



Databases - 1

BS3033 Data Science for Biologists

Dr Wilson Goh

School of Biological Sciences

Learning Objectives

By the end of this topic, you should be able to:

- Explain the generic design considerations for databases.
- Explain database normalisation.
- Explain relational, flatfile and XML database models.





**NANYANG
TECHNOLOGICAL
UNIVERSITY**
SINGAPORE

Database and its Design Considerations

BS3033 Data Science for Biologists

Dr Wilson Goh

School of Biological Sciences

What is a Database?

Structured collection of information.

Consists of basic units called **records** or entries.

Each record consists of fields, which hold **pre-defined** data related to the record.

For example, a protein database would have protein entries as records and protein properties as fields (e.g., name of protein, length, amino-acid sequence).

Database Design

Designing an efficient, useful database is a matter of following the proper process, including these phases:

- Requirements analysis, or identifying the purpose of your database
- Organising data into tables
- Specifying primary keys and analysing relationships
- Normalising to standardise the tables



**NANYANG
TECHNOLOGICAL
UNIVERSITY**
SINGAPORE

Requirements Analysis

BS3033 Data Science for Biologists

Dr Wilson Goh

School of Biological Sciences

Identifying the Purpose of the Database

- Understanding the purpose of your database will inform your choices throughout the design process.
- Make sure you consider the database from every perspective. For instance, if you were making a database for a public library, you'd want to consider the ways in which both patrons and librarians would need to access the data.

Identifying the Purpose of the Database

Here are some ways to gather information before creating the database:

- Interview the people who will use it
- Analyse business forms, such as invoices, timesheets, surveys
- Comb through any existing data systems (including physical and digital files)

Identifying the Purpose of the Database

Start by gathering any existing data that will be included in the database. Then list the types of data you want to store and the entities, or people, things, locations, and events, that those data describe, like:

Customers	
	Name
	Address
	City, State, Zip
	Email Address

Products	
	Name
	Price
	Quantity in Stock
	Quantity on Order

Orders	
	Order ID
	Sales Representative
	Date
	Product(s)
	Quantity
	Price
	Total

Be sure to break down the information into the smallest useful pieces. For instance, consider separating the street address from the country so that you can later filter individuals by their country of residence. Also, avoid placing the same data point in more than one table, which adds unnecessary complexity.



**NANYANG
TECHNOLOGICAL
UNIVERSITY**
SINGAPORE

Organising Data

BS3033 Data Science for Biologists

Dr Wilson Goh

School of Biological Sciences

Organising Data into Tables

Lay out a visual representation of your database.

To do that, you need to understand how relational databases are structured.

Within a database, related data are grouped into tables, each of which consists of rows (also called tuples) and columns, like a spreadsheet.

To convert your lists of data into tables, start by creating a table for each type of entity, such as products, sales, customers, and orders.

Organising Data into Tables

An example of the **Customers Table**

First Name	Last Name	Age	Zip Code
Roger	Williams	43	34760
Jerrica	Jorgensen	32	97453
Samantha	Hopkins	56	64829

Columns (also known as fields or attributes) contain a single type of information that appears in each record.

Each row of a table is called a record. Records include data about something or someone, such as a particular customer.

Organising Data into Tables

An example of the **Customers Table**

First Name	Last Name	Age	Zip Code
Roger	Williams	43	34760
Jerrica	Jorgensen	32	97453
Samantha	Hopkins	56	64829

To enforce consistency, each column can be assigned to only hold data of a particular type. Other data types:

CHAR: a specific length of text

TEXT: large amounts of text

FLOAT, DOUBLE: floating point numbers

BLOB: binary data

VARCHAR: text of variable lengths

INT: positive or negative whole number

Organising Data into Tables

An example of the **Customers Table**

First Name	Last Name	Age	Zip Code
Roger	Williams	43	34760
Jerrica	Jorgensen	32	97453
Samantha	Hopkins	56	64829

Not sufficiently unique. Assigning a unique user identifier or username would be better. A primary key (PK) is a **unique identifier** for a given entity (Table), meaning that you could pick out an exact customer even if you only knew that value.

Decide which **attribute** or **attributes** will serve as the **primary key** for each table, if any. Attributes chosen as primary keys should be **unique, unchanging, and always present** (never NULL or empty).

Organising Data into Tables

When it comes time to create the actual database, you'll put both the logical data structure and the physical data structure into the data definition language supported by your database management system.

At that point, you should also estimate the size of the database to be sure you can get the performance level and storage space it will require.



