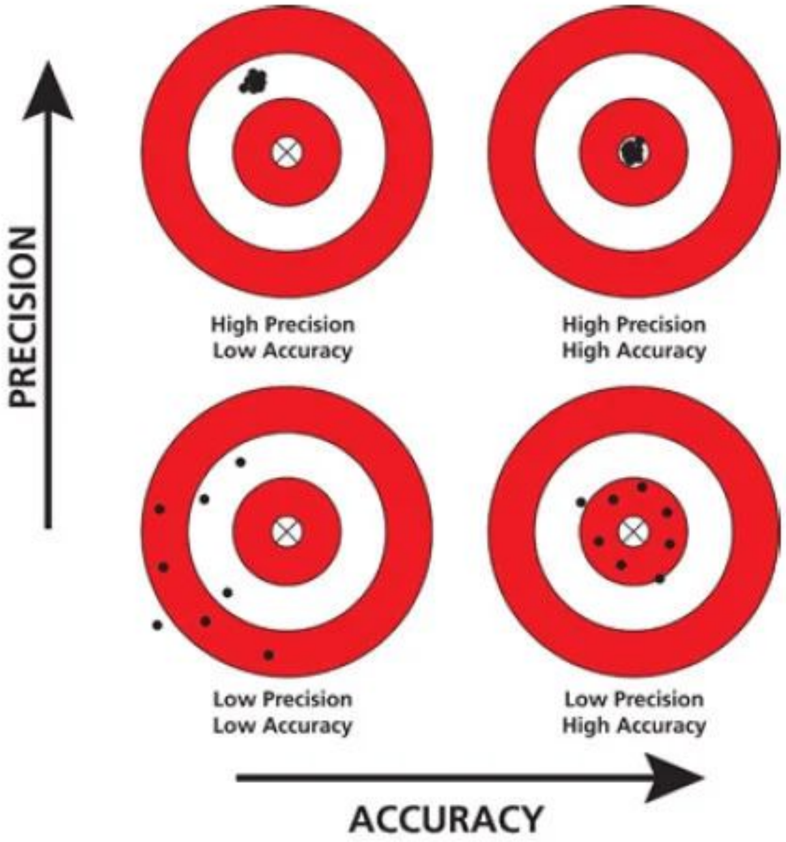
Databases and
Web Application Development UG2

Designing Databases: Databases as
Data structures

Lecture

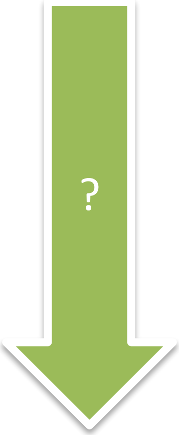
PLANNING A DATABASE DESIGN

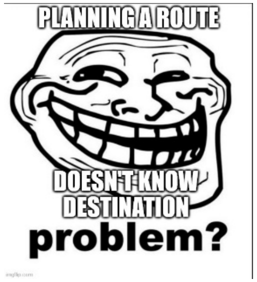
- Accuracy
  - the degree of closeness to true desired value
- Precision
  - the degree to which an process will repeatable result in the same value.



Planning and Designing

Birmingham





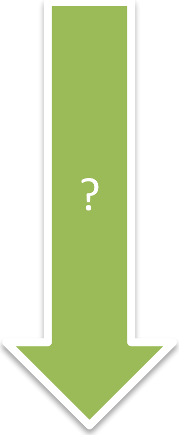
London

- Like planning a route
  - What is the **specific** starting point?
  - What is the **specific** destination?
  - What are the constraints on how to get there?
- In Science and Engineering we want to be as **specific (accurate** and **precise)** as possible
  - Formulating problems / questions
  - Formulating solutions / answers
- Implement your Software before you have a Plan?
  - No clear starting point?
  - No clear destination?
  - **No route!**

Planning and Designing

4 Cardigan St,
Birmingham
West Midlands
England
B4 7BD

United Kingdom?
Europe?
Earth?



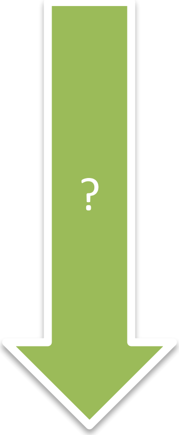
London

- Like planning a route
  - What is the **specific starting point**? ✓
  - What is the specific destination?
  - What are the constraints on how to get there?

Planning and Designing

BCU Cardigan Building
4 Cardigan St,
Birmingham
West Midlands
England
B4 7BD

United Kingdom?
Europe?
Earth?



LSE Cowdray House
6 Portugal St,
London,
Greater London,
England
WC2A 2HJ

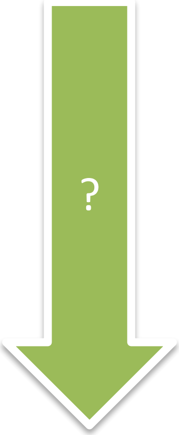
United Kingdom?
Europe?
Earth?

- Like planning a route
  - What is the **specific starting point**? ✓
  - What is the **specific destination**? ✓
  - What are the constraints on how to get there?

Planning and Designing

BCU Cardigan Building
4 Cardigan St,
Birmingham
West Midlands
England
B4 7BD

United Kingdom?
Europe?
Earth?



LSE Cowdray House
6 Portugal St,
London,
Greater London,
England
WC2A 2HJ

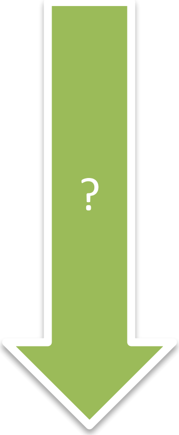
United Kingdom?
Europe?
Earth?

- Like planning a route
  - What is the **specific starting point**? ✓
  - What is the **specific destination**? ✓
  - What are the **constraints** on how to get there?
    - ~~By Air?~~
      - ~~Helicopter?~~
      - ~~Plane?~~
    - ~~By River?~~
      - ~~Boat?~~
      - ~~Hovercraft?~~
    - By Road?
      - Car?
      - Walking?
    - ~~By Train?~~
      - ~~Heavy rail locomotive?~~
      - ~~Light metro trail “The tube”?~~
      - ~~Walking?~~

Planning and Designing

BCU Cardigan Building
4 Cardigan St,
Birmingham
West Midlands
England
B4 7BD

United Kingdom?
Europe?
Earth?



LSE Cowdray House
6 Portugal St,
London,
Greater London,
England
WC2A 2HJ

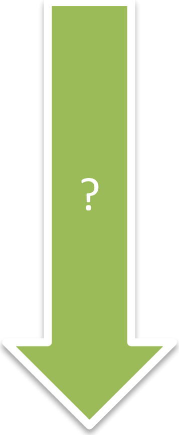
United Kingdom?
Europe?
Earth?

- Like planning a route
  - What is the **specific starting point**? ✓
  - What is the **specific destination**? ✓
  - What are the **constraints** on how to get there?
    - By Road?
      - Exclusive
        - » Car?
        - ~~» Walking?~~
      - ~~Mixture?~~
        - ~~» Car?~~
        - ~~» Walking?~~

Planning and Designing

BCU Cardigan Building
4 Cardigan St,
Birmingham
West Midlands
England
B4 7BD

United Kingdom?
Europe?
Earth?



LSE Cowdray House
6 Portugal St,
London,
Greater London,
England
WC2A 2HJ

United Kingdom?
Europe?
Earth?

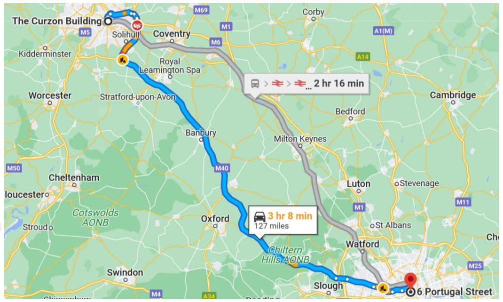
- Like planning a route
  - What is the **specific starting point**? ✓
  - What is the **specific destination**? ✓
  - What are the **constraints** on how to get there?
    - By Road
      - Exclusive
        - Car
    - Anything else?

