Imperial College
Business School

Imperial means
Intelligent Business

Lecture 3
Normalisation: keeping our data tidy

Dr Fintan Nagle
f.nagle@imperial.ac.uk

Reading

Video lectures:

4.5 - Normalisation.mp4
4.6 - Second normal form.mp4
4.7 - Third normal form.mp4
4.8 - Datatypes and CREATE TABLE.mp4

Constructing a new database

There are three stages:

- Decide what information you want to store in the DB.
- Decide how this information will be laid out in tables
  *This is **normalisation** – finding a suitable way to fit our data into the relational paradigm.*
- Creating the database (**CREATE DATABASE**) and the tables (**CREATE TABLE**).

Creating tables (and data types)

Buiding your own databases

CREATE DATABASE movies;

Creating a new table

```
CREATE TABLE table_name(
        column1 datatype,
        column2 datatype,
        column3 datatype,
        .....
        columnN datatype
);
```

Buiding your own databases

```
CREATE TABLE students(first_name text,
last_name text,
gender text,
age integer,
home_country text,
home_continent text,
favourite_animal text)
```

Buiding your own databases

INSERT INTO students VALUES

('joe', 'bloggs', 'm', 22, 'Britain', 'Europe', 'dog')

Updating

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

Deleting

```
DELETE
FROM students
WHERE last_name = 'bloggs'
```

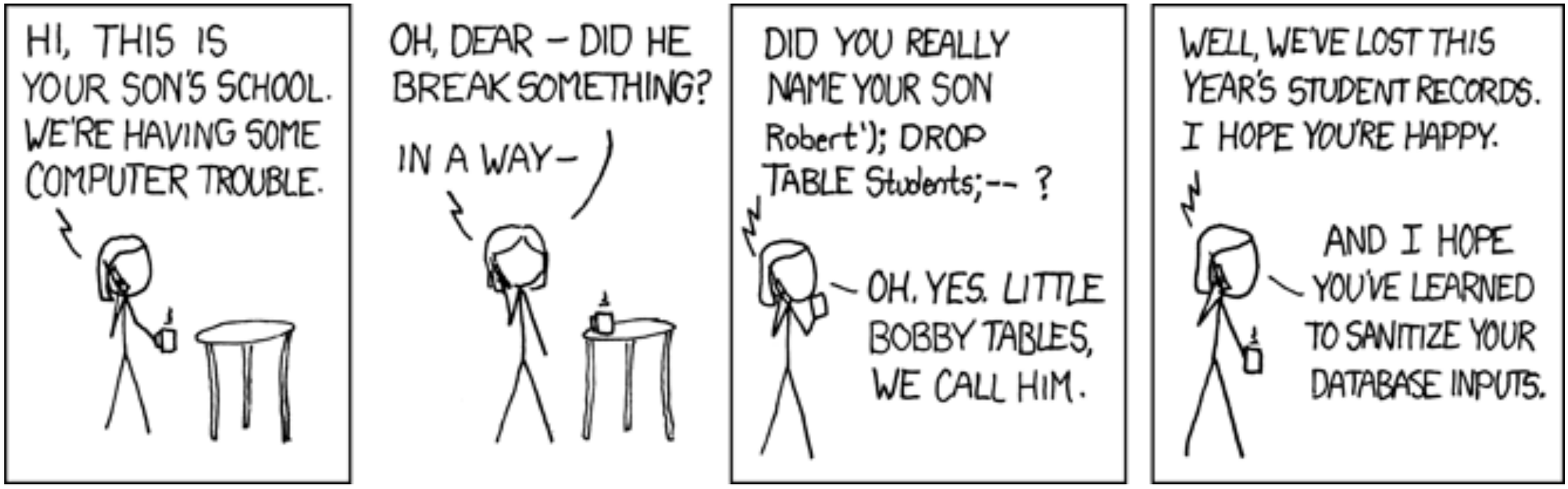
It's a good idea to run a SELECT query first, then change the SELECT * to DELETE once you have verified the right rows are being affected.

