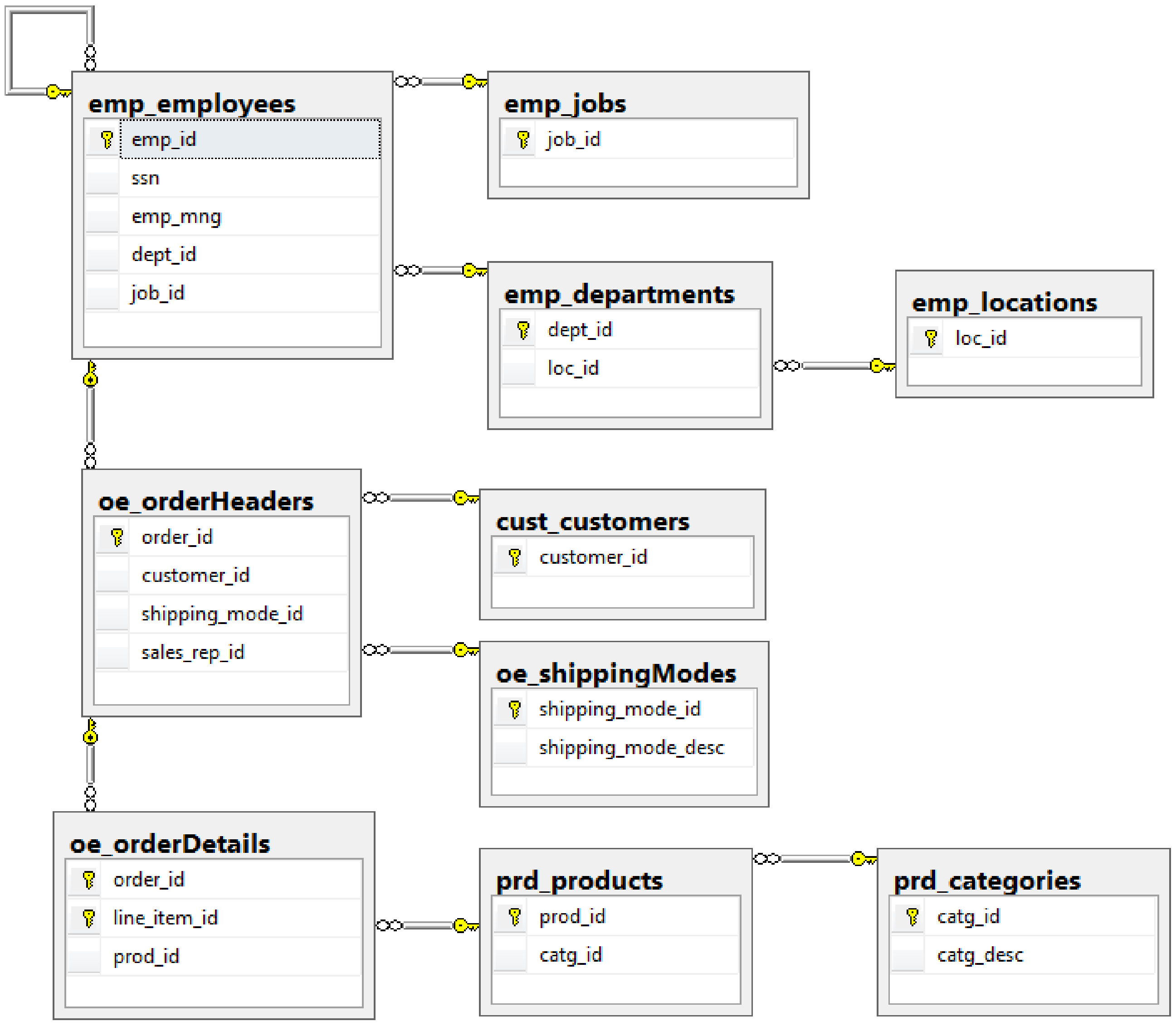
Many of the demos will use the following sets of tables belong to AltgeldMart. This discusses the business rules and the relationships for this database. You should read the Create Table statements for the details. Those statements are in the supplied scripts.

Our company (AltgeldMart) 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.

I am going to use diagrams produced by SQL Server because they have better diagramming tools. The table names will be the same. The column names are the same and the data types are compatible with Oracle.

This first diagram shows the relations between the tables. I have limited the display to show only the table names and the keys (pk and fk) attributes.



