Topic 04: Normalisation

ICT285 Databases

Dr Danny Toohey

About this topic

In this topic, we’ll look at one of the central ideas in relational database design, normalisation. Normalisation is a method of ensuring a 'good' relational database design. Put simply, this means that the data is stored as efficiently as possible, with no unnecessary duplication, and can be modified without the potential for inconsistencies (anomalies) arising. A well designed database will also support the processing required for information retrieval and decision making.

Topic Learning Outcomes

- **After completing this topic you should be able to:**

- Describe the aims of good relational database design through normalisation
- Explain the potential modification anomalies (update, insert and delete) associated with redundant information in tables
- Identify functional dependencies among attributes
- Give definitions of the following normal forms: 1NF, 2NF, 3NF, BCNF
- Be able to identify which normal form a given relation is in from examining its functional dependencies
- Normalise a given relation to a higher normal form
- Discuss some practical limitations that may be associated with normalisation

Resources for this topic

READING

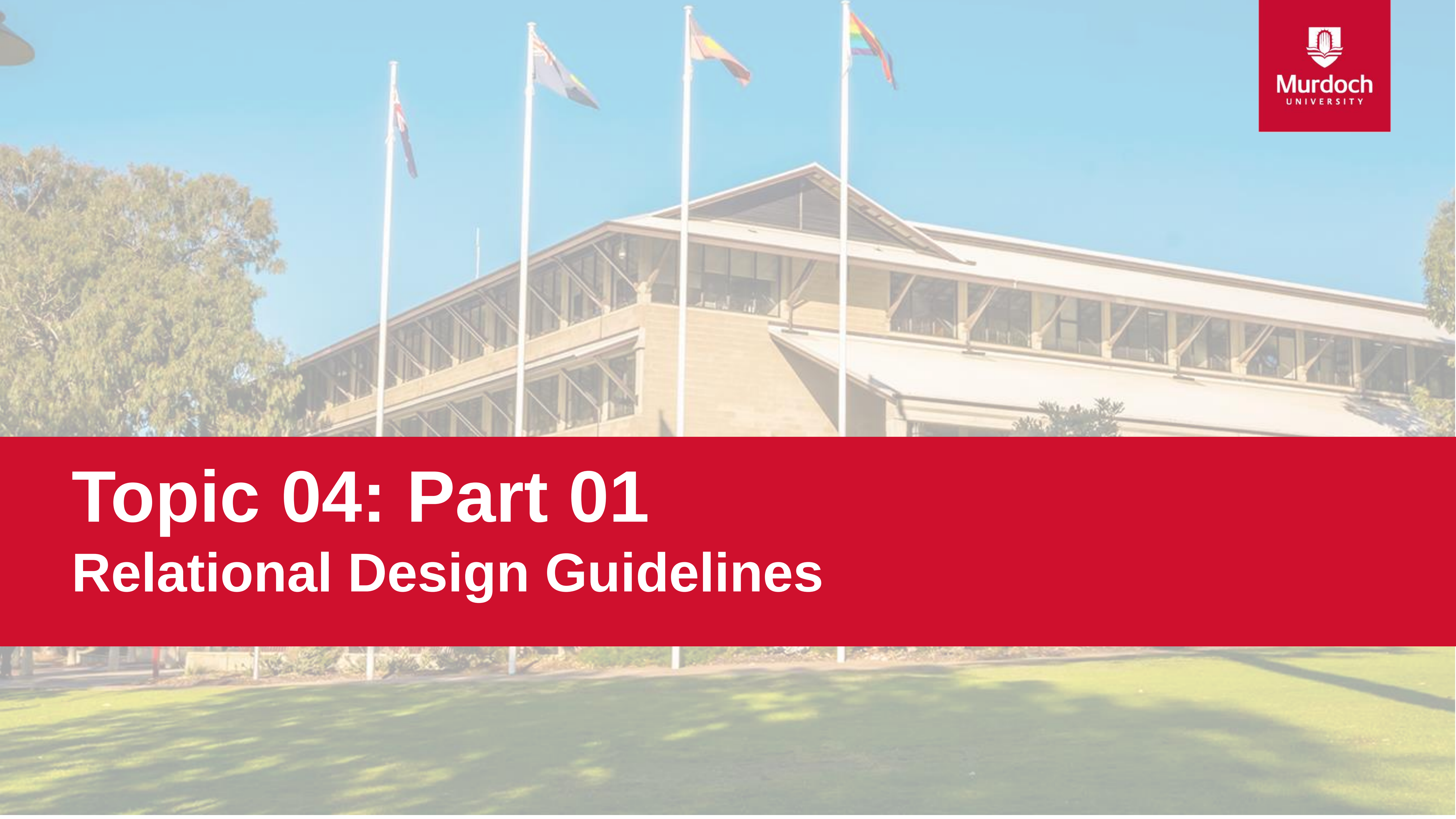
- Text, Chapter 3 “The Relational Model and Normalisation".
- Also skim through Chapter 4 “Database Design Using Normalisation

- There are many demonstrations on YouTube that you may find useful, e.g. Normalization Example, http://www.youtube.com/watch?v=uO80f642LCY

Kroenke, D.M., and Auer, D.J., 2016, Database Processing: Fundamentals, Design and Implementation, 14th Edition, Pearson, Boston.

Topic Outline

1. Relational design guidelines
2. Modification anomalies:
   - Insertion anomalies
   - Deletion anomalies
   - Update anomalies
3. Functional dependencies
4. Normal forms: 1NF, 2NF, 3NF
5. Higher normal forms: BCNF, 4NF (briefly)
6. Normalisation in practice



Topic 04: Part 01

Relational Design Guidelines

Relational database design

Is about ensuring that:

- Relations can be added, deleted and updated without damaging the consistency of the database
- The database can be queried to retrieve useful data – often by combining records from more than one table
- The database performance is optimal for its uses

This is achieved by ensuring a good database design through the process of **normalisation**

Reminder from Topic 2…

Properties of a relation

A valid relation has the following properties:

- A name that is distinct from all other relations
- Each **attribute** in a relation has a distinct name
- All cell values are **atomic** (multi-valued attributes are not allowed) – i.e., each row/column intersection represents a single data value
- Values in attributes are from the same *domain*
    - The **attribute domain** is the set of all possible values it may take. Definition covers both physical (data type) and logical (semantic) values
- All **tuples** must be **unique** – i.e., there must be an attribute or set of attributes that uniquely identifies each row
- **Attributes** are **not ordered**
- **Tuples** are **not ordered**

Reminder from Topic 2…

Example relation

StudentID	FamilyName	Degree	Major	GPA
12345678	WELLS	BSc	BIS	3.00
12456789	NORBERT	BSc	CS	2.70
23456789	KENDALL	BSc	GT	3.50

Note the following features of the **Student** relation above:

1. Each *tuple* is unique
2. Each *tuple* is about ONE student
3. Each *attribute* contains data from the same *domain*
4. Each *attribute* has a unique name
5. Each *cell* is *atomic*

