数据库系统实验课程设计报告

房地产销售管理系统的设计与实现

姓名：刘佳瑜

学号：21311592

学院：计算机学院

专业：计算机科学与技术

1 引言

随着房地产市场的不断发展，房地产销售业务变得越来越复杂和庞大。为了更有效地管理和优化销售过程，我根据试验要求设计并实现了一个房地产销售管理数据库系统。该系统旨在提供一个完整的、集成的解决方案，以满足房地产公司和销售团队的需求，从而实现更高效、透明和可追踪的销售活动。

1.1 设计目的

本房地产销售管理系统主要的设计目的如下所示：

- 提高销售效率：通过整合客户信息、房源信息和销售活动，系统旨在提高销售团队的工作效率，使得销售人员可以更轻松地访问、更新和跟踪客户信息，以便更有针对性地进行销售活动；
- 优化销售过程：房地产销售涉及到众多环节，包括客户接触、信息收集、房源匹配、成交管理等，本系统旨在优化这些过程，使其更加协调和无缝，从而提高销售效果；
- 增强客户体验：通过提供客户信息的全面性和及时性，系统将有助于提升客户体验，客户可以更方便地获取房源信息、了解交易进展，并与销售团队进行更即时的沟通。

1.2 设计要求

- 数据集成性：系统需要能够集成和管理多样化的数据，包括客户信息、房源数据、销售业绩等，以确保数据的一致性和准确性；
- 用户友好性：系统应具备友好的用户界面，使销售人员能在不同的设备上轻松地进行操作和查询；
- 安全性：系统需要采取适当的安全措施，确保敏感信息的保密性和完整性；
- 可扩展性：考虑到房地产市场的不断变化，系统应具备良好的可扩展性，以便未来能够灵活适应业务的发展和变化。

1.3 设计环境

- 数据库系统：PostgreSQL 15；
- 操作系统：Windows x64；
- 开发平台：JavaScript；

2 概要设计

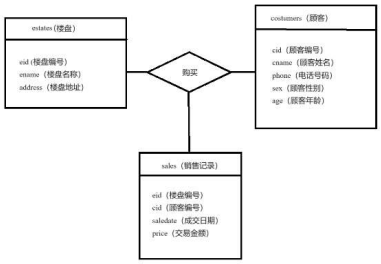
2.1 系统需求分析

根据常见的房地产销售管理需求，针对房地产销售管理系统的需求分析如下：

- 楼盘信息管理：楼盘信息的添加、修改和查询；
- 顾客信息管理：顾客信息的添加、修改和查询；
- 销售信息管理：销售数据的录入、查询、统计；

2.2 E-R 图

根据上述系统需求分析，可绘制出由 E-R 图表示的房地产销售管理系统概念模型：



3 详细设计

3.1 关系模式设计

首先将概念模型转化为满足 3NF 的关系模式。
创建记录楼盘信息数据的关系 estates，包含 eid（楼盘编号）、ename（楼盘名称）和 address（楼盘地址）三个属性，以 eid 作为关系的主键，同时为符合实际情况，对 ename 和 address 进行唯一性约束（unique）：

```
create table estates(
  eid char(6) primary key,
  ename varchar(15) unique,
```

```
    address varchar(15) unique);
```

创建记录顾客信息数据的关系 costumers，包含 cid（客户编号）、cname（客户姓名）、phone（客户电话号码）、sex（客户性别）和 age（客户年龄）六个属性，以 cid 作为关系的主键，同时为了符合实际情况，对客户的手机号码进行唯一性约束，只允许性别为‘女’或‘男’，并只允许年满十八周岁的公民购买房产：

```
create table costumers(
  cid char(6) primary key,
  cname varchar(4),
  phone char(11) unique,
  sex char(1) check (sex in ('女','男')),
  age int check ( age > 18));
```

创建记录销售信息数据的关系 sales，包含 eid（楼盘编号）、cid（客户编号）、saledate（成交日期）和 price（成交价格），因为 eid 和 cid 为另外两个关系的主键，所以这两个属性为 sales 关系的外键：

```
create table sales(
  eid char(6),
  cid char(6),
  saledate date,
  price decimal(10,2),
  foreign key (eid) references estates(eid),
  foreign key (cid) references costumers(cid));
```

另外，为了符合实际，防止添加不存在的楼盘或顾客的购买记录，添加两个触发器（trigger），每次添加销售记录之前检查对应的 eid 及 cid 是否存在，代码如下：

