Normalisation 1

Chapter 4.1
V4.0

Copyright @ Napier University

Normalisation

- Overview
  - discuss entity integrity and referential integrity
  - describe functional dependency
  - normalise a relation to first formal form (1NF)
  - normalise a relation to second normal form (2NF)
  - normalise a relation to third normal form (3NF)

What is normalisation?

- Transforming data from a “*problem*” into relations while ensuring data integrity and eliminating data redundancy.
  - Data integrity : data is consistent and satisfies data constraint rules
  - Data redundancy: if data can be found in two places in a single database (direct redundancy) or calculated using data from different parts of the database (indirect redundancy) then redundancy exists.
- Normalisation should remove redundancy, but not at the expense of data integrity.

Problems of redundancy

- If redundancy exists then this can cause problems during normal database operations because:
  - When data is inserted into the database, the data must be duplicated wherever redundant versions of that data exists. **Insertion error/anomaly**
  - When data is updated, all redundant data must be simultaneously updated to reflect that change.

Update error/anomaly

Normal forms

- The data in the database can be considered to be in one of a number of `normal forms'. Basically the normal form of the data indicates how much redundancy is in that data. The normal forms have a strict ordering:
  - 1st Normal Form
  - 2nd Normal Form
  - 3rd Normal Form
  - BCNF

$$1^{st} \leq 2^{nd} \leq 3^{rd} \leq \text{BCNF}$$

- There are more forms after BCNF. These are rarely utilised in system design and are not considered further here.

What does unique mean?

Is there any redundancy allowed?

Edinburgh Napier UNIVERSITY

Integrity Constraints

- An integrity constraint is a rule that restricts the values that may be present in the database.
- **entity integrity** - The rows (or tuples) in a relation represent entities, and each one must be uniquely identified. Hence we have the **primary key** that must have a unique, non-null value for each row.
- **referential integrity** - This constraint involves the foreign keys. Foreign keys tie the relations together, so it is vitally important that the links are correct. Every **foreign key** must either be null or its value must be the actual value of a key in another relation.

Understanding Data

- Sometimes the starting point for understanding a problem's data requirements is given using functional dependencies.
- A functional dependency is two lists of attributes separated by an arrow. Values given for the LHS uniquely identifiy a single set of values for the RHS attributes.
- Consider:
  R (<u>matric_no</u>, firstname, surname, tutor_no, tutor_name)
- tutor_no **->** tutor_name
- LHS (Left Hand Side) -> RHS (Right Hand Side)

- R (<u>matric_no</u>, firstname, surname, tutor_no, tutor_name)

  tutor_no **->** tutor_name

  - A given tutor_no uniquely identifies (AKA *functionally determines*) a tutor_name.
  - Tutor_name is dependent on tutor_no

  - An implied determinant (underlined– the primary key) is also present:
    - matric_no -> firstname, surname, tutor_no, tutor_name

Extracting understanding

- It is possible that the functional dependencies have to be extracted by looking at real data from the database.
- This is problematic as it is possible that the data does not contain enough information to extract all the dependencies, but it is a starting point.

Repeating group

Example

matric_no	Name	date_of_birth	subject	grade
960100	Smith, J	14/11/1977	Databases Soft_Dev ISDE	C A D
960105	White, A	10/05/1975	Soft_Dev ISDE	B B
960120	Moore, T	11/03/1970	Databases Soft_Dev Workshop	A B C
960145	Smith, J	09/01/1972	Databases	B
960150	Black, D	21/08/1973	Databases Soft_Dev ISDE Workshop	B D C D