Keys

- Recall that each *tuple* in a relation must be unique for it to be a valid relation
- Therefore, there must be *an attribute or set of attributes that is unique* and so can be used to identify each tuple
- This attribute or set of attributes is called a **key**
- There are several types of key…
    - Superkey
    - Candidate Key
    - Primary Key
    - Alternate Key
    - Foreign Key

Reminder from Topic 2…

Candidate Keys

A candidate key is…

- A **minimal** superkey
  - A superkey is minimal (i.e. it is a candidate key) if you can’t remove any attributes without it ceasing to be a key
- There can be more than one candidate key in a relation
- For example:
  - MOTOR VEHICLE (EngineNo, RegistrationNo, Make, Colour, Model)
  - --- EngineNo and RegistrationNo are both unique, so are both candidate keys

Reminder from Topic 2…

Primary Keys

A Primary Key is

- *The candidate key that is chosen to be the key for the relation*
- A relation can only have **one** primary key
- The value of the primary key:
    - MUST be UNIQUE
    - MUST NOT be NULL
- If a primary key is made up of > 1 attribute, it is known as a **compound, composite** or **concatenated** primary key
    - TUTORIAL (<u>TutorialDay, TutorialStartTime</u>, TutorName)

Reminder from Topic 2…

How do we find candidate keys?

- Look at the data set - if you know that it is representative
- Formally – from the *functional dependencies* among the data
  - We will look more at this in Topic 4, Normalisation
- In practice – from the meaning of the data in the real world
  - e.g. your student number is designed to be a unique identifier
  - Phone numbers and email addresses must be unique to be useful

Reminder from Topic 2…

Foreign Keys

A Foreign Key is:

- An attribute in one relation that is used to reference the primary key in another relation
- This allows us to determine which records are *related*

STUDENT

StudentNo	LastName
20123456	Wells
20987654	Kendall
20876567	Norbert

UNIT

UnitCode	UnitName
ICT285	Databases
ICT292	IS Management
ICT301	Enterprise Architectures

ENROLMENT

StudentNo	**UnitCode**
20123456	ICT285
20123456	ICT292
20876567	ICT301
20876567	ICT285
20987654	ICT285

Reminder from Topic 2…

Foreign keys

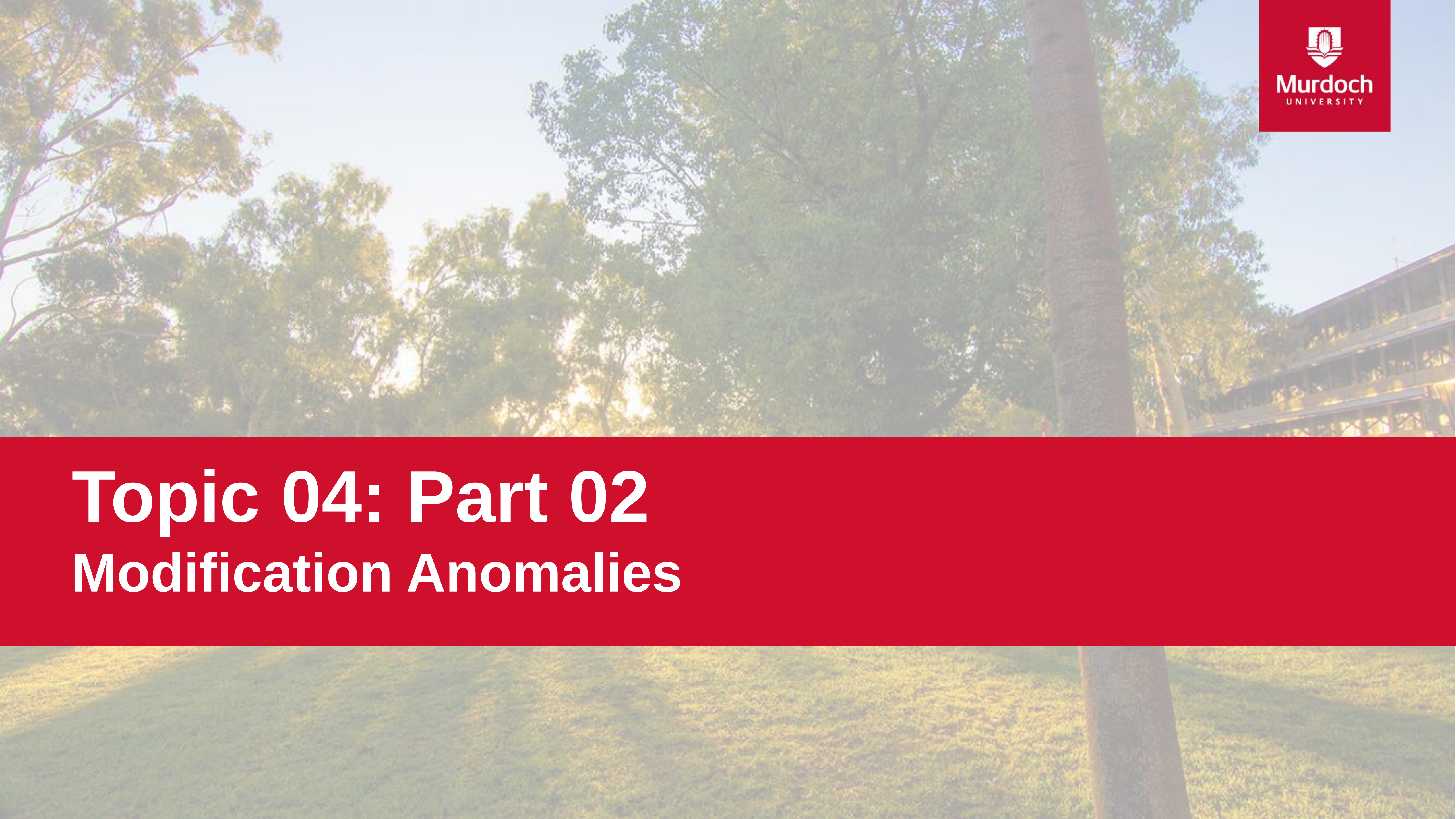
- A well designed relational database will be able to link all its tables through primary keys and foreign keys in a way that represents the meaning in the data
- This gives us great flexibility in formulating queries to retrieve combinations of information
- We’ll look at this formally when we cover **normalisation**…

Relational design guidelines

- The relations should be **easy to understand**
  - So a relation should relate to a single “real-world” concept EG: table about student
- Relations must not have any **modification anomalies**
  - Therefore, any modification anomalies must undergo normalisation to remove
- Relations should be able to be joined together (to get information from more than one relation) without generating 'extra' records or losing information from original relation
  - **‘Lossless join’** property: allows original relation to be identified from corresponding smaller relations
- The **Functional dependencies** found in the original data should still be in the design
- Relations should be in at least 3NF

The takeaways…

- The relational database model provides great flexibility in design, through having a number of tables that can be joined via primary and foreign keys to provide combinations of information
- Obtaining a good design that achieves this flexibility while maintaining correctness and consistency is done through a process called **normalisation**



Topic 04: Part 02

Modification Anomalies

Modification anomalies

When there are redundant data and any changes to the data in relation may cause Modification Anomalies to arise(i.e. bad design!)

