Alden Jettpace

Warehouse Database Narrative

In the logistics supply chain, warehouses must at the very least be able to the track the vast amounts of packages and materials that it takes in, stores, and distributes, and thus databases must be built to where all these materials can be queried and audited. The large amounts of material stored within warehouses and the many orders it is capable of fulfilling means it is better than the database for such a warehouse has automatic processes which allow it to update its values in real time. Such warehouses can vary in the other assets which they control and thus must monitor and maintain, but can generally be categorized under labor, operational capital, and transaction capital. That is: monitoring workers, monitoring necessary tools such as trucks and forklifts, and monitoring the material that flows in and out of the facility. For such a database to be sufficiently comprehensive, associated details for these categories and their interactions must also have maintained records for sufficiently detailed auditing of performance and finance.

In the warehouse in question, it is capable of tracking customers and their orders to be fulfilled, each within their own table. Since customers can make multiple orders, customer identification is tracked as an attribute within the associated order. The order is then to be fulfilled by a package, but can be divided into multiple to meet realistic space and weight needs, and not every order will be fulfilled at a time. Since the operations are recorded in real time, the order and its parts are recorded at the time the customer makes the purchase online, and then over time is fulfilled by potentially many packages. While the goal is to fulfill every order, the real time updating nature of the system means they must temporarily not have associated packages.

Packages and Orders each contain items of varying amounts, though the amount of items in packages cannot exceed the amount in associated orders. As such, an automatic trigger must be put in place to ensure that this overpacking is not allowed to happen. At the same time, packages items must be drawn from existing stocks of said item, and cannot draw from more than that. Said limitation does not apply to orders items, since they are to be fulfilled later rather than right at insert. It is assumed for this warehouse that the items it supplies are purchased from outside suppliers, each item being supplied by only one supplier at a time, but the warehouse can get that item from another supplier if it chooses, but can only ever have one associated supplier for an item at a time.

For tracking employee performance and labor costs, it is assumed that while all employees must login and all employees have an associated wage, shift, and title, the only relevant title for day to day operation and realistic performance auditing would be for packers and drivers, tracking how many orders they fulfill. While other positions theoretically exist, they are not relevant to the realtime operation needs of the warehouse database system. Along with specialization in labor, there is also a specialization in operational capital. With the most relevant tools of logistics being tools for movement, the generalized term for such tools would be Fleet. While identified by purchase price and purchase date in a generalized sense, it is then specialized into two specific fleet vehicles: Trucks and Forklifts. Trucks are especially important as they are recorded alongside the shipments which deliver packages to next leg of the logistics journey.

While from this assumption, generalized tracking of details, costs, and status of current operations can be tracked, changes in this president will cause the details of past operations to disappear and skew any attempts to audit performance and associated costs during time accrued. While restock of supplies from purchasers must be monitored, it must be monitored with price at time of purchase along with cost of purchase being recorded. While this creates an apparent anomaly in data with a supposed Item Price in one part of the database not matching the price of the associated item when audited later, it serves overall auditing purposes and is thus acceptable. Along with recording per item price of restock purchases from suppliers, the variation in cost of labor must also be recorded to have accurate measures of labor costs when auditing later. As such, newly hired employees will have their initial wage recorded in a wage register which gains new entries each time an employee's wage is update thanks to a raise, and time of change is recorded to ensure that if monitoring timeclock at a previous date, by matching it to wages at time of said timeclock, associated labor costs of the past can be accurately determined.

With these considerations in mind, the database for said warehouse consists of a total of 19 tables: CUSTOMER, ORDER, PACKAGE, SUPPLIER, ITEM, RESTOCK, ORDER_ITEM, PACK_ITEM, EMPLOYEE, DRIVER, PACKER, TIMECLOCK, WAGE_REGISTER, ABSENSE, SHIPMENT, FLEET, FORKLIFT, TRUCK, and INSURANCE. Their relational strata and triggers ensure that real time adjustment and recording can be made to give accurate information for employees packing and delivering orders, as well as financial teams determining operational costs and efficiency.

Table Name	Attribute Name	Contents	Data Type	Format	Range	Required	PK/FK	Reference
CUSTOMER	CSTM_ID	Customer ID Code	INT(5)	99999	0 – 999…	Y	PK	
	CSTM_UNAME	Customer User Name	VARCHAR(30)	Xxxxxxx		Y		
	CSTM_FNAME	Customer First Name	VARCHAR(30)	Xxxxxxx		Y		
	CSTM_LNAME	Customer Last Name	VARCHAR(30)	Xxxxxxx		Y		
	CSTM_JOIN	Customer Join Date	DATE	yyyy-mm-dd		Y		
	CSTM_PNTS	Customer Reward Points	INT(3)	999	0-999	Y		
	CSTM_PHONE	Customer Phone Number	CHAR(12)	999-999-9999		Y		
	CSTM_ADD	Customer Billing Address	VARCHAR(50)	Xxxxxx		Y		
	CSTM_CITY	Customer Billing City	VARCHAR(50)	Xxxxxx		Y		
	CSTM_ST	Customer Billing State	CHAR(2)	XX		Y		
	CSTM_ZIP	Customer Billing Zip	CHAR(5)	99999	00000-99999	Y		
ORDER	ORDER_ID	Order ID Code	INT(8)	99999999	1 – 999…	Y	PK	
	CSTM_ID	Order Customer ID Code	INT(5)	99999	0 – 999…	Y	FK	CUSTOMER
	ORDER_TIME	Time Ordered	DATETIME	Yyyy-mm-dd Hh:mm:ss		Y		

	ORDER_COMP	Order Completed	BOOL	0/1	0 - 1	Y		
	ORDER_COST	Order Cost	DECIMAL(6,2)	9999.99	1.00 –9999.99	Y		
	ORDER_ADD	Order Delivery Address	VARCHAR(50)	Xxxxxx		Y		
	ORDER_CITY	Order Delivery City	VARCHAR(50)	Xxxxxx		Y		
	ORDER_ST	Order Delivery State	CHAR(2)	XX		Y		
	ORDER_ZIP	Order Delivery Zip	CHAR(5)	99999	00000-99999	Y		
PACKAGE	PACK_ID	Pack ID Code	INT(11)	99999999999	1 – 999…	Y	PK	
	ORDER_ID	Pack Order ID Code	INT(8)	99999999	1 – 999…	Y	FK	ORDER
	EMP_ID	Packer Employee ID Code	INT(3)	999	0-999…	Y	FK	PACKER
	PACK_TIME	Time Packed	DATETIME	Yyyy-mm-dd Hh:mm:ss		Y		
	PACK_TYPE	Type of Packaging	CHAR(4)	Xxxx	“Bag”, “Box”, “Skid”	Y		
	PACK_WGT	Package Weight	DECIMAL(5,2)	999.99	0.01 - 600.00	Y		
	SHIP_ID	Package Shipment ID Code	INT(7)	9999999	0-999…	Y	FK	SHIPMENT
ITEM	ITEM_ID	Item ID Code	INT(5)	99999	1-999…	Y	PK	
	ITEM_NAME	Item Name	VARCHAR(20)	Xxxxxx		Y		

