



## Introduction to Database

Dr Simon Jones – [simon.jones@nyulangone.org](mailto:simon.jones@nyulangone.org)  
Thanks to Mariam Mohaideen



- **Today – database theory**
  - Key learning outcome - is to understand data normalization
- **Thursday, 19 November**
  - Introduction to SQL & MySQL practical
  - Key learning outcome - write SQL queries
- **Tuesday, 24 November**
  - R and MySQL
  - Key learning outcome – call SQL from R. Use SQLDF in R to manage large datasets

ACID is a set of requirements of database systems intended to guarantee validity even in the event of:

- power failures,
- Errors
- etc..

**Atomicity:** All or nothing.

**Consistency:** Consistent state of data and transactions.

**Isolation:** Transactions are isolated from each other.

**Durability:** When the transaction is committed, state will be durable.

By giving up ACID properties, one can achieve higher performance and scalability.



# Analysis and Modeling

- ✓ Explain the role of conceptual data modeling in the overall analysis and design of an information system.

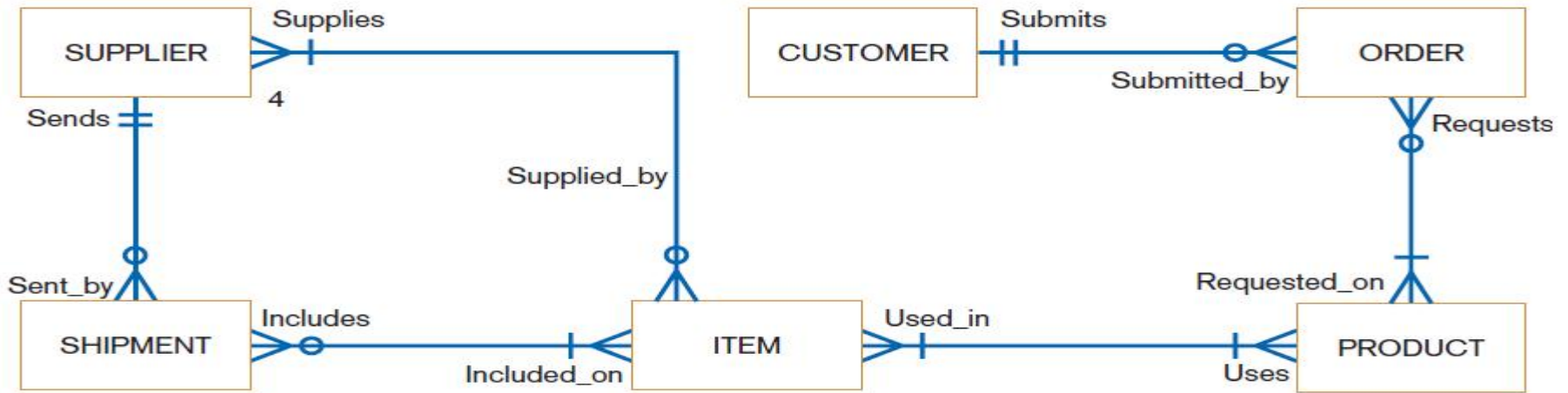
- Conceptual data modeling:** a detailed model that captures the overall structure of data in an organization
- Independent of any database management system (DBMS) or other implementation considerations

Entity-relationship (E-R) diagram or UML class diagram

- Entities (or classes) – categories of data, represented as rectangles
- Relationships (or associations) – lines between the entities

Set of entries about data objects to be stored in repository project dictionary, or data modeling software

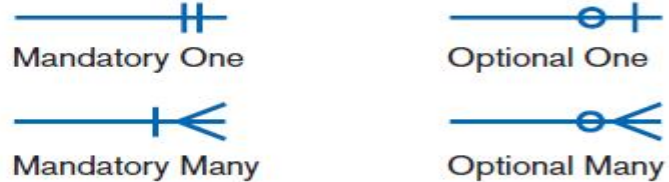
- Repository links data, process, and logic models of an information system.
- Data elements included in the data flow diagram (DFD) must appear in the data model and vice versa.
- Each data store in a process model must relate to business objects represented in the data model.