Planning and Designing

BCU Cardigan Building
4 Cardigan St,
Birmingham
West Midlands
England
B4 7BD

United Kingdom?
Europe?
Earth?

Route
Plan



LSE Cowdray House
6 Portugal St,
London,
Greater London,
England
WC2A 2HJ

United Kingdom?
Europe?
Earth?

- Like planning a route
  - What is the specific **starting point**? ✓
  - What is the specific **destination**? ✓
  - What are the **constraints** on how to get there? ✓

Requirements

- What
  - Does it represent?
  - etc.
- Why
  - Is it needed?
  - etc.
- Who
  - Does it involve?
  - etc.
- When
  - Should a process / transaction happen?
  - etc.
- How
  - Should it be structured?
  - etc.

What you need to focus on?

- **Who** is our Web Application for?
  - The User
- What features will the web app have?
  - What would the user want / need from your system?
  - How will this work from the user's perspective?
    - How will they use it?

What you need to focus on?

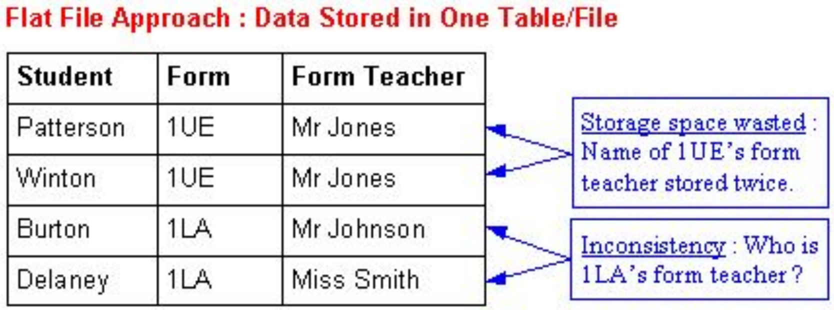
- What **data** will need to be stored / accessed to support this feature set?
  - Based on the desired features
  - Think about specifics
    - Instead of generalities like “Customer details” or “Order details”
    - Think of specific items of data (attributes)
      - The name of a customer
      - The address of a customer
      - The email address of a customer
      - The product reviews a customer might have provided for items they have purchased
      - etc

- What processing will we want to do on that data?
  - Based on the desired features
  - Starting point -> Destination
  - Inputs -> Outputs

CONCEPTS AND DEFINITIONS

“Flatfile” Database Design

- Data stored in a single big “table” or file
- Issues
  - Redundant data
    - Storage space is wasted
    - Maintenance becomes more complex
  - Inconsistent data
    - Different values representing the same item being stored?



Flat File Approach : Data Stored in One Table/File

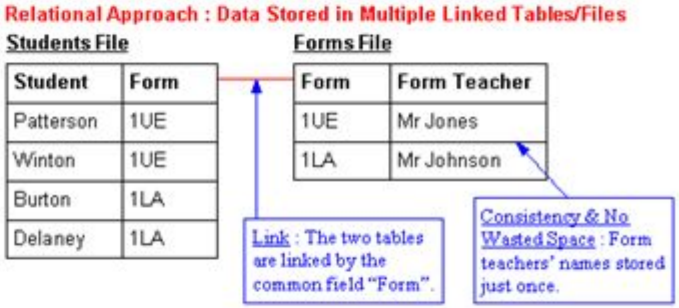
Student	Form	Form Teacher
Patterson	1UE	Mr Jones
Winton	1UE	Mr Jones
Burton	1LA	Mr Johnson
Delaney	1LA	Miss Smith

Storage space wasted : Name of 1UE’s form teacher stored twice.

Inconsistency : Who is 1LA’s form teacher ?

Relational Database Design

- Allow data to be stored in separate groups.
  - Relations / Tables
    - **Relations** of attributes
  - “Groups” are connected to each other via relationships
    - **Relations** of records
- By breaking the data into groups
  - Reduce the chance of mistakes happening
  - Does not take up any more space than necessary
  - Needs to be well designed



Relational Approach : Data Stored in Multiple Linked Tables/Files

Students File

Student	Form
Patterson	1UE
Winton	1UE
Burton	1LA
Delaney	1LA

Forms File

Form	Form Teacher
1UE	Mr Jones
1LA	Mr Johnson

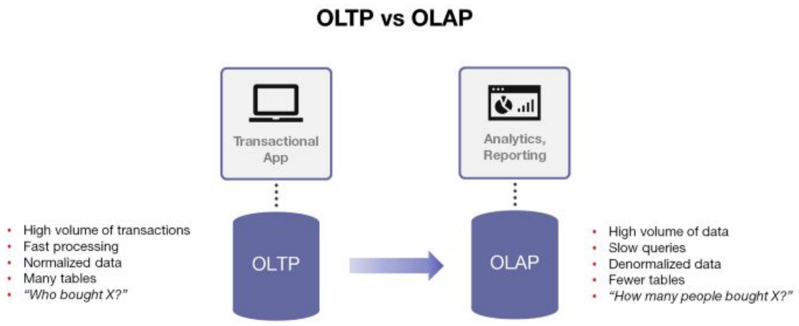
Link : The two tables are linked by the common field “Form”.

Consistency & No Wasted Space : Form teachers’ names stored just once.

Online Transaction Processing

- Up-to-date operational data
- Used for day-to-day operations
  - Frequent **read** operations
  - Frequent **write** operations
  - Transactions / Queries
    - Should aim to deal with only a small amount of data at once
    - Many and small

OLTP vs OLAP

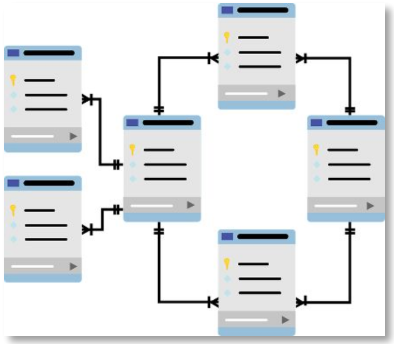


- High volume of transactions
- Fast processing
- Normalized data
- Many tables
- *"Who bought X?"*

- High volume of data
- Slow queries
- Denormalized data
- Fewer tables
- *"How many people bought X?"*

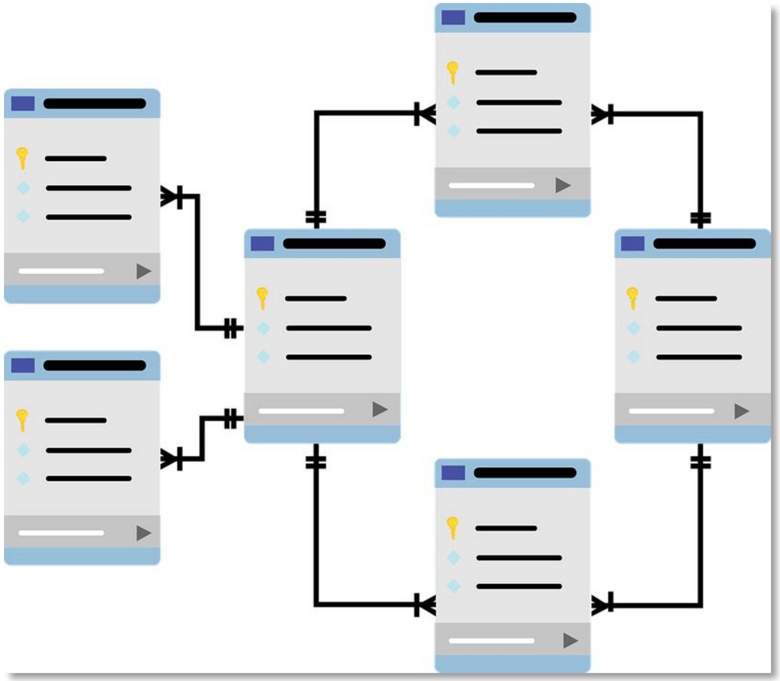
Database Schema

- Describes the **structure** of a database in a formal language that is supported by the database management system.
- Create a blueprint of the database.
  - This blueprint will not contain any data.
- The database schema uses logical formulas to create integrity constraints.
  - It is not possible to insert data into the database that violates these integrity constraints.
  - All constraints use the same language.



