Conceptual Model
For E-Shop

Table of Contents

1. Describe the database in text

We will develop an ecommerce system for coffee shop.
First, the database will handle a customer with contact details(like name, email, and address), and a product(with product code, name, short description and price) where each product is in one or more product categories.

The database also needs to contain a warehouse where you can see how many of each product are in the warehouse and a note about where the product is in the warehouse (which shelf). One and the same product can be spread over different shelves in the warehouse.

When the customer orders a product, an order is created that contains the customer's details together with which products have been ordered and its ordered number.

Based on the order, a pick list is created that can be sent to the warehouse for delivery. The pick list contains the same information as the order, but with the addition that each product line is mapped to a warehouse shelf so that the warehouse staff can see which shelf they can pick up the product on.

When the delivery is packed, an invoice is attached that has the same content as the order but now with the price per product line and the summed price.

There should be a log where you can see important events in the system, what happened, when it happened. This can be, for example, when an order / invoice was created or deleted.

2. All entities

We will have the entities as the following:

- Customer (id, name, email, address)
- Product (code, name, category, description, price)
- Category (type)
- Warehouse (name, order, shelf, product code, number of product)
- Order (id, date, customer, total_price)
- Order_detail (order, product, number of product)
- Invoice (id, date, order, summed price)
- Log (id, date, event type, event description)