Anomalies lead to inconsistency in the database:

- **Insertion anomalies**

  The need to enter more data than is required (and which may not be available). Because a primary key attribute may need to be entered or it breaks entity integrity rule
- **Update anomalies**

  Not all instances of data get updated so we have to make sure all instances of duplicated (same) data is updated. So think repeated attribute in column = must make sure all updated
- **Deletion anomalies**

  Certain attributes are lost because of the deletion of other attributes

Example: Insertion anomaly

- To insert information about a new *unit* in the example above, we ***also*** need to add a student and an offering of the unit (why?)

StudentID	StudentName	Address	Unit Code	Unit Title	Offering	Mark
20012001	Bruce	1 Random Cres	ICT218	Databases	S1 2010	65
20012001	Bruce	1 Random Cres	ICT231	Systems Analysis and Design	S1 2010	62
20011999	Mary	46 The Mews	ICT231	Systems Analysis and Design	S1 2010	48
20011999	Mary	46 The Mews	ICT218	Databases	S2 2009	50

- Issue can’t add new unit because some of primary key attribute missing i.e offering is missing, studentID thus break entity integrity rule.

Example: Update anomaly

- If we change Mary's address we need to change two rows – and may end up doing it inconsistently

StudentID	StudentName	Address	Unit Code	Unit Title	Offering	Mark
20012001	Bruce	1 Random Cres	ICT218	Databases	S1 2010	65
20012001	Bruce	1 Random Cres	ICT231	Systems Analysis and Design	S1 2010	62
20011999	Mary	46 The Mews	ICT231	Systems Analysis and Design	S1 2010	48
20011999	Mary	46 The Mews	ICT218	Databases	S2 2009	50

Example: Deletion anomaly

- If we delete Bruce's enrolment in Databases in Semester 1, 2010, we **lose** all information regarding that offering – but we only wanted to delete Bruce

StudentID	StudentName	Address	Unit Code	Unit Title	Offering	Mark
20012001	Bruce	1 Random Cres	ICT218	Databases	S1 2010	65
20012001	Bruce	1 Random Cres	ICT231	Systems Analysis and Design	S1 2010	62
20011999	Mary	46 The Mews	ICT231	Systems Analysis and Design	S1 2010	48
20011999	Mary	46 The Mews	ICT218	Databases	S2 2009	50

What anomalies could occur in this relation?

StaffNum	Name	Position	Salary	SchoolNum	School	School Phone
S21	Barry Brightwater	School Dean	30000	MH47	Happiness	x005
S37	Deloris Delight	Lecturer	12000	LH53	Hopeless Causes	x003
S14	Carrie Cumbersome	Professor	18000	LH53	Hopeless Causes	x003
S09	Albert Algorithm	School Dean	9000	XS67	Low Self Esteem	x007
S05	Edwina Elocution	School Dean	24000	LH53	Hopeless Causes	x003
S41	Frank Forensic	Cleaner	9000	MH47	Happiness	x005

Are those problems resolved?

StaffNum	Name	Position	Salary	SchoolNum
S21	Barry Brightwater	School Dean	30000	MH47
S37	Deloris Delight	Lecturer	12000	LH53
S14	Carrie Cumbersome	Professor	18000	LH53
S09	Albert Algorithm	School Dean	9000	XS67
S05	Edwina Elocution	School Dean	24000	LH53
S41	Frank Forensic	Cleaner	9000	MH47

SchoolNum	School	School Phone
MH47	Happiness	x005
LH53	Hopeless Causes	x003
XS67	Low Self Esteem	x007

StaffNum	Name	Position	Salary	SchoolNum
S21	Barry Brightwater	School Dean	30000	MH47
S37	Deloris Delight	Lecturer	12000	LH53
S14	Carrie Cumbersome	Professor	18000	LH53
S09	Albert Algorithm	School Dean	9000	XS67
S05	Edwina Elocution	School Dean	24000	LH53
S41	Frank Forensic	Cleaner	9000	MH47

SchoolNum	School	School Phone
MH47	Happiness	x005
LH53	Hopeless Causes	x003
XS67	Low Self Esteem	x007

Can we still get the original relation back by joining the two new tables?

This is called a ***lossless join***

StaffNum	Name	Position	Salary	SchoolNum	School	School Phone
S21	Barry Brightwater	School Dean	30000	MH47	Happiness	x005
S37	Deloris Delight	Lecturer	12000	LH53	Hopeless Causes	x003
S14	Carrie Cumbersome	Professor	18000	LH53	Hopeless Causes	x003
S09	Albert Algorithm	School Dean	9000	XS67	Low Self Esteem	x007
S05	Edwina Elocution	School Dean	24000	LH53	Hopeless Causes	x003
S41	Frank Forensic	Cleaner	9000	MH47	Happiness	x005

The takeaways…

- The aim of relational database design is to eliminate potential modification anomalies caused by redundantly stored data
- Modification anomalies lead to *inconsistencies* in the database
- There are three types of modification anomalies:
  - **Insertion anomalies**
    - The need to enter more data than is required
  - **Update anomalies**
    - Not all instances of data get updated
  - **Deletion anomalies**
    - The *unintentional* deletion of data

Topic 04: Part 03

Functional Dependencies

Functional dependencies

- A **functional dependency** is a relationship between attributes such that the value of one attribute can be found from the value of another

  StudentNo → Student Name

- FDs represent information about the system we are modelling – how the attributes relate to one another in the real world
- Normal forms 1NF, 2NF, 3NF and BCNF are based on functional dependencies between the attributes in a relation (higher normal forms are based on other dependencies)

Importance of functional dependencies

- When we normalise a design to a ‘better’ one, the design must preserve the functional dependencies found in the original relation
- This is called **dependency preservation**
- As functional dependencies provide the constraints on what the data means, dependency preservation ensures the data remains correct and consistent

Functional dependencies

We can say that ***X determines Y*** if:

- There exists ***at most one*** value of *Y* for every value of *X*
- We write this as:

$$X \rightarrow Y$$

For example: we can say that student number *determines* date of birth, as for each student number there is one and only one date of birth

$$\text{StudentNo} \rightarrow \text{DateOfBirth}$$

(Can we say that *date of birth* determines *student number*?)

FDs - terminology

Consider the following relation:

EMPLOYEE (<u>EmployeeID</u>, Name, Address, DeptNo)

If the value of EmployeeID is known, the values of Name, Address and DeptNo can be found - in other words:

- EmployeeID **functionally determines** Name, Address, DeptNo
- Name, Address, DeptNo are **functionally dependent** on EmployeeID
- EmployeeID is called the **determinant** of the FD

Writing FDs

We write FDs as:

EmployeeID → Name, Address, DeptNo

Can also write as:

EmployeeID → Name

EmployeeID → Address

EmployeeID → DeptNo

(StudentNo, Unit, Semester, Year) → Grade

Note these four attributes TOGETHER determine Grade

Some rules about FDs

- Reflexivity
  - If $Y \subseteq X$, *then* $X \rightarrow Y$
- Augmentation
  - If $X \rightarrow Y$, *then* $XZ \rightarrow YZ$
- **Transitivity**
  - If $X \rightarrow Y$, and $Y \rightarrow Z$, *then* $X \rightarrow Z$
- Decomposition
  - If $X \rightarrow YZ$, *then* $X \rightarrow Y$ and $X \rightarrow Z$
- Union
  - If $X \rightarrow Y$, and $X \rightarrow Z$, *then* $X \rightarrow YZ$

Example:

What functional dependencies can you identify in the following relation (assuming the data is representative)?