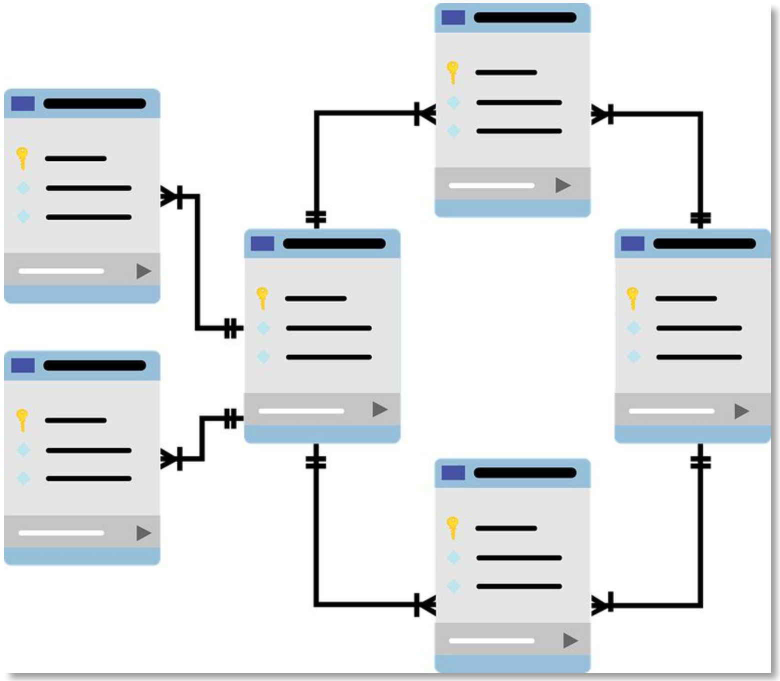
Database Schema

- There are different kinds of database schemas:
  - **Conceptual schema** expresses the concepts in the database, and how they relate to each other
  - **Logical schema** is a mapping of entities with their attributes, and the respective relations
  - **Physical schema** is a particular implementation of a logical schema.



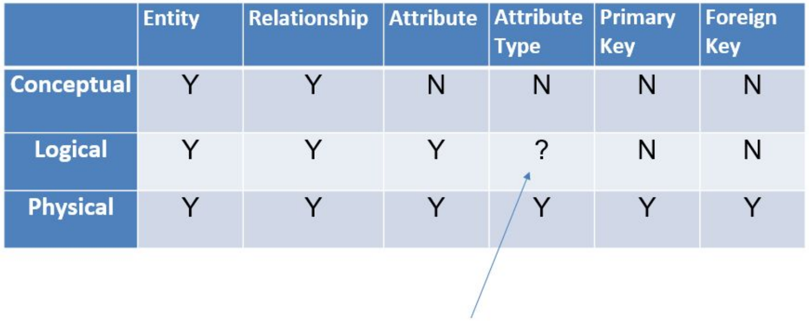
Database Schema

- Ideally, a database schema should have the following properties:
  - It should be *complete*: all information in the source should be included.
  - It should be *minimal*: it should not be possible to leave out a relation, without losing information
  - It should be *normalised*: A certain piece of data should be in the schema only once.



Database Design

- Before we create a database, we need to **design** one
- Conceptual Design
  - Build a model of the database independent of DBMS details
- Logical Design
  - Model further structure of the data
  - Still abstract enough to be independent of DBMS details
- Physical Design
  - How is the data within the database precisely stored and managed by the DBMS?
    - Specific types
    - Keys
    - Indexes
    - etc



	Entity	Relationship	Attribute	Attribute Type	Primary Key	Foreign Key
Conceptual	Y	Y	N	N	N	N
Logical	Y	Y	Y	?	N	N
Physical	Y	Y	Y	Y	Y	Y

Sometimes a “kind” of data can be identified for that attribute without necessarily specifying a specific concrete type

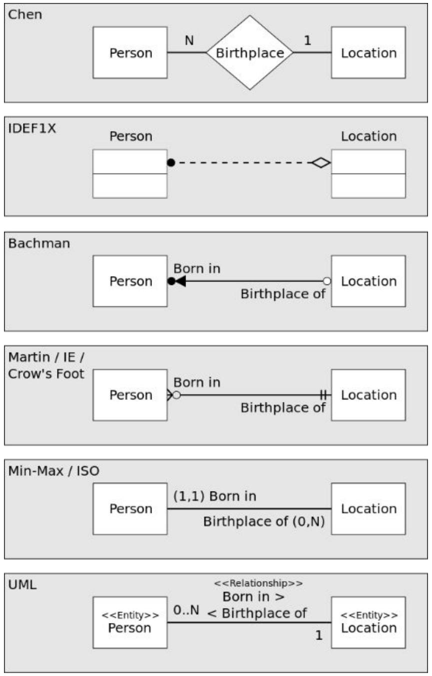
ERD Type

	Entity	Relationship	Attribute	Attribute Type	Primary Key	Foreign Key
Conceptual	Y	Y	N	N	N	N
Logical	Y	Y	Y	?	N	N
Physical	Y	Y	Y	Y	Y	Y

Sometimes a "kind" of data can be identified for that attribute without necessarily specifying a specific concrete type

Entity Relationship Diagrams

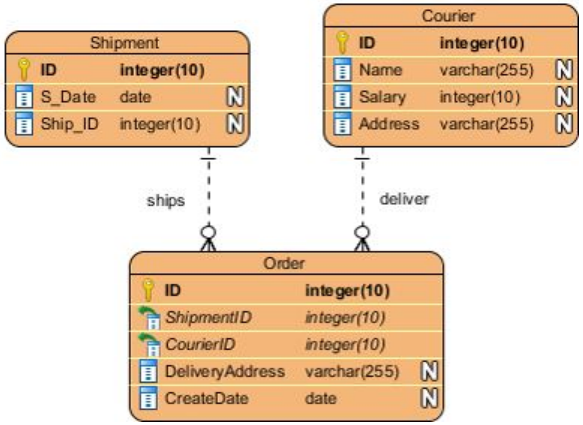
- A type of structural diagram for use in database design.
- Entity Relationship Modelling is an approach to semantic modelling of data within information systems
- The process of entity relationship modelling was originally defined by the work on Chen (1976) and has been continued to be refined through further work since.



Different notations / styles
Representing the same information
Recommended: “Crow’s Foot”

An Example ERD

- Entities
  - Shipment
  - Courier
  - Order
- Attributes
  - S_Date is a **date** and belongs to Shipment entity
  - Name is a variable length character array and belongs to the Courier entity
  - etc
- Relationships
  - An Order is shipped via a single Shipment
    - A Shipment can ship many Orders
  - An Order is delivered by a single Courier
    - A Courier can deliver many orders
- Records
  - Each individual Shipment, Order or Courier would be represented as a "row" in our final table
  - Think of the Records as the "Objects" to the Entity / Tables "Classes"



Entity/Relationship Modelling

- E/R Modelling is used for designing our data model
  1. Identify Entities
     - objects or items of interest
  2. Identify the Attributes
     - facts about, or properties of, an entity
  3. Identify the Relationships
     - links between entities
  4. Establish cardinality of relationships
     - How many records of one entity can be related to how many records of another entity?
  5. Establish modality of relationship
     - Is the relationship optional, or mandatory?
  6. Identify keys
     1. Primary key
        - What value can be used to uniquely identify each record in the entity?
          - » StudentID?
          - » StudentEmailAddress?