	SUPP_ID	Item Supplier ID Code	INT(4)	9999	0-999…	Y	FK	SUPPLIER
	ITEM_WGT	Item Average Weight	DECIMAL(5,2)	999.99		Y		
	ITEM_AMOUNT	Item Stock	INT(5)	99999	0-60000	Y		
	ITEM_COST	Item Supply Cost	DECIMAL(5,2)	999.99	0.01-700.00	Y		
	ITEM_PRICE	Item Sell Price	DECIMAL(5,2)	999.99	0.01-999.99	Y		
SUPPLIER	SUPP_ID	Supplier ID Code	INT(4)	9999	0-999..	Y	PK	
	SUPP_NAME	Supplier Name	VARCHAR(50)	Xxxxxx		Y		
	SUPP_ADD	Supplier Address	VARCHAR(50)	Xxxxxx		Y		
	SUPP_CITY	Supplier City	VARCHAR(50)	Xxxxxx		Y		
	SUPP_ST	Supplier State	CHAR(2)	XX		Y		
	SUPP_ZIP	Supplier Zip Code	CHAR(5)	99999	00000-99999	Y		
	SUPP_PHONE	Supplier Phone	CHAR(12)	999-999-9999		Y		
	SUPP_EMAIL	Supplier Email	VARCHAR(50)	Xxx@xxx		Y		
FLEET	FLEET_ID	Fleet ID Code	INT(2)	99	0-99…	Y	PK	
	FLEET_BUY_DATE	Fleet Purchase Date	DATE	Yyyy-mm-dd		Y		
	FLEET_COST	Fleet Purchase Cost	DECIMAL(8,2)	999999.99	0.01 – 999999.99	Y		
	FLEET_TYPE	Fleet Vehicle Type	CHAR(8)	Xxxxx	“Truck”, “Forklift”	Y		

TRUCK	FLEET_ID	Fleet ID Code (Truck)	INT(2)	99	0-99…	Y	PK/FK	FLEET
	TRUCK_MAKE	Truck Make	VARCHAR(20)	Xxxxxx		Y		
	TRUCK_MODEL	Truck Model	VARCHAR(20)	Xxxxxx		Y		
	TRUCK_MILE	Truck Milage	DECIMAL(3,1)	99.9	0.01 >=	Y		
	TRUCK_TANK	Trunk Tank (Gal)	INT(3)	999		Y		
	TRUCK_AXEL	Truck Axel Weight	INT(5)	99999		Y		
	TRUCK_PLATE	Truck Plate	CHAR(6)	Xxxxx		Y		
FORKLIFT	FLEET_ID	Fleet ID Code (Forklift)	INT(2)	99	1-99	Y	PK/FK	FLEET
	FORK_MAKE	Forklift Make	VARCHAR(20)	Xxxxxx		Y		
	FORK_TYPE	Forklift Type	SET	Xxxxx	“Sitting”, “Standing”	Y		
	FORK_MAST	Forklift Mast Type	INT(1)	9	1-3	Y		
	FORK_BATTERY	Forklift Battery Volt	INT(3)	999		Y		
	FORK_MAX_WGT	Forklift Max Weight	INT(5)	99999		Y		
	FORK_MAX_HGT	Forklift Max Height	DECIMAL(3,1)	99.9		Y		
EMPLOYEE	EMP_ID	Employee ID Code	INT(3)	999	0-999…	Y	PK	
	EMP_FNAME	Employee First Name	VARCHAR(20)	Xxxxxx		Y		

	EMP_LNAME	Employee Last Name	VARCHAR(20)	Xxxxxx		Y		
	EMP_PHONE	Employee Phone	CHAR(12)	999-999-9999		Y		
	EMP_HIRE	Employee Hire Date	DATE	Yyyy-mm-dd		Y		
	EMP_TITLE	Employee Title	VARCHAR(10)	Xxxxxx	Null, “Packer", “Driver”			
	EMP_WGE	Employee Hourly Wage	DECIMAL(4,2)	99.99	00.01>=	Y		
	EMP_SHIFT	Employee Shift	CHAR(7)	Xxxxxx	“Morning”, “Evening”, “Night”	Y		
DRIVER	EMP_ID	Employee ID Code	INT(3)	999	1-999…	Y	PK/FK	EMPLOYEE
	DRIVER_LCNS	Driver License	CHAR(12)	9999-99-9999		Y		
PACKER	EMP_ID	Employee ID Code	INT(3)	999	1-999…	Y	PK/FK	EMPLOYEE
	PCKR_FORK_CRT	Packer Forklift Certified	BOOLEAN (TINYINT(1))	0/1		Y		
TIMECLOCK	EMP_ID	Employee ID Code	INT(3)	999	1-999…	Y	PK/FK	EMPLOYEE
	CLOCK_IN	Time Clock-In	DATETIME	Yyyy-mm-dd Hh:mm:ss		Y	PK	
	CLOCK_OUT	Time Clock-Out	DATETIME	Yyyy-mm-dd Hh:mm:ss				
WAGE_ REGISTER	EMP_ID	Employee ID Code	INT(3)	999	1-999…	Y	PK/FK	EMPLOYEE
	WAGE_DATE	Date Wage Change	DATE	Yyyy-mm-dd		Y	PK	
	WAGE_NEW	New Wage	DECIMAL(4,2)	99.99	0.01 >=	Y		
INSURANCE	INSR_ID	Insurance ID Code	INT(2)	99	1-99…	Y	PK	

	FLEET_ID	Fleet ID Code	INT(2)	99	1-99…	Y	FK	TRUCK
	INSR_PROV	Insurance Provider	VARCHAR(30)	Xxxxxx		Y		
	INSR_PREM	Insurance Premium	DECIMAL(7,2)	99999.99	0.01 >=	Y		
	INSR_RENEW	Insurance Renewal Date	DATE	Yyyy-mm-dd		Y		
ABSENSE	EMP_ID	Employee ID Code	INT(3)	999	1-999…	Y	PK/FK	EMPLOYEE
	ABS_DATE	Absence Date	DATE	Yyyy-mm-dd		Y	PK	
	ABS_REASON	Absence Reason	TEXT	“XXXXXX”		Y		
SHIPMENT	SHIP_ID	Shipment ID	INT(7)	9999999	0-999…	Y	PK	
	FLEET_ID	Delivery Truck ID	INT(2)	99	0-99…	Y	FK	TRUCK
	EMP_ID	Employee ID Code	INT(3)	999	1-999…	Y	FK	DRIVER
	SHIP_LTIME	Shipment Leave Time	DATETIME	Yyyy-mm-dd Hh:mm:ss		Y		
	SHIP_ADD	Shipment Destination Address	VARCHAR(30)	Xxxxxx		Y		
	SHIP_CITY	Shipment Destination City	VARCHAR(30)	Xxxxx		Y		
	SHIP_ST	Shipment Destination State	CHAR(2)	XX		Y		
	SHIP_ZIP	Shipment Destination Zipcode	CHAR(5)	99999	00000-99999	Y		
	SHIP_DIST	Shipment Distance Miles	DECIMAL(4,1)	999.9	0.1 -999.9			

