ERD: Web-Based Tool for Drawing ER Diagrams

CS2102

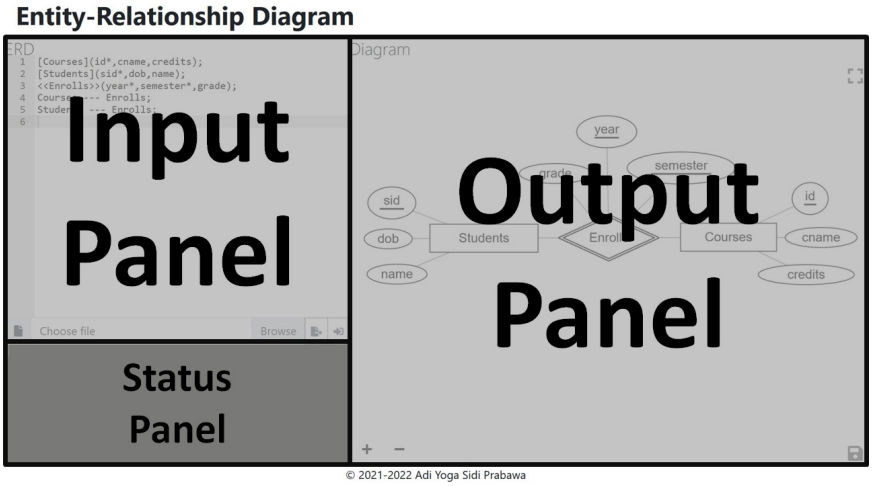
Database Systems

ERD is a web-based tool[1] for drawing ER diagrams. The tool takes an input an ER diagram specification (*or* ***ERD specification***) and translates it into a graphical representation using Cytoscape.js library. You may then interact with this graph directly to rearrange it if the layout is not up to your satisfaction.

The purpose of this document is to teach you how to use the ERD specification language and how to use the ERD tool to generate an ER diagram image file from an input ERD specification.

1 ERD Tool

The figure below shows the web-based interface of the ERD tool which consists of three panels. The top left panel is the **input panel** which is a text area for typing/editing your ERD specification. The right panel is the **output panel** which displays the ER diagram generated from the input ERD specification. Finally, the bottom left panel is the **status panel** which is a text area for displaying error messages.



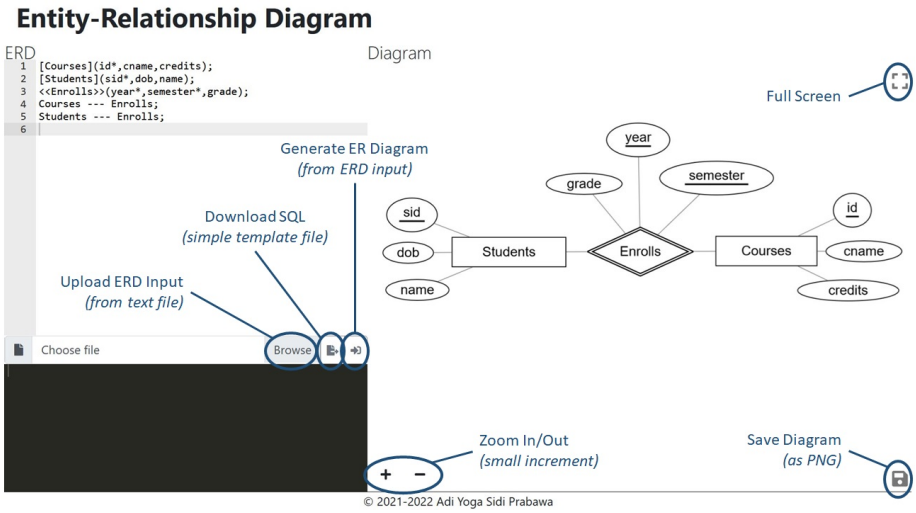
[1] The tool is available at `https://www.comp.nus.edu.sg/~adi-yoga/CS2102/ERD/` or `https://www.comp.nus.edu.sg/~CS2102/Tools/ERD/`

There are two ways to input a ERD specification using the ERD tool. The first way is to type the specification using the **input panel**. The second way is to load a ERD specification file by clicking the **Browse** button; the loaded specification file will be displayed in the input panel.

