Database Fundamentals

NORMALISATION CONCEPTS

Normalisation

- Database designs can be generated by decomposing “Mega” relational schemas into smaller schemas

  - Based on knowledge of the attributes and relationships between them called **Functional Dependencies (FDs)**

  - These smaller relations satisfy some normal form in that they contain **fewer** anomalies and that there is **no loss of information**

Normalisation

- The normalisation process takes a relational schema through a series of tests to “certify” whether or not it belongs to a certain **normal form**

- Separating **Functional Dependencies** into new relational schemas will generate tables in a specific **normal form**

  - 1NF, 2NF, 3NF or even Boyce-Codd normal form (BCNF)

- Each increasing **normal form** provides greater protection against data anomalies from developing

Normalisation

- Example Mega Schema

empID	empName	dob	address	deptNo	deptName	deptMgr
01	Danielle	88	1 abc st	3	Research	02
02	Fredrich	87	13 def rd	3	Research	02
03	Jonathan	65	22 Wilson Ave	2	Admin	04
04	Stacy	73	7a Rose Tc	2	Admin	04
05	Bev	91	386 NE Rd	1	IT	05
06	Ashleigh	95	14 Bull Tce	1	IT	05
07	Peter	95	18 Corn St	4	Comedy	07

Insertion Anomaly:
- Creating a new employee requires all the department information to be filled in correctly
- Creating a new department requires a bogus empID number

Deletion Anomaly:
- Deleting a record for a department with only one employee will lose info about the department

Update Anomaly:
- If the deptMgr is changed, it needs to be changed for ALL records

Flexibility Issue:
- What if an employee ends up working across multiple departments?

Null Values:
- A good design should avoid NULL values because of their side-effects in some Queries

Normalisation

- Normalisation **when done correctly**:
  - Is **lossless**
    - Lossless: When the resulting normalised tables are joined using SQL queries, the original data/schema can be re-created without **spurious tuples**

  - **Retains all the Functional Dependencies** identified
    - They may be represented in some of the individual normalised relations that result
    - No data is lost
    - No “information” contained in the data identified by FDs should be lost

Normalisation

- Example spurious tuples
  - Assumption 1: a specific car (ie plateNo) can be only one colour, plateNo --> colour
  - Assumption 2: colours are associated with specific makes of car, colour + make are a key

plateNo	make	colour
abc-123	kia	black
abc-567	subaru	black
abc-890	cherri	red

Cars(plateNo, make, colour)

Normalise

plateNo	colour
abc-123	black
abc-567	black
abc-890	red

Cars(plateNo, colour)

make	colour
kia	black
subaru	black
cherri	red

CarMakes(make, colour)

- What Happens with Cars JOIN CarMakes?

Normalisation

- What Happens with Cars JOIN CarMakes?

plateNo	make	colour
abc-123	kia	black
abc-567	subaru	black
abc-890	cherri	red

Cars(plateNo, make, colour)

Normalise

plateNo	colour
abc-123	black
abc-567	black
abc-890	red

Cars(plateNo, colour)

make	colour
kia	black
subaru	black
cherri	red

CarMakes(make, colour)


```
SELECT * FROM Cars AS C
JOIN CarMakes AS CM
ON C.colour = CM.colour
```

plateNo	make	colour
abc-123	kia	black
abc-123	subaru	black
abc-567	subaru	black
abc-567	kia	black
abc-890	cherri	red

Normalisation – Functional Dependencies

- A **Functional Dependency (FD)** is a property that describes **how the attributes in a relation relate to one another**

  - A student number is always associated with the same student name
    - A student's ID number uniquely determines the student name
    - studentID --> studentName

  - A course number is always associated with the same course name
    - A course number uniquely determines a course name
    - courseNo --> courseName

- Functional Dependences can be used to help determine **candidate keys** in a relation

Normalisation – Functional Dependencies

- Example:
  - Suppose a student's grade is determined by their mark
  - FD: Mark --> Grade

mark	grade
>= 85	HD
>= 75	D
>= 65	C
>= 50	P
< 50	F

student	mark	grade
Mary	82	D
Ben	92	HD
James	82	D
Peter	81	D
Stacy	65	C

- If this relationship is guaranteed in the data then any two tuples with the same Mark will **ALWAYS** have the same Grade.
  - This would be an example of a Functional Dependency where the grade is determined by the mark
- A FD is based on real world knowledge of the data (not necessarily patterns observed in the data)
  - If we specify an FD then all instances of the data must adhere to the FD
  - Violations of FDs result in redundant information being stored in a table = potential anomalies

Functional Dependencies - Examples

- A **Functional Dependency** may or may not hold true for a given instance
  - Data can be checked to see if it violates a Functional Dependency

