目录

[1. 数据库设计 2](#_Toc146173743)

[1.1. 用户需求和业务规则 2](#_Toc20806433)

[(1) 学生角色： 2](#_Toc1801368617)

[(2) 教师角色： 2](#_Toc377890513)

[(3) 管理员角色： 2](#_Toc1096707812)

[(1) 考勤规则： 3](#_Toc516054083)

[(2) 请假规则： 3](#_Toc1782006395)

[(3) 课程管理规则： 3](#_Toc1374539703)

[(4) 系统操作规则： 3](#_Toc1407197542)

[1.2. 概念设计 4](#_Toc531683983)

[1.2.1 实体及其属性 4](#_Toc333247114)

[1.2.2 实体关系 14](#_Toc246893622)

[1.2.3 数据完整性约束 21](#_Toc602698950)

[1.3. 视图设计 21](#_Toc2028373398)

[1.4. 存储过程与触发器 22](#_Toc1716287708)

1. **数据库设计**

数据库设计是指为数据的存储、管理和检索创建一个结构化的系统。它包括定义数据库的结构、关系和约束，以确保数据的一致性、完整性和高效访问。数据库设计的主要目标是满足应用程序和用户的需求，同时优化性能和维护便利性。

* 1. **用户需求和业务规则**

在该项目中使用的用户主要分为三种角色：**学生、老师和管理员**。这三种角色对应的主要需求如下所示：

1. **学生角色：**
2. **登录和注册：**学生应能够创建个人账户并登录系统。
3. **查看个人考勤记录：**学生能够查看自己的考勤记录，包括已出勤、缺勤、迟到等情况。
4. **请假申请：**学生可以提出请假申请，并在申请中提供请假原因和请假时间段。
5. **查阅课程表：**学生可以查看自己的课程表，以便了解每天的课程安排。
6. **考勤签到：**学生通过输入签到码对老师发布的考勤任务进行签到。
7. **教师角色：**
8. **登录和注册：**教师可以创建个人账户并登录系统。
9. **考勤记录管理：**教师能够查看自己任教课程学生的考勤情况，包括“已签到”和“缺勤”，并能具体查询到学生的签到时间和缺勤名单。
10. **发布和结束考勤任务：**教师能够发布和结束考勤任务，包括设定考勤时间、生成课程对应的签到码、立即结束考勤等。
11. **请假审批：**教师可以审批学生的请假申请，包括批准或拒绝请假请求，并在必要时提供审批意见。
12. **课程表管理：**教师可以创建和管理课程表，包括添加、编辑和删除课程信息。
13. **管理员角色：**
14. **系统设置：**管理员能够设置系统的全局参数，如学期开始和结束日期、考勤统计周期等。
15. **用户管理：**管理员可以管理学生和教师的账户，包括添加、编辑和删除用户信息。
16. **报表生成：**管理员可以生成考勤相关的报表，如学生出勤率报表、教师考勤记录报表等。
17. **审计日志：**管理员可以查看系统操作的审计日志，以确保系统安全性和追踪意外操作。

而对于整个系统来说，系统需要具备如下两个主要功能。

1. **数据统计和分析**：系统应能够收集和分析考勤数据，生成相关统计报表和图表，以帮助教师和管理员了解学生出勤情况。
2. **数据备份和恢复**：系统应定期备份数据，以防止数据丢失，并提供数据恢复功能。

根据上述提出的需求，我们能够大致得出该项目的业务规则，具体的业务规则如下所示。

1. **考勤规则：**
2. 学生需在规定的考勤时间内签到，否则将被标记为缺勤或迟到。
3. 教师可以随时查看并管理学生的考勤记录。
4. **请假规则：**
5. 学生需提前提交请假申请，并注明请假原因和时间段。
6. 教师需在一定时间内审批请假申请，并提供审批意见。
7. **课程管理规则：**
8. 教师可以创建、编辑和删除课程信息，并维护课程表的准确性。
9. 学生可以查看自己的课程表，以便按时上课。
10. **系统操作规则：**
11. 管理员需定期检查系统的审计日志，以确保系统操作的安全性。
12. 系统需定期备份数据，并提供数据恢复功能，以防止数据丢失。

用户需求和业务规则为系统设计提供了明确的指导，有助于确保系统功能的完整性和一致性。

* 1. **概念设计**

因为数据库设计的修改，产生的连带性是很大的。因此数据库设计适合使用瀑布模型。因为要提前设计好完整健壮的数据库，所以主要使用实体-关系图（ER图）来描述数据模型，定义实体、属性和关系。ER也称实体关系图（Entity Relationship Diagram），它提供了表示实体类型、属性和关系的方法，用来描述现实世界的概念模型，本学生考勤系统项目的概念设计包括以下主要实体和关系。**实体是设计阶段的概念，表是实现阶段的结构，限于篇幅限制下面将展示15个主要的实体，并且其都对应数据库表。**

**1.2.1 实体及其属性**

**下面这些表格将会详细展示各个实体结构与其属性的含义（其中使用加粗的字体来区分Key）**

|  |  |
| --- | --- |
| **学生**(Student)**实体** |  |
| **student\_id** | 学生ID,唯一标识学生的ID |
| name | 姓名，学生的姓名 |
| class\_no | 班级编号，学生所属的班级编号 |
| email | 电子邮件，学生的电子邮件地址 |
| passWord | 密码，学生登录系统的密码 |

表 1 学生实体表

|  |  |
| --- | --- |
| **教师**(Teacher)实体 |  |
| **teacher\_id** | 教师ID,唯一标识教师的ID |
| name | 姓名，教师的姓名 |
| department | 系别，教师所属的系别 |
| email | 电子邮件，教师的电子邮件地址 |
| passWord | 密码，教师登录系统的密码 |

表 2 老师实体表

|  |  |
| --- | --- |
| **课程**(Course)实体 |  |
| **course\_id** | 课程ID,唯一标识课程的ID |
| course\_name | 课程名称，课程的名称 |
| teacher\_id | 教师ID,教授该课程的教师ID |
| description | 课程描述，课程的详细描述 |

表 3 课程实体表

|  |  |
| --- | --- |
| **考勤(**Attendance**)实体** |  |
| **attendance\_id** | 考勤ID,唯一标识考勤记录的ID |
| student\_id | 学生ID,被考勤的学生ID |
| course\_id | 课程ID,课程ID |
| attendance\_date | 考勤日期，考勤的日期 |
| status | 状态，考勤状态(如已出勤、缺勤、迟到) |
| reason | 原因，考勤状态的原因(如请假原因) |

表 4 考勤实体表

|  |  |
| --- | --- |
| **请假申请(**LeaveApplication**)实体** |  |
| **leave\_id** | 请假ID,唯一标识请假申请的ID |
| student\_id | 学生ID,提出请假申请的学生ID |
| course\_id | 课程ID,申请请假的课程ID |
| leave\_date | 请假日期，申请请假的日期 |
| reason | 请假原因，请假的原因 |
| status | 审批状态，请假申请的审批状态(如待审批、已批准、已拒绝) |

表 5 请假申请实体表

|  |  |
| --- | --- |
| **考勤任务(**AttendanceTask**)实体** |  |
| **task\_id** | 任务ID,唯一标识考勤任务的ID |
| course\_id | 课程ID,课程ID |
| teacher\_id | 教师ID,发布考勤任务的教师ID |
| task\_date | 任务日期，考勤任务的日期 |
| code | 签到码，学生用于签到的码 |

表 6 考勤任务实体表

|  |  |
| --- | --- |
| **课程表(**ClassSchedule**)实体** |  |
| **schedule\_id** | 课程表ID,唯一标识课程表记录的ID |
| course\_id | 课程ID,课程ID |
| day\_of\_week | 星期几，上课的星期几 |
| start\_time | 开始时间，上课的开始时间 |
| end\_time | 结束时间，上课的结束时间 |
| location | 地点，上课的地点 |

表 7 课程表实体表

|  |  |
| --- | --- |
| **请假缺勤状态记录**（LeaveRecord）实体 | |
| **RecordID** | **每一条请假缺勤记录的唯一标识符** |
| stu\_id | 学生ID，唯一标识学生的ID，关联到表attendance\_information |
| course\_id | 学生所属班级的唯一标识符，关联到班级信息表 |
| teacher\_id | 教师ID，唯一标识教师的ID |
| status | 请假缺勤的状态，分别为“请假”和“缺勤”两种状态 |
| week | 考勤任务所处周次 |
| date | 考勤任务所处日期 |
| leave\_reason | 请假原因 |