Deleting a table or DB

DROP TABLE table_name;

DROP DATABASE database_name;

Deleting a table



Inserting a query result (table) into another table

```
INSERT INTO items_ver (item_id, name, item_group)
SELECT item_id, name, item_group
FROM items
WHERE item_id=2;
```

Normalisation: intro

"A row should describe a fact about
the key,
the whole key,
and nothing but the key."

"A row should describe a fact about
the key,
the whole key,
and nothing but the key,
so help me Codd."

"A row should describe a fact about
the key, **1NF**
the whole key, **2NF**
and nothing but the key, **3NF**
so help me Codd."

The Movies database

Everything is in one table!

What are the advantages and disadvantages of this approach?

The Movies database

Everything is in one table!

What are the advantages and disadvantages of this approach?

Advantages:

- You don’t need to remember the names of the other tables

Disadvantages:

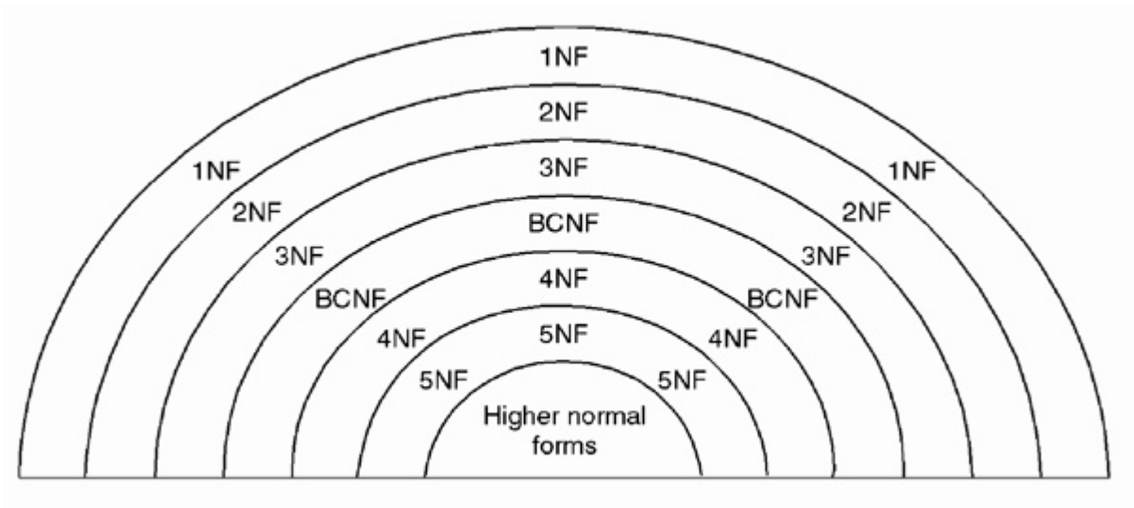
- There are too many columns
- For certain rows, columns may be blank
- If we delete a film, we delete other information too
- Information is unnecessarily copied