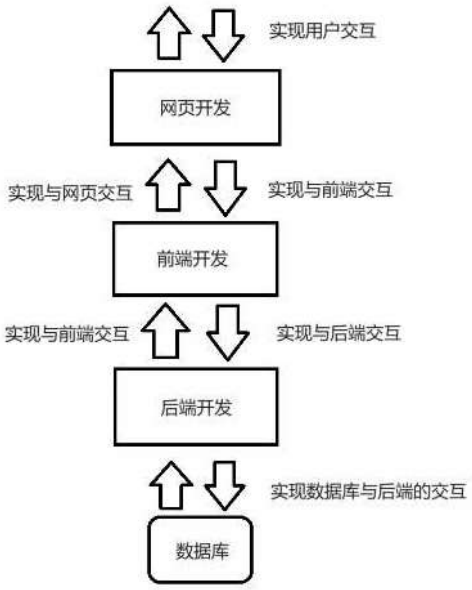
```
create or replace function se() returns trigger as $$
begin
    if not exists (select 1 from estates where eid = NEW.eid)
         then
        raise exception '请勿记录不存在的楼盘，请先建设新的楼
            盘';
    end if;
    RETURN NEW;
end;
$$ language plpgsql;

create trigger se_trigger
before insert on sales
```

```
for each row
execute function se();


create or replace function sc() returns trigger as $$
begin
    if not exists (select 1 from costumers where cid = NEW.
        cid) then
        raise exception '请勿记录不存在的客户，请先建立新的客
            户档案';
    end if;
    RETURN NEW;
end;
$$ language plpgsql;

create trigger sc_trigger
before insert on sales
for each row
execute function sc();
```

3.2 用户友好的数据库管理系统开发

本项目采用 JavaScript 房地产销售管理系统页面的前后端开发，整体开发结构层次如下所示：具体的项目架构如下所示，下面将从上至下逐一讲解项目每个主要部分的实现：

```
estatesystem/
|-- frontend/
|   |-- css/
|   |   |-- styles.css
|   |-- js/
|   |   |-- app.js (前端实现)
|   |-- index.html (网页实现)
|-- backend
|-- |-- server.js (后端实现)
|-- node_modules
|-- package-lock.json
|-- package.json
|-- README.md
```



3.2.1 用户友好的网页界面实现

在 index.html 完成用户友好的网页界面的编写，实现思路如下：

- 网页头条显示当前交易成交总额度和总次数
- 销售记录管理
  - 按钮录入：点击后弹出五个表项，使用户填空，填完后点击下方的“确定”按钮，数据保存到数据库，网页恢复原状
  - 按钮查询：点击后显示所有的销售记录表项
- 楼盘信息管理
  - 按钮添加：点击后弹出楼盘表项，使用户填空，填完后点击下方的“确定”按钮，数据保存到数据库，网页恢复原状
  - 按钮修改：点击后弹出一个表项，请用户输入要修改的楼盘编号，输入后弹出对应的三个表项，用户修改然后点击保存
  - 按钮查询：点击后显示所有的楼盘记录表项
- 客户信息管理
  - 按钮添加：点击后弹出客户表项，使用户填空，填完后点击下方的“确定”按钮，数据保存到数据库，网页恢复原状

