期末课设：车辆信息管理系统

学号：2311983　　姓名：余辰民

1. 应用领域需求描述

车辆信息管理系统旨在对各类车辆的相关信息进行集中、高效的管理。该系统主要面向汽车制造商、经销商、消费者、汽车爱好者以及汽车行业的研究人员等，满足他们对车辆品牌、车型、售价、上市年份等信息的查询、统计和管理需求。在此次设计中，我主要以宝马品牌为例，后续可添加其他品牌及型号。

具体需求如下：

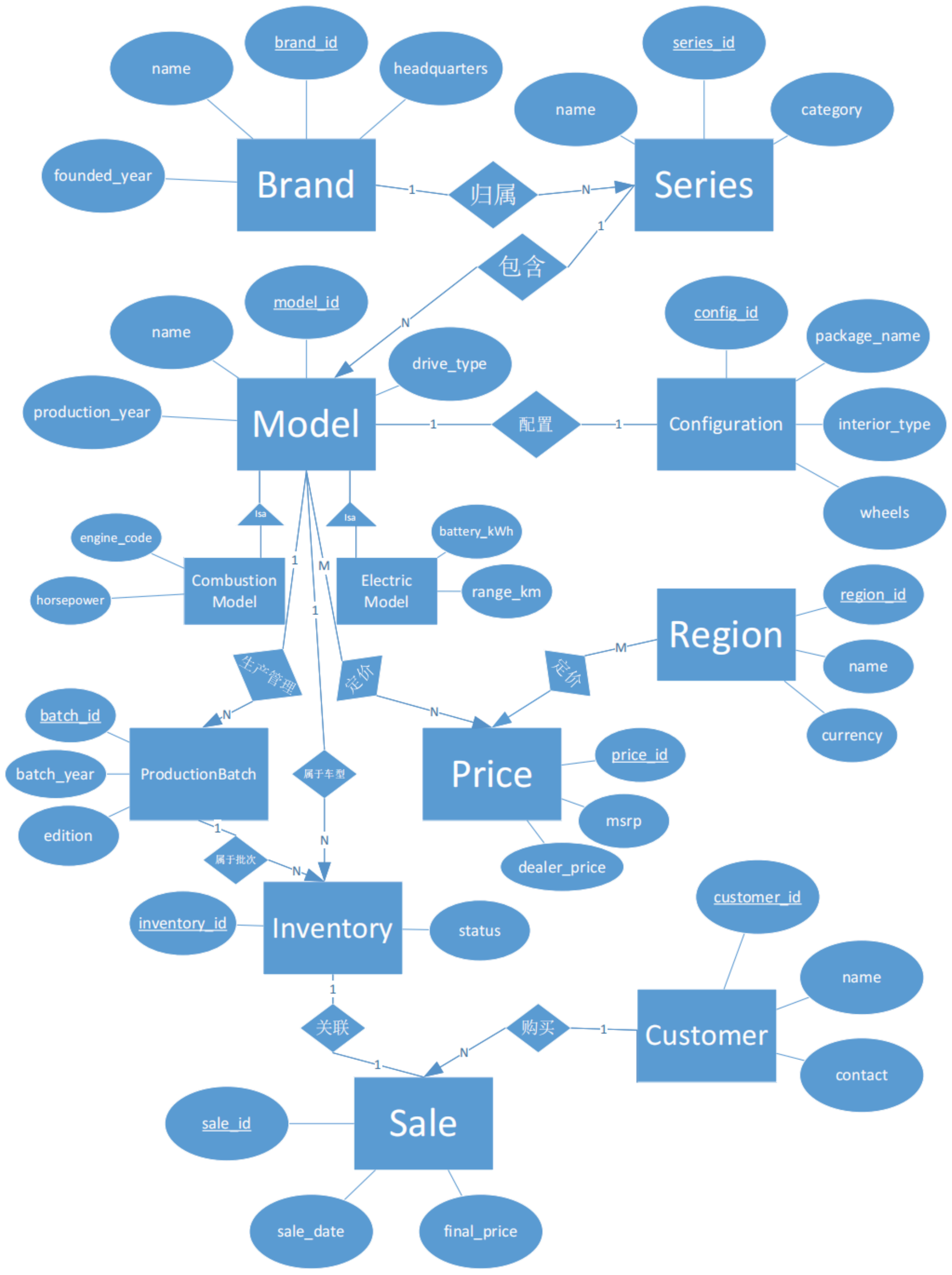
- **品牌管理**：记录宝马品牌历史、总部地址、全球市场分布。
- **车型管理**：
    - **车系分类**：按系列划分（如3系、5系、X系列、i系列电动车）。
    - **子车型扩展**：同一车系下的细分型号（如320i、330e插电混动、X5 xDrive40i）。
- **性能数据**：发动机参数（型号、马力、扭矩）、驱动方式（后驱、四驱）、加速性能、续航（电动车专属）。
- **售价管理**：记录不同地区（国家/地区）、不同配置的官方指导价及经销商报价。
- **生产批次**：管理车型的生产年份、改款版本及特殊限量版信息。
- **销售与库存**：跟踪车辆库存状态（在库、已售、在途）、销售记录及客户信息。