**NANYANG
TECHNOLOGICAL
UNIVERSITY**
SINGAPORE

Creating Relationships between Data Tables

BS3033 Data Science for Biologists

Dr Wilson Goh

School of Biological Sciences

Creating Relationships between Tables/ Entities

With your data now broken down into tables, you're ready to analyse the relationships between those tables.

Customers

Name

Address

City, State, Zip

Email Address

Products

Name

Price

Quantity in Stock

Quantity on Order

Orders

Order ID

Sales Representative

Date

Product(s)

Quantity

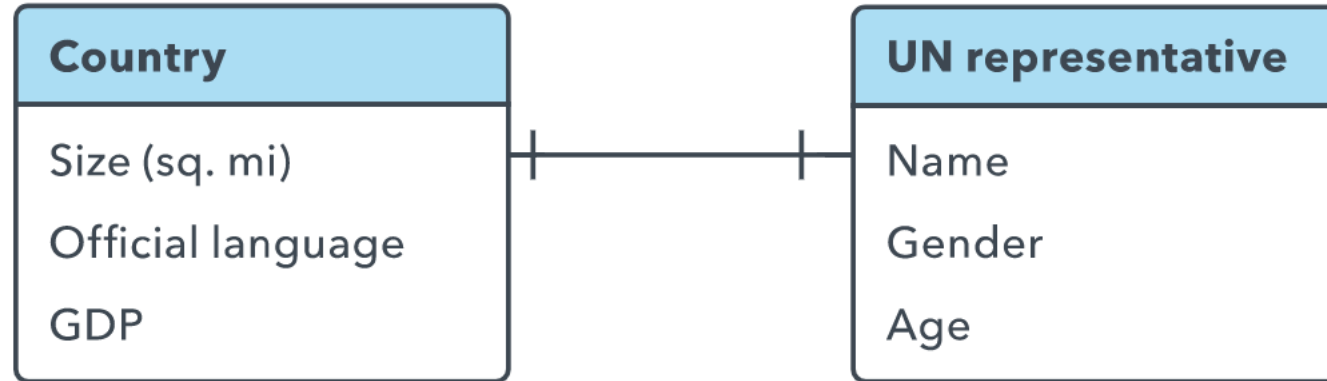
Price

Total

Cardinality refers to the quantity of elements that interact between two related tables. Identifying the cardinality helps make sure you've divided the data into tables most efficiently.

Creating Relationships between Tables/ Entities

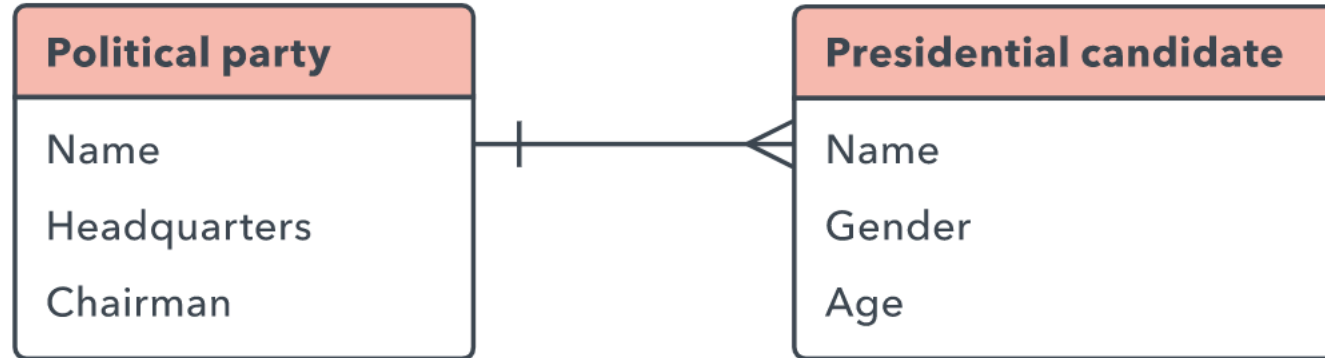
One-to-one Relationships



- When there's only one instance of Entity A for every instance of Entity B, they are said to have a one-to-one relationship (often written 1:1).
- A 1:1 relationship usually indicates that you'd be better off combining the two tables' data into a single table.
- To guarantee that the data matches up correctly, you'd then have to include at least one identical column in each table, most likely the primary key.