Employee_Dept

EName	**EmpID**	**BDate**	**Dept**	**Dname**	**DeptMgr**
Tom	1	20-Dec	CS	Comp Sci	3
John	2	21-Dec	CS	Comp Sci	3
Mary	3	22-Dec	CS	Comp Sci	3
Tom	4	21-Dec	Math	Mathematics	4

EmpID --> {Ename, Bdate, Dept}

Dept --> {Dname, DeptMgr}

Functional Dependencies - Examples

- A **Functional Dependency** may or may not hold true for a given instance
  - Data can be checked to see if it violates a Functional Dependency

employeeID	name	phone	room
1001	Chris	8375	D1-11
1002	Connie	8221	Front Office
1003	Sarah	8221	Front Office
1004	Mary	8375	D2-15

Room --> Phone

The Room functionally determines the phone extension number

The "Front Office" is always associated with the Phone no. 8221

Functional Dependencies - Examples

- A **Functional Dependency** may or may not hold true for a given instance
  - Data can be checked to see if it violates a Functional Dependency

employeeID	name	phone	room
1001	Chris	8375	D1-11
1002	Connie	8221	Front Office
1003	Sarah	8221	Front Office
1004	Mary	8375	D2-15

Phone --> Room

The Phone number DOES NOT determine the Room number

The same phone number is not always associated with the same room

Functional Dependencies - Examples

- A **Functional Dependency** may or may not hold true for a given instance
  - Data can be checked to see if it violates a Functional Dependency

employeeID	name	phone	room
1001	Chris	8375	D1-11
1002	Connie	8221	Front Office
1003	Sarah	8221	Front Office
1004	Mary	8375	D2-15

?

EmployeeID --> Name, Phone, Room is a reasonable **FD** to recommend
Room --> Phone number may be an unsafe **FD**

- What if using VoIP and the phone number follows the staff member?

Functional Dependencies and Keys

- **Functional Dependencies** generalise the notion of keys
  - Suppose we have a relation R(A, B) with no duplicate tuples
    - If A --> B
    - Then for any two or more tuples with the same value(s) for "A", the values for attribute "B" must also be the same.
    - This would result in duplicate tuples (ie violating the primary key)
    - "A" must be a primary key of the relation
  - "A" is called the **Determinant**
  - "B" is called the **Dependent**
    - **Determinant --> Dependent**

If my bike has license plate XYZ-123 and I'm speeding past a speed camera the government can send me a fine

- They can determine me (the **dependent**) via my license plate (the **determinant**)
- Only I have this license plate
- XYZ-123 --> Me!

Rules for Functional Dependencies

- **Splitting Rule**
  - A good functional dependency cannot be broken down further by splitting it into smaller components

  - CourseMarks(studentID, courseID, mark)

    - FD: {studentID, courseID} --> mark

  - Splitting the FD, do the following still hold true?
    - FD1: studentID --> mark
    - FD2: courseID --> mark

Rules for Functional Dependencies

- **Combining Rule**
  - A good functional dependency can be added to other like dependencies without changing the meaning

  - Student(studentID, studentName, phoneNumber, address)
    - FD1: studentID --> studentName
    - FD2: studentID --> phoneNumber
    - FD3: studentID --> address
  - Combining the FDs, do they still hold or does it change the meaning?
    - FDall studentID --> {studentName, phoneNumber, address}

Rules for Functional Dependencies

- **Transitive Rule**
  - If A --> B and B --> C then *technically* A --> C

  - Student(studentID, suburb, postcode, state)

    - FD1: studentID --> suburb, postcode
    - FD2: {suburb, postcode} --> state

    - Transitively: studentID --> state
    - Called a **Transitive Dependency (TD)**

Rules for Functional Dependencies

- **Closure of Attributes**
  - A relation may have numerous **FDs** that hold across all tuples in the relation
  - The set of all **FDs** is called the **closure** and is denoted R^+ or $[R]^+$

  - Lets suppose we are given a relation **R**, a **set** of **Functional Dependencies** for the relation ∑ and a set of attributes "A" that determine the values of all other attributes

    - Then we should be able to find ALL attributes in the relation **R** using "**A**" and the functional dependencies contained in ∑
    - "A" would be a candidate key for the relation R
    - If this is the case then the normalisation has been **Lossless**
      The original data can be re-created by re-combining the FDs

Rules for Functional Dependencies

- **Calculating Closure**
  - The closure of a FD helps us to find the ~~primary~~ **candidate keys** of a relation
  - If the closure $[A]^+$ contains **all** the available attributes via their **FDs**, then "A" is a **candidate key**

  - Student(studentID, studentName, address, mark, grade)

    - FD1: studentID --> studentName, address, mark
    - FD2: mark --> grade

$\sum = FD1 + FD2$

    - Closure $[mark]^+$
    - Closure $[studentID]^+$

Closure Example

Closure Example

- Given the relation R(ABCDEFGH) and the functional dependencies:

R(A,B,C,D,E,F,G,H)