表 8 请假缺勤状态记录实体表

|  |  |
| --- | --- |
| **每位老师的课程信息**（TeacherSchedule）实体 | |
| **RecordID** | **每一条课程记录的唯一标识符** |
| teacher\_id | 教师ID，唯一标识教师的ID |
| course\_id | 课程编号，唯一标识课程的ID |
| course\_name | 课程名称 |
| date | 课程所在日期 |
| section | 课程所在节次 |
| semester | 课程所在学期 |

表 9 每位老师的课程信息实体表

|  |  |
| --- | --- |
| **学生学期考勤记录**（StudentAttendanceRecord）实体 | |
| **RecordID** | **每一条考勤记录的唯一标识符** |
| stu\_id | 学生ID，唯一标识学生的ID |
| stu\_name | 学生ID对应的姓名 |
| course\_id | 学生所属班级的唯一标识符，关联到班级信息表 |
| teacher\_id | 教师ID，唯一标识教师的ID |
| date | 课程所在日期 |
| semester | 课程所在学期 |
| status | 学生本次课程出勤情况 |
| reason | 没有正常出勤的原因（已出勤则为空） |

表 10 学生学期考勤记录实体表

|  |  |
| --- | --- |
| **课程考勤状态表**（CourseAttendance）实体 | |
| **RecordID** | **每一条考勤记录的唯一标识符** |
| course\_id | 学生所属班级的唯一标识符，关联到班级信息表 |
| stu\_id | 学生ID，唯一标识学生的ID |
| stu\_name | 学生ID对应姓名 |
| teacher\_id | 教师ID，唯一标识教师的ID |
| date | 课程所在日期 |
| status | 学生出勤状态 |
| leave\_reason | 请假原因 |

表 11 课程考勤状态实体表

|  |  |
| --- | --- |
| **单独课程信息**（CourseInfo）实体 | |
| **course\_id** | 唯一标识课程的ID |
| course\_name | 课程的名称 |
| teacher\_id | 教师ID |
| description | 课程的详细描述 |

表 12 单独课程信息实体表

|  |  |
| --- | --- |
| **学生每学期课程考勤情况**（StudentCourseAttendance）**实体** | |
| **student\_id** | 唯一标识学生的ID，联合主键之一 |
| **course\_id** | 唯一标识课程的ID，联合主键之一 |
| **semester** | 唯一标识学期，联合主键之一 |
| total\_attendance | 该课程的总考勤次数 |
| total\_absent | 该课程的缺情次数 |
| total\_late | 该课程的吃到次数 |

表 13 学生每学期课程考勤情况实体表

|  |  |
| --- | --- |
| **IP和API访问关系**（IPApiAccess）**实体** | |
| **access\_id** | 唯一标识访问记录的ID |
| ip\_address | 访问的IP地址 |
| api\_endpoint | 访问的API端点 |
| access\_date | 访问的日期 |
| access\_port | 访问使用的端口 |
| access\_count | 在该日期该端口的访问次数 |

表 14 IP和API访问关系实体表

|  |  |
| --- | --- |
| **IP黑名单**（IPBan）**实体** | |
| **ban\_id** | 唯一标识禁止记录的ID |
| ip\_address | 访问设备的IP地址 |
| mac\_address | 访问设备的物理地址 |
| ban\_date | 禁止的开始日期 |
| ban\_duration | 禁止访问的时限 |

表 15 IP黑名单实体表

根据如上实体的罗列情况（由于篇幅限制只列出15个实体），我们可以进行数据库中各个表的设计，具体使用的SQL代码如下：

-- 表 1 学生实体表

CREATE TABLE Student (

student\_id INT PRIMARY KEY,

name VARCHAR(100),

class\_no VARCHAR(10),

email VARCHAR(100),

password VARCHAR(50)

);

-- 表 2 教师实体表

CREATE TABLE Teacher (

teacher\_id INT PRIMARY KEY,

name VARCHAR(100),

department VARCHAR(100),

email VARCHAR(100),

password VARCHAR(50)

);

-- 表 3 课程实体表

CREATE TABLE Course (

course\_id INT PRIMARY KEY,

course\_name VARCHAR(100),

teacher\_id INT,

description TEXT,

FOREIGN KEY (teacher\_id) REFERENCES Teacher(teacher\_id)

);

-- 表 4 考勤实体表

CREATE TABLE Attendance (

attendance\_id INT PRIMARY KEY AUTO\_INCREMENT,

student\_id INT,

course\_id INT,

attendance\_date DATE,

status VARCHAR(20),

reason TEXT,

FOREIGN KEY (student\_id) REFERENCES Student(student\_id),

FOREIGN KEY (course\_id) REFERENCES Course(course\_id)

);

-- 表 5 请假申请实体表

CREATE TABLE LeaveApplication (

leave\_id INT PRIMARY KEY AUTO\_INCREMENT,

student\_id INT,

course\_id INT,

leave\_date DATE,

reason TEXT,

status VARCHAR(20),

FOREIGN KEY (student\_id) REFERENCES Student(student\_id),

FOREIGN KEY (course\_id) REFERENCES Course(course\_id)

);

-- 表 6 考勤任务实体表

CREATE TABLE AttendanceTask (

task\_id INT PRIMARY KEY AUTO\_INCREMENT,

course\_id INT,

teacher\_id INT,

task\_date DATE,

code VARCHAR(20),

FOREIGN KEY (course\_id) REFERENCES Course(course\_id),

FOREIGN KEY (teacher\_id) REFERENCES Teacher(teacher\_id)

);

-- 表 7 课程表实体表

CREATE TABLE ClassSchedule (

schedule\_id INT PRIMARY KEY AUTO\_INCREMENT,

course\_id INT,

day\_of\_week VARCHAR(10),

start\_time TIME,

end\_time TIME,

location VARCHAR(100),

FOREIGN KEY (course\_id) REFERENCES Course(course\_id)

);

-- 表 8 请假缺勤状态记录实体表

CREATE TABLE LeaveRecord (

record\_id INT PRIMARY KEY AUTO\_INCREMENT,

student\_id INT,

course\_id INT,

teacher\_id INT,

status VARCHAR(20),

week INT,

date DATE,

leave\_reason TEXT,

FOREIGN KEY (student\_id) REFERENCES Student(student\_id),

FOREIGN KEY (course\_id) REFERENCES Course(course\_id),

FOREIGN KEY (teacher\_id) REFERENCES Teacher(teacher\_id)

);

-- 表 9 每位老师的课程信息实体表

CREATE TABLE TeacherSchedule (

record\_id INT PRIMARY KEY AUTO\_INCREMENT,

teacher\_id INT,

course\_id INT,

course\_name VARCHAR(100),

date DATE,

section VARCHAR(20),

semester VARCHAR(20),

FOREIGN KEY (teacher\_id) REFERENCES Teacher(teacher\_id),

FOREIGN KEY (course\_id) REFERENCES Course(course\_id)

);

-- 表 10 学生学期考勤记录实体表

CREATE TABLE StudentAttendanceRecord (

record\_id INT PRIMARY KEY AUTO\_INCREMENT,

student\_id INT,

student\_name VARCHAR(100),

course\_id INT,

teacher\_id INT,

date DATE,

semester VARCHAR(20),

status VARCHAR(20),

reason TEXT,

FOREIGN KEY (student\_id) REFERENCES Student(student\_id),

FOREIGN KEY (course\_id) REFERENCES Course(course\_id),

FOREIGN KEY (teacher\_id) REFERENCES Teacher(teacher\_id)

);

-- 表 11 课程考勤状态实体表

CREATE TABLE CourseAttendance (

record\_id INT PRIMARY KEY AUTO\_INCREMENT,

course\_id INT,

student\_id INT,

student\_name VARCHAR(100),

teacher\_id INT,

date DATE,

status VARCHAR(20),

leave\_reason TEXT,

FOREIGN KEY (course\_id) REFERENCES Course(course\_id),

FOREIGN KEY (student\_id) REFERENCES Student(student\_id),

FOREIGN KEY (teacher\_id) REFERENCES Teacher(teacher\_id)

