date: 2020/3/14 **S#:** U201814168 **name:** Li Ziyi/李子怡 **class:** 自实1801

Tourism

ODL

```
interface AdministrativeDivision (key code)
{
    attribute string code;
    attribute string name;
    attribute string population;
    attribute string size;

    relationship set<Tourist> inChargeOf
        inverse Tourist::comesFrom;
    relationship set<Destination> haveJuristictionOf
        inverse Destination::belongsTo;
}

interface Tourist (key ID)
{
    attribute string ID;
    attribute string name;
    attribute string gender;
    attribute string birthdate;

    relationship AdministrativeDivision comesFrom
        inverse AdministrativeDivision::inChargeOf;

}

interface Destination (key name)
{
    attribute string name;
    attribute string type;
    attribute string level;
    attribute Set<string> content;

    relationship AdministrativeDivision belongsTo
```

```
        inverse AdminstrativeDivison::haveJuristictonOver;
}
```

Entity-Relation Diagram

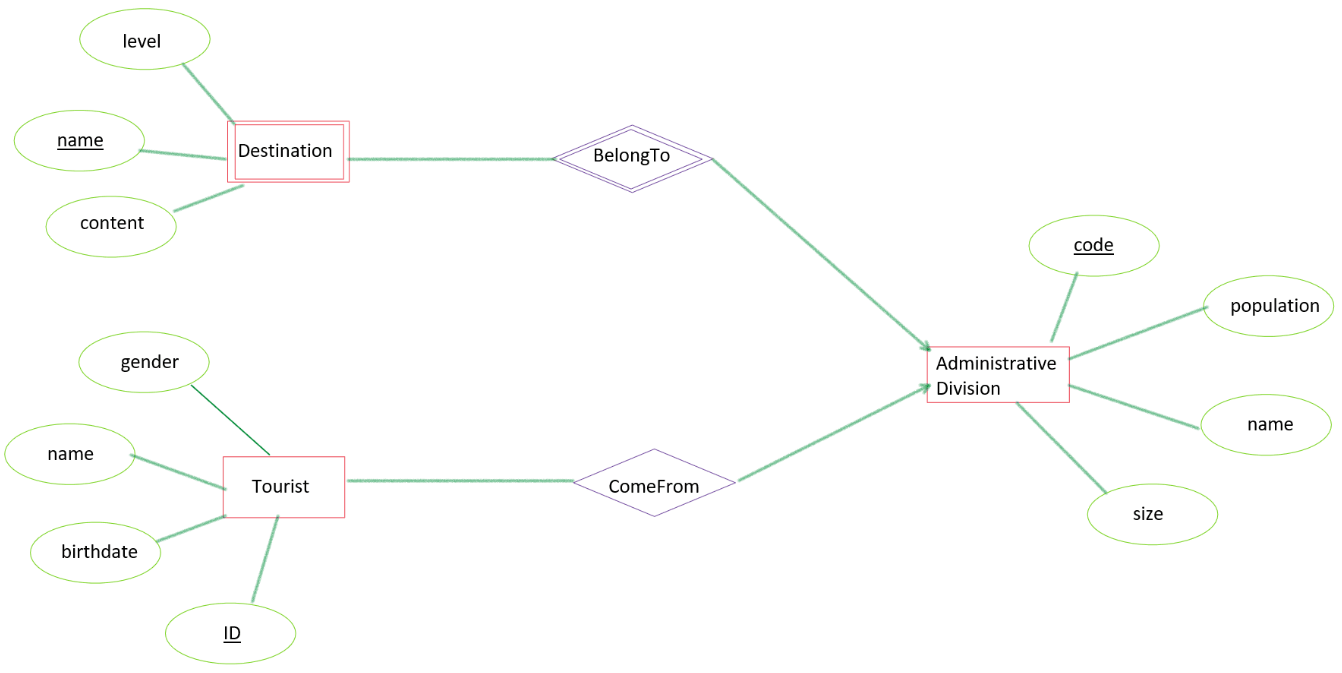

figure 1 Weak Entity

Relation Schema

AdiministrationDivision(<u>code</u>, name, population, size)

Tourist(<u>ID</u>, name, gender, birthdate)

Destination(<u>name</u>,type,content,<u>ADcode</u>)

Student and course

ODL

My consideration of the design contains:

1. each student can choose more than one course
2. each course can be chosed by more than one student
3. each course hold more than one exam (different term and different time)