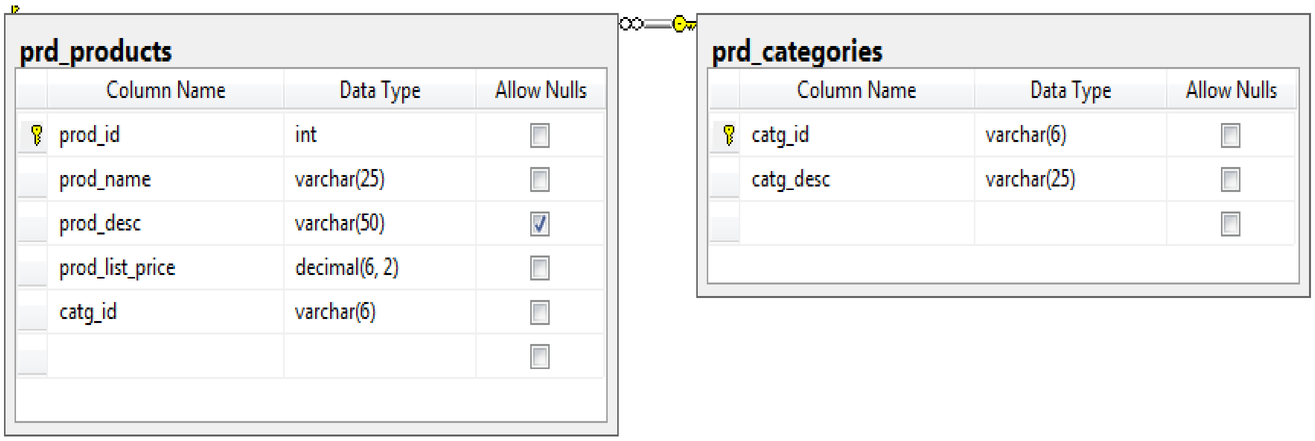
PRODUCT

Altgeld_Mart sells a variety of products.

prd_categories: Each product is classified as being in a category such as Housewares, Pet Supplies, Sporting Goods. This table is used to limit the values that can be entered as a category for a product. The table has a short value for each category as the primary key and a longer descriptive value. The descriptive values are unique. For example we have the rows:

catg_id	catg_desc
APL	APPLIANCES
BK	BOOKS

prd_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.



prd_products

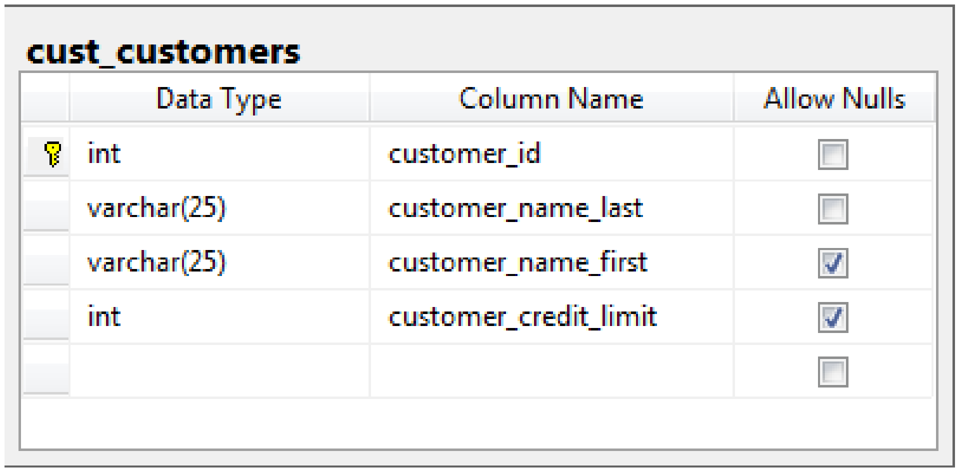
	Column Name	Data Type	Allow Nulls
🔑	prod_id	int	☐
	prod_name	varchar(25)	☐
	prod_desc	varchar(50)	☑
	prod_list_price	decimal(6, 2)	☐
	catg_id	varchar(6)	☐
			☐

prd_categories

	Column Name	Data Type	Allow Nulls
🔑	catg_id	varchar(6)	☐
	catg_desc	varchar(25)	☐
			☐

CUSTOMER

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



cust_customers

	Data Type	Column Name	Allow Nulls
🔑	int	customer_id	☐
	varchar(25)	customer_name_last	☐
	varchar(25)	customer_name_first	☑
	int	customer_credit_limit	☑
			☐

ORDERENTRY

These are the tables associated with the order entry component.

oe_orderHeaders: Each sales order belongs to a single customer(customer_id) ; an order has an order_date, a order mode, a shipping mode and a numeric order status. For some sales we know the sales rep (employee) who handled the order.

