COMP1204: Modelling and SQL

Oli Bills

ofb@ecs.soton.ac.uk

What we have covered so far

- Database Design Theory
- Functional dependencies
- Keys/Superkeys
- Normalisation
- Next… data modelling

Terminology

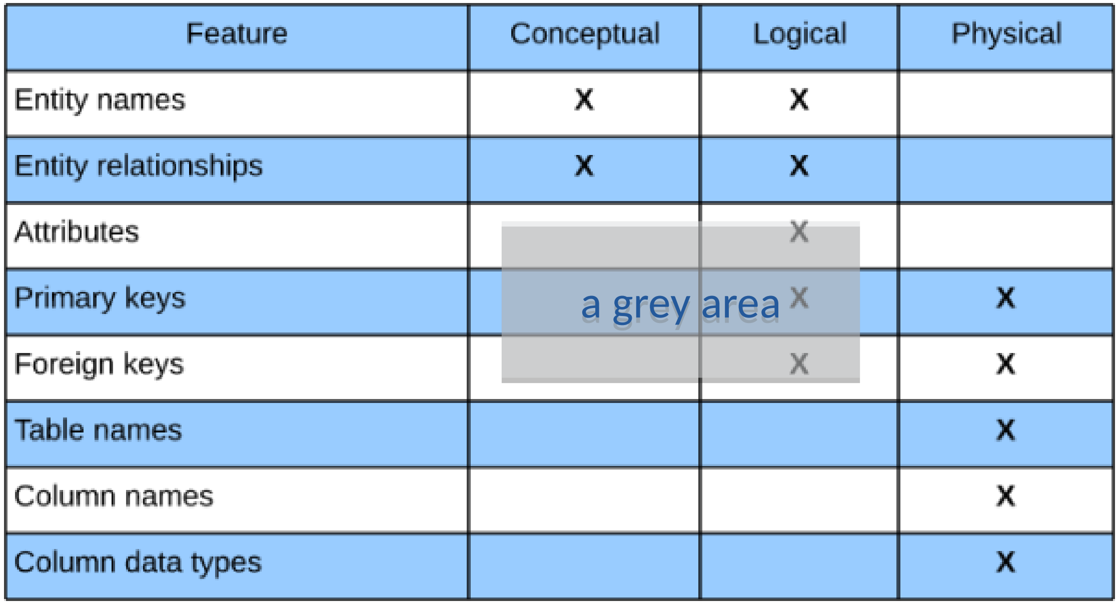
- Student as an **entity** (an object, a concept) has a number of **attributes** (ID,Name)
- Student as a **Relation**: Student(ID,Name) has the attributes/columns (ID,Name)
- Student **HAS** a Tutor (that’s a **relationship**)

Data Model

- “Things” of importance to the system
- How they relate to each other
- Built / Modified - iterated until suited to system model
- Can be represented as UML class diagrams
    - add behaviour of each entity to the model

Three types of Data Modelling

- Conceptual -> Ideas
- Logical -> High Level Design
- Physical -> Low Level

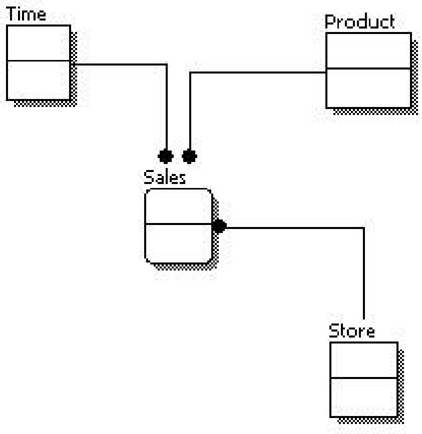


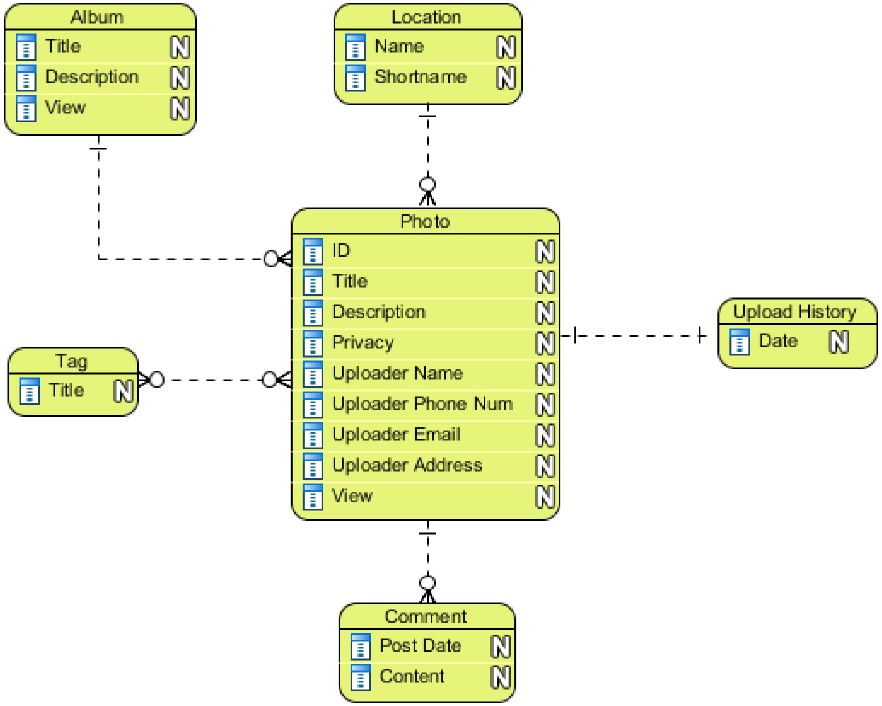
Feature	Conceptual	Logical	Physical
Entity names	X	X	
Entity relationships	X	X	
Attributes		X	
Primary keys		X	X
Foreign keys		X	X
Table names			X
Column names			X
Column data types			X

a grey area

Conceptual Model

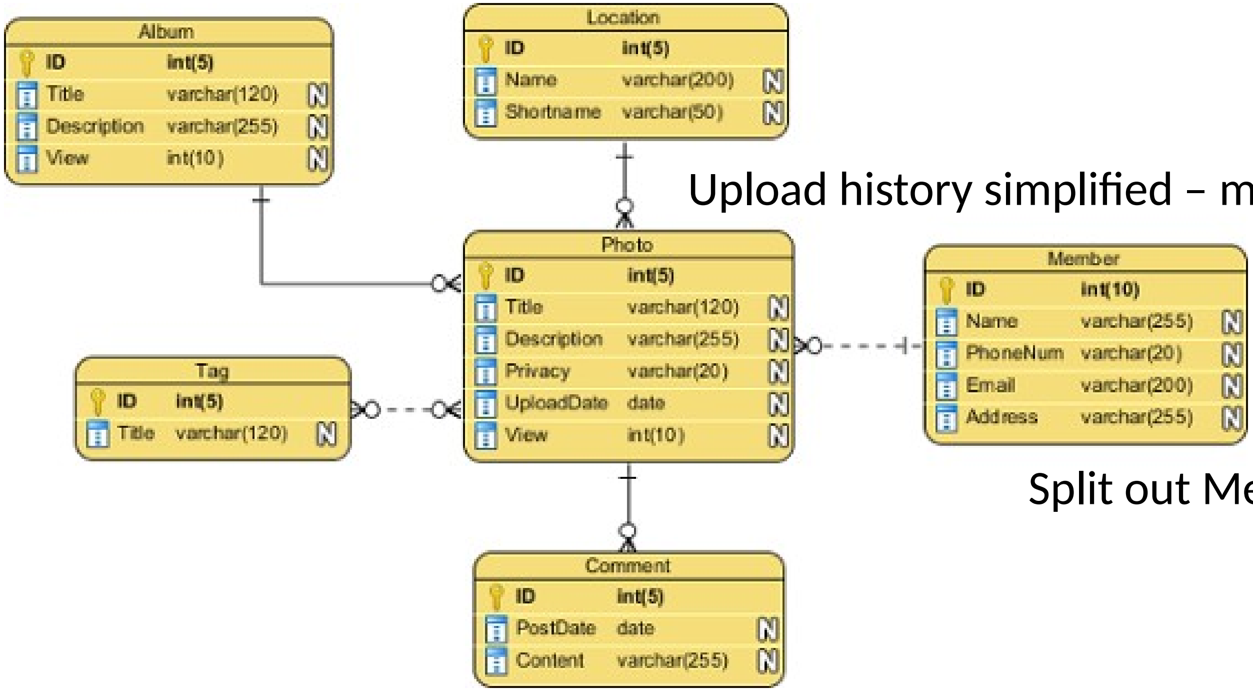
- Directly from the requirements and domain
- No thought of database design
- Entity names and relationships and sometimes attributes