	SHIP_COST	Shipment Cost	DECIMAL(6,2)	9999.99	0.00 >=	Y		
	SHIP_WGT	Shipment Weight	DECIMAL(7,2)	99999.99	1-(5*FLEET_ID .TRUCK_AXEL	Y		
ORDER_ITEM	ORDER_ID	Order ID Code	INT(8)	99999999	1-999…	Y	PK/FK	ORDER
	ITEM_ID	Item ID Code	INT(5)	99999	1-999…	Y	PK/FK	ITEM
	ORIT_AMOUNT	Order Item Amount	INT(3)	999	1-999	Y		
	ORIT_PACKED	Order Item Amount Packed	INT(3)	999	0-999	Y		
PACK_ITEM	PACK_ID	Package ID Code	INT(11)	99999999999	1-999…	Y	PK/FK	PACKAGE
	ITEM_ID	Item ID Code	INT(5)	99999	1-99999	Y	PK/FK	ITEM
	PKIT_AMOUNT	Package Item Amount	INT(3)	999	1-ORIT_AMOUNT	Y		
RESTOCK	ITEM_ID	Item ID Code	INT(5)	99999	1-99999	Y	PK/FK	ITEM
	RSTK_DATE	Restock Date	DATE	Yyyy-mm-dd		Y	PK	
	RSTK_SUPPLIER	Restock Supplier	INT(5)	99999	1-999…	Y		
	RSTK_ITEM_COST	Restock Item Cost	DECIMAL(5,2)	999.99	0.01-999.99	Y		
	RSTK_AMOUNT	Restock Amount	INT(5)	99999	1-(99999 – ITEM.ITEM_AMOUNT)	Y		
	RSTK_TOTAL	Restock Total Cost	DECIMAL(12,2)	9999999999.99	0.01 >=	Y		

Entity Relationship Model (ERM):

Warehouse Database Entity Relationship Model			
ENTITY	RELATIONSHIP	CONNECTIVITY	**ENTITY**
EMPLOYEE	…phones in…	(0:M)	ABSENSE
EMPLOYEE	…must clockin at…	(1:M)	TIMECLOCK
EMPLOYEE	…records wages in…	(1:M)	WAGE_REGISTER
EMPLOYEE	…specializes as…	(0:1) (subtype)	PACKER
EMPLOYEE	…specializes as…	(0:1) (subtype)	DRIVER
FLEET	…has…	(0:1) (subtype)	TRUCK
FLEET	…has…	(0:1) (subtype)	FORKLIFT
TRUCK	…covered by…	(1:1)	INSURANCE
TRUCK	…delivers…	(1:M)	SHIPMENT
DRIVER	…drives…	(1:M)	SHIPMENT
SHIPMENT	…contains…	(1:M)	PACKAGE
PACKER	…packs…	(1:M)	PACKAGE
ORDER	…fulfilled by…	(1:M)	PACKAGE
CUSTOMER	…purchases…	(0:M)	ORDER
ORDER	…composed of…	(1:M)	ORDER_ITEM
PACKAGE	…contains…	(1:M)	PACK_ITEM
ITEM	…ordered by…	(0:M)	ORDER_ITEM
ITEM	…packed in…	(0:M)	PACK_ITEM
ITEM	…records restock in..	(1:M)	RESTOCK
SUPPLIER	…supplies…	(1:M)	ITEM

Business Rules:

EMPLOYEE/ABSENSE

1. EMPLOYEE can have no ABSENSE or can have many
2. Every ABSENSE instance associated with only a single EMPLOYEE

EMPLOYEE/TIMECLOCK

1. Every EMPLOYEE must clock in at least once to remain employeed
2. Every TIMECLOCK records work hours of only a single EMPLOYEE
3. Employee must clock in to be able to clockout. TIMECLOCK.CLOCK_OUT can be null, but must have a value in CLOCK_IN

EMPLOYEE/WAGE_REGISTER

1. Whenever a new employee is entered into EMPLOYEE table, their wage is recorded in WAGE_REGISTER at time of hiring. New register than made each time EMP_WAGE changes in EMPLOYEE table. So each employee has at least one matching instance in WAGE_REGISTER
2. WAGE_REGISTER instance associated with only one EMPLOYEE

EMPLOYEE/PACKER

1. PACKER is a disjoint partial covering subtype of the EMPLOYEE supertype. As such, not every instance in PACKER, nor is every instance of EMPLOYEE also an instance of DRIVER
2. Every PACKER has a matching EMPLOYEE instance

EMPLOYEE/DRIVER

1. DRIVER is a disjoint partial covering subtype of the EMPLOYEE supertype. As such, not every instance in EMPLOYEE is PACKER, nor is every instance of EMPLOYEE also an instance of PACKER
2. Every DRIVER has a matching EMPLOYEE instance
3. Exists a dummy employee EMP_ID = 0 with a matching dummy driver with EMP_ID = 0 for driving dummy shipment SHIP_ID for yet to be shipped packages

FLEET/TRUCK

1. TRUCK is a disjoint complete covering subtype of the FLEET supertype. As such, not every instance in FLEET is TRUCK, but is either a TRUCK or a FORKLIFT
2. Every TRUCK has a matching FLEET FLEET_ID instance

FLEET/FORKLIFT

1. FORKLIFT is a disjoint complete covering subtype of the FLEET supertype. As such, not every instance in FLEET is FORKLIFT, but is either a FORKLIFT or a TRUCK
2. Every FORKLIFT has a matching FLEET FLEET_ID instance

TRUCK/INSURANCE

1. Every TRUCK must have exactly one INSURANCE policy on it
2. Every INSURANCE policy covers exactly one TRUCK

TRUCK/SHIPMENT

1. TRUCK must deliver at least one SHIPMENT
2. SHIPMENT can only be delivered by one TRUCK
3. If a TRUCK is wrecked and deleted, SHIPMENT.FLEET_ID switchs to FLEET_ID= 0 dummy truck, which can take any weight
4. SHIPMENT can only be attached to a truck that can carry its weight, such that SHIP_WGT <= 5*TRUCK_AXEL

DRIVER/SHIPMENT

1. A DRIVER must driver at least one SHIPMENT, but can drive many
2. A SHIPMENT must have exactly one DRIVER
3. Dummy shipment SHIP_ID = 0 has associated driver EMP_ID = 0 for holding unshipped packages

SHIPMENT/PACKAGE

1. A SHIPMENT must contain at least one PACKAGE, and usually contains multiple
2. A PACKAGE goes in exactly one SHIPMENT
3. Packages that have not yet been packed are given a dummy shipment rather than a null: SHIP_ID = 0. Shipments and Packages are not intended to be easily deleted.