);

-- 表 12 单独课程信息实体表

CREATE TABLE CourseInfo (

course\_id INT PRIMARY KEY,

course\_name VARCHAR(100),

teacher\_id INT,

description TEXT,

FOREIGN KEY (teacher\_id) REFERENCES Teacher(teacher\_id)

);

-- 表 13 学生每学期课程考勤情况实体表

CREATE TABLE StudentCourseAttendance (

student\_id INT,

course\_id INT,

semester VARCHAR(20),

total\_attendance INT,

total\_absent INT,

total\_late INT,

PRIMARY KEY (student\_id, course\_id, semester),

FOREIGN KEY (student\_id) REFERENCES Student(student\_id),

FOREIGN KEY (course\_id) REFERENCES Course(course\_id)

);

-- 表 14 IP和API访问关系实体表

CREATE TABLE IPApiAccess (

access\_id INT PRIMARY KEY AUTO\_INCREMENT,

ip\_address VARCHAR(45),

api\_endpoint VARCHAR(255),

access\_date DATE,

access\_port INT,

access\_count INT

);

-- 表 15 IP黑名单实体表

CREATE TABLE IPBan (

ban\_id INT PRIMARY KEY AUTO\_INCREMENT,

ip\_address VARCHAR(45),

mac\_address VARCHAR(255),

ban\_date DATE,

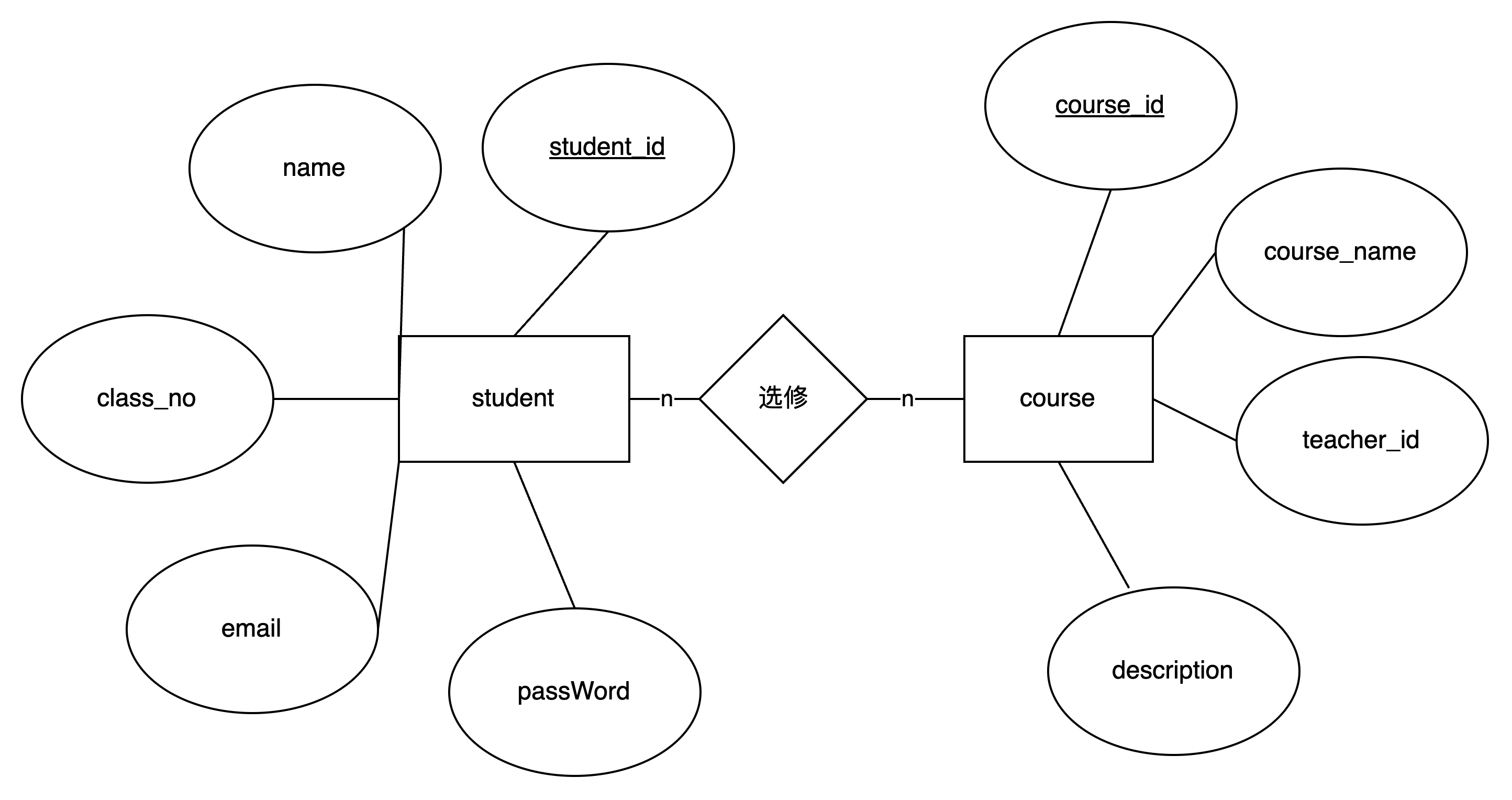
ban\_duration INT

);

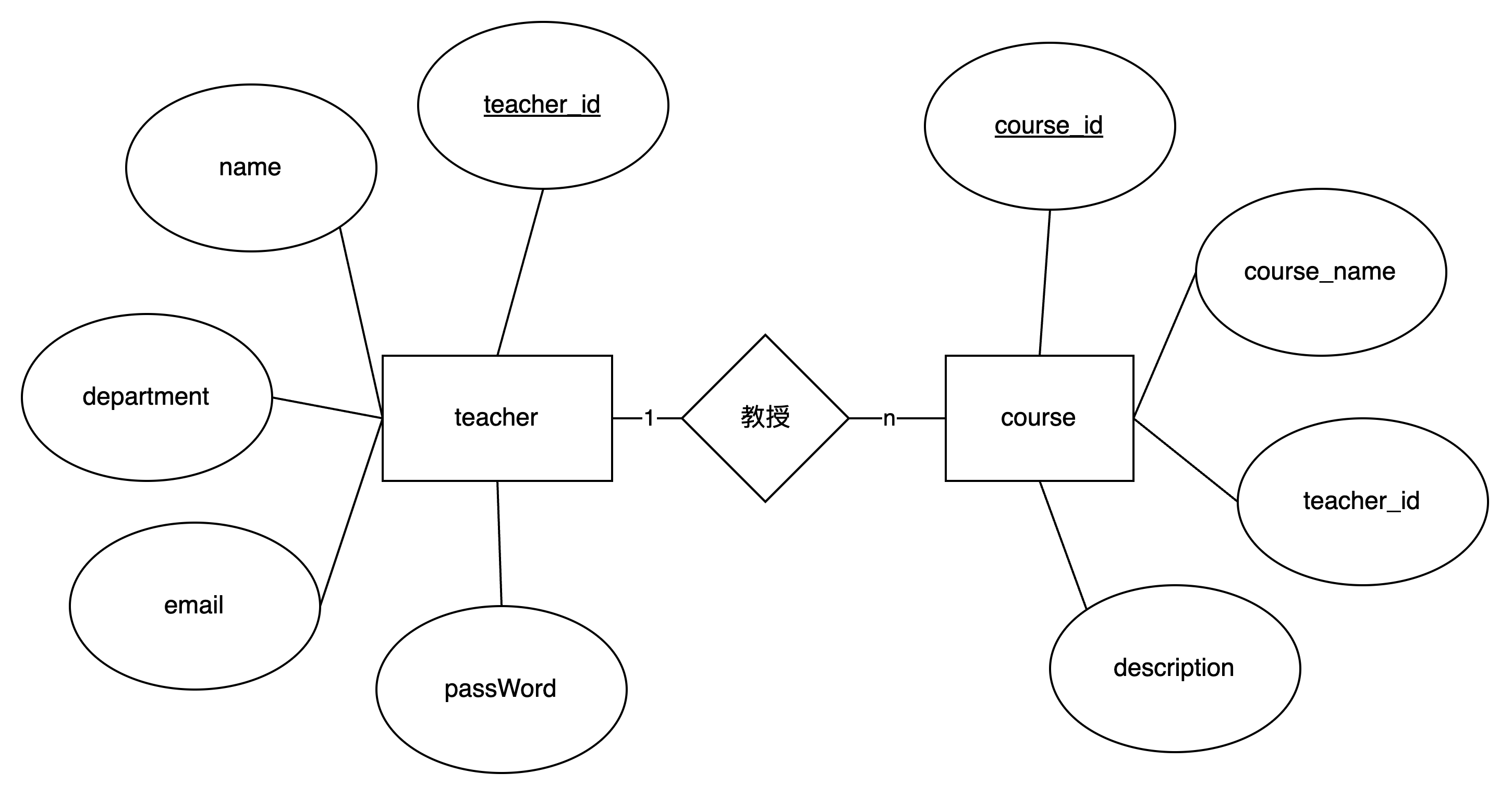
**1.2.2 实体关系**

为了更加清晰地理解本项目中的实体关系，我们可以通过详细描述每个实体之间的关系，并解释其逻辑连接。下面我们将会通过关系描述、实现方式、ER图关系三种方式来展示实体关系。