Creating Relationships between Tables/ Entities

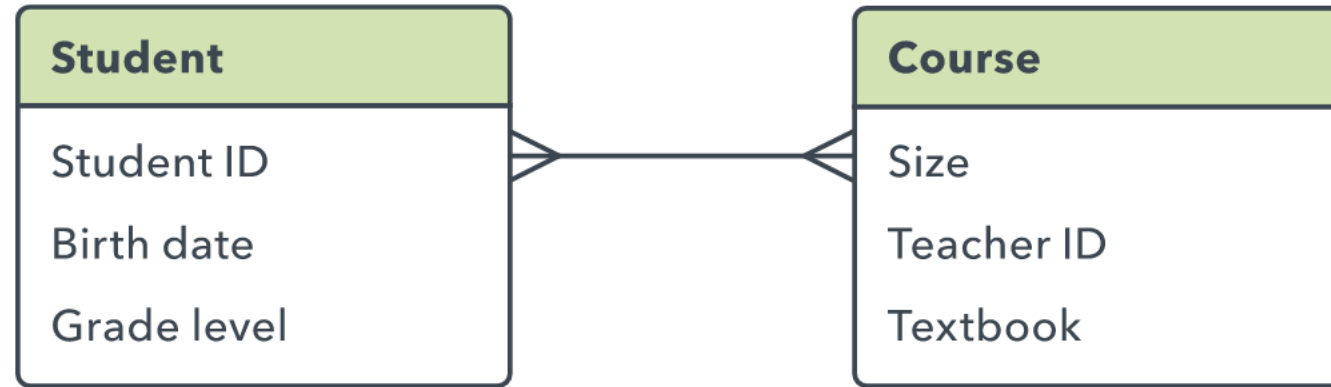
One-to-many Relationships



- To implement a 1:M relationship as you set up a database, simply add the primary key from the “one” side of the relationship as an attribute in the other table.
- When a primary key is listed in another table in this manner, it’s called a foreign key.
- The table on the “1” side of the relationship is a considered a parent table to the child table on the other side.

Creating Relationships between Tables/ Entities

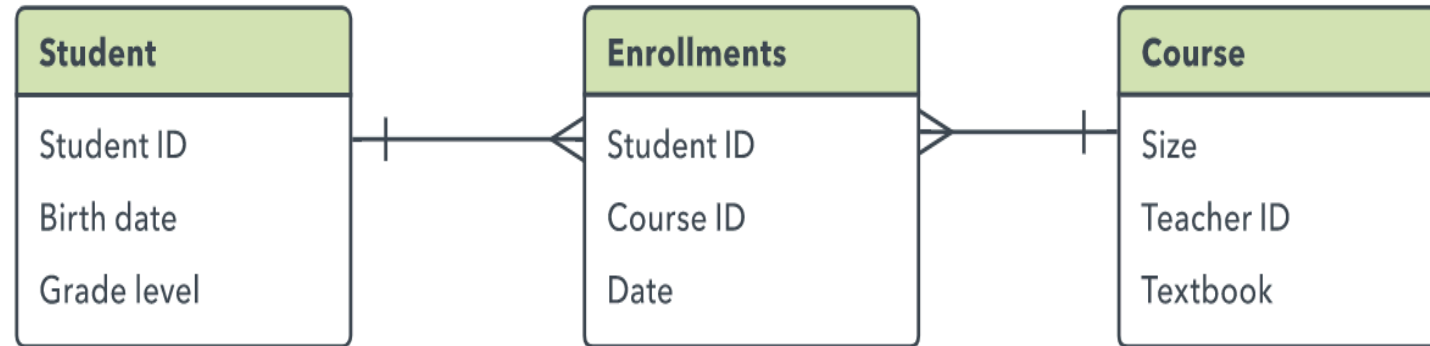
Many-to-many Relationships



- It's not directly possible to implement this kind of relationship in a database. Instead, you have to break it up into two one-to-many relationships.

