OpenERP WMS v8

Josse Colpaert – Developer

Quentin De Paoli – Developer

OpenERP WMS Should Manage This

– Advanced features for efficient routing and bins management –

As well as this…

– Keep it super simple & customizable –

Context

- Analyze of lacks/missing features
  - Collaboration with camptocamp INNOVATIVE SOLUTIONS BY OPEN SOURCE EXPERTS
  - Deep analyze of the existing compared to other WMS (SAP, Oracle...)
  - http://bit.ly/12gol33
  - POC made
- 2 conclusions:
  - Gonna be legendary!
  - Need a refactoring effort

New features v8

- Support All routes:
  - Pick → Pack → Ship
  - Cross-dock
  - Drop Shipping
  - Quality Control
- Smart Location Mgt
  - Removal Strategies
  - Put Away Strategies
- Operations
  - Packing
  - Barcode scanner
  - Batch and Waves Picking
- Costing
  - Multi-company
  - FIFO/LIFO/Avg/FEFO
  - Price Adjustment
  - Negative Stocks
- Lots management
  - Reservation
  - Boxes, Palets, ...
- User Interface
  - Barcode on all reports
  - Packing User Interface

OpenERP
OPEN SOURCE MANAGEMENT SOLUTION

Technical Objectives

- Scalable
  - Support millions of records
  - Computation not related to the # of moves
- Fully Customizable
  - Generic
  - Support most logistic flows without developments
- Easier to extend
  - New strategies
  - New routes
  - Specific lots
- Better Modularity
  - Inventory Management vs Warehouse Management

→ All of this while remaining super simple.

→ Less code than in v7.0 for all these features!

Introduction

Example of operations to manage:

- Pick → Pack → Ship
- Pack → Ship
- Receive → Control → Store
- Cross dock (from Reception to Packing, without going through the stock)
- Drop Shipping (from supplier to customer)
- Manual decisions must prevail

Definition of Concepts

Push rule:

- When products arrives in a location, push them to another:
  - e.g. When receiving 3 computers, move to QC zone.

Pull rule:

- When you need products in a location, bring them from another location (or produce, purchase, ...)
  - e.g. A delivery order (output → customer) should pull a packing operation (stock → output)

Routes:

- A set of push & pull rules to set on a product
  - e.g. Pick → Pack → Ship, Cross-docking, ...

→ These simple concepts handle all use cases!

Example: Pick → Pack → Ship

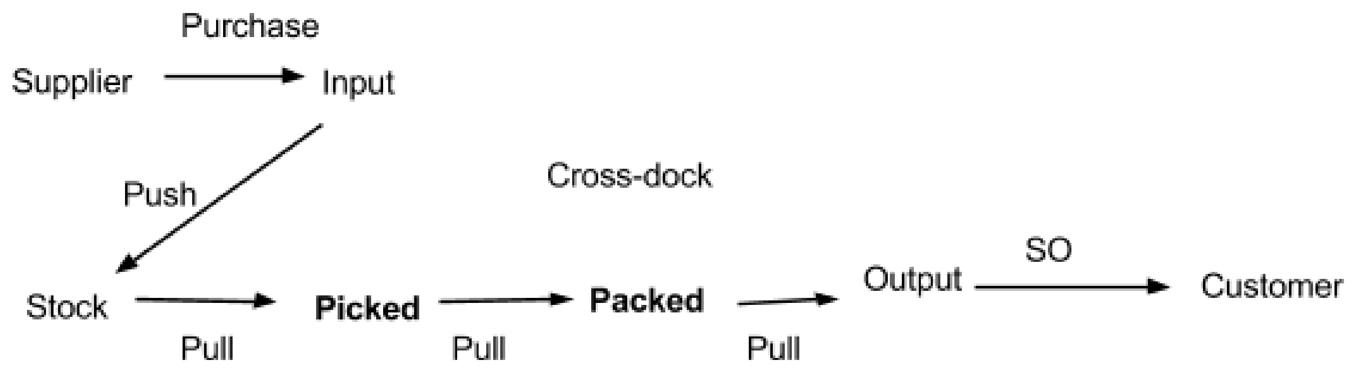
- Refers to the following configuration

$$\begin{tikzcd}\text{Stock} \arrow[r, "\text{pull}"] & \text{Picked} \arrow[r, "\text{pull}"] & \text{Packed} \arrow[r, "\text{pull}"] & \text{Customer}\end{tikzcd}$$