CH --> G
A --> BC
B --> CFH
E --> A
F --> EG

- Determine the available **candidate keys**

Closure Example

1. Determine which attributes are always on the LHS (the **determinants**)
   - Look for attributes whose values are not dependent on other attributes (i.e. attributes that DON'T appear on the RHS of the -->)
   - These are usually not the **dependent** attributes

2. Work out the closure for those attribute(s)

- D+

   - If un-successful, calculate other closures…

3. Work out the closure of determinants + other attributes
   - Look for attributes that feature rarely on the RHS but also appear on the LHS
   - DA +, DB +, DE +, DF +, …

R(A,B,C,D,E,F,G,H)

CH --> G
A --> BC
B --> CFH
E --> A
F --> EG

Closure Example

- Closure of D^+

R(A,B,C,D,E,F,G,H)

CH --> G
A --> BC
B --> CFH
E --> A
F --> EG

Closure Example

- Roriginal(C1, C2, C3, C4, C5, C6)

- Closure of {C1, C2}+

R(C1, C2, C3, C4, C5, C6)

{C1, C2} --> C5
C1 --> C3
C2 --> C4
C5 --> C6

Normalisation

SUMMARY OF THE NORMAL FORMS

Normalisation - Overview

- **Normalisation** is the process of efficiently organising table schemas based on relationships within data
  - **Eliminate redundant data**
    - Prevent storing the same data in more than one table
    - Prevents anomalies and data inconsistencies
  - Creates **smaller schemas** that contain only **relevant information**
    - **Functional Dependencies** are separated into their own tables that are readily updated
    - Only store related data in a table
  - Retain the original information captured
    - Normalisation should not result in a loss of data or information
    - No Spurious tuples

Works well where data is updated frequently but at the expense of information retrieval

Normalisation - Normal Forms Overview

- **1st Normal Form (1NF)**
  - All tables are flat and hold atomic values
    - One cell, one value
    - One column one domain (data type)

- **2nd Normal Form (2NF)**
  - 1NF must hold true
  - All tables must have a valid PK and all values must be "**functionally**" **dependent** on the **whole key**
    - All non-key attributes (dependents) must provide a fact about the key (ie, have data relating to the key)
      - There are **no partial dependencies**
      - No attribute should be dependent on only a portion of the PK
    - At this point many anomalies will have been eliminated

Normalisation - Normal Forms Overview

- **3rd Normal Form (3NF)**
  - 2NF must hold true
  - No attribute must be **transitively dependent** on any other non-key (dependent) attribute
  - Sufficient to guard against anomalies

- **Boyce-Codd Normal Form (BCNF)**
  - Commonly used in preference to 3rd normal form
  - Is a more specific extension of 3rd Normal Form
  - For any dependency, A → B, A must be a **super-key**
    - It is concerned with tables that contain multiple overlapping candidate keys

- **Other Normal Forms**:
  - 4th, and 5th Normal Forms and beyond

Normalisation

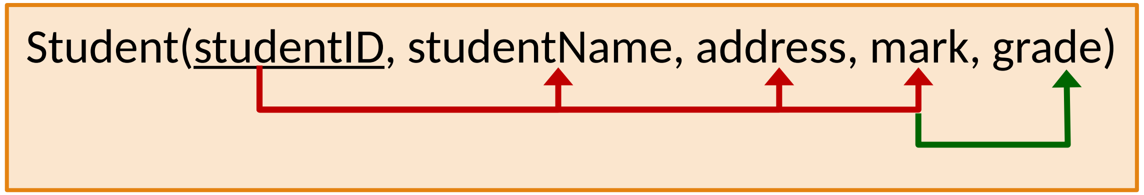
SUMMARY THE STEPS

Normalisation - Steps

1. **Eliminate repeating Groups.**
   Present data in a tabular format, where each cell has a single value and there are no repeating groups
2. **Identify the Primary Key**
   The primary key must uniquely identify any given row.
   If necessary a surrogate key can be used if no key exists
3. **Identify all Dependencies**
   These can be depicted using a dependency diagram to help visualise the relationship between values
4. **Break apart the relation**
   Separate out the various functional dependencies:
   1. Remove Partial Dependencies (PDs) into their own relation
   2. Remove Transitive Dependencies (TDs) into their own relation

- A **Dependency Diagram** depicts all the FDs within the relational schema and can help with the process

Student(studentID, studentName, address, mark, grade)
FD1: studentID -> {studentName, address, mark}
FD2: mark -> grade



Break apart the relation

- **Breaking apart relations based on FDs**

1. For every **functional dependency**, write its **determinant** as a PK for a new table
   - Name the table to reflect its contents