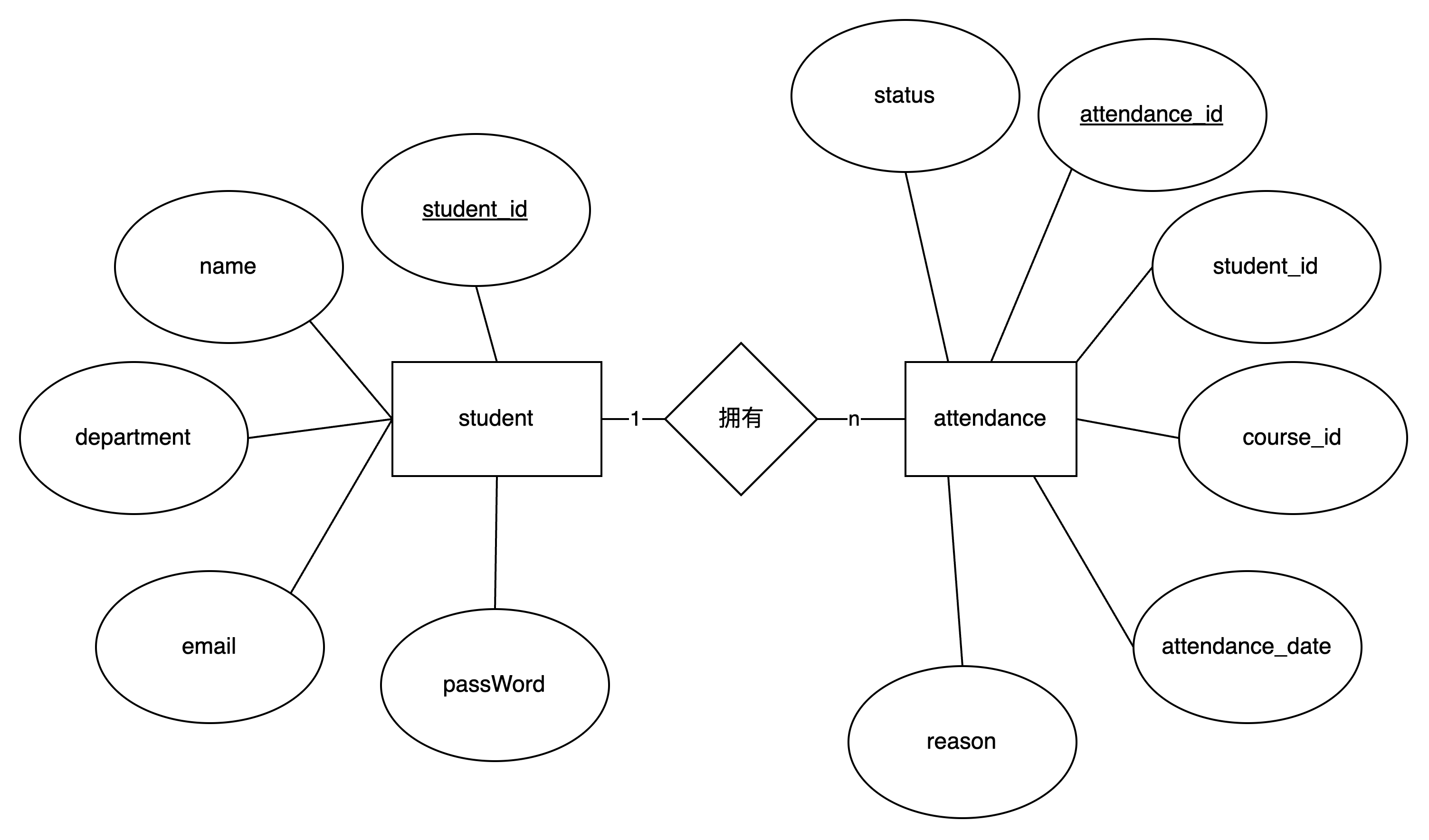
1. 学生与课程
   1. 关系描述：一个学生可以选修多门课程，一个课程可以有多个学生选修，因此学生和课程之间存在**多对多（N:N）**的关系。
   2. 实现方式：使用中间表**CourseSelection**来表示学生和课程之间多对多的关系，该表包含**student\_id**和**course\_id**字段，其中学生**ID**是关联到学生实体的唯一标识，课程**ID**同理。
   3. ER图关系：



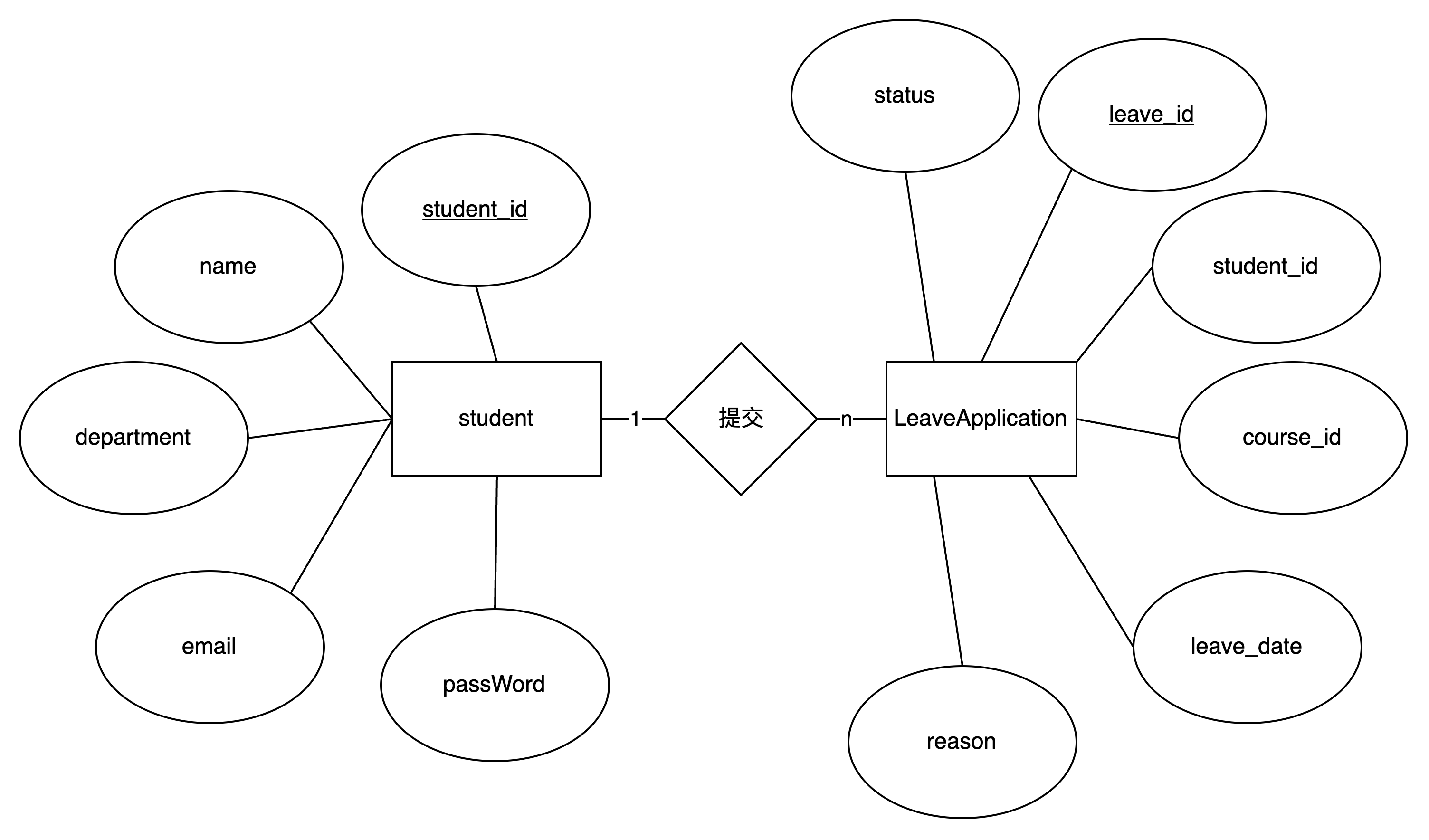
1. 教师与课程
   1. 关系描述:一个教师可以教授多门课程，一门课程由一个教师教授。教师和课程之间存在一对多（1:N）的关系。
   2. 实现方式：课程表中的teacher\_id字段外键关联到教师表的teacher\_id字段。
   3. ER图关系：



