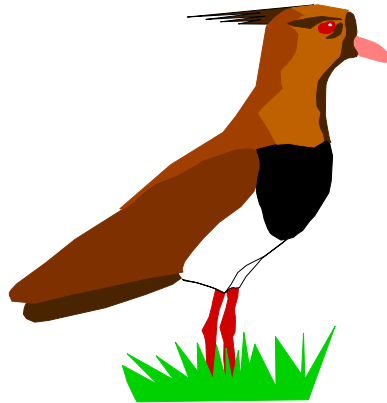


Mindelo WMS - Request for Comments

Daniel Peccini Correa
daniel.correa@puhrs.br
economics

Filipi Damasceno Vianna
filipi@em.puhrs.br
control and automation engineering

November, 24 2004



Contents

1. How does Mindelo work?	1
1.1. Data Acquisition	2
1.2. Simplification	2
1.3. Storing Plan	2
1.4. Intelligence	2
1.5. Picking	2
1.6. Routing	2
1.7. Tracking	3
2. Functionality	3
3. Table definitions	3

Abstract

Mindelo is an open and multi-platform warehouse management system(WMS). The main goals of Mindelo are an intelligent map, routing and tracking systems, an web based interface for management and for RF terminals and integration with big ERP¹ logistics modules such as SAPTM, MagnusTM, MicrosigaTM, BANTM². The aim of **MINDELO** is to be used by logis-

¹ERP: acronym for Enterprise Resource Planing and, in this text, will always be called ERP.

²SAPTM, MagnusTM, MicrosigaTM, and BANTM are registered trademarks of their owners.

tics centers eider by small transportation companies.

1. How does Mindelo work?

Everything starts with one production request. When some customer calls the company ordering some product, imediatly the ERP program (eider SAPTM, MagnusTM, MicrosigaTM, BANTM, etc, ...) get into action, and is there where the process begins.

obs.: Different ERP programs works with different DBMS³. The MagnusTM works with ProgressTM and, SAPTM and BANTM work with OracleTM and Microsoft SQL ServerTM⁴. SAPTM is certainly the ERP that is most widely used around the world.

The ERP, after receive the product's sell information, checks the DBMS for some information about the product. In the factory environment, it is necessary to check if there is product in the warehouse and if not, the system must generates an production order. In this case the production order is printed and and delivered to the production in charge.

All the steps described above happen in an industrial environment, in a logistic center the steps would

³DBMS: acronym for Data Base Management System. And in this text, will always be called DBMS.

⁴ProgressTM, OracleTM and Microsoft SQL ServerTM are registered trademarks of their owners.

be a little bit different, but for the **MINDELO** this should be transparent. Because the **MINDELO** hasn't played his role until now. Until now only the ERP has been working.

The production order has lots of information. These informations can be read by lots of ways: typing the bar code of being read by the proper hardware. To this functionality is used a data collector and from this point on, the **MINDELO** starts to play his role.

1.1. Data Acquisition

The **MINDELO** will be integrated with another Enterprise System (ERP) and with lots of kind of Enterprises (industries, transport/delivery companies, etc). The informations must come to the **MINDELO** in the most standardized way.

In the industry, we take the reading of the production order as the starting point for the transactions. But in transportation, delivery or logistic companies, a truck should arrive at the warehouse carrying a document called "manifest" which, for **MINDELO**, would play the same role of the production order. The "manifest" and/or the production order could also be tagged with bar codes. The bar code, read will identify the products in the ERP's data base. As we already told, the ERP itself generates this code, in logistic centers the data is electronically transmitted or even is typed.

The production order could be read by bar code, transponder or manually typed. But depending on each business we are running on, it is crucial to eliminate any kind of hand typing, using only automated data collecting systems.

Using barcode readers, the system could also ask for the employee ID, printed in his badge. This could be useful for auditing purposes.

1.2. Simplification

The software used in data collectors is normally complex. Independent of the language used, the software is usually complex and hard to implement changes or new features even to fix some kind of errors. Our idea is to replace those programs by `HTML` and scripting languages (`PHP`) using html forms. This can be done by connecting the data collector with one remote server using telnet (which in most of data collectors, comes already installed in the firmware) and then, in the remote server, the data collector's shell will be one text based browser, like links or lynx for example.

This simplification made possible by the **MINDELO** lets the own customer make his own modifications in a simpler way. Or even lets companies get **MINDELO** certified.

1.3. Storing Plan

After being produced, it is time to the product to be stored. The system should search for a proper place and must communicate with the DBMS (here comes progress, sql or java programming) storing all this information for further consulting.

All the informations about the warehouse will be centralized at **MINDELO**'s tables. These tables must be in the DBMS of the ERP because of the huge amount of data to be handled. And keeping the system over the same DBMS will make faster data transferance.