Key



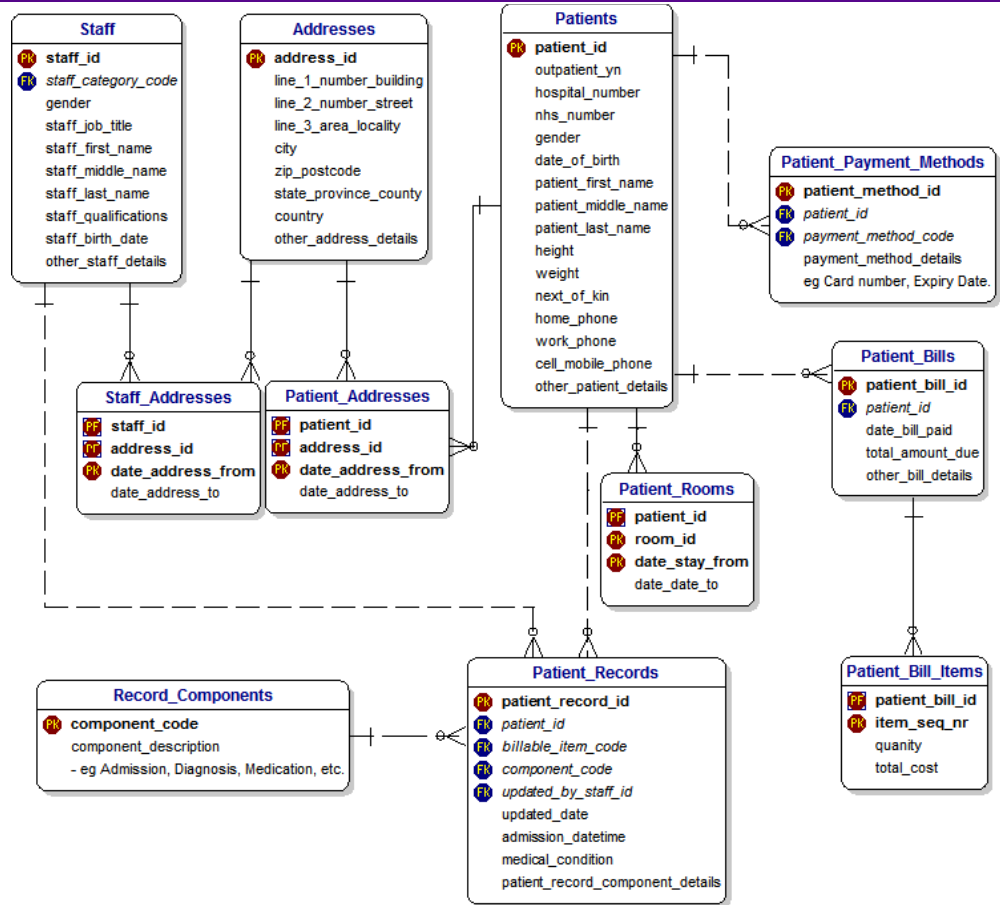
Cardinalities



**FIGURE 8-3**

Sample conceptual data model





**Entity:** a person, place, object, event or concept in the user environment about which data is to be maintained

**Entity type:** collection of entities that share common properties or characteristics

**Entity instance:** single occurrence of an entity type

**Entity-Relationship data model (E-R model):** a detailed, logical representation of the entities, associations and data elements for an organization or business area

