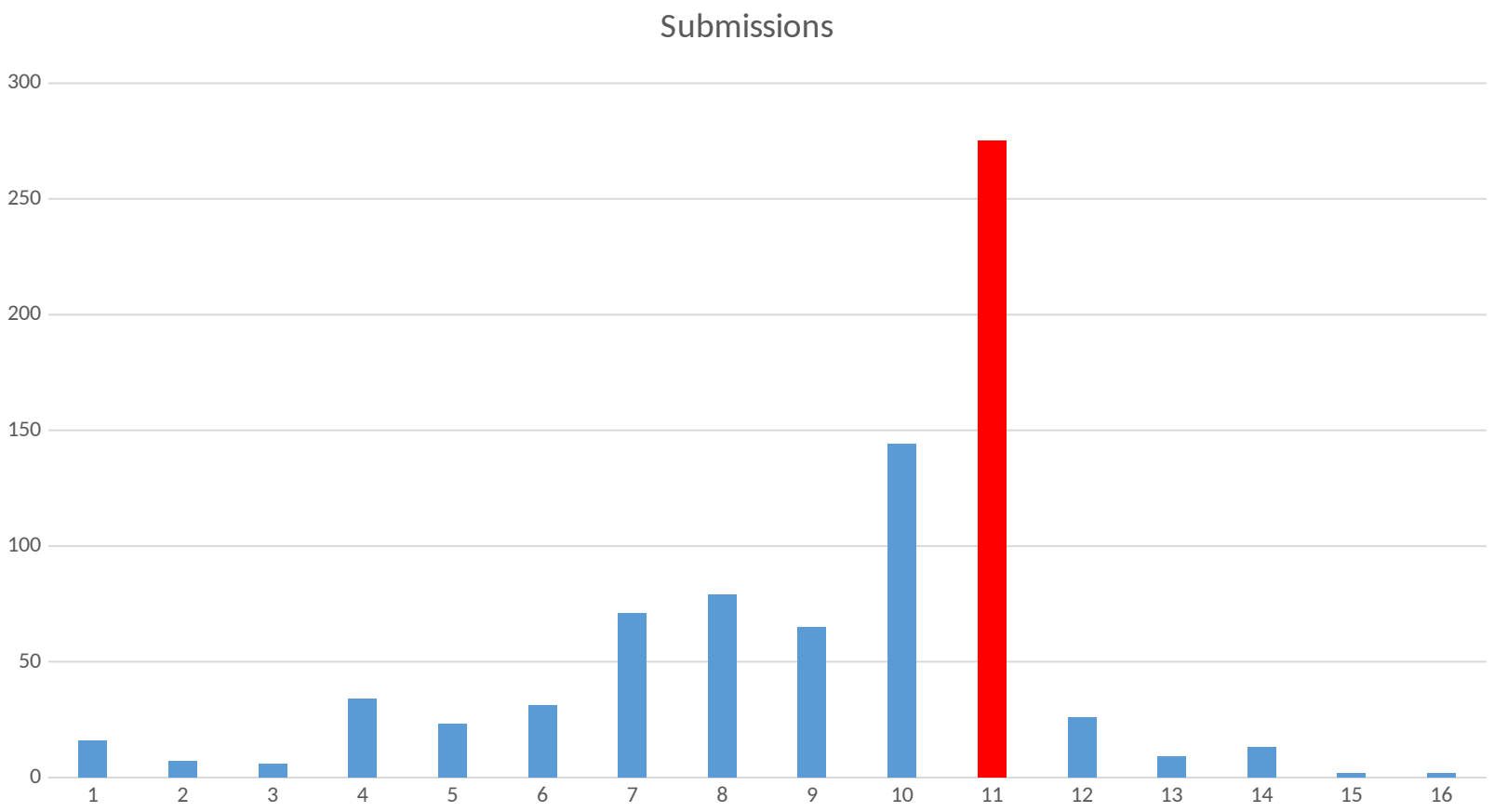
COMP1204: Normalisation

Oli Bills

ofb@ecs.soton.ac.uk

Some data...


Submissions
300
250
200
150
100
50
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16

Some cool stuff...

Some reminders...

- Relations
- Keys
- Functional Dependencies
- Anomalies
  - Redundancy
  - Update
  - Insert
  - Delete

Relations

Properties of Relations

- A relation $R(A_1,\ldots A_k)$ has the following properties
  1. Each row represents a k-tuple of R
  2. The ordering of rows is immaterial (**relations are just sets**)
  3. All rows are distinct (**relations are just sets**)
  4. The ordering of the attributes is **not significant**
  5. The significance of each column is conveyed by the name we give it

Keys

Keys

- A set of attributes forms a key for a relation if we do not allow two different tuples in a relation instance to have the same values in all the attributes of the key

- Courses(<u>CourseID</u>, Name, DeptID)
- Students(<u>Name,Surname</u>)
  - i.e., name AND surname

Functional Dependencies

Functional Dependencies

- Let a relation with schema $R(A_1,\ldots,A_n,B_1,B_2,\ldots,B_m)$ and let r an instance of R

- We say that r satisfies the functional dependency

determinant • $A_1,\ldots,A_{n'} \rightarrow B_1,B_2,\ldots,B_{m'}$ if **dependent**

  - whenever two tuples in r agree on the values of $A_1,\ldots,A_{n'}$, then they also agree on the values of $B_1,B_2,\ldots,B_{m'}$
  - in other words, there are no two tuples in r that have the same value on the attributes of $A_1,\ldots,A_{n'}$, but differ on the values of $B_1,B_2,\ldots,B_{m'}$