PACKER/PACKAGE

1. A PACKER can pack multiple PACKAGEs, and must pack at least one to be considered a packer
2. A PACKAGE can only be packed by a single PACKER

ORDER/PACKAGE

1. Every ORDER must eventually be fulfilled thus every ORDER must eventually have a PACKAGE. (Note: By way of use in real-time, Orders would exist in system before package is put into system)
2. Every PACKAGE must belong to exactly one ORDER

CUSTOMER/ORDER

1. Customers can make multiple orders, but not every registered customer has to make an order
2. Orders can only have one associated customer it belongs to

ORDER/ORDER_ITEM

1. Every order must has at least one item ordered, so at least one matching instance in ORDER_ITEM
2. Every instance in ORDER_ITEM matches exactly one in ITEM.

PACKAGE/PACK_ITEM

1. Every package has at least one item to pack, so every PACKAGE has at least one matching value for PACK_ID in PACK_ITEM
2. Every instance of PACK_ITEM has an associated package containing the items

ITEM/ORDER_ITEM

1. Every instance of ORDER_ITEM must contain one item
2. Not every item is ordered, thus not every item appears in ORDER_ITEM
3. Can order more of item than actually exist in stock, but cannot pack more than exist in stock

ITEM/PACK_ITEM

1. Every instance of PACK_ITEM must contain one item
2. Not every item is packed, thus not every item appears in PACK_ITEM
3. Inserting into PACK_ITEM takes stock from ITEM, decreasing ITEM_AMOUNT by PKIT_AMOUNT
4. Cannot pack of an item than exist in stock. Thus, if ITEM_AMOUNT < PKIT_AMOUNT before insert on PACK_ITEM, that insert will fail.

ITEM/RESTOCK

1. Every item must be restocked at least once in its time at the warehouse
2. Every restock is only of one item at a time
3. Restock automatically sends restock amount to item stock, adding RSTK_AMOUNT to ITEM_AMOUNT in table ITEM.
4. Every instance of RESTOCK records price and supplier of ITEM at ITEM_ID at time of restock request. Changes in ITEM or deleting of supplier will not reflect on RESTOCK record
5. Cannot request restock of items that currently do not have a supplier. IE: Cannot request restock of items with SUPP_ID = 0.

SUPPLIER/ITEM

1. Every supplier must supply at least 1 item
2. Every item can only have one supplier
3. Items without a supplier are given the dummy supplier, SUPP_ID = 0
4. Deleting instance in SUPPLIER causes all matching values of SUPP_ID in item to take the value for the dummy supplier, SUPP_ID = 0. SUPP_ID = 0 should not be deleted to ensure table stability.

PACK_ITEM/ORDER_ITEM

1. Every insert on PACK_ITEM can only fullfill a corresponding ORDER_ITEM instance in related package order. Thus, cannot try to insert into pack item items not in the appropriate ORDER_ITEM instance
2. Insert on PACK_ITEM adds to appropriate ORDER_ITEM attribute ORIT_PACKED by PKIT_AMOUNT. Cannot more in PKIT_AMOUNT than is stated for that item in ORIT_AMOUNT.

Intended Use

The database is designed around multiple users within the context of a warehouse, and much like a warehouse many of its processes are automatic, and if the database is used in the wrongfully intended way, unaccepted anomalies can pop up. Within a database, a packer will with just several keystrokes automatically update several data tables without even realizing it, thanks to the automatic processes behind the scenes, and the user-interface seamlessly fusing what is actually several data tables into one screen to browse to check and pack items. Thus, just as well, when purchasing new stock, the one usually wishes to automate restock orders such that they automatically update item stores, rather than auditing many orders after the fact to then update the item amounts one at a time.

Thus, several automatic processes have been implemented into the database, and the database must be utilized with these in mind. When wanting to update item values after an order, we do not update items than place the order, but instead we place the order in the restock table, and if the item is able to be restocked by having a valid/non-dummy supplier, then the amount ordered is automatically added to the item amount. Just the same, packing items from orders will automatically draw the item amount packed from the amount of that item currently in stock. Thus, updates done on items amount information would be interpreted as lost/damaged stock or a sudden gift of free stock that didn't require a restock order.

As well, to ensure as few null values as possible within the database, which could be interpreted as either missing data or a glitch/error, instead certain "lack" qualities for foreign keys have been implemented as (1,1) or (1,M) relationships where the "null" is instead replaced with a 0-index dummy. These dummies can be found for suppliers, employee, fleet, truck, driver, shipment, customer. Employee and Driver dummy exist to drive a dummy truck. Truck and the other dummy entity instances are meant to represent a yet-filled or now deleted value. That is, when a customer leaves the system, every order which has that customer in their foreign key instead gains the dummy customer "FORMER_CUSTOMER" index 0 for that key, to have as few nulls as possible. This is also true for deleted trucks, deleted/lost suppliers. Shipment is unique in that its 0 value represents not a deleted value, but a yet to be determined value. If a package has value 0 for SHIP_ID, that indicates that package has not yet been assigned to a shipment and is still in the warehouse waiting to be shipped out.

The dummy variables serve a purpose in ensuring as few nulls as possible, while ensuring database deletes as few instances in a chain as possible. Things which are considered non-renewable operation costs, such as trucks or suppliers, are delete-able in a straightforward way,

as the dummy allows a reset of any references to that instance to be converted to 0 rather than deleted or set to null. Operating costs such as items, orders, shipments, packages, and employees are not to be easily deleted. If an employee were to leave in the middle of the month, we would want to preserve their information at least until the end of the month/until the financial auditing period to have a fully accurate view of costs of operations during the time period. Deletions of these instances in these tables would be done only in large scale data-clearing by the warehouse after many years when this information is no longer relevant. Even then, they would still be expected to create a copy of that information or the derived financial records therein.

