CP363: Database I

by Scott King

Dr. Siu-Cheung Chau

Chatper 1: Introduction

1 Database Management Systems (DBMS)

A *DBMS* is a collection of software to enable a user to create, maintain and utilize a DB (Ex. Oracle, MySQL, etc).

1.1 Advantages of DBMS

- Efficient data access
- Data integrity and security
- Data administration
- Data independence
- Concurrent processing crash recovery
- Reduce application development time and cost

Concurrent Processing

{figure out diagram}

2 Logical Schema of a Database

- Schema is the data structureIBM had the hierarchical model, other companies created the Network and Lodeysol models
- Logical (conceptual) scheme is the logical data structure

3 Physical Schema

- How data is stored physically: tree, heap, etc
- View on the logical schema, different ways of viewing data; sometimes public facing

4 Logical Data Independence

- Changing to logical schema will not effect the view on it

5 Physical Data Independence

- Changing to physical schema will not effect the logical schema

5.1 Lingo

{figure out diagram}

6 Data Models

- IBM had the hierarchical model, other companies created the Network and Lodeysol models

Chapter 2: Introduction to the Relational model

1 Relational DB Model

- Data is stored using tables, also known as relations
- Each column has a name, AKA attribute
- Each row is record

1.1 Schema

- Defining the structure of a relation

Example 1.

```
student_table(Id int(8),
              Name varchar(8),
              Major varchar(8),
              GPA float);
```

- Tables are defined as such: $R(A_1, ..., A_n)$
- An *instance* is a record in a table

A **database instance** is all the data in the DB. **Database schema** contains schemas for all tables in the form $S = R_1, ...R_m$.

The table must satisfy four constraints:

1. Domain constraint
2. Key integrity constraint
3. Entity integrity constraint
4. Referential integrity constraint

Example 2.

Eid	Name	Dependent
1234	*Tom*	Mary
1234	*Tom*	Peter
2345	*John*	Tim
2345	*John*	Jane
3456	David	Jim

Remember **NO** duplicates. Instead change those records to:

Eid	Name	Dependent
1234	*Tom*	Mary,Peter
2345	*John*	Tim,Jane
3456	David	Jim

Each attribute can only contain *atomic* values. Ex. Cannot contain a set of values. FFS, do this:

Eid	Name	Dependent 1	Dependent 2
1234	*Tom*	Mary	Peter
2345	*John*	Tim	Jane
3456	David	Jim	--

This is also **NOT** ideal.

R_1:

Eid	Name
1234	*Tom*
2345	*John*
3456	David

R_2:

Eid	Dependent
1234	Mary
1234	Peter
2345	Tim
2345	Jane
3456	Jim

1. Key Integrity Constraint

- Observation: No tuples in a table are the same; in other words, every tuple in a table needs to be unique..SO DONT STORE RECORDS TWICE
- $\Rightarrow$ Each record, tuple, can be <u>uniquely identified</u> by a certain set of attributes
- The *super key* is a set of attributes that can uniquely identify a record (tuple)
- The trivial super key is the one with all the attributes
- A minimal super key is called a *candidate key*
- A candidate key be picked as the *primary key* $\rightarrow$ Refer to next Example 3
- Every relation must have a primary key `student_table(Id,Name,Major,GPA)` where `Id` is the primary key

Example 3. `id,dept_name` are super keys but not candidate keys. But two candidate keys are `id,Name` as they are unique. Then pick on candidate to be the primary, this `Id` will be picked as the primary key.

Example 4. {table}

We have the minimal super key = candidate key. $\{A, D\}$ is not a super key. $\{B, C\}$ is a super key. Is it a candidate key? No, thus, C is a super key $\Rightarrow C$ is a candidate key $\Rightarrow C$ is a primary key

2. Entity Integrity Constraint

- The primary key cannot be `null`; in our example, that means `Id` cannot be null
- The rest of the attribute can be `null`

3. Referential Integrity Constraint

- A *foreign key* must either contain a `null` value or a value in the **referenced key**

Example 5. Back to the example with R_1 and R_2. We have `R1(Eid,Name)` and `R2(Eid,Dependent)`.

Thus, `Eid`, is an foreign key in R_2 because `Eid` is the primary key in R_1. `Eid` in R_1 is the referenced key.