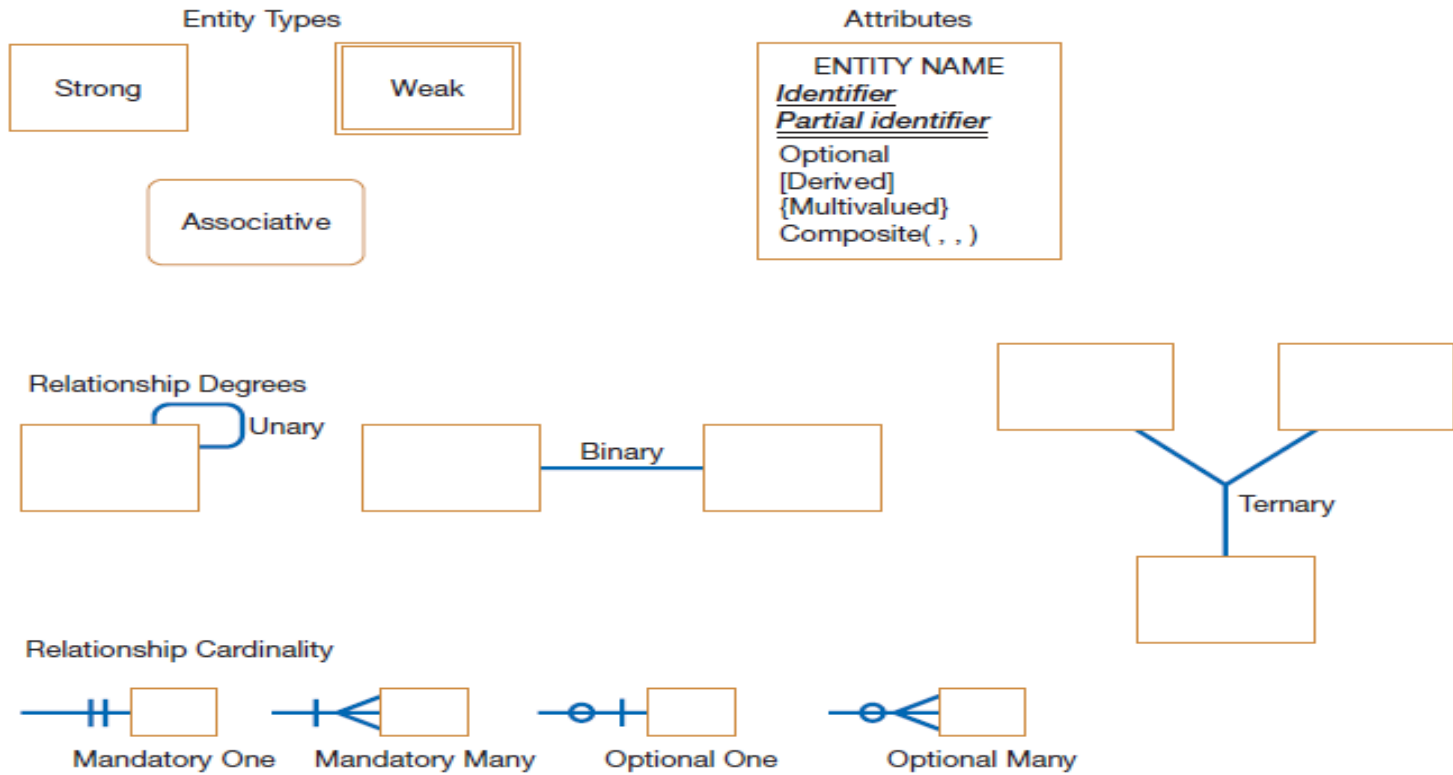
**Entity-relationship diagram (E-R diagram):** a graphical representation of an E-R model

# Requirements Determination Questions for Data Modeling:

- What are subjects/objects of the business?
  - Data entities and descriptions
- What unique characteristics distinguish between subjects/objects of the same type?
  - Primary keys

- What characteristics describe each subject/object?
  - **Attributes and secondary keys**
- How do you use the data?
  - **Security controls and user access privileges**
  - **Who knows the meaning of the data?**
- Over what period of time are you interested in the data?
  - **Cardinality and time dimensions**

- Are all instances of each object the same?
  - **Supertypes, subtypes, and aggregations**
- What events occur that imply associations between objects?
  - **Relationships and cardinalities**
- Are there special circumstances that affect the way events are handled?
  - **Integrity rules, minimum and maximum cardinalities, time dimensions**



**FIGURE 8-5** Basic E-R notation

An entity type name should be:

- *A singular noun.*
- *Descriptive and specific to the organization.*
- *Concise.*

*Event entity type* should be named for the *result of the event*, not the activity or process of the event.



## An entity type definition:

- Includes a statement of *what the unique characteristic(s) is (are) for each instance*.
- Makes clear *what entity instances are included and not included* in the entity type.
- Often includes a description of *when an instance of the entity type is created or deleted*.

**Attribute:** a named property or characteristic of an entity that is of interest to the organization

- Naming an attribute: i.e. Vehicle\_ID
- Place its name inside the rectangle for the associated entity in the E-R diagram.

An attribute name is a *noun* and should be *unique*.

To make an attribute name unique and for clarity, *each attribute name should follow a standard format*.

*Similar attributes of different entity types should use similar but distinguishing names.*

## An attribute definition:

- *States what the attribute is and possibly why it is important.*
- *Should make it clear what is included and what is not included.*
- *Contains any aliases or alternative names.*
- *States the source of values for the attribute.*

An attribute definition should indicate:

- *If a value for the attribute is required or optional.*
- *If a value for the attribute may change.*
- *Any relationships that attribute has with other attributes.*

**Candidate key:** an attribute (or combination of attributes) that uniquely identifies each instance of an entity type

**Identifier:** a candidate key that has been selected as the unique, identifying characteristic for an entity type

## Selection rules for an identifier

- Choose a candidate key that will not change its value.
- Choose a candidate key that will never be null.
- Avoid using intelligent keys.
- Consider substituting single value surrogate keys for large composite keys.