List only the *direct* dependencies

StudentID	Name	Course	Semester	Year	Grade	UnitCode	Title
S1	Kyle	IT	1	2020	HD	ICT231	Systems Analysis
S1	Kyle	IT	1	2020	D	ICT208	BI T&T
S2	Helen	IT	1	2019	D	ICT231	Systems Analysis
S2	Helen	IT	1	2019	C	ICT218	Databases
S2	Helen	IT	2	2018	N	ICT218	Databases

Example:

What functional dependencies can you identify in the following relation (assuming the data is representative)?

List only the *direct* dependencies

StaffNum	Name	Position	Salary	SchoolNum	School	School Phone
S21	Barry Brightwater	School Dean	30000	MH47	Happiness	x005
S37	Deloris Delight	Lecturer	12000	LH53	Hopeless Causes	x003
S14	Carrie Cumbersome	Professor	18000	LH53	Hopeless Causes	x003
S09	Albert Algorithm	School Dean	9000	XS67	Low Self Esteem	x007
S05	Edwina Elocution	School Dean	24000	LH53	Hopeless Causes	x003
S41	Frank Forensic	Cleaner	9000	MH47	Happiness	x005

FDs and candidate keys

A candidate key is simply a more restrictive type of FD - a FD in which the *value of the determinant is unique*

e.g. In the relation:

EMP_DEPT (EmpID, Name, Address, DeptNo, DeptName)

we have the direct FDs:

EmpID → Name, Address, DeptNo

DeptNo → DeptName

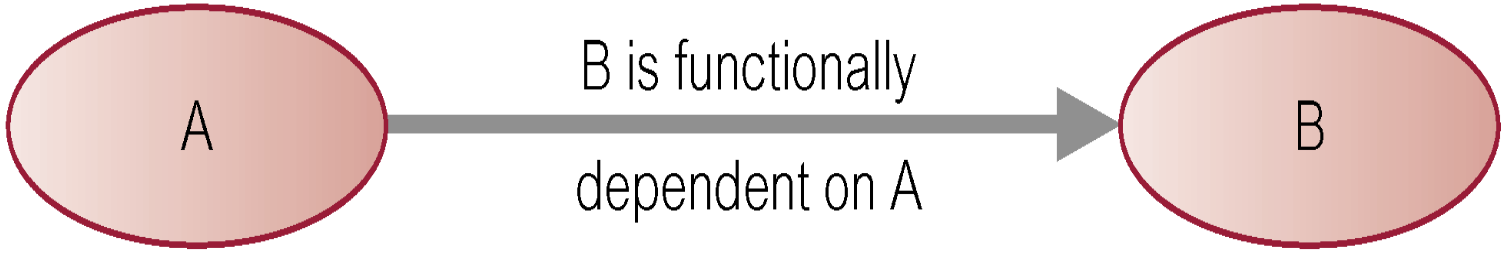
In any EMP_DEPT table the same DeptNo may occur many times, but the value for EmpID (the **candidate key**) is unique

- So this gives us a way of finding the candidate key(s) of a relation from the functional dependencies in it

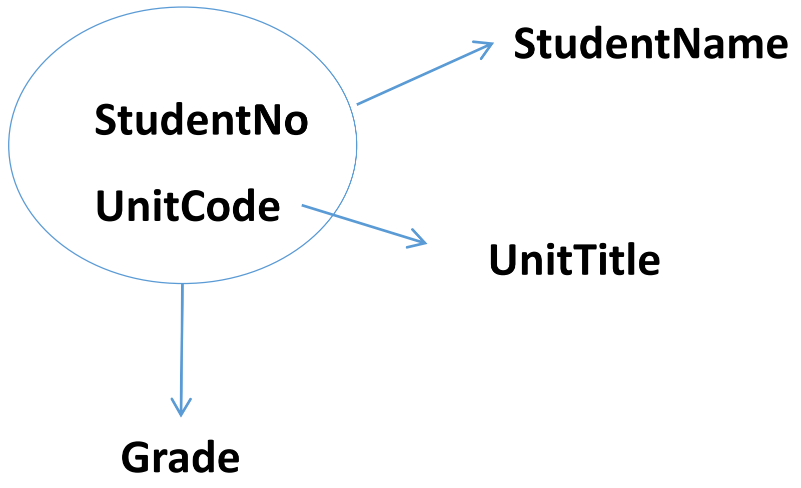
Deriving candidate keys from FDs

If we do not know the candidate key(s) of a relation, it can be found by examining the FDs that occur in the relation:

- identify the **minimum combination of attributes** from which all others can be found
- can use a **dependency diagram** as a visual aid



Dependency diagrams

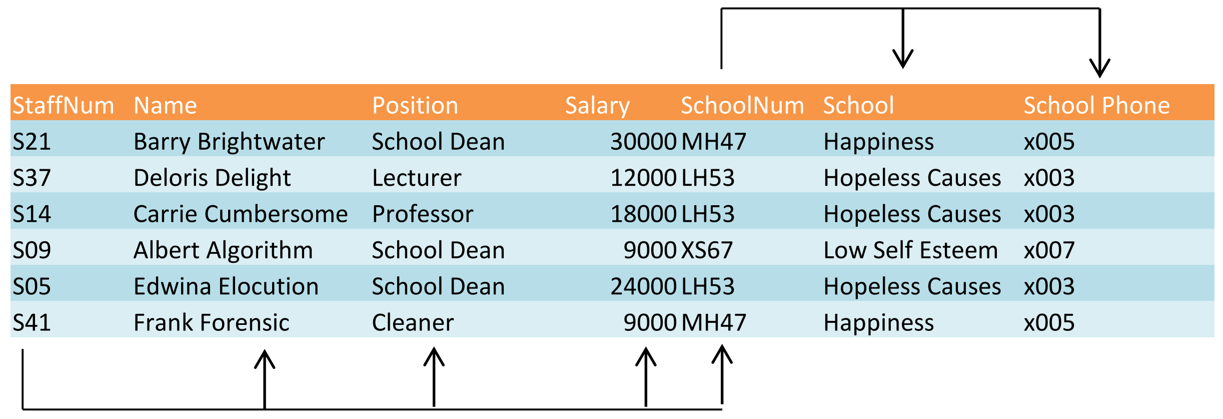


StudentNo → StudentName

UnitCode → UnitTitle

StudentNo, UnitCode → Grade

SchoolNum→School, Phone



StaffNum	Name	Position	Salary	SchoolNum	School	School Phone
S21	Barry Brightwater	School Dean	30000	MH47	Happiness	x005
S37	Deloris Delight	Lecturer	12000	LH53	Hopeless Causes	x003
S14	Carrie Cumbersome	Professor	18000	LH53	Hopeless Causes	x003
S09	Albert Algorithm	School Dean	9000	XS67	Low Self Esteem	x007
S05	Edwina Elocution	School Dean	24000	LH53	Hopeless Causes	x003
S41	Frank Forensic	Cleaner	9000	MH47	Happiness	x005

StaffNum → Name, Position, Salary, SchoolNum

Example:

What is the **candidate key** of this relation?