Creating Relationships between Tables/ Entities

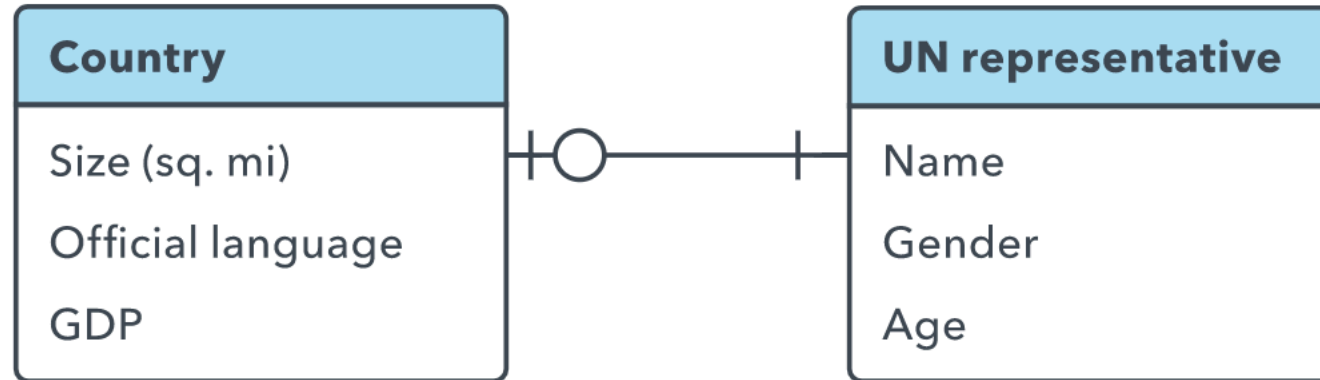
Many-to-many Relationships



- Each record in the link table would match together two of the entities in the neighbouring tables (it may include supplemental information as well).
- For instance, a link table between students and classes might look like the above.

Creating Relationships between Tables/ Entities

Mandatory or Not?



- Another way to analyse relationships is to consider which side of the relationship has to exist for the other to exist. The mandatory side can be marked with a circle on the line where a dash would be.
- For instance, a country has to exist for it to have a representative in the United Nations, but the opposite is not true.
- Two entities can be mutually dependent (one could not exist without the other).



**NANYANG
TECHNOLOGICAL
UNIVERSITY**
SINGAPORE

Normalisation of Data

BS3033 Data Science for Biologists

Dr Wilson Goh

School of Biological Sciences

Normalisation

Once you have a preliminary design for your database, you can apply normalisation rules to make sure the tables are structured correctly.

- Think of these rules as industry standards
- Normalisation in tiers (levels 1, 2, 3)
- Inheritance: each form, or level of normalisation, includes the rules associated with the lower forms

Database Normalisation

First Normal Form (1NF)

Rule: Each cell in the table can have only one value, never a list of values.

Product ID	Colour	Price
1	Brown, Yellow	\$15
2	Red, Green	\$13
3	Blue, Orange	\$11

X

Database Normalisation

First Normal Form (1NF)

Product ID	Colour	Price
1	Brown, Yellow	\$15
2	Red, Green	\$13
3	Blue, Orange	\$11

Try splitting this up?



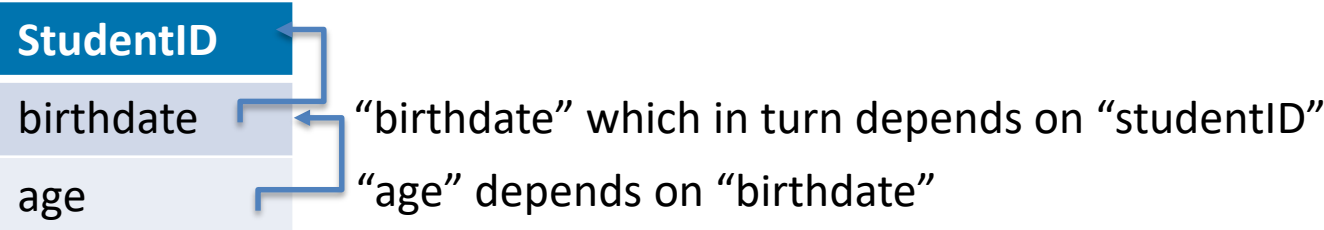
Products
Color1
Color2
Color3
Price

A table with groups of repeated or closely related attributes does not meet the first normal form.

Database Normalisation

Second Normal Form (2NF)

Rule: Each attribute should be fully dependent on the primary key.