**Question: Is the vehicle number plate a good identifier for a car?**

**Question: Is the New York State License Number a good identifier for a nurse?**

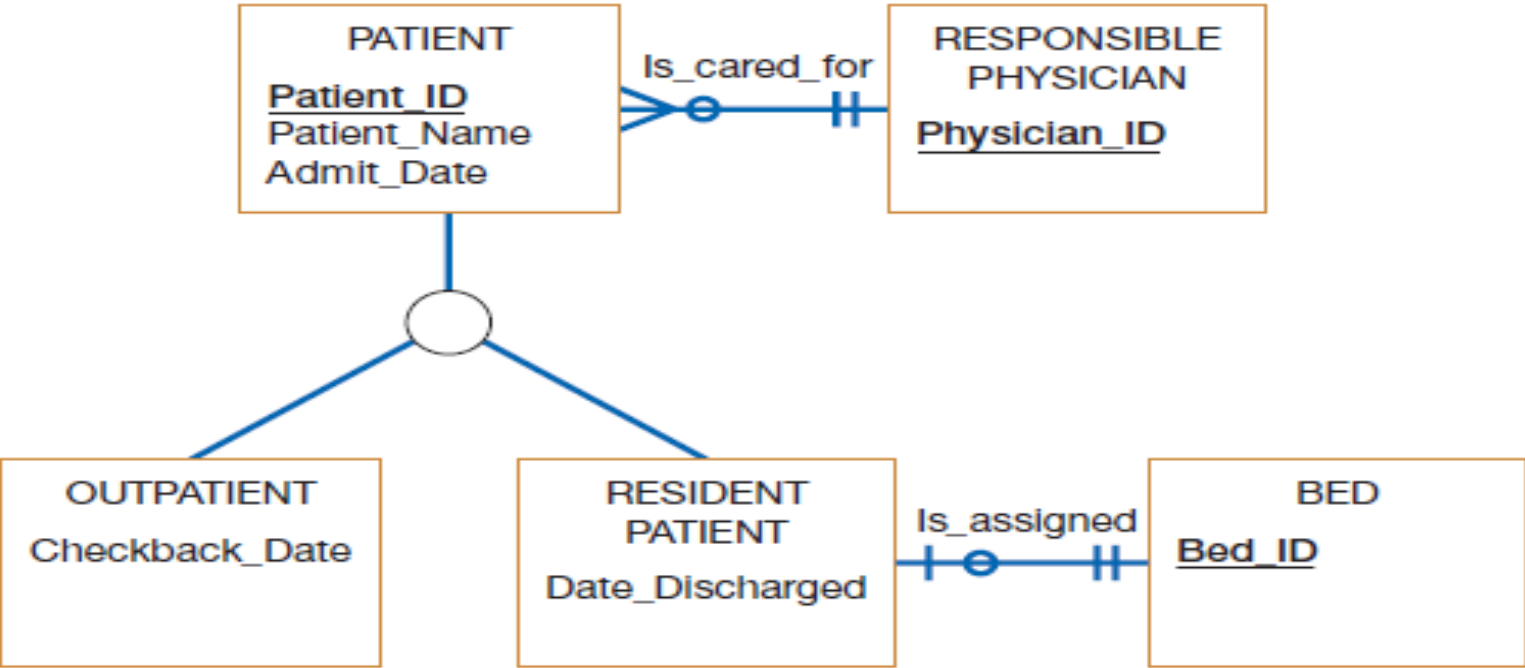
**Relationship:** an association between the instances of one or more entity types that is of interest to the organization