Anomalies

"Bad" Relations lead to Anomalies

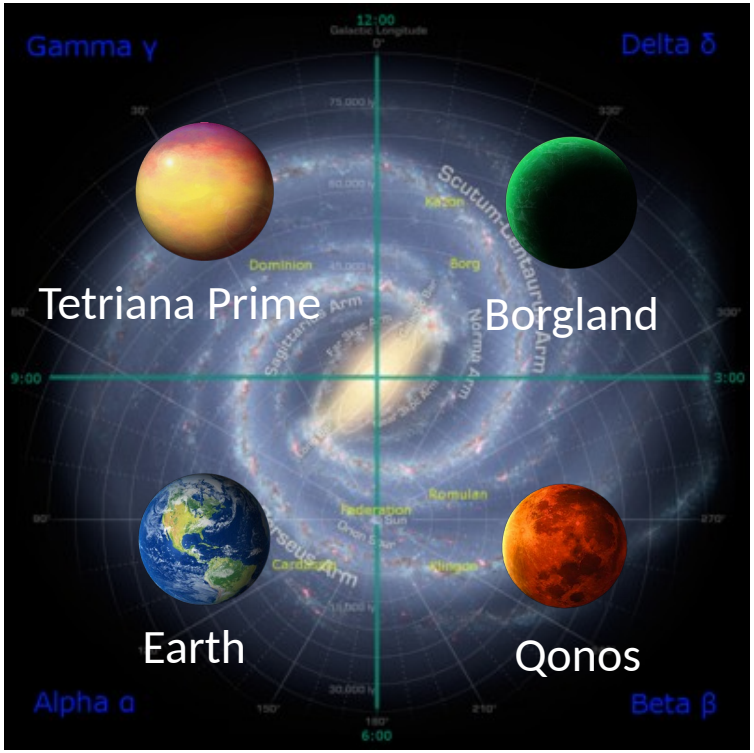
- **Redundancy**: information unnecessarily repeated (E.g., length, genre in many tuples)
- **Update Anomalies**: change information in one tuple and leave same info unchanged in another
- **Insert Anomalies**: we could insert data incorrectly
- **Deletion anomalies** (e.g., delete Vivien Leigh from Gone with the Wind)

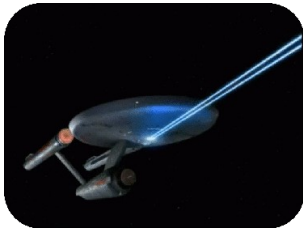
title	year	length	genre	studioName	starName
Star Wars	1977	124	SciFi	Fox	Carrie Fisher
Star Wars	1977	124	SciFi	Fox	Mark Hamill
Star Wars	1977	124	SciFi	Fox	Harrisson Ford
Gone with the Wind	1939	231	drama	MGM	Vivien Leigh

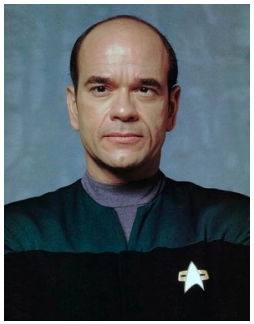
What is Normalisation?

- What we want to avoid
  - Redundant data
- Why do we want to avoid it
  - Efficiency
  - Maintenance
  - Size
- How do we do it?
  - Divide database into multiple tables
  - Define relationships between them

We’re going to make a game

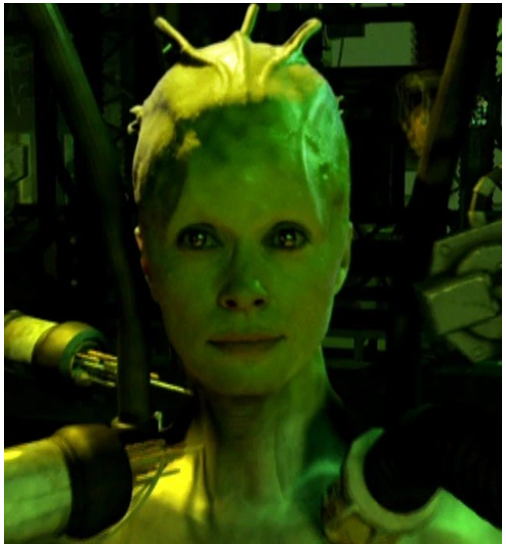






We’re going to make a game

- With a few random players
  - Kathryn
    - Human
  - Borg Queen
    - Borg
  - James
    - Human
  - More different James
    - Human



We start playing it on paper

It goes into a spreadsheet

Player	Name	Race	Ship Name	Ship Type	Ship Speed	Planet	Planet Colour	Location	Items
1	Kathryn	Human	Voyager	Intrepid	9	Earth	Blue	Alpha	Transporter,Phaser,Torpedo,Hologram
1	Kathryn	Human	Deadalus	Intrepid	8	Qonos	Red	Beta	Transporter,Phaser
2	Borg Queen	Borg	Borg Cube	Cube	10	Borgland	Green	Delta	Transporter,Phaser,Torpedo
3	James	Human	Enterprise	Enterprise	7	Earth	Blue	Alpha	Phaser
3	James	Human	Enterprise B	Enterprise	8	Tetriana Prime	Yellow	Gamma	Transporter,Phaser
4	James	Human	Gauntlet	Sovereign	5	Earth	Blue	Alpha	Transporter,Phaser,Torpedo

- Player details
  - Name
- Ships
  - Who has what ship
  - Name of ship
  - Ship specifications (e.g. speed)
- Planets
  - Quadrant they belong to
- Items
  - Who has what items

Now we want to make it into a computer game

- The playtesting was a success!
- How can we turn this into something useful?
- Maintaining the spreadsheet becomes very difficult…
- Trying to work with this for an online game would be even harder as it stands

Update Anomalies

- Redundancy
  - Same data held in multiple locations
- Update Anomalies
  - Data inconsistency
  - Partial updates
  - Having to update in multiple places
- Insert Anomalies
  - Attributes cannot be inserted without needing the presence of other attributes
- Deletion Anomalies
  - Attributes lost due to deletion of other attributes
  - Can’t delete some information without deleting others

Anomalies

Player	Name	Race	Ship Name	Ship Type	Ship Speed	Planet	Planet Colour	Location	Items
1	Kathryn	Human	Voyager	Intrepid	9	Earth	Blue	Alpha	Transporter,Phaser,Torpedo,Hologram
1	Kathryn	Human	Deadalus	Intrepid	8	Qonos	Red	Beta	Transporter,Phaser
2	Borg Queen	Borg	Borg Cube	Cube	10	Borgland	Green	Delta	Transporter,Phaser,Torpedo
3	James	Human	Enterprise	Enterprise	7	Earth	Blue	Alpha	Phaser
3	James	Human	Enterprise B	Enterprise	8	Tetriana Prime	Yellow	Gamma	Transporter,Phaser
4	James	Human	Gauntlet	Sovereign	5	Earth	Blue	Alpha	Transporter,Phaser,Torpedo

- Some examples
  - **Redundancy**: We have a lot of redundant data
  - **Insert**: A new player has to have a race, ship and location when they start
  - **Update**: To change a planet name, will need to update lots of entries
  - **Delete**: Deleting a player may remove all references to a planet or race or location

The Steps to being normal

- **First normal form**: atomicity, find keys
- **Second normal form**: remove partial dependencies
  - Non-key attributes must depend on every part of the primary key
- **Third normal form**: remove transitive dependencies
  - No non-key attributes depend on another non-key attribute
- **Boyce-Codd normal form**: Stricter than 3NF
  - Every attribute must be a fact about a key

- There's also fourth and fifth normal form (but we don't really worry about those)…

