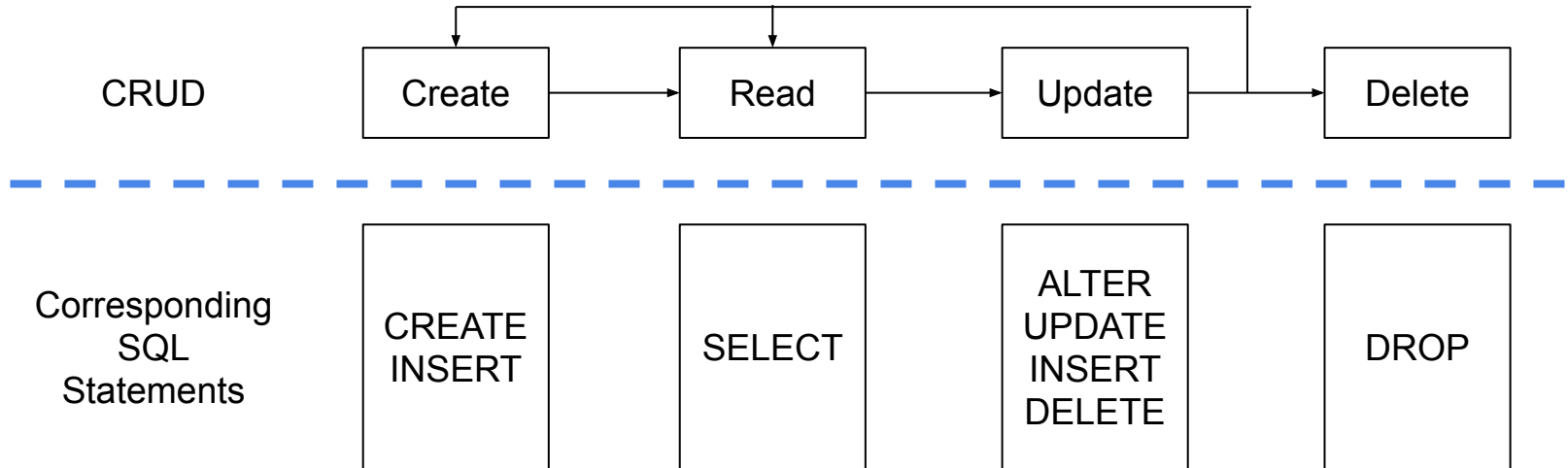


Simple INSERT

Data Lifecycle

How data exists in computer systems: Create, Read, Update, Delete (CRUD)

- Generic lifecycle of data in ANY computer system, not SQL specific.



Simple INSERT Statement

- Insert data with a column list

```
INSERT INTO <table_name> (<column_list>)  
VALUES (<value_list>)
```

- Keyword: INSERT INTO
- Followed by table name and column names, enclosed in parenthesis
- Followed by VALUES keyword and values to be inserted, enclosed in parenthesis

Column List in INSERT Statement

- Columns don't need to be in the same order as table definition. But column list and value list must match with each other:

```
INSERT INTO Instructor  
  (Instructor_Affiliation, Instructor_Name)  
VALUES ('Jedi', 'Obi-wan Kenobi')
```

INSERT without Column List

- Insert without column list: default to all columns in the table, following the order defined in table CREATE statement.

```
INSERT INTO <table_name>  
VALUES (<value_list>)
```

```
INSERT INTO Instructor  
VALUES ('Obi-wan Kenobi', 'Jedi')
```

Data Compatibility

- By default, you can only insert without data loss
 - VARCHAR(100) -> VARCHAR(200): OK
 - VARCHAR(200) -> VARCHAR(100): Will have issue
 - INT -> Decimal: OK
 - Decimal -> INT: Will have issue
- Can manually convert.
 - Will cover this conversion in future lectures