Logical Model

- Identification of attributes
- Identification of attribute types

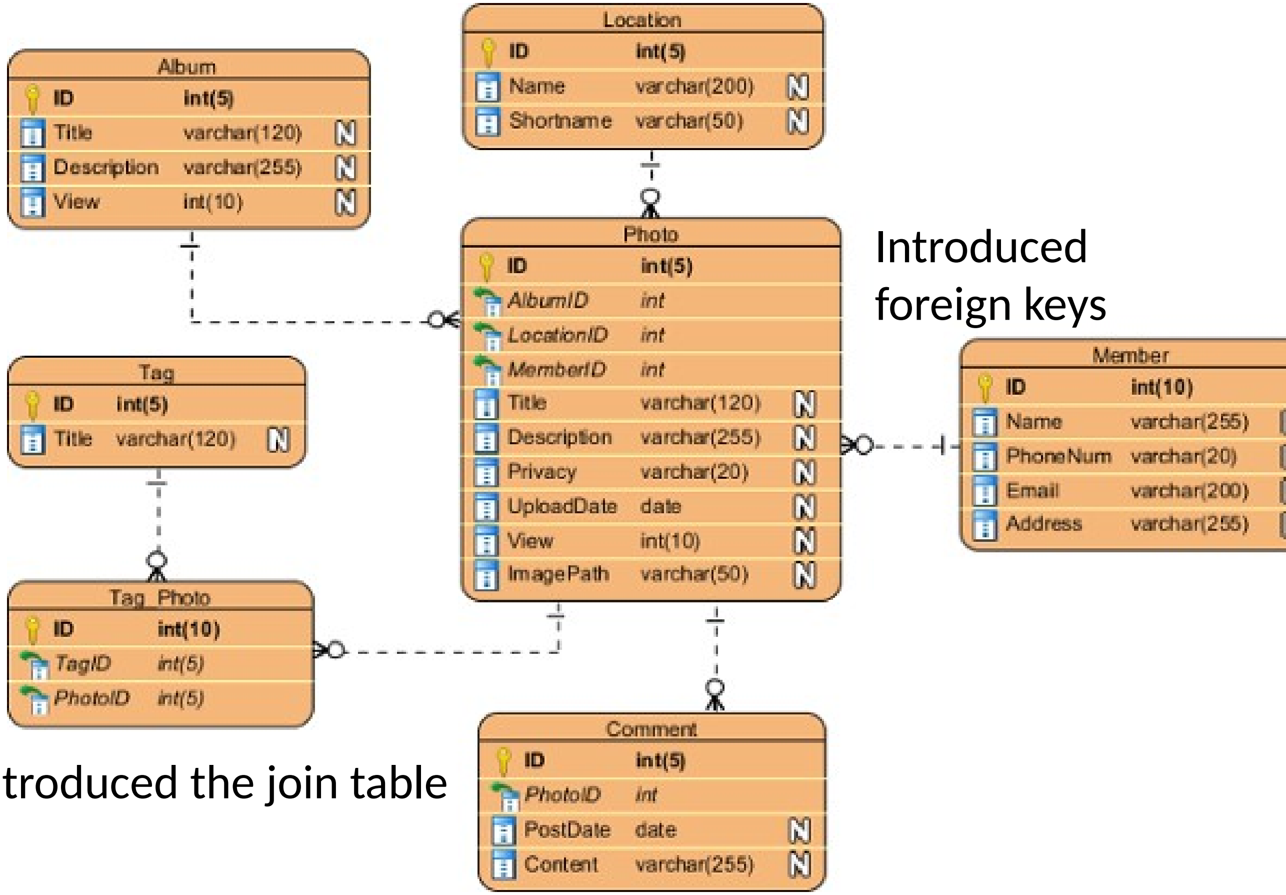


Upload history simplified – multiple photos

Split out Member

Physical Model

- Consideration of database structure
- Actual tables and fields
- Implementation of relationships (e.g. keys, join tables), indexes etc.



Introduced foreign keys

Introduced the join table

Album		
ID	int(5)	
Title	varchar(120)	N
Description	varchar(255)	N
View	int(10)	N

Location		
ID	int(5)	
Name	varchar(200)	N
Shortname	varchar(50)	N

Photo		
ID	int(5)	
AlbumID	*int*	
LocationID	*int*	
MemberID	*int*	
Title	varchar(120)	N
Description	varchar(255)	N
Privacy	varchar(20)	N
UploadDate	date	N
View	int(10)	N
ImagePath	varchar(50)	N

Member	
ID	int(10)
Name	varchar(255)
PhoneNum	varchar(20)
Email	varchar(200)
Address	varchar(255)

Tag		
ID	int(5)	
Title	varchar(120)	N

Tag_Photo	
ID	int(10)
TagID	*int(5)*
PhotoID	*int(5)*

Comment		
ID	int(5)	
PhotoID	*int*	
PostDate	date	N
Content	varchar(255)	N

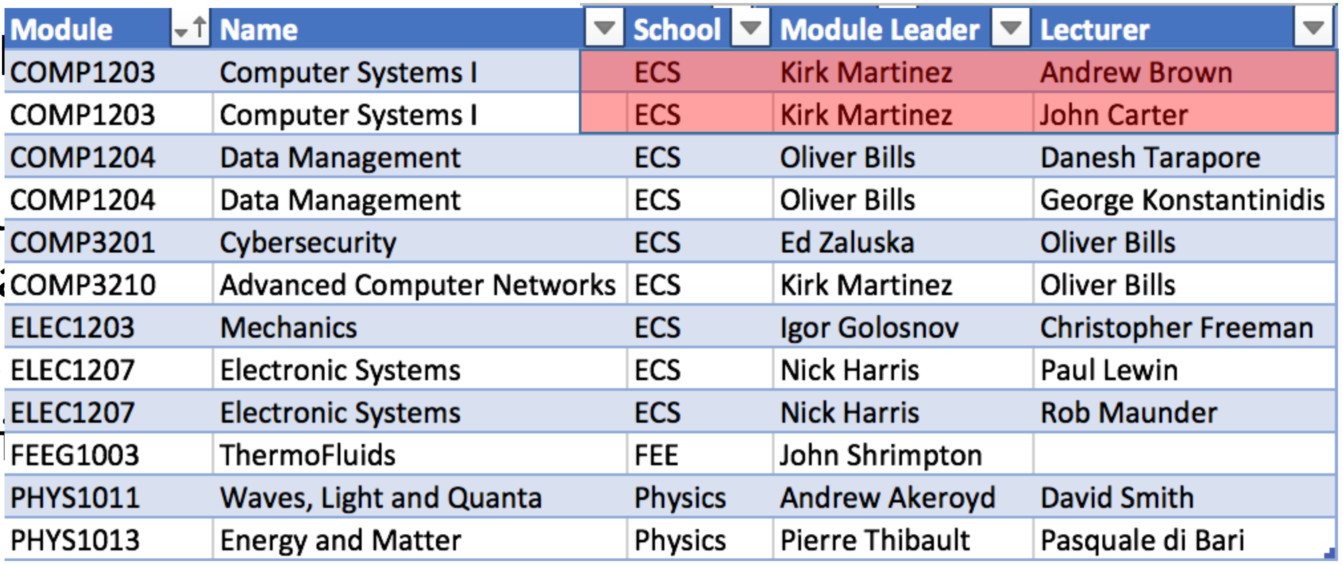
From Logical to Physical Design

- From the high-level design (concepts and ideas) to the low-level design (implementation)
- Logical modelling is for attributes used in the real world
    - You shouldn't be making attributes up in logical stages
    - Your natural keys uniquely identify entities in world
    - Your normalisation should be based on your logical modelling

- For people who have done some database design before, one of the first things people want to do in normalisation is start adding ID fields
    - But things like these these come later, on implementation, not at modelling
    - At the logical stage, you should only be thinking about the data that actually exists in the world

What can go wrong?