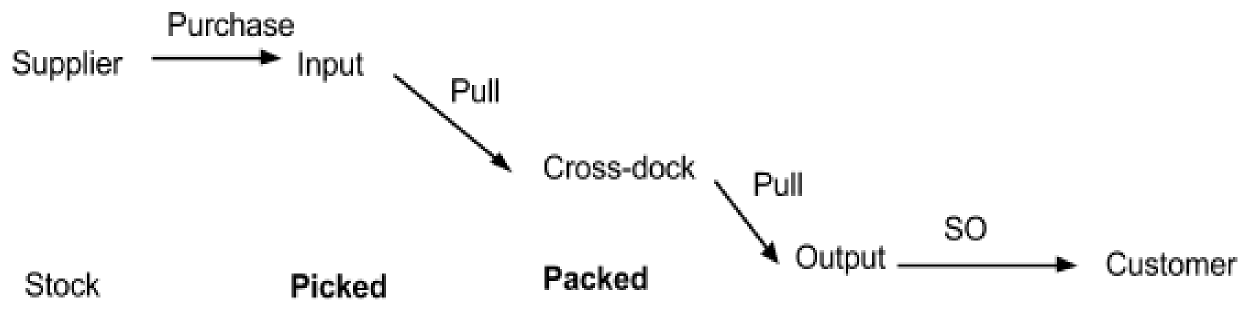
- Location Structure:
  - Warehouse
  - Stock
  - Picked: where you put goods before being packed (= pack in)
  - Packed: where you put the packed goods (= pack out)