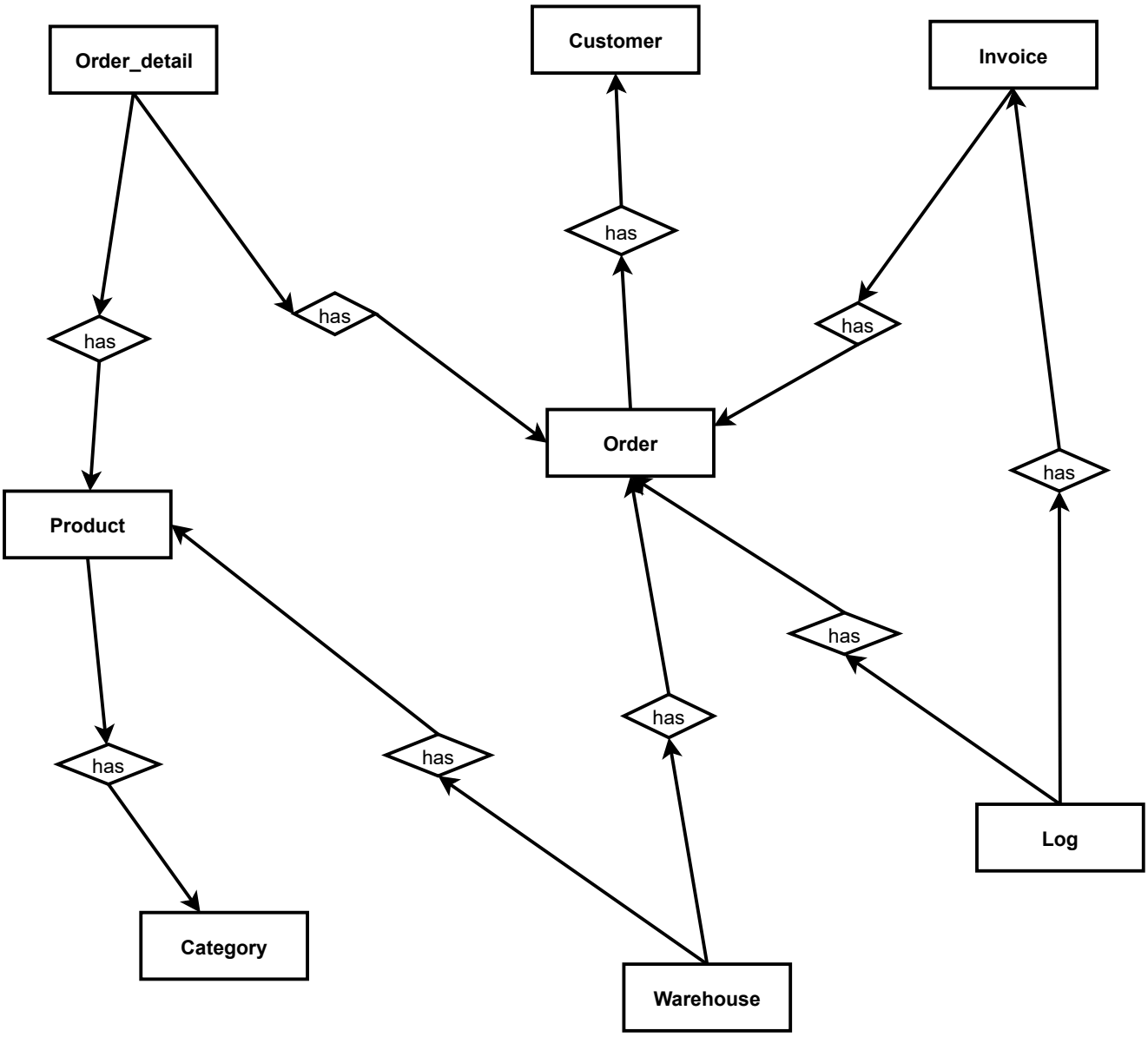
3. Matrix of relationships

We will have the entities as the following:

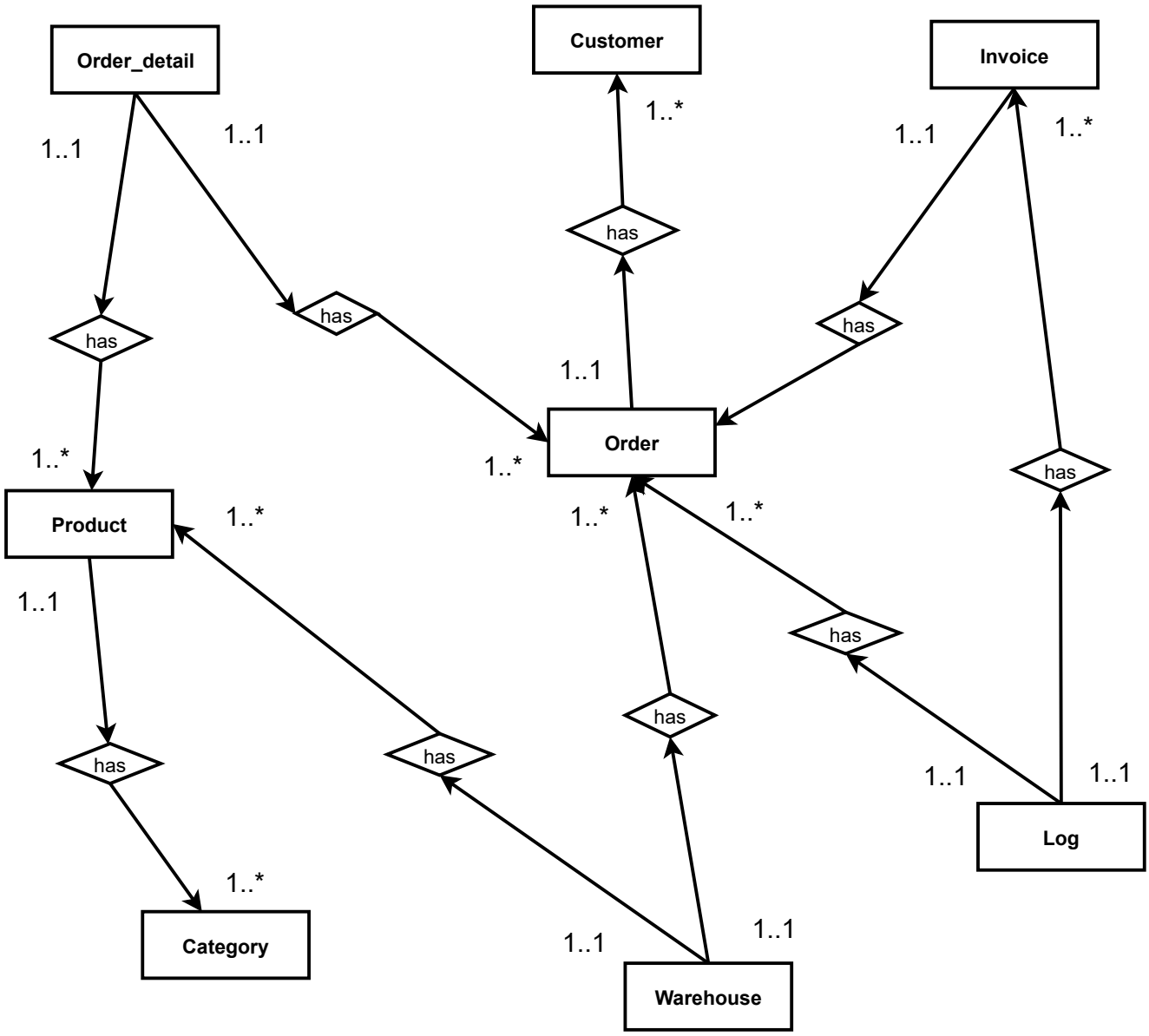
- All products are categorized by one category.
- One order has a number of products.
- One invoice has one order.
- All products are stored on warehouse shelves.
- A customer has several orders.
- All events are logged.