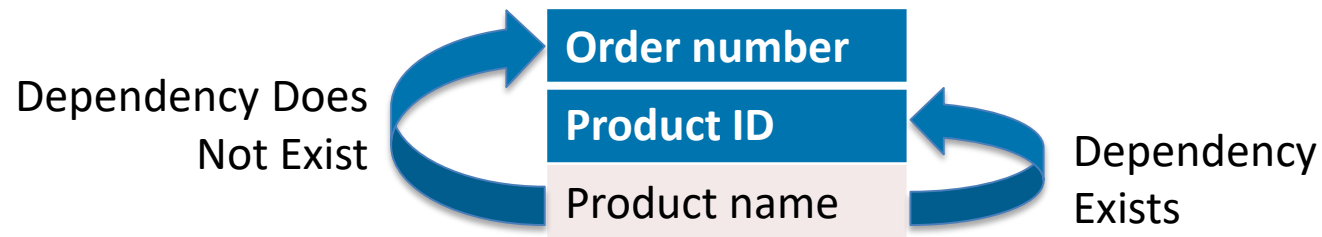
X 2NF

Age does not depend directly on studentID.

Database Normalisation

Second Normal Form (2NF)

Rule: A table with a primary key made up of multiple fields violates the second normal form if one or more of the other fields do not depend on every part of the key. So be careful of using multi-attribute keys.



X 2NF

Database Normalisation

Third Normal Form (3NF)

Rule: Every non-key column be independent of every other column. This keeps you from storing any derived data in the table.



Order	Price	Tax
14325	\$40.99	\$2.05
14326	\$13.73	\$0.69
14327	\$24.15	\$1.21

The “tax” column directly depends on the total price of the order.

Database Normalisation

While these forms explain the best practices to follow generally, the degree of normalisation depends on the context of the database.

Online Transaction Processing (OLTP)

- Users are concerned with creating, reading, updating, and deleting records, should be normalised.
- Ease of updating and changing.

Online Analytical Processing (OLAP)

- Favour analysis and reporting might fare better with a degree of denormalisation, since the emphasis is on speed of calculation. These include decision support applications in which data needs to be analysed quickly but not changed.
- Ease of analysis.

Data Integrity Rules

Entity Integrity Rule: Primary key must be unique.



The primary key can never be NULL. If the key is made up of multiple columns, none of them can be NULL. Otherwise, it could fail to uniquely identify the record.

Referential Integrity Rule: Each foreign key must be matched with 1 primary key.



If the primary key changes or is deleted, those changes will need to be implemented wherever that key is referenced throughout the database.

Database Management Systems

Many of the design choices you will make also depend on which database management system you use. Some of the most common systems include:



Oracle DB

My SQL

Microsoft SQL Server

PostgreSQL

IBM DB2



**NANYANG
TECHNOLOGICAL
UNIVERSITY**
SINGAPORE

Database Models

BS3033 Data Science for Biologists

Dr Wilson Goh

School of Biological Sciences

What is a Database Model?

A database model shows the logical structure of a database, including the relationships and constraints that determine how data can be stored and accessed.

Types of Database Models

You may choose to describe a database with any one of these depending on several factors. The biggest factor is whether the database management system you are using supports a particular model.