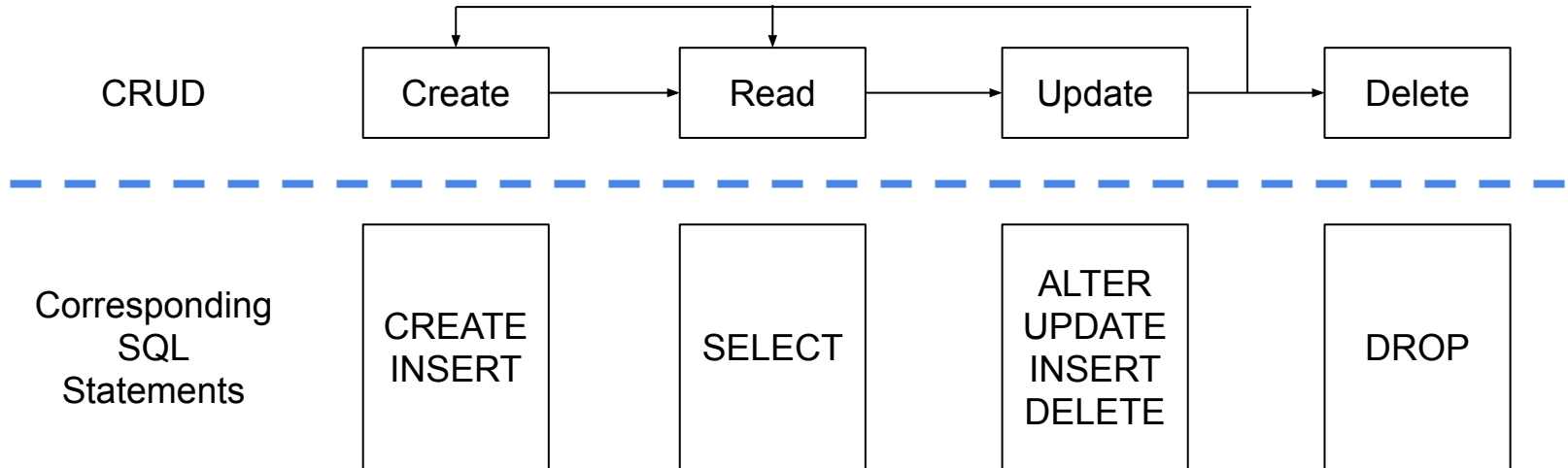
Single Table

SELECT Statement

Data Lifecycle

How data exists in computer systems: Create, Read, Update, Delete (CRUD)

- Generic lifecycle of data in ANY computer system, not SQL specific.



Simple SELECT Statement

- Show selected data from table

```
SELECT <column_name>, <column_name> ...  
FROM <table_name>
```

- Keyword: SELECT
- Followed by column names, separated by comma.
 - No comma after last column.
- Keyword: FROM, followed by table name

Example SELECT Statement

- Select columns: No need to be in the same order as CREATE. Will show up in the order as selected.

```
SELECT City, Student_Name FROM Student
```

SELECT with *

- Select all columns: Asterisk (*) means all columns, default order as defined in table CREATE statement.

```
SELECT * FROM <table_name>
```

```
SELECT * FROM Instructor
```

- Can mix * and column names. Must have comma in between.

```
SELECT Instructor_Name, * FROM Instructor
```

SELECT with Calculation

- Retrieving certain column computations

```
SELECT Quantity * Price FROM Sales
```

SELECT Statement: Column Alias

- You can rename a column or specifying the name for an expression using AS:

```
SELECT <column_computation> AS <column_alias>  
FROM <table_name>  
WHERE <boolean_expression>
```

```
SELECT Quantity * Price AS TotalSales  
FROM Sales
```

WHERE Clause

- Use WHERE to identify a subset of data which meets a certain condition (filtering condition)

```
SELECT * FROM <table_name>  
WHERE <boolean expression>
```

- Add WHERE keyword after table name, followed by condition

```
SELECT * FROM Instructor  
WHERE Instructor_Affiliation = 'Jedi'
```