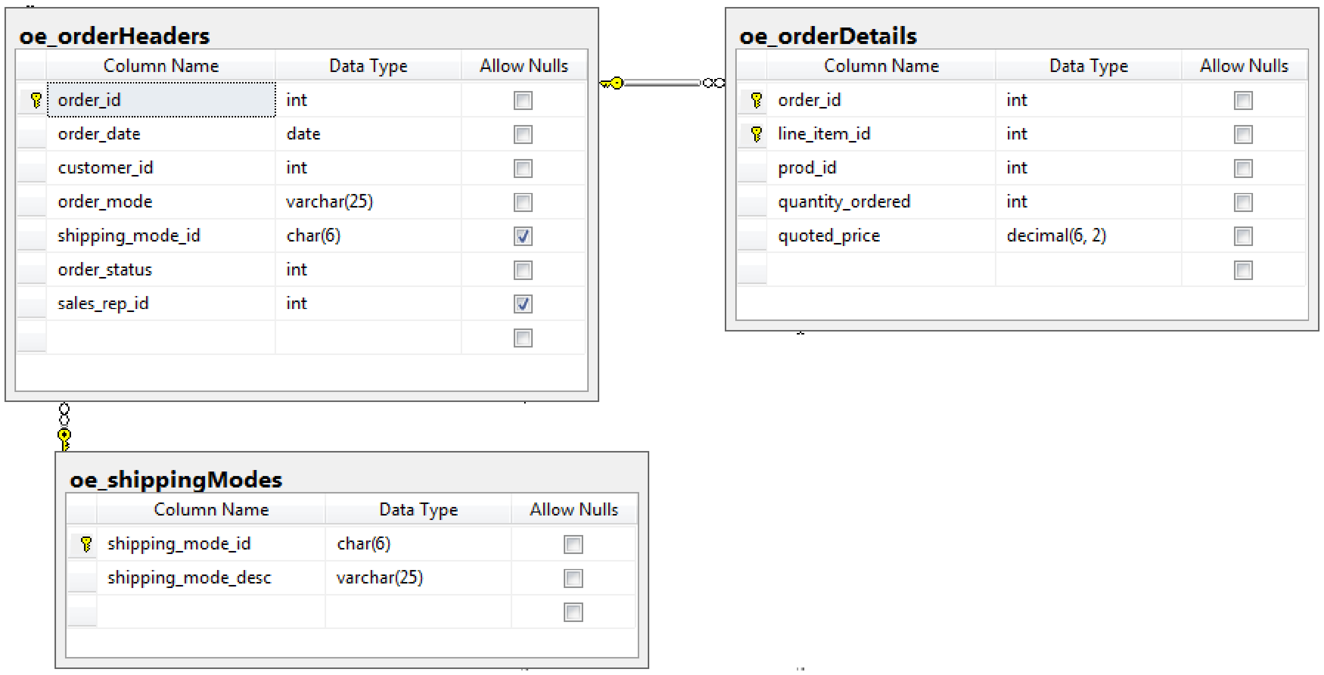
oe_orderDetails: The order details tables includes the product being ordered, the quantity and the price at which the item was actually sold. By putting the actual sales price here, we could sell a product for less or more than its list price. If the list price changes, the actual price to the customer 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.

The orderDetails table include the order_id to link to the orderHeaders table. It also has a line_item_id- usually numbered 1,2,3, etc. Together order_id and line_item_id make up the primary key for the orderDetails table.

.

oe_shippingModes: 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.



oe_orderHeaders

Column Name	Data Type	Allow Nulls
order_id	int	☐
order_date	date	☐
customer_id	int	☐
order_mode	varchar(25)	☐
shipping_mode_id	char(6)	✓
order_status	int	☐
sales_rep_id	int	✓
		☐

oe_orderDetails

Column Name	Data Type	Allow Nulls
order_id	int	☐
line_item_id	int	☐
prod_id	int	☐
quantity_ordered	int	☐
quoted_price	decimal(6, 2)	☐
		☐

oe_shippingModes

Column Name	Data Type	Allow Nulls
shipping_mode_id	char(6)	☐
shipping_mode_desc	varchar(25)	☐
		☐