Hierarchical Database Model

Relational Model

Network Model

Object-oriented Database Model

Entity-relationship Model

Document Model

Entity-attribute-value Model

Relational Model

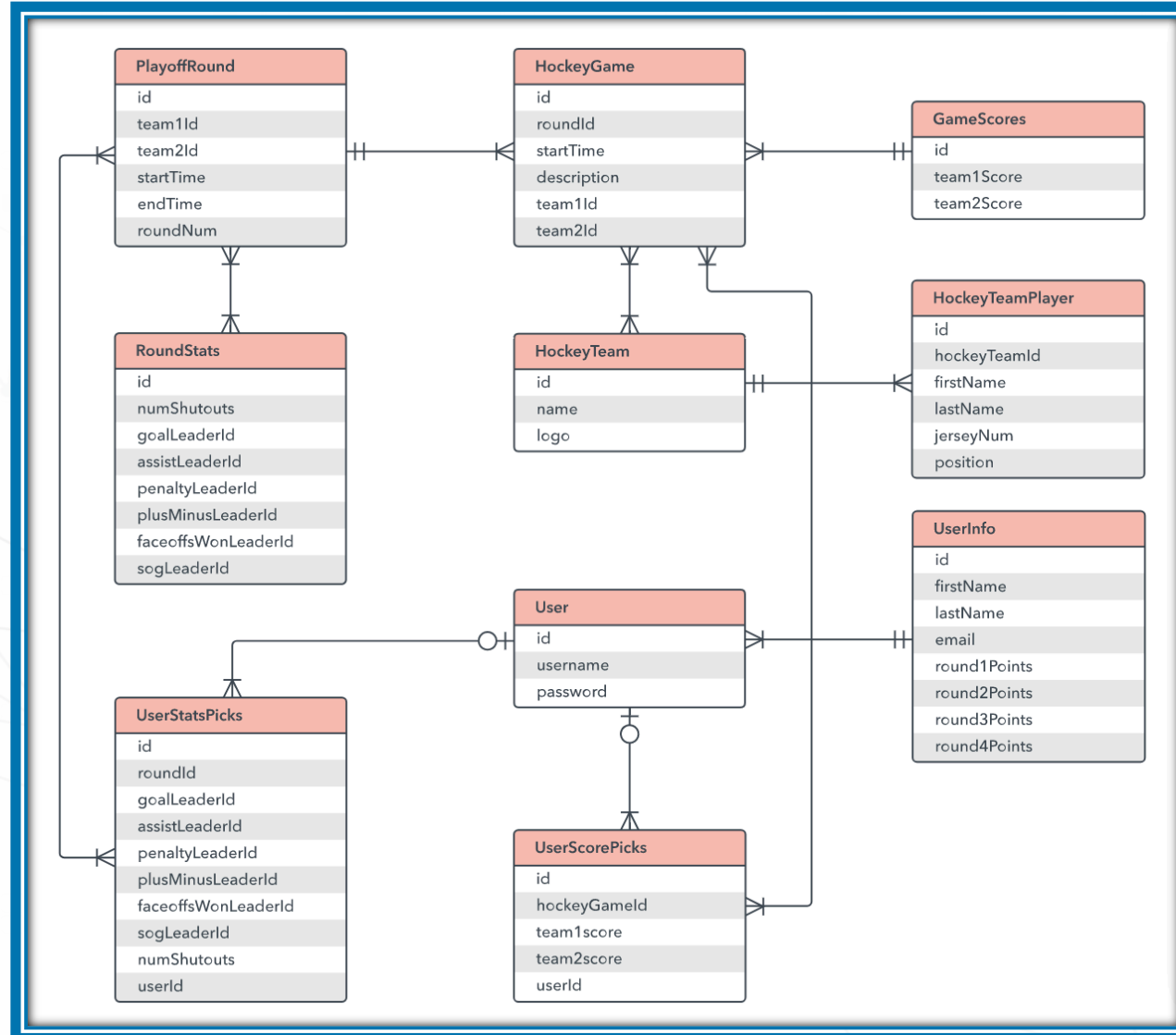
Student ID	First name	Last name
52-743965	Charles	Peters
48-209689	Anthony	Sondrup
14-204968	Rebecca	Phillips

ProviderID	Provider name
156-983	UnitedHealth
146-823	Blue Shield
447-784	Carefirst Inc.

Student ID	ProviderID	Type of plan	Start date
52-743965	156-983	HSA	04/01/2016
48-209689	146-823	HMO	12/01/2015
14-204968	447-784	HSA	03/14/2016

- Data is organised into tables.
- Each row is called a tuple.
- Types of relationships between tables.
- Within the database, tables can be normalised.
- Relational databases are typically written in Structured Query Language (SQL).

Entity Relational Models



Other Database Model

Flat Model

```
LOCUS       eIF4E             2881 bp    DNA             INV             31-AUG-1999
DEFINITION   Drosophila melanogaster eukaryotic initiation factor 4E (eIF4E)
             gene, alternative splice products, complete cds.
ACCESSION   eIF4E
VERSION
KEYWORDS
SOURCE      .
ORGANISM    Drosophila melanogaster
            Eukaryota; Metazoa; Arthropoda; Tracheata; Hexapoda; Insecta;
            Pterygota; Diptera; Brachycera; Muscomorpha; Ephydroidea;
            Drosophilidae; Drosophila.
REFERENCE   1 (bases 1 to 2881)
AUTHORS     Lavoie,C.A., Lachance,P.E.D., Sonenberg,N. and Lasko,P.
TITLE       Alternatively spliced transcripts from the Drosophila eIF4E gene
            produce two different Cap-binding proteins
JOURNAL     J. Biol. Chem. 271 (27), 16393-16398 (1996)
MEDLINE    96279193
REFERENCE   2 (bases 1 to 2881)
AUTHORS     Darwin,C.R.
TITLE       Direct Submission
JOURNAL     Submitted (31-AUG-1999) Evolutionary Biology Department, Oxbridge
            University, 1859 Tennis Court Lane, Camford OX1 2BH, England
FEATURES    Location/Qualifiers
     source          1..2881
                   /organism="Drosophila melanogaster"
                   /strain="Oregon R"
                   /chromosome="3"
                   /map="67A8-B2"
     gene            80..2881
                   /gene="eIF4E"
     CDS             join(201..224,1550..1920,1986..2085,2317..2404,2466..2629)
                   /gene="eIF4E"
                   /codon_start=1
                   /product="eukaryotic initiation factor 4E-II"
                   /translation="MVLLETEKTSAPSTEQGAPPEPPTSAAAPAEAKDKVPKEDPQETG
EPAGNTATTTAPAGDDAVRTEHLYKHPLMNVVTLWVLENDRAKSWEDMQNEI TSFDTV
EDFVSLVNH I KPPSE I KLGSDYSLFKKN I RPMWEDARKKGGRAWV I TLNKSSKTDLDN
LWLDVLLCL I GEAFDHSQ I CGAV I N I RGKSNK I S I WTADGNNEEARLE I GHKLRDAL
RLGRNNSLQYQLHKD TMVKQGSNVKSI YTL"
```