- 按钮修改：点击后弹出一个表项，请用户输入要修改的顾客编号，输入后弹出对应的五个表项，用户修改然后点击保存
- 按钮查询：点击后显示所有的顾客记录表项

实现主要代码如下所示：

```
div class="section"
    !-- 标题 --
    h1 style="font-weight: bold; font-size: 35px; line-height:
        1.5;"房地产销售管理系统/h1
      !-- 统计 --
      div class="sales-summary"
        p总销售额： span id="totalSales"/span 元； 成交量： span
            id="totalTransactions"/span 笔/p
    /div

    !-- 销售数据管理 --
    div class="section"
        h2销售记录管理/h2
        div class="horizontal-buttons"
            button type="button"
                onclick="showSalesForm()"录入/button
            button type="button"
                onclick="showSalesTable()"查询/button
        /div

        !-- 销售录入 --
        div id="salesForm" class="form" style="display: none;"
            label for="eid"楼盘编号:/label
            input type="text" id="eid" name="eid" required
            label for="cid"客户编号:/label
            input type="text" id="cid" name="cid" required
            label for="saledate"销售日期:/label
            input type="date" id="saledate" name="saledate" required
            label for="price"销售价格:/label
            input type="number" id="price" name="price" required
            div class="button-container"
                button type="button" id="recordSaleBtn"
                    onclick="saveSalesRecord()"确定/button
                button type="button" id="recordSaleBtn"
                    onclick="hideSalesForm()"取消/button
```

```
        /div
    /div
    !-- 销售查询 --
    div id="salesTableContainer" /div
/div


!-- 楼盘数据管理 --
div class="section"
    h2楼盘信息管理/h2
    div class="horizontal-buttons"
        button type="button" onclick="showEstatesForm()(' 添
            加')"添加/button
        button type="button" onclick="showModifyEstateForm(' 修
            改')"修改/button
        button type="button"
            onclick="showEstatesTable()"查询/button
    /div

    !-- 楼盘添加 --
    div id="estatesForm" class="form" style="display: none;"
        label for="eeid"楼盘编号:/label
        input type="text" id="eeid" name="eeid" required
        label for="ename"楼盘名称:/label
        input type="text" id="ename" name="ename" required
        label for="address"楼盘地址:/label
        input type="text" id="address" name="address" required
        div class="button-container"
            button type="button" id="recordSaleBtn"
                onclick="saveEsatesRecord()"确定/button
            button type="button" id="recordSaleBtn"
                onclick="hideEstatesForm()"取消/button
        /div
    /div
    !-- 楼盘修改 --
    div id="modifyEstateForm" class="form" style="display: none;"
        label for="modifyEid"请输入要修改的楼盘编号:/label
        input type="text" id="modifyEid" name="modifyEid"
            required
        div class="button-container"
```

```
            button type="button" id="recordSaleBtn"
                onclick="confirmModifyEstate()"确认/button
            button type="button" id="recordSaleBtn"
                onclick="cancelModifyEstate()"取消/button
        /div
    /div
    !-- 楼盘查询 --
    div id="estatesTableContainer" /div
/div


!-- 客户数据管理 --
div class="section"
    h2客户信息管理/h2
    div class="horizontal-buttons"
        button type="button" onclick="showCostumersForm(' 添
            加')"添加/button
        button type="button" onclick="showModifyCostumersForm('
            修改')"修改/button
        button type="button"
            onclick="showCostumersTable()"查询/button
    /div

    !-- 顾客添加 --
    div id="costumersForm" class="form" style="display: none;"
        label for="ccid"顾客编号:/label
        input type="text" id="ccid" name="ccid" required
        label for="cname"顾客姓名:/label
        input type="text" id="cname" name="cname" required
        label for="phone"手机号码:/label
        input type="text" id="phone" name="phone" required
        label for="sex"顾客性别:/label
        input type="text" id="sex" name="sex" required
        label for="age"顾客年龄:/label
        input type="text" id="age" name="age" required
        div class="button-container"
            button type="button" id="recordSaleBtn"
                onclick="saveCostumersRecord()"确定/button
            button type="button" id="recordSaleBtn"
                onclick="hideCostumersForm()"取消/button
```

```
            /div
        /div
          !-- 顾客修改 --
        div id="modifyCostumersForm" class="form" style="display:
            none;"
            label for="modifyCid"请输入要修改的顾客编号:/label
            input type="text" id="modifyCid" name="modifyCid"
                required
            div class="button-container"
                button type="button" id="recordSaleBtn"
                    onclick="confirmModifyCostumers()"确认/button
                button type="button" id="recordSaleBtn"
                    onclick="cancelModifyCostumers()"取消/button
            /div
        /div
        !-- 顾客查询 --
        div id="costumersTableContainer"/div
    /div
  /div
/div
```

