INFINITYWORKS
ACADEMY

Generation

Data Normalisation

Overview

- An introduction to data normalisation
- Advantages and Disadvantages
- Normal Forms

Learning Objectives

- Be able to explain what data normalisation is
- Understand some of the problems that normalisation helps to fix
- Practice implementing the steps to reach first, second and third normal form
- List some of the Pros and Cons of normalisation

What is Data Normalisation?

Formally: The process of structuring a relational database in accordance with a series of defined normal forms in order to reduce data redundancy and increase data integrity.

Informally: Organising our data to make it easy to work with, reduce duplication, and increase accuracy.

Why do we Normalise data?

The need to normalise is best seen by looking at the problems we might encounter when data is not normalised!

Unnormalised Data - Example

student_number	student_name	course	enrolled	modules	course_instructor	instructor_email
123	Joe Bloggs	Data Eng	2020-10-23	Python, Docker	Sarah D	sarahd@example.com
456	Jane Doe	Data Eng	2020-10-10	Python, Databases	Sarah D	sarahd@example.com
789	Tom Jones	Data Eng	Sickness	Python	Sarah D	sarahd@example.com

- What looks strange about this data? What possible problems can we see?

Unnormalised Data - Problems

student_number	student_name	course	enrolled	modules	course_instructor	instructor_email
123	Joe Bloggs	Data Eng	2020-10-23	Python, Docker	Sarah D	sarahd@example.com
456	Jane Doe	Data Eng	2020-10-10	Python, Databases	Sarah D	sarahd@example.com
789	Tom Jones	Data Eng	Sickness	Python	Sarah D	sarahd@example.com

- Duplicate data - extra storage is used and updates need to be made in multiple places when something is changed
- Multiple values in a single column - we cannot easily work with that data when searching or querying
- Different data types in a single column - we cannot predict what data will be held for a given row
- Data about different entities (Student, Course and Instructor) in the same row - can lead to unintentional loss of data when rows are removed

Anomalies

Anomaly

An `Anomaly` is a data problem caused by insufficient normalisation in our table, which can occur when we try to change the data that table holds.

Three formal anomalies:

1. Insertion Anomaly
2. Update Anomaly
3. Deletion Anomaly

Insertion Anomaly

- An Inability to add data due to dependence on other data.

Example scenario: A new Course Instructor starts work, but does not have any students enrolled yet

student_number	student_name	course	enrolled	modules	course_instructor	instructor_email
123	Joe Bloggs	Data Eng	2020-10-23	Python, Docker	Sarah D	sarahd@example.com
456	Jane Doe	Data Eng	2020-10-10	Python, Databases	Sarah D	sarahd@example.com
??	??	Ethics	??	??	Tom New	tomnew@example.com

Trying to add this data would result in empty (`null`) values for student fields. Student number and course are both required in this table to uniquely identify a row as its `composite key`

Adding the instructor is blocked until they have a student.

Update Anomaly

- Inconsistency caused when duplicated data is only partially changed

Example scenario: An instructor updates their email address

student_number	student_name	course	enrolled	modules	course_instructor	instructor_email
123	Joe Bloggs	Data Eng	2020-10-23	Python, Docker	Sarah D	sarahd_new@example.com
456	Jane Doe	Data Eng	2020-10-10	Python, Databases	Sarah D	sarahd@example.com
789	Tom Jones	Data Eng	Sickness	Python	Sarah D	sarahd_new@example.com

The email address must change in many records, but it is possible some of these records may be missed. There are now two different values in the database for the same instructor's email.

Deletion Anomaly

- Unintended loss of data when other unrelated data is removed

Example scenario: When a student's course ends, their row is deleted from the database

student_number	student_name	course	enrolled	modules	course_instructor	instructor_email
123	Joe Bloggs	Data Eng	2020-10-23	Python, Docker	Sarah D	sarahd@example.com
456	Jane Doe	Data Eng	2020-10-10	Python, Databases	Sarah D	sarahd@example.com
202	Matt Binns	Ethics	2020-01-05	Business, Personal	Tom New	tomnew@example.com

If all students for a particular instructor are removed, there will be no copies of that instructor's name or email left, and that data has then been lost.

When Matt Binns is removed, we also delete information about Tom New.