Chapter 7: DB Design

1. Requirement Analysis

- Data to be stored
- Applications
- Performance analysis

Example 6. Registration system

```
student records
class records
professor records
```

Actions:

- To register a student in the class
- Print class list
- Print professor schedule

2. Conceptual Database Design

- Entity-Relationship model (ER model)
- Entity is equivalent to record type
- The relationship between entities

3. Logical Database Design

- Convert the ER model into a relational model (ie. convert to tables - relations)

4. Scheme Refinement

- Theory of relational database - 1^{st}, 2^{nd}, 3^{rd}, 4^{th} BC (Boyce-Codd) normal forms

5. "Physical" Database Design

- Add index files to speed up certain applications

6. Application and Security Design

1 Entity-Relationship Model

Entity $\equiv$ record

- An entity is an object in the real word that is distinguishable from other objects

- A **diamond** indicates a relationship between two entities
- A **triangle** indicates a subclass of
- A **circle** represents an attribute
- A **rectangle** represents an entity

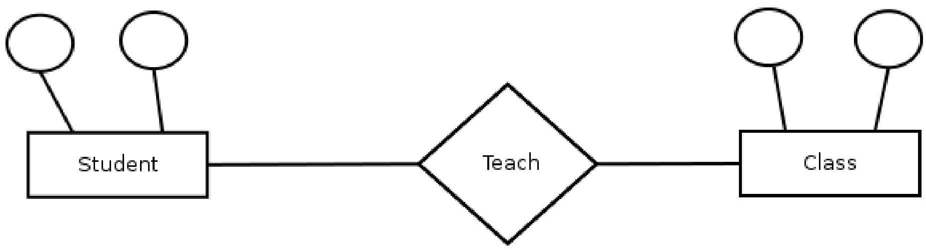
Example 7. A student record

These are attributes

`Id      Name      GPA      Major`

- An entity is a collection of entities of the same type
- All entities in the same set have the same attributes (cicles)
- *Student* is the entity and *{Id, Name, GPA, Major}* are the attributes (circles)
- Each entity has a primary key
- The key should depend on a real life situation and not the current set of data

2 Relationship Between Entities



2.1 Many to Many Relationship

1. A student can take many courses
2. A class can have many students

2.2 1 to Many Relationship

1. A professor can teach many classes
2. A class is taught by 1 professor

2.3 1 to 1 Relationship

1. A department has only one chair
2. A professor is the chair of only one department

Note: Every entity has a key and every relationship has a key.

A relationship does not have to be binary.

2.4 Total Participation

- Everyone is involved ⇒ total participation (**thick line**)

2.5 Recursive Relationship

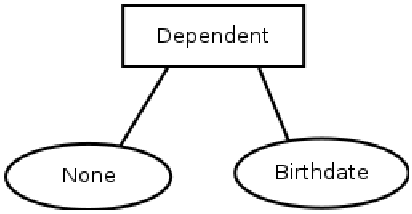
- An entity can only appear once in a design
- The key for the relationship (Id) - in our example

2.6 Weak Entity

- Entity without superkey

Note: Thick [box] lines implies weak entity.

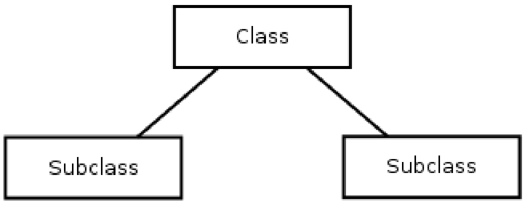
Example 8.