SELECT Execution

- When running SELECT, DBMS will go through each row
 - What is the result of the following query?

```
SELECT 'This is a test' FROM Instructor
```

- How will understanding this help us understand WHERE clause?
 - DBMS goes through each row and apply SQL action (such as SELECT) to the rows that meet the filtering condition

Executing WHERE Clause

```
SELECT * FROM Instructor  
WHERE Instructor_Affiliation = 'Jedi'
```

Instructor	Instructor Affiliation
Obi-wan Kenobi	Jedi
Darth Sidious	Sith

General SQL Execution

- DBMS will go through each row and test if the row meets the filtering condition
 - If yes, apply SQL action (such as SELECT) to the row
 - Will see more examples in the future
- Rows in table (instances in a set) does not have an order. Execution can happen to any row at any time.
 - Ideal for distributed processing

SELECT DISTINCT

- Make sure each row value only show up once
 - Each column value can repeat, but the combined value of the row will only show up once.
 - `DISTINCT` is a keyword, no parenthesis or quotes.

```
SELECT DISTINCT Course_Name, Instructor_Name  
FROM Course
```

Sorting: ORDER BY Clause

- Data returned by SELECT is not sorted
 - If you have been seeing sorted results, that's a coincidence
 - If you want the data to be sorted, use ORDER BY clause

```
SELECT DISTINCT Instructor_Name  
FROM Course  
ORDER BY Instructor_Name
```

- Keyword `ORDER BY`, followed by columns to sort the dataset.

Sorting: ORDER BY Clause

- Sorting order: Descending (DESC)/Ascending (ASC, default)

```
SELECT Instructor_Name, Course_Name  
FROM Course  
ORDER BY Instructor_Name ASC
```

Sorting: ORDER BY Clause

- Always at the end of the SELECT statement
- Can use any columns in the dataset, not necessarily in the SELECT clause
- Can have multiple columns, and the order of columns determines sorting priorities

```
SELECT Instructor_Name, Course_Name  
FROM Course  
ORDER BY Instructor_Name ASC, Course_Name DESC
```

Sorting: ORDER BY Clause

- Can use the position of columns in the SELECT clause instead of column names.

```
SELECT Instructor_Name, Course_Name  
FROM Course  
ORDER BY 1 ASC, 2 DESC
```

SQL Demo

SQL Convention in PostgreSQL

- Running multiple statements
 - Highlighting
 - Semicolon (;)

SQL Convention in PostgreSQL

- SQL Statements
 - Keywords and table/column names are not case sensitive.
 - Line breaks and extra spaces are ignored.
 - But still, try to form a convention
 - Using upper case for keywords.
 - Proper indentation

Physical Implementation of Relational Database

Relation as a 2D table

- Relation consists of Tuples. All tuples in a relation have same attributes, in the same order. As such, a relation can be simply displayed as a 2 dimensional table

Code	Name	Credit	Date/Time
DS530	Big Data and Data Management	3 Credits	Wed 6-10pm
DS620	Data Visualization	3 Credits	Tue 6-10pm

Relation

Relational model makes defining datasets simple

- Each dataset can be represented with a series of relations
- Each relation is a 2-dimensional table
 - Note this is the “conceptual table”, not the “database table” that we will discuss in a minute.