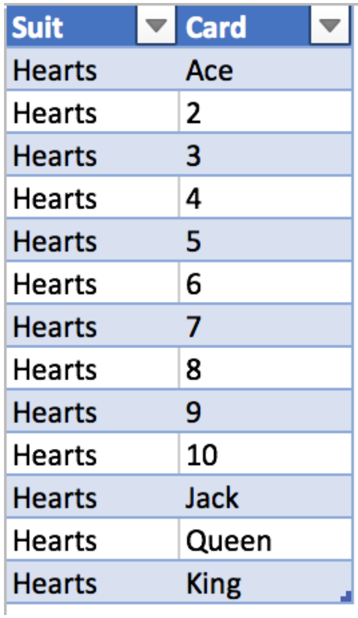
- Bunging an
  to uniquely
- But now you
  about the da
  - We've got
- Breaks relati
  design
  - "Just link everything together by IDs"
- You're basically just linking spreadsheets together…



Module	Name	School	Module Leader	Lecturer
COMP1203	Computer Systems I	ECS	Kirk Martinez	Andrew Brown
COMP1203	Computer Systems I	ECS	Kirk Martinez	John Carter
COMP1204	Data Management	ECS	Oliver Bills	Danesh Tarapore
COMP1204	Data Management	ECS	Oliver Bills	George Konstantinidis
COMP3201	Cybersecurity	ECS	Ed Zaluska	Oliver Bills
COMP3210	Advanced Computer Networks	ECS	Kirk Martinez	Oliver Bills
ELEC1203	Mechanics	ECS	Igor Golosnov	Christopher Freeman
ELEC1207	Electronic Systems	ECS	Nick Harris	Paul Lewin
ELEC1207	Electronic Systems	ECS	Nick Harris	Rob Maunder
FEEG1003	ThermoFluids	FEE	John Shrimpton	
PHYS1011	Waves, Light and Quanta	Physics	Andrew Akeroyd	David Smith
PHYS1013	Energy and Matter	Physics	Pierre Thibault	Pasquale di Bari

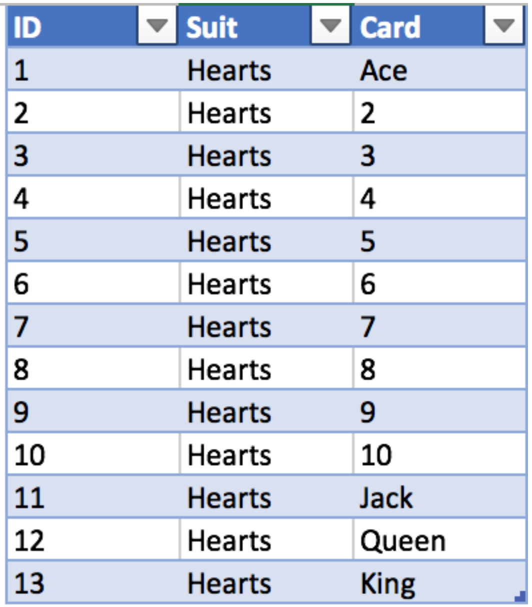
The classic example

Primary key (Suit, Card)



Suit	Card
Hearts	Ace
Hearts	2
Hearts	3
Hearts	4
Hearts	5
Hearts	6
Hearts	7
Hearts	8
Hearts	9
Hearts	10
Hearts	Jack
Hearts	Queen
Hearts	King

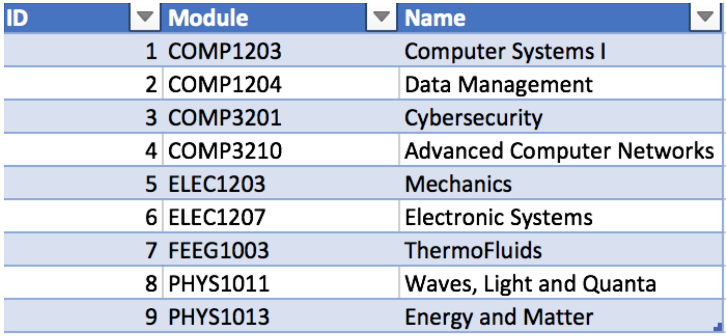
Primary key (ID)



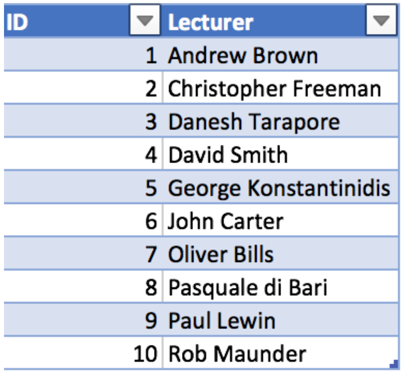
ID	Suit	Card
1	Hearts	Ace
2	Hearts	2
3	Hearts	3
4	Hearts	4
5	Hearts	5
6	Hearts	6
7	Hearts	7
8	Hearts	8
9	Hearts	9
10	Hearts	10
11	Hearts	Jack
12	Hearts	Queen
13	Hearts	King

Surrogate Keys: ID Fields

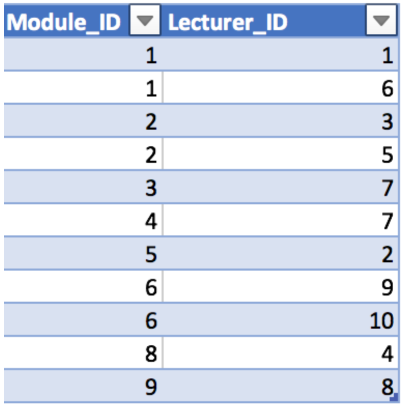
- Automatic ID fields are what we term surrogate keys: They have **no** business meaning
- Surrogate keys are attributes created and maintained **by the system** to aid in uniquely identifying an instance of an entity – they do not occur in the real world
- They are used because natural keys aren't generally stable, consistent or efficient to use in a database system
- They stand in place of the natural keys for technical purposes only, and are added as part of physical design – after the logical modelling and normalisation
- But you don't need to use them everywhere – some developers want an ID in every table…



ID	Module	Name
1	COMP1203	Computer Systems I
2	COMP1204	Data Management
3	COMP3201	Cybersecurity
4	COMP3210	Advanced Computer Networks
5	ELEC1203	Mechanics
6	ELEC1207	Electronic Systems
7	FEEG1003	ThermoFluids
8	PHYS1011	Waves, Light and Quanta
9	PHYS1013	Energy and Matter



ID	Lecturer
1	Andrew Brown
2	Christopher Freeman
3	Danesh Tarapore
4	David Smith
5	George Konstantinidis
6	John Carter
7	Oliver Bills
8	Pasquale di Bari
9	Paul Lewin
10	Rob Maunder