SchoolNum → School, Phone

StaffNum	Name	Position	Salary	SchoolNum	School	School Phone
S21	Barry Brightwater	School Dean	30000	MH47	Happiness	x005
S37	Deloris Delight	Lecturer	12000	LH53	Hopeless Causes	x003
S14	Carrie Cumbersome	Professor	18000	LH53	Hopeless Causes	x003
S09	Albert Algorithm	School Dean	9000	XS67	Low Self Esteem	x007
S05	Edwina Elocution	School Dean	24000	LH53	Hopeless Causes	x003
S41	Frank Forensic	Cleaner	9000	MH47	Happiness	x005

StaffNum → Name, Position, Salary, SchoolNum

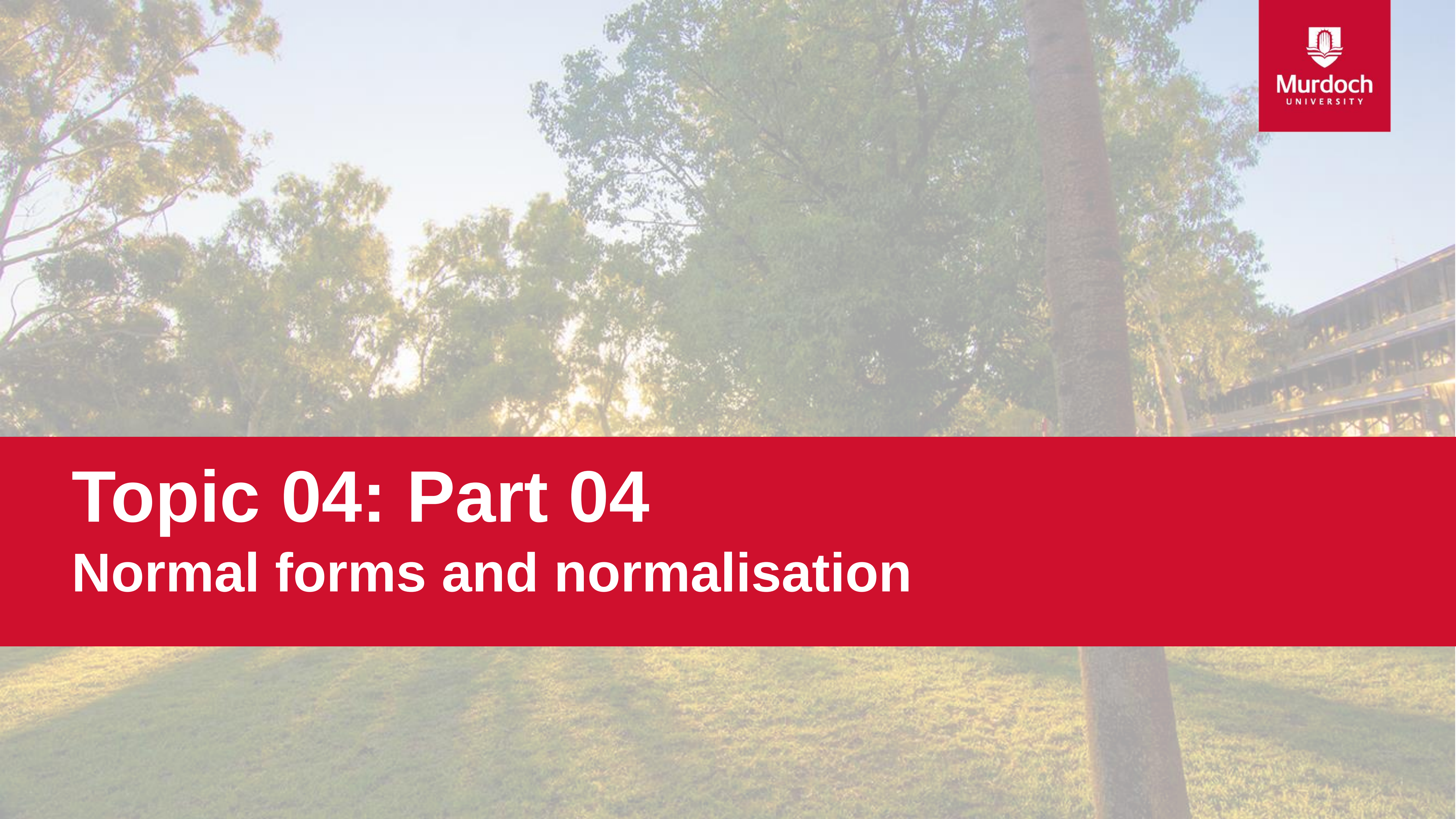
The takeaways…

- Functional dependencies show the *relationships between attributes* in a relation
- In a FD X → Y, we say that:

  X functionally determines Y

  X is the determinant of the FD

  Y is functionally dependent on X
- We can use the FDs in a relation to derive its candidate key(s), by finding the minimum set of attribute(s) that determines all the others



Topic 04: Part 04

Normal forms and normalisation

Normal Forms

- A ‘normal form’ is a particular set of conditions that are true of a given relation
- Normal forms were first defined as part of the relational model theory by Codd, to address modification anomalies that could occur in certain designs
- Although the theory on normal forms covers others (such as 5NF and DKNF), for practical purposes we are most interested in 1NF, 2NF, 3NF and BCNF
- The process of converting a relation to a set of relations in a higher NF is called **normalisation**

Normalisation process

- Normalisation is a way of transforming an unsatisfactory relation schema (one with modification anomalies) into a 'better' design
- Normalisation makes use of **functional dependencies** to:
    1. Identify the *candidate key(s)* of the relation,
    2. and then to examine *how the other attributes relate to the candidate key*
    3. If the design is less than optimal, **normalise** to a higher normal form

1NF -- BCNF

Each NF becomes progressively more restrictive, which means that the definition of any NF includes the definition of the preceding NF:

- **1NF**: All valid relations are in 1NF
- **2NF**: 1NF PLUS no partial functional dependencies
- **3NF**: 2NF PLUS no transitive functional dependencies
- **BCNF**: 3NF PLUS all determinants are candidate keys

First Normal Form, 1NF

First Normal Form (1NF)

For a relation to be in 1NF, it needs to be a valid relation:

- No repeating groups or nesting in relations

This table is **NOT** in 1NF (i.e., it is **unnormalised**) – it is not a valid relation

StudentID	Name	Course	Semester	Year	Grade	UnitCode	Title
S1	Kyle	IT	1	2020	HD	ICT231	Systems Analysis
			1	2020	D	ICT208	BI T&T
S2	Helen	IT	1	2020	D	ICT231	Systems Analysis
			1	2020	C	ICT218	Databases
			2	2019	N	ICT218	Databases

Solution: complete the table

- Repeat StudentID, Name and Course for each tuple:

StudentID	Name	Course	Semester	Year	Grade	UnitCode	Title
S1	Kyle	IT	1	2020	HD	ICT231	Systems Analysis
			1	2020	D	ICT208	BI T&T
S2	Helen	IT	1	2020	D	ICT231	Systems Analysis
			1	2020	C	ICT218	Databases
			2	2019	N	ICT218	Databases

The relation is now in at least First Normal Form, **1NF**

(Are there still potential modification anomalies in this design?)