**Degree:** the number of entity types that participate in a relationship



# Figure 8-18

## Supertype/subtype relationships in a hospital

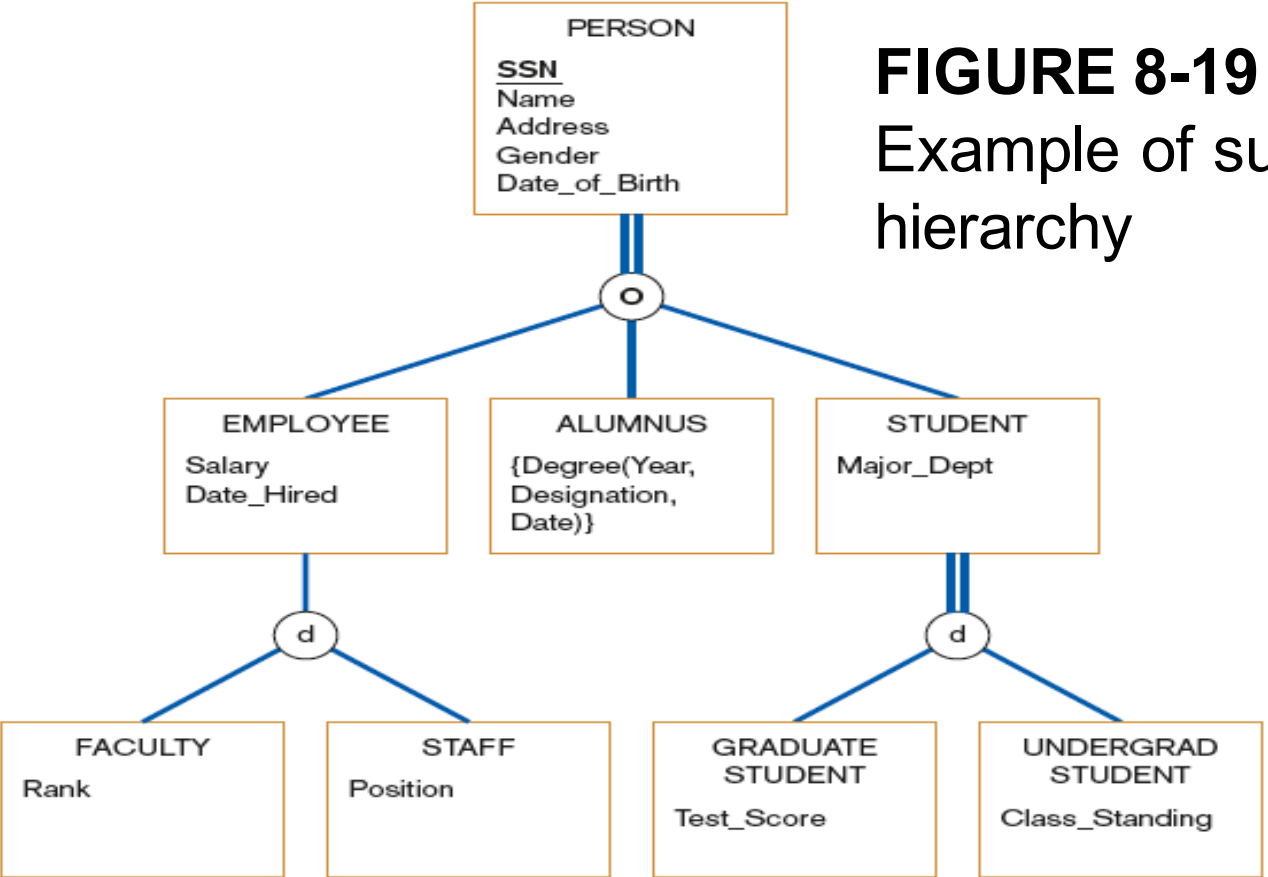


## Business Rules for Supertype/subtype Relationships:

- **Total specialization** specifies that each entity instance of the supertype must be a member of some subtype in the relationship.
- **Partial specialization** specifies that an entity instance of the supertype does not have to belong to any subtype, and may or may not be an instance of one of the subtypes.

- **Disjoint rule** specifies that if an entity instance of the supertype is a member of one subtype, it cannot simultaneously be a member of any other subtype.
- **Overlap rule** specifies that an entity instance can simultaneously be a member of two (or more) subtypes.

**FIGURE 8-19**  
Example of supertype/subtype hierarchy



**Associative Entity:** an entity type that associates the instances of one or more entity types and contains attributes that are peculiar to the relationship between those entity instances

- Sometimes called a gerund

The data modeler chooses to model the relationship as an entity type.

# FIGURE 8-15 An associative entity



Attribute on a relationship



An associative entity (CERTIFICATE)



An associative entity using Microsoft Visio®

The purpose of E-R diagramming is to capture the richest possible understanding of the meaning of the data necessary for an information system or organization.