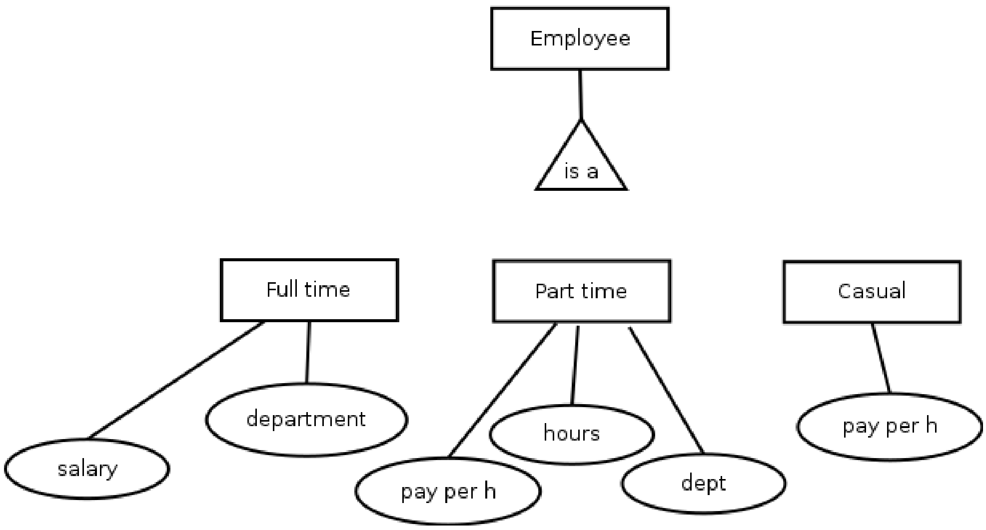


2.7 Strong Entity

- Entity with a super key (primary key)
- Everything so far is a strong entity

3 Class Hierarchies





The subclasses are disjoint.

4 Aggregation

- Abstraction to group relationships

Refer to customer - borrower - loan.

5 Issues in Conceptual Design Using the ER Model

1. Should a concept be modeled as an attribute or as an entity

    - Refer to employee one
    - Employees can have multiple addresses (sub address with employee to create new entity with attributes)
    - If we wanted to conduct searches in the address - it would be easier with address as an entity (ex. search for a city within a country)
    - Refer to picture - it is wrong, solution: look to next picture (make from and to as entity instead of attribute)

6 ER to Relational Database Mapping

Step 1

For each strong entity set, E, create a table (relation), R, that includes all attributes of E. The key for the entity will be the key for the table (relation).

$$\text{strong entity} \Rightarrow \text{table}$$

Step 2

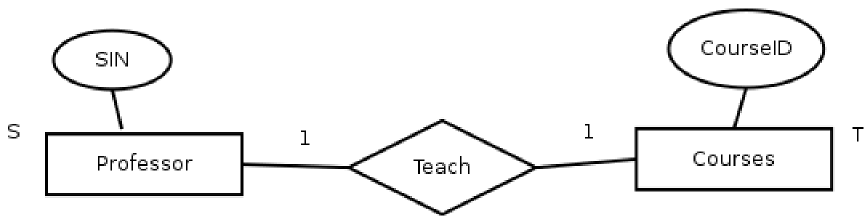
For each weak entity set, W, create a table, R, that includes all attributes of W and include it as a foreign key, the primary key of the owner entity. The key for the relation is the partial key and the key of the own entity.

{table for employee-dependents}

Step 3

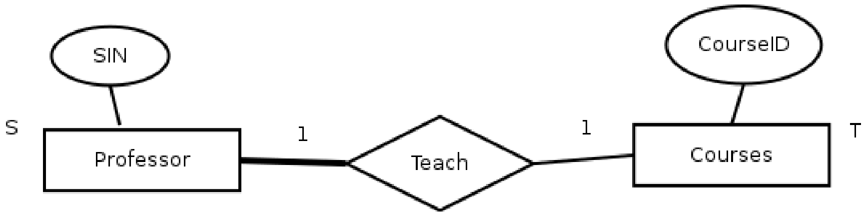
For each 1:1 relationship between two strong entitie, S and T. **No** new table (relation) is required. Just add the primary key of S as a foreign key in T.

Example 9.




```
professor(SIN, ..., ...)
courses(courseID, ..., ..., SIN)
```

If total participation exists in $\underline{S}$, we should add the primary key of T as a foreign key to $\underline{S}$.



Every professor much teach one course and the course cannot be taught by more than one professor.

Step 4

For each 1:n relationship between two strong entities S and T. No new table is required. Just add the primary key of S as a foreign key in T.