← Header

← Feature

Flat model is the earliest, simplest data model. It lists all the data in a single table. In order to access or manipulate the data, the entire flat file must be read into memory (inefficient). **GenBank**, which stores info on biological sequences, is based on flat model.

← Sequence

Other Database Model

XML

- XML stands for eXtensible Markup Language
- XML is a markup language (like HTML)
- Use for storing and transport data
- Self-descriptive; flexible and expandable

```
<note>           Opening
  <date>2015-09-01</date>
  <hour>08:30</hour>
  <to>Tove</to>
  <from>Jani</from>
  <body>Don't forget me this weekend!</body>
</note>         Closing with /
```

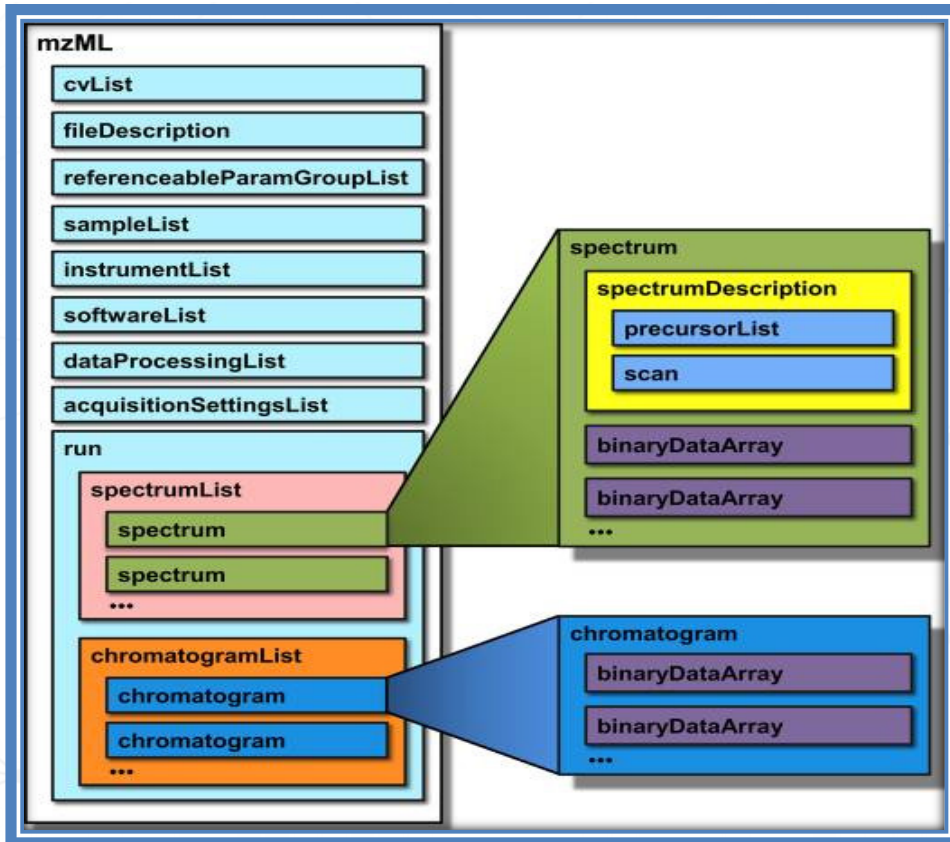
Nesting of Different Tags

Data is Stored within Tag Elements

Other Database Model

Proteomics uses a lot of XML-based databases

Schematic representation key elements of the MZMLformat.