     Determinant: any attribute whose value determines other values within a row

2. Identify the **dependent attributes**

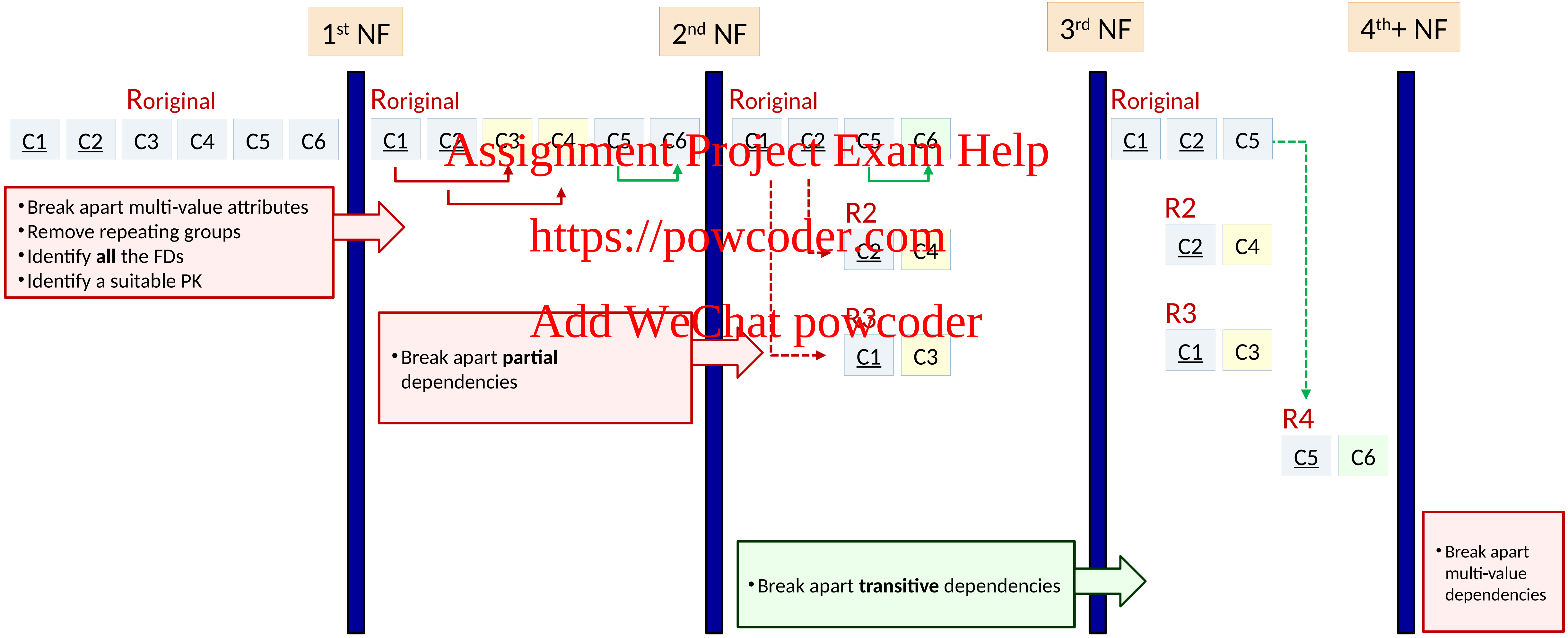
   Dependent: any attribute whose value is dependent on a determinant

   Determinant --> Dependent (ie, A --> B)