Database Implementation

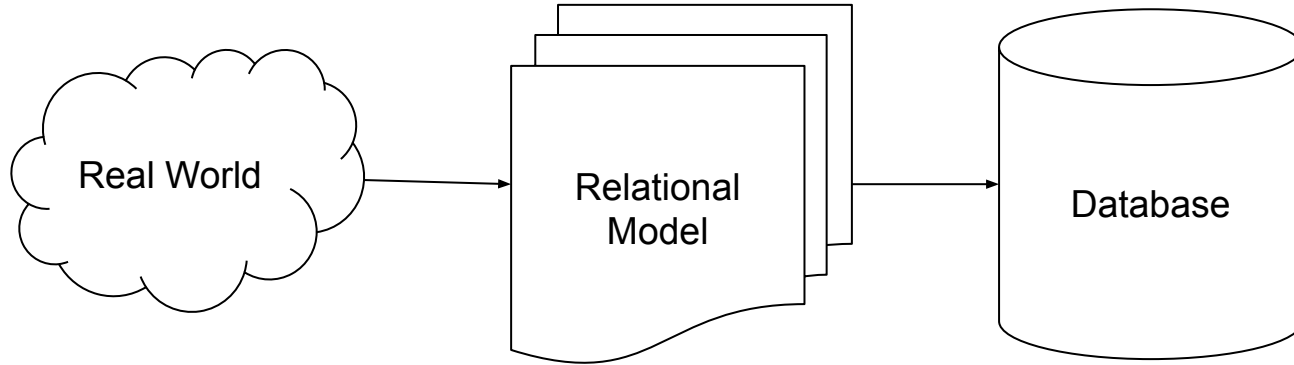
- Data is organized into a series of database tables. Each table corresponds to one relation and thus stores data as a 2 dimensional table.
 - From now on when we say “table” it is always database table.
- Database that is based on Relational Model is called Relational Database
 - DBMS for Relational Database is called RDBMS

Relational Database Design

- How to create a relational database?
 - Conceptual/Logical Design: Through a systematic process of database modelling, we map real world questions into a logical model which contains a collection of relations. The result is a Logical Model.
 - Physical Design: Map the relations in the Logical Model into database objects such as tables. The result is a Physical Model.

Relational Model

- Introduces a mathematical layer between real world problems and computer implementations



Logical Modeling

- Each relation represents a real world “thing”: Will cover in detail later
- Each tuple represents a real world instance of that “thing”
- Each attribute represents a characteristics of that instance

Mapping a thing/instance/characteristics into relation/tuple/attribute is called “logical modeling”

Defining a relation = describing the attributes of the relation.

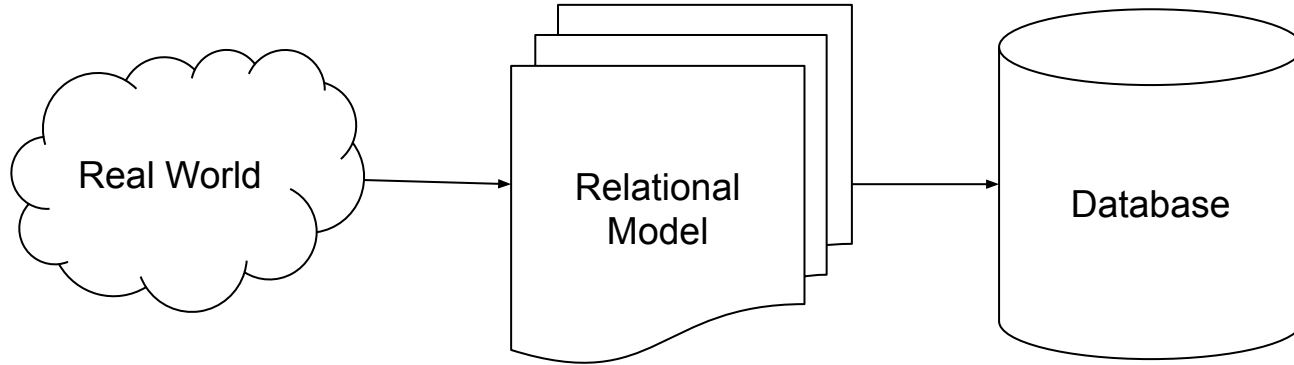
Populating a relation = inserting instance into the relation

Relational Model Example

- Relation: Course
- Attribute: Each element in the tuple (Course code, Name, Credit, Time)
 - All tuples in a relation have same attributes, in the same order.
 - Each attribute has its own data type: for better processing of data
- Tuple (defined sequence of elements):
 - DS530, Big Data and Data Management, 3 Credits, Wed 6-10pm
 - DS620, Data Visualization, 3 Credits, Tue 6-10pm

Relational Model

- Relational model can be systematically converted to database implementation



Physical Mapping

- Each table represents a “relation”
- Each row represents a tuple, which is a real world instance
- Each column represents an attribute

Defining a table = describing the columns (attributes) of the table.