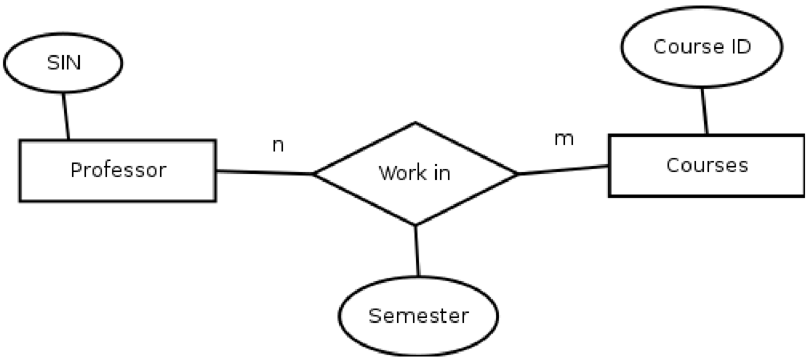
Example 10.

`Employee(id, ..., ..., departmentID)`

Step 5

For each many to many relatonship between S and T, create a new table (relation) that contains the primary key of S and T. The key for the new table is the primary key of both S and T together.

Example 11.

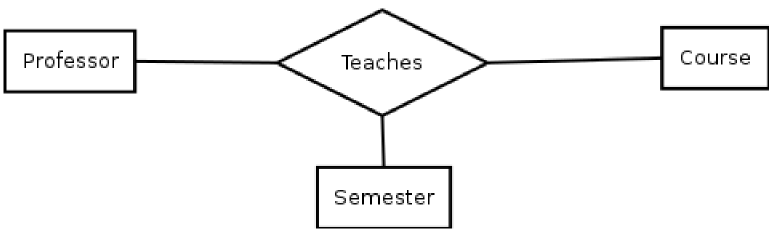


`Teaches(SIN, CourseID, semester)`

Step 6

n-ary relationships.

Example 12.

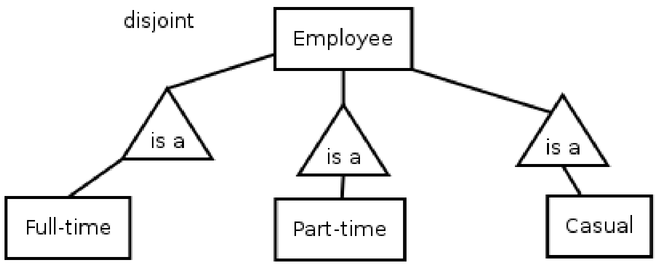


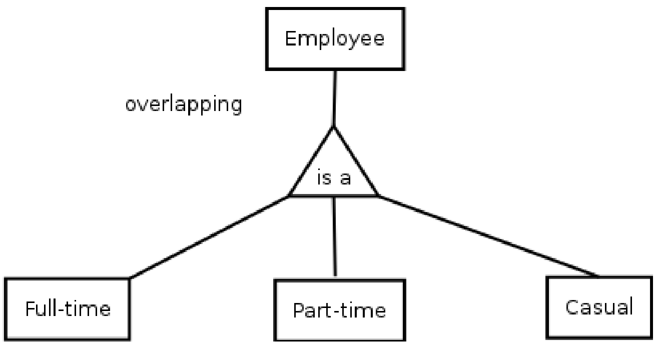
Teaches(SIN, semester_id, course_no)

Create a new table that contains the primary key of all the entities. The keys for the table are the primary keys of all entities together.

Step 7

Translating class hierarchies.





Disjoint implies that we create a table for each disjoint entity and it is necessary to create one for the parent. **Overlapping** implies that we create a table for all overlapping entities and create a table for the parent; only put the key of the parent in table of the subclass.

Step 8

Aggregates.

Create a table for an aggregate that contains the key for the aggregate and the key for the entity.

Example 13.

Supervise(SIN,Graduate_SIN, p_no) all with dotted underlines

7 Data Definition Language (DDL) - part of SQL

Example 14. Create a table in SQL

Employee(Eid,name,address,salary,Supervisor) - last one is dotted
Supervisor(Eid, S_ID) both dotted

Here's the code:

```
CREATE TABLE   Employee
(Eid           integer(9),      // or int(9)
```

```
 Name          varchar(30),
 address       varchar(50),
 salary        integer(5) unique,    // CANNOT BE null
 Sid           integer(9),
 primary key   (Eid)
 foreign key   (SupervisorID) reference (Employee))

CREATE TABLE   Supervisor
(Eid           int(9),
 Sid           int(9),
 primary key   (Eid,Sid),
 foreign key   (Eid) reference (Employee),
 foreign key   (Sid) reference (Employee))
```