功能需求：

- 记录品牌与车型的从属关系。
- 管理不同车型的详细配置参数。
- 追踪客户购买记录及销售数据。
- 统计品牌销量和车型市场表现。

2.数据库设计

a) 概念模型ER图



- **实体及主键**：
  - **实体及主键**：
    - **Brand**（`brand_id`，此处存储宝马品牌信息）
    - **Series**（`series_id`，如3系、X5等）
    - **Model**（`model_id`，子类：`Combustion`/`Electric`，外键：`series_id`）
    - **Configuration**（`config_id`，外键：`model_id`）

        - **Price**（`price_id`，外键：`model_id`，`region_id`）
        - **ProductionBatch**（`batch_id`，外键：`model_id`）
        - **Region**（`region_id`，存储销售地区如中国、美国）
        - **Inventory**（`inventory_id`，外键：`model_id`，`batch_id`）
        - **Sale**（`sale_id`，外键：`inventory_id`，`customer_id`）
        - **Customer**（`customer_id`）

b) ER图转关系模式

1. **Brand**（`brand_id` PK, name, headquarters, founded_year）
    - 示例数据：`(1, 'BMW', 'Munich, Germany', 1916)`
2. **Series**（`series_id` PK, name, category ENUM('Sedan', 'SUV', 'Coupe', 'Electric')）
    - 示例数据：`(1, '3 Series', 'Sedan'), (2, 'iX', 'Electric')`
3. **Model**（`model_id` PK, series_id FK, name, production_year, drive_type ENUM('RWD', 'AWD')）
    - 子类表：
        - **CombustionModel**（`model_id` PK/FK, engine_code, horsepower, fuel_type）
        - **ElectricModel**（`model_id` PK/FK, battery_kWh, range_km）
4. **Configuration**（`config_id` PK, model_id FK, package_name, interior_type, wheels）
    - 示例：`(1, 1, 'M Sport Package', 'Leather', '19-inch')`
5. **Region**（`region_id` PK, name, currency）
    - 示例：`(1, 'China', 'CNY'), (2, 'USA', 'USD')`
6. **Price**（`price_id` PK, model_id FK, region_id FK, msrp DECIMAL(12,2), dealer_price DECIMAL(12,2)）
7. **ProductionBatch**（`batch_id` PK, model_id FK, batch_year, edition ENUM('Standard', 'Limited')）
8. **Inventory**（`inventory_id` PK, model_id FK, batch_id FK, status ENUM('In Stock', 'Sold', 'In Transit')）
9. **Customer**（`customer_id` PK, name, contact, region_id FK）
10. **Sale**（`sale_id` PK, inventory_id FK, customer_id FK, sale_date, final_price DECIMAL(12,2)）

c) 实体及联系说明

1. **Brand（品牌）**
    - **主键**：`brand_id`
    - **属性**：品牌名称（`name`）、总部地址（`headquarters`）、创立年份（`founded_year`）。
    - **联系**：
        - **归属（1:N）**：一个品牌对应多个车系（`series`）。