Populating a table = inserting row (instance) into the table

Synonyms

Mathematical	Logical	Physical
Relation	Conceptual table	Database table
Tuple	Instance	Row or Record
Attribute	Attribute	Column or Field

Importance of Relational Model

- Why is it important: Relational Model can be systematically, consistently, and logically represented in databases.
 - Has mathematical theory behind it (Relational Algebra)
 - Simple to develop and implement
 - Consistent in results across different platforms
 - Can be applied to almost all business scenarios

Logical Design and Entity Relationship Model

Simplest Relational Database

- Excel spreadsheet

Course DS530 Gradesheet

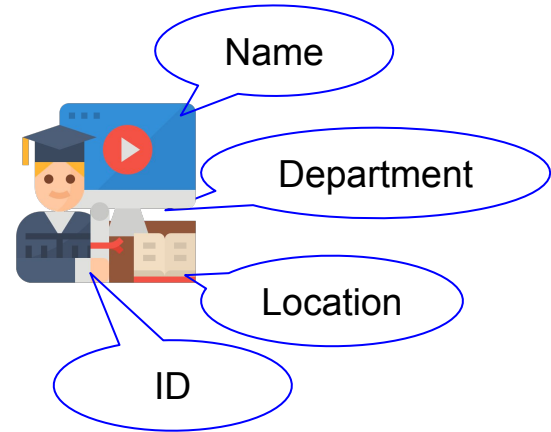
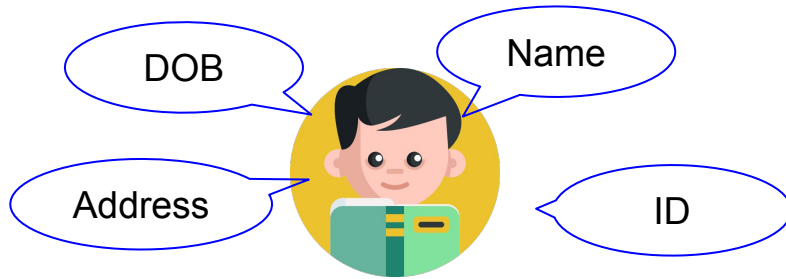
Student ID
Student Name
Attendance 1
Attendance 2
:
Assignment 1
Assignment 2
:
Final Score
Project Score
Total Score
Final Grade

Entity Relationship Model

- Most popular logical design model is Entity Relationship Model
 - Divides relations into two categories: entity and relationship
 - Provides a systematic way of mapping real world objects into entity/relationship
 - Defines how they interact with each other.

Entity

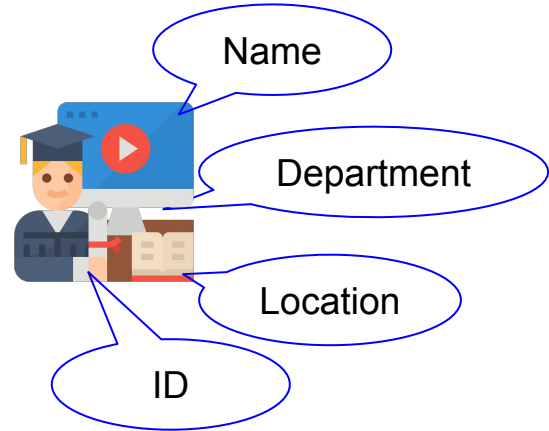
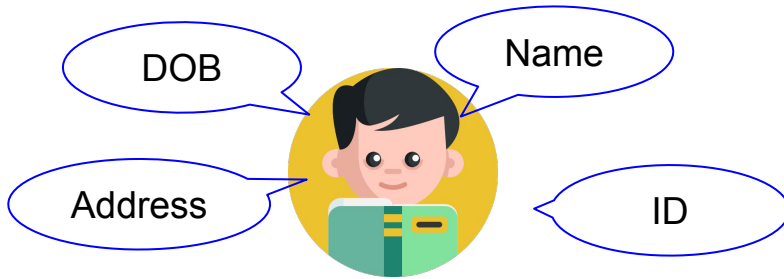
- Entity: a noun; a person, place, or thing, such as a Student, Course, Instructor, etc..



