:        rampId, catalogId

Entity-Typ Ramp
        Attribute:              rampId, name, poxX, posY, Tin, Tout,
                                maxWeight, maxHeight, Tout_in
        Primärschlüssel:        rampId



Beziehungstyp Board_Palette
        Beteiligte Entity-Typen: Board, Palette
        Komplexitätsgrad:        1:1

Beziehungstyp Box_Palette
        Beteiligte Entity-Typen: Box, Palette
        Komplexitätsgrad:       cn:1

Beziehungstyp Box_Catalog_Box
        Beteiligte Entity-Typen: Box, Catalog_Box
        Komplexitätsgrad:       cn:1

Beziehungstyp Box_Freight
        Beteiligte Entity-Typen: Box, Freight
        Komplexitätsgrad:        n:1

Beziehungstyp Catalog_Freight_Freight
        Beteiligte Entity-Typen: Catalog_Freight, Freight
        Komplexitätsgrad:        1:cn

Beziehungstyp Catalog_Freight_Frequency
        Beteiligte Entity-Typen: Catalog_Freight, Frequency
        Komplexitätsgrad:        1:cn

Beziehungstyp Frequency_Ramp
        Beteiligte Entity-Typen: Frequency, Ramp
        Komplexitätsgrad:       cn:1
```

2.2 Das Data-Dictionary mit Meta-Daten

Board

Attribute	Bedeutung	Datentyp	Nullable/ Leerstring	Indiziert
posX	Primärschlüssel >0 and <=10	Long Integer	No	No
posY	Primärschlüssel >0 and <=5	Long Integer	No	No
paletteId		Long Integer	Yes	Yes without duplicates
height	>0	Long Intger	No	No
maxWeight	>0	Long Intger	No	No

Palette

Attribute	Bedeutung	Datentyp	Nullable/ Leerstring	Indiziert
paletteId	Autowert, Primärschlüssel	Long Integer	No	Yes without duplicates
width	>0	Long Integer	No	No
height	>0	Long Integer	No	No

Box

Attribute	Bedeutung	Datentyp	Nullable/ Leerstring	Indiziert
paletteId	Primärschlüssel	Long Integer	No	Yes
freightId	Primärschlüssel	Long Integer	No	Yes
boxId	Primärschlüssel	Long Integer	No	Yes
ix		Long Integer	No	No
iy		Long Integer	No	No
iz		Long Integer	No	No

Catalog_Box

Attribute	Bedeutung	Datentyp	Nullable/ Leerstring	Indiziert
boxId	Autowert, Primärschlüssel	Long Integer	No	Yes without duplicates
height	>0	Long Integer	No	No
width	>0	Long Integer	No	No
depth	>0	Long Integer	No	No

Freight

Attribute	Bedeutung	Datentyp	Nullable/ Leerstring	Indiziert
freightId	Autowert, Primärschlüssel	Long Integer	No	Yes without duplicates
catalogId		Long Integer	No	Yes
checkIn		Date	No	No
expiration		Date	Yes	No
weight	>0	Date	No	No
height	>0	Date	No	No
amount	>0	Date	No	No

Catalog_Freight

Attribute	Bedeutung	Datentyp	Nullable/ Leerstring	Indiziert
catalogId	Autowert, Primärschlüssel	Long Integer	No	Yes without duplicates
name		String[50]	No, No	Yes without duplicates

Frequency

Attribute	Bedeutung	Datentyp	Nullable/ Leerstring	Indiziert
rampId	Primärschlüssel	Long Integer	No	Yes
catalogId	Primärschlüssel	Long Integer	No	Yes
countIn	>=0	Long Integer	No	No
countOut	>=0	Long Integer	No	No

Ramp

Attribute	Bedeutung	Datentyp	Nullable/ Leerstring	Indiziert
rampId	Autowert, Primärschlüssel	Long Integer	No	Yes without duplicates
name		String[50]	No, No	No
posX		Long Integer	No	No
posY		Long Integer	No	No
Tin		Boolean	No	No
Tout		Boolean	No	No
maxWeight		Long Integer	No	No
maxHeight		Long Integer	No	No
Tout_in		Long Integer	Yes	No

3 Strategien

Es wurden verschiedene Ein- und Auslagerungsstrategien sowie Turmstrategien implementiert. Diese werden in den folgenden Unterkapiteln beschrieben.
Da in unserem Programm beliebige Positionen für eine beliebige Anzahl von Rampen möglich sind, der Treiber allerdings nur 2 Rampen fest implementiert hat, werden die in der Datenbank eingetragenen Rampen auf diese Beiden umgelenkt. Dabei gilt:
Ist die X-Position der Rampe kleiner gleich 5, so wird die Rampe 1 an der Position (1/1) verwendet. Ansonsten wird Rampe 2 an der Position (10/1) genutzt.

Für die Berechnung in den Strategien wird immer die eingetragene Position der Rampe in der Datenbank verwendet.

3.1 Einlagerungsstrategien

Bei allen Einlagerungsstrategien wird, sobald kein Platz mehr im Regal frei ist, dies dem Benutzer mitgeteilt und die Einlagerung abgebrochen.

3.1.1 Random (Zufällig)

Zufällige Auswahl eines freien Platzes aus der Menge aller freien Positionen im Regal.

3.1.2 First-Fit

Auswahl einer freien Position im Regal, die den kleinsten Abstand zur Position der Einlagerungsstation hat.
Benutzter SQL-Ausdruck:

```
SELECT posX, posY FROM
  ( SELECT posX,posY,
       (
         (rampX-posX)*(rampX-posX) +
         (rampY-posY)*(rampY-posY)
       ) AS Abstand
    FROM Board WHERE paletteId IS NULL
  )
WHERE Abstand <= ALL
  ( SELECT
       (
         (rampX-posX)*(rampX-posX) +
         (rampY-posY)*(rampY-posY)
       ) AS Abstand
    FROM Board WHERE paletteId IS NULL
  );
```

Die Variablen *rampX* und *rampY* werden beim Aufruf mit der aktuellen X- und Y-Position der Einlagerungsstation besetzt.

3.1.3 Statistisch

Die Auswahl der Position erfolgt anhand von statistischen Informationen, an welcher Rampe die einzulagernde Fracht am wahrscheinlichsten wieder ausgelagert wird.
Um dies zu erreichen werden den einzelnen Rampen Gewichte zugeordnet, die sich aus den Statistiken ergeben.

$$posX = \sum_{i=1}^{n} \frac{CheckOut(i)}{CheckOutGesamt} * rampX(i)$$

$$posY = \sum_{i=1}^{n} \frac{CheckOut(i)}{CheckOutGesamt} * rampY(i)$$

n ist die Anzahl der Rampen.
CheckOut ist die Anzahl der Auslagerungen der Ware an der jeweiligen Rampe.
CheckOutGesamt ist die Anzahl aller Auslagerungen der Ware.

Das Ergebnis liefert eine günstige Position zurück. Existieren aber keine Statistiken zu der Ware, dann wird für *posX* und *posY* die Mitte des Regals angegeben. In der Nähe von *posX* und *posY* wird wiederum, unter Verwendung des SQL Ausdrucks, der auch im First-Fit benutzt wird, die freie Position, mit dem geringsten Abstand zu dem berechneten Position gesucht. Dazu werden in dem Ausdruck die Variablen *rampX* und *rampY* mit dem berechneten Werten *posX* und *posY* belegt.

3.2 Auslagerungsstrategien

Ist keine Palette im Regal vorhanden, die die angeforderte Menge beinhaltet, wird die Anfrage abgebrochen und dem Benutzer angegeben, wieviel er maximal pro Palette von der Fracht entnehmen kann.

3.2.1 nach Haltbarkeitsdatum

Die Auswahl der Ware erfolgt über das Haltbarkeitsdatum.
Die Ware, die am ehesten abläuft und zusätzlich noch über den Warentyp und über die angeforderte Menge auf der Palette verfügt wird aus dem Regal entfernt und zur angegebenen Rampe gebracht.

3.2.2 Random / Zufällig

Es wird eine Ware zufällig aus dem Hochregal entnommen, die sowohl dem Warentyp, als auch von der Menge her, den Angaben entspricht und dann zur angegebenen Rampe gebracht.

3.3 Turm-Strategien

Turm-Strategien sind Strategien, die angeben, wie sich der Beförderungsturm sich nach einer Ein-/Auslagerung verhalten soll.

3.3.1 Letzte-Position

Der Turm bleibt an seiner letzten Position stehen.

3.3.2 Strategische-Positionierung

Der Turm wird an eine strategisch günstige Position gefahren.
Für eine strategisch günstige Position verwenden wir Einlagerungsstatistiken aller Waren an den jeweiligen Rampen.

$$posX = \sum_{i=1}^{n} \frac{CheckIn(i)}{CheckInGesamt} * rampX(i)$$

$$posY = \sum_{i=1}^{n} \frac{CheckIn(i)}{CheckInGesamt} * rampY(i)$$

n ist die Anzahl der Rampen.
CheckIn ist die Anzahl aller Einlagerungen an der jeweiligen Rampe.
CheckInGesamt ist die Anzahl aller Einlagerungen.