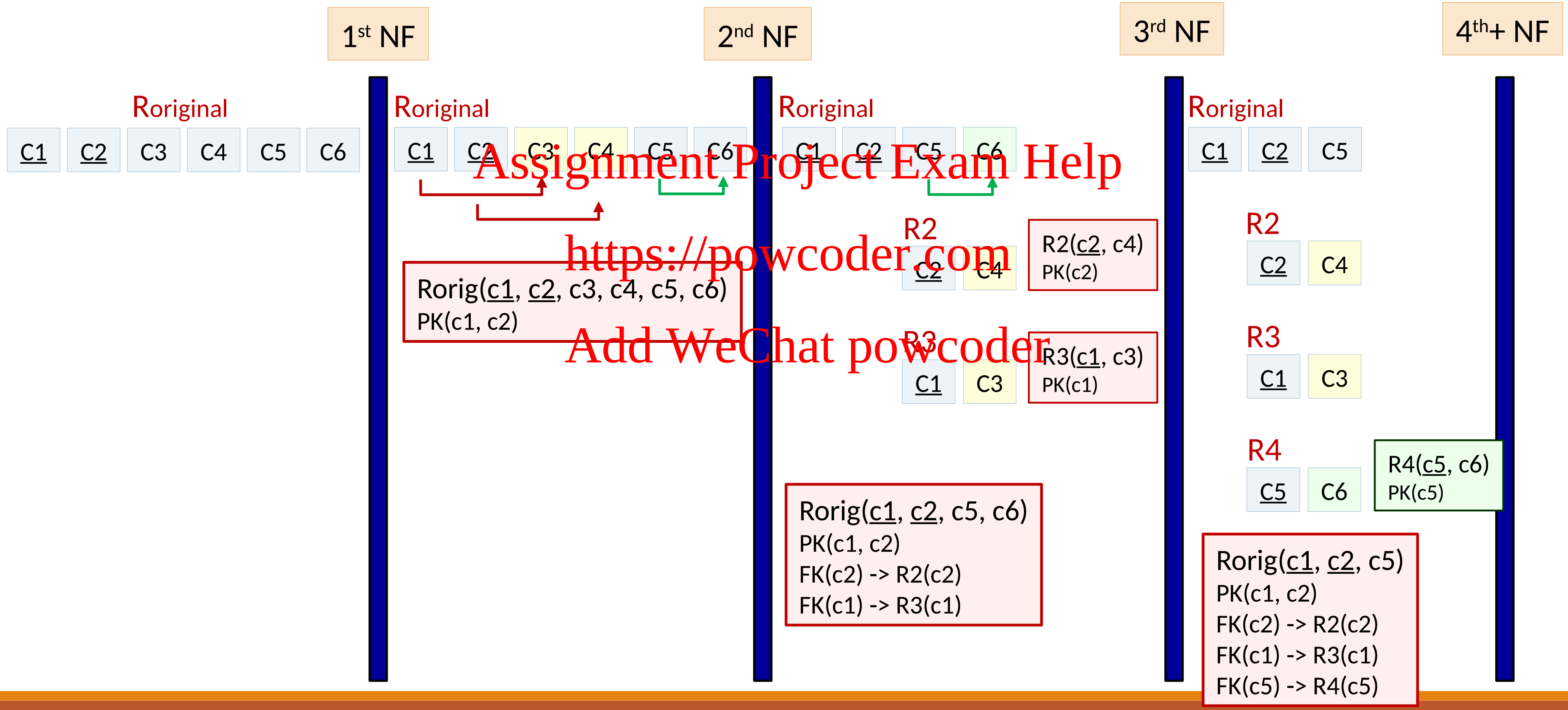
3. Remove the dependent attributes from the partial/transitive dependencies and place them in the new table with the determinant

4. Check the new tables and modified table to ensure that none contain other partial/transitive dependencies

Normalisation – Normal Forms Overview



Normalisation – Normal Forms Overview



Normalisation

EXAMPLES

Normalisation – Example 1

- 1st Normal Form
  - All tables should be “flat”
  - All occurrences of a record must contain the same number of fields
    - All values in a given column must be of the same data type (what does this mean?)
    - A value should NOT be composed of multiple values
      - Atomic values

student	courses
Mary (1001)	{CS145,CS229}
Joe (1002)	{CS145,CS106}
…	…

Not in 1st NF

student	studentID	course
Mary	1001	CS145
Mary	1001	CS229
Joe	1002	CS145
Joe	1002	CS106

1st NF Equivalent

Normalisation – 2nd Normal Form

- Is this in 2nd NF?

StudentEnrolments

studentID	courseID	name	postcode	suburb	courseName	room	mark
1001	CS145	Mary	5000	Adel	FIT	B01	D
1001	CS229	Mary	5000	Adel	PF	B01	C
1003	CS106	Jane	5092	Modbury Nth	DBF	B02	HD
1002	CS145	Joe	5092	Modbury	FIT	B01	P
1002	CS106	Joe	5092	Modbury	DBF	B02	HD

Partial Dependencies

studentID → name, postcode, suburb

courseID → courseName, room

studentID, courseID → mark

Normalisation – 2nd Normal Form

- Is this in 2nd NF?

StudentEnrolments

studentID	name	…	mark
1001	Mary	…	D
1001	Mary	…	C
1003	Jane	…	HD
1002	Joe	…	P
1002	Joe	…	HD

1st NF

Students

<u>studentID</u>	name	postcode	suburb
1001	Mary	5000	Adel
1001	Mary	5000	Adel
1003	Jane	5092	Modbury Nth
1002	Joe	5092	Modbury
1002	Joe	5092	Modbury

2nd NF Equivalent

Courses

<u>courseID</u>	courseName	room
CS145	FIT	R01
CS229	PF	R01
CS106	DBF	R02

2nd NF Equivalent

Normalisation – 2nd Normal Form

- Is this in 2nd NF?

StudentEnrolments

studentID	name	…	mark
1001	Mary	…	D
1001	Mary	…	C
1003	Jane	…	HD
1002	Joe	…	P
1002	Joe	…	HD

1st NF

StudentEnrolments

studentID	courseID	mark
1001	CS145	D
1001	CS229	C
1003	CS106	HD
1002	CS145	P
1002	CS106	HD

2nd NF Equivalent

Original Schema:
Enrolment(studentID, courseID, name, suburb, postcode, courseName, room, mark)
PK(studentID, courseID)
FD: studentID --> name, suburb, postcode
FD: courseID --> courseName, room
FD: {studentID, courseID} --> mark

A Better Schema:
Enrolments(studentID, courseID, mark)
Students (studentID, name, postcode, suburb)
Courses (courseID, courseName, room)

Normalisation – 3rd Normal Form

- Is this in 3rd Normal Form?