Example of actual MZML file

```
Ecoli_MS2_small.mzML
1 <?xml version="1.0" encoding="ISO-8859-1"?>
2 <mzML xmlns="http://psi.hupo.org/ms/mzml" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://
psi.hupo.org/ms/mzml http://psidev.info/files/ms/mzML/xsd/mzML1.1.0.xsd" accession="" version="1.1.0">
3   <cvList count="2">
4     <cv id="MS" fullName="Proteomics Standards Initiative Mass Spectrometry Ontology" URI="http://
psidev.cvs.sourceforge.net/*checkout*/psidev/psi-ms/mzML/controlled/vocabulary/psi-ms_obo"/>
5     <cv id="UO" fullName="Unit Ontology" URI="http://obo.cvs.sourceforge.net/obo/obo/ontology/phenotype/unit.obo"/>
6   </cvList>
7   <fileDescription>
8     <fileContent>
9       <cvParam cvRef="MS" accession="MS:1000580" name="MSn spectrum" />
10    </fileContent>
11  </fileDescription>
12  <sampleList count="1">
13    <sample id="sa_0" name="">
14      <cvParam cvRef="MS" accession="MS:1000004" name="sample mass" value="0" unitAccession="UO:0000021" unitName="gram"
unitCvRef="UO" />
15      <cvParam cvRef="MS" accession="MS:1000005" name="sample volume" value="0" unitAccession="UO:0000098" unitName="
milliliter" unitCvRef="UO" />
16      <cvParam cvRef="MS" accession="MS:1000006" name="sample concentration" value="0" unitAccession="UO:0000175" unitName="
gram per liter" unitCvRef="UO" />
17    </sample>
18  </sampleList>
19  <softwareList count="6">
20    <software id="so_in_0" version="2.4 SP1" >
21      <cvParam cvRef="MS" accession="MS:1000532" name="Xcalibur" />
22    </software>
23    <software id="so_default" version="" >
24      <cvParam cvRef="MS" accession="MS:1000799" name="custom unreleased software tool" value="" />
25    </software>
26    <software id="so_dp_sp_0_pm_0" version="1.6.0" >
27      <cvParam cvRef="MS" accession="MS:1000615" name="ProteWizard" />
28    </software>
29    <software id="so_dp_sp_0_pm_1" version="1.7.0" >
30      <cvParam cvRef="MS" accession="MS:1000757" name="FileFilter" />
31    </software>
32    <software id="so_dp_sp_0_pm_2" version="1.9.0" >
33      <cvParam cvRef="MS" accession="MS:1000757" name="FileFilter" />
34    </software>
35    <software id="so_dp_sp_0_pm_3" version="1.9.0" >
36      <cvParam cvRef="MS" accession="MS:1000757" name="FileFilter" />
37    </software>
38  </softwareList>
39  <instrumentConfigurationList count="1">
40    <instrumentConfiguration id="ic_0">
41      <cvParam cvRef="MS" accession="MS:1000556" name="LTQ Orbitrap XL" />
42      <userParam name="instrument serial number" type="xsd:string" value="015798"/>
43      <componentList count="3">
44        <source order="1">
45          <cvParam cvRef="MS" accession="MS:1000485" name="nanospray inlet" />
```

Source: Martens et al. mzML—a Community Standard for Mass Spectrometry Data. MCP 2011



**NANYANG
TECHNOLOGICAL
UNIVERSITY**
SINGAPORE

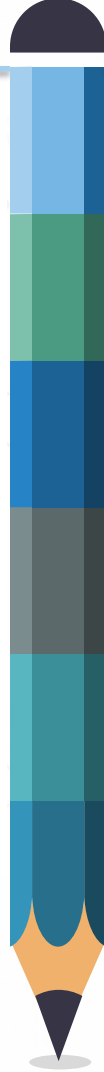
Summary

BS3033 Data Science for Biologists

Dr Wilson Goh

School of Biological Sciences

Key Takeaways from this Topic

- 
1. A database is an organised collection of data, stored and accessed electronically. Database designers typically organise the data to model aspects of reality in a way that supports practical usage.
 2. Database normalisation is the process of restructuring a relational database in accordance with a series of so-called normal forms in order to reduce data redundancy and improve data integrity.
 3. The Relational Model (RM) for database management is an approach to managing data using a tabular structure, with entries recorded as rows, and defined uniquely by a primary key.
 4. A flat file database is a database that stores data in a plain text file. Each line of the text file holds one field.
 5. An XML database stores and represents data as a series of nested tags with appropriate opening and closing statements.