Student(matric_no, name, date_of_birth, (subject, grade))

name, date_of_birth -> matric_no

Flatten table and extend primary key

STUDENT #2

Redundancy

matric_no	name	date_of_birth	Subject	grade
960100	Smith, J	14/11/1977	Databases	C
960100	Smith, J	14/11/1977	Soft_Dev	A
960100	Smith, J	14/11/1977	ISDE	D
960105	White, A	10/05/1975	Soft_Dev	B
960105	White, A	10/05/1975	ISDE	B
960120	Moore, T	11/03/1970	Databases	A
960120	Moore, T	11/03/1970	Soft_Dev	B
960120	Moore, T	11/03/1970	Workshop	C
960145	Smith, J	09/01/1972	Databases	B
960150	Black, D	21/08/1973	Databases	B
960150	Black, D	21/08/1973	Soft_Dev	D
960150	Black, D	21/08/1973	ISDE	C
960150	Black, D	21/08/1973	Workshop	B

Repeating Group

- The Student table with the repeating group removed (i.e., flattened) can be written as:

  Student(matric_no, name, date_of_birth, subject, grade)
- Although the repeating group was removed, this has introduced redundancy. For every matric_no/subject combination [the NEW PRIMARY KEY], the student name and date of birth is replicated. This can lead to errors.

- Sometimes you will miss spotting the repeating group. However, using the redundancy removal techniques of this lecture it does not matter if you spot these issues or not, as the end result is always a normalised set of relations.

First Normal Form

- First normal form (1NF) deals with the `shape' of the record.
- A relation is in 1NF if, and only if, it contains no repeating attributes or groups of attributes.
- Example:
  - The Student table with the repeating group is not in 1NF
  - It has repeating groups -- it is an `unnormalised table'.
- To remove repeating groups, either:
  - flatten the table and extend the key, or
  - decompose (split) the relation- leading to First Normal Form

Flattened table problems

- With the relation in its flattened form, strange anomalies appear in the system. Redundant data is the main cause of insertion, deletion, and updating anomalies.
  - Insertion anomaly – subject is now in the primary key, we cannot add a student until they have at least one subject. Remember, no part of a primary key can be NULL.
  - Update anomaly – changing the name of a student means finding all rows of the database where that student exists and changing each one separately.
  - **Deletion anomaly** - deleting all D*atabases* (subject) information also deletes student 960145.

What is the primary key of the Record relation?

Decomposing the relation

- The alternative approach is to split the table into two parts, one for the repeating groups and one of the non-repeating groups.
- the primary key from the original relation is included in both of the new relations ! !

Record

matric_no	**subject**	**grade**
960100	Databases	C
960100	Soft_Dev	A
960100	ISDE	D
960105	Soft_Dev	B
960105	ISDE	B
...	...	...
960150	Workshop	B

Student

matric_no	name	date_of_birth
960100	Smith,J	14/11/1977
960105	White,A	10/05/1975
960120	Moore,T	11/03/1970
960145	Smith,J	09/01/1972
960150	Black,D	21/08/1973

Relations

- We now have two relations, Student and Record.
  - Student contains the original non-repeating groups
  - Record has the original repeating groups and the matric_no

Student (<u>matric_no</u>, name, date_of_birth)
Record (<u>matric_no</u>, <u>subject</u>, grade)

- This version of the relations does not have insertion, deletion, or update anomalies.
- Without repeating groups, we say the relations are in First Normal Form (1NF).

Edinburgh Napier UNIVERSITY

<u>KeyPart1 + KeyPart2</u> => attribute1, attribute2, …

Second Normal Form

- A relation is in 2NF if, and only if, it is in 1NF **and every** non-key attribute is fully functionally dependent on the <u>whole key</u>.
- Thus the relation is in 1NF with no repeating groups, and all non-key attributes must depend on the whole key, not just some part of it. Another way of saying this is that there must be no partial key dependencies (PKDs).
- The problems arise when there is a compound key, e.g. the key to the Record relation - **<u>matric_no, subject</u>**. In this case it is possible for non-key attributes to depend on only part of the key - i.e. on only one of the two key attributes. This is what 2NF tries to prevent.