StudentID	Name	Course	Semester	Year	Grade	UnitCode	Title
S1	Kyle	IT	1	2020	HD	ICT231	Systems Analysis
S1	Kyle	IT	1	2020	D	ICT208	BI T&T
S2	Helen	IT	1	2020	D	ICT231	Systems Analysis
S2	Helen	IT	1	2020	C	ICT218	Databases
S2	Helen	IT	2	2019	N	ICT218	Databases

Partial Functional Dependencies

In the **1NF** relation below the FDs are:

StudentID → Name, Course
UnitCode → Title
StudentID, UnitCode, Year, Semester → Grade

StudentID	Name	Course	Semester	Year	Grade	UnitCode	Title
S1	Kyle	IT	1	2020	HD	ICT231	Systems Analysis
S1	Kyle	IT	1	2020	D	ICT208	BI T&T
S2	Helen	IT	1	2020	D	ICT231	Systems Analysis
S2	Helen	IT	1	2020	C	ICT218	Databases
S2	Helen	IT	2	2019	N	ICT218	Databases

So the candidate key is <u>StudentID, UnitCode, Year, Semester</u>

- But Name, Course depend only on StudentID – *partial* FDs
- Similarly Title depends only on UnitCode

Modification anomalies in relations with partial FDs

1NF still has the potential for modification anomalies:

- If we were to change the title of ICT208, then we would have to *update* the title for every offering of the unit
- If we were to *delete* the only offering of unit ICT208, then we would lose all details of that unit
- Could we *insert* a new unit into this table?

StudentID	Name	Course	Semester	Year	Grade	UnitCode	Title
S1	Kyle	IT	1	2020	HD	ICT231	Systems Analysis
S1	Kyle	IT	1	2020	D	ICT208	BI T&T
S2	Helen	IT	1	2020	D	ICT231	Systems Analysis
S2	Helen	IT	1	2020	C	ICT218	Databases
S2	Helen	IT	2	2019	N	ICT218	Databases

First Normal Form, 1NF

- A relation is in 1NF if it is a valid relation – this is part of the definition
- It is in 1NF but no higher if it contains *partial* FDs

There are potential modification anomalies in 1NF relations

Second Normal Form, 2NF

Second Normal Form (2NF)

A relation is in 2NF if:

- It is in 1NF, AND
- There are **no partial functional dependencies**

- A partial FD exists **where a non-key attribute is determined by only part of the (compound) candidate key**
- 2NF *removes* any partial FDs

Partial Functional Dependencies

The relation below is in **1NF but not 2NF** because it contains partial FDs

StudentID	Name	Course	Semester	Year	Grade	UnitCode	Title
S1	Kyle	IT	1	2020	HD	ICT231	Systems Analysis
S1	Kyle	IT	1	2020	D	ICT208	BI T&T
S2	Helen	IT	1	2020	D	ICT231	Systems Analysis
S2	Helen	IT	1	2020	C	ICT218	Databases
S2	Helen	IT	2	2019	N	ICT218	Databases

StudentID → Name, Course
UnitCode → Title
StudentID, UnitCode, Year, Semester → Grade

Candidate key is <u>StudentID, UnitCode, Year, Semester</u>

But Name, Course depends only on StudentID – a *partial* FD
Similarly Title depends only on UnitCode

Solution: remove partial FDs

The solution is to remove the partial FDs from the relation, by making each of them a new relation consisting of the determinant as primary key and the attributes it determines directly:

RELATION1 (StudentID, Name, Course)

RELATION2 (UnitCode, Title)

We also make a third relation with the remaining FD:

StudentID, UnitCode, Semester, Year $\rightarrow$ Grade

RELATION3 (**StudentID, UnitCode**, Semester, Year , Grade)

Each of these relations is now in at least **2NF** *(as it turns out, they are also in 3NF)*

Second Normal Form, 2NF

- A relation is in 2NF when all non-key attributes are **fully** (non-partially) dependent on the candidate key

However, there may still be potential anomalies in 2NF relations due to ***transitive FDs*** (next)

Transitive FDs

The candidate key of this relation is <u>StaffNum</u>

The relation is in at least **2NF**, as there are no partial FDs

SchoolNum→School, SchoolPhone

StaffNum	Name	Position	Salary	SchoolNum	School	School Phone
S21	Barry Brightwater	School Dean	30000	MH47	Happiness	x005
S37	Deloris Delight	Lecturer	12000	LH53	Hopeless Causes	x003
S14	Carrie Cumbersome	Professor	18000	LH53	Hopeless Causes	x003
S09	Albert Algorithm	School Dean	9000	XS67	Low Self Esteem	x007
S05	Edwina Elocution	School Dean	24000	LH53	Hopeless Causes	x003
S41	Frank Forensic	Cleaner	9000	MH47	Happiness	x005

StaffNum → Name, Position, Salary, SchoolNum

BUT there are **transitive FDs** as School, SchoolPhone depend on SchoolNum – not directly on the candidate key <u>StaffNum</u>

Can anomalies still arise??

Modification anomalies in relations with transitive FDs

2NF still has the potential for modification anomalies:

- If we were to change the title of School LH53, then we would have to *update* the title for every staff member in it
- If we were to *delete* staff member 509 then we would lose all details of the School of Low Self Esteem
- Could we *insert* a new School into this table?

StaffNum	Name	Position	Salary	SchoolNum	School	School Phone
S21	Barry Brightwater	School Dean	30000	MH47	Happiness	x005
S37	Deloris Delight	Lecturer	12000	LH53	Hopeless Causes	x003
S14	Carrie Cumbersome	Professor	18000	LH53	Hopeless Causes	x003
S09	Albert Algorithm	School Dean	9000	XS67	Low Self Esteem	x007
S05	Edwina Elocution	School Dean	24000	LH53	Hopeless Causes	x003
S41	Frank Forensic	Cleaner	9000	MH47	Happiness	x005

Third Normal Form, 3NF

Third Normal Form (3NF)

A relation is in 3NF if:

- It is in 2NF, and
- there are **no transitive functional dependencies**

- A transitive FD exists **where a non-key attribute is determined by another non-key attribute**
- 3NF *removes* any transitive FDs

Transitive FDs

- The candidate key of this relation is <u>StaffNum</u>
- The relation is in **2NF but not 3NF** because of the transitive FDs:

SchoolNum→School, SchoolPhone

StaffNum	Name	Position	Salary	SchoolNum	School	School Phone
S21	Barry Brightwater	School Dean	30000	MH47	Happiness	x005
S37	Deloris Delight	Lecturer	12000	LH53	Hopeless Causes	x003
S14	Carrie Cumbersome	Professor	18000	LH53	Hopeless Causes	x003
S09	Albert Algorithm	School Dean	9000	XS67	Low Self Esteem	x007
S05	Edwina Elocution	School Dean	24000	LH53	Hopeless Causes	x003
S41	Frank Forensic	Cleaner	9000	MH47	Happiness	x005

StaffNum → Name, Position, Salary, SchoolNum

Solution?