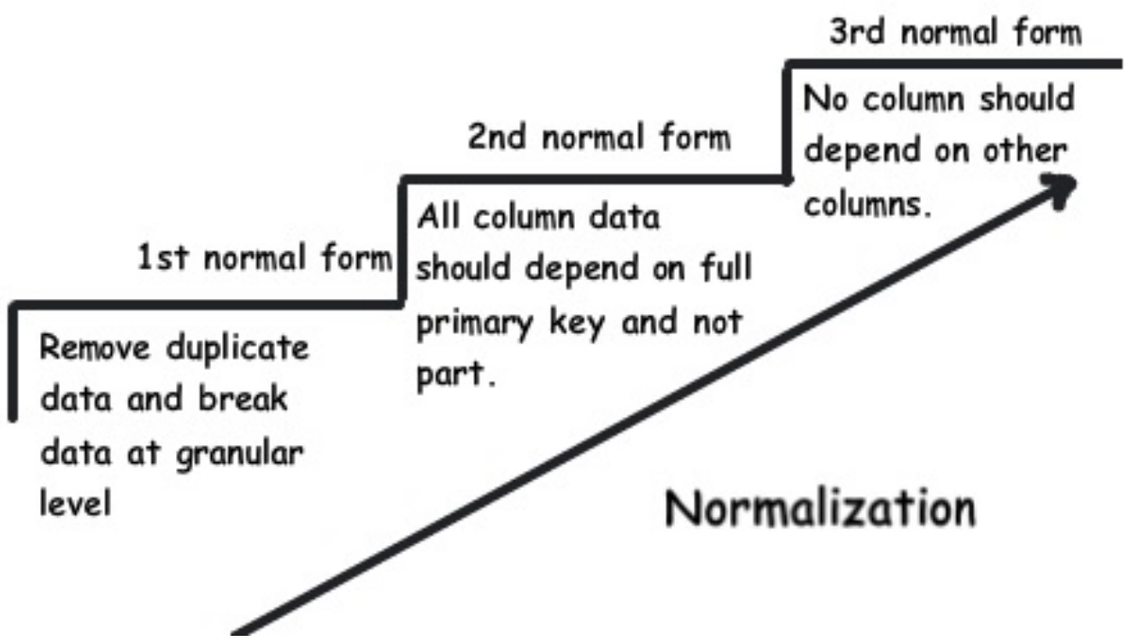
Problems we are trying to prevent

- Wasted space
- Deletion anomalies: deleting information we don't want to e.g. deleting actor information when we drop a film
- Update anomalies: information is stored in multiple places and so must be updated at the same time (**atomic update** – nothing else can happen between updates)

How do we split up our tables?
Normalisation

How do we split up our tables?
Normalisation





Form name	Abbreviation	Rules
First Normal Form	1NF	• Each field should contain the smallest meaningful value (atomic form). • No repeated groups of fields. • Each record is identified with a primary key.
Second Normal Form	2NF	• Must meet 1NF requirements. • Remove fields that aren't related to the primary key.
Third Normal Form	3NF	• Must meet 1NF and 2NF requirements. • Any fields that aren't directly dependent on the primary key are eliminated.

How do we split up our tables?
Normalisation

Normal forms are complex – but there are some basic rules:

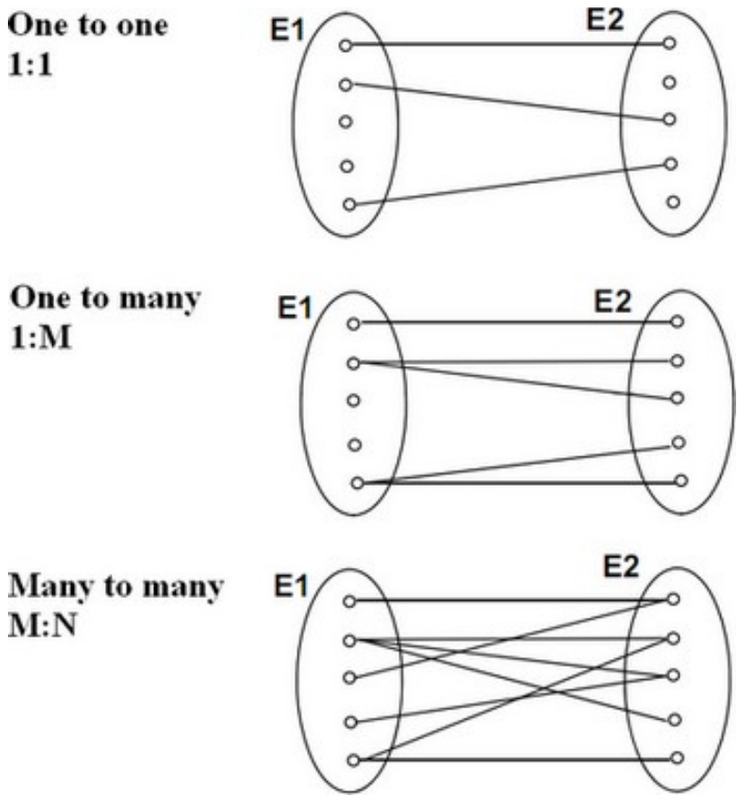
- **Don’t copy data**
  Otherwise, with fast reads or if something goes wrong, we may read the wrong thing