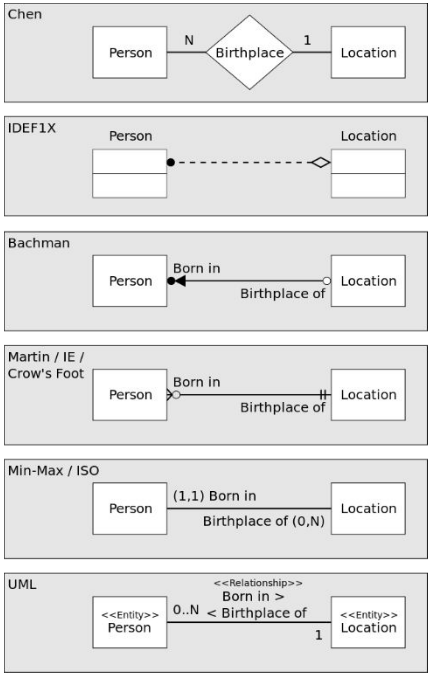
- Example
  - For a university database:
  - Entities – Students, Modules, Lecturers
  - Students attributes could be – Student ID, Name and course
  - Could have relationships with Modules (enrolment) and Lecturers (tutor/tutee)
  - Cardinality
    - E.g. A student may study multiple modules while a module may be studied by many students
  - Modality
    - e.g.

Lab Talk

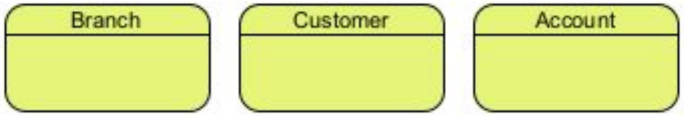
ENTITY RELATIONSHIP MODELLING

Entity Relationship Diagrams

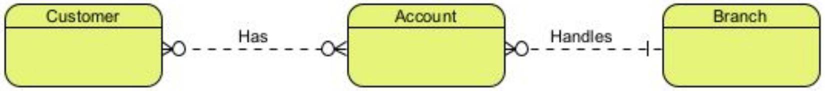
- A type of structural diagram for use in database design.
- Entity Relationship Modelling is an approach to semantic modelling of data within information systems
- The process of entity relationship modelling was originally defined by the work on Chen (1976) and has been continued to be refined through further work since.



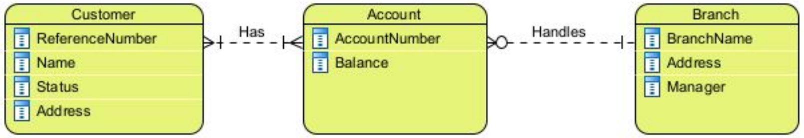
Entity



Relationship



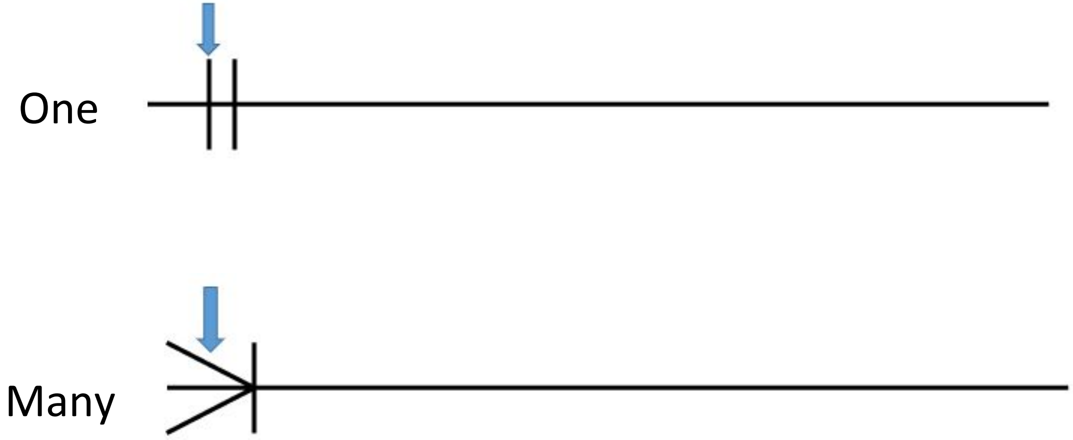
Attributes



Customer
ReferenceNumber
Name
Status
Address
Has
Account
AccountNumber
Balance
Handles
Branch
BranchName
Address
Manager

Cardinality and Modality

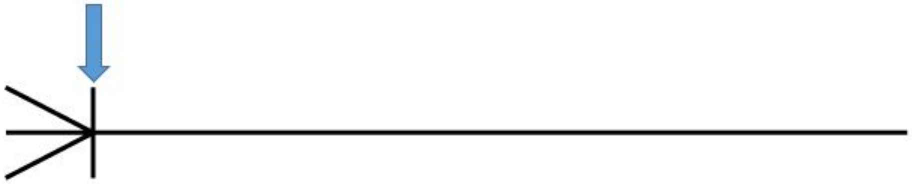
- Cardinality: Multiplicity
  - Maximum amount of times an instance of one entity can be associated with instances of another entity.



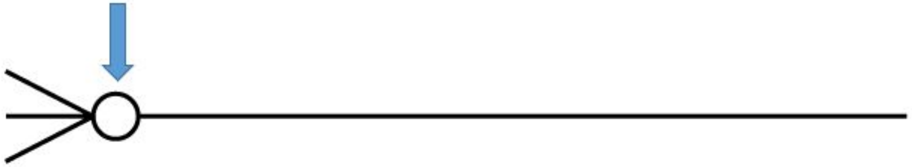
Cardinality and Modality

- Modality: Minimality
  - *minimum* number of times one instance can be related to others.