**Subtype:** a subgrouping of the entities in an entity type

- Is meaningful to the organization
- Shares common attributes or relationships distinct from other subgroupings

**Supertype:** a generic entity type that has a relationship with one or more subtypes



**Field** – the smallest unit of meaningful data to be stored in a database

- the physical implementation of a data attribute

**Primary key** – a field that uniquely identifies a record.

**Secondary key** – a field that identifies a single record or a subset of related records.

**Foreign key** – a field that points to records in a different file.

**Descriptive field** – any nonkey field.

**Record** – a collection of fields arranged in a predetermined format.

- Fixed-length record structures
- Variable-length record structures

**Blocking factor** – the number of logical records included in a single read or write operation (from the computer's perspective).

**File** – the set of all occurrences of a given record structure.

**Table** – the relational database equivalent of a file.

**Master files** – Records relatively permanent though values may change

**Transaction files** – Records describe business events

**Document files** – Historical data for review without overhead of regenerating document

**Archival files** – Master and transaction records that have been deleted

**Table lookup files** – Relatively static data that can be shared to maintain consistency

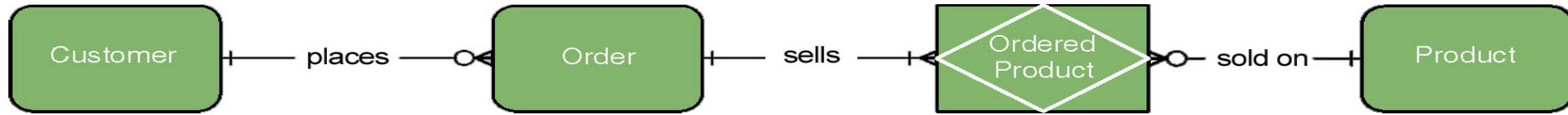
**Audit files** – Special records of updates to other files

**Data architecture** – a definition of how:

- Files and databases are to be developed and used to store data
- The file and/or database technology to be used
- The administrative structure set up to manage the data resource

Data is stored in some combination of:

- **Conventional files**
- **Operational databases** – databases that support day-to-day operations and transactions for an information system. Also called transactional databases.
- **Data warehouses** – databases that store data extracted from operational databases.
  - To support data mining
- **Personal databases**
- **Work group databases**





### Customers Table

Customer Number (primary key)	Customer Name	Customer Balance	...
10112	Luck Star	1455.77	
10113	Pemrose	12.14	
10114	Hartman	0.00	
10117	K-Jack Industries	- 20.00	

### Orders Table

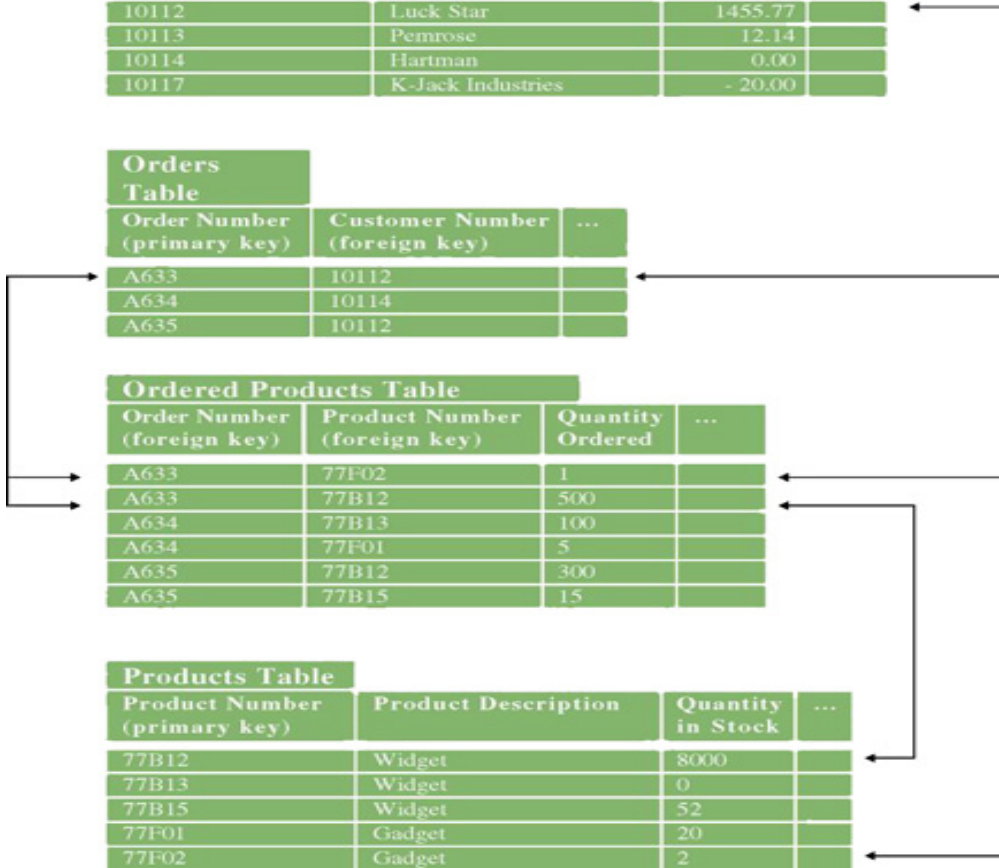
Order Number (primary key)	Customer Number (foreign key)	...
A633	10112	
A634	10114	
A635	10112	