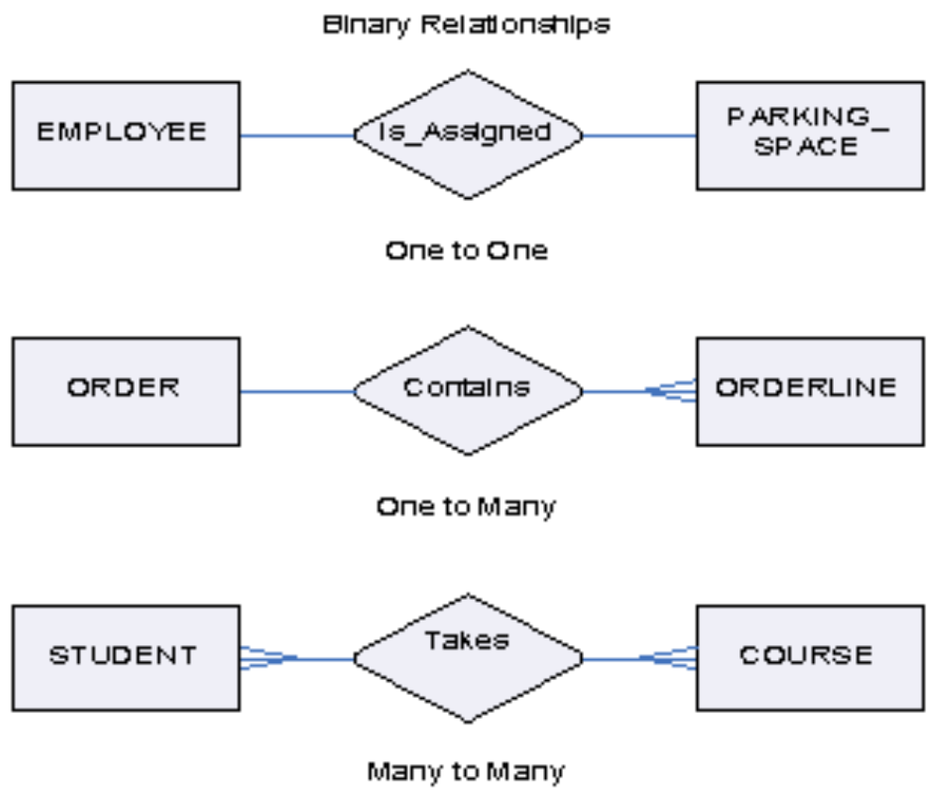
- **Prevent corruptions on delete**
  If everything is in one table and we delete the last film with Johnny Depp acting in it, Johnny will disappear from the database

- **Prevent corruptions on multiple inserts/updates**
  If Johnny Depp’s marriage status updates, we don’t want to read the wrong status if we do a read when only some of the relevant rows have been updated