Module_ID	Lecturer_ID
1	1
1	6
2	3
2	5
3	7
4	7
5	2
6	9
6	10
8	4
9	8

More keys

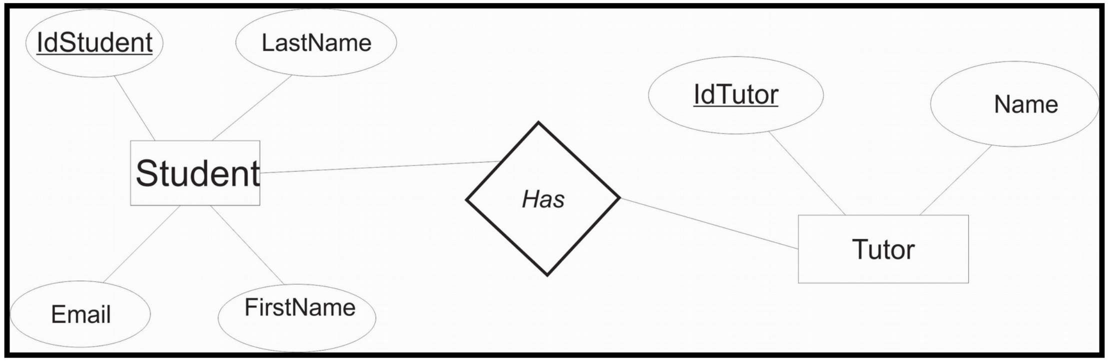
- **Primary key**: a key with attributes that are not allowed to be **Null**
- **Foreign key**: an attribute of one relation that references an attribute (primary key) of another relation
  - Employee(ID,Name,**DeptNo**)
  - Department(ID,Name)

Conceptual Data Models

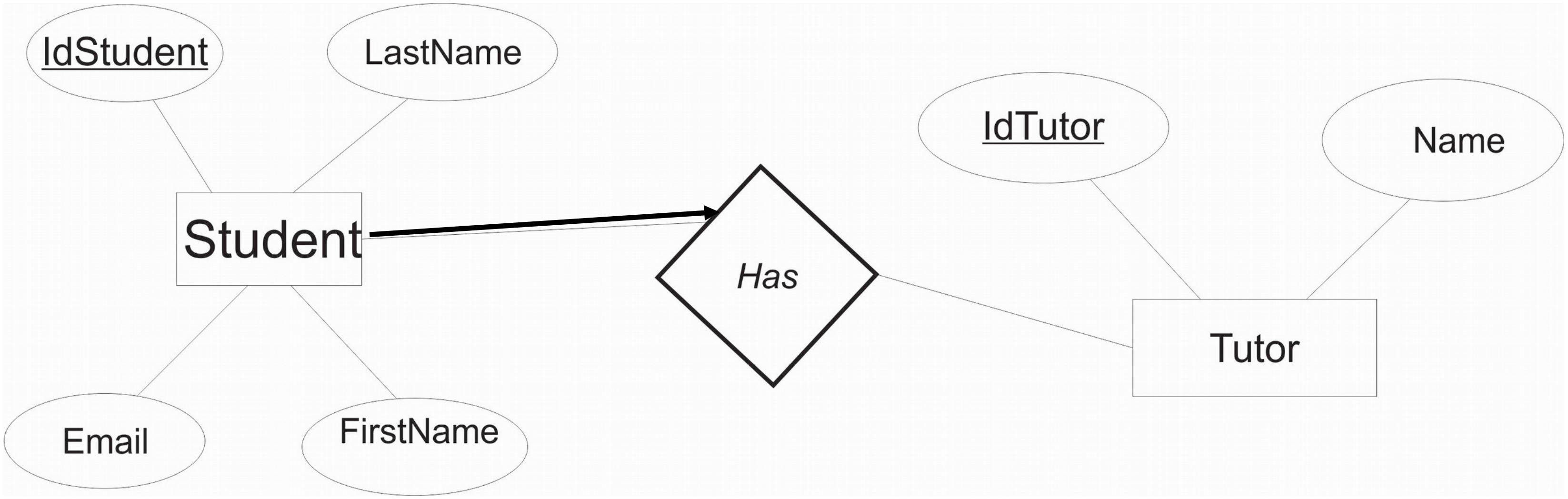
- Include important entities and the relationship between them.
- Do not specify attributes.
- Do not specify primary keys.
- Used as the foundation for logical data models.

Logical Data Model

- Include all entities and relationships between them.
- Specify attributes for each entity.
- Specify primary key for each entity.
- Specify foreign keys, which identify the relationship between different entities.
- Involve normalization



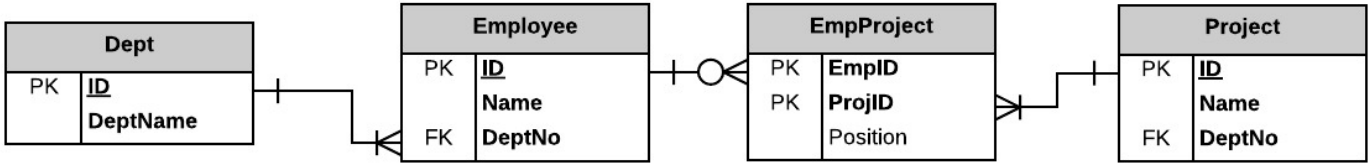
Logical data models



Directionality indicates a constraint

Physical Data Model

- Specify all tables and columns.
- Include foreign keys to identify relationships between tables.
- May include normalization, depending on user requirements.
- May be significantly different from the logical data model.
- Will differ depending on which DBMS (database management system) is used.



Dept	
PK	ID
	DeptName

Employee	
PK	ID
	Name
FK	DeptNo

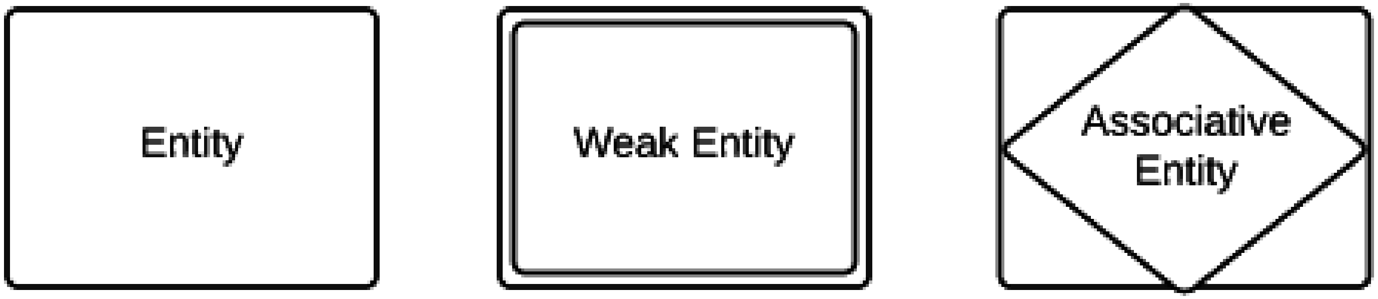
EmpProject	
PK	EmpID
PK	ProjID
	Position

Project	
PK	ID
	Name
FK	DeptNo

Entity

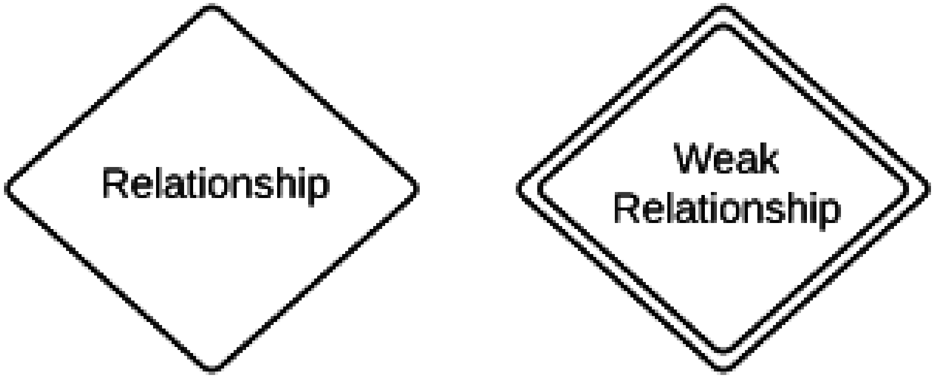
- An Object we identify in our system
- Entities have **attributes** (e.g., Employee has name/dept)
- Some of these attributes may *functionally determine others* (see previous lectures)
- So how do we visually express entities/relationships?