Students

<u>studentID</u>	name	postcode	suburb
1001	Mary	5000	Adel
1001	Mary	5000	Adel
1003	Jane	5092	Modbury Nth
1002	Joe	5092	Modbury
1002	Joe	5092	Modbury

2nd NF 3rd NF

Courses

<u>courseID</u>	courseName	room
CS145	FIT	R01
CS229	PF	R01
CS106	DBF	R02

2nd NF 3rd NF

StudentEnrolments

<u>studentID</u>	<u>courseID</u>	mark
1001	CS145	D
1001	CS229	C
1003	CS106	HD
1002	CS145	P
1002	CS106	HD

2nd NF 3rd NF

Are there any values that depend on an attribute other than the PK?

Normalisation – 3rd Normal Form

- Is this in 3rd Normal Form?

Courses

<u>CourseID</u>	CourseName	Building	Room
CS145	FIT	Reid	R1-01
CS229	PF	Reid	R2-01
CS106	DBF	Planetarium	P1-13

2nd NF

The Room functionally determines the Building the course is held in.

The Building can be identified by Room No as well as the CourseID. Neither Building nor Room are a primary key attribute.

CourseID → Room → Building, so there exists **transitive dependency**.

Are there any values that depend on an attribute other than the PK?

Normalisation – 3rd Normal Form

- Is this in 3rd Normal Form?

Courses

<u>CourseID</u>	CourseName	Room
CS145	FIT	R1-01
CS229	PF	R2-01
CS106	DBF	P1-13

3rd NF?

Rooms

<u>Room</u>	Building
R1-01	Reid
R2-01	Reid
P1-13	Planetarium

3rd NF?

A Better Schema:
Courses(<u>CourseID</u>, CourseName, Room)
PK(CourseID)

Rooms(<u>Room</u>, Building)
PK(Room)

Now:
CourseID --> CourseName, Room
Room --> Building

- All values depend on the key and nothing but the key!

Normalisation

EXAMPLES

Normalisation – 1st Normal Form

- Is this in 1st Normal Form?

Character	Voice actor(s)	Character's role	Episode debut	Original air
Homer Simpson	Dan Castellaneta[2]	Husband of Marge; father of Bart, Lisa, and Maggie.	"Good Night" (*The Simpsons* shorts)[3]	19/04/1987
Marge Simpson	Julie Kavner[2]	Wife of Homer; mother of Bart, Lisa, and Maggie.	"Good Night"[3]	19/04/1987
Bart Simpson	Nancy Cartwright[2]	Eldest child and only son of Homer and Marge; brother of Lisa and Maggie.	"Good Night"[3]	19/04/1987
		and eldest Homer and Marge; sister of Bart and Maggie.	"Good Night"[3]	19/04/1987
Maggie Simpson	Liz Georges Gábor Csupó Harry Shearer Yeardley Smith Nancy Cartwright Elizabeth Taylor James Earl Jones Jodie Foster	Youngest child and daughter of Homer and Marge; sister of Bart and Lisa.	"Good Night"[3]	19/04/1987
Akira	George Takei,[4] Hank Azaria[2]	Waiter at The Happy Sumo;[5] karate teacher.[6]	"One Fish, Two Fish, Blowfish, Blue Fish"	24/01/1991

Multi-valued attribute/Repeating Group

Normalisation – 1st Normal Form

- Is this in 1st Normal Form?

Maggie Simpson	Liz Georges
	Gábor Csupó
	Harry Shearer
	Yeardley Smith
	Nancy Cartwrigh
	Elizabeth Taylo
	James Earl Jones
	Jodie Foster
Akira	George Takei,[4] Hank Azaria[2]

Homer Simpson	Dan Castellaneta	Maggie.	Good Night	19/04/1987
Marge Simpson	Julie Kavner	Wife of Homer; mother of Bart, Lisa, and Maggie.	Good Night	19/04/1987
Bart Simpson	Nancy Cart	Eldest child and only		
Lisa Simpson	Yeardley S			
Maggie Simpson	Liz Georges	Youngest child and daughter of Homer and Marge; sister of Bart and Lisa.	Good Night	19/04/1987
Maggie Simpson	Gábor Csupó	...	Good Night	19/04/1987
Maggie Simpson	Harry Shearer	...	Good Night	19/04/1987
Maggie Simpson	Yeardley Smith	...	Good Night	19/04/1987
Maggie Simpson	Nancy Cartwright	...	Good Night	19/04/1987
Maggie Simpson	Elizabeth Taylor	...	Good Night	19/04/1987
Maggie Simpson	James Earl Jones	...	Good Night	19/04/1987
Maggie Simpson	Jodie Foster	...	Good Night	19/04/1987
Akira	George Takei	Waiter at The Happy Sumo;[5] karate teacher.[6]	One Fish, Two Fish, Blowfish	24/01/1991
		Waiter at The Happy Sumo;[5] karate		