Entities	Customer	Product	Category	Warehouse	Order	Order_detail	Invoice	Log
Customer								
Product			has					
Category								
Warehouse		has			has			
Order	has					has		
Order_detail		has			has			
Invoice					has			
Log					has		has	

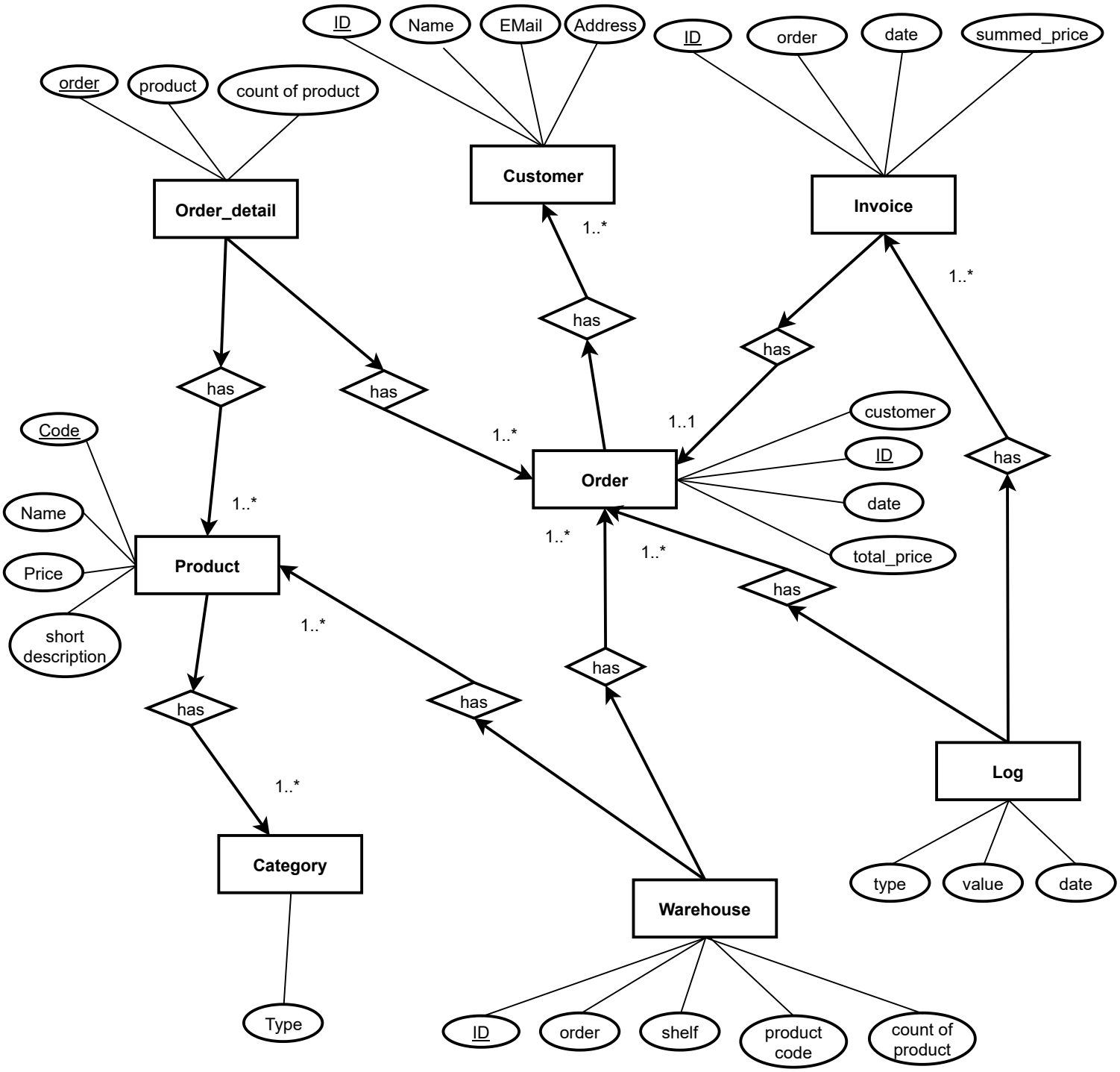
4. ER diagram with entities and relationships