Alden Jettpace Entity Relationship Diagrams

CUSTOMER (**CSTM_ID**, CSTM_UNAME, CSTM_FNAME, CSTM_LNAME, CSTM_JOIN, CSTM_PNTS, CSTM_PHONE, CSTM_ADD, CSTM_CITY, CSTM_ST, CSTM_ZIP)

ORDER (**ORDER_ID,** CSTM_ID, ORDER_TIME, ORDER_COMP, ORDER_COST, ORDER_ADD, ORDER_CITY, ORDER_ST, ORDER_ZIP)

ORDER_ITEM (**ORDER_ID**, **ITEM_ID**, ORIT_AMOUNT, ORIT_PACKED)

SUPPLIER (**SUPP_ID**, SUPP_NAME, SUPP_ADD, SUPP_CITY, SUPP_ST, SUPP_ZIP, SUPP_PHONE, SUPP_EMAIL)

FLEET (**FLEET_ID**, FLEET_BUY_DATE, FLEET_COST, FLEET_TYPE)

FORKLIFT (**FLEET_ID**, FORK_MAKE, FORK_TYPE, FORK_MAST, FORK_BATTERY, FORK_MAX_WGT, FORK_MAX_HGT)

INSURANCE (**INSR_ID**, FLEET_ID, INSR_PROV, INSR_PREM, INSR_RENEW)

RESTOCK (**ITEM_ID**, **RSTK_DATE**, RSTK_SUPPLIER, RSTK_ITEM_COST, RSTK_AMOUNT, RSTK_TOTAL)

TRUCK (**FLEET_ID**, TRUCK_MAKE, TRUCK_MODEL, TRUCK_MILE, TRUCK_TANK, TRUCK_AXEL, TRUCK_PLACE)

ITEM (**ITEM_ID,** ITEM_NAME, SUPP_ID, ITEM_WGT, ITEM_AMOUNT, ITEM_COST, ITEM_PRICE)

SHIPMENT (**SHIP_ID**, FLEET_ID, EMP_ID, SHIP_LTIME, SHIP_ADD, SHIP_CITY, SHIP_ST, SHIP_ZIP, SHIP_DIST, SHIP_COST, SHIP_WGT)

PACKAGE (**PACK_ID**, PACK_ID, EMP_ID, PACK_TIME, PACK_TYPE, PACK_WGT, SHIP_ID

EMPLOYEE (**EMP_ID**, EMP_FNAME, EMP_LNAME, EMP_PHONE, EMP_HIRE, EMP_TITLE, EMP_WGE, EMP_SHIFT)

PACK_ITEM (**PACK_ID**, **ITEM_ID**, PKIT_AMOUNT)

ABSENSE (**EMP_ID**, **ABS_DATE**, ABS_REASON)

WAGE_REGISTER (**EMP_ID**, **WAGE_DATE**, WAGE_NEW)

TIMECLOCK (**EMP_ID**, **CLOCK_IN**, CLOCK_OUT)

PACKER (**EMP_ID**, PCKR_FORK_CRT)

DRIVER (**EMP_ID**, DRIVER_LCNS)

Pre Normalization

The table below is to record the associated information of every item and order combination, including supplier of item at time of ordering. Dependent upon ORDER_ID and ITEM_ID as composite primary keys, plenty is missing when dependent on ORDER_ID. So to begin normalizing, we must fill out null values to rid us of repeating groups.

ORDER_ID	ITEM_ID	ORIT_AMOU	ORIT_PACKE	ITEM_NAME	ITEM_WGT	ITEM_AMOU	ITEM_COST	ITEM_PRICE	ORDER_COI	ORDER_COS	ORDER_ADD	ORDER_CIT	ORDER_ST	SUPP_ID	SUPP_NAME	SUPP_ADD	SUPP_CITY	SUPP_ST
1	1	5	5	HeaterCoil	3.50	102	7.00	10.00	1	300.97	Another Street	Somwhere	IN	1	Mikel Company	444 Parkway St.	Detroit	MI
	2	5	5	Bufferter	25.00	8	25.00	30.00	1					1	Mikel Company	444 Parkway St	Detroit	MI
	3	2	2	CTRL_ELEC	1.20	13	19.00	25.00	1					2	TechFutures	333 Narkway St	Los Angeles	CA
	7	50	50	SCREW-0.5	0.05	3450	0.01	0.05	1					4	ScrewyNails	111 Clarkway St	Zionsville	IN
	9	50	50	NAIL-1.1	0.02	2950	0.03	0.05	1					4	ScrewyNails	111 Clarkway St	Zionsville	IN
2	1	2	2	HeaterCoil	3.50	102	7.00	10.00	1	21.50	Some Street	Somewhere	IN	1	Mikel Company	444 Parkway St	Detroit	MI
	8	30	30	SCREW-0.1	0.02	4470	0.05	0.10	1					4	ScrewyNails	111 Clarkway St	Zionsville	IN
3	12	1	1	Washer Frame	250.50	7	300.00	599.99	1	99.99	Pennsyl vania Avn	DC	DC	1	Mikel Company	444 Parkway St	Detroit	MI
4	3	4	2	CTRL_ELEC	1.20	13	19.00	25.00	0	101.10	466 Oak Brv	Chicago	IL	2	TechFutures	333 Narkway St	Los Angeles	CA

1NF: Removal of repeating groups

The composite primary keys of ORDER_ID and ITEM_ID are recognized to create unqiue instances, and every associated value is filled out, leaving behind nulls and repeating groups. However, as shown on the next page there is plenty of partial and transative dependency to be found here..

As can be seen, ORDER_ID key determines the value of attributes ORDER_COMP, ORDER_COST, ORDER_ADD, ORDER_CITY, ORDER_ST, while the ITEM_ID key determines ITEM_NAME, ITEM_WGT, ITEM_COST, ITEM_PRICE, ITEM_AMOUNT, SUPP_ID, SUPP_NAME, SUPP_ADD, SUPP_CITY, and SUPP_ST. Thus we see two examples of partial dependency. However, SUPP_ID determines SUPP_NAME, SUPP_ADD, SUPP_CITY, and SUPP ST, showing transitive dependency within the partial dependency.

ORDER_ID	ITEM_ID	ORIT_AMOU	ORIT_PACKE	ITEM_NAME	ITEM_WGT	ITEM_AMOU	ITEM_COST	ITEM_PRICE	ORDER_COI	ORDER_COS	ORDER_ADD	ORDER_CIT	ORDER_ST	SUPP_ID	SUPP_NAME	SUPP_ADD	SUPP_CITY	SUPP_ST
1	1	5	5	HeaterCoil	3.50	102	7.00	10.00	1	300.97	Another Street	Somwhere	IN	1	Mikel Company	444 Parkway St.	Detroit	MI
1	2	5	5	Bufferter	25.00	8	25.00	30.00	1	300.97	Another Street	Somewhere	IN	1	Mikel Company	444 Parkway St	Detroit	MI
1	3	2	2	CTRL_ELEC	1.20	13	19.00	25.00	1	300.97	Another Street	Somewhere	IN	2	TechFutures	333 Narkway St	Los Angeles	CA
1	7	50	50	SCREW-0.5	0.05	3450	0.01	0.05	1	300.97	Another Street	Somewhere	IN	4	ScrewyNails	111 Clarkway St	Zionsville	IN
1	9	50	50	NAIL-1.1	0.02	2950	0.03	0.05	1	300.97	Another Street	Somwhere	IN	4	ScrewyNails	111 Clarkway St	Zionsville	IN
2	1	2	2	HeaterCoil	3.50	102	7.00	10.00	1	21.50	Some Street	Somewhere	IN	1	Mikel Company	444 Parkway St	Detroit	MI
2	8	30	30	SCREW-0.1	0.02	4470	0.05	0.10	1	21.50	Some Street	Somewhere	IN	4	ScrewyNails	111 Clarkway St	Zionsville	IN
3	12	1	1	Washer Frame	250.50	7	300.00	599.99	1	99.99	Pennsyl vania Avn	DC	DC	1	Mikel Company	444 Parkway St	Detroit	MI
4	3	4	2	CTRL_ELEC	1.20	13	19.00	25.00	0	101.10	466 Oak Brv	Chicago	IL	2	TechFutures	333 Narkway St	Los Angeles	CA