First normal form

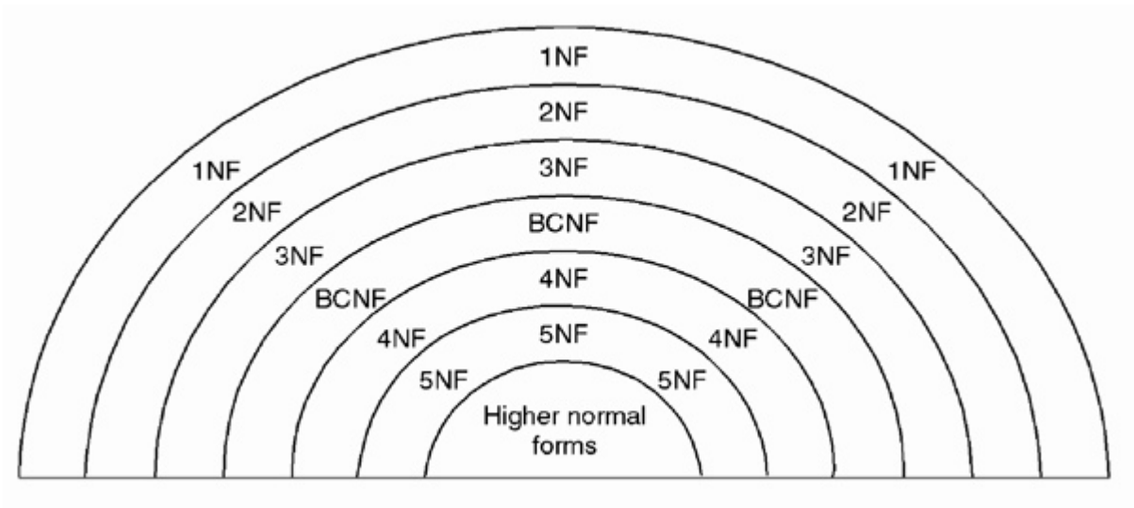
Types of relationship between entities (between tables)

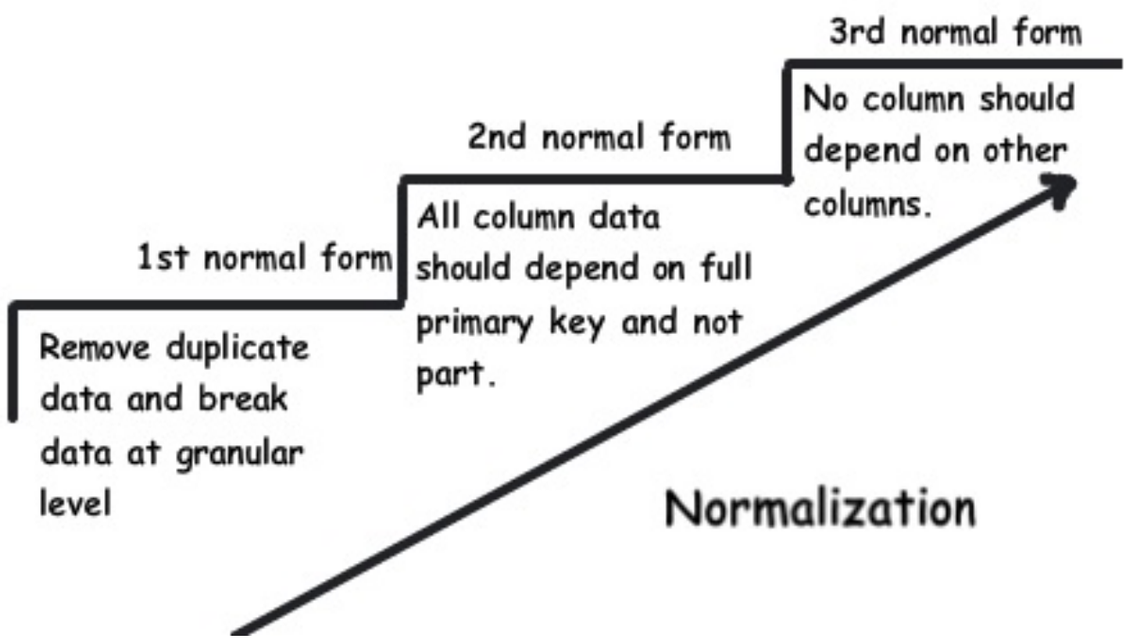




- **One to one:** each entity can only connect to one other entity.
  (Symmetric)
- **One to many:** each source entity can connect to many target entities (but each target can only connect to one source).
  (Asymmetric)
- **Many to many:** no restrictions (one source can have many targets; one target can have many sources)
  (Symmetric)