Example: Cross-Docking

- Plastic cups

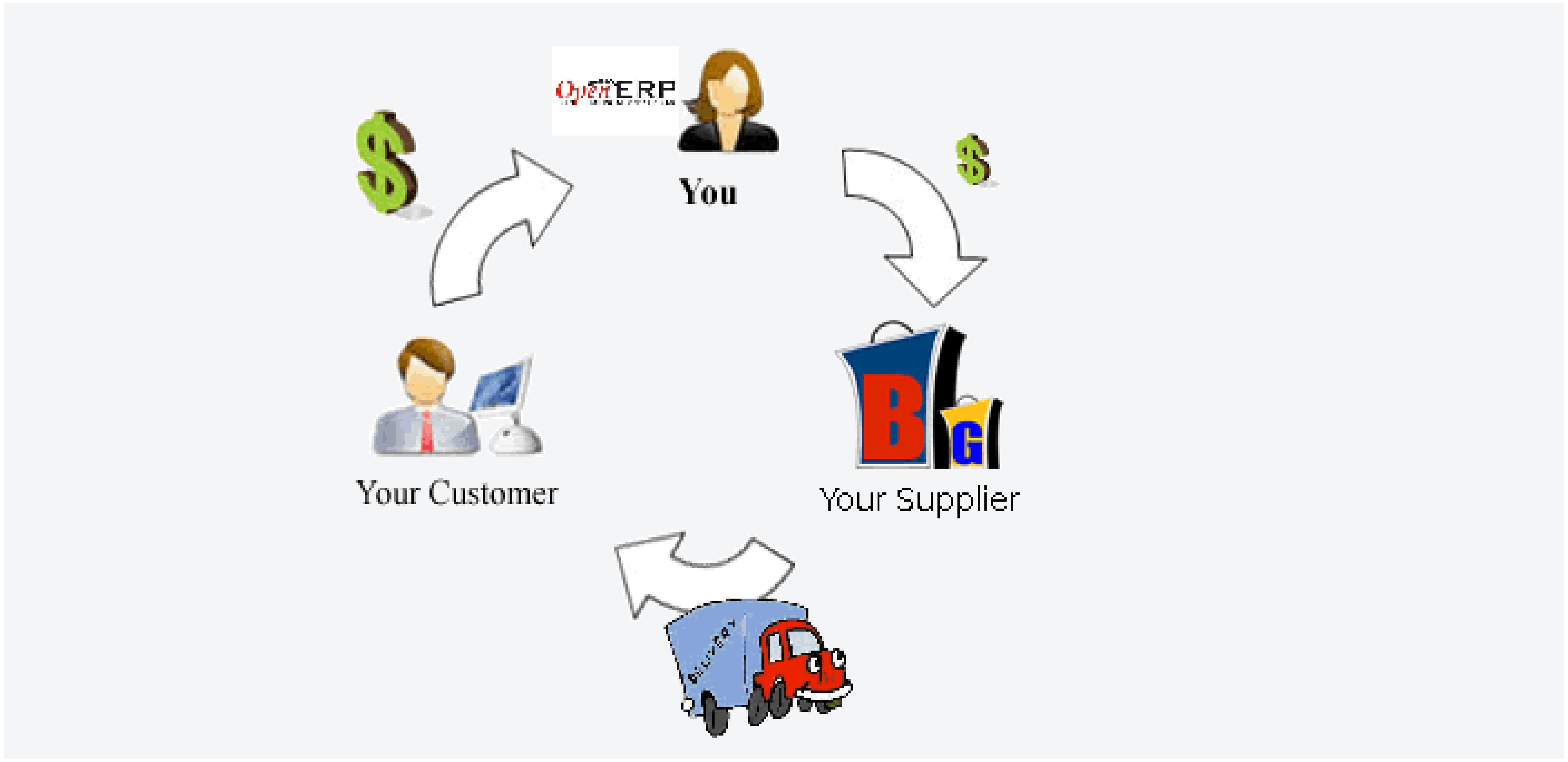


- Fresh products



Example: Drop-Shipping (1/2)

- Supplier direct delivery (in MTO from SO)



Example: Drop-Shipping (2/2)

- In v7.0, a sale order creates:
  - stock.picking
  - stock.move
  - procurement.order on output location

- In v8.0, a sale order creates:
  - procurement.order on customer location

→ stock.move are created automatically by a pull rule on the “Customers” location

→ less code, more flexibility (allows from supplier to customer, without taking products from the stock)

Technical Explanation

When action_confirm on stock.move:

- Trigger push rules (other moves)
- Trigger pull rules (procurements)

Modularity:

- Push rules defined in stock_location module
- Simple pull (stock → Customer): stock module
- Advanced pull (by product/location): stock_location module

Grouping stock.move into pickings is made with another principle. See later section.

Introduction

- Bins are sub-locations for which you don't care about virtual stock
- Operations to manage
  - Bins management
    - Choose a bin/lot when taking products
    - Choose a bin when dropping products
  - Lots management
    - Upstream/downstream traceability
    - Implement costing method depending on the real purchase price
  - Manual decisions must prevail

Quants

- Concept explained:
  - Smallest homogeneous part of stock
  - Holds information previously on stock.move (Purchase price, lot, stock ownership, ...)
  - Implements the lot management

- Quants ≠ Stock Moves ≠ Procurements

	Products	Consumable	Services
Procurement	Yes	Yes	Yes
Stock.move	Yes	Yes	
Quants	Yes		

Quants: use case

Stock move

- ID 1: 5 PCE, Supplier → Input
- ID 2: 3 PCE, Supplier → Input
- ID 3: 6 PCE, Input → Stock
- ID 4: 2 PCE, Input → Output

Quants

- 5 PCE Input (history_ids: ID1)
- 3 PCE Input (history_ids: ID2)
- 5 PCE Stock (history_ids: ID1, ID
- 1 PCE Stock (history_ids: ID2, ID
- 2 PCE Output (history_ids: ID2, ID4)

Quants

- Benefits
  - Optimization of real stock computation
    - Computation becomes trivial
    - Query doesn't grow with time (depends only on the physical products in stock)
    - Inventory analysis is super simple
  - Upstream/Downstream traceability (lot management)
  - Allows to implement FIFO costing (see next slide...)