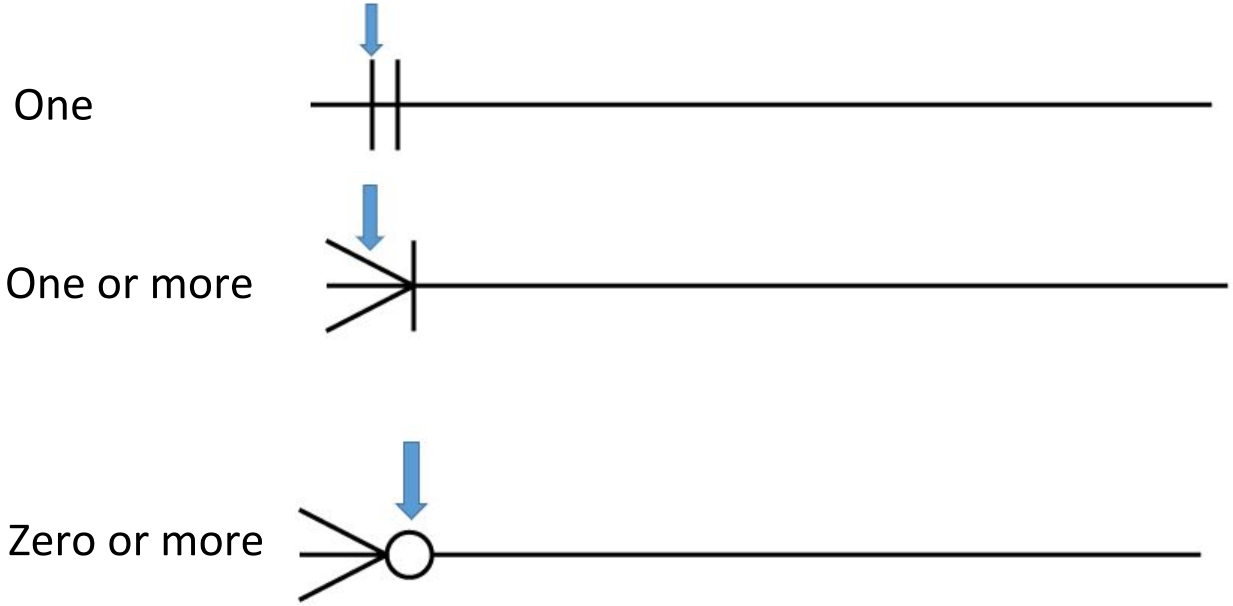
One



Zero



Cardinality and Modality

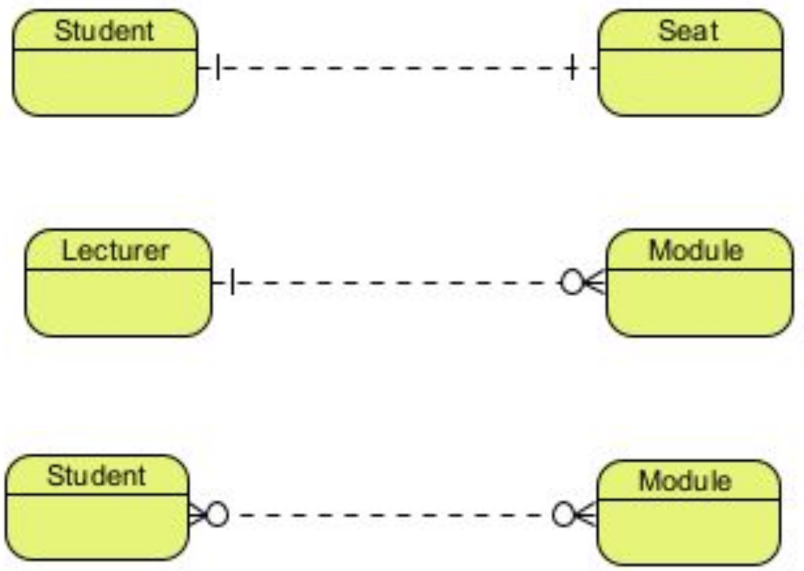


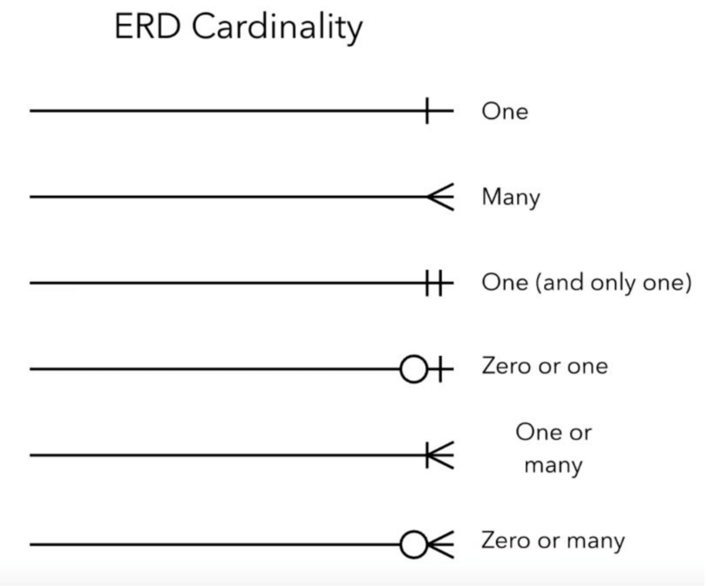
One

One or more

Zero or more

Cardinality and Modality





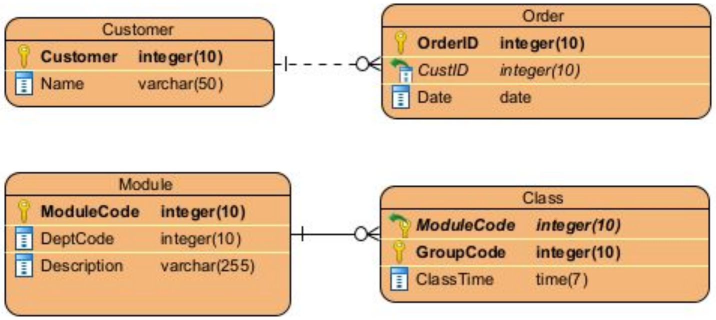
Keys

- Candidate key
  - A collection of one or more attributes which can be used to uniquely identify records belonging to a given table / entity
- Primary Key
  - Chosen from the candidate keys
  - A unique identifier for a record within a table / entity, ensuring that no two records have the same identifier.
  - Unique constraint
  - Not null constraint
- Foreign Key
  - A reference to the primary key of another table / entity

Identifying Relationship

- Weak
  - Non-identifying
  - Existence is independent
  - Usually PK of Child doesn’t contain PK component of Parent Entity
- Strong
  - Identifying
  - Existence is dependent on another entity
  - Usually PK of Child contains PK component from a Parent Entity

“If we delete a record for one entity what happens to the records linked to it for another entity?”



Customer	
Customer	integer(10)
Name	varchar(50)

Order	
OrderID	integer(10)
CustID	integer(10)
Date	date

Module	
ModuleCode	integer(10)
DeptCode	integer(10)
Description	varchar(255)

Class	
ModuleCode	integer(10)
GroupCode	integer(10)
ClassTime	time(7)

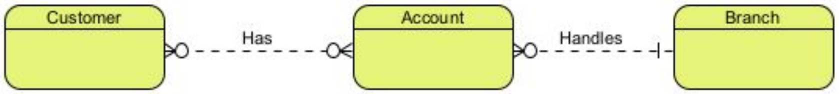
ERD Type

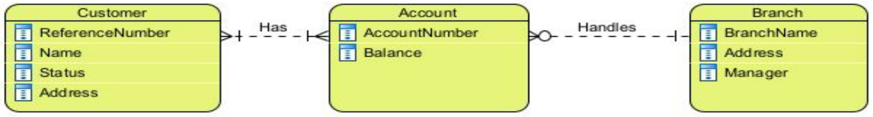
	Entity	Relationship	Attribute	Attribute Type	Primary Key	Foreign Key
Conceptual	Y	Y	N	N	N	N
Logical	Y	Y	Y	?	N	N
Physical	Y	Y	Y	Y	Y	Y

Sometimes a “kind” of data can be identified for that attribute without necessarily specifying a specific concrete type

ERD Type

	Entity	Relationship	Attribute	Attribute Type	Primary Key	Foreign Key
Conceptual	Y	Y	N	N	N	N
Logical	Y	Y	Y	?	N	N
Physical	Y	Y	Y	Y	Y	Y

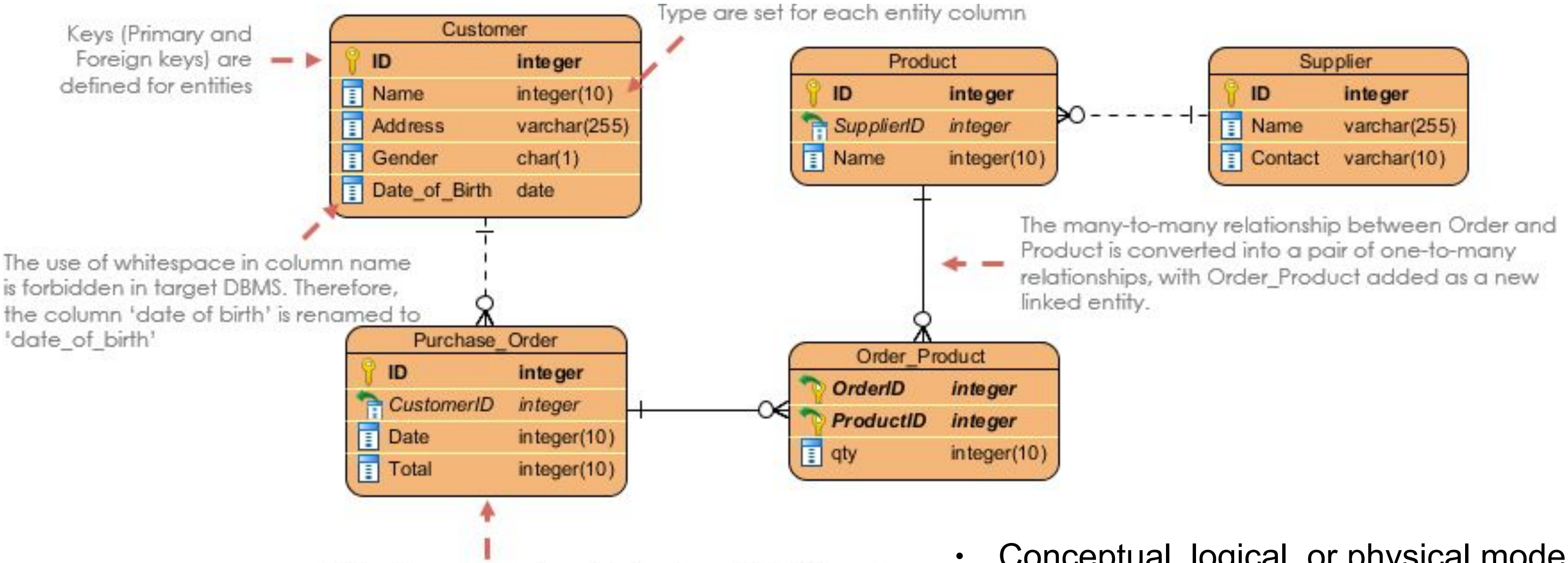




?