1st Normal Form:
Transform **repeating** values and **groups** of values into individual rows of data and duplicate any missing data as necessary

Normalisation – 2nd Normal Form

- Is this in 2nd Normal Form?
  - A table of characters, the actor that played the character, the episode and date the character 1st aired

Character	Actor	Episode	originalAirDate
Homer Simpson	[illegible]	Good Night	19/04/1987
Marge Simpson	Julie Kavner	Good Night	19/04/1987
Bart Simpson	Nancy Cartwright	Good Night	19/04/1987
Lisa Simpson	Yeardley Smith	Good Night	19/04/1987
[illegible]	[illegible]	[illegible]	[illegible]87
[illegible]	[illegible]	[illegible]	[illegible]87
[illegible]	[illegible]	[illegible]	[illegible]87
[illegible]	[illegible]	[illegible]	[illegible]87
[illegible]	[illegible]	[illegible]	[illegible]87
Maggie Simpson	Elizabeth Taylor	Good Night	19/04/1987
Maggie Simpson	James Earl Jones	Good Night	19/04/1987
Maggie Simpson	Jodie Foster	Good Night	19/04/1987
Akira	George Takei	One Fish, Two Fish, Blowfish	24/01/1991
Akira	Hank Azaria	One Fish, Two Fish, Blowfish	24/01/1991

NO

Only the Character determines the Episode and originalAirDate

Character → Episode, originalAirDate

Actor = irrelevant (using the provided data)

Normalisation – 2nd Normal Form

- Is this in 2nd Normal Form?
  - PK(Character, Actor) and PK(Character)

Character	Actor
Homer Simpson	Dan Castellaneta
Marge Simpson	Julie Kavner
Bart Simpson	Nancy Cartwright
Lisa Simpson	Yeardley Smith
Maggie Simpson	Liz Georges
Maggie Simpson	Gábor Csupó
Maggie Simpson	Harry Shearer
Maggie Simpson	Yeardley Smith
Maggie Simpson	Nancy Cartwright
Akira	Hank Azaria

YES
The table only consists of a Composite PK – No partial dependencies can exist

Character	Episode	originalAirDate
Homer Simpson	Good Night	19/04/1987
Marge Simpson	Good Night	19/04/1987
Bart Simpson	Good Night	19/04/1987
Lisa Simpson	Good Night	19/04/1987
Maggie Simpson	Good Night	19/04/1987
Maggie Simpson	Good Night	19/04/1987
Maggie Simpson	Good Night	
Maggie Simpson	Good Night	
Maggie Simpson	Good Night	
		19/04/1987
		19/04/1987
Akira	One Fish, Two Fish, Blowfish	24/01/1991
Akira	One Fish, Two Fish, Blowfish	24/01/1991

YES
The table has a single value PK – No partial dependencies can exist

These Duplicate records are not needed

Normalisation – 2nd Normal Form

- Is this in 2nd Normal Form?
  - PK(Character, Actor) and PK(Character)

Character	Actor
Homer Simpson	Dan Castellaneta
Marge Simpson	Julie Kavner
Bart Simpson	Nancy Cartwright
Lisa Simpson	Yeardley Smith
Maggie Simpson	Liz Georges
Maggie Simpson	Gábor Csupó
Maggie Simpson	Harry Shearer
Maggie Simpson	Yeardley Smith
Maggie Simpson	Nancy Cartwright
Akira	Hank Azaria

YES
The table only consists of a Composite PK – No partial dependencies can exist

Character	Episode	originalAirDate
Homer Simpson	Good Night	19/04/1987
Marge Simpson	Good Night	19/04/1987
Bart Simpson	Good Night	19/04/1987
Lisa Simpson	Good Night	19/04/1987
Maggie Simpson	Good Night	19/04/1987
Akira	One Fish, Two Fish, Blowfish	24/01/1991

The advantage is no duplicates

YES
The table has a single value PK – No partial dependencies can exist

Normalisation – 3rd Normal Form

- Is this in 2nd Normal Form?
  - PK(Character, Actor) and PK(Character)

Character	Actor
Homer Simpson	Dan Castellaneta
Marge Simpson	Julie Kavner
Bart Simpson	Nancy Cartwright
Lisa Simpson	Yeardley Smith
Maggie Simpson	Liz Georges
Maggie Simpson	Gábor Csupó
Maggie Simpson	Harry Shearer
Maggie Simpson	Yeardley Smith
Maggie Simpson	Nancy Cartwright
Akira	Hank Azaria

Character	Episode	originalAirDate
Homer Simpson	Good Night	19/04/1987
Marge Simpson	Good Night	19/04/1987
Bart Simpson	Good Night	19/04/1987
Lisa Simpson	Good Night	19/04/1987
Maggie Simpson	Good Night	19/04/1987
Akira	One Fish, Two Fish, Blowfish	24/01/1991

YES
The table only consists of a only a key and thus no transitive dependencies can exist

NO
The Episode determines the original AirDate and neither is the PK
Episode -> originalAirDate

Normalisation – 3rd Normal Form

- Is this in 3rd Normal Form?