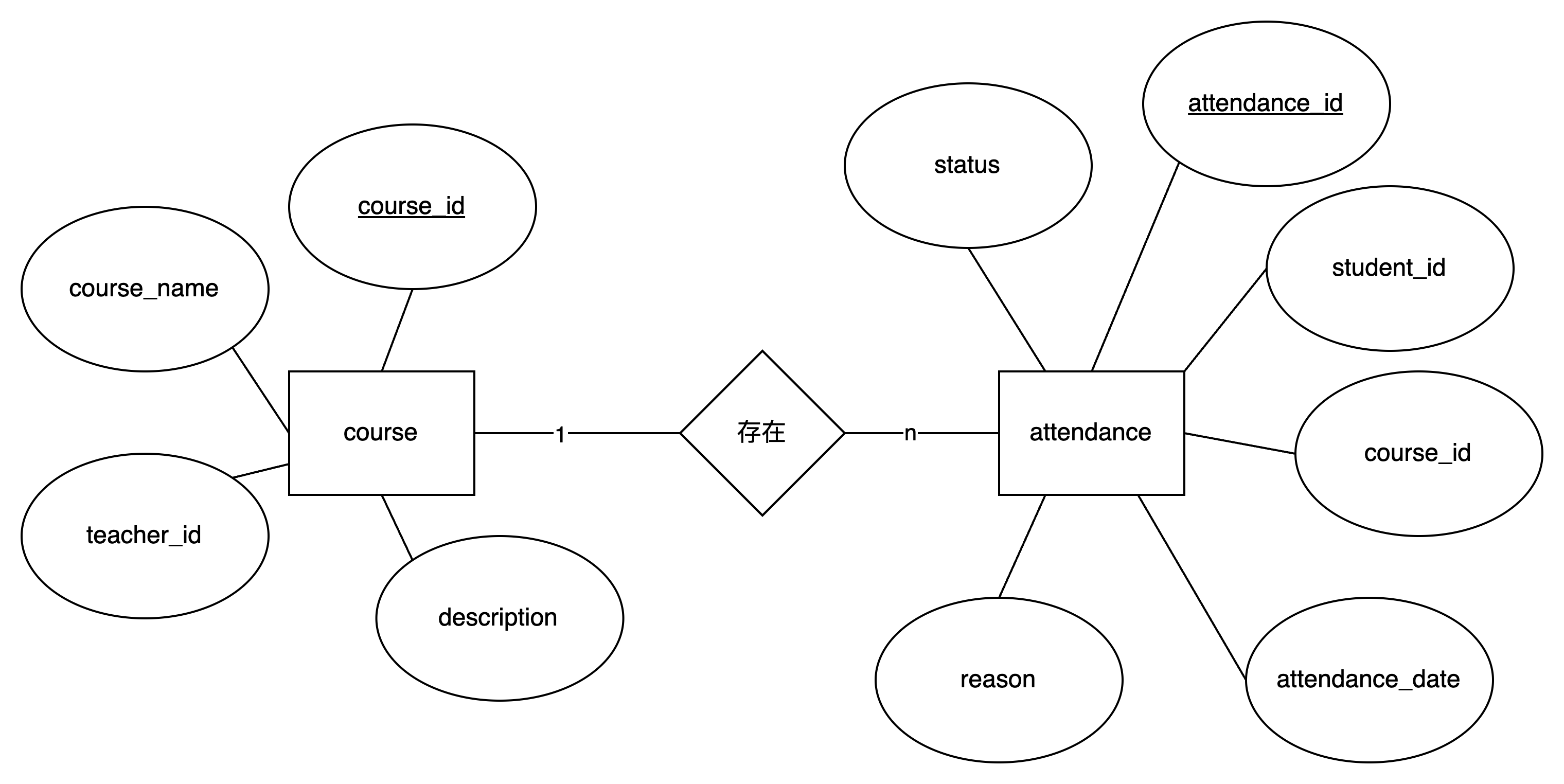
1. 学生与考勤
   1. 关系描述；一个学生可以有多条考勤记录，一条考勤记录属于一个学生。学生和考勤之间存在一对多（1:N）的关系。
   2. 实现方式：考勤表中的student\_id字段外键关联到学生表的student\_id字段
   3. ER图关系：



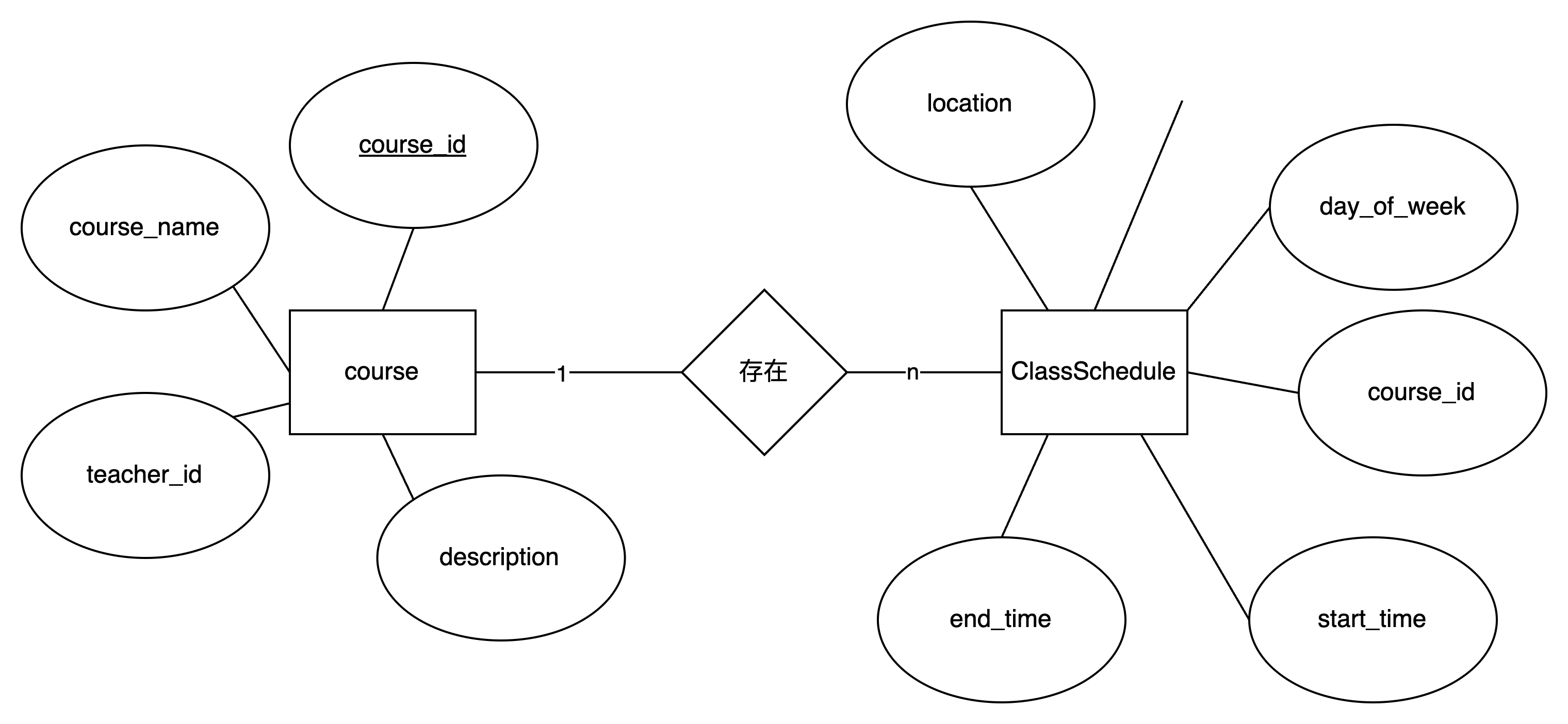
1. 学生与请假申请
   1. 关系描述：一个学生可以提交多条请假申请，一条请假申请属于一个学生。学生和请假申请之间存在一对多的关系（1:N）
   2. 实现方式：请假申请表中的student\_id字段外键关联到学生表的student\_id字段
   3. ER图关系：