To generate a ER diagram from the ERD specification, you simply need to click on the button next to the **Browse** button shown below.

To save the generated ER diagram layout into a PNG image file, click on the bottom right save icon.

The buttons and their functions can be seen on the layout below.



1.1 Shortcuts

- `<CTRL> + <ENTER>`: Generate ER diagram from ERD input
- `<CTRL> + <+>`: Zoom in with small increment
- `<CTRL> + <->`: Zoom out with small increment
- `<CTRL> + <S>`: Save ER diagram as a PNG file
- `<CTRL> + <G>`: Full screen

2 ERD Specification Language

An ERD specification consists of a sequence of one or more statements (the order of the statements does not matter). All names in ERD specification consists only of alphabets (lowercase and uppercase), underscore (_), and numbers. Similar to most variable names, you cannot begin a name with numbers.

Each statement specifies one of the following six ER Diagram constructs and must terminate with a semicolon. Before we explain each constructs, common to most of these constructs is the concept of attributes. We will first explain the different kinds of attributes.

2.1 Comments

You can add C-style multi-line comments in the code (*i.e.,* `/* comment */`).

```
/* <-- this is the start of the comment
this is the end of the comment --> */
```

2.2 Attributes

There are four different kinds of attributes, excluding a normal attributes. The set of attributes are specified enclosed within parentheses.

1. **Key Attribute:** A key attribute ends with a `*`. (*e.g.,* `sid*`).
2. **Composite Attribute:** A composite attribute is like a nested attributes. It is an attribute with additional set of attributes enclosed within round brackets. (*e.g.,* `name(first,last)`).
3. **Multivalued Attribute:** A multivalued attribute ends with a `#`. (*e.g.,* `phone#`).
4. **Derived Attribute:** A derived attribute ends with a `?`. (*e.g.,* `age?`).

The order of the attributes do not matter. Note that the parentheses (*i.e.,* `()`) must be specified even when there is no attributes (*e.g., for relationship set*).

2.3 Entity Set

An entity set is specified by the entity name (*enclosed by square brackets*) followed by the list of its attributes (*see subsection on attributes above*).

Idea: square brackets for `[rectangle]`.

2.4 Relationship Set

A relationship set is specified by the relationship name (*enclosed by angled brackets*) followed by the list of its attributes (*see subsection on attributes above*).

Idea: angled brackets for `<diamond>`.

2.5 Weak Entity Set and Identifying Relationship Set

A weak entity set is specified by the weak entity name (*enclosed by double square brackets*) followed by the list of its attributes (*see subsection on attributes above*).

The identifying relationship set is similarly *enclosed by double angled brackets.*

Idea: double square brackets for `[[double-lined rectangle]]` and double angle brackets for `<<double-lined diamond>>`.

2.6 Aggregate

A relationship set that is also an aggregation is specified similar to a regular relationship set except that the name of the relationship is *enclosed by angle brackets followed by square brackets* (*e.g.,* `[<Sponsors>]`).

Idea: angled brackets inside square brackets for `[<diamond inside rectangle>]`.

2.7 Relationship Edge

A relationship edge is specified in terms of four components:

- the entity/aggregation name
- the relationship edge annotation
- the relationship name
- an optional rolename (*enclosed by square brackets*)

2.7.1 Relationship Edge Annotation

There are four options for a relationship edge annotation for an entity/aggregation named E that is participant in a relationship named R:

- If the participation of E wrt R is partial without any key constraint, the edge annotation is `---`
- If the participation of E wrt R is partial with a key constraint on E wrt R, the edge annotation is `-->`
- If the participation of E wrt R is total without any key constraint, the edge annotation is `===`
- If the participation of E wrt R is total with a key constraint on E wrt R, the edge annotation is `==>`

2.8 ISA Hierarchy

An ISA hierarchy is specified in terms of three components:

- the superclass entity name
- the ISA hierarchy edge annotation (*to be described later*)
- the set of subclass entity names (*enclosed by curly braces and separated by commas*)
  - Note that the order of the subclass names does not matter

2.8.1 ISA Hierarchy Edge Annotation

There are four options for a relationship edge annotation for an entity/aggregation named E that is participant in a relationship named R:

- If it satisfies the overlap constraint but not the covering constraint, the edge annotation is `---`
- If it satisfies neither the overlap nor covering constraint, the edge annotation is `-->`
- If it satisfies both the overlap & covering constraints, the edge annotation is `===`
- If it does not satisfy the overlap constraint but satisfies the covering constraint, the edge annotation is `==>`

NOTE: For `---` and `-->`, we use dashed line to connect to superclass. The best practice is to have the superclass above the ISA triangle connected to the tip and the subclass below the ISA triangle connected to the sides. Unfortunately, since the arrangement is done automatically, we have no control over where the superclass and subclass are placed initially. Afterwards, you should move them to follow the best practice.

2.9 Unchecked

Currently, the ERD tool do not check for any violation in the ERD construct except for syntax error (*since syntax error is handled by the parser automatically*). You are responsible for making sure that the generated diagram does not violate any construct (*e.g., duplicate entity set, cyclic ISA hierarchies, etc*).

3 Template

The ERD tool is capable of creating a simple SQL DDL template (*i.e.,* `CREATE TABLE`). This is only a simple template that will not satisfy any constraints (*e.g., key constraints, participation constraints, domain constraints, foreign key*

constraints, etc). In fact, the template will simply produce a simple `CREATE TABLE name (...)` for each entity and relationship sets. Additionally, for each attribute, we will simply add `attribute INT`.

While this may simplify the process of starting the relational schema translation, you will still need to do work to make it correct. In particular, you will need to (*but not limited to*):

1. Change the type from `INT` to the appropriate type
2. Add `PRIMARY KEY` and `FOREIGN KEY` constraints
   - In the case of `FOREIGN KEY`, you may need to rearrange the SQL `CREATE TABLE` order such that you will only refer to tables defined above the referencing table (*i.e., you cannot refer to table that will only be defined below the referencing table*)
3. Add other constraints such as `NOT NULL`, `UNIQUE`, `CHECK (...)`
4. Merge tables if necessary (*e.g., to satisfy key constraints*)
5. Delete tables if necessary (*e.g., if the table has been merged*)
6. Rename the tables if necessary (*e.g., to make the name more meaningful in the case where the tables are merged*)

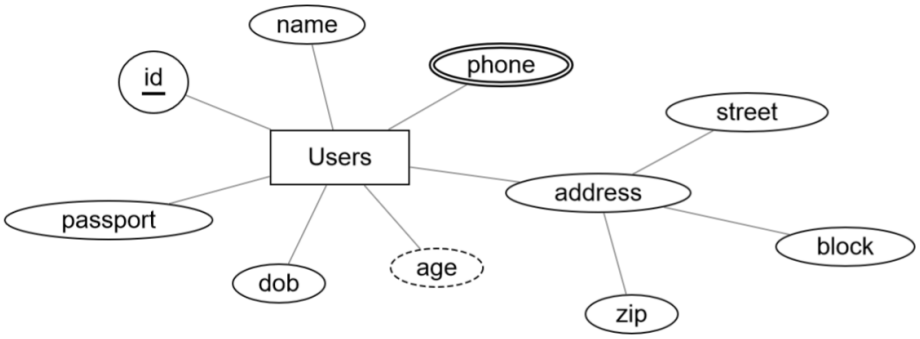
4 Examples

In all the examples below, the images are taken directly without modification. You may interact with the image to rearrange the elements.

You may also upload the provided `.erd` file corresponding to each examples. This way, you can test the tools easily. Note that the file extension does not matter as long as it is a text file.

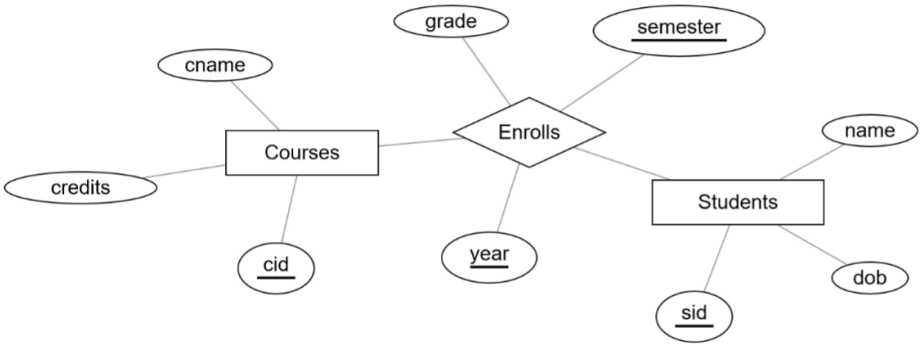
4.1 Example 1: All Kinds of Attributes

```
[Users](id*,name,passport,dob,address(zip,street,block),phone#,age?);
```