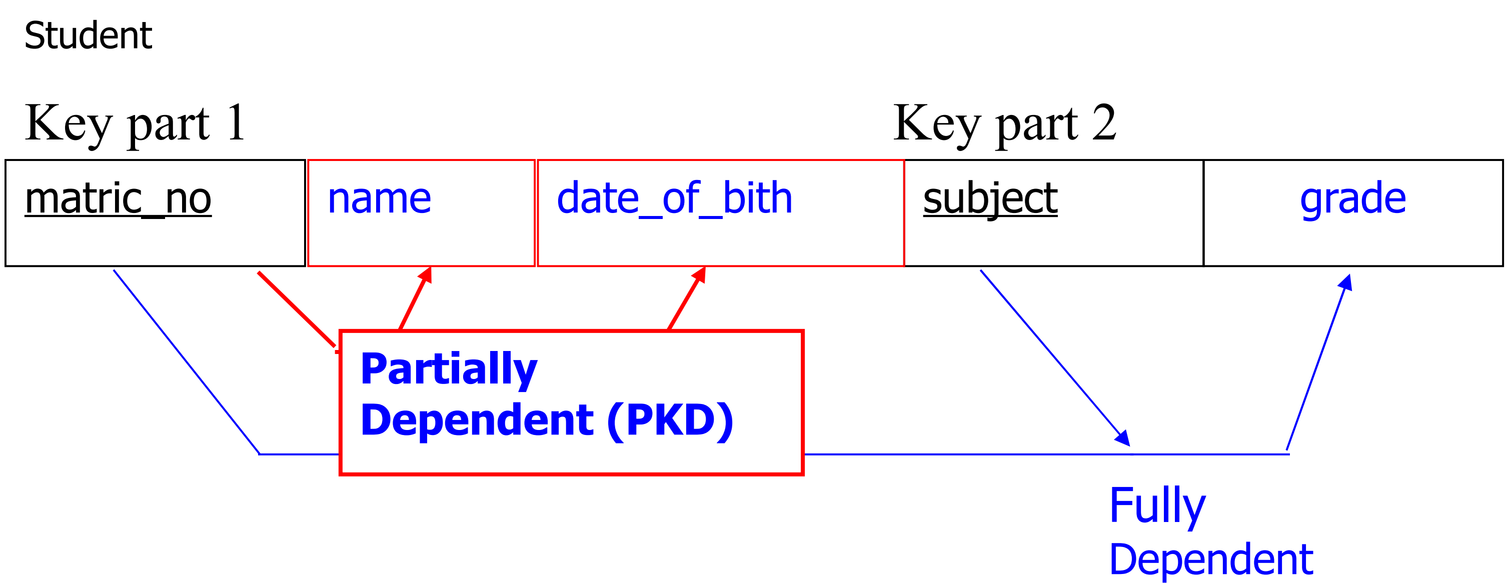
Edinburgh Napier UNIVERSITY

Example

- Consider again the Student relation from the flattened Student #2 table:
  Student(matric_no, name, date_of_birth, subject, grade)
- There are no repeating groups, so the relation is in 1NF
- However, we have a compound primary key - so we must check all of the non-key attributes against each part of the key to ensure they are functionally dependent on it.
  – matric_no determines name and date_of_birth, but not grade.
  – subject together with matric_no determines grade, but not name or date_of_birth.
- So there is a problem with potential redundancies

Dependency Diagram

- A dependency diagram is used to show how non-key attributes relate to each part or combination of parts in the primary key.



- So this relation is not in 2NF
  - It appears to be two tables squashed into one.
  - the solutions is to split the relation into component parts (to decompose it).

1. separate out all the attributes that are solely dependent on matric_no - put them in a new Student_details relation, with matric_no as the primary key
2. separate out all the attributes that are solely dependent on subject - *in this case no attributes are solely dependent on subject*.
3. separate out all the attributes that are solely dependent on matric_no + subject - put them into a separate Student relation, keyed on matric_no + subject

Student Details

matric_no	name	date_of_birth

Student

matric_no	subject	grade

All attributes in each relation are fully functionally dependent upon its primary key

These relations are now in Second Normal Form (2NF)

What is interesting is that this set of relations are the same as the ones where we realised that there was a repeating group.