Conceptual Model of Entities



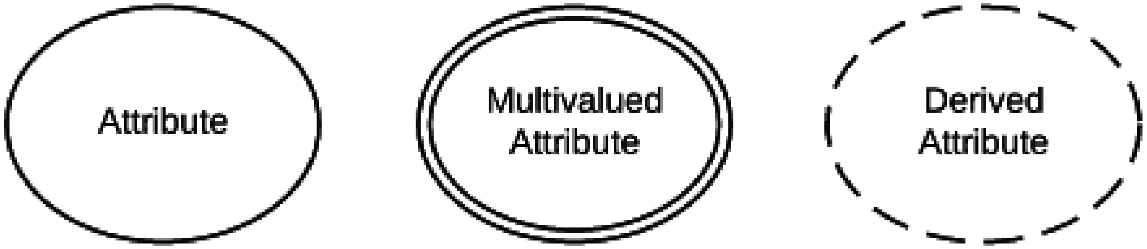
- **Strong entities** exist independently from other entity types. They always possess a key.
- **Weak entities** depend on some other entity type (e.g., Representing Employees in Projects using an EmployeeProject entity)
- **Associative entities** are entities that associate the instances of one or more entity types (e.g., Representing Matches played by Players from a certain Team)

Relationships



- **Relationships** are meaningful associations between or among entities. A relationship provides useful information that **could not be discerned** with just the entity types.
- **Weak relationships**: connections that exist between a weak entity type and its owner (e.g., EmployeeProject)

Attributes

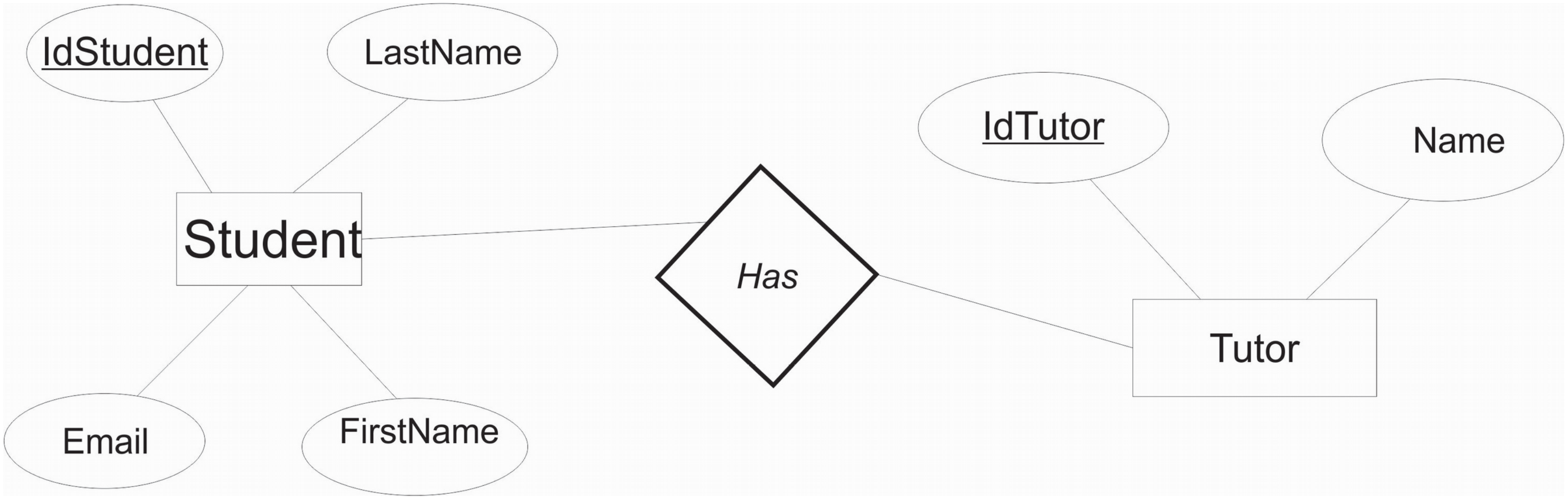


- **Attributes** : characteristics of either an entity, a many-to-many relationship, or a one-to-one relationship.
- **Multivalued attributes**: take on more than one value.
- **Derived attributes**: value can be calculated from related attribute values.

Turning Conceptual into Logical Model

- Primary Keys: underlined
- Foreign Keys: underlined
- Add attributes

Example: Conceptual to Logical



Physical ERDs

- Table structures,
  - column name, column data type, column constraints,
  - primary key, foreign key
  - relationships between tables

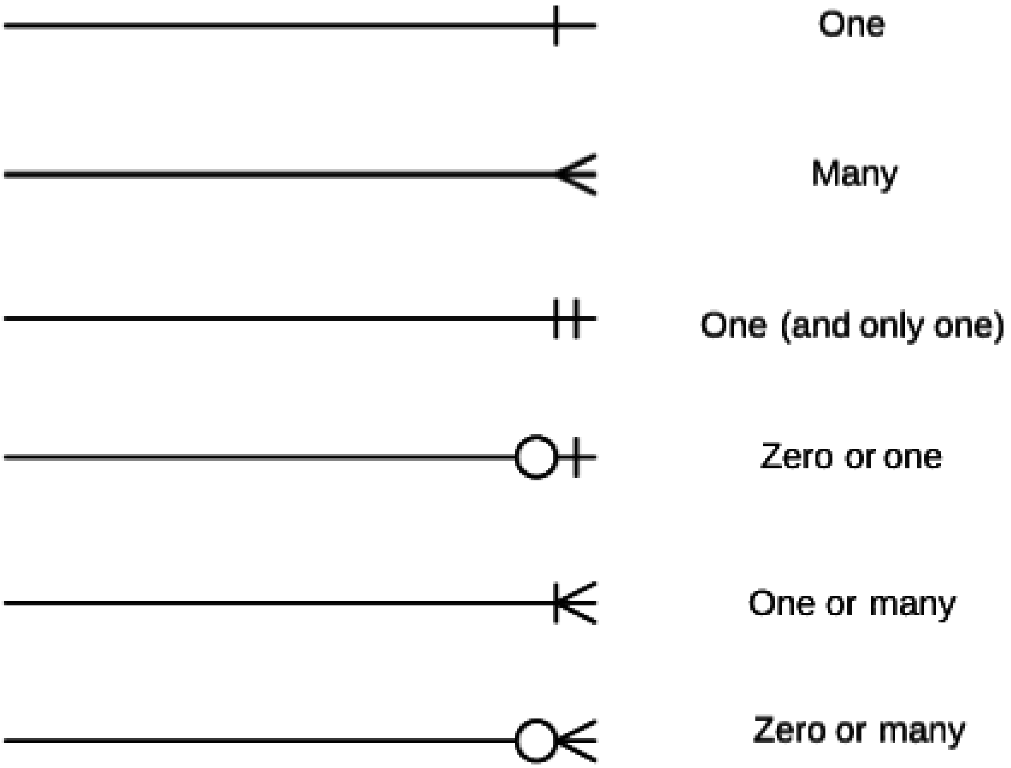
Example: Portfolio of Research Projects

- **Entities**:
  - Employees
  - Research Projects
- **Relationships**:
  - Employees are associated with one dept.
  - Projects have multiple Employees associated, with different roles
  - Team leads