Note: some diagramming tools use colour to differentiate, instead get into the habit of looking at what constraints are met with the diagram

Example



Keys (Primary and Foreign keys) are defined for entities

Type are set for each entity column

Customer	
ID	**integer**
Name	integer(10)
Address	varchar(255)
Gender	char(1)
Date_of_Birth	date

Product	
ID	**integer**
SupplierID	*integer*
Name	integer(10)

Supplier	
ID	**integer**
Name	varchar(255)
Contact	varchar(10)

The use of whitespace in column name is forbidden in target DBMS. Therefore, the column 'date of birth' is renamed to 'date_of_birth'

The many-to-many relationship between Order and Product is converted into a pair of one-to-many relationships, with Order_Product added as a new linked entity.

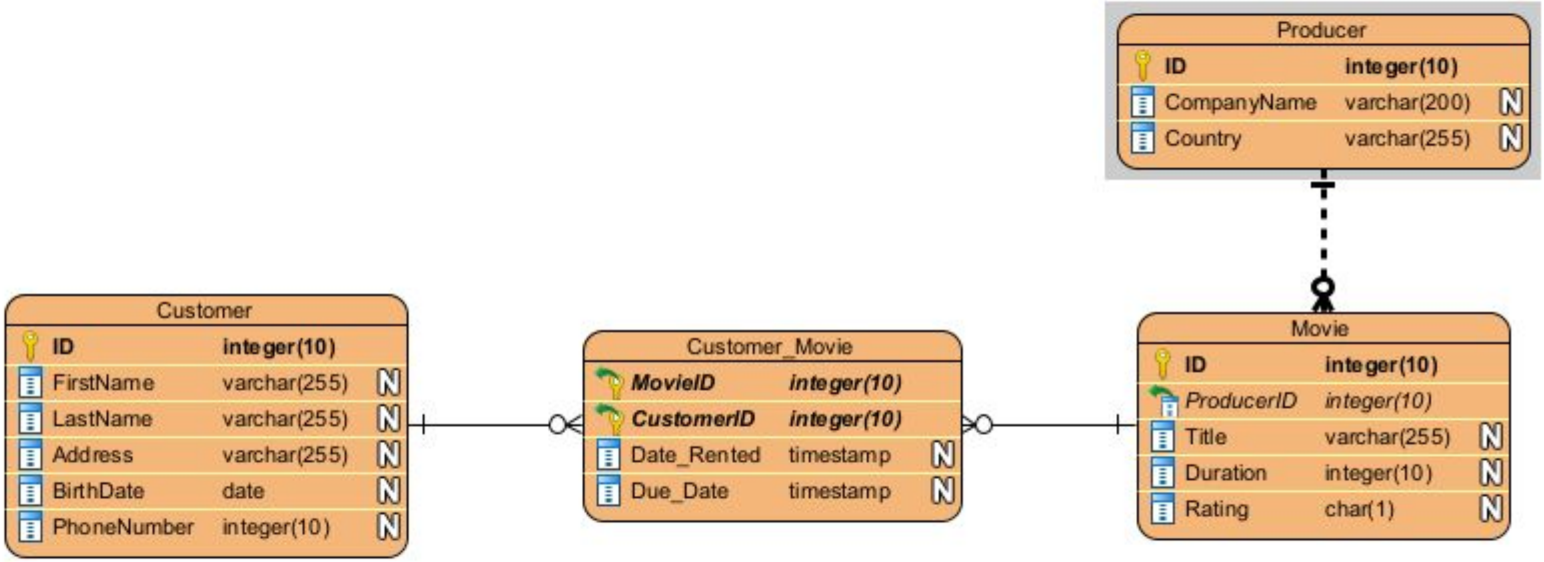
Purchase_Order	
ID	**integer**
CustomerID	*integer*
Date	integer(10)
Total	integer(10)

Order_Product	
OrderID	***integer***
ProductID	***integer***
qty	integer(10)

'Order' is a reserved word for the target DBMS. Therefore, the entity Order is renamed to Purchase Order.

- Conceptual, logical, or physical model of the database? Why do you think so?
- Think of at least 3 sample record tuples to represent "rows" for each relation/table.
- What limitations does this specific database structure have?
  - What possible use cases may not be possible based on these limitations?
- What additional constraints could be placed on the attributes within each table to prevent invalid data being stored?

Example



Producer

ID	**integer(10)**	
CompanyName	varchar(200)	N
Country	varchar(255)	N

Customer

ID	**integer(10)**	
FirstName	varchar(255)	N
LastName	varchar(255)	N
Address	varchar(255)	N
BirthDate	date	N
PhoneNumber	integer(10)	N