Third Normal Form

- 3NF is an even stricter normal form and removes virtually all the redundant data :
- A relation is in 3NF if, and only if, it is in 2NF and there are no transitive functional dependencies

Key -> non-key attribute -> non-key attribute

transitivity

Third Normal Form

- Transitive functional dependencies arise:
  - when one non-key attribute is functionally dependent on another non-key attribute:
    - FD: non-key attribute **->** non-key attribute
  - and when there is redundancy in the database
- By definition transitive functional dependency can only occur if there is <u>more than one non-key field</u>, so we can say that a relation in 2NF with zero or one non-key field must automatically be in 3NF.

Example

Key field Non-Key fields

Project_no	Manager	Address
p1	Black,B	32 High Street
p2	Smith,J	11 New Street
p3	Black,B	32 High Street
p4	Black,B	32 High Street

Project has more than one non-key field so we must check for transitive dependencies

Extract

- Address depends on the value of manager.
- From the table we can propose:

  Project (project_no, manager, address)

  manager -> address

- In this case address is transitively dependent on manager. The primary key is project_no, yet the LHS and RHS (of the dependency) have no reference to this key, and both sides are present in the relation.

Problem

- Data redundancy arises from this situation:
  - we will duplicate **address** if a manager is in charge of more than one project
  - this causes problems if we have to change the address – it requires changing several entries, and this can lead to errors.

Fix

- Eliminate the transitive functional dependency by splitting (decomposing) the table
  - create two relations - one with the transitive dependency in it, and another for all of the remaining attributes.
  - split Project into *Project* and *Manager*.
- the determinant attribute becomes the primary key in the new relation i.e., manager becomes the primary key to the Manager relation
- the original key is the primary key to the remaining non-transitive attributes - in this case, project_no remains the key to the new Projects table.

Result : 3NF

- So now we need to store the address only once
- If we need to know a manager's address we can look it up in the Manager relation
- The manager attribute is the link between the two tables -- in the Projects table, manager is now a foreign key.
- These relations are now in third normal form.

Project

Project_no	**Manager**
p1	Black,B
p2	Smith,J
p3	Black,B
p4	Black,B

Manager

Manager	**Address**
Black,B	32 High Street
Smith,J	11 New Street

Primary Key for the whole relation

Edinburgh Napier UNIVERSITY

Summary: 1NF

- A relation is in 1NF if it contains no repeating groups
- To convert an unnormalised relation to 1NF either:
  - Flatten the table and change the primary key, or
  - DECOMPOSE the relation into smaller relations, one for the repeating groups and one for the non-repeating groups.
    - Remember to put the primary key from the original relation into both new relations.
    - This (decompose) option is liable to give the best results.

R(a,b,(c,d)) becomes

R(a,b)

R1(a,c,d)

Summary: 2NF

- A relation is in 2NF if it contains no repeating groups and no partial key functional dependencies (PKDs)
  - Rule: A relation in 1NF with a single key field must (inevitably) be in 2NF
  - To convert a relation with partial functional dependencies to 2NF, create a set of new relations (DECOMPOSE):
    - One relation for the attributes that are fully dependent upon the key.
    - One relation for each part of the key that has partially dependent attributes

    R(<u>a,b</u>,c,d) and a->c (a PKD) becomes

    R(<u>a,b</u>,d) and R1(<u>a</u>,c)

Summary: 3NF

- A relation is in 3NF if it contains no repeating groups, no partial functional dependencies, and no transitive functional dependencies
- To convert a relation with transitive functional dependencies to 3NF, remove the attributes involved in the transitive dependency and put them in a new relation (i.e., DECOMPOSE)

3NF continued

R(<u>a,b</u>,c,d)

c -> d

Becomes

R(<u>a,b</u>,c)

R1(<u>c</u>,d)

Summary: 3NF

- Rule: A relation in 2NF with only one <u>non-key</u> attribute must (inevitably) be in 3NF
- **In a normalised relation, a non-key field must provide a fact about the key, the whole key and nothing but the key.**
- Relations in 3NF are sufficient for most practical database design problems. However, 3NF does not guarantee that all anomalies have been removed.