Normalisation to the Rescue! 🎉

We can solve these problems with `Data Normalisation.`

After normalising:

- Redundant data (duplicate data) is eliminated
- Changes to data values are always consistent (because there is no duplication!)
  - Update Anomaly Fixed ✅
- Unintended removal of unrelated data is prevented
  - Deletion Anomaly Fixed ✅
- New data is always able to be added without inappropriately depending on other data
  - Insertion Anomaly Fixed ✅

Further Benefits of Normalisation

- Storage space on disk is reduced
- Ordering can be added to the data with `indexing` to speed up searching and access
- Data is grouped together logically in related tables, in a way that is easy for humans to understand and write code for

Quiz Time! 🤓

Which of the below best describes 'Data Normalisation' ?

1. Removing all unnecessary data from a database.
2. Organising fields from one or more tables into smaller tables.
3. Organising data to remove duplicates and increase data accuracy.
4. Making sure all tables have unique primary keys.

Answer: 3

Which of these is NOT a database anomaly?

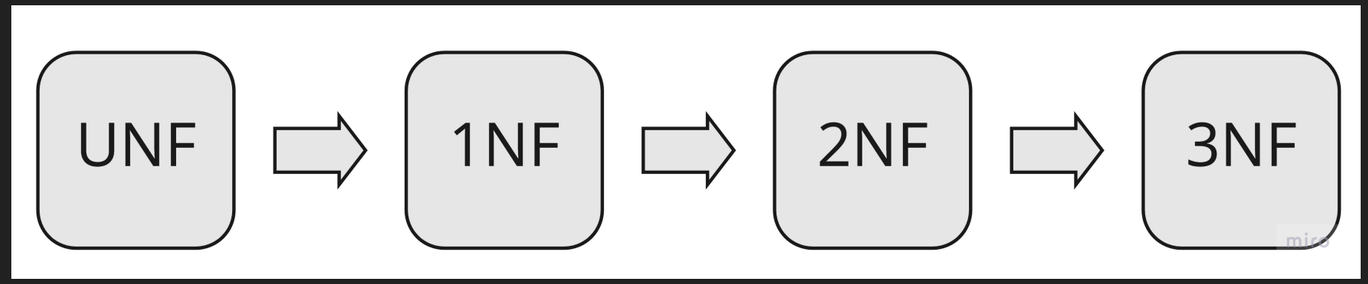
1. Deletion Anomaly
2. Read Anomaly
3. Update Anomaly
4. Insert Anomaly

Answer: 2

The Normalisation Process

How do we Normalise?

Starting from an Unnormalised data set (UNF) we can proceed through each `normal form` applying rules at each stage until we reach Third Normal Form (3NF).



If data already meets the rules for a given normal form, we may not need to do anything and can simply move to the next form!

The Normalisation Process

Normalisation is usually an "on-paper" exercise - it doesn't happen in the database!

The output from normalisation will be the information needed to technically implement a database structure in the correct way:

- Required `Tables`
- Data `Fields` each table should store
- `Primary keys` and `Foreign keys` that relate those tables together

Rules for First Normal Form (1NF)

For a table to be in 1NF then:

1. Unique name for attributes/columns: Each column should have a unique name
2. Order doesn't matter: The order in which columns or rows appear should not affect the data
3. Attribute Domain should not change: Values stored in a column must represent the same sort of information for all rows/records
4. Single Valued Attributes: Each column in the table should hold a single value. (Formal term is to be `atomic`)

Applying 1NF

1. Unique name for attributes/columns
2. Data order doesn't matter

These requirements are usually trivial and we won't need to think about them much.

- The database software will prevent us from creating any tables that have duplicate column names.
- The database software will also always allow columns to be in any order or for rows to be retrieved based on queries, no matter the order those rows actually appear.

Let's move on to the next requirements.

Applying 1NF

- **Attribute Domain should not change**: Values stored in a column must represent the same sort of information for all rows/records

student_number	student_name	course	enrolled	modules	course_instructor	instructor_email
123	Joe Bloggs	Data Eng	2020-10-23	Python, Docker	Sarah D	sarahd@example.com
456	Jane Doe	Data Eng	2020-10-10	Python, Databases	Sarah D	sarahd@example.com
789	Tom Jones	Data Eng	Sickness	Python	Sarah D	sarahd@example.com

Is the rule violated?