3.2.2 前端开发

在 app.js 中实现前端开发，向上连接网页与用户的交互，向下连接后端与数据库的交互。下面以销售部分为例讲解具体的实现过程：

```
// 销售汇总信息
async function updateSalesSummary() {
    console.log('Starting updateSalesSummary function');

    try {
        const response = await fetch('http://localhost:3000/
            api/sales/summary'); // 根据实际后端路由进行调整

        if (!response.ok) {
            throw new Error(`Failed to fetch sales summary
                data. Status: ${response.status}`);
        }

        const salesSummaryData = await response.json();
```

```
            // 更新总销售额和成交量
            const totalSalesElement = document.getElementById('
                totalSales');
            const totalTransactionsElement = document.
                getElementById('totalTransactions');
            totalSalesElement.textContent = salesSummaryData.
                totalsales;
            totalTransactionsElement.textContent =
                salesSummaryData.totaltransactions;

        } catch (error) {
            console.error('Error fetching or updating sales
                summary data:', error);
        }
}
// 在页面加载完成后调用该函数，实现统计数据的实时更新
updateSalesSummary();

// 显示销售记录录入表单（点击“录入”按钮后调用）
function showSalesForm() {
    const salesForm = document.getElementById('salesForm');
    salesForm.style.display = 'block';
}

// 隐藏销售记录录入表单（点击“取消”或“完成”按钮后调用）
function hideSalesForm() {
    const salesForm = document.getElementById('salesForm');
    salesForm.style.display = 'none';
}

// 保存销售记录到数据库
async function saveSalesRecord() {
    const eid = document.getElementById('eid').value;
    const cid = document.getElementById('cid').value;
    const saledate = document.getElementById('saledate').
        value;
    const price = document.getElementById('price').value;

    try {
```

```
            const response = await fetch('http://localhost:3000/
                api/sales', {
                method: 'POST',
                headers: {
                    'Content-Type': 'application/json',
                },
                body: JSON.stringify({ eid, cid, saledate, price
                    }),
            });

            if (response.ok) {
                // 数据保存成功，可以执行其他操作
                console.log('Record saved successfully');
                hideSalesForm(); // 隐藏表单
            } else {
                const errorMessage = await response.json();
                alert(errorMessage.error); // 在网页上弹出错误提
                    示
                console.error('Failed to save record:',
                    errorMessage);
            }
        } catch (error) {
            console.error('Error:', error);
        }
    }

    // 隐藏销售记录总表
    function hideSalesTable() {
        const tableContainer = document.getElementById('
            salesTableContainer');
        tableContainer.innerHTML = '';
    }

    // 显示销售记录总表
    async function showSalesTable() {
        try {
            const response = await fetch('http://localhost:3000/
                api/sales');

            if (!response.ok) {
```

```
            throw new Error(`Failed to fetch sales data.
                Status: ${response.status}`);
        }

        const salesData = await response.json();

        const tableContainer = document.getElementById('
            salesTableContainer');
        tableContainer.innerHTML = generateSalesTable(
            salesData);

    } catch (error) {
        console.error('Error fetching or processing sales
            data:', error);
        // 显示错误信息给用户或执行其他错误处理逻辑
    }
}

// 根据销售记录数据生成统计总表的HTML
function generateSalesTable(data) {
  const tableHTML = `
        <h2 style="font-size: smaller; color: gray;">截至目前
            时分，房地产销售总记录如下: </h2>
        <table style="border-collapse: collapse; width: 60%;
            margin: auto;">
            <thead>
                <tr style="border: 1px solid #ddd; background
                    -color: #f2f2f2;">
                    <th style="border: 1px solid #ddd;
                        padding: 5px;">楼盘编号</th>
                    <th style="border: 1px solid #ddd;
                        padding: 8px;">顾客编号</th>
                    <th style="border: 1px solid #ddd;
                        padding: 8px;">销售日期</th>
                    <th style="border: 1px solid #ddd;
                        padding: 8px;">成交价格</th>
                </tr>
            </thead>
            <tbody>
                ${data.map(record => `
```

```
                    <tr style="border: 1px solid #ddd;">
                        <td style="border: 1px solid #ddd;
                            padding: 8px;">${record.eid}</td>
                        <td style="border: 1px solid #ddd;
                            padding: 8px;">${record.cid}</td>
                        <td style="border: 1px solid #ddd;
                            padding: 8px;">${record.saledate
                            }</td>
                        <td style="border: 1px solid #ddd;
                            padding: 8px;">${record.price}</td
                            >
                    </tr>
                `).join('')}
            </tbody>
        </table>
        <div class="button" style="margin-top: 20px; text-
            align: center;">
            <button type="button" id="recordSaleBtn" onclick=
                "hideSalesTable()">返回</button>
        </div>
    </div>
    `;
    return tableHTML;
}
```

其他部分的实现方式类似。

3.3 后端开发

这里同样以销售记录的实现为例，讲解后端开发的实现。

首先建立与本地数据库的连接：

```
// 静态文件服务
app.use(express.static('frontend'));

const PORT = 3000;

// 创建一个 PostgreSQL 数据库连接池
const pool = new Pool({
    user: 'postgres',
    host: 'localhost',
```

```
    database: 'estate',
    password: '20030808',
    port: 5432,
});
```