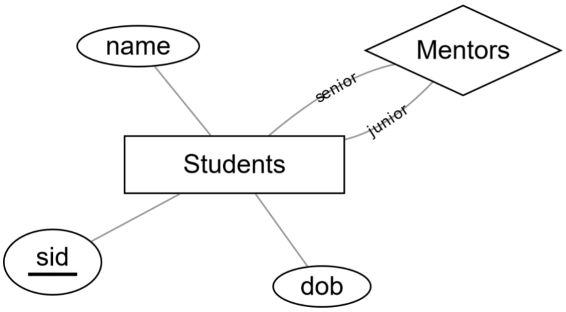
4.2 Example 2: Many-to-Many Relationship

```
[Students](sid*, name, dob);
[Courses](cid*, cname, credits);
<Enrolls>(year*, semester*, grade);
Students --- Enrolls;
Courses --- Enrolls;
```



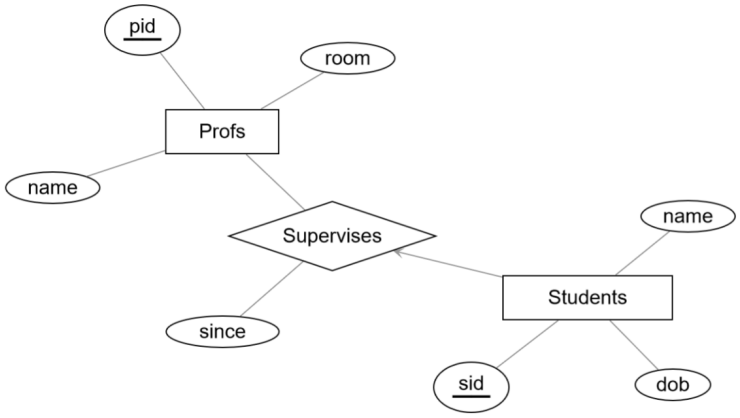
4.3 Example 3: Relationship Role

```
[Students](sid*, name, dob);
<Mentors>();
Students --- Mentors [senior];
Students --- Mentors [junior];
```



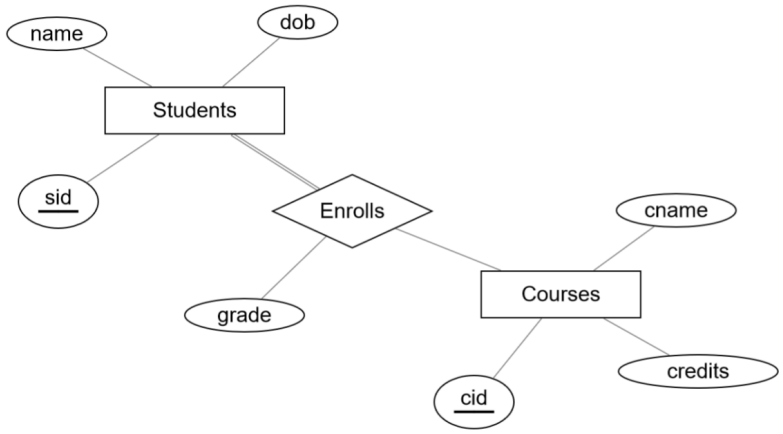
4.4 Example 4: Key Constraint

```
[Profs](pid*, name, room);
[Students](sid*, name, dob);
<Supervises>(since);
Profs --- Supervises;
Students --> Supervises;
```