How do we split up our tables?
Normalisation





Form name	Abbreviation	Rules
First Normal Form	1NF	• Each field should contain the smallest meaningful value (atomic form). • No repeated groups of fields. • Each record is identified with a primary key.
Second Normal Form	2NF	• Must meet 1NF requirements. • Remove fields that aren't related to the primary key.
Third Normal Form	3NF	• Must meet 1NF and 2NF requirements. • Any fields that aren't directly dependent on the primary key are eliminated.

How do we split up our tables?
Normalisation

First normal form

A table is in first normal form if there is **not more than one entity of the same kind** in every slot (a slot is identified by row and column) in the table.

Enforcing 1NF involves generating extra rows (not new tables). Each row with more than one entity type is inflated/copied into more rows.

Other sources may stipulate additional restrictions for 1NF – keys present, etc. **On this course we just look at the one-entity-per-slot limit.**

How do we split up our tables?
Normalisation

STUD_NO	STUD_NAME	STUD_PHONE	STUD_STATE	STUD_COUNTRY
1	RAM	9716271721, 9871717178	HARYANA	INDIA
2	RAM	9898297281	PUNJAB	INDIA
3	SURESH		PUNJAB	INDIA

Table 1

Conversion to first normal form

STUD_NO	STUD_NAME	STUD_PHONE	STUD_STATE	STUD_COUNTRY
1	RAM	9716271721	HARYANA	
1	RAM	9871717178	HARYANA	INDIA
2	RAM	9898297281	PUNJAB	INDIA
3	SURESH		PUNJAB	INDIA

Table 2

1NF does not prevent us storing an address in a cell (even though it can be broken up into smaller parts).

1NF does prevent us storing *two* addresses in a table cell.

Fixing 1NF violations

To enforce 1NF, copy "overfilled" cells, making more rows and copying in the other cells.

Second normal form

Remember that

- In database theory, a table is called a relation, a row is called a record and a column is called an attribute.

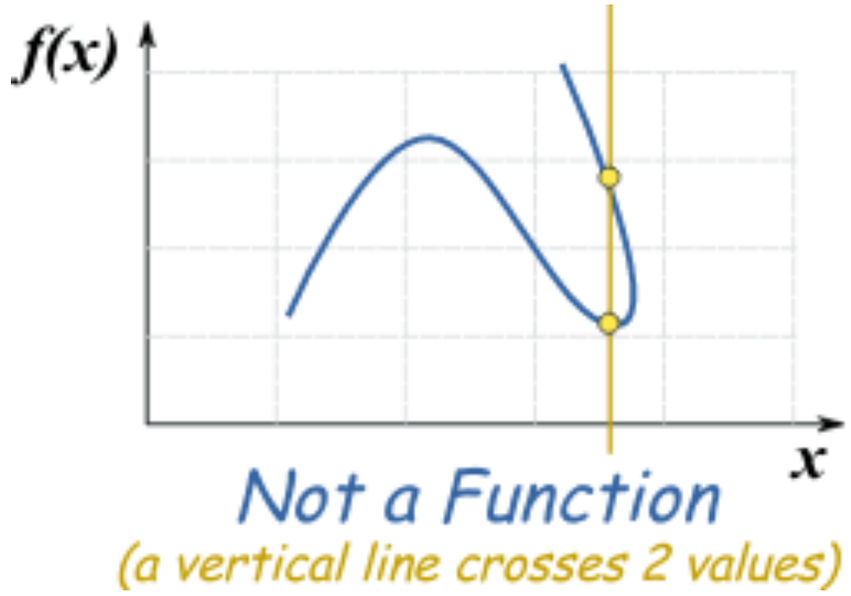
- A key is an attribute, or set of attributes, which uniquely identifies a row.