            - **示例**：宝马品牌（`brand_id=1`）包含3系、X5等车系。

2. **Series（车系）**
    - **主键**：`series_id`
    - **属性**：车系名称（`name`）、类别（`category`，如Sedan、SUV）。
    - **联系**：
        - **包含（1:N）**：一个车系包含多个车型（`Model`）。
            - **示例**：3系车系（`series_id=1`）包含320i、330e等车型。
3. **Model（车型）**
    - **主键**：`model_id`
    - **属性**：车型名称（`name`）、生产年份（`production_year`）、驱动类型（`drive_type`）。
    - **子类**：
        - **CombustionModel（燃油车型）**：继承自`Model`，扩展属性包括发动机型号（`engine_code`）、马力（`horsepower`）。
        - **ElectricModel（电动车型）**：继承自`Model`，扩展属性包括电池容量（`battery_kwh`）、续航里程（`range_km`）。
    - **联系**：
        - **配置（1:1）**：每个车型对应唯一配置（`Configuration`），如“M Sport Package”。
        - **定价（M:N）**：车型在不同地区（`Region`）有不同定价（`Price`）。
        - **生产批次（1:N）**：车型可关联多个生产批次（`ProductionBatch`），如2023年标准版、2024年限量版。
4. **Configuration（配置）**
    - **主键**：`config_id`
    - **属性**：配置包名称（`package_name`）、内饰类型（`interior_type`）、轮毂类型（`wheels`）。
    - **联系**：
        - **一对一关联**：与`Model`表通过`model_id`外键绑定。
5. **Region（地区）**
    - **主键**：`region_id`
    - **属性**：地区名称（`name`）、货币类型（`currency`）。
    - **联系**：
        - **定价（1:N）**：每个地区关联多个车型的定价（`Price`）。
        - **客户归属（N:1）**：客户（`Customer`）属于某个地区。
6. **Price（价格）**
    - **主键**：`price_id`
    - **属性**：厂商建议零售价（`msrp`）、经销商报价（`dealer_price`）。
    - **联系**：

        - **多对多关联**：通过`model_id`和`region_id`外键，连接`Model`和`Region`。

7. **ProductionBatch（生产批次）**
    - **主键**：`batch_id`
    - **属性**：生产年份（`batch_year`）、版本（`edition`，如标准版、限量版）。
    - **联系**：
        - **生产管理（1:N）**：每个车型可对应多个生产批次。
8. **Inventory（库存）**
    - **主键**：`inventory_id`
    - **属性**：库存状态（`status`，如在库、已售）。
    - **联系**：
        - **库存关联（N:1）**：通过`model_id`和`batch_id`外键，连接`Model`和`ProductionBatch`。
9. **Customer（客户）**
    - **主键**：`customer_id`
    - **属性**：客户姓名（`name`）、联系方式（`contact`）。
    - **联系**：
        - **购买记录（1:N）**：一个客户可进行多次购买（`Sale`）。
10. **Sale（销售记录）**
    - **主键**：`sale_id`
    - **属性**：销售日期（`sale_date`）、成交价（`final_price`）。
    - **联系**：
        - **一对一关联**：通过`inventory_id`外键绑定唯一库存项（`Inventory`）。

