MONASH University

MONASH
INFORMATION
TECHNOLOGY

Week 4

Normalisation

Workshop 2022 Semester 2

INSERT, UPDATE and DELETE Anomalies

- INSERT Anomaly
  - When adding data to a relation you are required to add other (related) data
  - Danger: other data may not be available so cannot proceed with the insert
- UPDATE Anomaly
  - Changing a value for an attribute requires multiple tuples to be changed
  - Danger: only some tuples will be updated leading to inconsistent data
- DELETE Anomaly
  - When a tuple in a relation is deleted, all tuple data is removed
  - Danger: related data, which may be the only such data will be lost

INSERT, UPDATE and DELETE Anomalies Example

DRUG data

DRUG_CODE	DRUG_NAME	SLSREP_ID	SLSREP_NAME	SLSREP_MOBILE
977HSW	CS-Brain	4	Mala Attaway	2379307017
682KBI	Gemfibrozil	69172	Paula Gregoletti	6154866270
993JVA	Acne Solutions Clarifying	901	Graham Oxherd	7247448365
807WZO	Piroxicam	69172	Paula Gregoletti	6154866270
381EXT	Prednicarbate	69172	Paula Gregoletti	6154866270
363PNN	OxygenOX	4	Mala Attaway	2379307017
975YZK	Celebrex	4	Mala Attaway	2379307017
177CUZ	Diflunisal	4	Mala Attaway	2379307017
325GZQ	Rigidity HP	37	Sherilyn Sturney	4647420304
010VNK	Calamine	901	Graham Oxherd	7247448365

INSERT, UPDATE and DELETE Anomalies Example

- INSERT Anomaly
  - cannot add a new SLSREP until they have been assigned a DRUG or add a new DRUG until assigned to a SLSREP
- UPDATE Anomaly
  - changing a SLSREP mobile number requires changes to multiple rows
- DELETE Anomaly
  - delete a DRUG may lose SLSREP details eg. "Rigidty HP" or deleting a SLSREP may lose DRUG details eg. "Sherilyn Sturney"

DRUG data

DRUG_CODE	DRUG_NAME	SLSREP_ID	SLSREP_NAME	SLSREP_MOBILE
977HSW	CS-Brain	4	Mala Attaway	2379307017
682KBI	Gemfibrozil	69172	Paula Gregoletti	6154866270
993JVA	Acne Solutions Clarifying	901	Graham Oxherd	7247448365
807WZO	Piroxicam	69172	Paula Gregoletti	6154866270
381EXT	Prednicarbate	69172	Paula Gregoletti	6154866270
363PNN	OxygenOX	4	Mala Attaway	2379307017
975YZK	Celebrex	4	Mala Attaway	2379307017
177CUZ	Diflunisal	4	Mala Attaway	2379307017
325GZQ	Rigidity HP	37	Sherilyn Sturney	4647420304
010VNK	Calamine	901	Graham Oxherd	7247448365

Data Normalisation

- Relations MUST be normalised in order to avoid anomalies which may occur when inserting, updating and deleting data.
- Normalisation is a ***systematic series of steps*** for progressively refining the data model.
- A formal approach to analysing relations based on their primary key / candidate keys and functional dependencies.
- Used:
  - as a design technique "bottom up design", and
  - as a way of validating structures produced via "top down design" (ER model converted to a logical model - see next week)
  - for *this unit* only concerned with conversion to third normal form - higher normal forms exist (Boyce Codd Normal Form, fourth normal form …)

The Normalisation Process Goals

- Creating valid relations, i.e. each relation meets the properties of the relational model. In particular:
  - Entity integrity
  - Referential integrity
  - No many-to-many relationship
  - Each cell contains a single value (is atomic).
- In practical terms when implemented in an RDBMS:
  - Each table represents a single subject
  - No data item will be *unnecessarily* stored in more than one table (remember some redundancy still exists - minimal redundancy).
  - The relationship between tables can be established (via PK and FK pairs).
  - Each table is void of insert, update and delete anomalies.