- Primary keys can be composite keys (keys consisting of more than one column).

- A candidate key is a **minimal** set of attributes which uniquely identifies rows. There can be many candidate keys, and one of them is selected as the primary key.

- (Minimal means we can't remove any of the attributes, and still identify the row).

Functional dependencies



Is a relationship a functional dependency or not?

A dependency from X to Y means that Y is *determined by* X. So Y must be constant as well as unique.

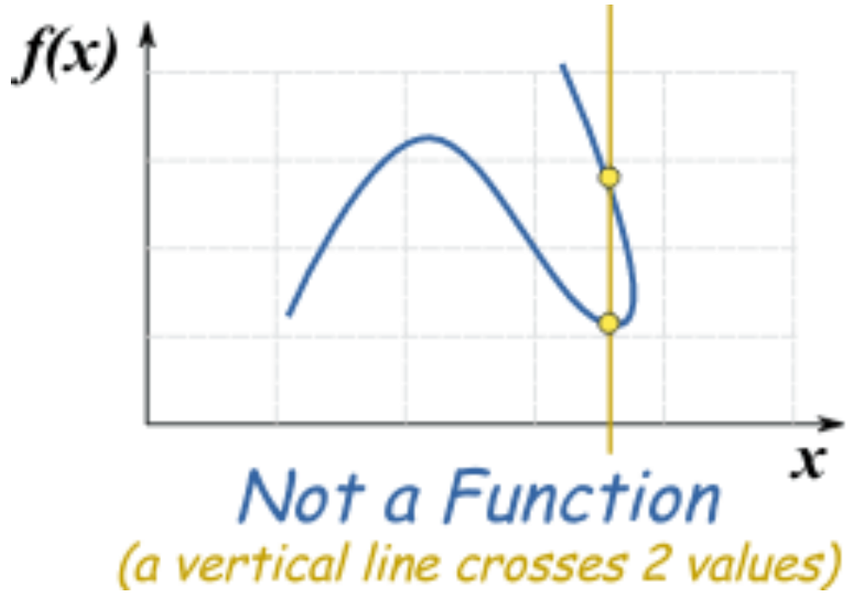
Sometimes this comes from the data, and sometimes from the business domain.

employee_id	name
1	John C
2	Anna E

age	name
25	John C
32	Anna E

age	name
25	John C
32	Anna E
25	Eric R

Functional dependencies



Is a relationship a functional dependency or not?

Sometimes this comes from the data, and sometimes from the business domain.

Probably dependent (unique ID)

employee_id	name
1	John C
2	Anna E

age	name
25	John C
32	Anna E

Could be dependent, but probably isn't

age	name
25	John C
32	Anna E
25	Eric R

Definitely not dependent

Terms

None of these are database-specific terms

Functional dependency

A dependency between any entities (not just in databases).

Proper subsets

A proper subset Y of a set X is one which is *strictly smaller* than X.
items in Y < items in X (proper subset)
rather than
items in Y <= items in X (subset)

Transitivity

The property that if A is related to B and B is related to C, then A is related to C and so on.

How do we split up our tables?
Normalisation

Second normal form

A table in second normal form must first be in 1st normal form; also, it has no partial dependencies (i.e. dependencies from any proper subsets of any of the candidate keys).

Essentially: no redundancy.

Candidate key: set of columns which can uniquely identify a row

Non-prime attribute: attribute which is not part of any candidate key

A **partial dependency** is a dependency from a proper subset of any candidate key of the table, to any non-prime attribute.

Example: second normal form?

STUD_NO	COURSE_NO	COURSE_NAME
1	C1	DBMS
2	C2	Computers Network
1	C2	Computers Network

Example: second normal form?