d) SQL建表语句

```
-- 品牌表（仅宝马）
CREATE TABLE Brand (
    brand_id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(50) NOT NULL DEFAULT 'BMW',
    headquarters VARCHAR(100),
    founded_year YEAR
);

-- 车系表
CREATE TABLE Series (
    series_id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(50) NOT NULL,
    category ENUM('Sedan', 'SUV', 'Coupe', 'Electric')
);

-- 车型表
CREATE TABLE Model (
    model_id INT PRIMARY KEY AUTO_INCREMENT,
    series_id INT,
```

```sql
    name VARCHAR(50) NOT NULL,  -- 如 "330i M Sport"
    production_year YEAR,
    drive_type ENUM('RWD', 'AWD'),
    FOREIGN KEY (series_id) REFERENCES Series(series_id) ON DELETE CASCADE
);

-- 燃油车型扩展表
CREATE TABLE CombustionModel (
    model_id INT PRIMARY KEY,
    engine_code VARCHAR(20),      -- 如 "B48B20"
    horsepower INT,
    fuel_type ENUM('Petrol', 'Diesel'),
    FOREIGN KEY (model_id) REFERENCES Model(model_id) ON DELETE CASCADE
);

-- 电动车型扩展表
CREATE TABLE ElectricModel (
    model_id INT PRIMARY KEY,
    battery_kWh DECIMAL(5,1),     -- 如 80.5
    range_km INT,
    FOREIGN KEY (model_id) REFERENCES Model(model_id) ON DELETE CASCADE
);

-- 配置表
CREATE TABLE Configuration (
    config_id INT PRIMARY KEY AUTO_INCREMENT,
    model_id INT UNIQUE,          -- 一对一关系
    package_name VARCHAR(50),
    interior_type VARCHAR(30),
    wheels VARCHAR(20),
    FOREIGN KEY (model_id) REFERENCES Model(model_id) ON DELETE CASCADE
);

-- 地区表
CREATE TABLE Region (
    region_id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(50) UNIQUE,     -- 如 "China"
    currency CHAR(3)              -- 如 "CNY"
);

-- 价格表（支持多地区定价）
CREATE TABLE Price (
    price_id INT PRIMARY KEY AUTO_INCREMENT,
    model_id INT,
    region_id INT,
    msrp DECIMAL(12,2),          -- 厂商建议零售价
    dealer_price DECIMAL(12,2),  -- 经销商报价
    FOREIGN KEY (model_id) REFERENCES Model(model_id) ON DELETE CASCADE,
    FOREIGN KEY (region_id) REFERENCES Region(region_id),
    UNIQUE (model_id, region_id) -- 防止重复定价
);

-- 生产批次表
CREATE TABLE ProductionBatch (
    batch_id INT PRIMARY KEY AUTO_INCREMENT,
    model_id INT,
    batch_year YEAR,
    edition ENUM('Standard', 'Limited'),
    FOREIGN KEY (model_id) REFERENCES Model(model_id) ON DELETE CASCADE
);

-- 库存表
CREATE TABLE Inventory (
    inventory_id INT PRIMARY KEY AUTO_INCREMENT,
    model_id INT,
    batch_id INT,
    status ENUM('In Stock', 'Sold', 'In Transit'),
    FOREIGN KEY (model_id) REFERENCES Model(model_id) ON DELETE SET NULL,
    FOREIGN KEY (batch_id) REFERENCES ProductionBatch(batch_id) ON DELETE SET
NULL
);

-- 客户表
CREATE TABLE Customer (
    customer_id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(50) NOT NULL,
    contact VARCHAR(100),
    region_id INT,
    FOREIGN KEY (region_id) REFERENCES Region(region_id)
);

-- 销售记录表
CREATE TABLE Sale (
    sale_id INT PRIMARY KEY AUTO_INCREMENT,
    inventory_id INT UNIQUE,    -- 确保每辆车只售一次
    customer_id INT,
    sale_date DATE DEFAULT (CURRENT_DATE),
    final_price DECIMAL(12,2) CHECK (final_price > 0),
    FOREIGN KEY (inventory_id) REFERENCES Inventory(inventory_id),
    FOREIGN KEY (customer_id) REFERENCES Customer(customer_id) ON DELETE SET NULL
);
```

e) 查询示例

单表查询：查询所有四驱车型的名称及生产年份。

```
SELECT name, production_year FROM Model WHERE drive_type = 'AWD';
```

多表连接查询：查询中国地区宝马X5的厂商建议零售价。

```
SELECT Model.name, Price.msrp
FROM Price
JOIN Model ON Price.model_id = Model.model_id
JOIN Region ON Price.region_id = Region.region_id
WHERE Region.name = 'China' AND Model.name LIKE '%X5%';
```

嵌套查询：查询马力超过300匹的燃油车型。

```sql
SELECT m.name, c.horsepower
FROM Model m
JOIN CombustionModel c ON m.model_id = c.model_id
WHERE c.horsepower > 300;
```

EXISTS查询：查询有限量版生产的车系。

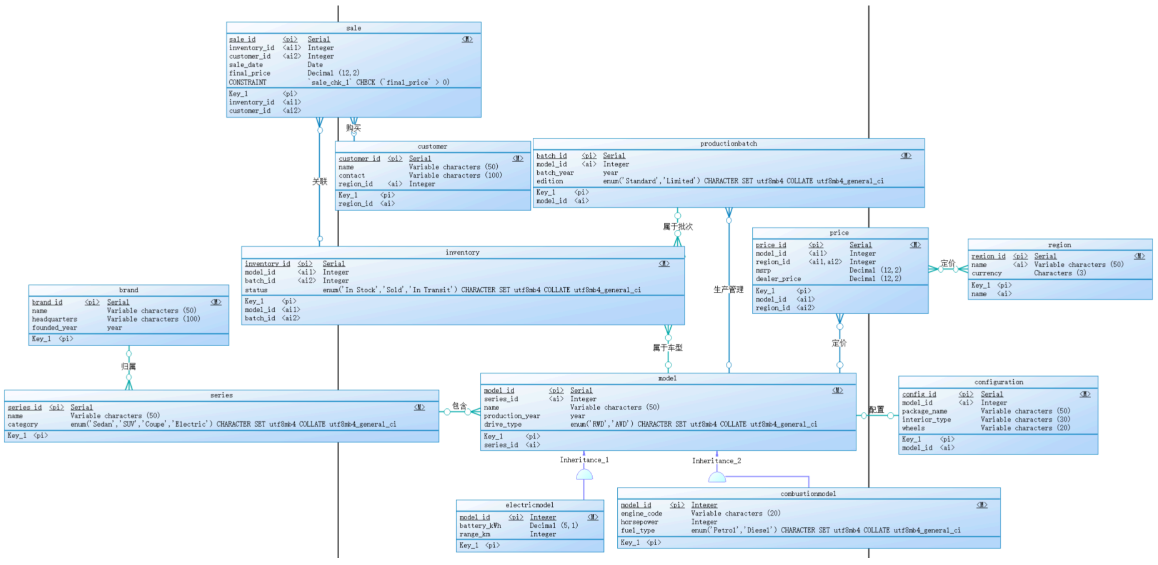
```sql
SELECT s.name
FROM Series s
WHERE EXISTS (
    SELECT 1 FROM ProductionBatch pb
    JOIN Model m ON pb.model_id = m.model_id
    WHERE m.series_id = s.series_id AND pb.edition = 'Limited'
);
```