5. ER chart with cardinality

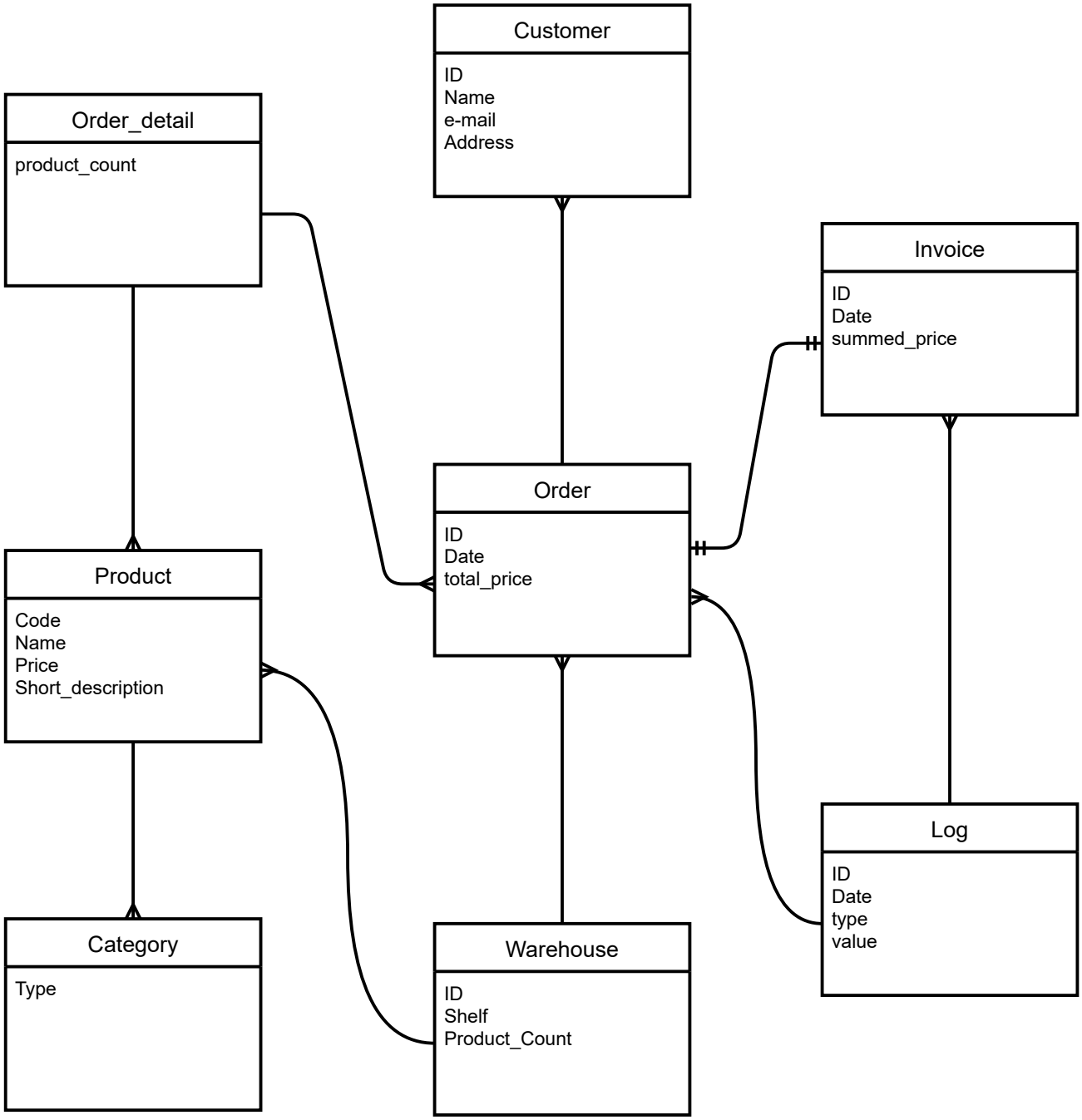


6. ER diagram with attributes and candidate keys

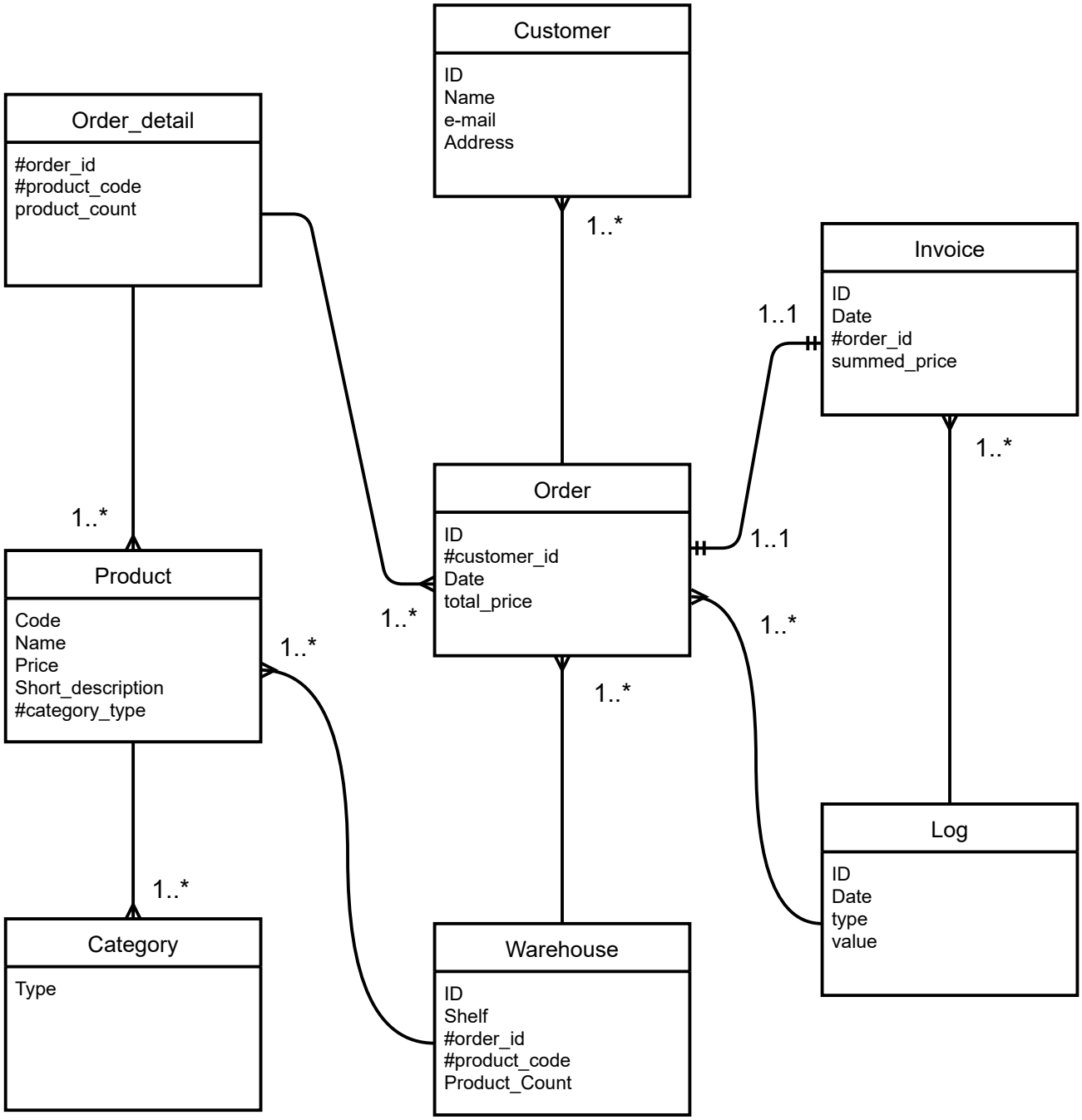


Logical Model For E-Shop

7. Modify the ER diagram according to the relational model



8. Extend the ER diagram with the primary and foreign keys as well as complementary attributes



Logical Model (text)

- Customer (id, name, email, address)
- Product (code, name, #category_type, description, price)
- Category (type)
- Warehouse (name, shelf, #order_ID, #product code, product_count)
- Order (id, date, #customer_ID, total_price)
- Order_detail (#order_ID, #product_code, product_count)
- Invoice (id, date, #order_ID, summed price)
- Log (id, date, type, value), if type='order' then value=order_id else if type='invoice' then value=invoice_id

Physical Model For E-Shop

9. Create SQL DDL for the tables

```
CREATE  TABLE Customer
(
  id  INT PRIMARY KEY  NOT  NULL ,
  name CHAR ( 100 ) NOT  NULL ,
  email VARCHAR ( 255 ) NOT  NULL ,
  address VARCHAR ( 255 ) NOT  NULL
);

CREATE  TABLE Product
(
  code INT PRIMARY KEY  NOT  NULL ,
  name VARCHAR ( 255 ) NOT  NULL ,
  short_description VARCHAR ( 255 ) ,
  category_type VARCHAR ( 255 ) NOT  NULL,
   FOREIGN  KEY (category_type) REFERENCES Category (type)
);

CREATE  TABLE Category
(
  type VARCHAR ( 255 ) PRIMARY KEY NOT  NULL
);

CREATE  TABLE Order
(
  id  INT PRIMARY KEY  NOT  NULL ,