### Ordered Products Table

Order Number (foreign key)	Product Number (foreign key)	Quantity Ordered	...
A633	77F02	1	
A633	77B12	500	
A634	77B13	100	
A634	77F01	5	
A635	77B12	300	
A635	77B15	15	

### Products Table

Product Number (primary key)	Product Description	Quantity in Stock	...
77B12	Widget	8000	
77B13	Widget	0	
77B15	Widget	52	
77F01	Gadget	20	
77F02	Gadget	2	



A good data model is simple

- The data attributes that describe an entity should describe only that entity

A good data model is essentially nonredundant

- Each data attribute exists in at most one entity (except for foreign keys)

A good data model should be flexible and adaptable to future needs

*These goals are achieved through database normalization.*

# Goals of Database Design

A database should provide for efficient storage, update, and retrieval of data.

A database should be reliable—the stored data should have high integrity and promote user trust in that data.

A database should be adaptable and scalable to new and unforeseen requirements and applications.

A database should support the business requirements of the information system.

Database schema – a model or blueprint representing the technical implementation of the database.

- Also called a physical data model

Can be a single attribute or composite attribute.

Can be called **identifier**.

Weak entity may have no key.

Has the following criteria:

- Should not change its value
- Not null
- Avoid intelligent keys.
- Substitute large composite keys with surrogate keys (system generated keys for unique numbers).

# Why Database Normalization?

Eliminate redundancy

Organize data efficiently

Reduce the potential for data anomalies.

inconsistencies in the data stored in a database as a result of an operation such as

- update,
- insertion
- deletion.

Inconsistencies may arise when a record is stored in multiple locations and not all of the copies are updated.

Normalization help reduce Data Anomalies

The values in each column of a table are atomic (No multi-value attributes allowed).

There are no repeating groups: two columns do not store similar information in the same table.

(Each table has a primary key: minimal set of attributes which can uniquely identify a record)





# 1<sup>st</sup> Normal Form Example

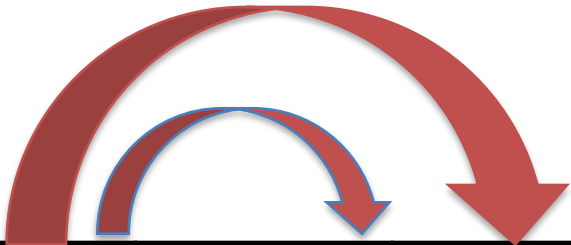
Un-normalized Students table:

<u>Student#</u>	TutorID	TutorName	TutorRoom	ClassList
123	123A	James	555	102-8, 104-9
124	123B	Smith	467	209-0, 102-8

1<sup>st</sup> Normal Form Students table :

<u>Student#</u>	TutorID	TutorName	TutorRoom	Class
123	123A	James	555	102-8
123	123A	James	555	104-9
124	123B	Smith	467	209-0
124	123B	Smith	467	102-8

# Functional Dependence



<u>Student#</u>	TutorID	TutorName	TutorRoom	ClassList
123	123A	James	555	102-8, 104-9
124	123B	Smith	467	209-0, 102-8

All requirements for 1<sup>st</sup> NF must be met.

Redundant data across multiple rows of a table must be moved to a separate table.

- The resulting tables must be related to each other by use of foreign key.
- In 1NF AND every non-prime attribute of the relation is dependent on the whole of every candidate key.