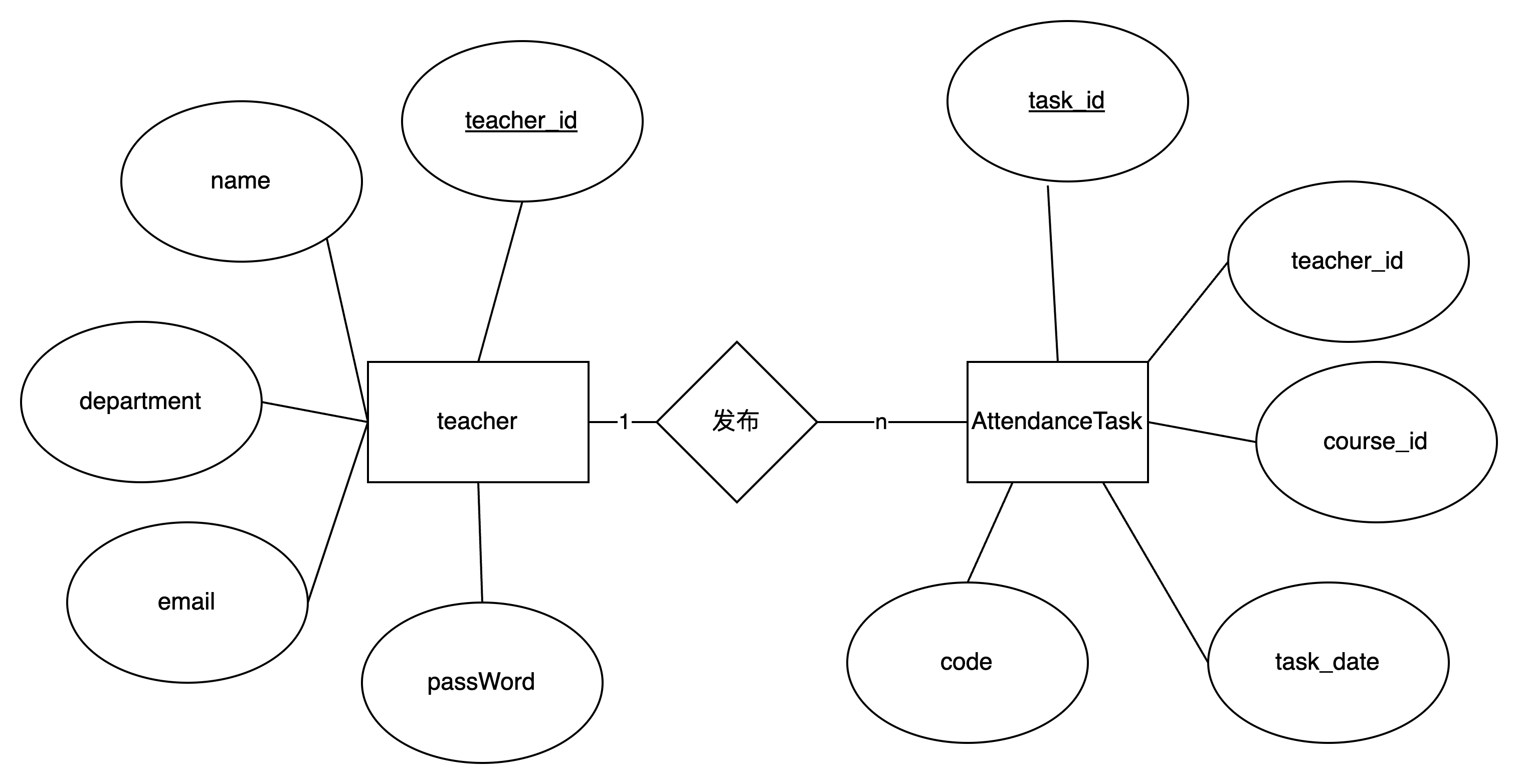
1. 课程与考勤
   1. 关系描述：一个课程可以有多条考勤记录，一条考勤记录属于一个课程。课程和考勤之间存在一对多（1:N）的关系。
   2. 实现方式：考勤表中的course\_id 字段外键关联到课程表的course\_id 字段。
   3. ER 图关系：



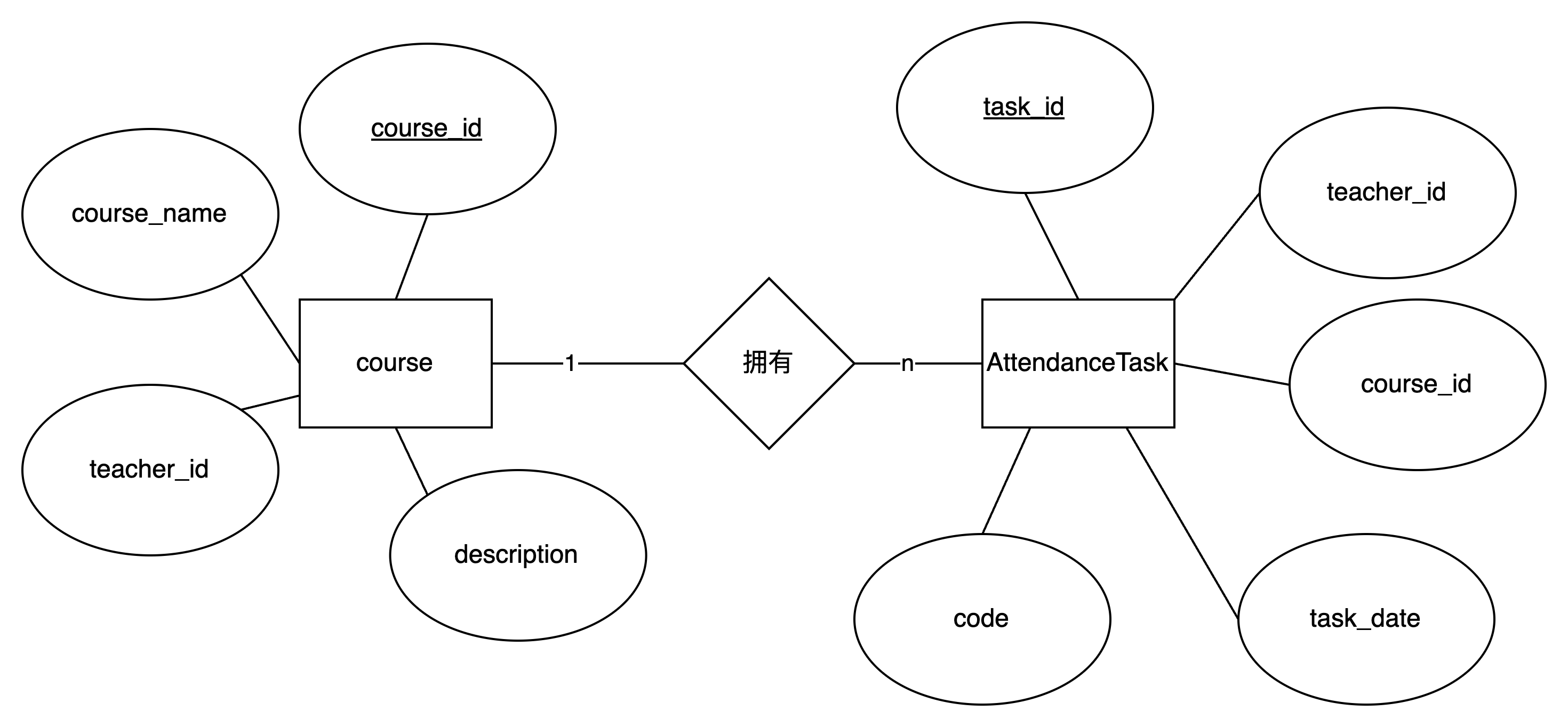
1. 课程与课程表
   1. 关系描述：一个课程可以出现在多条课程表记录中，一条课程表记录属于一个课程。课程和课程表之间存在一对多（1:N）的关系。
   2. 实现方式：课程表中的 course\_id 字段外键关联到课程表的 course\_id 字段。
   3. ER 图关系：



