Many of the demos will use the following sets of tables belong to altgeld_mart. The tables will be in three different databases. With MySQL we can create relationships across databases.

I suggested creating the following databases in document 01-02. If you did not do that then, create them now. Login as root and create the three databases: a_emp, a_prd, a_oe. Then give your regular user account permissions to those databases.

The database a_emp will store tables relating to employees. The database a_prd will store tables relating to products we sell. The database a_oe will store tables relating to orders placed and customers.

This discusses the business rules and the relationships. You should read the Create Table statements for the details. Those statements are in the supplied scripts.

Our company (Altgeld_Mart) stores data about employees and the products sold and customer orders. These tables store only some of the data that would be needed by a company.

Altgeld_Mart sells a variety of products. Each product is classified as being in a category such as Housewares, Pet Supplies, or Sporting Goods. The products are stored in different warehouses; a product might be stored in more than one warehouse.

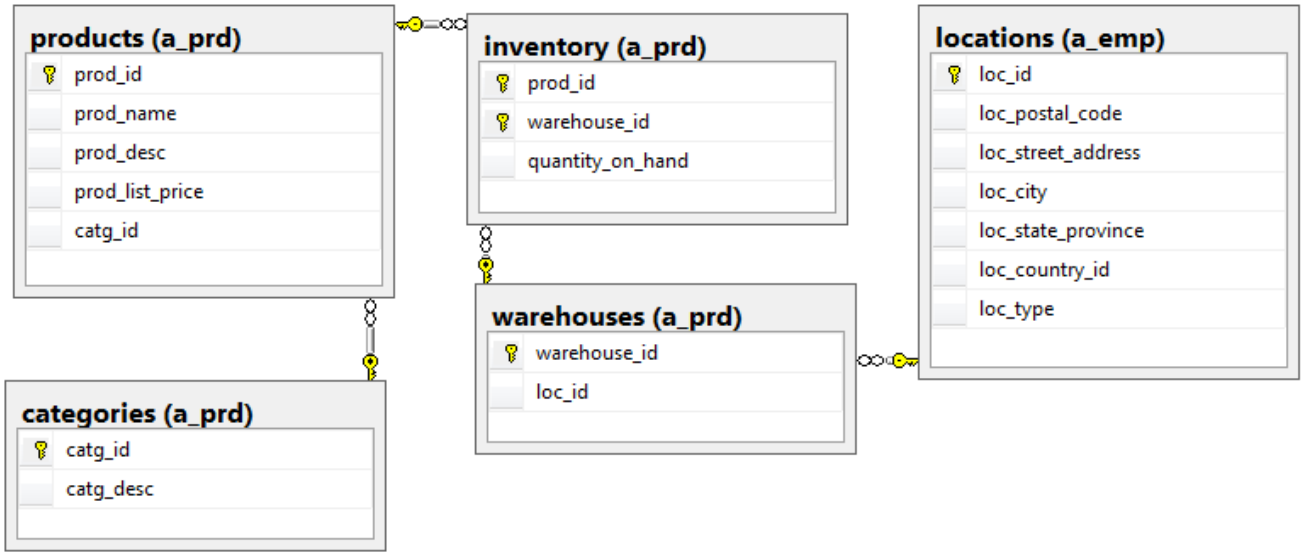
**categories**: Each product is classified as being in a category such as Housewares, Pet Supplies, and Sporting Goods. This table is used to limit the values that can be entered as a category for a product.

**products**: A product has a name (fairly short) and a description (which could be longer). A product has only a single category. Products also have a list price but we do not always sell items at their list price.

**warehouse**: The products are stored in different warehouses; a product might be stored in more than one warehouse. The loc_id for the warehouses is associated with the locations table.

**locations**: The locations for the company sites are stored in a table of locations; we store the address information for each location.

**inventory**: this keeps track of how many of each item we have at each warehouse.



These are the tables associated with the order entry component.

**customers**: For customers we are storing only the customer name and credit limit to keep the tables smaller.

**credit_rating** This table is unusual in that it is not directly related to any other table. This includes a descriptive term and a lower and upper limit. If a customer has a credit rating between 5001 and 10000, we will say that this customer has an "Excellent" rating.

**order_headers**: Each sales order belongs to a single customer; an order has an order mode, a shipping mode and a numeric order status. For some sales we know the sales rep who handled the order.