Representing a form as a relation

- This process follows a **standard** approach:
  - arrive at a name for the form which indicates what it represents (its subject)
  - determine if any attribute is multivalued (repeating) **for a given entity instance of the forms subject**
    - if an attribute (or set of attributes) appears multiple times then the group of related attributes need to be shown enclosed in brackets to indicate there are multiple sets of these values for each instance
- Looking at our DRUG data
  - Name: DRUG_SLSREP
    - DRUG_SLSREP (drug_code, drug_name, slsrep_id, slsrep_name, slsrep_mobile)
  - i.e. the form consists of repeating rows (instances) of drugs assigned to sales representatives data

Q1: Representing a form as a relation

STOCK DETAILS

20th March 2021

Part Number: 103
Part Name: 8" Heavy Duty Secateurs

Category Code: GT
Category Name: Gardening Tools

Stock: 35

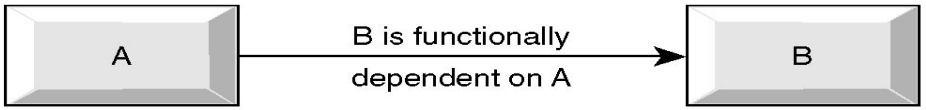
Sell Price: $19.00

Vendor No	Vendor Name	Date Purchased	Cost per Unit	Qty Supplied	Payment
12	Saxon	01 Oct 2020	$13.00	20	$260.00
23	Fiskers	15 Dec 2020	$14.50	30	$435.00
12	Saxon	15 Dec 2020	$16.00	20	$320.00

Unnormalised Form (UNF)

- The UNF representation of a relation is the representation which you have mapped from your inspection of the form
  - it is a **single** named representation (name is not pluralised)
  - no PK etc have as yet been identified
- PART (part_no, part_name, cat_code, cat_name, part_stock, part_sell (vendor_no, vendor_name, restock_date_purchased, restock_costpu, restock_qtysupplied, restock_payment))
  - Is this a relation
    - Reasons?

Functional Dependency Revisited



- An attribute B is FUNCTIONALLY DEPENDENT on another attribute A, if a value of A determines a single value of B at any one time.
  - A ➔ B
  - PRODNO ➔ PRODDESC
  - CUSTNUMB ➔ CUSTNAME
  - ORDERNO ➔ ORDERDATE
    - ORDERNO - independent variable, also known as the DETERMINANT
    - ORDERDATE - dependent variable
- **TOTAL DEPENDENCY**
  - attribute A determines B AND attribute B determines A
    - EMPLOYEE-NUMBER ➔ TAX-FILE-NUMBER
    - TAX-FILE-NUMBER ➔ EMPLOYEE-NUMBER

Functional Dependency

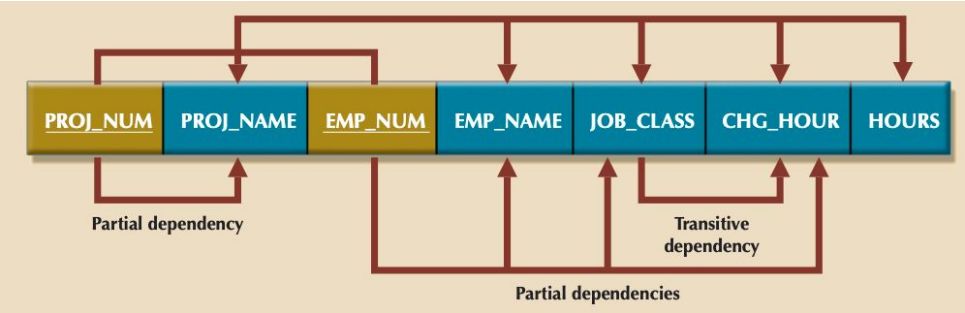
- For a ***composite*** PRIMARY KEY, it is possible to have **FULL** or **PARTIAL** dependency.
- **FULL DEPENDENCY**
  - occurs when an attribute is always dependent on all attributes in the composite PK
  - ORDERNO, PRODNO ➔ QTYORDERED
- Lack of full dependency for multiple attribute key = **PARTIAL DEPENDENCY**
  - ORDERNO, PRODNO
    ➔ PRODDESC, QTYORDERED
  - here although qtyordered is ***fully dependent*** on orderno and prodno, *only* prodno is required to determine proddesc
  - proddesc is said to be ***partially dependent*** on orderno and prodno