Quants in FIFO 1 location: use case

Stock move

- ID 1: 20 PCE at 20€, Supplier → Stock
- ID 2: 20 PCE at 50€, Supplier → Stock
- ID 3: 30 PCE, Stock → Customer at (20 * 20 + 10 *50) / 30 = 30€
  - Product price: 30 €

Quants

- 20 PCE Stock at 20 € (history_ids: ID1)
  Customer at 20 € , ID3)
- 20 PCE Stock at 50 € (history_ids: ID2)
  1 (history_ids: ID2)
- 10 PCE Customer at 50 € (history_ids: ID2, ID3)

Bins Management

- Need to have strategies to find child location from parent location

- Removal Strategies:
  - How do you reserve/select products, defined by location or product categories
  - e.g. FIFO/LIFO,Nearest Available, ...

- Put Away Strategies:
  - When receiving products, where do I have to put them

Push/Pull vs Removal/Put Away

	Push	Pull	Removal	Put Away
When	Move Confirmation	Move Confirmation	Move Assignation	Move Assignation
Goal	Route Products	Route Products	Bin Allocation	Bin Allocation
Impact	Stock.move	Stock.move	Quants	Quants
Task	Create stock.move.	Create Procurement	Select Quants	Set Quants Destination
Apply On	Products & Consumables	Products & Consumables	Products Only	Products Only

Examples: Reservation (in)

The steps:

- When the purchase order is validated, a stock.move is created (in state: confirmed)
  - Confirmed: Supplier → Input

- This stock.move triggers a push rule (input → Stock) that creates the following move:
  - Waiting: Input → Stock

- At the product reception, we execute the first move and assign the second move, we get:
  - Done: Supplier → Input (will create quants)
  - Assigned: Input → Stock (*) with quants from first move

(*) Put Away strategy: Quants get a destination reservation to go to Bin A

Examples: Reservation (out)

The steps:

When the sales order is validated, a stock move is created: (pull rule)

Confirmed: Stock → Customer

When we assign it, we get:

Assigned: Stock → Customer (*)

(*) Quants from Removal Strategy (e.g. take FIFO from bin A)