Decomposition

- We decompose **R**(A_1,…A_n) by creating
  - **R_1**(B_1,…,B_m)
  - **R_2**(C_1,….C_l)
- Where {B_1,…,B_m} u {C_1,…,C_l} = {A_1,…,A_n}

- R_1 is the projection of R onto B_1,…,B_m
- R_2 is the projection of R onto C_1,…,C_l

R

A1 [Player]	A2 [Name]	A3 [Ship Name]	A4 [Ship Type]
1	Kathryn	Voyager	Intrepid
1	Kathryn	Deadalus	Intrepid
2	Borg Queen	Borg Cube	Cube

R1

B1 (A1)	B2 (A2)
1	Kathryn
2	Borg Queen

R2

C1 (A1)	C2 (A3)	C3 (A4)
1	Voyager	Intrepid
1	Deadalus	Intrepid
2	Borg Cube	Cube

Good Decomposition

- Minimise redundancy
- **Lossless-join**: Avoid information loss
  - Joining back the decomposed relations recovers the original relation
- Preserve the functional dependencies
- Ensure good query performance

Losing Information

Player	Name	Race	Ship Name
1	Kathryn	Human	Voyager
1	Kathryn	Human	Deadalus
2	Borg Queen	Borg	Borg Cube
3	James	Human	Enterprise
3	James	Human	Enterprise B
4	James	Human	Gauntlet

Race	Ship Name
Human	Voyager
Human	Deadalus
Borg	Borg Cube
Human	Enterprise
Human	Enterprise B
Human	Gauntlet

Name	Race
Kathryn	Human
Kathryn	Human
Borg Queen	Borg
James	Human
James	Human
James	Human

Is it possible to put it back together?

Lossless-Join

Player	Name	Race	Ship Name
1	Kathryn	Human	Voyager
1	Kathryn	Human	Deadalus
2	Borg Queen	Borg	Borg Cube
3	James	Human	Enterprise
3	James	Human	Enterprise B
4	James	Human	Gauntlet

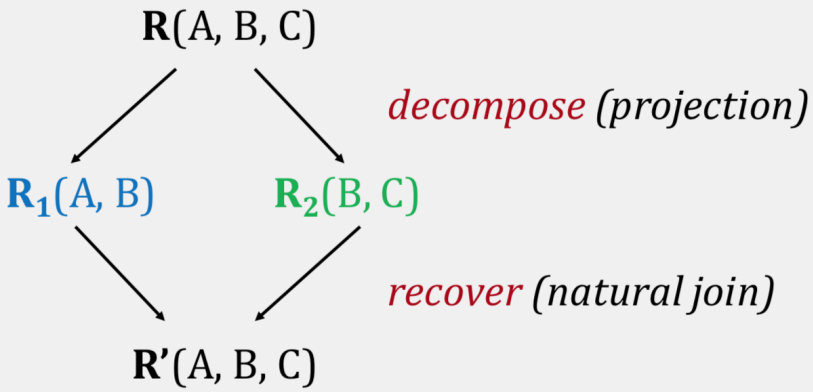
Player	Ship Name
1	Voyager
1	Deadalus
2	Borg Cube
3	Enterprise
3	Enterprise B
4	Gauntlet

Player	Race	Name
1	Human	Kathryn
2	Borg	Borg Queen
3	Human	James
4	Human	James

No loss of information

Can reconstruct the original relation

Allows for recovery



R

A1 [Player]	A2 [Name]	A3 [Ship Name]	A4 [Ship Type]
1	Kathryn	Voyager	Intrepid
1	Kathryn	Deadalus	Intrepid
2	Borg Queen	Borg Cube	Cube

R1

B1 (A1)	B2 (A2)
1	Kathryn
2	Borg Queen

R2

C1 (A1)	C2 (A3)	C3 (A4)
1	Voyager	Intrepid
1	Deadalus	Intrepid
2	Borg Cube	Cube

R'

C1 (A1)	B2 (A2)	C2 (A3)	C3 (A4)
1	Kathryn	Voyager	Intrepid
1	Kathryn	Deadalus	Intrepid
2	Borg Queen	Borg Cube	Cube

Lossless-Join: If for any initial instance

R, R = R'

0 Normal Form

- Unnormalised
- The classic “spreadsheet”

Player	Name	Race	Ship Name	Ship Type	Ship Speed	Planet	Planet Colour	Location	Items
1	Kathryn	Human	Voyager	Intrepid	9	Earth	Blue	Alpha	Transporter,Phaser,Torpedo,Hologram
1	Kathryn	Human	Deadalus	Intrepid	8	Qonos	Red	Beta	Transporter,Phaser
2	Borg Queen	Borg	Borg Cube	Cube	10	Borgland	Green	Delta	Transporter,Phaser,Torpedo
3	James	Human	Enterprise	Enterprise	7	Earth	Blue	Alpha	Phaser
3	James	Human	Enterprise B	Enterprise	8	Tetriana Prime	Yellow	Gamma	Transporter,Phaser
4	James	Human	Gauntlet	Sovereign	5	Earth	Blue	Alpha	Transporter,Phaser,Torpedo