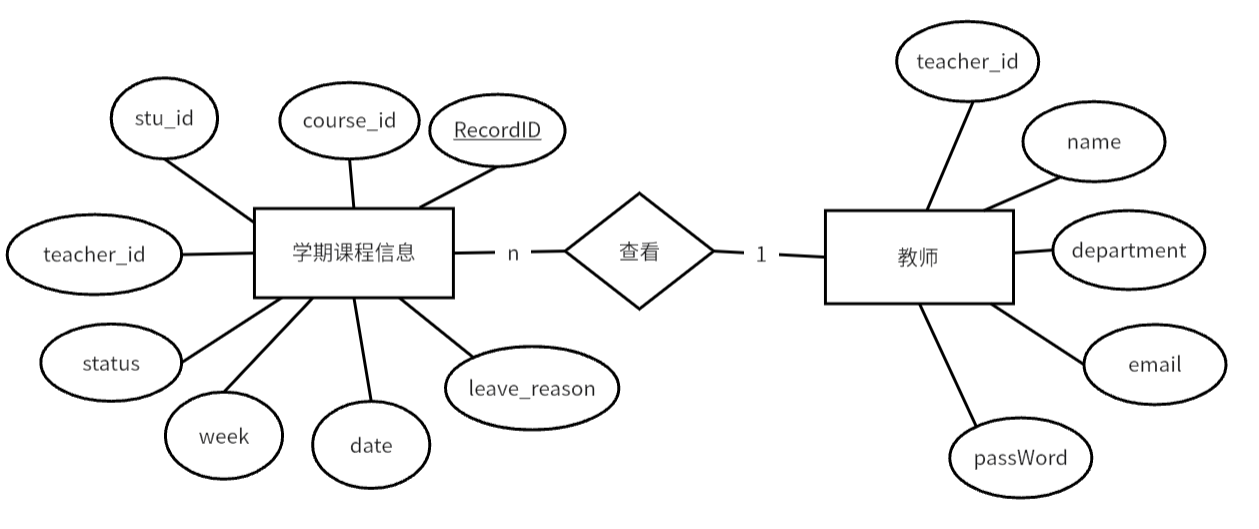
1. 教师与考勤任务
   1. 关系描述：一个教师可以发布多条考勤任务，一条考勤任务属于一个教师。教师和考勤任务之间存在一对多（1:N）的关系。
   2. 实现方式：考勤任务表中的teacher\_id字段外键关联到教师表的teacher\_id字段。
   3. ER图关系：



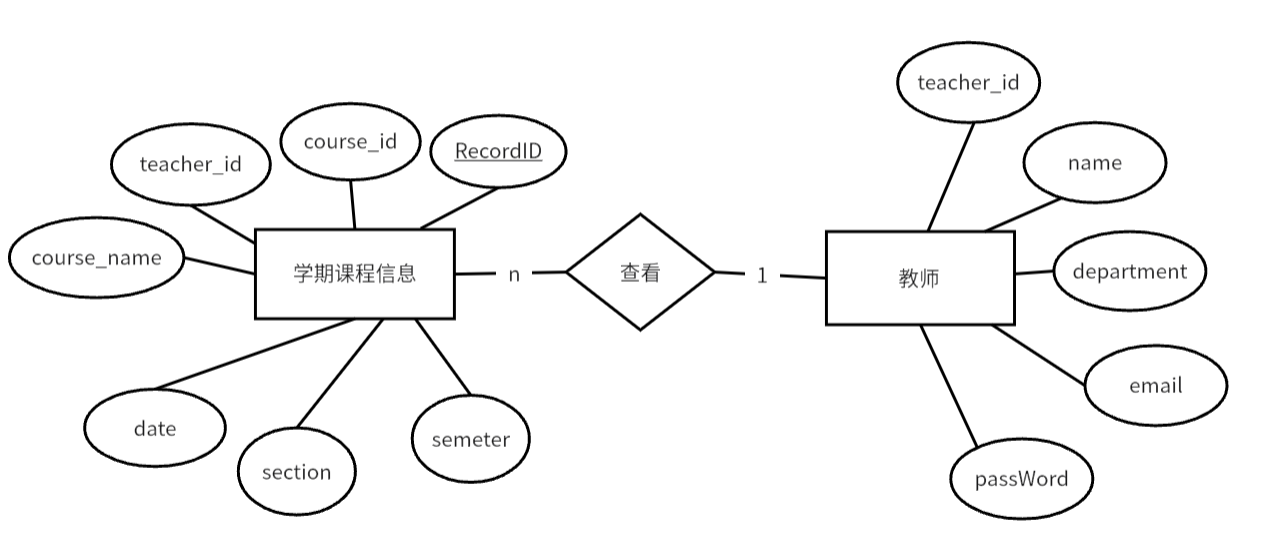
1. 课程与考勤任务
   1. 关系描述：一个课程可以有多条考勤任务，一条考勤任务属于一个课程。课程和考勤任务之间存在一对多（1:N）的关系。
   2. 实现方式：考勤任务表中的course\_id字段外键关联到课程表的course\_id字段。
   3. ER 图关系：



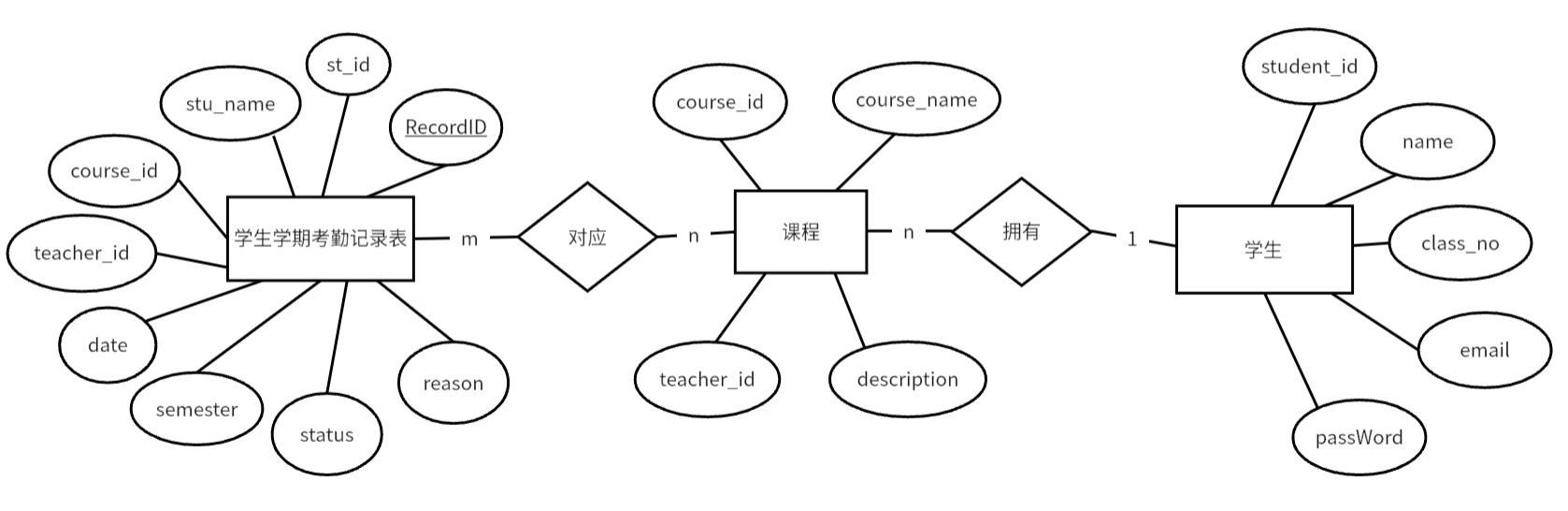
1. 教师与学生课程请假缺勤状态记录
   1. 关系描述：一个教师可以查看自己所教课程关于学生请假与缺勤的状态情况，一个老师对应多节课程，每节课程可以有多位同学。教师和各位同学的请假缺勤状态记录存在一对多（1:N）的关系。
   2. 实现方式：请假缺勤状态记录表中的stu\_id字段外键关联到考勤表的student\_id字段；leave\_reason字段外键关联到考勤表的reason字段。
   3. ER图关系：