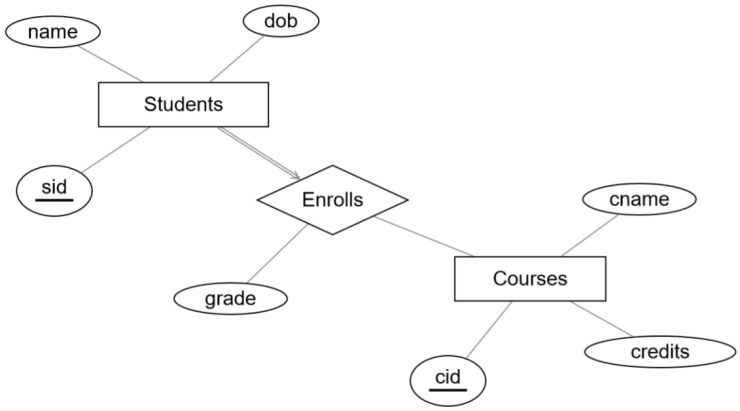
4.5 Example 5: Total Participation Constraint

```
[Students](sid*, name, dob);
[Courses](cid*, cname, credits);
<Enrolls>(grade);
Students === Enrolls;
Courses --- Enrolls;
```



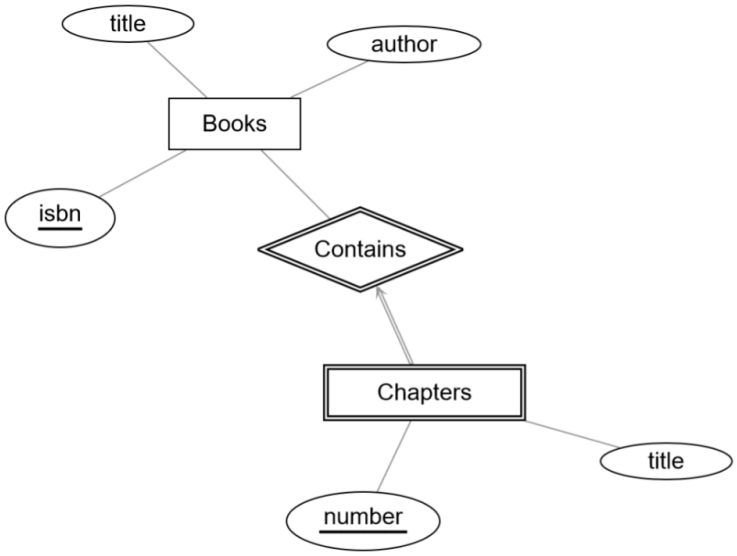
4.6 Example 6: Key & Total Participation Constraint

```
[Students](sid*, name, dob);
[Courses](cid*, cname, credits);
<Enrolls>(grade);
Students ==> Enrolls;
Courses --- Enrolls;
```



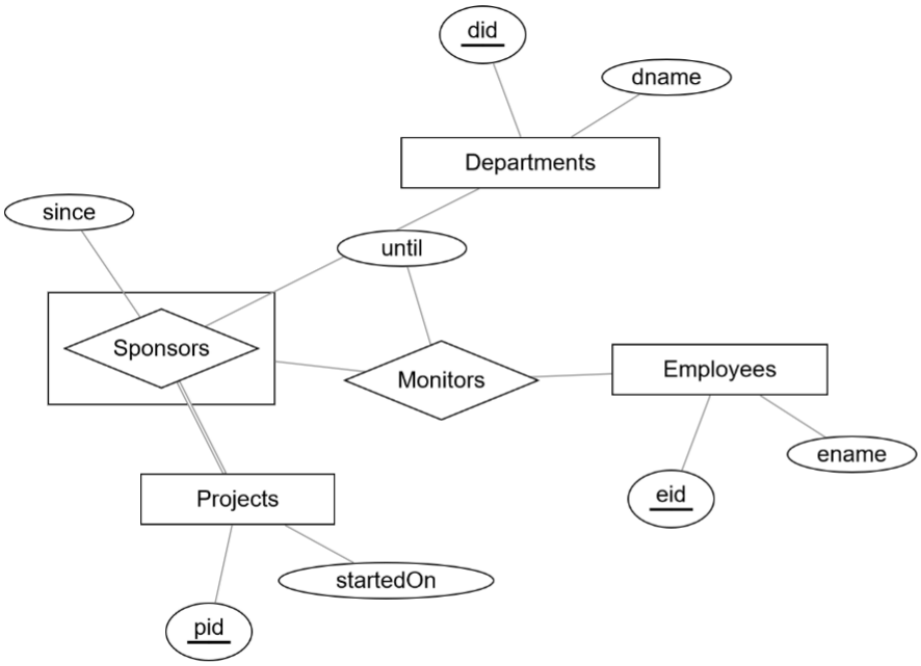
4.7 Example 7: Weak Entity Set

```
[Books](isbn*, title, author);
[[Chapters]](number*, title);
<<Contains>>();
Books --- Contains;
Chapters ==> Contains;
```