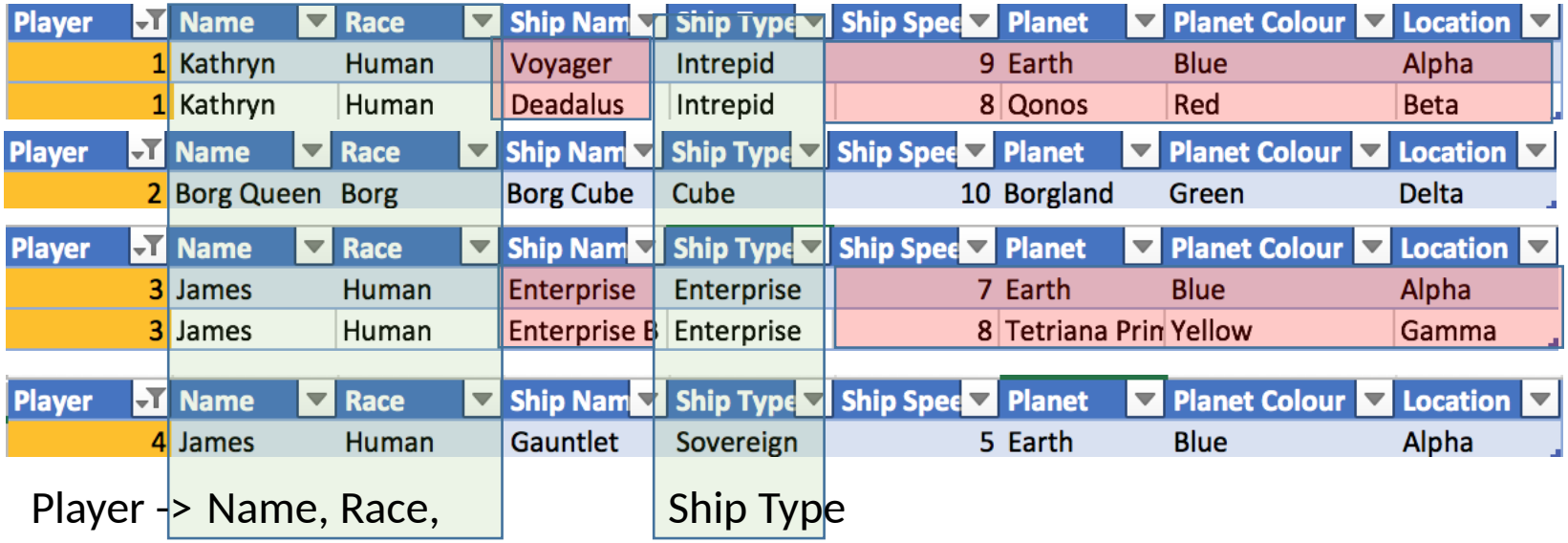
Step 1: Select the primary key

- We need to find the **candidate keys**
  - An attribute or combination of attributes that **uniquely** identifies a row
  - A candidate key is **always** a determinant (on the left hand side in a functional dependency) - knowing it allows us to know other things
  - A special case of a determinant: one which determines *all* attributes of a relation

Examine the dependencies

Player	Name	Race	Ship Name	Ship Type	Ship Speed	Planet	Planet Colour	Location
1	Kathryn	Human	Voyager	Intrepid	9	Earth	Blue	Alpha
1	Kathryn	Human	Deadalus	Intrepid	8	Qonos	Red	Beta
2	Borg Queen	Borg	Borg Cube	Cube	10	Borgland	Green	Delta
3	James	Human	Enterprise	Enterprise	7	Earth	Blue	Alpha
3	James	Human	Enterprise B	Enterprise	8	Tetriana Prin	Yellow	Gamma
4	James	Human	Gauntlet	Sovereign	5	Earth	Blue	Alpha

Functional Dependency A -> B, if we know A, we also know B



Examine the dependencies

Player	Name	Race	Ship Name	Ship Type	Ship Speed	Planet	Planet Colour	Location
1	Kathryn	Human	Voyager	Intrepid	9	Earth	Blue	Alpha
1	Kathryn	Human	Deadalus	Intrepid	8	Qonos	Red	Beta
2	Borg Queen	Borg	Borg Cube	Cube	10	Borgland	Green	Delta
3	James	Human	Enterprise	Enterprise	7	Earth	Blue	Alpha
3	James	Human	Enterprise B	Enterprise	8	Tetriana Prim	Yellow	Gamma
4	James	Human	Gauntlet	Sovereign	5	Earth	Blue	Alpha

Functional Dependency A -> B, if we know A, we also know B

Player -> Name, Race, Ship Type
Name -> *Race*
Race
Ship Name -> *Player*, *Name*, *Race*, *Ship Type*, *Ship Speed*, *Planet*, *Planet Colour*, *Location*
Ship Type -> Race
Ship Speed -> *Race*
Planet -> Planet Colour, Location
Planet Colour -> Planet, Location
Location -> Planet, Planet Colour

Identify a primary key

- We need a **primary key** which **uniquely** identifies **each row** in the relation
  - You could give the entire row, but can we do better?
  - What is the **least information** needed to uniquely identify a single row?
  - Imagine you wanted to delete one row, what is the *least* criteria would you need to give?

Player	Name	Race	Ship Name	Ship Type	Ship Speed	Planet	Planet Colour	Location
1	Kathryn	Human	Voyager	Intrepid	9	Earth	Blue	Alpha
1	Kathryn	Human	Deadalus	Intrepid	8	Qonos	Red	Beta
2	Borg Queen	Borg	Borg Cube	Cube	10	Borgland	Green	Delta
3	James	Human	Enterprise	Enterprise	7	Earth	Blue	Alpha
3	James	Human	Enterprise B	Enterprise	8	Tetriana Prim	Yellow	Gamma
4	James	Human	Gauntlet	Sovereign	5	Earth	Blue	Alpha

Player	Name	Race	Ship Nam	Ship Type	Ship Spee	Planet	Planet Co	Location
1	Kathryn	Human	Voyager	Intrepid	9	Earth	Blue	Alpha
1	Kathryn	Human	Deadalus	Intrepid	8	Qonos	Red	Beta
2	Borg Queen	Borg	Borg Cube	Cube	10	Borgland	Green	Delta
3	James	Human	Enterprise	Enterprise	7	Earth	Blue	Alpha
3	James	Human	Enterprise B	Enterprise	8	Tetriana Prin	Yellow	Gamma
4	James	Human	Gauntlet	Sovereign	5	Earth	Blue	Alpha

Player	Name	Race	Ship Nam	Ship Type	Ship Spee	Planet	Planet Co	Location
1	Kathryn	Human	Voyager	Intrepid	9	Earth	Blue	Alpha
1	Kathryn	Human	Deadalus	Intrepid	8	Qonos	Red	Beta
2	Borg Queen	Borg	Borg Cube	Cube	10	Borgland	Green	Delta
3	James	Human	Enterprise	Enterprise	7	Earth	Blue	Alpha
3	James	Human	Enterprise B	Enterprise	8	Tetriana Prin	Yellow	Gamma
4	James	Human	Gauntlet	Sovereign	5	Earth	Blue	Alpha

Player	Name	Race	Ship Nam	Ship Type	Ship Spee	Planet	Planet Colour	Location
1	Kathryn	Human	Voyager	Intrepid	9	Earth	Blue	Alpha
1	Kathryn	Human	Deadalus	Intrepid	8	Qonos	Red	Beta
2	Borg Queen	Borg	Borg Cube	Cube	10	Borgland	Green	Delta
3	James	Human	Enterprise	Enterprise	7	Earth	Blue	Alpha
3	James	Human	Enterprise B	Enterprise	8	Tetriana Prin	Yellow	Gamma
4	James	Human	Gauntlet	Sovereign	5	Earth	Blue	Alpha

Player	Name	Race	Ship Nam	Ship Type	Ship Spee	Planet	Planet Colour	Location
1	Kathryn	Human	Voyager	Intrepid	9	Earth	Blue	Alpha
1	Kathryn	Human	Deadalus	Intrepid	8	Qonos	Red	Beta
2	Borg Queen	Borg	Borg Cube	Cube	10	Borgland	Green	Delta
3	James	Human	Enterprise	Enterprise	7	Earth	Blue	Alpha
3	James	Human	Enterprise B	Enterprise	8	Tetriana Prin	Yellow	Gamma
4	James	Human	Gauntlet	Sovereign	5	Earth	Blue	Alpha