1NF DEPENDENCY DIAGRAM

1NF(**<u>ORDER_ID, ITEM_ID,</u>** ORIT_AMOUNT, ORIT_PACKED, ITEM_NAME, ITEM_WGT, ITEM_AMOUNT, ITEM_COST, ITEM_PRICE, ORDER_COST, ORDER_COMP, ORDER_ADD, ORDER_CITY, ORDER_ST, SUPP_ID, SUPP_NAME, SUPP_ADD, SUPP_CITY, SUPP_ST)

<u>PARTIAL DEPENDENCIES</u>
(ORDER_ID --> ORDER_COST, ORDER_COMP, ORDER_ADD, ORDER_CITY, ORDER_ST)

(ITEM_ID --> ITEM_NAME, ITEM_WGT, ITEM_AMOUNT, ITEM_COST, ITEM_PRICE, SUPP_ID, SUPP_NAME, SUPP_ADD, SUPP_CITY, SUPP_ST)

<u>TRANSITIVE DEPEDENCIES</u>
(SUPP_ID --> SUPP_NAME, SUPP_ADD, SUPP_CITY, SUPP_ST)

TO FIX/SET TO 2NF

To set to 2NF, the partial dependencies must be addressed by seperating them into two new tables, with the determinate attributes being singular primary keys for said tables. These tables will be tables ORDER and ITEM

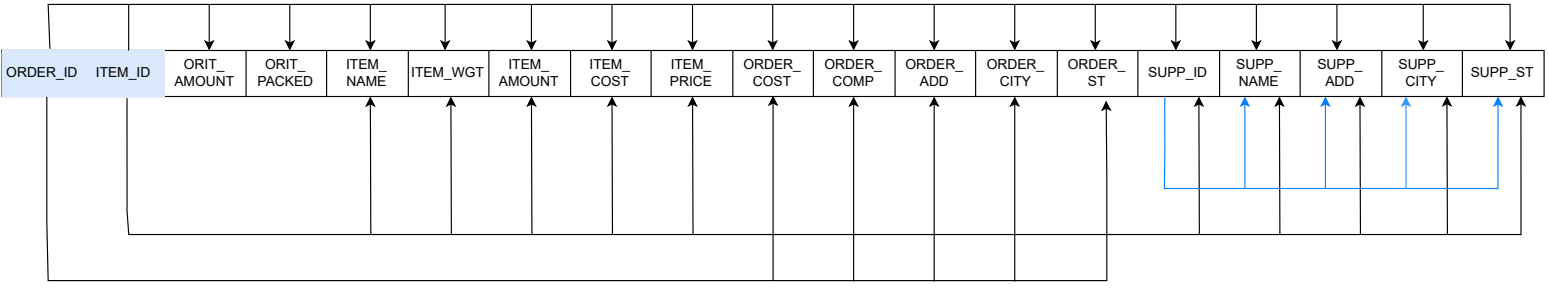


Table: ORDER_ITEM

ORDER_ID	ITEM_ID	ORIT_AMOU	ORIT_PACKE
1	1	5	5
1	2	5	5
1	3	2	2
1	7	50	50
1	9	50	50
2	1	2	2
2	8	30	30
3	12	1	1
4	3	4	2

Table: ORDER

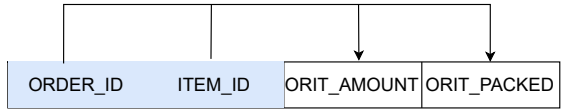
ORDER_ID	ORDER_COM	ORDER_COST	ORDER_ADD	ORDER_CITY	ORDER_ST
1	1	300.97	Another Street	Somwhere	IN
2	1	21.50	Some Street	Somewhere	IN
3	1	99.99	Pennsyl vania Avn	DC	DC
4	0	101.10	466 Oak Brv	Chicago	IL

2NF and DEPENDENCY DIAGRAMS

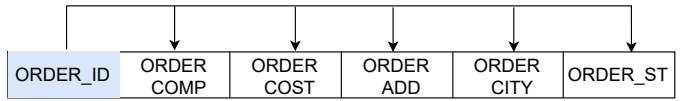
As shown in the following tables, the primary keys of ORDER_ITEM are also foreign keys to the tables ORDER and ITEM. This takes away the partial dependency and redundancy by putting the details of specific items and orders into seperate tables for the entities. In the meantime, ORDER_ITEM retains the combination specific attributes of ORIT_AMOUNT and ORIT_PACKED, standing for the specific amount of the item in that specific order in amount requested and amount packed in packages currently. However, there still remains the partial dependency of (SUPP_ID -> SUPP_NAME, SUPP_ADD, SUPP_CITY, SUPP_ST), which must be made into a final table SUPPLIER to finally be in a 3NF state.

Table: ITEM

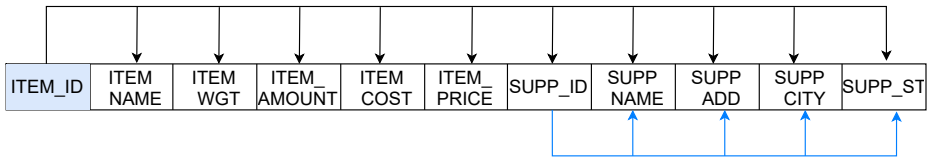
ITEM_ID	ITEM_NAME	ITEM_WGT	ITEM_AMOU	ITEM_COST	ITEM_PRICE	SUPP_ID	SUPP_NAME	SUPP_ADD	SUPP_CITY	SUPP_ST
1	HeaterCoil	3.50	102	7.00	10.00	1	Mikel Company	444 Parkway St.	Detroit	MI
2	Bufferter	25.00	8	25.00	30.00	1	Mikel Company	444 Parkway St	Detroit	MI
3	CTRL_ELEC	1.20	13	19.00	25.00	2	TechFutures	333 Narkway St	Los Angeles	CA
7	SCREW-0.5	0.05	3450	0.01	0.05	4	ScrewyNails	111 Clarkway St	Zionsville	IN
8	SCREW-0.1	0.02	4470	0.05	0.10	4	ScrewyNails	111 Clarkway St	Zionsville	IN
9	NAIL-1.1	0.02	2950	0.03	0.05	4	ScrewyNails	111 Clarkway St	Zionsville	IN
12	Washer Frame	250.50	7	300.00	599.99	1	Mikel Company	444 Parkway St	Detroit	MI