  date DATETIME NOT  NULL ,
  customer_id INT NOT  NULL ,
   FOREIGN  KEY (customer_id) REFERENCES Customer (id),
  total_price FLOAT(8, 4) NOT NULL
);
```

```
CREATE  TABLE Order_detail
(
  order_id INT NOT  NULL,
   FOREIGN  KEY (order_id) REFERENCES Order (id),
  product_code INT NOT  NULL,
   FOREIGN  KEY (product_code) REFERENCES Product (code),
  product_count INT NOT  NULL
);

CREATE  TABLE Invoice
(
  id INT PRIMARY KEY  NOT NULL ,
  order_id INT NOT  NULL,
   FOREIGN  KEY (order_id) REFERENCES Order (id),
  date DATETIME NOT NULL,
  summed_price FLOAT(8, 4) NOT NULL
);

CREATE  TABLE Warehouse
(
  id INT PRIMARY KEY NOT  NULL ,
  order_id INT NOT NULL,
   FOREIGN  KEY (order_id) REFERENCES Order (id),
  product_code INT NOT NULL,
   FOREIGN  KEY (product_code) REFERENCES Product (code),
  shelf VARCHAR (255) NOT NULL,
  product_count INT NOT NULL
);
```

```
CREATE  TABLE Log
(
  id  INT PRIMARY KEY  NOT  NULL ,
  date DATETIME NOT  NULL ,
  type VARCHAR (255) NOT NULL,
  value INT NOT NULL
);
```

10. List of functions that the database should support (API)

1. Add new customer
2. Update customer
3. Delete customer
4. Add new product
5. Update product
6. Delete product
7. Add new order
8. Update order
9. Delete order
10. Add new invoice
11. Update invoice
12. Delete invoice
13. Add new log
14. Show all product list
15. Show order detail
16. Show invoice detail
17. Show product detail
18. Search product list