聚合查询：统计各车系在中国的平均售价。

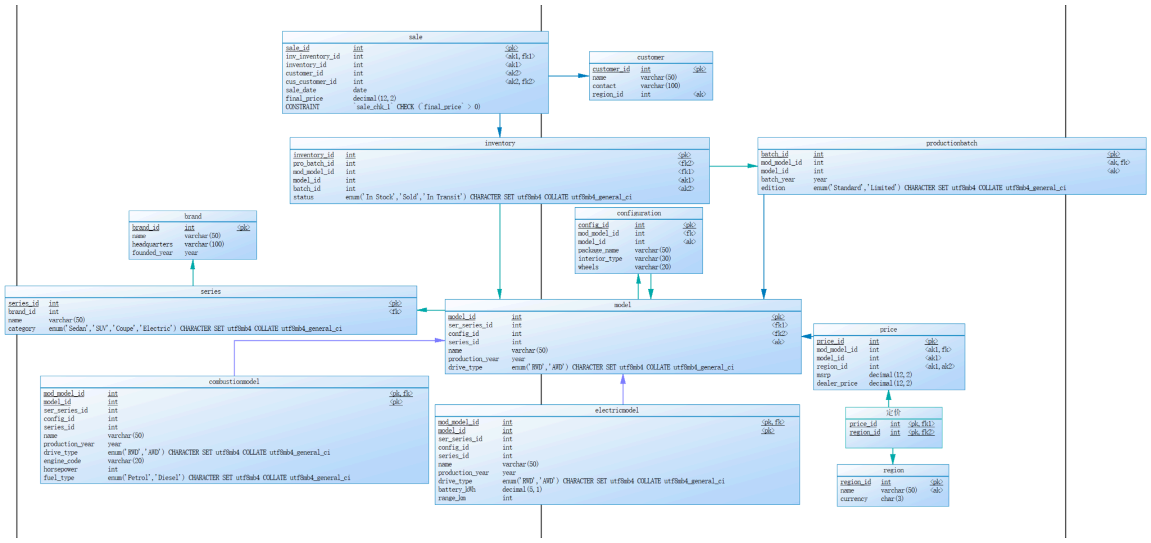
```sql
SELECT s.name, AVG(p.msrp) AS avg_msrp
FROM Series s
JOIN Model m ON s.series_id = m.series_id
JOIN Price p ON m.model_id = p.model_id
JOIN Region r ON p.region_id = r.region_id
WHERE r.name = 'China'
GROUP BY s.series_id;
```