1.4. Intelligence

One of the main features of the **MINDELO** is the intelligence. The system must learn the behavior of the managed warehouse. That means that the most required product must be placed near by the exit, etc...

Of course that initially the system must be configured, or "trained".

1.5. Picking

The outgoing products, as the incoming will all be asked by the ERP. So this transaction will be triggered by a bar code reading.

After the read the requirement the system can inform where is each product to be retrieved from the warehouse and the faster retrieval sequence. In its first version the **MINDELO** will obey the FIFO⁵ order.

To build the retrieval sequence the system must previously know the warehouse layout, and so it knows where is each item to be retrieved. So the system builds a sequence where each next item is the most near item. This can also be useful to the warehouse managers to locate the operators during a possible emergency.

1.6. Routing

After a truck loading, must be generated a document that identifies everything the truck is carrying, this document is called "manifest". And by the build of the manifest, the **MINDELO** must generate the sequence each volume must be delivered, because the system previously knows the delivery address of each item.

For the first version, the delivery routing will be build based on the zip code sequence. After that the **MINDELO** can communicate with a geographical

⁵FIFO: acronym for First In First Out

information system (GIS) to build a smarter delivery routing.

1.7. Tracking

A customer should can locate a product in any warehouse, as from the data collector as from a desktop computer. At the desktop, the **MINDELO** should show a map showing where is the desired product.

The **MINDELO** should have a interface that makes possible for every user find any product. The user being or not inside the warehouse. That means that even after the truck loaded, the **MINDELO** should keep a table with the status of the loaded volumes.

FAQ.
Why PHP, not JSP?
PHP is open and java (required for JSP) not.
Yes, there is open java, but it not runs in all plataforms that Sun's Java runs.

What is the meaning of the name Mindelo?

What is that stupid bird on the RFCs?
That is a Lapwing, a small bird very common in the south of Brazil.

Why the Lapwing?
Well, in first place because it is the bird symbol of my home state in Brazil as the road runner is the bird symbol of New Mexico so, long before I think in develop Mindelo I was already decided to use the lapwing as symbol.

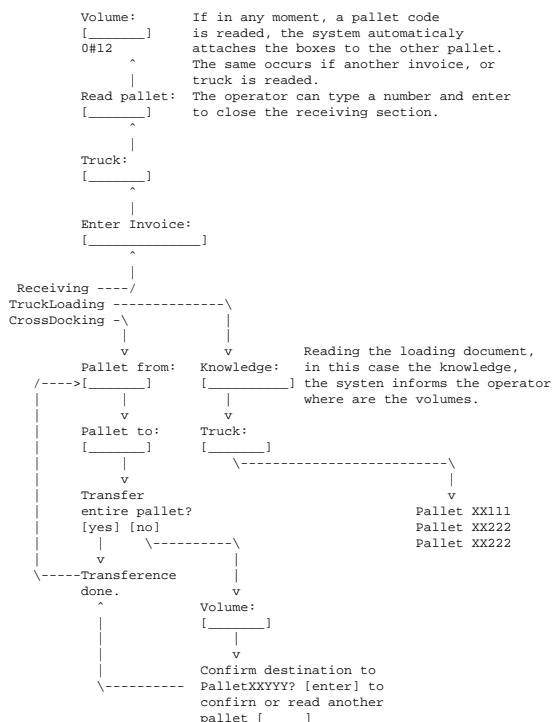
Terminal screens sketch:

There is no limit in how many terminals can, concurrently, load or unload a truck.

If the operator enter another truck, the systems automatically assumes this as the truck that is being unloaded.

If the operator enter another Invoice, the system assumes that the last Invoice is completely unloaded and start to check the consistences to close the Invoice. (????)

Remember to attach a special state for cicling.