4. one certain exam is held by the exact course
5. each exam can be attended by more than one student
6. each student attend a list of exams and thus attain the result
   the exams is sorted by time, so the students can gain results from
   i. different courses
   ii. different results of the same course (in case of failing)

```
interface Student (key S#)
{
    attribute string S#;
    attribute string name;
    attribute string gender;
    attribute string birthdate;
    attribute string tel;
    attribute string QQ;

    relationship Set<Course> choose
        inverse Course::chosedBy;
    relationship List<Exam> attend
        inverse Exam::attendedBy;
}

interface Course (key C#)
{
    attribute string C#;
    attribute string name;
    attribute string department;
    attribute float hours;
    attribute float credits;

    relationship Set<Student> chosedBy
        inverse Student::choose;
    relationship List<Exam>  hold
        inverse Exam heldBy
}

interface Exam
{
    attribute string date;
    attribute float result;

    relationship Course heldBy
        inverse Course::hold;
    relationship Set<Student>::attendedBy
        inverse Student::attend;
}
```

Entity-Relation Diagram

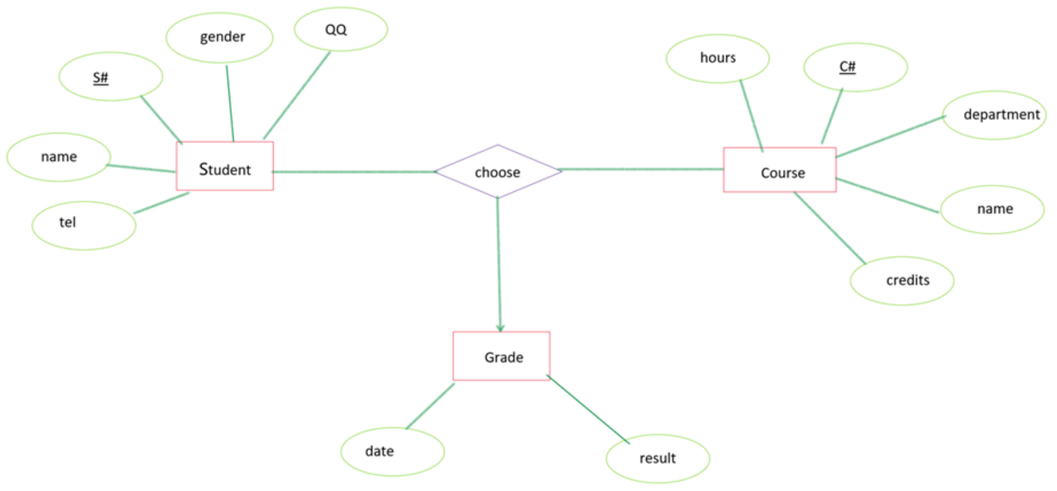

figure 2 Mutiway relationship

Use weak entity sets to eliminate multiway relationship:

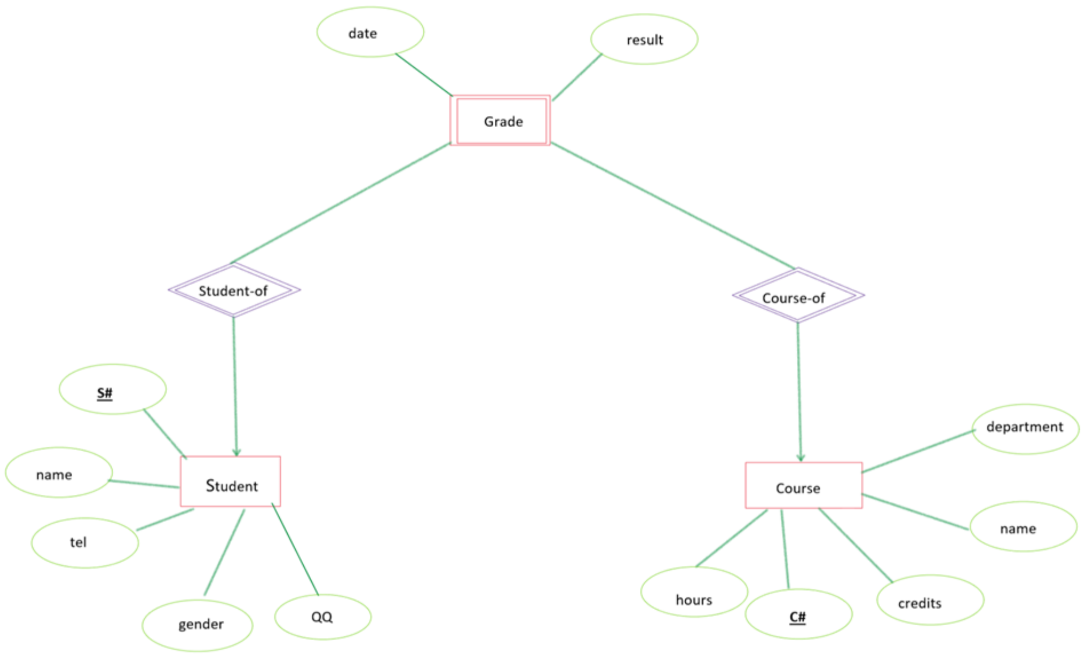

figure 3 Weak entity sets

Relation Schema

Student(S#, name, gender, birthdate, tel, QQ)

Course(C#, name, department, hours, credits)

Grade(name, date, result,S#, C#,)