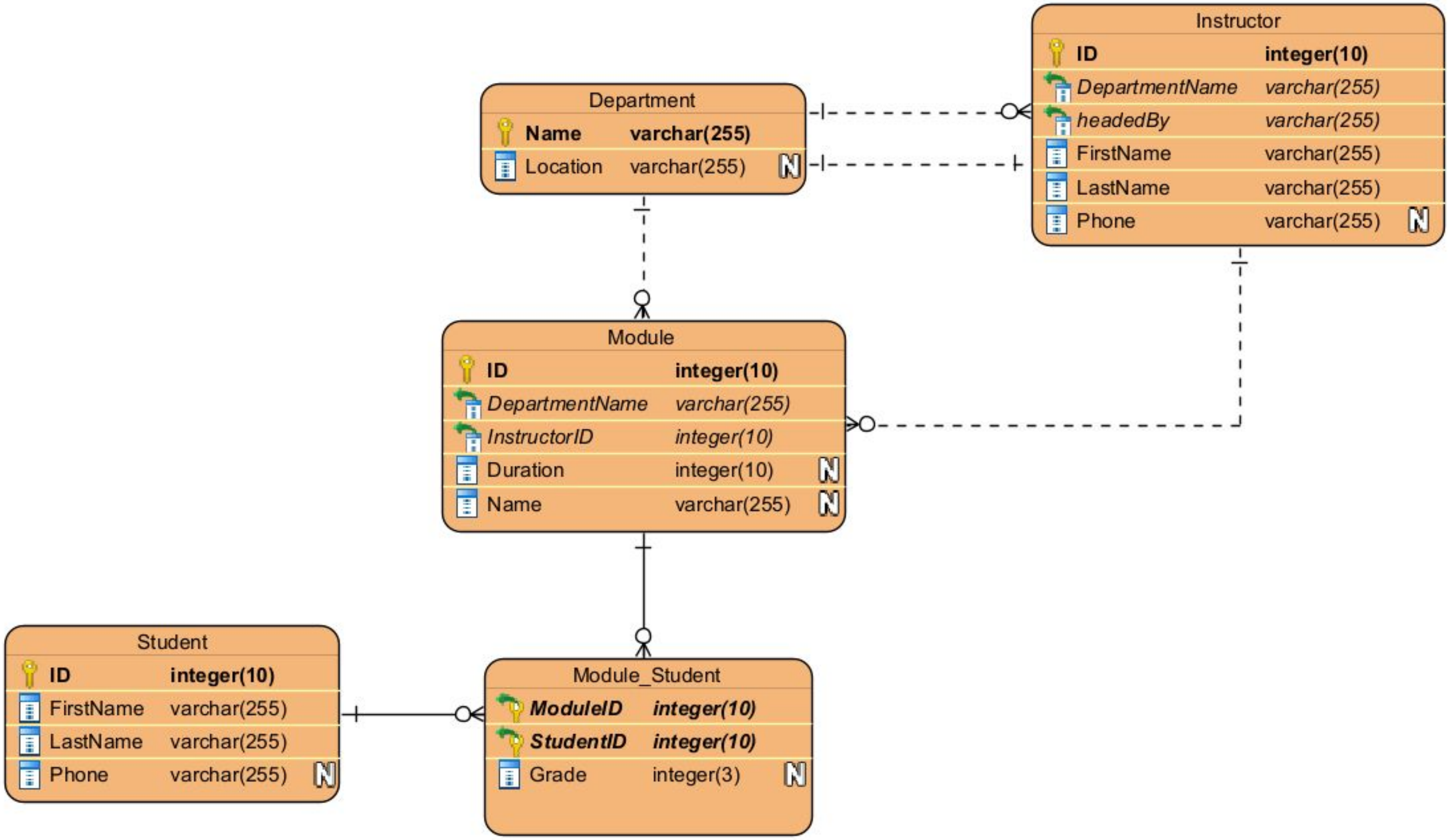
Customer_Movie

MovieID	***integer(10)***	
CustomerID	***integer(10)***	
Date_Rented	timestamp	N
Due_Date	timestamp	N

Movie

ID	**integer(10)**	
ProducerID	*integer(10)*	
Title	varchar(255)	N
Duration	integer(10)	N
Rating	char(1)	N

Example



Department

Name	**varchar(255)**	
Location	varchar(255)	N

Instructor

ID	**integer(10)**	
DepartmentName	*varchar(255)*	
headedBy	*varchar(255)*	
FirstName	varchar(255)	
LastName	varchar(255)	
Phone	varchar(255)	N

Module

ID	**integer(10)**	
DepartmentName	*varchar(255)*	
InstructorID	*integer(10)*	
Duration	integer(10)	N
Name	varchar(255)	N

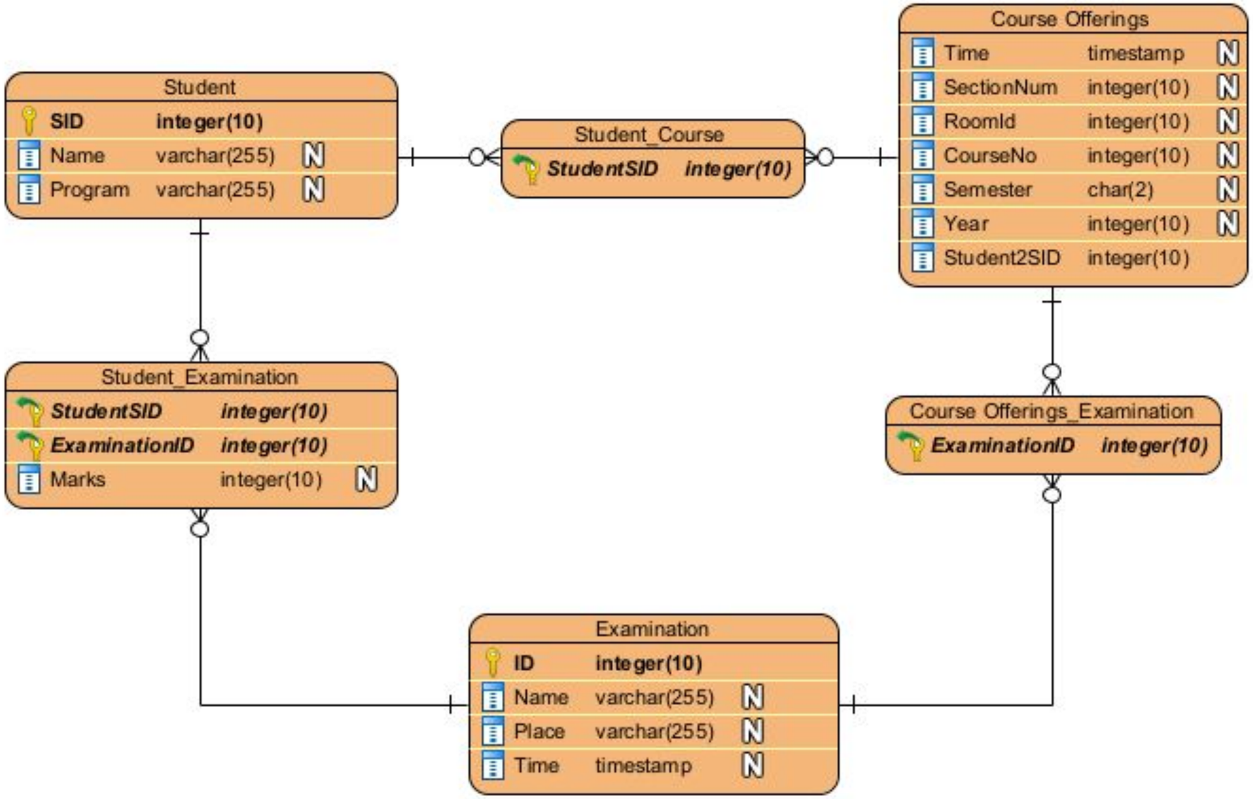
Student

ID	**integer(10)**	
FirstName	varchar(255)	
LastName	varchar(255)	
Phone	varchar(255)	N

Module_Student

ModuleID	***integer(10)***	
StudentID	***integer(10)***	
Grade	integer(3)	N

Example



Student

SID	**integer(10)**	
Name	varchar(255)	N
Program	varchar(255)	N

Student_Course

StudentSID	***integer(10)***

Course Offerings

Time	timestamp	N
SectionNum	integer(10)	N
RoomId	integer(10)	N
CourseNo	integer(10)	N
Semester	char(2)	N
Year	integer(10)	N
Student2SID	integer(10)	

Student_Examination

StudentSID	***integer(10)***	
ExaminationID	***integer(10)***	
Marks	integer(10)	N