Player	Name	Race	Ship Nam	Ship Type	Ship Spee	Planet	Planet Colour	Location
1	Kathryn	Human	Voyager	Intrepid	9	Earth	Blue	Alpha
1	Kathryn	Human	Deadalus	Intrepid	8	Qonos	Red	Beta
2	Borg Queen	Borg	Borg Cube	Cube	10	Borgland	Green	Delta
3	James	Human	Enterprise	Enterprise	7	Earth	Blue	Alpha
3	James	Human	Enterprise B	Enterprise	8	Tetriana Prim	Yellow	Gamma
4	James	Human	Gauntlet	Sovereign	5	Earth	Blue	Alpha

Player	Name	Race	Ship Nam	Ship Type	Ship Spee	Planet	Planet Colour	Location
1	Kathryn	Human	Voyager	Intrepid	9	Earth	Blue	Alpha
1	Kathryn	Human	Deadalus	Intrepid	8	Qonos	Red	Beta
2	Borg Queen	Borg	Borg Cube	Cube	10	Borgland	Green	Delta
3	James	Human	Enterprise	Enterprise	7	Earth	Blue	Alpha
3	James	Human	Enterprise B	Enterprise	8	Tetriana Prim	Yellow	Gamma
4	James	Human	Gauntlet	Sovereign	5	Earth	Blue	Alpha

Player	Name	Race	Ship Nam	Ship Type	Ship Spee	Planet	Planet Colour	Location
1	Kathryn	Human	Voyager	Intrepid	9	Earth	Blue	Alpha
1	Kathryn	Human	Deadalus	Intrepid	8	Qonos	Red	Beta
2	Borg Queen	Borg	Borg Cube	Cube	10	Borgland	Green	Delta
3	James	Human	Enterprise	Enterprise	7	Earth	Blue	Alpha
3	James	Human	Enterprise B	Enterprise	8	Tetriana Prim	Yellow	Gamma
4	James	Human	Gauntlet	Sovereign	5	Earth	Blue	Alpha

Player	Name	Race	Ship Nam	Ship Type	Ship Spee	Planet	Planet Colour	Location
1	Kathryn	Human	Voyager	Intrepid	9	Earth	Blue	Alpha
1	Kathryn	Human	Deadalus	Intrepid	8	Qonos	Red	Beta
2	Borg Queen	Borg	Borg Cube	Cube	10	Borgland	Green	Delta
3	James	Human	Enterprise	Enterprise	7	Earth	Blue	Alpha
3	James	Human	Enterprise B	Enterprise	8	Tetriana Prim	Yellow	Gamma
4	James	Human	Gauntlet	Sovereign	5	Earth	Blue	Alpha

We need to consider the domain as well

- What do we know about the domain that might not be present in the data?
- The Borg Queen gets another ship
  - **Ship Type** is no longer functionally dependent on Player
- We have another player join, and they serve on the same ship
  - **Ship Name** is no longer unique

Player	Name	Race	Ship Nam	Ship Type	Ship Spee	Planet	Planet Colour	Location
1	Kathryn	Human	Voyager	Intrepid	9	Earth	Blue	Alpha
1	Kathryn	Human	Deadalus	Intrepid	8	Qonos	Red	Beta
2	Borg Queen	Borg	Borg Cube	Cube	10	Borgland	Green	Delta
2	Borg Queen	Borg	Borg Sphere	Sphere	10	Borgland	Green	Delta
3	James	Human	Enterprise	Enterprise	7	Earth	Blue	Alpha
3	James	Human	Enterprise B	Enterprise	8	Tetriana Prin	Yellow	Gamma
4	James	Human	Gauntlet	Sovereign	5	Earth	Blue	Alpha
5	Andrew	Human	Gauntlet	Sovereign	5	Earth	Blue	Alpha

Composite Keys

- Sometimes, you will need a composite key, made up of multiple attributes, to uniquely identify

- What combined attributes can we use to uniquely identify a line?

Player	Name	Race	Ship Nam	Ship Type	Ship Spee	Planet	Planet Colour	Location
1	Kathryn	Human	Voyager	Intrepid	9	Earth	Blue	Alpha
1	Kathryn	Human	Deadalus	Intrepid	8	Qonos	Red	Beta
2	Borg Queen	Borg	Borg Cube	Cube	10	Borgland	Green	Delta
2	Borg Queen	Borg	Borg Sphere	Sphere	10	Borgland	Green	Delta
3	James	Human	Enterprise	Enterprise	7	Earth	Blue	Alpha
3	James	Human	Enterprise B	Enterprise	8	Tetriana Prin	Yellow	Gamma
4	James	Human	Gauntlet	Sovereign	5	Earth	Blue	Alpha
5	Andrew	Human	Gauntlet	Sovereign	5	Earth	Blue	Alpha

(Player, Ship Name) -> Name, Race, Ship Type, Ship Speed, Planet, Planet Colour, Location

1st Normal Form

- Relation must contain only **atomic** values
  - Cannot be broken down any further
  - Single values
  - Cannot be composite, multi-valued or nested
  - No objections, arrays etc.
- No repeating groups
  - The same attribute across multiple columns

Player	Name	Race	Ship Name	Ship Type	Ship Speed	Planet	Planet Colour	Location	Items
1	Kathryn	Human	Voyager	Intrepid	9	Earth	Blue	Alpha	Transporter,Phaser,Torpedo,Hologram
1	Kathryn	Human	Deadalus	Intrepid	8	Qonos	Red	Beta	Transporter,Phaser
2	Borg Queen	Borg	Borg Cube	Cube	10	Borgland	Green	Delta	Transporter,Phaser,Torpedo
3	James	Human	Enterprise	Enterprise	7	Earth	Blue	Alpha	Phaser
3	James	Human	Enterprise B	Enterprise	8	Tetriana Prime	Yellow	Gamma	Transporter,Phaser
4	James	Human	Gauntlet	Sovereign	5	Earth	Blue	Alpha	Transporter,Phaser,Torpedo

Not Atomic

1st Normal Form

- Relation must contain only **atomic** values
  - Cannot be broken down any further
  - Single values
  - Cannot be composite, multi-valued or nested
  - No objections, arrays etc.
- No repeating groups
  - The same attribute across multiple columns

