**Normalisation**

**Relations** – the most **important** stuff:

**Properties** of relations: A relation R(A1,…Ak) has the following properties:

1) Each **row** represents a **k-tuple** of R.

2) The **ordering** of **rows** is **immaterial**.

3) **All** **rows** are **distinct**.

4) The **ordering** of the **attributes** is **not** **significant**.

5) The **significance** of **each** **column** is conveyed by the **name** we give it.

**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.

**Functional** **dependencies**: Let a relation with schema R(A1,…,An,B1,B2,….,Bm) and let **r** an instance of **R**.

We say that **r** **satisfies** the **functional** **dependency** A1,…,An -> B1,B2,…Bm if there are **no** **two** **tuples** in **r** that have the **same** **value** in the attributes **A1,…An** **but** **differ** on the **values of B1,B2,…,Bm**.

Back to normalisation:

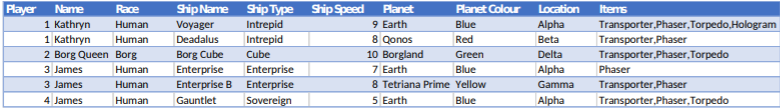
**Anomalies** = bad relations.

Types of anomalies:

* **Redundancy**: information unnecessarily repeated.
* **Update** **Anomalies**: change information in one tuple and leave same info unchanged in another.
* **Insert** **Anomalies**: we could insert data incorrectly.
* **Deletion** **Anomalies**.

**Normalisation** **deals** **with** **anomalies**. Normalisation helps the user avoid redundant data, increases the efficiency and decreases the size. We **accomplish** **normalisation** by: **dividing the database into multiple tables and defining the relationships between them.**

**LOOK AT PDF. The presentation has all the examples explained.**



Example anomalies:

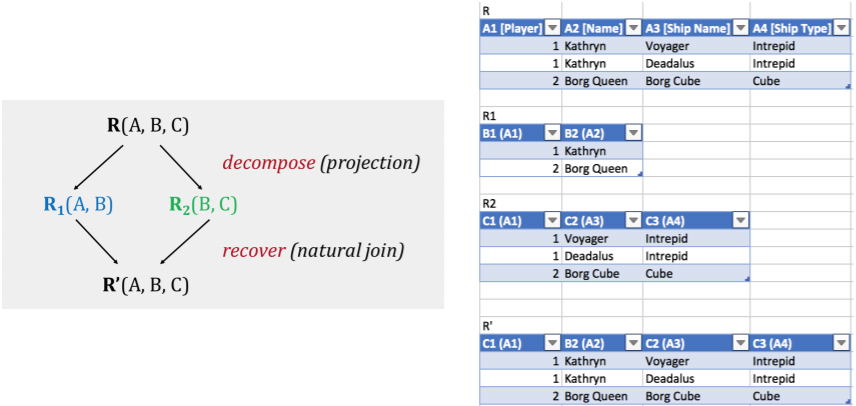
* **Redundancy**: Same data held in multiple locations.
* **Insert**: A new player has to have a race, ship and location when they start. Attributes cannot be inserted without needing the presence of other attributes.
* **Update**: To change a planet name, will need to update lots of entries.
* **Delete**: Deleting a player may remove all references to a planet or race or location.

**Steps for achieving normalisation:**

* **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**: every attribute must be a fact about a key. (**Strict** **3NF**).

**Side note:**

**Decomposition: Splitting** a **table** **without** **losing** any **data**. It **minimises** **redundancy**, **preserves** the **functional** **dependencies** and **ensures** **good** **query** **performance**.



**Achieving normalisation:**

**Step 1: Select the primary key**

We need to find the **candidate keys** – an **attribute** **or** **combination** **of** **attributes** that **uniquely** identifies a row. (Imagine you wanted to delete a single row, what is the least criteria you would need to give).

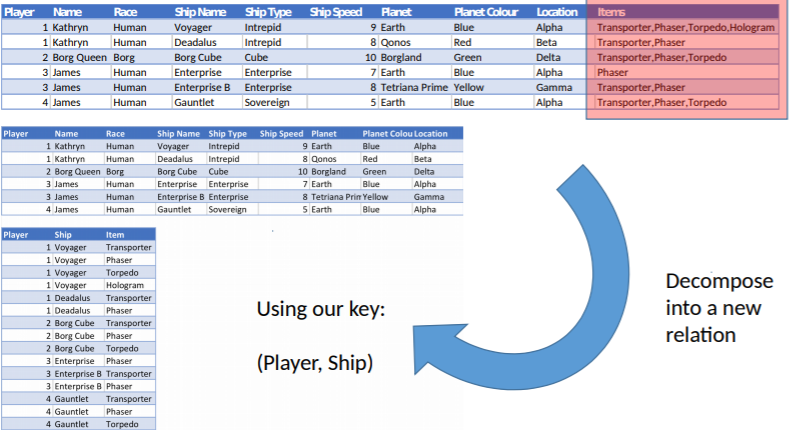
**Candidate keys** made from multiple attributes are called **composite keys.**

**First normal form:**

A **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).