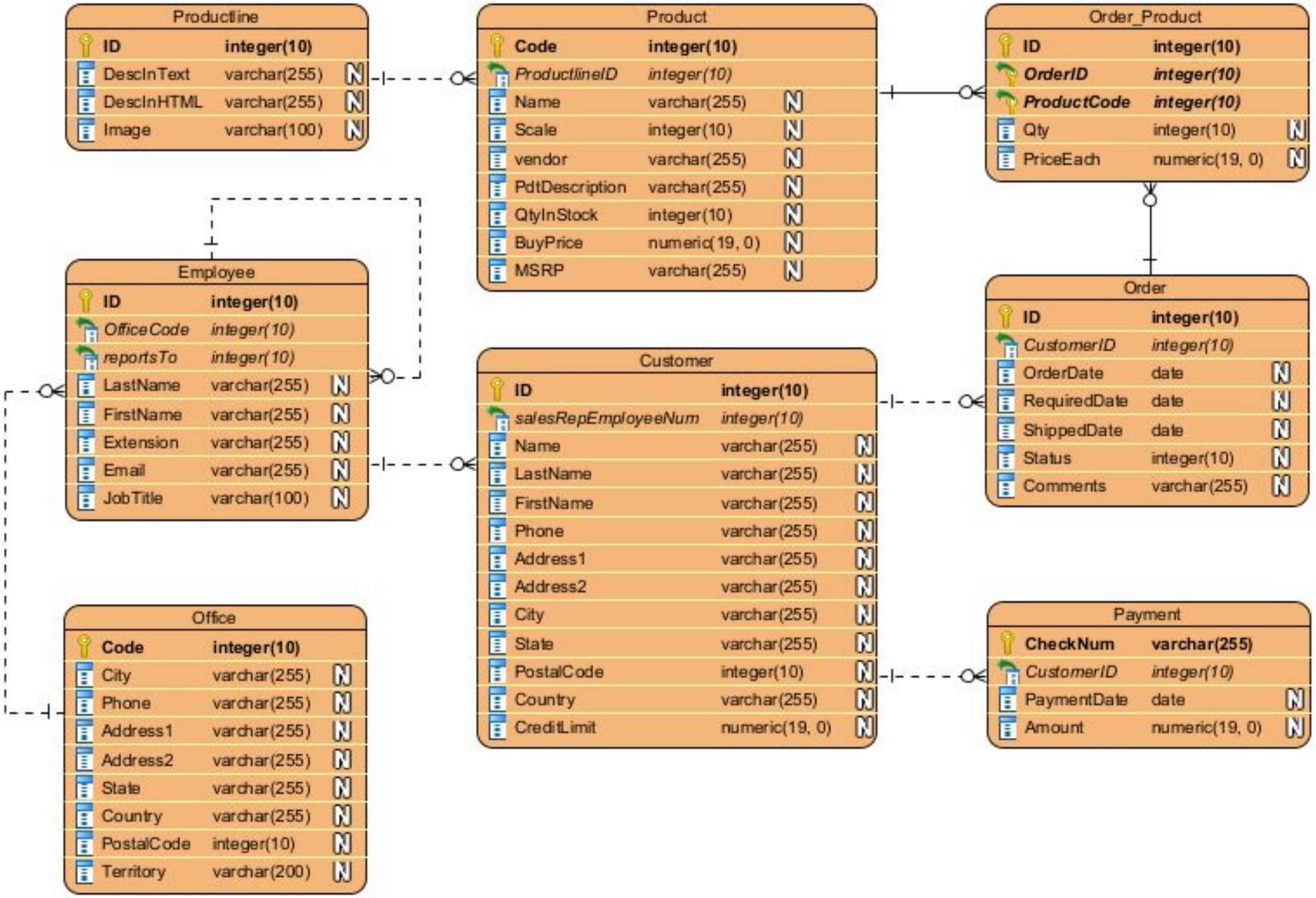
Course Offerings_Examination

ExaminationID	***integer(10)***

Examination

ID	**integer(10)**	
Name	varchar(255)	N
Place	varchar(255)	N
Time	timestamp	N

Example



Productline		
ID	**integer(10)**	
DescInText	varchar(255)	N
DescInHTML	varchar(255)	N
Image	varchar(100)	N

Product		
Code	**integer(10)**	
ProductlineID	*integer(10)*	
Name	varchar(255)	N
Scale	integer(10)	N
vendor	varchar(255)	N
PdtDescription	varchar(255)	N
QtyInStock	integer(10)	N
BuyPrice	numeric(19, 0)	N
MSRP	varchar(255)	N

Order_Product		
ID	**integer(10)**	
OrderID	***integer(10)***	
ProductCode	***integer(10)***	
Qty	integer(10)	N
PriceEach	numeric(19, 0)	N

Employee		
ID	**integer(10)**	
OfficeCode	*integer(10)*	
reportsTo	*integer(10)*	
LastName	varchar(255)	N
FirstName	varchar(255)	N
Extension	varchar(255)	N
Email	varchar(255)	N
JobTitle	varchar(100)	N

Customer		
ID	**integer(10)**	
salesRepEmployeeNum	*integer(10)*	
Name	varchar(255)	N
LastName	varchar(255)	N
FirstName	varchar(255)	N
Phone	varchar(255)	N
Address1	varchar(255)	N
Address2	varchar(255)	N
City	varchar(255)	N
State	varchar(255)	N
PostalCode	integer(10)	N
Country	varchar(255)	N
CreditLimit	numeric(19, 0)	N

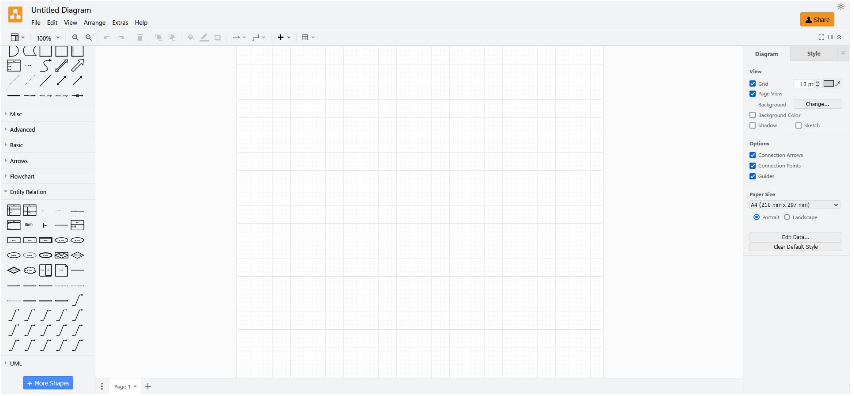
Order		
ID	**integer(10)**	
CustomerID	*integer(10)*	
OrderDate	date	N
RequiredDate	date	N
ShippedDate	date	N
Status	integer(10)	N
Comments	varchar(255)	N

Office		
Code	**integer(10)**	
City	varchar(255)	N
Phone	varchar(255)	N
Address1	varchar(255)	N
Address2	varchar(255)	N
State	varchar(255)	N
Country	varchar(255)	N
PostalCode	integer(10)	N
Territory	varchar(200)	N

Payment		
CheckNum	**varchar(255)**	
CustomerID	*integer(10)*	
PaymentDate	date	N
Amount	numeric(19, 0)	N

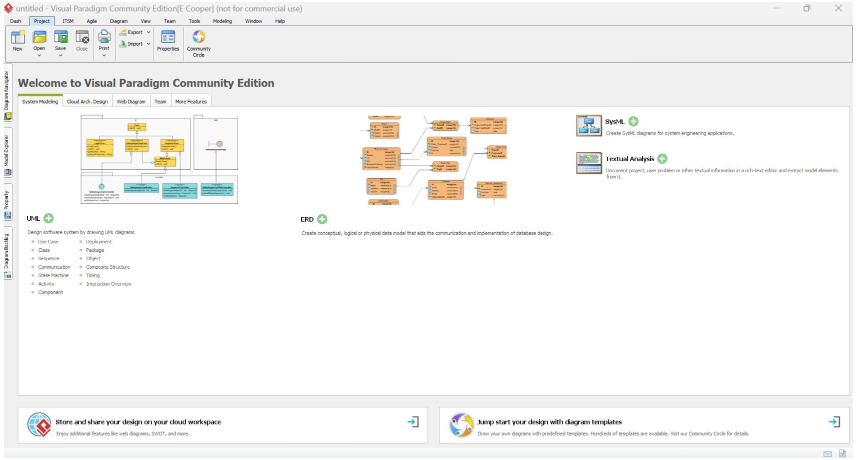
Diagramming Software

Draw.io (free)



☑ Grid 10 pt
☑ Page View
Background Change...
☐ Background Color
☐ Shadow ☐ Sketch
☑ Connection Arrows
☑ Connection Points
☑ Guides
A4 (210 mm x 297 mm)
● Portrait ○ Landscape

Visual Paradigm Community (free but limitations)



Welcome to Visual Paradigm Community Edition

UML

Design software system by drawing UML diagrams

- Use Case
- Class
- Sequence
- Communication
- State Machine
- Activity
- Component
- Deployment
- Package
- Object
- Composite Structure
- Timing
- Interaction Overview

ERD

Create conceptual, logical or physical data model that aids the communication and implementation of database design.

SysML

Create SysML diagrams for system engineering applications.

Textual Analysis

Document project, user problem or other textual information in a rich-text editor and extract model elements from it.

Store and share your design on your cloud workspace

Enjoy additional features like web diagrams, SWOT, and more.

Jump start your design with diagram templates

Draw your own diagrams with predefined templates. Hundreds of templates are available. Visit our Community Circle for details.

- Recommend draw.io as least likely to cause issues with locking off features.
- However, Visual Paradigm (full version) is a very useful feature rich application for Software Design in general, see it as a possible alternative.