Player	Name	Race	Ship Name	Ship Type	Ship Speed	Planet	Planet Colour	Location	Item 1	Item 2	Item 3	Item 4
1	Kathryn	Human	Voyager	Intrepid	9	Earth	Blue	Alpha	Transporter	Phaser	Torpedo	Hologram
1	Kathryn	Human	Deadalus	Intrepid	8	Qonos	Red	Beta	Transporter	Phaser		
2	Borg Queen	Borg	Borg Cube	Cube	10	Borgland	Green	Delta	Transporter	Phaser	Torpedo	
3	James	Human	Enterprise	Enterprise	7	Earth	Blue	Alpha	Phaser			
3	James	Human	Enterprise B	Enterprise	8	Tetriana Prime	Yellow	Gamma	Transporter	Phaser		
4	James	Human	Gauntlet	Sovereign	5	Earth	Blue	Alpha	Transporter	Phaser	Torpedo	

Repeating Group

1st Normal Form

- Identify any non-atomic values

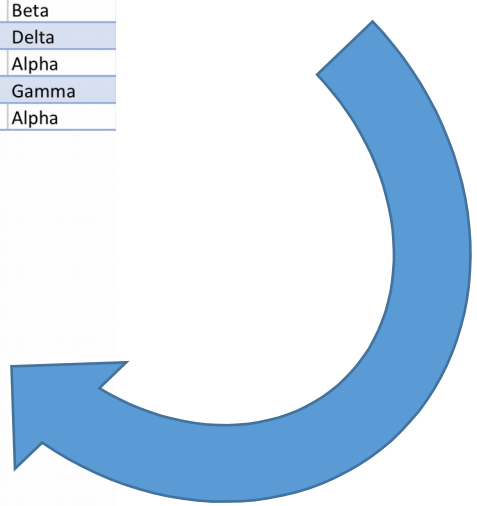
Player	Name	Race	Ship Name	Ship Type	Ship Speed	Planet	Planet Colour	Location	Items
1	Kathryn	Human	Voyager	Intrepid	9	Earth	Blue	Alpha	Transporter,Phaser,Torpedo,Hologram
1	Kathryn	Human	Deadalus	Intrepid	8	Qonos	Red	Beta	Transporter,Phaser
2	Borg Queen	Borg	Borg Cube	Cube	10	Borgland	Green	Delta	Transporter,Phaser,Torpedo
3	James	Human	Enterprise	Enterprise	7	Earth	Blue	Alpha	Phaser
3	James	Human	Enterprise B	Enterprise	8	Tetriana Prime	Yellow	Gamma	Transporter,Phaser
4	James	Human	Gauntlet	Sovereign	5	Earth	Blue	Alpha	Transporter,Phaser,Torpedo

Player	Name	Race	Ship Name	Ship Type	Ship Speed	Planet	Planet Colou	Location
1	Kathryn	Human	Voyager	Intrepid	9	Earth	Blue	Alpha
1	Kathryn	Human	Deadalus	Intrepid	8	Qonos	Red	Beta
2	Borg Queen	Borg	Borg Cube	Cube	10	Borgland	Green	Delta
3	James	Human	Enterprise	Enterprise	7	Earth	Blue	Alpha
3	James	Human	Enterprise B	Enterprise	8	Tetriana Prim	Yellow	Gamma
4	James	Human	Gauntlet	Sovereign	5	Earth	Blue	Alpha

Player	Ship	Item
1	Voyager	Transporter
1	Voyager	Phaser
1	Voyager	Torpedo
1	Voyager	Hologram
1	Deadalus	Transporter
1	Deadalus	Phaser
2	Borg Cube	Transporter
2	Borg Cube	Phaser
2	Borg Cube	Torpedo
3	Enterprise	Phaser
3	Enterprise B	Transporter
3	Enterprise B	Phaser
4	Gauntlet	Transporter
4	Gauntlet	Phaser
4	Gauntlet	Torpedo

Using our key:

(Player, Ship)



Decompose into a new relation

2nd Normal Form

- **No partial-key dependencies** allowed
  - Every non-prime attribute (not part of a candidate key) is **dependent on all attributes of a candidate key**
  - Every non-key attribute is **fully functionally dependent** on the primary key (needs the full primary key for identification)
- Identify primary key and functional dependencies in the relation
  - If partial dependencies exist on the primary key, decompose into a new relation
- TLDR: All attributes must be dependent on all parts of the key

2nd Normal Form

- Put simply:

  K -> A_i for each non-key attribute A_i
- There is no subset K' such that K' -> A_i

- Each non-key attribute is fully functionally dependent on the primary key (every attribute in the primary key)

Examine the dependencies

Player	Name	Race	Ship Nam	Ship Type	Ship Spee	Planet	Planet Colour	Location
1	Kathryn	Human	Voyager	Intrepid	9	Earth	Blue	Alpha
1	Kathryn	Human	Deadalus	Intrepid	8	Qonos	Red	Beta
2	Borg Queen	Borg	Borg Cube	Cube	10	Borgland	Green	Delta
2	Borg Queen	Borg	Borg Sphere	Sphere	10	Borgland	Green	Delta
3	James	Human	Enterprise	Enterprise	7	Earth	Blue	Alpha
3	James	Human	Enterprise B	Enterprise	8	Tetriana Prin	Yellow	Gamma
4	James	Human	Gauntlet	Sovereign	5	Earth	Blue	Alpha
5	Andrew	Human	Gauntlet	Sovereign	5	Earth	Blue	Alpha

Functional Dependency A -> B, if we know A, we also know B

(Player, Ship Name) -> Name, Race, Ship Type, Ship Speed, Planet, Planet Colour, Location
Player -> Name, Race
Ship Name -> Ship Type, Ship Speed, Planet, Planet Colour, Location

Only partially dependent

Player	Name	Race	Ship Nam	Ship Type	Ship Spee	Planet	Planet Colour	Location
1	Kathryn	Human	Voyager	Intrepid	9	Earth	Blue	Alpha
1	Kathryn	Human	Deadalus	Intrepid	8	Qonos	Red	Beta
2	Borg Queen	Borg	Borg Cube	Cube	10	Borgland	Green	Delta
2	Borg Queen	Borg	Borg Sphere	Sphere	10	Borgland	Green	Delta
3	James	Human	Enterprise	Enterprise	7	Earth	Blue	Alpha
3	James	Human	Enterprise B	Enterprise	8	Tetriana Prin	Yellow	Gamma
4	James	Human	Gauntlet	Sovereign	5	Earth	Blue	Alpha
5	Andrew	Human	Gauntlet	Sovereign	5	Earth	Blue	Alpha