**order_details**: The order details tables includes the product being ordered, the quantity and the price at which the item was actually sold. By putting the actual sales price here we could sell a product for less or more than its list price. If the list price changes, the actual price to the customer is not changed. The product id references the products table in the previous diagram.

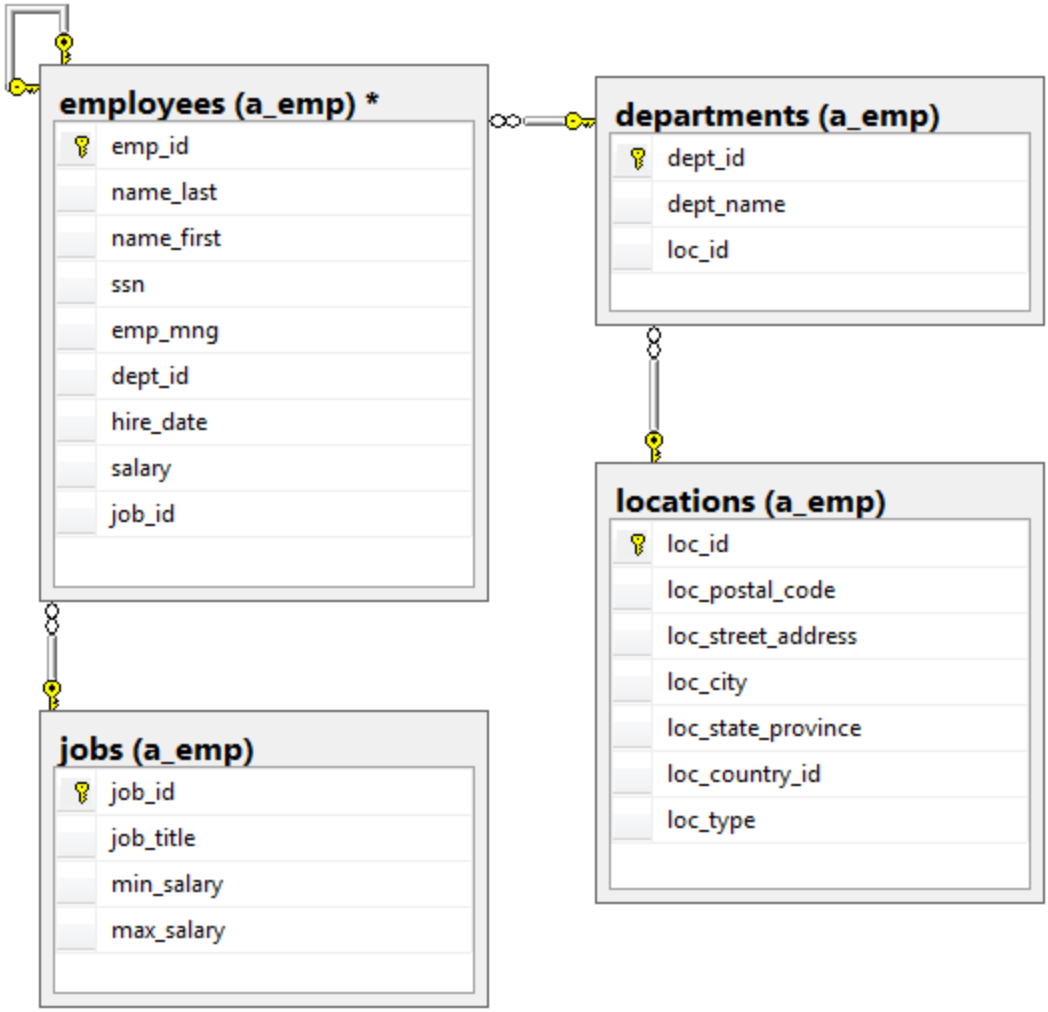
The price in the order_details table is the price per item. For example, on order 114 the customer bought 5 mini freezers for $125.00 each. For this order the customer is charged $625.00 ( 5 * 125) plus any tax and shipping.

**shipping_mode**: this table is similar to the product categories table. It lists the various types of shipping modes we use and each order is limited to one of these shipping modes.

**employees**: We store an employee's name. An employee has a single job title and is assigned to a single department. An employee may have a manager who is also an employee. This results in a relationship from the employees table back to itself. We store the employee's hire date and current salary.

**jobs:** Each employee has a single job title and there is a table of the valid job titles. This also contains a minimum and maximum salary for that job

**departments**: For each department, we store a department name and a single location.



The next diagram shows the relations between all the tables.