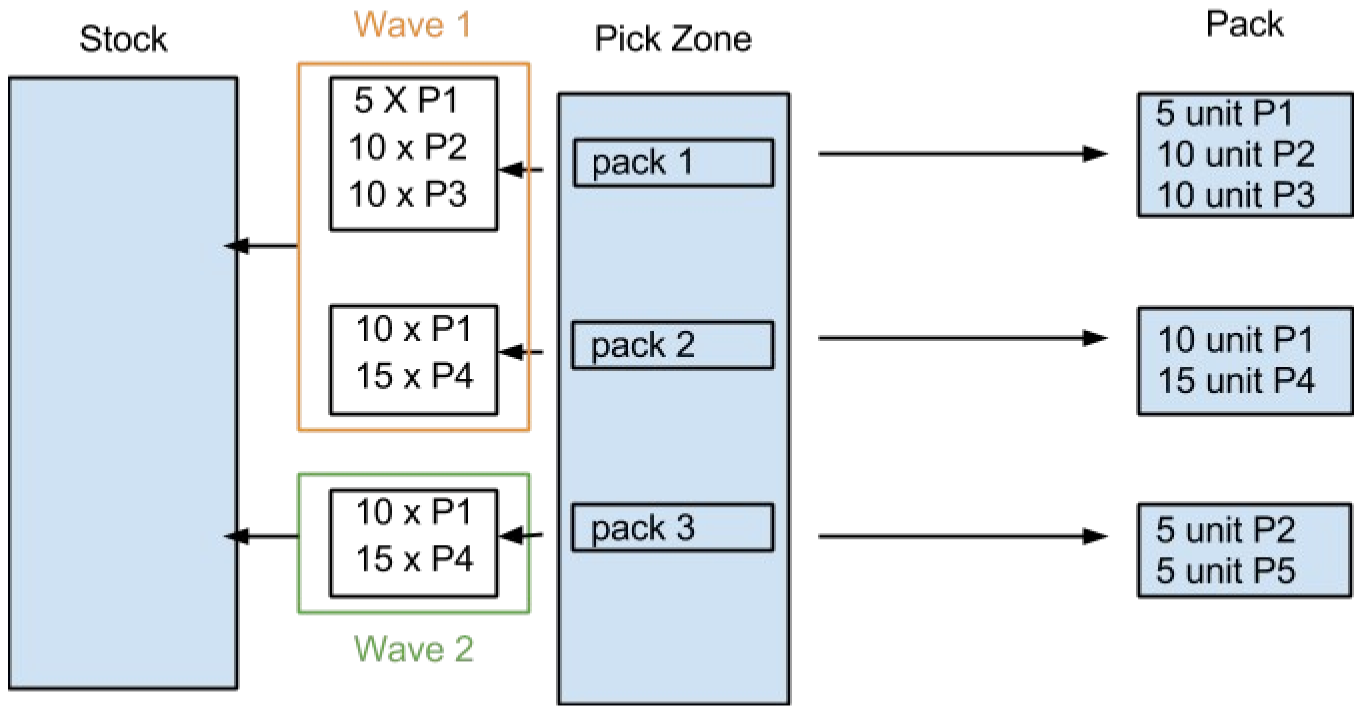
Introduction

- Operations to manage
  - Efficient Picking management
    - Batch/Wave picking
    - Trip assignation
  - Packing management
    - Hierarchical packing (Box → palet → container → ship)
    - Could be made at each step
    - (offline) Dedicated screen
  - Full support of barcode scanners

Pickings

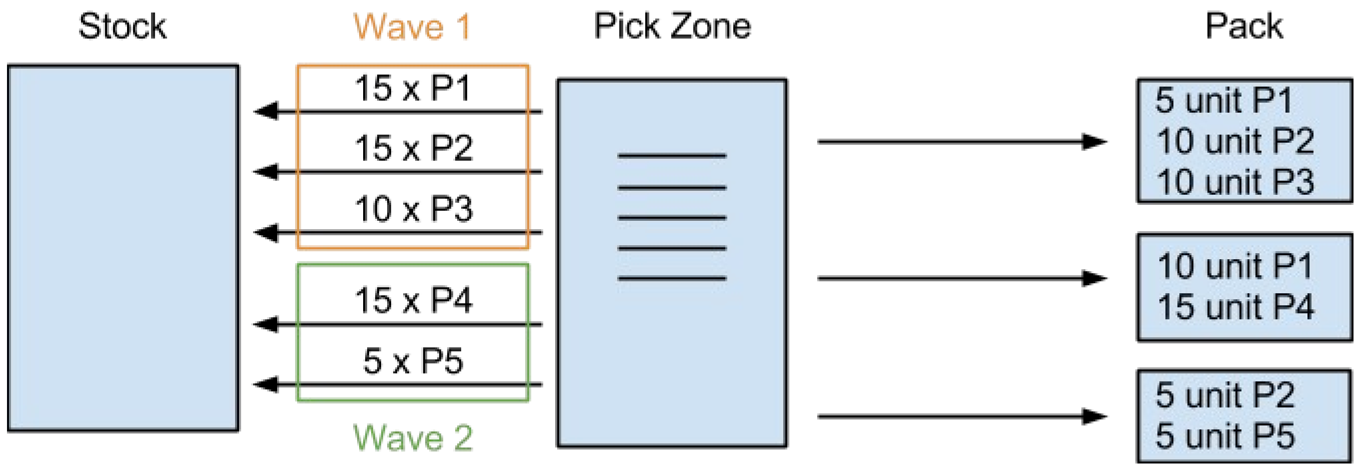
- Act differently:
  - Stock moves try to add themselves to an existing picking
  - If they can't, they create a new one
  - Procurement.group: new object implementing the grouping strategy of moves into a single picking
- Picking.wave:
  - Offers a way to work on several pickings at a time
  - Implemented in a new module
  - Wave assignation can be manual or based on complex algorithm (load constraints, product type...)
- Reports always show the barcode (product/box..)