Further decomposition

Player	Name	Race	Ship Nam	Ship Type	Ship Spee	Planet	Planet Colour	Location
1	Kathryn	Human	Voyager	Intrepid	9	Earth	Blue	Alpha
1	Kathryn	Human	Deadalus	Intrepid	8	Qonos	Red	Beta
2	Borg Queen	Borg	Borg Cube	Cube	10	Borgland	Green	Delta
2	Borg Queen	Borg	Borg Sphere	Sphere	10	Borgland	Green	Delta
3	James	Human	Enterprise	Enterprise	7	Earth	Blue	Alpha
3	James	Human	Enterprise B	Enterprise	8	Tetriana Prin	Yellow	Gamma
4	James	Human	Gauntlet	Sovereign	5	Earth	Blue	Alpha
5	Andrew	Human	Gauntlet	Sovereign	5	Earth	Blue	Alpha

Player	Name	Race	Ship Nam
1	Kathryn	Human	Voyager
1	Kathryn	Human	Deadalus
2	Borg Queen	Borg	Borg Cube
2	Borg Queen	Borg	Borg Sphere
3	James	Human	Enterprise
3	James	Human	Enterprise B
4	James	Human	Gauntlet
5	Andrew	Human	Gauntlet

Copying the determinant – the part of the key they are dependent on

Decompose into a new relation

Ship Name	Ship Type	Ship Speed	Planet	Planet Colour	Location
Voyager	Intrepid	9	Earth	Blue	Alpha
Deadalus	Intrepid	8	Qonos	Red	Beta
Borg Cube	Cube	10	Borgland	Green	Delta
Borg Sphere	Sphere	10	Borgland	Green	Delta
Enterprise	Enterprise	7	Earth	Blue	Alpha
Enterprise B	Enterprise	8	Tetriana Prin	Yellow	Gamma
Gauntlet	Sovereign	5	Earth	Blue	Alpha

We lose the redundant data!

Further decomposition

Player	Name	Race	Ship Name
1	Kathryn	Human	Voyager
1	Kathryn	Human	Deadalus
2	Borg Queen	Borg	Borg Cube
2	Borg Queen	Borg	Borg Sphere
3	James	Human	Enterprise
3	James	Human	Enterprise B
4	James	Human	Gauntlet
5	Andrew	Human	Gauntlet

But we're not done yet!

Primary key is:
(Player, Ship Name)

(Player, Ship Name) -> Name, Race

Player -> Name, Race

Name and Race are only **partially** dependent on the key (Player, Ship Name)

Continue to apply decomposition

Player	Name	Race
1	Kathryn	Human
2	Borg Queen	Borg
3	James	Human
4	James	Human
5	Andrew	Human

Player	Ship Name
1	Voyager
1	Deadalus
2	Borg Cube
2	Borg Sphere
3	Enterprise
3	Enterprise B
4	Gauntlet
5	Gauntlet

And we lose more redundant data!

Copying the determinant – the part of the key they are dependent on

3rd Normal Form

- Non-prime attributes are **only** dependent on candidate keys
  - They are not dependent on any non-prime attribute
  - No non-prime attribute is **transitively dependent** on the primary key

- Put formally,
  - Whenever $A_1,\ldots,A_n$ -> $B_1,\ldots,B_n$ is a non-trivial functional dependency
    - Either $\{A_1,\ldots A_n\}$ is a super key
    - Or those of $B_1,\ldots,B_n$ that are not among the A's are each a member of some key (not necessarily the same key)
- TLDR: All attributes determined only by the key

Trivial dependency

- A -> B is a trivial functional dependency if B is a subset of A

- For example,
- Player -> Player
- {Player, Ship Name} -> Player
- {Player, Ship Name} -> Ship Name

Player	Name	Race	Ship Nam	Ship Type	Ship Spee	Planet	Planet Colour	Location
1	Kathryn	Human	Voyager	Intrepid	9	Earth	Blue	Alpha
1	Kathryn	Human	Deadalus	Intrepid	8	Qonos	Red	Beta
2	Borg Queen	Borg	Borg Cube	Cube	10	Borgland	Green	Delta
2	Borg Queen	Borg	Borg Sphere	Sphere	10	Borgland	Green	Delta
3	James	Human	Enterprise	Enterprise	7	Earth	Blue	Alpha
3	James	Human	Enterprise B	Enterprise	8	Tetriana Prin	Yellow	Gamma
4	James	Human	Gauntlet	Sovereign	5	Earth	Blue	Alpha
5	Andrew	Human	Gauntlet	Sovereign	5	Earth	Blue	Alpha

Transitive Dependence

- If A -> B -> C (but it is not the case that B -> A)
- Then A -> C
  - Transitive dependency

- If K $\not\rightarrow$ A_i (2^{nd} Normal Form), there must be no set of attributes X such that K -> X -> A_i

Transitive Dependence

Ship Name	Ship Type	Ship Speed	Planet	Planet Colour	Location
Voyager	Intrepid	9	Earth	Blue	Alpha
Deadalus	Intrepid	8	Qonos	Red	Beta
Borg Cube	Cube	10	Borgland	Green	Delta
Borg Sphere	Sphere	10	Borgland	Green	Delta
Enterprise	Enterprise	7	Earth	Blue	Alpha
Enterprise B	Enterprise	8	Tetriana Prime	Yellow	Gamma
Gauntlet	Sovereign	5	Earth	Blue	Alpha

Ship Name -> Ship Type, Ship Speed, Planet, Planet Colour, Location
Planet -> Planet Colour, Location

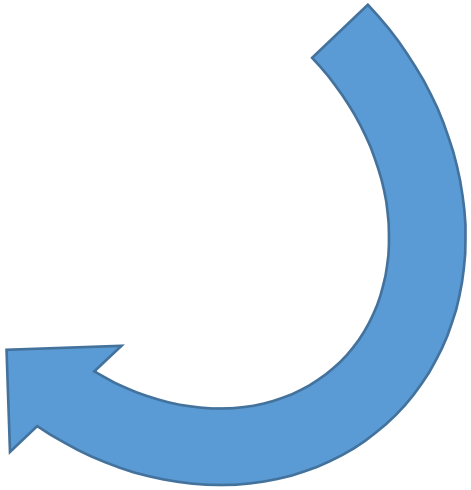
Ship Name -> Planet -> Planet Colour, Location

Planet Colour and Location have transitive dependence on the Ship Name via Planet

Further Decomposition

Ship Name	Ship Type	Ship Speed	Planet	Planet Colour	Location
Voyager	Intrepid	9	Earth	Blue	Alpha
Deadalus	Intrepid	8	Qonos	Red	Beta
Borg Cube	Cube	10	Borgland	Green	Delta
Borg Sphere	Sphere	10	Borgland	Green	Delta
Enterprise	Enterprise	7	Earth	Blue	Alpha
Enterprise B	Enterprise	8	Tetriana Prime	Yellow	Gamma
Gauntlet	Sovereign	5	Earth	Blue	Alpha

Decompose into a new relation



Ship Name	Ship Type	Ship Speed	Planet
Voyager	Intrepid	9	Earth
Deadalus	Intrepid	8	Qonos
Borg Cube	Cube	10	Borgland
Borg Sphere	Sphere	10	Borgland
Enterprise	Enterprise	7	Earth
Enterprise B	Enterprise	8	Tetriana Prime
Gauntlet	Sovereign	5	Earth

Planet	Planet Colour	Location
Earth	Blue	Alpha
Qonos	Red	Beta
Borgland	Green	Delta
Tetriana Prime	Yellow	Gamma

Copying the determinant – the part of the key they are dependent on

We lose the redundant data!