Solution: Remove transitive FDs

The solution is to remove the transitive FDs from the relation, by making each of them a new relation consisting of the determinant as primary key and the attributes it determines directly:

RELATION1 (<u>StaffNum</u>, Name, Position, Salary, SchoolNum**)**
RELATION2 (<u>SchoolNum</u>, School, Phone)

- Each of these relations is now in **3NF**
- And note that we can join the new relations on *primary key* and foreign key

Are there any potential modification anomalies in these relations?

Third Normal Form, 3NF

- A relation is in 3NF when all non-key attributes are **fully and non-transitively** dependent on the candidate key

- A relation in 3NF should have no modification anomalies

1NF – 3NF

1NF:	All valid relations are in 1NF by definition *(but may still have partial FDs)*
2NF:	1NF PLUS no partial functional dependencies *(but may still have transitive FDs)*
3NF:	2NF PLUS no transitive functional dependencies *(no partial or transitive FDs)*

Criteria for a good design

Recall that we also want the set of relations we produce to be a good design *as a whole*

For this we need:

1. Each relation in at least 3NF
2. And the set of relations preserves all the FDs in the original relation(s) (dependency preserving)
3. And the set of relations has the lossless join property

Once you have normalised to a set of 3NF relations, check for the dependency preserving and lossless join properties too

How to normalise to 3NF – simply

- Write down all direct FDs for the relation (omit any that can be derived from other FDs)
- Create a relation for each different determinant and the attributes it determines, with the determinant as key
- If none of the relations contains a key of the original relation, create another relation that consists only of the key (this ensures relations can be joined successfully)

The takeaways…

Creating a good, 3NF design:

- If the table isn't even in First Normal Form (1NF), put it in into 1NF
- Look at the sample data, or other information you have been given, and list the functional dependencies in the data
- From the functional dependencies, find the candidate key(s)
- Examine the FDs in terms of the candidate key(s). Are any FDs partial or transitive?
- Remove partial and/or transitive FDs to convert to a set of relations each in 3NF
- Check that your set of relations has the dependency preserving, lossless join properties

Examples

1NF – 3NF

Example 1

Given the following relation and functional dependencies:

1. What is (are) the candidate key(s)?
2. Which NF is the relation in?
3. Convert the relation into a relation or set of relations in at least 3NF

STUDENT (StudentNo, Name, PrimaryMajor, School)

FDs: StudentNo → Name, PrimaryMajor

PrimaryMajor → School

Example 2

Given the following relation and functional dependencies:

1. What is (are) the candidate key(s)?
2. Which NF is the relation in?
3. Convert the relation into a relation or set of relations in at least 3NF

ENROLS (StudentNo, Name, UnitCode, UnitName, Grade)

FDs: StudentNo → Name

UnitCode → UnitName

StudentNo, UnitCode → Grade

Example 3

Given the following relation and functional dependencies:

1. What is (are) the candidate key(s)?
2. Which NF is the relation in?
3. Convert the relation into a relation or set of relations in at least 3NF

COORDINATES (UnitCode, Semester, Option, Coordinator)

FD: UnitCode, Semester, Option → Coordinator

Example 4

Given the following relation and functional dependencies:

1. What is (are) the candidate key(s)?
2. Which NF is the relation in?
3. Convert the relation into a relation or set of relations in at least 3NF

SHOP (ShopNo, ItemNo, Date, QtySold)

Each shop sells a number of items on each day

Shop No	Item No	Date	Qty Sold
S1	M2	3/10	4
S1	M3	3/10	4
S2	M2	3/10	2
S1	M2	4/10	7

Example 5

Given the following relation and functional dependencies:

1. What is (are) the candidate key(s)?
2. Which NF is the relation in?
3. Convert the relation into a relation or set of relations in at least 3NF

APPLICATION (ApplicNo, Customer, CustAddress, DateApproved)

- Each loan application is by one customer but each customer may make many applications

ApplicNo	Customer	Address	Date Appr
X97	JoeBlog	Perth	2/3
X99	Vicki	Sydney	3/3
Y72	JoeBlog	Perth	3/3

Example 6

Part No	Description	Vendor	Address	UnitCost
1234	Logic Chip	Fast Chips	Perth	10.00
		Smart Chips	Sydney	5.00
5678	Memory chip	Fast Chips	Perth	3.00
		Quality Chips	Sydney	2.00
		Smart Chips	Sydney	5.00

a) Convert this table to a single relation (called PART-SUPPLIER) in first normal form. Illustrate the relation with the same sample data.

b) List the functional dependencies in PART-SUPPLIER and identify the candidate key(s).

c) Convert PART-SUPPLIER to a set of relations in at least third normal form.

Topic 04: Part 05

Higher Normal Forms

Higher Normal Forms

- 1NF, 2NF, and 3NF are based on functional dependencies and the relationship of *non-key attributes to the candidate key*
    - As well as these there are higher normal forms that are based on other things – we will look briefly at BCNF and 4NF

Boyce-Codd Normal Form, BCNF

Boyce-Codd Normal Form, BCNF

Anomalies can still arise in relations in 3NF

Consider the following relation:

STUDENT_ADVISOR (StudentID, Advisor, Major)

Where:

- a student can have many majors, and has a particular advisor for each
- a staff member will only advise for one major (although a major can have more than one staff member as advisor)

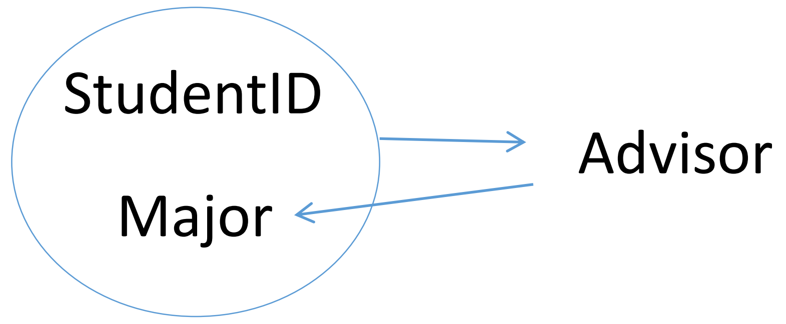
The functional dependencies are:

- StudentID, Major → Advisor
- Advisor → Major

StudentID	Advisor	Major
654	Bill	BIS
655	Bob	GT
656	Hui Min	CS
656	Mary	GT
657	Shabnam	CFISM
658	Bill	BIS

BCNF

There are two possible candidate keys:



StudentID, Major

StudentID, Advisor

And we still have anomalies – e.g. can’t add new Major or Advisor unless there is a Student enrolled

StudentID	Advisor	Major
654	Bill	BIS
655	Bob	GT
656	Hui Min	CS
656	Mary	GT
657	Shabnam	CFISM
658	Bill	BIS

BCNF

However, the relation is in 3NF as there are *no non-key attributes partially or transitively dependent on a candidate key* – as there are no non-key attributes