Applying 1NF

Violation: Mixed information for enrolled column

student_number	student_name	course	enrolled	modules	course_instructor	instructor_email
123	Joe Bloggs	Data Eng	2020-10-23	Python, Docker	Sarah D	sarahd@example.com
456	Jane Doe	Data Eng	2020-10-10	Python, Databases	Sarah D	sarahd@example.com
789	Tom Jones	Data Eng	Sickness	Python	Sarah D	sarahd@example.com

Solution: Separate columns for separate purposes

student_number	student_name	course	enroll_date	deferred_reason	modules	course_instructor	instructor_email
123	Joe Bloggs	Data Eng	2020-10-23	Null	Python, Docker	Sarah D	sarahd@example.com
456	Jane Doe	Data Eng	2020-10-10	Null	Python, Databases	Sarah D	sarahd@example.com
789	Tom Jones	Data Eng	Null	Sickness	Python	Sarah D	sarahd@example.com

Applying 1NF

- **Single Valued Attributes**: Each column in the table should hold a single value.

student_number	student_name	course	enroll_date	deferred_reason	modules	course_instructor	instructor_email
123	Joe Bloggs	Data Eng	2020-10-23	Null	Python, Docker	Sarah D	sarahd@example.com
456	Jane Doe	Data Eng	2020-10-10	Null	Python, Databases	Sarah D	sarahd@example.com
789	Tom Jones	Data Eng	Null	Sickness	Python	Sarah D	sarahd@example.com

Is the rule violated?

Violation: Module column stores multiple modules

student_number	student_name	course	enroll_date	deferred_reason	modules	course_instructor	instructor_email
123	Joe Bloggs	Data Eng	2020-10-23	Null	Python, Docker	Sarah D	sarahd@example.com
456	Jane Doe	Data Eng	2020-10-10	Null	Python, Databases	Sarah D	sarahd@example.com
789	Tom Jones	Data Eng	Null	Sickness	Python	Sarah D	sarahd@example.com

Solution: Duplicate other row data to allow single value for module

student_number	student_name	course	enroll_date	deferred_reason	module	course_instructor	instructor_email
123	Joe Bloggs	Data Eng	2020-10-23	Null	Python	Sarah D	sarahd@example.com
123	Joe Bloggs	Data Eng	2020-10-23	Null	Docker	Sarah D	sarahd@example.com
456	Jane Doe	Data Eng	2020-10-10	Null	Python	Sarah D	sarahd@example.com
456	Jane Doe	Data Eng	2020-10-10	Null	Python	Sarah D	sarahd@example.com
789	Tom Jones	Data Eng	Null	Sickness	Python	Sarah D	sarahd@example.com

Applying 1NF has increased the `redundancy` (bad!) but decreased the `complexity` and made the data more structured and easier to work with.

Rules for Second Normal Form (2NF)

For a table to be in 2NF then:

1. The table should already be in `1NF`
2. The table should have no `Partial Dependency`

(For simplicity, we will no longer considering course 'modules' in our example table)

Dependency

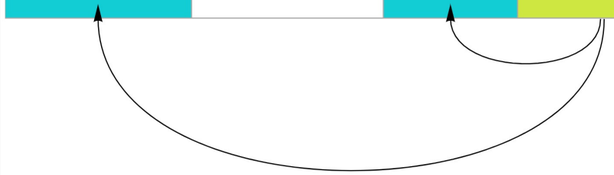
Let's understand `Dependency` before discussing `Partial Dependency`

`student_number` and `course` are the columns which allow us to uniquely identify a row. These together are the table's `composite key`

student_number	student_name	course	enroll_date	deferred_reason	course_instructor	instructor_email
123	Joe Bloggs	Data Eng	2020-10-23	Null	Sarah D	sarahd@example.com
123	Joe Bloggs	Comp Sci	2020-10-20	Null	Felix M	felixm@example.com
456	Jane Doe	Data Eng	2020-10-10	Null	Sarah D	sarahd@example.com
789	Tom Jones	Data Eng	Null	Sickness	Sarah D	sarahd@example.com

Table properties are dependent on the whole key if they need all parts of the key to be determined.

student_number	student_name	course	enroll_date	deferred_reason	course_instructor	instructor_email
123	Joe Bloggs	Data Eng	2020-10-23	Null	Sarah D	sarahd@example.com
123	Joe Bloggs	Comp Sci	2020-10-20	Null	Felix M	felixm@example.com
456	Jane Doe	Data Eng	2020-10-10	Null	Sarah D	sarahd@example.com
789	Tom Jones	Data Eng	Null	Sickness	Sarah D	sarahd@example.com



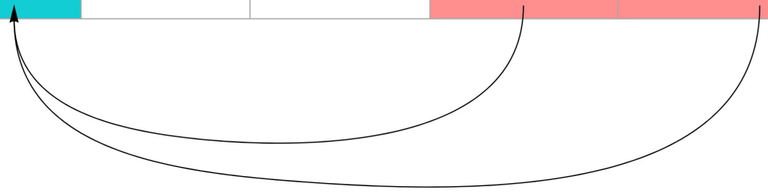
enrol_date is the date when a particular student enrolled on a particular course. It does not relate only to the course or the student, but to both

enrol_date is therefore dependent on the whole key.