Batch Picking



- Only pull rules
- Grouping strategy stock moves: we keep the reference of the initial SO (default)

Wave Picking



- Pull rule from pick zone → pack zone
- Minimum stock rule on pick zone

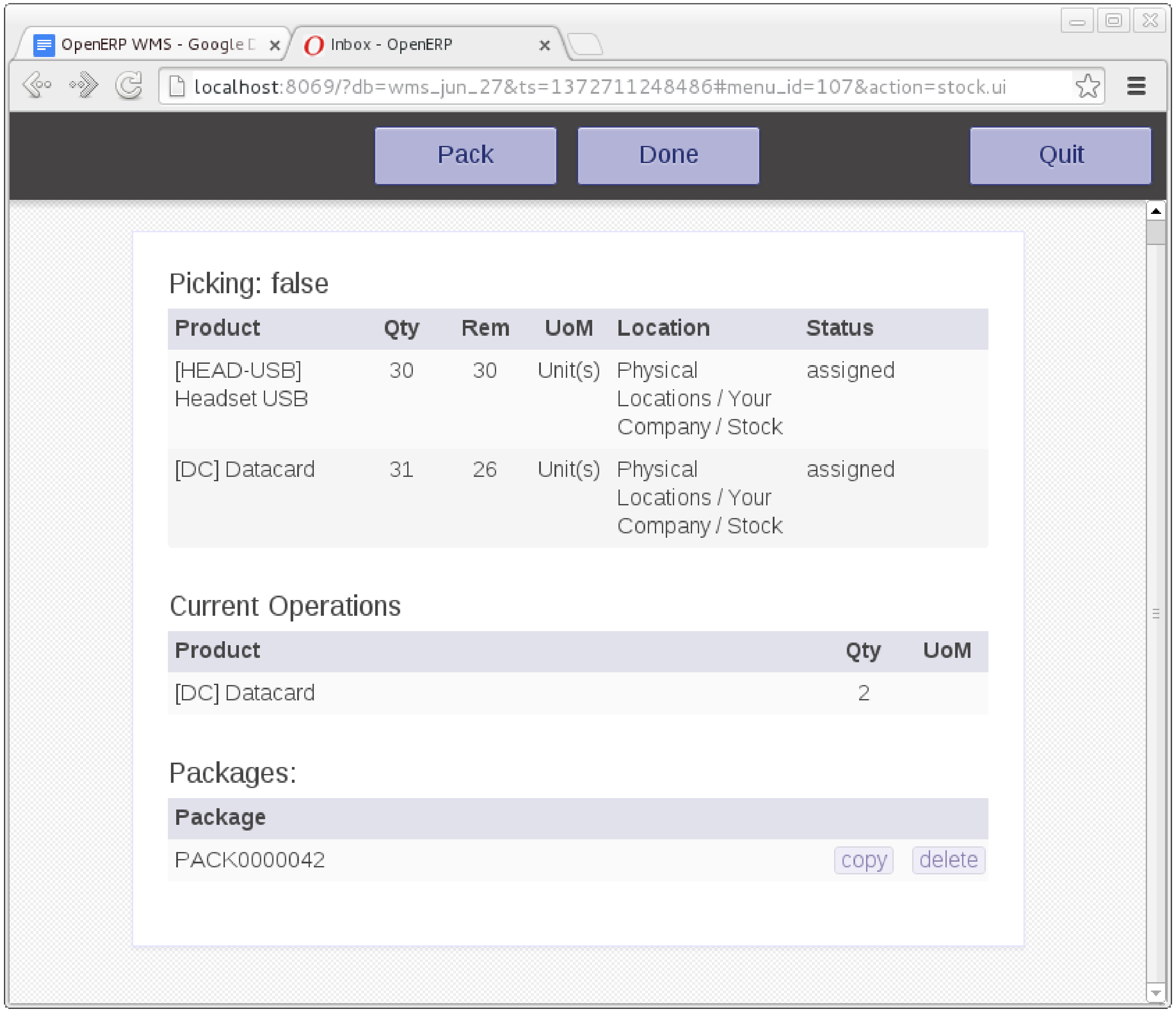
Trip assignation

- Nothing new:
    - the wave can be assigned thanks to the destination of the pickings
    - Correspond exactly to a batch picking between output and customer locations

