1 Normalization

There is a concept of 'Normal Forms' that is often[1] used to design and judge the quality of design of databases.

There are five normal forms. They're numbered—conveniently enough—one thorough five. There are also a bunch of other intermediate forms named after something or other (usually whoever came up with it).

Higher numbered normal forms have all the goodness qualities of the lower forms. For example, third-normal form database is also in second-normal form and first normal form. Without putting too fine a point on it, we want our databases to be in as higher normal form as possible. Higher is better.

Generally, you're only concerned about the first three normal forms—and in this class, we'll also be concerned with Boyce-Codd Normal Form.

1.1 First-Normal Form

This is the most basic normal form, and the only requirement is that data is stored in tables. If your data is stored in tables, then you've achieved first-normal form.

1.2 Second Normal Form

Ok, here it goes:

> "A database is in second-normal form if it is in first-normal form and every attribute is *fully* functionally dependent on the primary key."

And now to explain it: primary keys are fields that uniquely identify a record. Attributes are everything else in the record.

Now, *functionally dependent* means that given a primary key, we can get the value of any attribute. For example, given your student id, we can find your first name and last name: your first and last name are *functionally dependent* on your student id.

The *fully functionally dependent* mostly applies to composite primary keys. It basically means that the attribute needs to be functionally dependent on the *whole* primary key (not one of its parts). For example, some applications use first name, last name, and date of birth as a composite primary key. Every attribute in that record needs to depend on *all* first name, last name, and date of birth.

1.2.1 Design

It is very easy to achieve second-normal form by simply choosing a unique abstract primary key that doesn't depend on the data. i.e.: adding an auto-increment primary key to your tables.

[1] Which is to say *never*.

1.3 Third-Normal Form

> "A database is in third-normal form if it is in second-normal form and contains no transitive attribute dependencies."

This one is a bit easier. What this means is that none of our attributes imply any other attributes. For example, in a "Person" table, we might have person's phone, address, and zip code. More often than not, the zip code will imply part of the street address, and possibly the phone's area code. This is precisely what third-normal form must avoid.

You can't generally avoid this problem in real life—when you're using someone else's identification systems in your databases. Let's just say that for 'common' things like addresses and zip codes, it is 'ok', while for your own data, no field should imply other fields (ie: you shouldn't have 'date of birth' and 'age' in your database, etc.).

1.4 Boyce-Codd Normal Form

> "A relation R is in Boyce-Codd normal form if its primary key, K, implies all nonkey attributes—$a, b, c, \ldots$—and K is a superkey."

Well, a key is a *superkey* if it implies all other attributes and no portion of K less than the whole implies all other attributes.

Basically if we pick a new unique id for each record (the auto-increment, etc.) and pick fields 'appropriately' then we'll get Boyce-Codd normal form.

The Boyce-Codd normal form doesn't exactly fit into the numbered normal form idea: it starts with normal form (assumes that data is in tables). It has been proved[2] that Boyce-Codd is also in third-normal form. So we can consider Boyce-Codd as a 3.5-normal form.

[2] By someone.