Boyce-Codd Normal Form

- Every relation in BCNF is also 3NF
  - But not vice versa
- It is a slightly stronger version of 3NF (Sometimes called 3.5NF)
- One further restriction on 3NF…
  - **Every determinant is a candidate key** (can uniquely identify any row)
- Formally,
  - If A_1,…,A_n -> B is a non-trivial dependency in R, then {A_1, …,A_n} is a superkey for R

Boyce-Codd Normal Form

- A table is in 3NF but not BCNF if
  - The table has two or more candidate keys
  - At least two of the candidate keys are composed of more than one attribute
  - The keys are not disjoint: the composite candidate keys share some attributes

Back to our example

- Each ship has a defence slot, offense slot and enhancement slot for enhancement items which can be equipped
- We want to keep track of what items are in what slots on each ship
- So we have a relation of Ship, Slot and Item
  - For example: Deadalus has a shield in it’s defence slot and a phaser in it’s offence slot

It’s 3NF…

- Our new table is 3NF
  - (Ship, Slot) -> Item
  - (Ship, Item) -> Slot
  - Item -> Slot
- There are no partial key dependencies (2NF)
  - To determine item, we need to know the ship **and** the slot
- Non-prime key are only dependent on candidate keys (3NF)
  - Item is **only** functionally dependent on (Ship, Slot) which is a candidate (and the primary) key

Ship	Slot	Item
Voyager	Defense	Armour
Deadalus	Defense	Shield
Deadalus	Offense	Phaser
Deadalus	Enhancement	Hologram
Borg Cube	Defense	Armour
Borg Cube	Offense	Torpedo
Borg Sphere	Defense	Deflector
Enterprise	Defense	Shield

…but it's not BCNF

- (Ship, Slot) -> Item
- (Ship, Item) -> Slot
- Item -> Slot

- BCNF Restriction: Every determinant (an attribute to determine others) is a candidate key
  - (Ship,Slot) and (Ship,Item) are both candidate keys (can uniquely identify a row)
  - Item is **not** a candidate key (it alone can't uniquely identify a row)

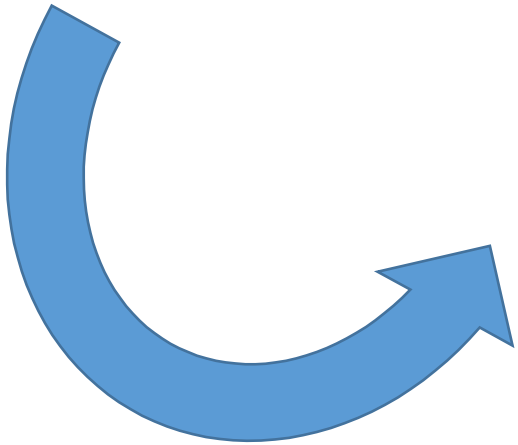
Ship	Slot	Item
Voyager	Defense	Armour
Deadalus	Defense	Shield
Deadalus	Offense	Phaser
Deadalus	Enhancement	Hologram
Borg Cube	Defense	Armour
Borg Cube	Offense	Torpedo
Borg Sphere	Defense	Deflector
Enterprise	Defense	Shield

The result: Redundant data!

If we know the item, we know slot it goes into

BCNF Decomposition

Ship	Slot	Item
Voyager	Defense	Armour
Deadalus	Defense	Shield
Deadalus	Offense	Phaser
Deadalus	Enhancement	Hologram
Borg Cube	Defense	Armour
Borg Cube	Offense	Torpedo
Borg Sphere	Defense	Deflector
Enterprise	Defense	Shield



Ship	Item
Voyager	Armour
Deadalus	Shield
Deadalus	Phaser
Deadalus	Hologram
Borg Cube	Armour
Borg Cube	Torpedo
Borg Sphere	Deflector
Enterprise	Shield

Item	Slot
Armour	Defense
Shield	Defense
Phaser	Offense
Hologram	Enhancement
Torpedo	Offense
Deflector	Defense

And we lose more redundant data!

What made sense as a logical concept introduced redundancy and didn't make sense from a database perspective

Why is it better?

- No redundancy
  - Less storage space required
- Efficiency
  - Less data to search through -> faster queries
- No duplication
  - Better integrity
  - Only have to modify something in 1 place
  - Less chance of mistakes
- Changes can cascade across relations
  - Example: Remove a person and anything related to them

What are the consequences?

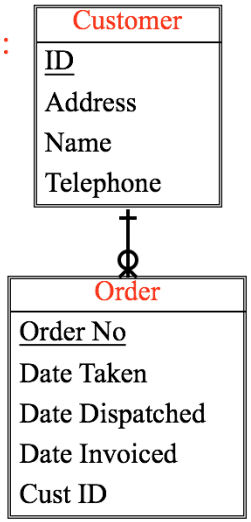
- More tables
- More complexity
- More relationships
- Queries become more complex

- But it’s worth it!
  - How many times have you been annoyed by a spreadsheet?

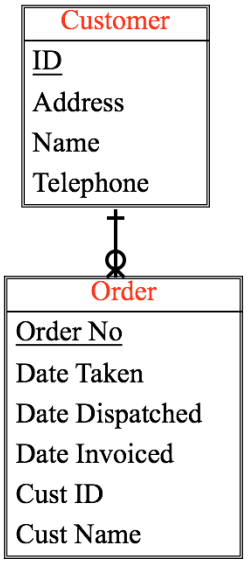
Denormalisation?

- Driven by the need to improve query speed
  - At the expense of less normalisation
  - More complex operations required to updates, deletions and insertions to avoid integrity issues

Before:



After:



Including the Customer Name in the order, as well as in the customer table

Summary

- Normalisation
- Normal forms
- Issues with normalising
- Not to leave submitting coursework to the last minute ;-)