Packing

- Want to know which quants are in which packs
  - => quant_package on quant instead of stock_tracking on move
  - Package can have parent to support pack in pack
- On the stock move, quant packaging operations are defined

Packing


OpenERP WMS - Google D
Inbox - OpenERP
localhost:8069/?db=wms_jun_27&ts=1372711248486#menu_id=107&action=stock.ui
Pack
Done
Quit

Picking: false

Product	Qty	Rem	UoM	Location	Status
[HEAD-USB] Headset USB	30	30	Unit(s)	Physical Locations / Your Company / Stock	assigned
[DC] Datacard	31	26	Unit(s)	Physical Locations / Your Company / Stock	assigned

Current Operations

Product	Qty	UoM
[DC] Datacard	2	

Packages:

Package		
PACK0000042	copy	delete

Costing (1)

- Multi-company
  - Cost price, valuation methods and cost method become company dependent (property fields)
- History
  - History table on the quants
  - History table for products

Costing (2)

- Product price gets updated when doing out
  - =price by default when no price from e.g. purchase
  - Finished products of production will also use this price
- Price adjustments:
  - Supplier invoice price != purchase price
  - Reconciliation with e.g. landed costs

Stock Replenishment

- Replenishment is currently handled by minimum stock rules (orderpoint)
- Need to be improved
  - Product categories
  - Quantity to Order: Fixed / Replenish Until Max
  - Cycle: Every day (current behaviour) / Week / Month
- Like the current features on recurring meeting: day/hour of week, day of month, X weeks, …
- Based on:
  - Minimum Qty
  - Other for opportunist supply

Stock ownership

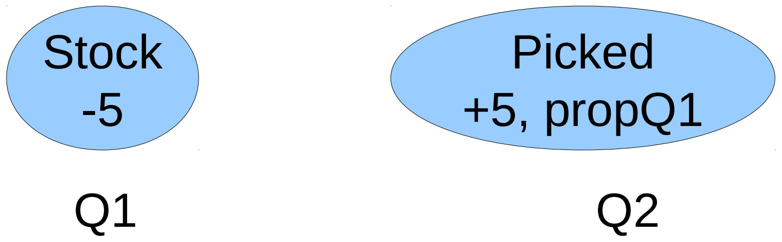
- Partner on the quants
- Needed for consigned stock:
  - stock is owned by supplier until it is sent
- Or 3PL, ...

Negative stock

- When we need to send when it has not been received yet
- Will be handled by negative quants
- FIFO/LIFO will value an out move when in move adjusts negative stocks
  - => will only generate accounting entries then