Students table

<u>StudentID</u>	TutorID	TutorName	TutorRoom
123	123A	James	555
124	123B	Smith	467

Registration table

<u>StudentID</u>	ClassID
123	102-8
123	104-9
124	209-0
124	102-8

All requirements for 2<sup>nd</sup> NF must be met.

Eliminate fields that do not depend on the primary key;

- i.e. any field that is dependent not only on the primary key but also on another field must be moved to another table.



# 3<sup>rd</sup> Normal Form Example

Students table:

<u>StudentID</u>	TutorID
123	123A
124	123B

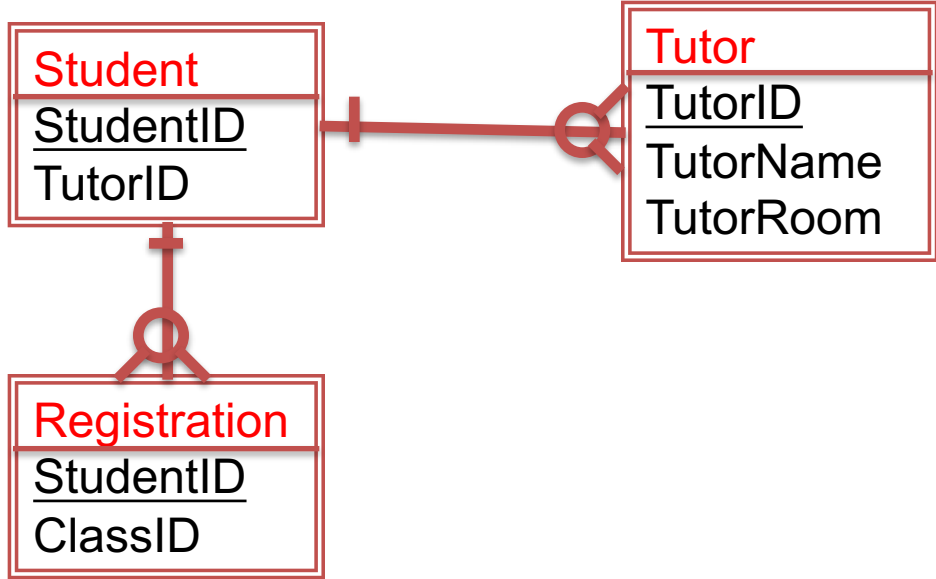
Registration table:

<u>StudentID</u>	ClassID
123	102-8
123	104-9
124	209-0
124	102-8

Tutor table:

<u>TutorID</u>	TutorName	TutorRoom
123A	James	555
123B	Smith	467

# What's Missing?



**Student**

StudentID

AddressLine1

AddressLine2

City

State

Zip Code

MobilePhone

SponsorName

SponsorContact

SponsorPhoneNumber

US / Non-US Student

Exercise – Put  
this table in 3<sup>rd</sup>  
Normal Form



Patient ID

Patient First Name

Patient Last Name

Patient Address

Admission Date

Discharge Date

Attending Physician

Attending Physician Office Number

Attending Physician Phone

Drug Prescribed

Drug Prescribed By

Date Drug Prescribed

Drug Dosage

Customer ID (primary key)	Address 1
Customer Name	Address 2
Customer Type	City
Contact Name 1	State
Contact Name Role 1	Zip Code
Contact Name 2	Country
Contact Name Role 2	
Order Date	
Order Item 1	
Order Item 2	
Delivery Date	
Person Who Singed for Delivery	
Person Who Singed for Delivery Role	

**Data distribution analysis** establishes which business locations need access to which logical data entities and attributes.

## Centralization

- Entire database on a single server in one physical location

## Horizontal distribution (also called partitioning)

- Tables or row assigned to different database servers/locations.
- Efficient access and security
- Cannot always be easily recombined for management analysis

## Vertical distribution (also called partitioning)

- Specific table columns assigned to specific databases/servers
- Similar advantages and disadvantages of Horizontal

## Replication

- Data duplicated in multiple locations
- DBMS coordinates updates and synchronization
- Performance and accessibility advantages
- Increases complexity

For each table sum the *field sizes*. This is the *record size*.

For each table, multiply the *record size* times the number of entity instances to be included in the table (planning for growth). This is the *table size*.

Sum the *table sizes*. This is the *database size*.

Optionally, add a slack capacity buffer (e.g. 10 percent) to account for unanticipated factors. This is the *anticipated database capacity*.

# Further Reading

Hoffer JA, George JF, Valacich JS (2014) Modern Systems Analysis and Design, 7th Edition, Pearson