3. PowerDesigner设计

a) 概念模型ER图截图


sale
sale_id <pi> Serial <M>
inventory_id <ai1> Integer
customer_id <ai2> Integer
sale_date Date
final_price Decimal (12,2)
CONSTRAINT `sale_chk_1` CHECK (`final_price` > 0)
Key_1 <pi>
inventory_id <ai1>
customer_id <ai2>
购买
关联
customer
customer_id <pi> Serial <M>
name Variable characters (50)
contact Variable characters (100)
region_id <ai> Integer
Key_1 <pi>
region_id <ai>
productionbatch
batch_id <pi> Serial <M>
model_id <ai> Integer
batch_year year
edition enum('Standard','Limited') CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci
Key_1 <pi>
model_id <ai>
属于批次
生产管理
inventory
inventory_id <pi> Serial <M>
model_id <ai1> Integer
batch_id <ai2> Integer
status enum('In Stock','Sold','In Transit') CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci
Key_1 <pi>
model_id <ai1>
batch_id <ai2>
price
price_id <pi> Serial <M>
model_id <ai1> Integer
region_id <ai1,ai2> Integer
msrp Decimal (12,2)
dealer_price Decimal (12,2)
Key_1 <pi>
model_id <ai1>
region_id <ai2>
定价
region
region_id <pi> Serial <M>
name <ai> Variable characters (50)
currency Characters (3)
Key_1 <pi>
name <ai>
brand
brand_id <pi> Serial <M>
name Variable characters (50)
headquarters Variable characters (100)
founded_year year
Key_1 <pi>
归属
属于车型
定价
series
series_id <pi> Serial <M>
name Variable characters (50)
category enum('Sedan','SUV','Coupe','Electric') CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci
Key_1 <pi>
包含
model
model_id <pi> Serial <M>
series_id <ai> Integer
name Variable characters (50)
production_year year
drive_type enum('RWD','AWD') CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci
Key_1 <pi>
series_id <ai>
配置
configuration
config_id <pi> Serial <M>
model_id <ai> Integer
package_name Variable characters (50)
interior_type Variable characters (30)
wheels Variable characters (20)
Key_1 <pi>
model_id <ai>
Inheritance_1
Inheritance_2
electricmodel
model_id <pi> Integer <M>
battery_kWh Decimal (5,1)
range_km Integer
Key_1 <pi>
combustionmodel
model_id <pi> Integer <M>
engine_code Variable characters (20)
horsepower Integer
fuel_type enum('Petrol','Diesel') CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci
Key_1 <pi>