Entity

Student (ID, Name, DOB, Address)

Course (ID, Name, Department, Location)



Relationship

- Relationship: the association between entities. Usually a verb.
 - Created by putting two entities together

Registration (Attributes from Student, Attributes from Course)

Registration (
 Student.Name, Student.ID, Student.DOB, Student.Address,
 Course.ID, Course.Name, Course.Department, Course.Location
)

Relationship Attribute

- Relationship can have attributes too

Registration (

**Student.Name, Student.ID, Student.DOB, Student.Address,
Course.ID, Course.Name, Course.Department, Course.Location,
Registration.Status, Registration.Grade, ...**

)

Representing Relationship

- Relationship: putting two entities together
 - The primary keys of both entities must be included in the relationship definition. The combination of these primary keys become the primary key of relationship.
 - Registration: Student + Course

Registration (Student.ID, Course.ID)

- Plus its own attributes (Descriptive Attributes)

Relationship Cardinality

- Different types of relationships
 - Many-to-many: Student - Course
 - One-to-many: Department - Course
 - One-to-one: State - State Capital
- How is relationship determined: Business Research and Data Discovery

Min/Max Cardinality

- One: can it be zero?
 - A course must come from a department (Department 1-1...M Course)
 - A customer may not belong to any TV cable company (Company 0...1-1...M Customer)
- Many: can it be zero? Is there a limit?
 - Each student can sign up multiple courses, or none (1-0...M)
 - Each student must sign up at least one course (1-1...M)
 - Each student must sign up at least one, up to five courses (1-1...5)

Relation vs Relationship

- Name is similar but meaning is different. Can be really confusing.
 - Relationship is a type of Relation.
 - But Relation also includes Entity.
- Entity Relationship Model is an enhanced version of Relational Model, and the most popular one.

Entity Relationship Diagram (ERD)

Diagrams for database design

- Entity-Relationship Diagram (ERD)
 - Entity => Rectangle
 - Relationship => Line
 - Use line style or line markers to represent relationship type: 1:1, 1:M, M:M

Chen's Diagram

- Diamond shape diagram
 - Relationships are displayed as diamonds between entities (

Chen, Peter (March 1976). "The Entity-Relationship Model - Toward a Unified View of Data". *ACM*

Transactions on Database Systems. 1 (1): 9–36.)