Partial Dependency

Table properties are `Partially Dependent` when they relate only to part of the key.

student_number	student_name	course	enroll_date	deferred_reason	course_instructor	instructor_email
123	Joe Bloggs	Data Eng	2020-10-23	Null	Sarah D	sarahd@example.com
123	Joe Bloggs	Comp Sci	2020-10-20	Null	Felix M	felixm@example.com
456	Jane Doe	Data Eng	2020-10-10	Null	Sarah D	sarahd@example.com
789	Tom Jones	Data Eng	Null	Sickness	Sarah D	sarahd@example.com



A course always has the same instuctor. This means course_instructor and instructor_email can be determined by course only, and do not need student_number at all. These properties are `partially dependent`

Applying 2NF

Violation: Instructor information is `Partially Dependent` on `course`

Solution: Split things that depend on `course` out to a new table 👌

student_number	student_name	course_id	enroll_date	deferred_reason
123	Joe Bloggs	1	2020-10-23	Null
123	Joe Bloggs	2	2020-10-20	Null
456	Jane Doe	1	2020-10-10	Null
789	Tom Jones	1	Null	Sickness

course_id	course_name	course_instructor	instructor_email
1	Data Eng	Sarah D	sarahd@example.com
2	Comp Sci	Felix M	felixm@example.com

We will give the new course table a unique number-based `course_id` at the same time, to avoid duplicating the course name between tables.

Records in the student table can refer to the new course table using the `course_id` as a `foreign key`

Third Normal Form - 3NF

For a table to be in 3NF, then:

1. The table should already be in `2NF`
2. There should be no `Transitive Dependency`

Transitive Dependency

A non-key column in a table has `Transitive Dependency` if it depends on some other non-key column, rather than depending on the key.

Reminder: A column 'A' is dependent on another column 'B' if when the value of B changes, A should change too.

course_id	course_name	course_instructor	instructor_email
1	Data Eng	Sarah D	sarahd@example.com
2	Comp Sci	Felix M	felixm@example.com

Given `course_id` is the table key, does the new Course table have any transitive dependency?

Applying 3NF

If the instructor for a course changes to a new person, then the instructor email is also going to need to change.

course_id	course_name	course_instructor	instructor_email
1	Data Eng	Sarah D	sarahd@example.com
2	Comp Sci	Felix M	felixm@example.com
3	Networking	Jeremy W	jeremyw@example.com

This shows that email depends on `course_instructor` and not on the key `course_id`

instructor_email is therefore `Transitively Dependent`

Applying 3NF

Violation: `instructor_email` is `Transitively Dependent` on `course_instructor`

Solution: Split things that depend on `course_instructor` out to a new table 👌

Course

course_id	course_name	instructor_id
1	Data Eng	12
2	Comp Sci	56
3	Networking	99

Instructor

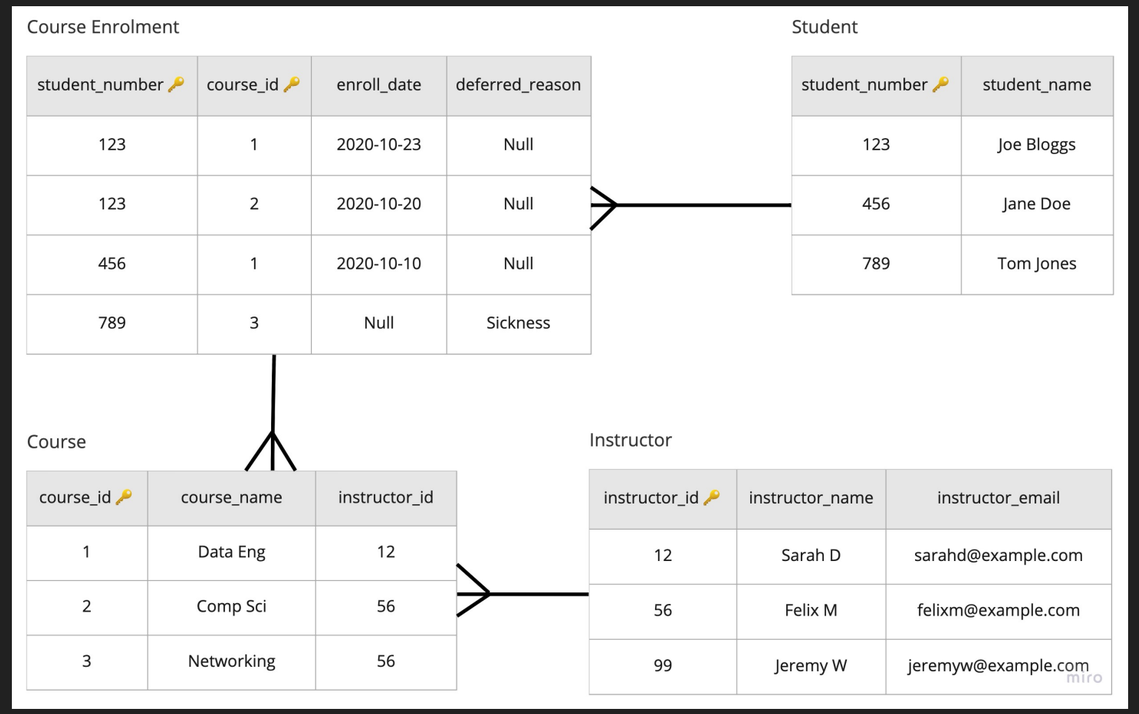
instructor_id	instructor_name	instructor_email
12	Sarah D	sarahd@example.com
56	Felix M	felixm@example.com
99	Jeremy W	jeremyw@example.com