Types of relationships

- Cardinality:
  - **one-to-one** (e.g., an employee has one address)
  - **one-to-many** (e.g., an employee may be involved in multiple projects)
  - **Many-to-many** (e.g., a number of employees may be involved on many projects)
- Identifying/non-Identifying: when foreign key is/not part of the primary key of a child table.
- Mandatory/Optional
  - Employee must have a department
  - Department may have one or more employees
  - An EmployeeProject entity must have 1 employee and 1 project

Relationships
(Crow’s Foot Notation)

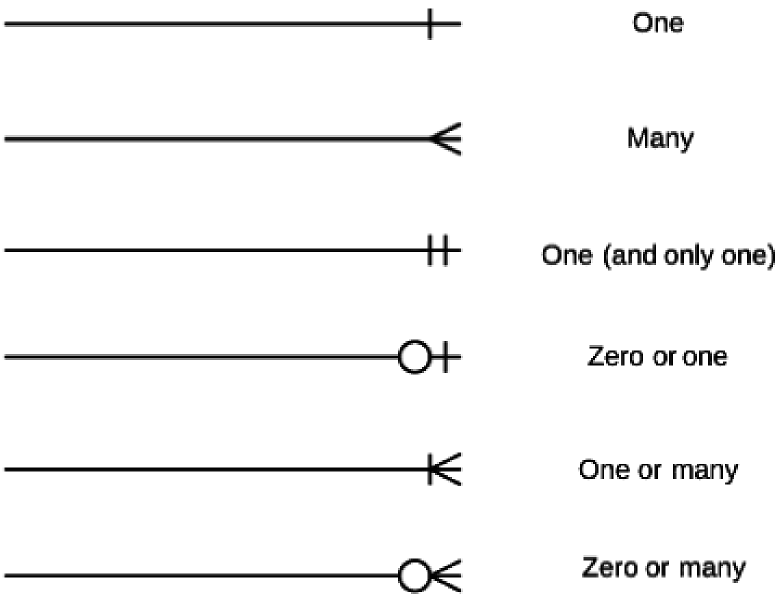


Cardinality and Modality

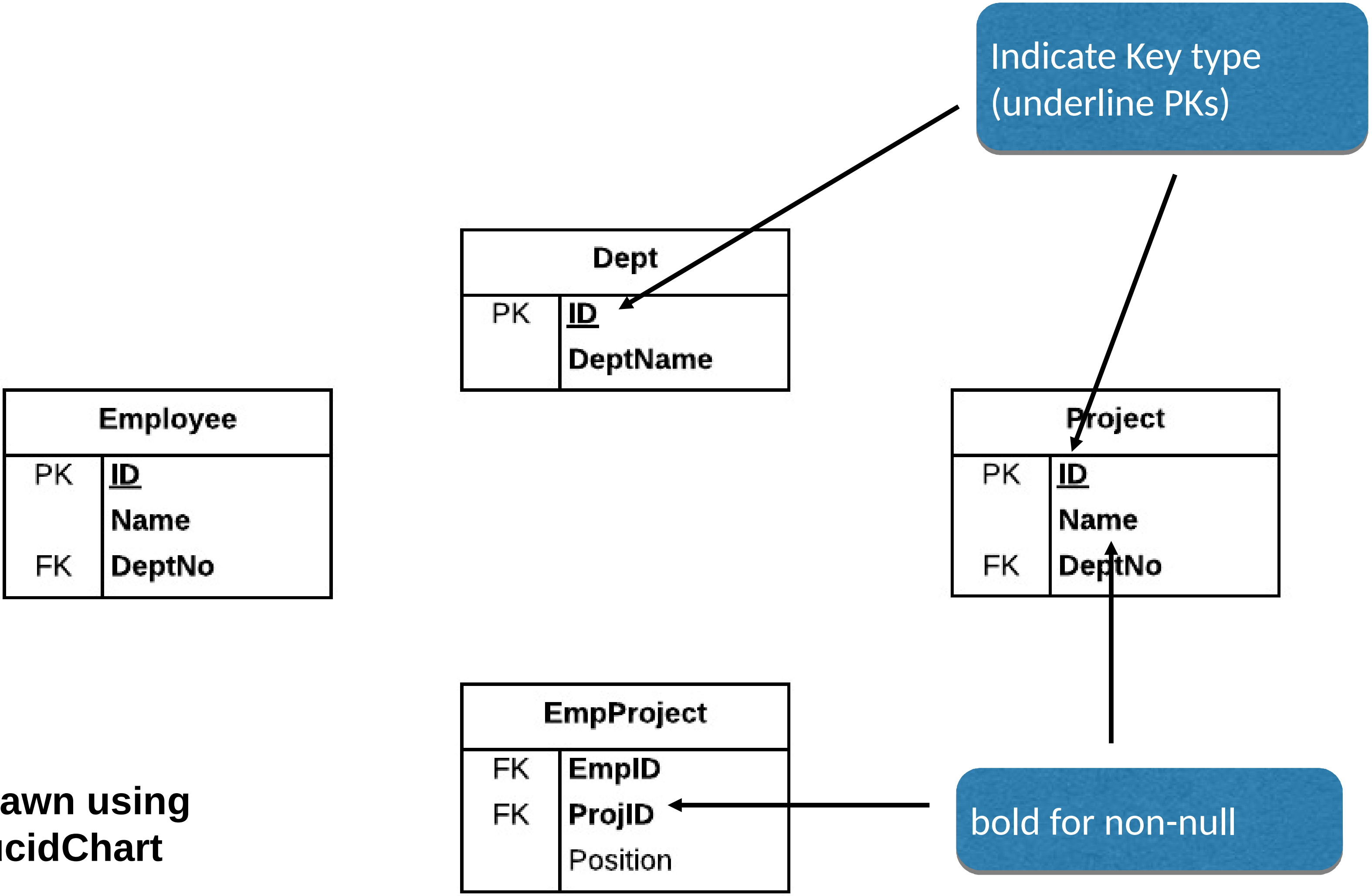
- **Cardinality** and **Modality** are the indicators of the business rules around a relationship.
- **Cardinality** refers to the **maximum number of times** an instance in one entity can be associated with instances in the related entity.
- **Modality** refers to the **minimum number of times** an instance in one entity can be associated with an instance in the related entity.

Modality and Cardinality

- Modality: min
- Cardinality: max



Representing Entities



Indicate Key type (underline PKs)