Character	Episode
Homer Simpson	Good Night
Marge Simpson	Good Night
Bart Simpson	Good Night
Lisa Simpson	Good Night
Maggie Simpson	Good Night
Akira	One Fish, Two Fish, Blowfish

Episode	originalAirDate
Good Night	19/04/1987
Good Night	19/04/1987
Good Night	19/04/1987
Good Night	19/04/1987
Good Night	19/04/1987
One Fish, Two Fish, Blowfish	24/01/1991

YES
The table only consists of a only a key and one other attribute - no transitive dependencies can exist

Yes
The Episode always determines the same original AirDate and the episode is the PK. There is also only one non-key attribute so no transitive dependency can exist
Episode -> originalAirDate

Normalisation – 3rd Normal Form

- These are all in 3rd Normal Form

Character	Actor
Homer Simpson	Dan Castellaneta
Marge Simpson	Julie Kavner
Bart Simpson	Nancy Cartwright
Lisa Simpson	Yeardley Smith
Maggie Simpson	Liz Georges
Maggie Simpson	Gábor Csupó
Maggie Simpson	Harry Shearer
Maggie Simpson	Yeardley Smith
Maggie Simpson	Nancy Cartwright
Maggie Simpson	Elizabeth Taylor
Maggie Simpson	James Earl Jones
Maggie Simpson	Jodie Foster
Akira	George Takei
Akira	Hank Azaria

Episode	originalAirDate
Good Night	19/04/1987
One Fish, Two Fish, Blowfish	24/01/1991

Character	Episode
Homer Simpson	Good Night
Marge Simpson	Good Night
Bart Simpson	Good Night
Lisa Simpson	Good Night
Maggie Simpson	Good Night
Akira	One Fish, Two Fish, Blowfish

A Better Schema

Characters(Character, Actor)
PK(Character, Actor)

Episodes(Episode, OriginalAirDate)
PK(Episode)

CharacterFirstEpsiode(Character, Episode)
PK(Character)

Normalisation

- Simpsons character database:

Our Original Simpsons DB

R(Character, Actor, Role, Episode, DateAired)

FD1 Character → Role, Episode ⇒ Partial Dependency (PD)

FD2 Episode → DateAired ⇒ Transitive Dependency (TD)

Take each Functional Dependency and place it into a new relation with ALL of its dependents

Normalisation

Our Simpsons DB

R(<u>Character, Actor</u>, Role, Episode, DateAired)

FD1 Character → Role, Episode ⇒ Partial Dependency (PD)

R1(<u>Character</u>, Role, Episode, DateAired)

R(<u>Character, Actor</u>)

Functional dependencies removed

FD2 Episode → DateAired ⇒ Transitive Dependency (TD)

R2(<u>Episode</u>, DateAired)

R1(<u>Character</u>, Role, Episode)

Take each Functional Dependency and place it into a new **primary** relation

Normalisation

Our Simpsons DB

R(Character, Actor, Role, Episode, DateAired)

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

R(Character, Actor) 3NF

R2(Episode, DateAired) 3NF

R1(Character, Role, Episode) 3NF

Normalisation – 2nd Normal Form

- Is this in 2nd Normal Form?
  - PK(CharacterID) **a surrogate key**

CharacterID	Character	actors	Episode	originalAirDate
1	Homer Simpson	Dan Castellaneta	Good Night	19/04/1987
2	Marge Simpson	Julie Kavner	Good Night	19/04/1987
3	Bart Simpson	Nancy Cartwright	Good Night	19/04/1987
4	Lisa Simpson	Yeardley Smith	Good Night	19/04/1987
5				/1987
6				/1987
7				/1987
8				/1987
9				/1987
10				/1987
11	Maggie Simpson	James Earl Jones	Good Night	/1987
12	Maggie Simpson	Jodie Foster	Good Night	19/04/1987
13	Akira	George Takei	One Fish, Two Fish, Blowfish	24/01/1991
14	Akira	Hank Azaria	One Fish, Two Fish, Blowfish	24/01/1991

YES

By definition a table with a surrogate/single column PK will be in 2nd Normal Form as there will be NO attribute(s) that rely on only part of the PK