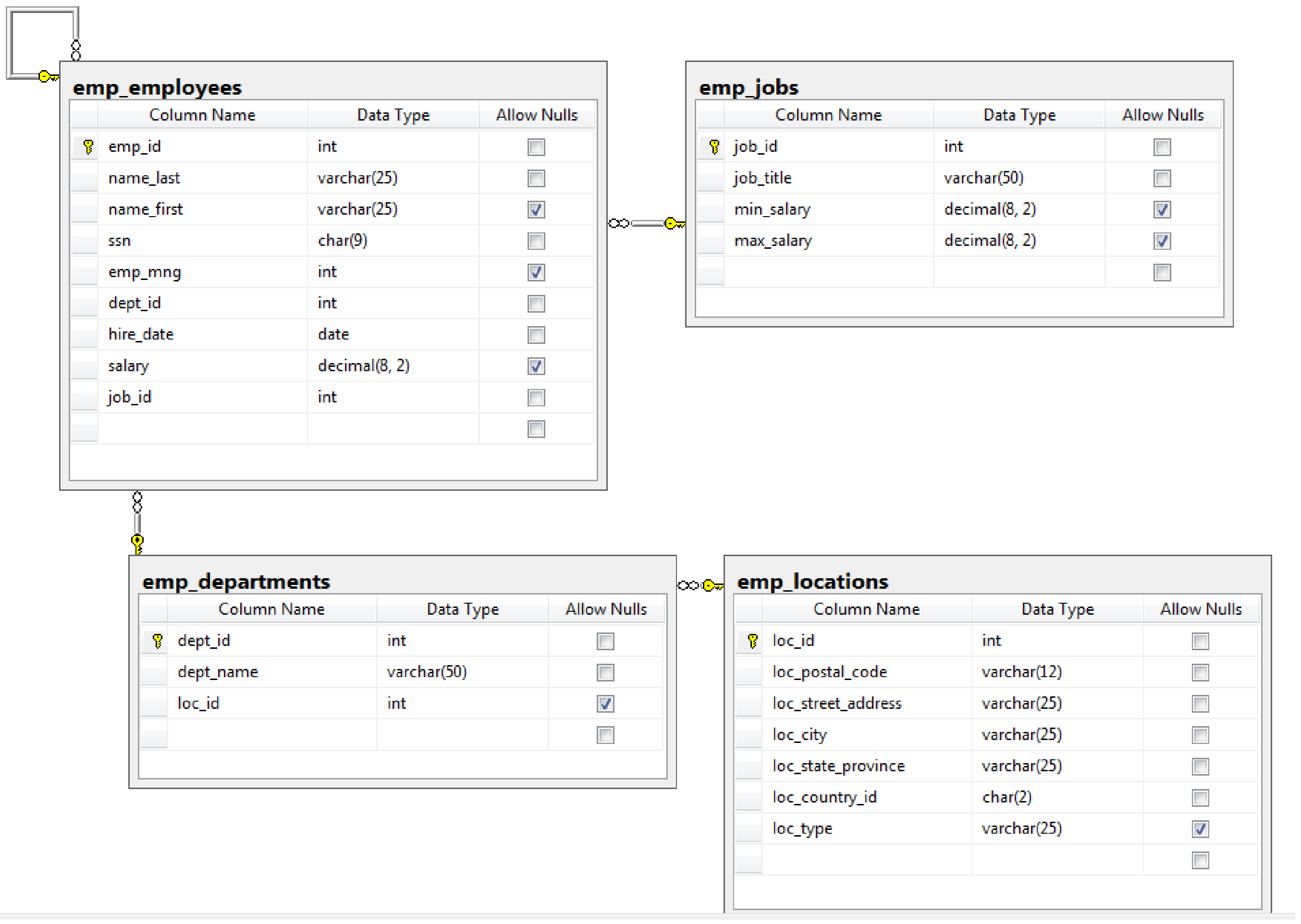
EMPLOYEE

emp_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.

emp_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

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

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



emp_employees

	Column Name	Data Type	Allow Nulls
🔑	emp_id	int	☐
	name_last	varchar(25)	☐
	name_first	varchar(25)	✓
	ssn	char(9)	☐
	emp_mng	int	✓
	dept_id	int	☐
	hire_date	date	☐
	salary	decimal(8, 2)	✓
	job_id	int	☐
			☐

emp_jobs

	Column Name	Data Type	Allow Nulls
🔑	job_id	int	☐
	job_title	varchar(50)	☐
	min_salary	decimal(8, 2)	✓
	max_salary	decimal(8, 2)	✓
			☐

emp_departments

	Column Name	Data Type	Allow Nulls
🔑	dept_id	int	☐
	dept_name	varchar(50)	☐
	loc_id	int	✓
			☐

emp_locations

	Column Name	Data Type	Allow Nulls
🔑	loc_id	int	☐
	loc_postal_code	varchar(12)	☐
	loc_street_address	varchar(25)	☐
	loc_city	varchar(25)	☐
	loc_state_province	varchar(25)	☐
	loc_country_id	char(2)	☐
	loc_type	varchar(25)	✓
			☐

You might notice the looped relation from employees to employees. This is called a pig's ear. What this relation says is that an employee generally has a manager who is also an employee.