We will also assign a new `instructor_id` numeric identifier. Records in the course table can refer to the instructor by this ID.

Final 3NF Design

Let's see what that looks like!

Final 3NF Design 👏



- Another new table has been created for `student` because `student_name` was also partially dependant on `student_number`

Final 3NF Design

What can we learn about Good Normalisation from the final design?

- Columns all store the same sort of data, with single values per field
- Related data is grouped together in a table named after the entity it belongs to (instructor, course, student, enrolment)
- Data in each table depends only on that table's key, not on any other data in the table
- When one table links to another, it's good practice to link it with IDs instead of using data that might change, such as names

Quiz Time! 🤓

Tables in First Normal Form (1NF) must not have any ___?

1. Partial dependencies
2. Multi-valued fields
3. Null values in primary key fields
4. Functional dependencies

Answer: 2

A table is in Second Normal Form (2NF) if the table is in 1NF and what other condition is met?

1. The type of data stored in a column does not change across different records
2. There are no composite primary keys
3. Columns have unique names
4. There are no partial dependencies

Answer: 4

A table is in Third Normal Form (3NF) if the table is in 2NF and what other condition is met?

1. The table primary key must be a number
2. There are no transitive dependencies
3. Order of data in the table doesn't matter
4. There are no commutative dependencies

Answer: 2

Further Normalisation

For most database normalisation requirements, stopping after achieving Third Normal Form (3NF) is enough.

It satisfies good relationship rules and will greatly improve your data structure in comparison to no normalisation. 🤩

There are further steps after third normal form that are optional. If you're interested, read about BCNF, 4NF, 5NF, ETNF and DKNF! 🤯

Pros and Cons of Normalisation

Normalisation is a great tool, but it may not always be the *right* choice.

In some cases, we may choose to intentionally not normalise depending on whether normalisation is suitable for the type of data processing.

Pros

- Querying for individual data records can be faster as we benefit from mutliple smaller tables with well-designed indexes
- Better and more logical database organisation makes it easier to build new functionality to the application
- Reduces redundant data
- Improves data consistency
- The database design itself enforces good data, and prevents inconsistencies, or orphans.

For Transactional workloads like a shopping cart or a booking system where data INTEGRITY is critical, normalisation is a great choice.

Cons

- Processing all data as a batch can be slower, due to the need to join between tables
- Table joins are needed on reading data which increases complexity of query operations
- Overly normalised data can be difficult to correctly join and put together without domain knowledge of its layout

For Analytical workloads like bulk data processing where SPEED is critical, allowing some amount of denormalisation may sometimes be preferred!

Exercise

Instructor will distribute exercise.

Learning Objectives Revisited

- Be able to explain what data normalisation is
- Understand some of the problems that normalisation helps to fix
- Practice implementing the steps to reach first, second and third normal form
- List some of the Pros and Cons of normalisation

Terms and Definitions Recap

Data Normalisation: The process of structuring a relational database in accordance with a series of so-called normal forms in order to reduce data redundancy and improve data integrity.

Normal Form: Steps used to eliminate or reduce redundancy in database tables.

Functional Dependency: A relationship between two attributes, typically between the primary key and other non-key attributes within a table.

Partial Dependency: A relationship where an attribute of a table depends only on part of that table's composite key

Transitive Dependency: A relationship where an attribute of a table depends on some other attribute which is not part of the table's key

Further Reading

- Another Normalisation Walk-through