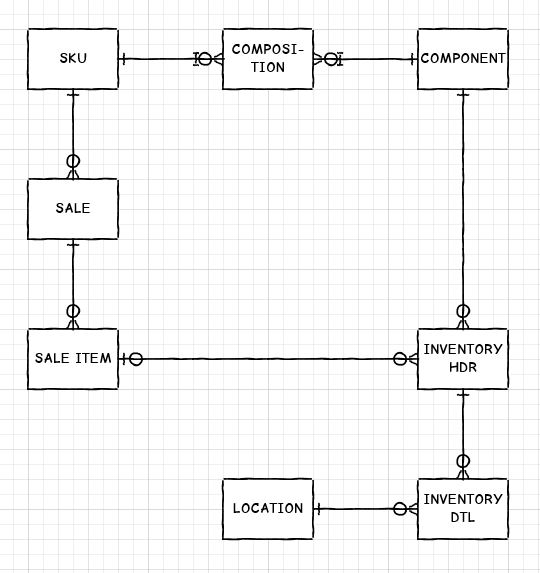
The solution you are looking for will rely on an accounting style model and a couple of bills of materials (BOM). Your major entity types will include:

* **SKU** This is the list of things that you sell. It's properties will include things like product description and current retail price. You can get fancy and break price out into a child table that gives prices over time. Let's assume that you are going to leave that wrinkle out for now. Some SKUs can be "combos" of the sort you are talking about.
* **COMPONENT** This is the list of things that make up a SKU, such as napkins, cups, buns, patties, coke syrup etc. - to use your example. Just as SKU has descriptions and prices, COMPONENTs have descriptions and unit costs. (Which can also be historized in a child table.) This table is where you would typically store your ROP too.
* **COMPOSITION** This is a BOM which intersects SKU and COMPONENT and says how many units of each COMPONENT go into a unit of a SKU. You need one of these to intersect two SKUs too (for combos). You can either use one table or two tables for this. Two tables will keep the purists happy, one table will be expedient from a coder point of view.
* **SALE** This is a transaction table that provides a header for recording a sale of one or more SKUs. This table would have things like transaction date, cashier ID, and other header items.
* **SALE\_ITEM** This is the transaction detail table that would include which SKU was sold (and how many) and for how much. The how much is a denormalization of the SKU price at time of sale, but could also include any special overrides to the price. The price actually charged for the SKU is a good thing to denormalize because someone could edit the list price in SKU and then you'd lose track of how much was actually charged for the item at the time.
* **INVENTORY\_HDR** This is a transactional table that is similar to the SALE conceptually, but it is the header for an inventory transaction, such as receiving new inventory, using up inventory (as in selling it) and for inventory adjustments. Again, this would be date/description stuff, but it can include a direct link to a SALE\_ITEM for inventory movements that are sales if you like. You don't have to do it that way, but some people like to establish the connection between revenues and costs on a transaction by transaction basis.
* **INVENTORY\_DTL** This is the detail for an inventory transaction. This indicates which COMPONENT is going in or out, the quantity that went in or out, and the INVENTORY\_HDR transaction that this movement applied to. This would also be where you keep the actual cost paid for the component item.
* **LOCATION** You can (if you wish) also track the physical location of the inventory that you receive and use/sell. In a restaurant this may not be important but if you have a chain or if your restaurant has an offsite warehouse for component ingredients then you might care.

Consider the following ERD: [](http://i.stack.imgur.com/jB83E.jpg)

To do your revenue accounting you would be adding up the money recorded in the SALE\_ITEM table.

**Stock levels are calculated** based on adding up the INVENTORY\_DTL ins and outs for each COMPONENT. (Don't store current stock levels in a table - This is doomed to cause reconciliation problems.)

To do your cost accounting you would be adding up the money recorded in the INVENTORY\_DTL table. Note that you won't usually know exactly which napkin or bun you sold, so it won't be possible to link specific component reciepts with specific SKU sales. Instead, you need to have a convention for determining which components were used for any given SKU. You may have accounting rules that specify what convention you are required to use. Most people use FIFO. Some industries use LIFO and I've even seen weighted average cost accounting.