**Second normal form:**

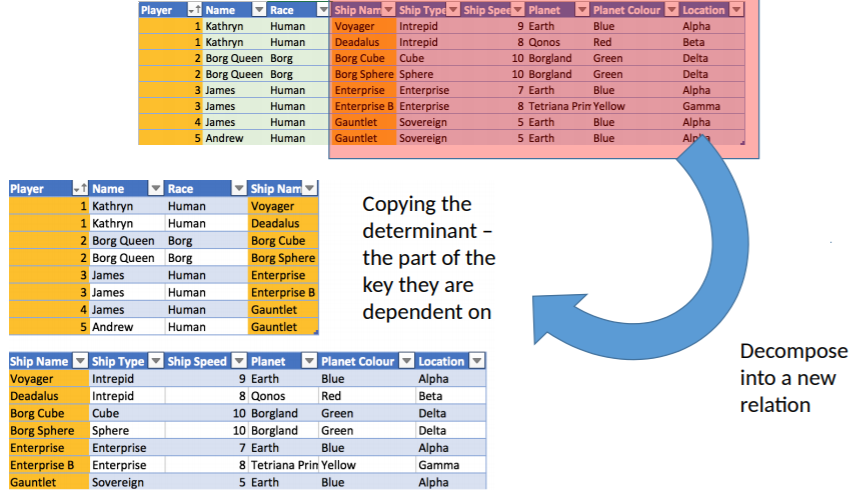
No partial-key dependencies allowed

* 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 (need the full primary key for identification).

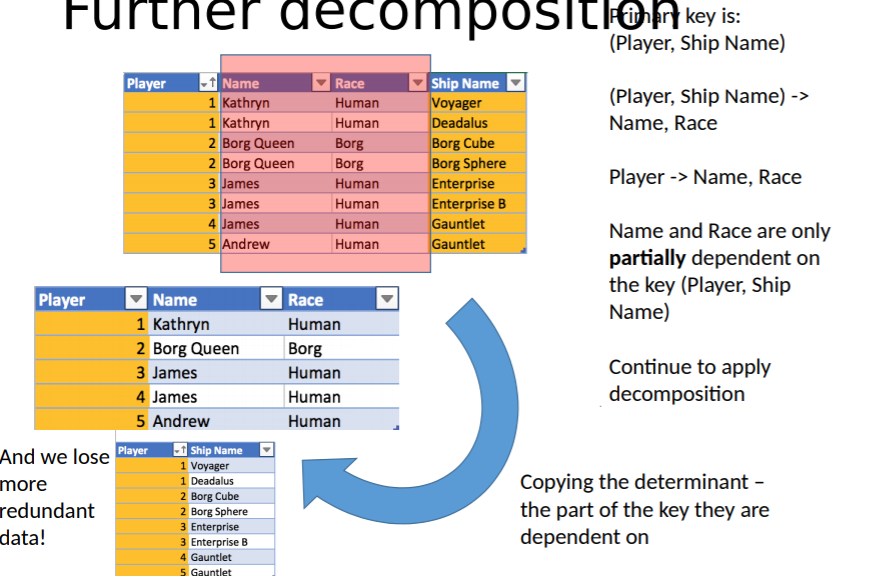
Identify primary key and functional dependencies in the relation.

* 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.**



And more:



**Third normal form:**

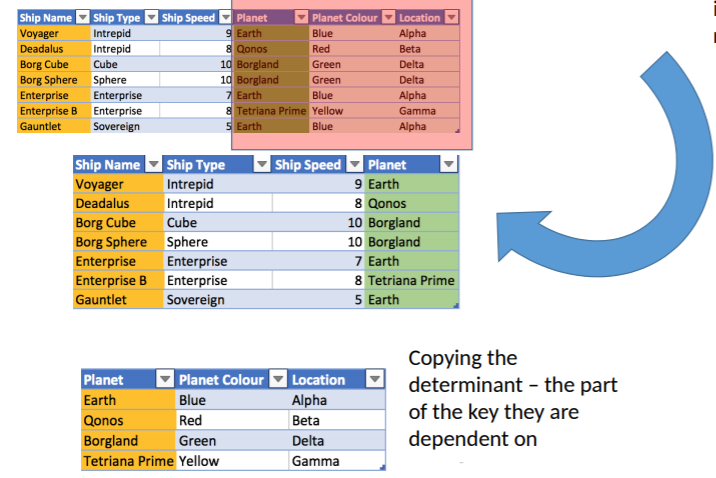
Non-prime attributes are only dependent on candidate keys.

* They are not dependent on any non-prime attribute.
* No non-prime attribute is transitively dependent on the primary key.

**TLDR: All attributes are determined only by the key**.

**We must implement transitive dependency (**That’s how I get it**):**

If A -> B -> C then A -> C



**Boyce-Codd Normal Form:**

Every relation in BCNF is also 3NF (but not vice versa).

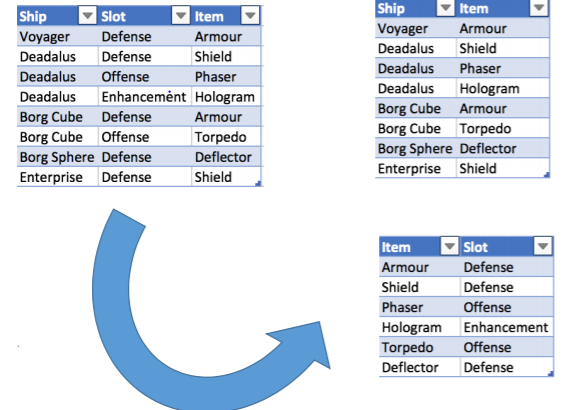
It is slightly stronger version of 3NF. Just has one more restriction on top of 3NF -> **every determinant is a candidate key.**

**(A determinant is a column. i.e. all candidate keys or primary keys are determinants but not all determinants are candidate keys or primary keys.)**

**(Diff between candidate and primary key -> only one candidate key can be a primary key. There can be multiple candidate key in one table. Each candidate key can qualify as a primary key).**

A table is in 3NF but not BCNF if:

* The table has two or move 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.



Why is normalisation better?

* No redundancy – less storage space required.
* Efficiency – less data to search through.
* No duplication – only have to modify in 1 place.
* Changed can cascade across relations.

But there are consequences to normalisation:

* More tables.
* More complexity.
* More relationships.
* Queries become more complex.