Negative stock

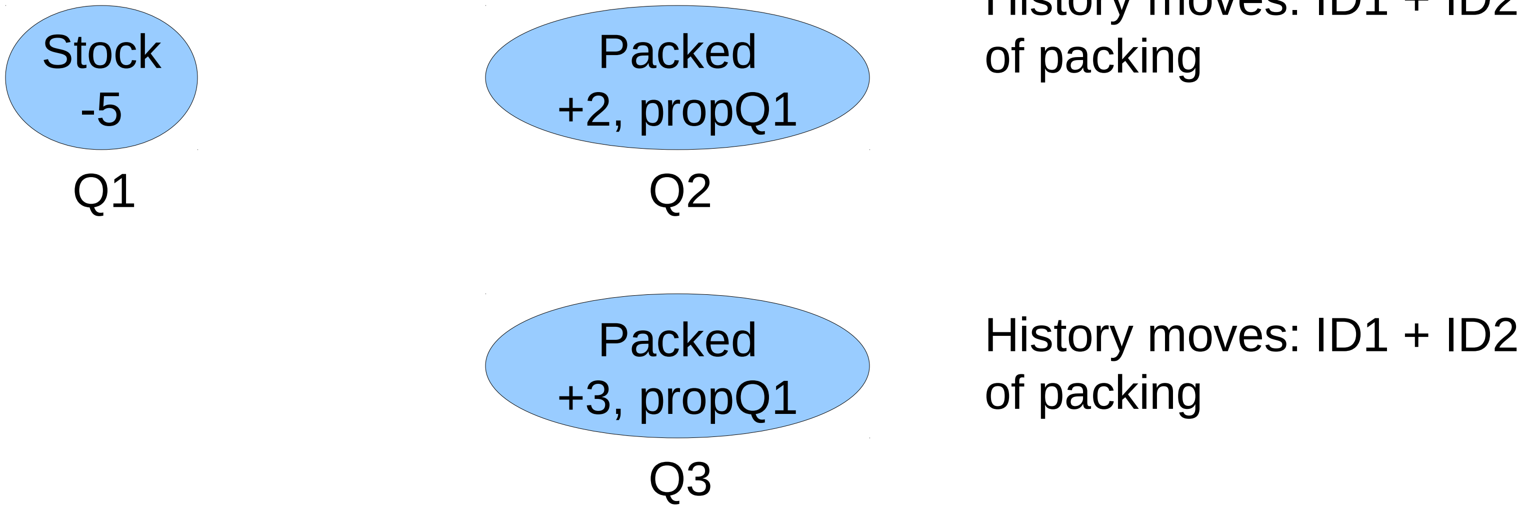
- Suppose have to send 5 pieces from stock without reception (=> force assign)



History moves: ID1 from stock to pick

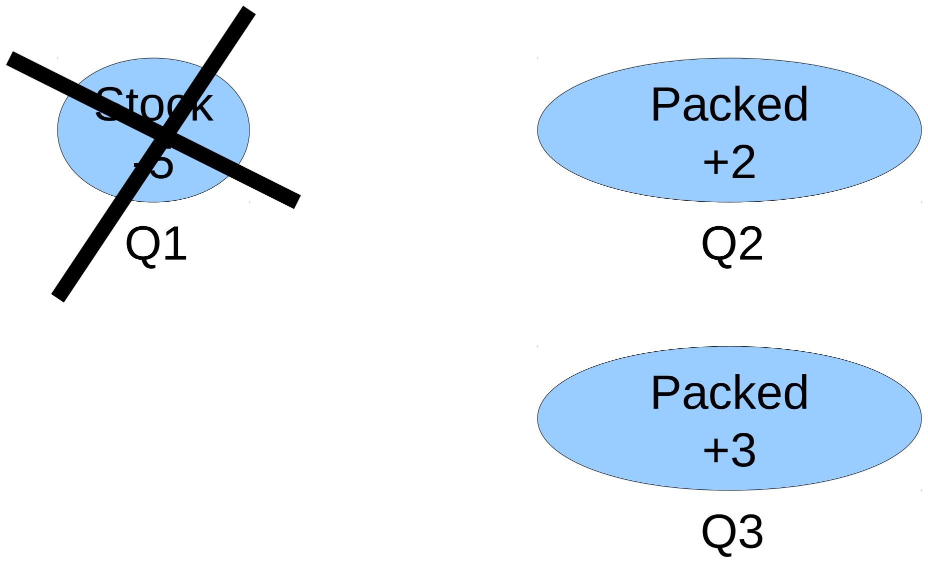
Negative stock

- Will split


Stock
-5
Q1
Packed
+2, propQ1
Q2
History moves: ID1 + ID2 of packing
Packed
+3, propQ1
Q3
History moves: ID1 + ID2 of packing

Negative stock

- Will reconcile when reception in stock or movement from stock



History moves: ID1 + ID2 + ID3 of reception

History moves: ID1 + ID2 + ID3 of reception