b) 关系模型图截图



c) 生成SQL语句

```sql
/*==============================================================*/
/* DBMS name:      MySQL 5.0                                    */
/* Created on:     2025/4/6 20:19:07                            */
/*==============================================================*/


drop table if exists brand;

drop table if exists combustionmodel;

drop table if exists configuration;

drop table if exists customer;

drop table if exists double_belong;

drop table if exists electricmodel;

drop table if exists inventory;

drop table if exists model;

drop table if exists price;

drop table if exists productionbatch;

drop table if exists region;

drop table if exists sale;

drop table if exists series;

/*==============================================================*/
/* Table: brand                                                 */
/*==============================================================*/
create table brand
```

```sql
(
   brand_id             int not null auto_increment,
   name                 varchar(50),
   headquarters         varchar(100),
   founded_year         year default NULL,
   primary key (brand_id)
)
ENGINE = InnoDB CHARACTER SET = utf8mb4 COLLATE = utf8mb4_general_ci ROW_FORMAT =
Dynamic;

/*==============================================================*/
/* Table: combustionmodel                                       */
/*==============================================================*/
create table combustionmodel
(
   mod_model_id         int not null,
   model_id             int not null,
   ser_series_id        int,
   config_id            int,
   series_id            int default NULL,
   name                 varchar(50),
   production_year      year default NULL,
   drive_type           enum('RWD','AWD') CHARACTER SET utf8mb4 COLLATE
utf8mb4_general_ci default NULL,
   engine_code          varchar(20),
   horsepower           int default NULL,
   fuel_type            enum('Petrol','Diesel') CHARACTER SET utf8mb4 COLLATE
utf8mb4_general_ci default NULL,
   primary key (mod_model_id, model_id)
)
ENGINE = InnoDB CHARACTER SET = utf8mb4 COLLATE = utf8mb4_general_ci ROW_FORMAT =
Dynamic;

/*==============================================================*/
/* Table: configuration                                         */
/*==============================================================*/
create table configuration
(
   config_id            int not null auto_increment,
   mod_model_id         int,
   model_id             int default NULL,
   package_name         varchar(50),
   interior_type        varchar(30),
   wheels               varchar(20),
   primary key (config_id),
   unique key model_id (model_id)
)
ENGINE = InnoDB CHARACTER SET = utf8mb4 COLLATE = utf8mb4_general_ci ROW_FORMAT =
Dynamic;

/*==============================================================*/
/* Table: customer                                              */
/*==============================================================*/
create table customer
(
   customer_id          int not null auto_increment,
   name                 varchar(50),
   contact              varchar(100),
   region_id            int default NULL,
   primary key (customer_id),
   key region_id (region_id)
)
ENGINE = InnoDB CHARACTER SET = utf8mb4 COLLATE = utf8mb4_general_ci ROW_FORMAT =
Dynamic;

/*==============================================================*/
/* Table: double_belong                                         */
/*==============================================================*/
create table double_belong
(
   price_id             int not null,
   region_id            int not null,
   primary key (price_id, region_id)
);

/*==============================================================*/
/* Table: electricmodel                                         */
/*==============================================================*/
create table electricmodel
(
   mod_model_id         int not null,
   model_id             int not null,
   ser_series_id        int,
   config_id            int,
   series_id            int default NULL,
   name                 varchar(50),
   production_year      year default NULL,
   drive_type           enum('RWD','AWD') CHARACTER SET utf8mb4 COLLATE
utf8mb4_general_ci default NULL,
   battery_kwh          decimal(5,1) default NULL,
   range_km             int default NULL,
   primary key (mod_model_id, model_id)
)
ENGINE = InnoDB CHARACTER SET = utf8mb4 COLLATE = utf8mb4_general_ci ROW_FORMAT =
Dynamic;

/*==============================================================*/
/* Table: inventory                                             */
/*==============================================================*/
create table inventory
(
   inventory_id         int not null auto_increment,
   pro_batch_id         int,
   mod_model_id         int,
   model_id             int default NULL,
   batch_id             int default NULL,
   status               enum('In Stock','Sold','In Transit') CHARACTER SET
utf8mb4 COLLATE utf8mb4_general_ci default NULL,
   primary key (inventory_id),
   key model_id (model_id),
   key batch_id (batch_id)
)
```

```sql
ENGINE = InnoDB CHARACTER SET = utf8mb4 COLLATE = utf8mb4_general_ci ROW_FORMAT =
Dynamic;

/*==============================================================*/
/* Table: model                                                 */
/*==============================================================*/
create table model
(
   model_id             int not null auto_increment,
   ser_series_id        int,
   config_id            int,
   series_id            int default NULL,
   name                 varchar(50),
   production_year      year default NULL,
   drive_type           enum('RWD','AWD') CHARACTER SET utf8mb4 COLLATE
utf8mb4_general_ci default NULL,
   primary key (model_id),
   key series_id (series_id)
)
ENGINE = InnoDB CHARACTER SET = utf8mb4 COLLATE = utf8mb4_general_ci ROW_FORMAT =
Dynamic;

/*==============================================================*/
/* Table: price                                                 */
/*==============================================================*/
create table price
(
   price_id             int not null auto_increment,
   mod_model_id         int,
   model_id             int default NULL,
   region_id            int default NULL,
   msrp                 decimal(12,2) default NULL,
   dealer_price         decimal(12,2) default NULL,
   primary key (price_id),
   unique key model_id (model_id, mod_model_id, region_id),
   key region_id (region_id)
)
ENGINE = InnoDB CHARACTER SET = utf8mb4 COLLATE = utf8mb4_general_ci ROW_FORMAT =
Dynamic;

/*==============================================================*/
/* Table: productionbatch                                       */
/*==============================================================*/
create table productionbatch
(
   batch_id             int not null auto_increment,
   mod_model_id         int,
   model_id             int default NULL,
   batch_year           year default NULL,
   edition              enum('Standard','Limited') CHARACTER SET utf8mb4 COLLATE
utf8mb4_general_ci default NULL,
   primary key (batch_id),
   key model_id (model_id, mod_model_id)
)
ENGINE = InnoDB CHARACTER SET = utf8mb4 COLLATE = utf8mb4_general_ci ROW_FORMAT =
Dynamic;
```

```sql
/*==============================================================*/
/* Table: region                                                */
/*==============================================================*/
create table region
(
   region_id            int not null auto_increment,
   name                 varchar(50),
   currency             char(3),
   primary key (region_id),
   unique key name (name)
)
ENGINE = InnoDB CHARACTER SET = utf8mb4 COLLATE = utf8mb4_general_ci ROW_FORMAT =
Dynamic;

/*==============================================================*/
/* Table: sale                                                  */
/*==============================================================*/
create table sale
(
   sale_id              int not null auto_increment,
   inv_inventory_id     int,
   inventory_id         int default NULL,
   customer_id          int default NULL,
   cus_customer_id      int,
   sale_date            date default curdate(),
   final_price          decimal(12,2) default NULL,
   CONSTRAINT           `sale_chk_1` CHECK (`final_price` > 0),
   primary key (sale_id),
   unique key inventory_id (inventory_id, inv_inventory_id),
   key customer_id (customer_id, cus_customer_id)
)
ENGINE = InnoDB CHARACTER SET = utf8mb4 COLLATE = utf8mb4_general_ci ROW_FORMAT =
Dynamic;

/*==============================================================*/
/* Table: series                                                */
/*==============================================================*/
create table series
(
   series_id            int not null auto_increment,
   brand_id             int,
   name                 varchar(50),
   category             enum('Sedan','SUV','Coupe','Electric') CHARACTER SET
utf8mb4 COLLATE utf8mb4_general_ci default NULL,
   primary key (series_id)
)
ENGINE = InnoDB CHARACTER SET = utf8mb4 COLLATE = utf8mb4_general_ci ROW_FORMAT =
Dynamic;

alter table combustionmodel add constraint FK_Inheritance_2 foreign key
(mod_model_id)
      references model (model_id) on delete restrict on update restrict;

alter table configuration add constraint FK_config foreign key (mod_model_id)
      references model (model_id) on delete restrict on update restrict;
```

```
alter table double_belong add constraint FK_double_belong foreign key (price_id)
      references price (price_id) on delete restrict on update restrict;

alter table double_belong add constraint FK_double_belong foreign key (region_id)
      references region (region_id) on delete restrict on update restrict;

alter table electricmodel add constraint FK_Inheritance_1 foreign key
(mod_model_id)
      references model (model_id) on delete restrict on update restrict;

alter table inventory add constraint FK_belong_model foreign key (mod_model_id)
      references model (model_id) on delete restrict on update restrict;

alter table inventory add constraint FK_belong_production foreign key
(pro_batch_id)
      references productionbatch (batch_id) on delete restrict on update
restrict;

alter table model add constraint FK_belong_series foreign key (ser_series_id)
      references series (series_id) on delete restrict on update restrict;

alter table model add constraint FK_config foreign key (config_id)
      references configuration (config_id) on delete restrict on update restrict;

alter table price add constraint price_ibfk_1 foreign key (mod_model_id)
      references model (model_id) on delete restrict on update restrict;

alter table productionbatch add constraint productionbatch_ibfk_1 foreign key
(mod_model_id)
      references model (model_id) on delete restrict on update restrict;
alter table sale add constraint sale_ibfk_2 foreign key (cus_customer_id)
      references customer (customer_id) on delete restrict on update restrict;

alter table sale add constraint sale_ibfk_1 foreign key (inv_inventory_id)
      references inventory (inventory_id) on delete restrict on update restrict;

alter table series add constraint FK_belong_brand foreign key (brand_id)
      references brand (brand_id) on delete restrict on update restrict;
```