ORDER_ITEM(**ORDER_ID, ITEM_ID**, ORIT_AMOUNT, ORIT_PACKED)



ORDER(**ORDER_ID**, ORDER_COMP, ORDER_COST, ORDER_ADD, ORDER_CITY, ORDER_ST)



ITEM(**ITEM_ID,** ITEM_NAME, ITEM_WGT, ITEM_AMOUNT, ITEM_COST, ITEM_PRICE, SUPP_ID, SUPP_NAME, SUPP_ADD, SUPP_CITY, SUPP_ST)

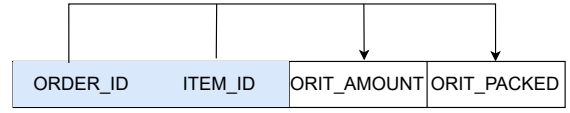
Partial Dependencies:
(SUPP_ID --> SUPP_NAME, SUPP_ADD, SUPP_CITY, SUPP_ST)

Table: ITEM

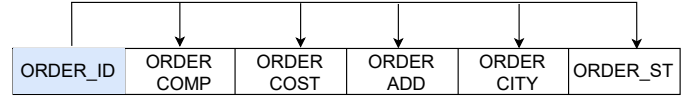
ITEM_ID	ITEM_ NAME	ITEM _WGT	ITEM_ AMOUNT	ITEM_ COST	ITEM_ PRICE	SUPP_ID
1	HeaterCoil	3.50	102	7.00	10.00	1
2	Bufferter	25.00	8	25.00	30.00	1
3	CTRL_ELEC	1.20	13	19.00	25.00	2
7	SCREW-0.5	0.05	3450	0.01	0.05	4
8	SCREW-0.1	0.02	4470	0.05	0.10	4
9	NAIL-1.1	0.02	2950	0.03	0.05	4
12	Washer Frame	250.50	7	300.00	599.99	1

Table: ORDER_ITEM

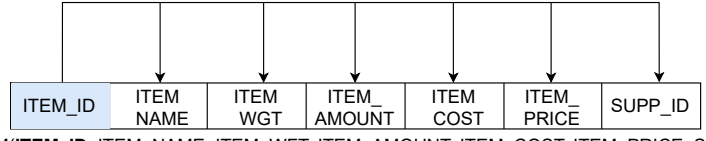
ORDER_ID	ITEM_ID	ORIT_AMOU	ORIT_PACKE
1	1	5	5
1	2	5	5
1	3	2	2
1	7	50	50
1	9	50	50
2	1	2	2
2	8	30	30
3	12	1	1
4	3	4	2



ORDER_ITEM(**ORDER_ID**, **ITEM_ID**, ORIT_AMOUNT, ORIT_PACKED)



ORDER(**ORDER_ID**, ORDER_COMP, ORDER_COST, ORDER_ADD, ORDER_CITY, ORDER_ST)



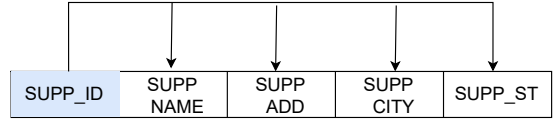
ITEM(**ITEM_ID**, ITEM_NAME, ITEM_WFT, ITEM_AMOUNT, ITEM_COST, ITEM_PRICE, SUPP_ID)

Table: SUPPLIER

SUPP_ID	SUPP_NAI	SUPP_ADI	SUPP_CIT	SUPP_ST
1	Mikel Company	444 Parkway St	Detroit	MI
2	TechFuture	333 Narkway St	Los Angeles	CA
4	ScrewyNai	111 Clarkway St	Zionsville	IN

Table: Order

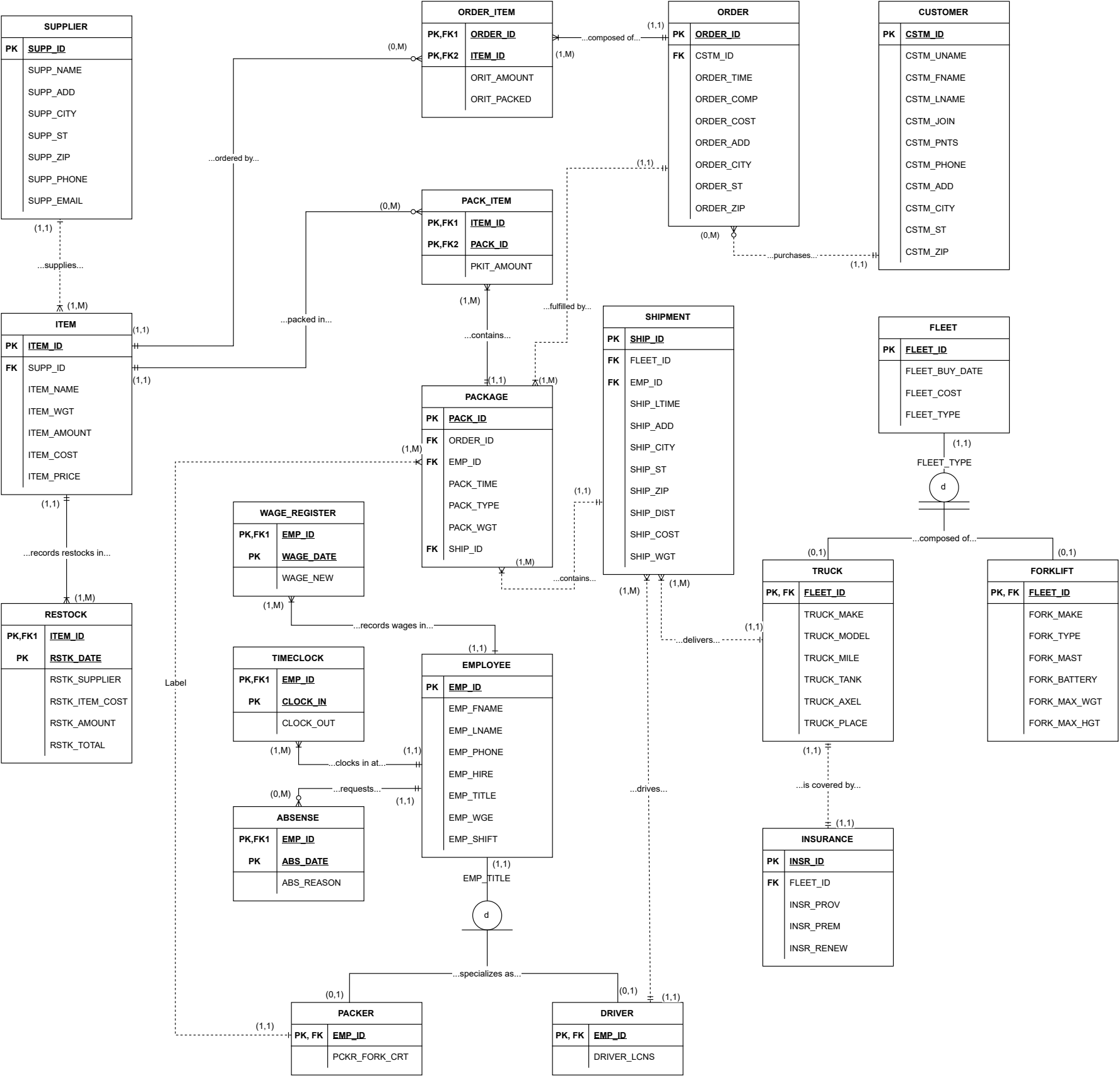
ORDER_ID	ORDER_COM	ORDER_COS	ORDER_ADD	ORDER_CITY	ORDER_ST
1	1	300.97	Another Street	Somwhere	IN
2	1	21.50	Some Street	Somewhere	IN
3	1	99.99	Pennsyl vania Avn	DC	DC
4	0	101.10	466 Oak Brv	Chicago	IL