2. Functionality

3. Table definitions

We can start defining some table of the **MINDELO**. These tables should be defined using standard SQL 92, to make possible to the data base creation script be runned in any DBMS server SQL compatible. By this moment we are not considering data base tuning.

```

--
-- This sql script is used to CREATE the first Mindelo DB, mean
-- while I'm not using any constranint. I'll put them soon.
--
--
-- First drops the old TABLEs
--
DROP TABLE pallets;
DROP TABLE loadingItems;
DROP TABLE warehouse;
DROP TABLE loaded;

--
-- So starts to CREATE the entire (tiny) database
--
--
-- Pallets catalog
--
CREATE TABLE pallets(
state char(2), -- State to where the volumes in the pallet go
code int,      -- An index, starts from zero to each state
status char(1),
PRIMARY KEY (state, code)
);
--
-- o Status can be
-- 0 - pallet empty
-- 1 - pallet been stored
-- 2 - pallet closed (already stored)
-- 3 - pallet been loaded into a truck
--
--
-- Main TABLE, shows the entire warehouse
--
CREATE TABLE warehouse(
code int PRIMARY KEY,
state varchar(2),
pallet_state char(2),
pallet_code int,
shipper int,
zip varchar(9),
item int,
total_items int,
status varchar(1),
invoice int,
delivery_date date,
shipped_date date,
factory varchar(20),
CONSTRAINT pallet FOREIGN KEY ( pallet_state, pallet_code )
REFERENCES pallets ( state, code )
ON DELETE cascade
ON UPDATE cascade
);
--
-- Temporary TABLE used to lock volumes while they
-- are been loaded into the truck.
--
CREATE TABLE loadingItems(
knowledge int PRIMARY KEY,
status varchar(1)
);
--
-- Temporary TABLE, shows what was loaded into
-- a truck.
--
CREATE TABLE loaded(
truck varchar(7) PRIMARY KEY,
code serial,
knowledge int
);
--
INSERT INTO pallets VALUES ('CL', 0, 0);
INSERT INTO pallets VALUES ('CL', 1, 0);
INSERT INTO pallets VALUES ('CL', 2, 0);
INSERT INTO pallets VALUES ('CL', 3, 0);
INSERT INTO pallets VALUES ('CL', 4, 0);
INSERT INTO pallets VALUES ('CL', 5, 0);
INSERT INTO pallets VALUES ('CL', 6, 0);
INSERT INTO pallets VALUES ('CL', 7, 0);
INSERT INTO pallets VALUES ('CL', 8, 0);
INSERT INTO pallets VALUES ('CL', 9, 0);
INSERT INTO pallets VALUES ('CL', 10, 0);
INSERT INTO pallets VALUES ('CL', 11, 0);
INSERT INTO pallets VALUES ('CL', 12, 0);

```

```
INSERT INTO pallets VALUES ('CL', 13, 0);
INSERT INTO pallets VALUES ('CL', 14, 0);
INSERT INTO pallets VALUES ('CL', 15, 0);
INSERT INTO pallets VALUES ('CL', 16, 0);
INSERT INTO pallets VALUES ('CL', 17, 0);
INSERT INTO pallets VALUES ('CL', 18, 0);
INSERT INTO pallets VALUES ('CL', 19, 0);
INSERT INTO pallets VALUES ('FL', 0, 0);
INSERT INTO pallets VALUES ('FL', 1, 0);
INSERT INTO pallets VALUES ('FL', 2, 0);
INSERT INTO pallets VALUES ('FL', 3, 0);
INSERT INTO pallets VALUES ('FL', 4, 0);
INSERT INTO pallets VALUES ('FL', 5, 0);
INSERT INTO pallets VALUES ('FL', 6, 0);
INSERT INTO pallets VALUES ('FL', 7, 0);
INSERT INTO pallets VALUES ('FL', 8, 0);
INSERT INTO pallets VALUES ('FL', 9, 0);
INSERT INTO pallets VALUES ('FL', 10, 0);
INSERT INTO pallets VALUES ('FL', 11, 0);
INSERT INTO pallets VALUES ('FL', 12, 0);
INSERT INTO pallets VALUES ('FL', 13, 0);
INSERT INTO pallets VALUES ('FL', 14, 0);
INSERT INTO pallets VALUES ('FL', 15, 0);
INSERT INTO pallets VALUES ('FL', 16, 0);
INSERT INTO pallets VALUES ('FL', 17, 0);
INSERT INTO pallets VALUES ('FL', 18, 0);
INSERT INTO pallets VALUES ('FL', 19, 0);
INSERT INTO pallets VALUES ('AZ', 0, 0);
INSERT INTO pallets VALUES ('AZ', 1, 0);
INSERT INTO pallets VALUES ('AZ', 2, 0);
INSERT INTO pallets VALUES ('AZ', 3, 0);
INSERT INTO pallets VALUES ('AZ', 4, 0);
INSERT INTO pallets VALUES ('AZ', 5, 0);
INSERT INTO pallets VALUES ('AZ', 6, 0);
INSERT INTO pallets VALUES ('AZ', 7, 0);
INSERT INTO pallets VALUES ('AZ', 8, 0);
INSERT INTO pallets VALUES ('AZ', 9, 0);
INSERT INTO pallets VALUES ('AZ', 10, 0);
INSERT INTO pallets VALUES ('AZ', 11, 0);
INSERT INTO pallets VALUES ('AZ', 12, 0);
INSERT INTO pallets VALUES ('AZ', 13, 0);
INSERT INTO pallets VALUES ('AZ', 14, 0);
INSERT INTO pallets VALUES ('AZ', 15, 0);
INSERT INTO pallets VALUES ('AZ', 16, 0);
INSERT INTO pallets VALUES ('AZ', 17, 0);
INSERT INTO pallets VALUES ('AZ', 18, 0);
INSERT INTO pallets VALUES ('AZ', 19, 0);

--
-- select state||to_char(code,'000') as pallet from pallets;
--
```