Dept	
PK	**ID**
	DeptName

Employee	
PK	**ID**
	Name
FK	**DeptNo**

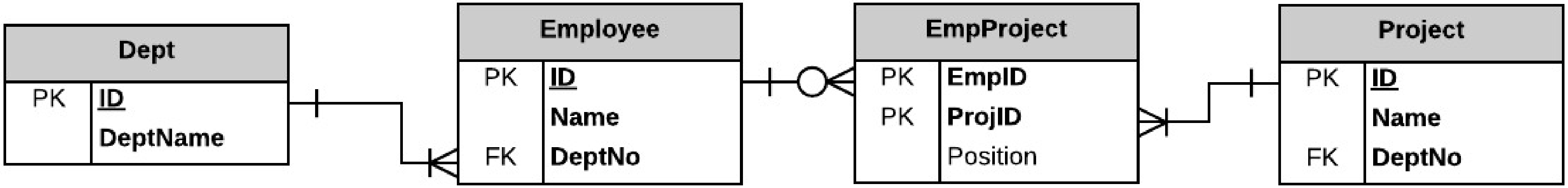
Project	
PK	**ID**
	Name
FK	**DeptNo**

EmpProject	
FK	**EmpID**
FK	**ProjID**
	Position

bold for non-null

Drawn using LucidChart

An ERD for Research Projects



Dept

PK	ID
	DeptName

Employee

PK	ID
	Name
FK	DeptNo

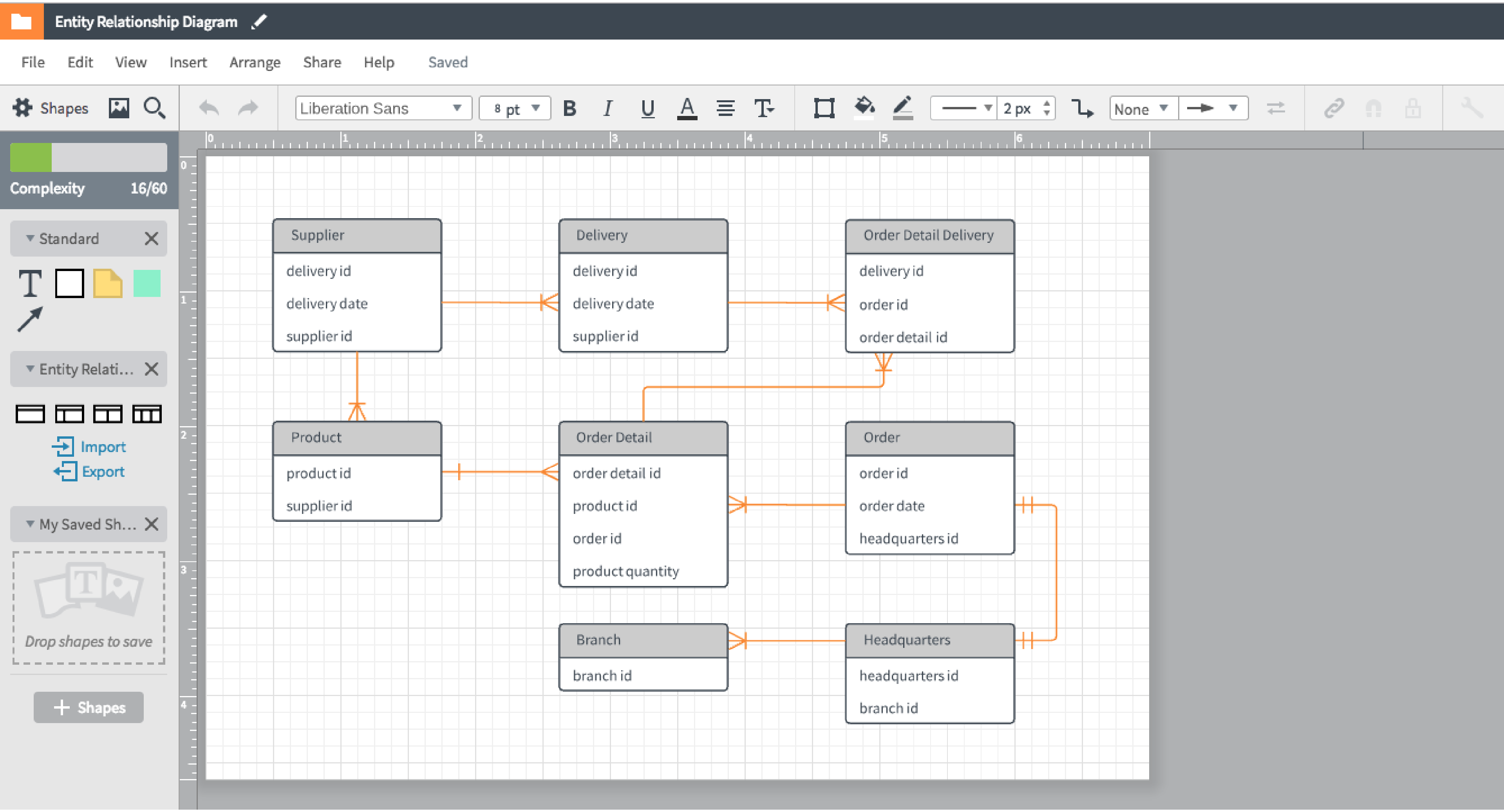
EmpProject

PK	EmpID
PK	ProjID
	Position

Project

PK	ID
	Name
FK	DeptNo

Making ERDs



Supplier
delivery id
delivery date
supplier id

Delivery
delivery id
delivery date
supplier id

Order Detail Delivery
delivery id
order id
order detail id

Product
product id
supplier id

Order Detail
order detail id
product id
order id
product quantity

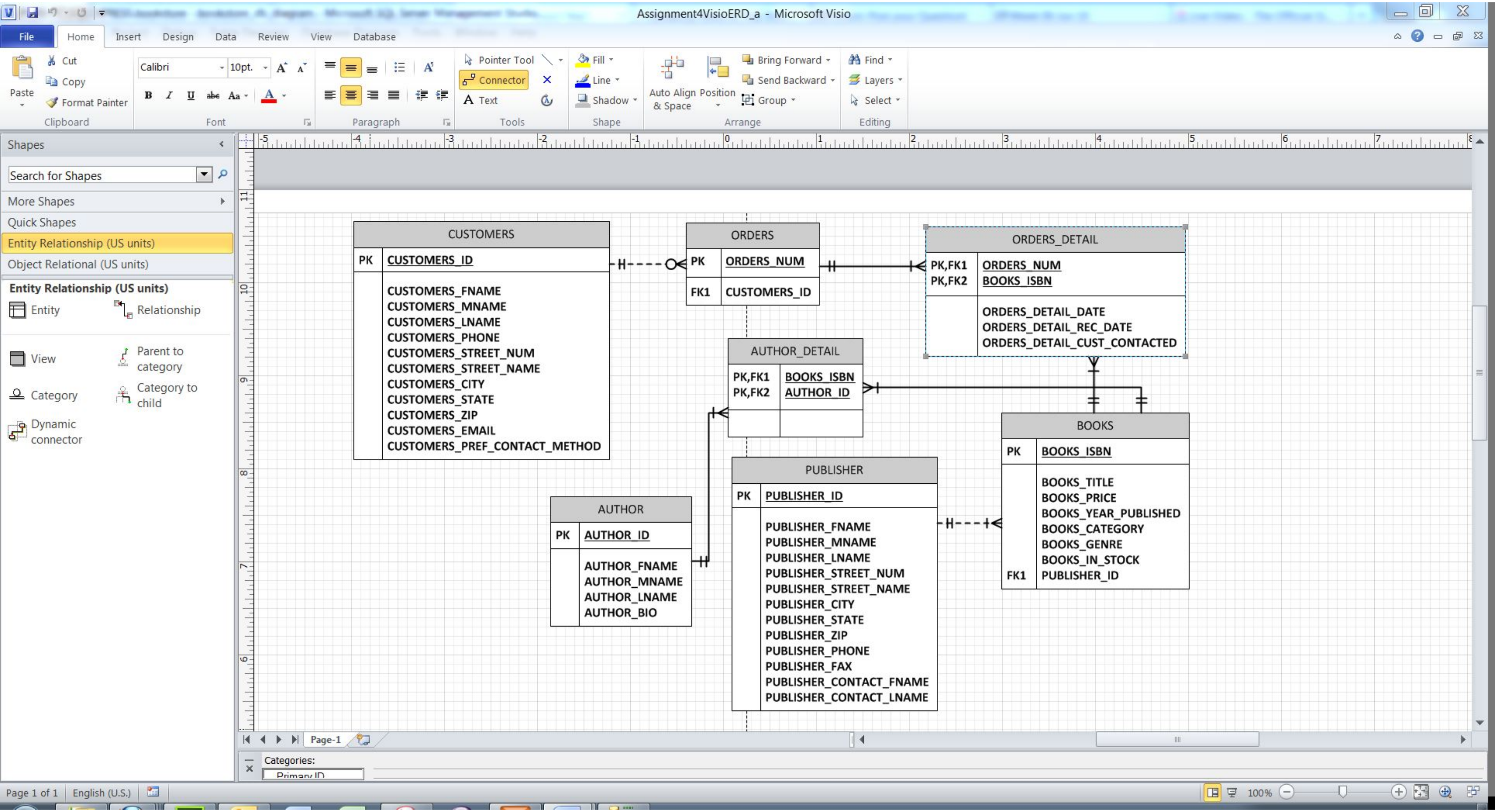
Order
order id
order date
headquarters id