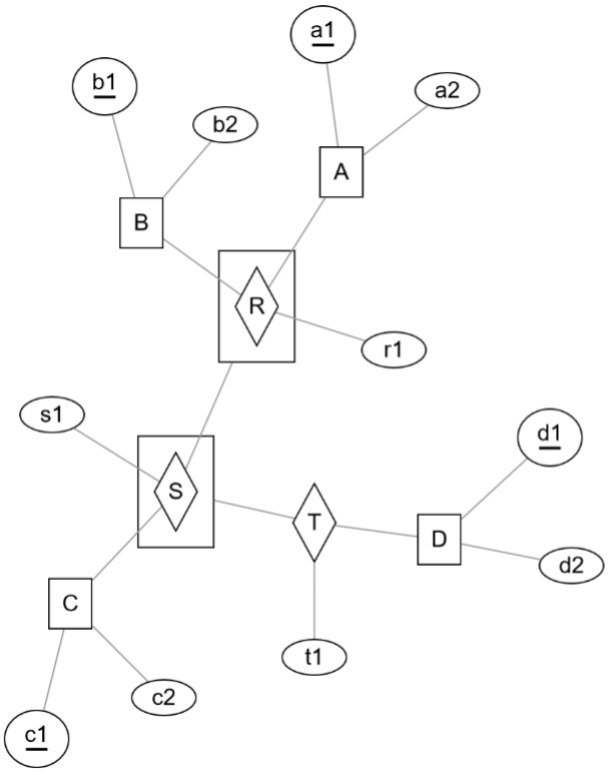
4.8 Example 8: Aggregation

```
[Projects](pid*, startedOn);
[Departments](did*, dname);
[Employees](eid*, ename);
[<Sponsors>](since);
Projects === Sponsors;
Departments --- Sponsors;
<Monitors>(until);
Employees --- Monitors;
Sponsors --- Monitors;
```



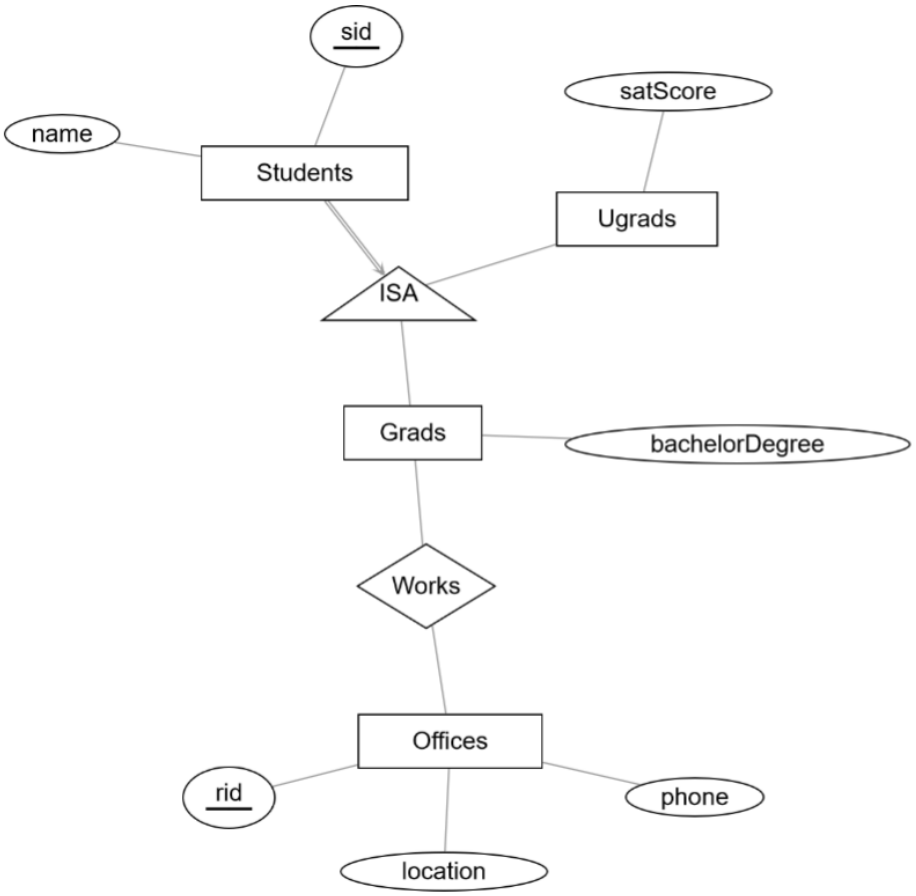
4.9 Example 9: Aggregation

```
[A](a1*, a2);
[B](b1*, b2);
[C](c1*, c2);
[D](d1*, d2);
[<R>](r1);
A --- R;
B --- R;
[<S>](s1);
C --- S;
R --- S;
<T>(t1);
D --- T;
S --- T;
```