然后实现路由的获取和查询语句:

```
// 获取销售汇总信息的路由
app.get('/api/sales/summary', async (req, res) => {
    try {
        const result = await pool.query('SELECT SUM(price) as
            totalSales, COUNT(*) as totalTransactions FROM
            sales');
        const salesSummaryData = result.rows[0];
        res.status(200).json(salesSummaryData);
    } catch (error) {
        console.error('Error fetching sales summary data:',
            error);
        res.status(500).json({ error: 'Internal Server Error'
            });
    }
});

// 处理保存销售记录的路由
app.post('/api/sales', async (req, res) => {
    const { eid, cid, saledate, price } = req.body;

    try {
        // 使用连接池执行 SQL 查询
        const result = await pool.query(
            'INSERT INTO sales (eid, cid, saledate, price)
                VALUES ($1, $2, $3, $4)',
            [eid, cid, saledate, price]
        );

        res.status(201).json({ message: 'Record saved
            successfully' });
    } catch (error) {
        console.error('Error saving record:', error);
        res.status(500).json({ error: 'Internal Server Error'
            });
```

```
    }
});

// 获取所有销售记录的路由
app.get('/api/sales', async (req, res) => {
    try {
        // 使用连接池执行 SQL 查询
        const result = await pool.query('SELECT * FROM sales'
            );
        const salesData = result.rows;

        res.status(200).json(salesData);
    } catch (error) {
        console.error('Error fetching sales data:', error);
        res.status(500).json({ error: 'Internal Server Error'
             });
    }
});
```

其他部分的实现方式类似。

4 调试与运行结果

首先在本地的 PostgreSQL 数据库中建立一个名为 estate 的数据库，依照前文所述的关系模式执行 SQL 查询语句。然后运行终端，打开/estatesystem/backend 文件夹，建立数据库连接：

```
pg_ctl start -D "C:\Program Files\PostgreSQL\15\data"
```

启动 npm，然后运行后端文件：

```
npm init
node server.js
```

若终端有如下显示则说明数据库连接成功，后端程序已开始运行：

然后在浏览器中打开 html 文件，路径为 file://estatesystem/frontend/index.html，

则现在可以开始利用用户友好的交互网页界面实现房地产销售数据系统管理。如下图所示，网页头条显示当前数据库统计数据：

点击查询按钮，显示对应的记录总表：

点击录入按钮，显示对应的条目，点击确认，数据保存到数据库：

5 课程设计小结

在进行数据库实验的课程设计过程中，我深刻体会到了数据库设计和管理的重要性，以及如何有效地利用数据库系统来支持应用程序的需求。我在这次课程设计中收获了许多感悟，在开始设计数据库之前，深入了解应用程序的需求是至关重要的。并且，利用概念设计阶段，我设计了数据库的实体关系模型(ERM)，标识了系统中的主要实体、属性和它们之间的关系。这有助于建立一个清晰的抽象模型，为后续的物理设计提供了基础。通过规范化过程，我优化了数据库表结构，消除了冗余数据，并确保了数据的一致性和完整性。合理使用规范化可以提高数据库的性能，并减少数据冗余。在实现数据库时，我编写了大量的 SQL 语句，包括表的创建、数据的插入、更新和查询等。熟练运用 SQL 是数据库设计和管理的重要技能之一。通过创建合适的索引，我提高了数据库查询的效率。同时，我了解了一些性能优化的方法，包括查询优化和数据库配置优化，以确保系统在处理大量数据时仍然能够高效运行。在设计数据库时，我考虑了事务的管理，以确保数据的一致性和完整性。合理使用事务可以防止数据损坏和丢失。了保护数据库中的敏感信息，我实施了一些安全性措施，包括合适的用户权限管理、加密等。确保数据库系统的安全性对于保护用户数据至关重要。

然而本次系统设计也有不足之处，例如设计的思路还是相对简单，对现实中更为复杂的情况考虑欠缺，而且提供给管理人员的功能较少，欠缺管理的灵活性。但无论如何，本次实验仍令我受益匪浅，通过这次数据库实验的课程设计，我不仅掌握了数据库设计和管理的基本原理，还培养了实际应用这些原理的能力。这将对我未来的数据库工作和项目中的数据库设计提供坚实的基础。