However there is a *key* attribute (Major) which depends on only *part of a key* (Advisor)

SOLUTION:

- STUDENT (StudentID, Advisor)
- MAJOR_ADVISOR (Advisor, Major)

We can now update the two relations independently with no modification anomalies - each is in BCNF

BCNF

BCNF is a ‘stricter’ version of 3NF that accounts for anomalies caused by overlapping candidate keys

- Simply, a relation **is in BCNF** if:
  - **It is in 3NF, and**
  - **Every determinant is a candidate key**

Note that most relations that are in 3NF are also in BCNF, so you are unlikely to have to deal with this in practice

Fourth normal form, 4NF

Fourth normal form, 4NF

- 4NF is based on multi-valued dependencies – where the value of one attribute may determine *several* values of another

  StudentNo ->> UnitCode

  StudentNo ->> Sport

  (a student is enrolled in many units, and plays many sports)

- This relation:

  ACTIVITIES (<u>StudentNo, UnitCode, Sport</u>)

  would result in modification anomalies – and would have to store EVERY combination so as not to imply that the sport and unit code were associated with each other

Fourth normal form, 4NF

- Solution: put the MVD in a relation of its own, with a compound PK:

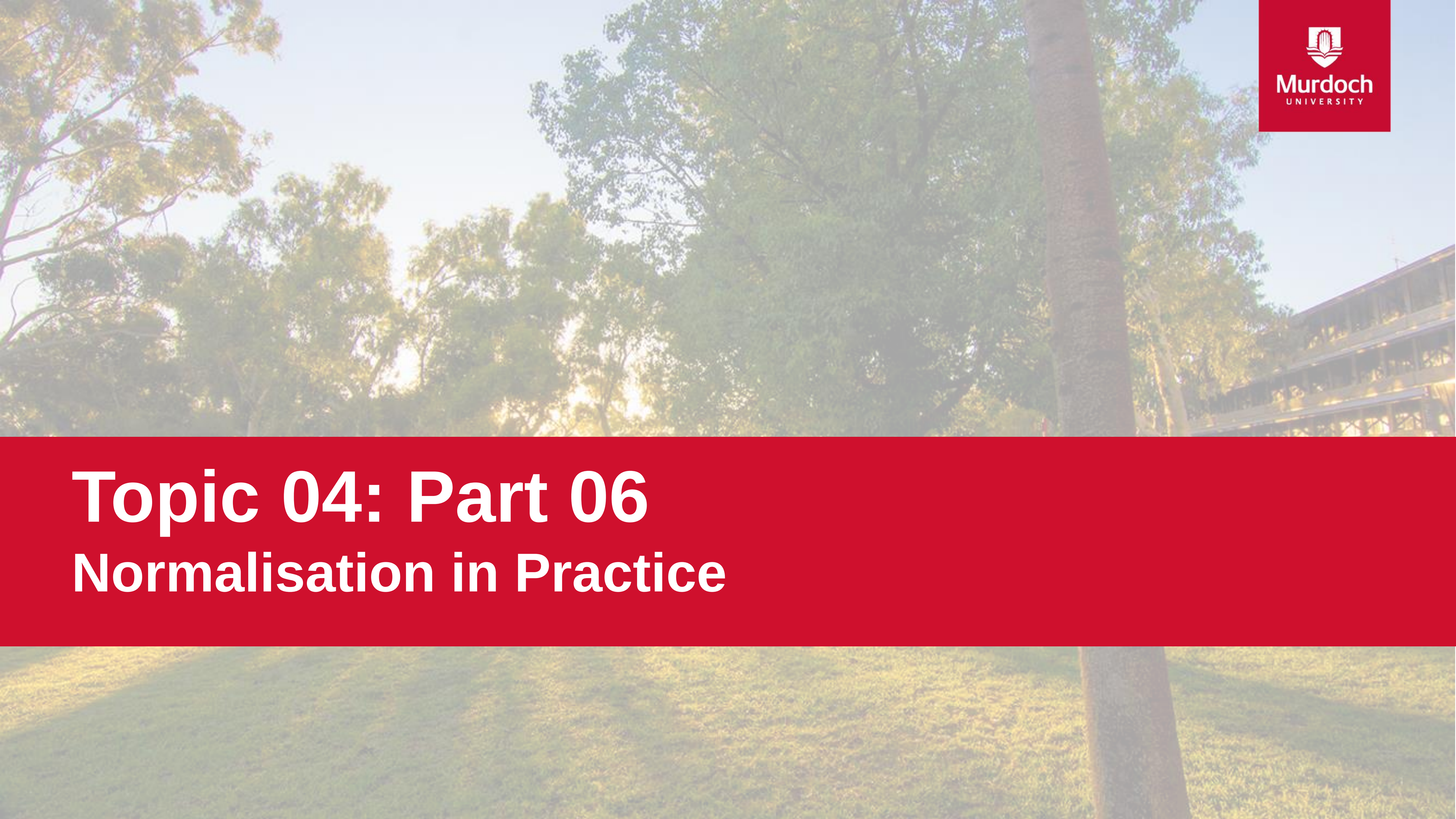
  STUDENT_UNIT (StudentNo, UnitCode)

  STUDENT_SPORT (StudentNo, Sport)

- A relation is in 4NF if it is in 3NF and has **no multi-valued dependencies**

The takeaways…

- BCNF is a ‘stronger’ form of 3NF in which every determinant in a relation is a candidate key
- 4NF eliminates multi-valued dependencies by making a relation for each MVD, with a compound PK



Topic 04: Part 06

Normalisation in Practice

Normalisation in practice

- Normalisation helps to create a 'good' design
- However, in practice we also need to take into account processing requirements on the database
- creating more tables means more joins, so slower processing – this is the tradeoff for avoiding redundancy

e.g. *What are the pros and cons for converting this 2NF table to 3NF?*

PERSON-ADDRESS (<u>Name</u>, Street, Suburb, Postcode)

Name → Street, Suburb

Suburb → Postcode

Denormalisation

- We can always make the choice to leave a table in a less than 3NF form ("**denormalised**"), based on how it will be used in practice
- We make these decisions during the *physical* database design stage

- But always *start* with a normalised design!

Topic 04: Part 07

Conclusion

The takeaways…

- Normalisation to at least 3NF will give a flexible database design with no modification anomalies
- In practice, we can use ‘denormalised’ tables in less than 3NF if processing requirements dictate
- But *starting* with a fully normalised design means that you can make more informed decisions about denormalisation

Topic Learning Outcomes Revisited

- **After completing this topic you should be able to:**

- Describe the aims of good relational database design through normalisation
- Explain the potential modification anomalies (update, insert and delete) associated with redundant information in tables
- Identify functional dependencies among attributes
- Give definitions of the following normal forms: 1NF, 2NF, 3NF, BCNF
- Be able to identify which normal form a given relation is in from examining its functional dependencies
- Normalise a given relation to a higher normal form
- Discuss some practical limitations that may be associated with normalisation

What’s next?

- We’ve now covered the theory and design of relational databases, and seen in the labs how they can be accessed and queried through SQL. In the next few topics, we will look at the database design process and how we move from a set of requirements through to a database that can be implemented in the target DBMS.