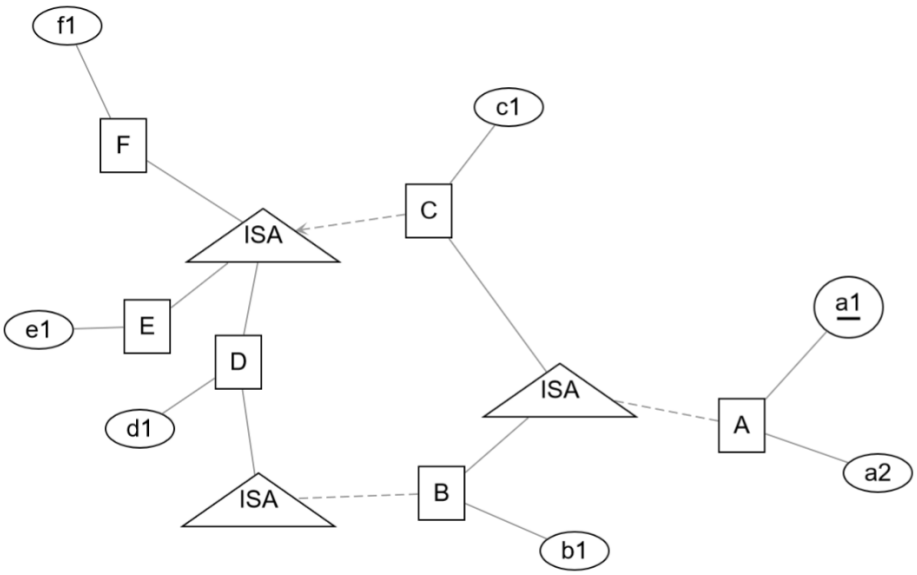
4.10 Example 10: ISA Hierarchy

```
[Students](sid*, name);
[Ugrads](satScore);
[Grads](bachelorDegree);
[Offices](rid*,phone,location);
<Works>();
Grads --- Works;
Offices --- Works;
Students ==> {Ugrads, Grads};
```



4.11 Example 11: ISA Hierarchy

```
[A](a1*,a2);
[B](b1);
[C](c1);
[D](d1);
[E](e1);
[F](f1);
A --- {B,C};
B --- {D};
C --> {D,E,F};
```



4.12 Example 12: Pharmaceutical Company

```
[Patients](ssn*, name, age, address);
[Doctors](ssn*, name, speciality, years);
[Pharmacies](address*);
[PharmaceuticalCompanies](name*, phone, address);
[Drugs](tradeName*, formula);
<PrimaryPhysician>();
<Prescribes>(date, qty);
<Sells>(price);
<Contracts>(startDate, endDate, text);
<Makes>();
Patients --- PrimaryPhysician;
Doctors --- PrimaryPhysician;
Doctors --- Prescribes;
Drugs --- Prescribes;
Pharmacies --- Sells;
Drugs --- Sells;
Pharmacies --- Contracts;
PharmaceuticalCompanies --- Contracts;
Drugs --- Makes;
PharmaceuticalCompanies --- Makes;
```