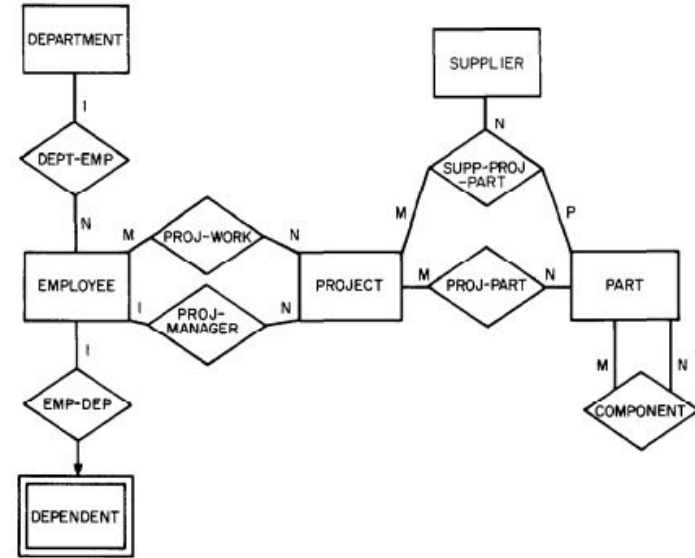
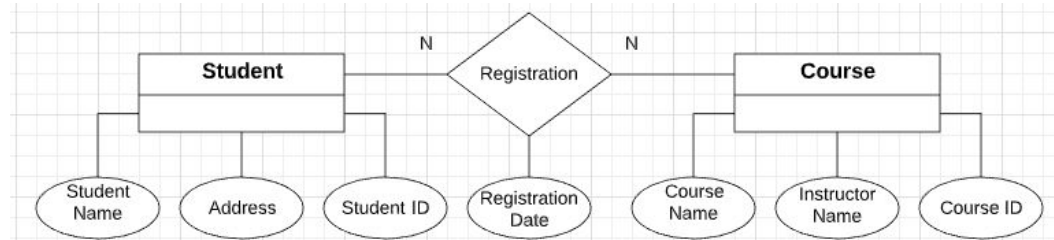


Fig. 11. An entity-relationship diagram for analysis of information in a manufacturing firm

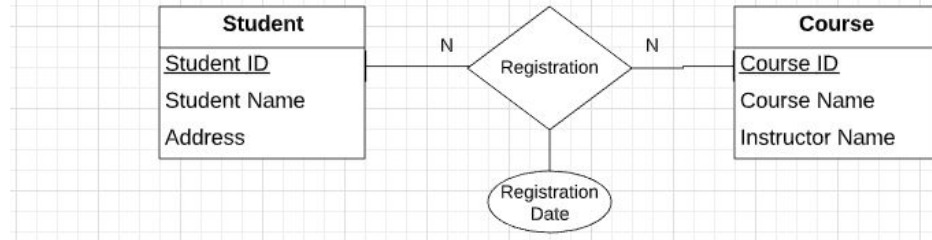
ACM Transactions on Database Systems, Vol. 1, No. 1, March 1976.

Example Chen's Diagram

Original:

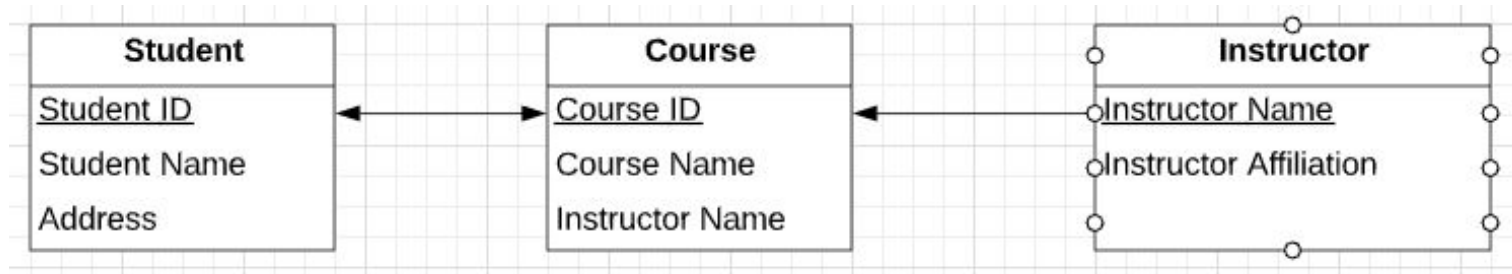


One variant:




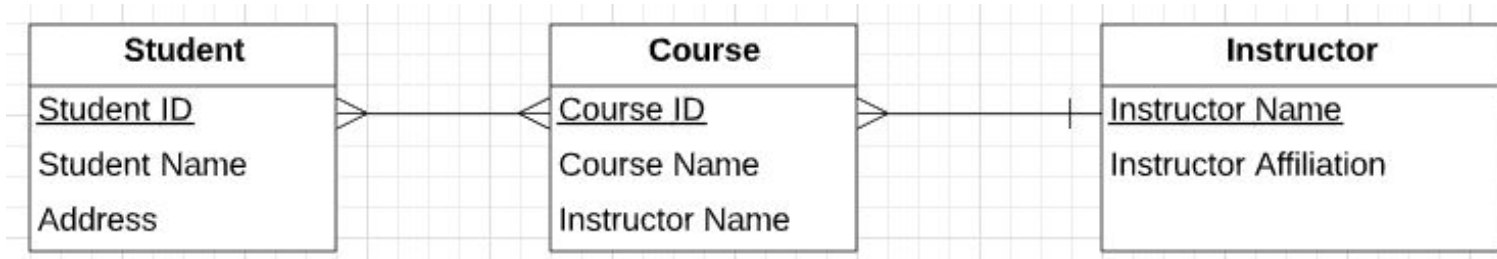
Arrowed Diagram

- Use arrow to indicate relationship type.
- Pay attention to the direction of arrows.
 - Many books use \leftarrow to represent 1:M
 - Some use \Rightarrow for 1:M.



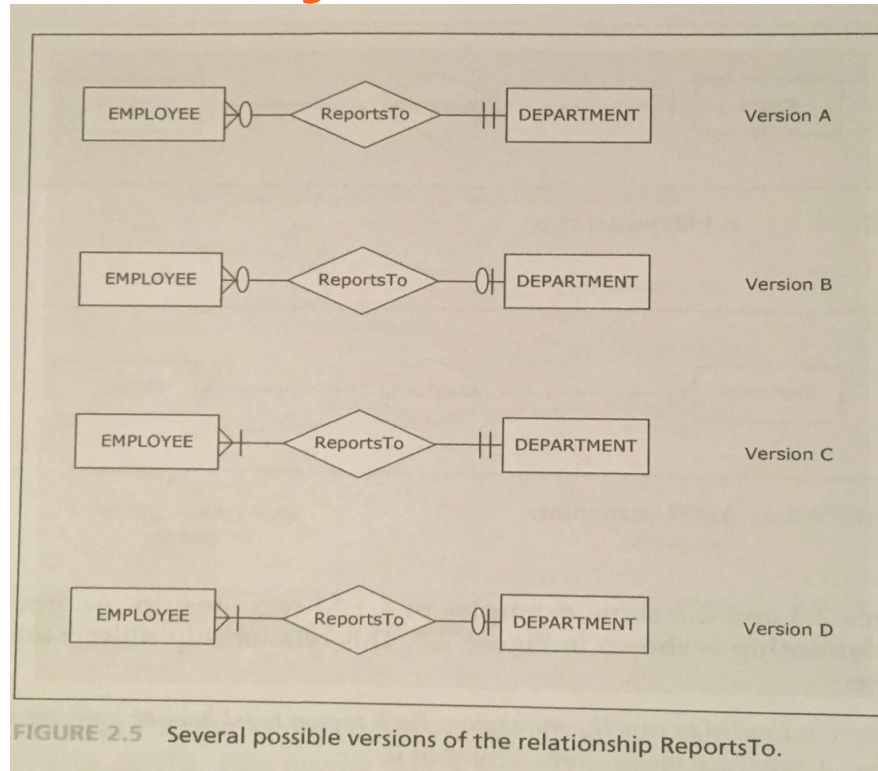
Crowfoot Diagram

- Use crowfoot  to indicate the relationship type.
 - That's the diagram style we will use.
 - Relationship attributes: insert rectangle to represent relationship.



Cardinality in Crowfoot

P16 Textbook



0...M - 1...1

0...M - 0...1

1...M - 1...1

0...M - 1...1