Branch
branch id

Headquarters
headquarters id
branch id

https://www.lucidchart.com

Making ERDs



CUSTOMERS

PK	CUSTOMERS_ID
	CUSTOMERS_FNAME CUSTOMERS_MNAME CUSTOMERS_LNAME CUSTOMERS_PHONE CUSTOMERS_STREET_NUM CUSTOMERS_STREET_NAME CUSTOMERS_CITY CUSTOMERS_STATE CUSTOMERS_ZIP CUSTOMERS_EMAIL CUSTOMERS_PREF_CONTACT_METHOD

ORDERS

PK	ORDERS_NUM
FK1	CUSTOMERS_ID

ORDERS_DETAIL

PK,FK1 PK,FK2	ORDERS_NUM BOOKS_ISBN
	ORDERS_DETAIL_DATE ORDERS_DETAIL_REC_DATE ORDERS_DETAIL_CUST_CONTACTED

AUTHOR_DETAIL

PK,FK1 PK,FK2	BOOKS_ISBN AUTHOR_ID

BOOKS

PK	BOOKS_ISBN
	BOOKS_TITLE BOOKS_PRICE BOOKS_YEAR_PUBLISHED BOOKS_CATEGORY BOOKS_GENRE BOOKS_IN_STOCK
FK1	PUBLISHER_ID

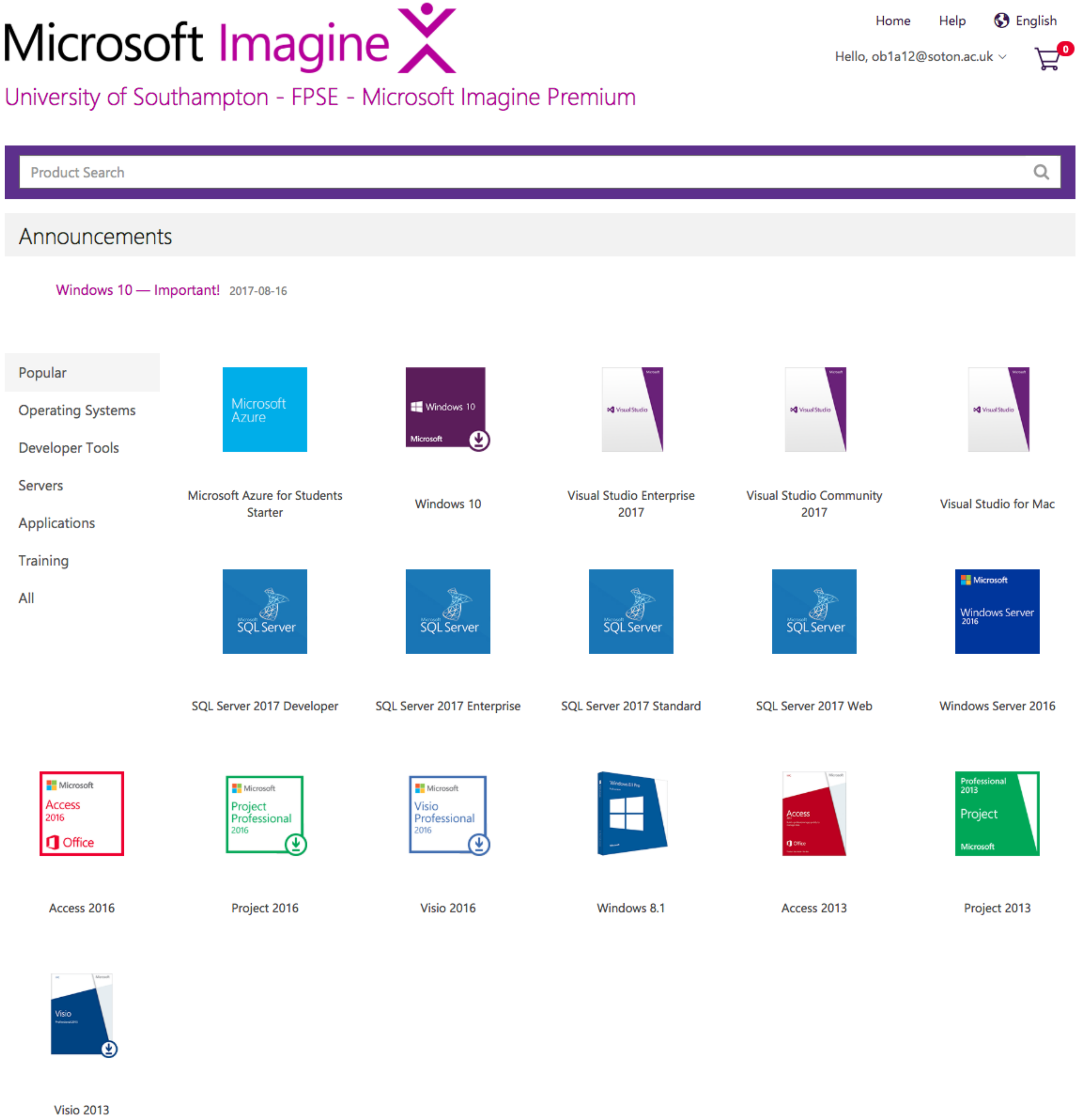
PUBLISHER

PK	PUBLISHER_ID
	PUBLISHER_FNAME PUBLISHER_MNAME PUBLISHER_LNAME PUBLISHER_STREET_NUM PUBLISHER_STREET_NAME PUBLISHER_CITY PUBLISHER_STATE PUBLISHER_ZIP PUBLISHER_PHONE PUBLISHER_FAX PUBLISHER_CONTACT_FNAME PUBLISHER_CONTACT_LNAME

AUTHOR

PK	AUTHOR_ID
	AUTHOR_FNAME AUTHOR_MNAME AUTHOR_LNAME AUTHOR_BIO

Visio is available on vdi.soton.ac.uk, lab machines and from http://ecs.gg/imagine



Microsoft Imagine

Home Help English

Hello, ob1a12@soton.ac.uk

University of Southampton - FPSE - Microsoft Imagine Premium

Product Search

Announcements

Windows 10 — Important! 2017-08-16

Popular

Operating Systems

Developer Tools

Servers

Applications

Training

All

Microsoft Azure for Students Starter

Windows 10

Visual Studio Enterprise 2017

Visual Studio Community 2017

Visual Studio for Mac

SQL Server 2017 Developer

SQL Server 2017 Enterprise

SQL Server 2017 Standard

SQL Server 2017 Web

Windows Server 2016

Access 2016

Project 2016

Visio 2016

Windows 8.1

Access 2013

Project 2013

Visio 2013

From Modelling to SQL

What is SQL

- **SQL: Structured query language**
- Specifies a **Data Definition Language (DDL)**
  - Tables and views (virtual tables).
  - Convert a data model to a (physical) database
- Specifies a **Data manipulation (DML)**
  - Programmatic data manipulation
  - Declarative (desired result)
  - INSERT, DELETE, UPDATE or retrieve (SELECT) data.
- Enforces Data integrity:
  - Referential integrity
  - Transactions
  - Checks keys for consistency
- Access control: security
- Data sharing: by concurrent users

SQL vs Programming Languages

- Restricted language : about 30 sub statements
- Restricted number of operations -> consistency across systems
- Hides low-level data operations
- Focus on programming data manipulation rather than control loops

SQL

- **Data definition** : define tables and views.
- **Data query** : extract data, add data and delete data.
- **Administration** : grant permissions to users to perform operations on our database.