STUD_NO	COURSE_NO	COURSE_NAME
1	C1	DBMS
2	C2	Computers Network
1	C2	Computers Network

```
FD set: {COURSE_NO->COURSE_NAME}
Candidate Key: {STUD_NO, COURSE_NO}
```

Example: second normal form?

STUD_NO	COURSE_NO	COURSE_NAME
1	C1	DBMS
2	C2	Computers Network
1	C2	Computers Network

```
FD set: {COURSE_NO->COURSE_NAME}
Candidate Key: {STUD_NO, COURSE_NO}
```

In FD COURSE_NO->COURSE_NAME, COURSE_NO (proper subset of candidate key) is determining COURSE_NAME (non-prime attribute). Hence, it is partial dependency and relation is not in second normal form.

Fixing 2NF violations

Make a new table.
Copy entity data to this table;
keep only keys in the previous tables.

Students table

STUD_NO	COURSE_NO
1	C1
2	C2
1	C2

Courses table

COURSE_NO	COURSE_NAME
C1	DBMS
C2	Computers Network

Third normal form

How do we split up our tables?
Normalisation

Third normal form

A table is in third normal form if it is in second normal form and

there are no transitive dependencies from candidate keys

(i.e. every non-prime attribute of the table is non-transitively (directly) dependent on every key of the table.

Transitive dependency:

if A > B and B > C are dependencies, then A > C is also a (transitive) dependency.

Non-prime attribute: one that does is not part of a primary key.

How do we split up our tables?

Normalisation

Third normal form

Third normal form

“[Every] non-key [attribute] must provide a fact about the key [1NF], the whole key [2NF], and nothing but the key [3NF].”

-Bill Kent

Transitive dependency:

if A > B and B > C are dependencies, then A > C is also a (transitive) dependency.

Non-prime attribute: one that does is not part of a primary key.

How do we split up our tables?
Example: 3NF?

Tournament Winners

Tournament	Year	Winner	Winner Date of Birth
Indiana Invitational	1998	Al Fredrickson	21 July 1975
Cleveland Open	1999	Bob Albertson	28 September 1968
Des Moines Masters	1999	Al Fredrickson	21 July 1975
Indiana Invitational	1999	Chip Masterson	14 March 1977

How do we split up our tables?
Example: 3NF?

Not in 3NF because the non-prime attribute Winner Date of Birth is transitively dependent on the candidate key {Tournament, Year} via the non-prime attribute Winner.

Tournament Winners

Tournament	Year	Winner	Winner Date of Birth
Indiana Invitational	1998	Al Fredrickson	21 July 1975
Cleveland Open	1999	Bob Albertson	28 September 1968
Des Moines Masters	1999	Al Fredrickson	21 July 1975
Indiana Invitational	1999	Chip Masterson	14 March 1977

Fixing 3NF violations

Again, make a new table.
Copy entity data to this table;
keep only keys in the previous tables.

Tournaments table

Tournament	Year	Winner
Indiana Invitational	1998	Al Fredrickson
Cleveland Open	1999	Bob Albertson
Des Moines Masters	1999	Al Fredrickson
Indiana Invitational	1999	Chip Masterson

Winners table

Winner	Winner Date of Birth
Al Fredrickson	21 July 1975
Bob Albertson	28 September 1968
Chip Masterson	14 March 1977

More examples

There are many more examples of 1NF, 2NF and 3NF normalisation online.

To get used to the concepts, work through some examples; there are also detailed walkthroughs in the video lectures.

"A row should describe a fact about
the key, **1NF**
the whole key, **2NF**
and nothing but the key, **3NF**
so help me Codd."

Joins are the opposite of normalisation

We normalise data to make it efficient and safe to store. This mainly involves splitting up tables.

This makes data harder to view and analyse.

So we have to use the JOIN operation to bring data back together for analysis.