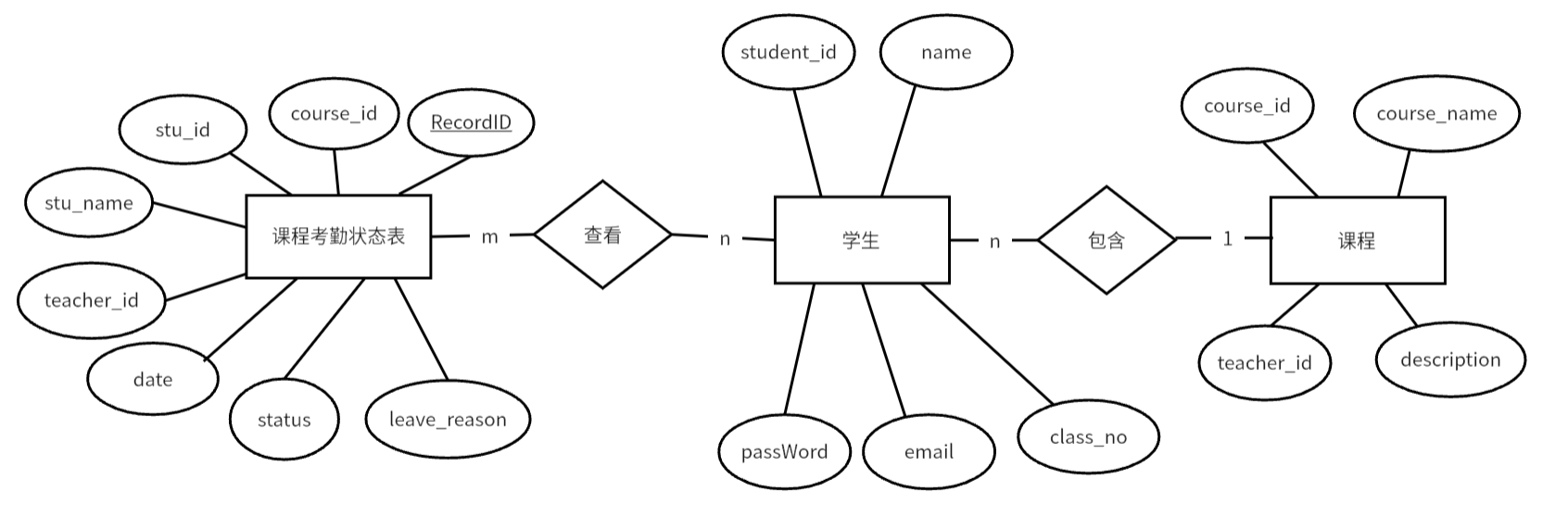
1. 教师与课程表
   1. 关系描述：每位教师可以查看自己每一学期需要教授的课程，且每位教师可以在同一学期教授多节课程。教师与所需教授课程存在一对多（1:N）的关系。
   2. 实现方式：每位教师教授的课程表中的teacher\_id字段外键关联到教师表的teacher\_id字段；course\_id和course\_name字段外键关联到课程表的course\_id和course\_name字段。
   3. ER图关系：



1. 学生与本人本学期考勤情况信息表
   1. 关系描述：学生每学期有多节需要上的课程，多节课程对应多个考勤任务。学生与所需参与的考勤任务存在一对多（1:N）的关系。
   2. 实现方式：每个同学考勤情况信息表中的stu\_id和stu\_name字段外键关联到学生表的student\_id和name字段；teacher\_id字段外键关联到教师表的teacher\_id字段；reason字段外键关联到考勤表的reason字段。
   3. ER图关系：



1. 课程与对应考勤状态表
   1. 关系描述：每个课程会包含多个学生，多个学生对应多个考勤状态。课程与学生考勤情况存在一对多（1:N）的关系。
   2. 实现方式：每个同学考勤情况信息表中的course\_id字段外键关联到课程表的course\_id字段；stu\_id和stu\_name字段外键关联到学生表的student\_id和name字段；teacher\_id字段外键关联到教师表的teacher\_id字段；leave\_reason字段外键关联到考勤表的reason字段。
   3. ER图关系：



**1.2.3 数据完整性约束**

**为了确保数据的准确性和一致性，数据库设计中需要定义一些完整性约束，包括了实体完整性、引用完整性、域完整性以及用户定义完整性。**

* 1. 实体完整性：**确保每个实体的主键是唯一的，不为空，OrderID、student\_id、teacher\_id、course\_id等必须唯一且不为空。**
  2. 引用完整性：确保外键引用的实体存在。Attendance表中的student\_id必须存在于Student表中，course\_id必须存在于Course表中。
  3. 域完整性：确保属性值在合理范围内。attendance\_date必须是合法的日期，status只能是预定义的状态（如”已出勤”、“缺勤”、“迟到”）。
  4. 用户定义完整性：特定业务规则约束。在这里每个Course必须有且仅有一名教师（teacher\_id）。
  5. **视图设计**

为了满足不同用户的需求，我们可以创建一些视图，以便用户查询和操作数据。例如可以创建学生考勤视图，用于展示某个学生的所有考勤记录。

CREATE VIEW StudentAttendanceView AS

SELECT s.student\_id, s.name, c.course\_name, a.attendance\_date, a.status

FROM Student s

JOIN Attendance a ON s.student\_id = a.student\_id

JOIN Course c ON a.course\_id = c.course\_id;

也可以创建课程考勤视图，用于展示某门课程的所有考勤记录。

CREATE VIEW CourseAttendanceView AS

SELECT c.course\_id, c.course\_name, t.name as teacher\_name, a.attendance\_date, a.status

FROM Course c

JOIN Teacher t ON c.teacher\_id = t.teacher\_id

JOIN Attendance a ON c.course\_id = a.course\_id;

同理还可以创建请假申请视图，用于展示所有代审批的请假申请。

CREATE VIEW PendingLeaveApplicationsView AS

SELECT l.leave\_id, s.name as student\_name, c.course\_name, l.leave\_date, l.reason

FROM LeaveApplication l

JOIN Student s ON l.student\_id = s.student\_id

JOIN Course c ON l.course\_id = c.course\_id

WHERE l.status = '待审批';

* 1. **存储过程与触发器**

为了自动化某些操作和维护数据完整性，可以设计一些存储过程和触发器，在创建新的考勤记录时我们可以用如下代码。

CREATE PROCEDURE AddAttendanceRecord(

IN p\_student\_id INT,

IN p\_course\_id INT,

IN p\_attendance\_date DATE,

IN p\_status VARCHAR(10),

IN p\_reason VARCHAR(255)

)

BEGIN

INSERT INTO Attendance (student\_id, course\_id, attendance\_date, status, reason)

VALUES (p\_student\_id, p\_course\_id, p\_attendance\_date, p\_status, p\_reason);

END;

而对于触发器，在该项目中可能涉及到的触发器设置是更新课程人数统计，当课程人数改变或者课程人数达到一定的值时触发该机制，更新表。

CREATE TRIGGER UpdateCourseStudentCount

AFTER INSERT ON CourseSelection

FOR EACH ROW

BEGIN

UPDATE Course

SET student\_count = student\_count + 1

WHERE course\_id = NEW.course\_id;

END;