SUPPLIER(**SUPP_ID**, SUPP_NAME, SUPP_ADD, SUPP_CITY, SUPP_ST)

3NF and Dependency Diagrams

With SUPPLIERS given their own table, connection to items via a foreign key to dictate that the item in question is supplied only by that supplier at this current time, and has only one supplier at any given moment. Thus all partial dependency and transitive dependency is taken care of.


SUPPLIER
PK SUPP_ID
SUPP_NAME
SUPP_ADD
SUPP_CITY
SUPP_ST
SUPP_ZIP
SUPP_PHONE
SUPP_EMAIL
ORDER_ITEM
PK,FK1 ORDER_ID
PK,FK2 ITEM_ID
ORIT_AMOUNT
ORIT_PACKED
ORDER
PK ORDER_ID
FK CSTM_ID
ORDER_TIME
ORDER_COMP
ORDER_COST
ORDER_ADD
ORDER_CITY
ORDER_ST
ORDER_ZIP
CUSTOMER
PK CSTM_ID
CSTM_UNAME
CSTM_FNAME
CSTM_LNAME
CSTM_JOIN
CSTM_PNTS
CSTM_PHONE
CSTM_ADD
CSTM_CITY
CSTM_ST
CSTM_ZIP
PACK_ITEM
PK,FK1 ITEM_ID
PK,FK2 PACK_ID
PKIT_AMOUNT
ITEM
PK ITEM_ID
FK SUPP_ID
ITEM_NAME
ITEM_WGT
ITEM_AMOUNT
ITEM_COST
ITEM_PRICE
SHIPMENT
PK SHIP_ID
FK FLEET_ID
FK EMP_ID
SHIP_LTIME
SHIP_ADD
SHIP_CITY
SHIP_ST
SHIP_ZIP
SHIP_DIST
SHIP_COST
SHIP_WGT
FLEET
PK FLEET_ID
FLEET_BUY_DATE
FLEET_COST
FLEET_TYPE
PACKAGE
PK PACK_ID
FK ORDER_ID
FK EMP_ID
PACK_TIME
PACK_TYPE
PACK_WGT
FK SHIP_ID
WAGE_REGISTER
PK,FK1 EMP_ID
PK WAGE_DATE
WAGE_NEW
RESTOCK
PK,FK1 ITEM_ID
PK RSTK_DATE
RSTK_SUPPLIER
RSTK_ITEM_COST
RSTK_AMOUNT
RSTK_TOTAL
TIMECLOCK
PK,FK1 EMP_ID
PK CLOCK_IN
CLOCK_OUT
EMPLOYEE
PK EMP_ID
EMP_FNAME
EMP_LNAME
EMP_PHONE
EMP_HIRE
EMP_TITLE
EMP_WGE
EMP_SHIFT
ABSENSE
PK,FK1 EMP_ID
PK ABS_DATE
ABS_REASON
TRUCK
PK, FK FLEET_ID
TRUCK_MAKE
TRUCK_MODEL
TRUCK_MILE
TRUCK_TANK
TRUCK_AXEL
TRUCK_PLACE
FORKLIFT
PK, FK FLEET_ID
FORK_MAKE
FORK_TYPE
FORK_MAST
FORK_BATTERY
FORK_MAX_WGT
FORK_MAX_HGT
INSURANCE
PK INSR_ID
FK FLEET_ID
INSR_PROV
INSR_PREM
INSR_RENEW
PACKER
PK, FK EMP_ID
PCKR_FORK_CRT
DRIVER
PK, FK EMP_ID
DRIVER_LCNS
...composed of...
...ordered by...
...supplies...
...packed in...
...contains...
...fulfilled by...
...purchases...
...records restocks in...
...records wages in...
...clocks in at...
...requests...
...delivers...
...drives...
...is covered by...
...specializes as...
FLEET_TYPE
EMP_TITLE
Label
d
(1,1)
(1,M)
(0,M)
(0,1)

Question 1:

Select employees hired by year (excluding dummy)

Answer-Query:

```
SELECT YEAR(EMP_HIRE), COUNT(*) as NUMBER_HIRED FROM EMPLOYEE
WHERE EMP_FNAME NOT LIKE 'DUMMY' GROUP BY YEAR(EMP_HIRE);
```

Question 2:

Select number of screws ordered in entire life-span of order, along with their total value based on current average selling price of that specific screw product, summed together.

Answer-Query:

```
SELECT

SUM(ORIT_AMOUNT) AS SCREWS_ORDERED,

SUM(ORIT_AMOUNT * ITEM_PRICE) AS DISCOUNTLESS_PRICE_TOTAL

FROM ORDER_ITEM INNER JOIN ITEM USING(ITEM_ID) WHERE
lower(ITEM_NAME) like "%screw%";
```

Question 3:

Select packages whose record weight differed from total average item weights by less than 10 pounds.

Answer-Query:

```
SELECT * FROM PACKAGE INNER JOIN

(SELECT SUM(PKIT_AMOUNT * ITEM_WGT) AS TOTAL_WGTS, PACK_ID

FROM PACK_ITEM INNER JOIN ITEM USING(ITEM_ID)

GROUP BY PACK_ID) AS SUM_TAB

USING(PACK_ID) WHERE ABS(TOTAL_WGTS - PACK_WGT) < 10
```