4. 分析比较

a) 关系模式差异

- 两种设计方法存在显著差异，主要体现在以下方面：
    - **效率与精确度**：手动设计要耗费大量时间和精力，且易出错。而使用像 PowerDesigner 这样的建模工具，能提升设计效率，减少人为失误，还可借助工具自动验证模型的完整性和准确性。
    - **可视化与交互性**：建模工具具备直观的图形界面，能通过拖拽和连接操作快速创建与编辑模型。手动设计往往需绘制图表或草图，不够直观，交互编辑也较困难。

    - **模型管理与版本控制**：建模工具提供了模型管理和版本控制功能，方便保存、分享和更新模型，还能跟踪模型的变更历史。手动设计通常缺乏这些功能，导致模型管理和维护困难。

  这些差异会对后期实现产生不同影响。使用建模工具设计，能更快速、准确地创建模型，借助工具自动验证和管理模型，减少错误，提高效率。而手动设计可能耗时更久、精力投入更多，易出错，且模型管理和维护不便。

b) PowerDesigner SQL特点

- PowerDesigner 生成的 SQL 语句具有以下特点：
    - **标准化**：生成的 SQL 语句一般符合 SQL 标准，可在多数关系数据库管理系统（RDBMS）中执行。
    - **可读性高**：语句结构清晰，便于理解和阅读。
    - **模板化**：依据模型中定义的实体、关系和约束等元素，生成对应的 SQL 语句模板，并填充具体信息。
    - **包含附加语句**：除了创建表、索引等基本操作，还可能包含一些附加语句。

  出现附加语句的原因在于要增强数据完整性和系统的健壮性。例如，在设计外键时会有很多相关设计，这些附加语句能确保数据在数据库中的一致性和有效性。附加语句能提供额外功能和灵活性，可根据具体需求进行调整和定制，让生成的 SQL 脚本更契合实际需求。