Functional Dependency

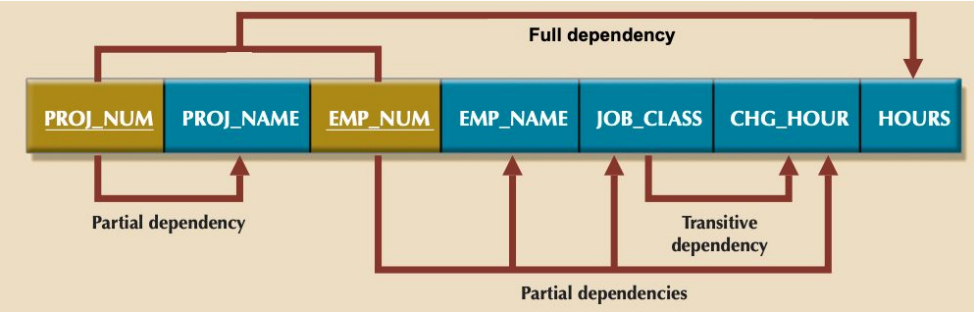
- **TRANSITIVE DEPENDENCY**
  - occurs when Y depends on X, and Z depends on Y - thus Z also depends on X ie. X ➔ Y ➔ Z
  - **and** Y is not a candidate key (or part of a candidate key)
  - ORDERNO ➔ CUSTNUMB ➔ CUSTNAME
- Dependencies are depicted with the help of a **Dependency Diagram**.
- Normalisation converts a relation into relations of progressively smaller number of attributes and tuples until an optimum level of decomposition is reached - little or no data redundancy exists.
- The output from normalisation is a set of relations that meet all conditions set in the relational model principles.

Dependency Diagrams

Figure 6.3 From the text:



For this unit (note dependencies show at each normal form - see following slides):



For ease of drawing we would show, for example, the full dependency above as:

proj_num, emp_num -> hours

First Normal Form

- FIRST NORMAL FORM (part of formal definition of a relation)
  - A RELATION IS IN FIRST NORMAL FORM (1NF) IF:
    - a *unique primary key has been identified* for each tuple/row.
    - it *is a valid relation*
      - Entity integrity (no part of PK is null)
      - Single value for each cell ie. no repeating group (multivalued attribute).
    - all attributes are functionally dependent on all or part of the primary key

UNF to 1NF

- Note we do not use the approach shown in your text of representing the data in tabular form ie. "flattening the data"
  - this works for simple cases but cannot always be used (you *must not* use this approach)
- Move from UNF to 1NF by:
  - identifying a unique identifier for the repeating group.
  - *remove any repeating group* **along with** the PK of the main relation.
  - The PK of the new relation resulting from the removal of repeating group will *normally* have a composite PK made up of the PK of the main relation and the unique identifier chosen in 1. above, but this ***must be checked***.

Q2: UNF to 1NF

UNF:
PART (part_no, part_name, cat_code, cat_name, part_stock, part_sell (vendor_no, vendor_name, restock_date_purchased, restock_costpu, restock_qtysupplied, restock_payment))

Yields which of the following in moving to 1NF:

A:
PART (part_no, part_name, cat_code, cat_name, part_stock, part_sell, vendor_no, vendor_name, restock_date_purchased, restock_costpu, restock_qtysupplied, restock_payment)

B:
PART (part_no, part_name, cat_code, cat_name, part_stock, part_sell)
RESTOCK (part_no, vendor_no, restock_date_purchased, vendor_name, restock_costpu, restock_qtysupplied, restock_payment)

C:
PART (part_no, vendor_no, restock_date_purchased, part_name, cat_code, cat_name, part_stock, part_sell, vendor_name, restock_costpu, restock_qtysupplied, restock_payment)

D:
PART (part_no, part_name, part_stock, part_sell)
CATEGORY (cat_code, cat_name))
RESTOCK (part_no, vendor_no, restock_date_purchased, vendor_name, restock_costpu, restock_qtysupplied, restock_payment)

E:
None of these

UNF to 1NF

Note show only partial dependencies at 1NF

UNF

PART (part_no, part_name, cat_code, cat_name, part_stock, part_sell
(vendor_no, vendor_name, restock_date_purchased, restock_costpu,
restock_qtysupplied, restock_payment))

1NF

PART (part_no, part_name, cat_code, cat_name, part_stock, part_sell)

RESTOCK (part_no, vendor_no, restock_date_purchased, vendor_name,
restock_costpu, restock_qtysupplied, restock_payment)

Partial Dependencies:
vendor_no -> vendor_name

1NF to 2NF

- A RELATION IS IN 2NF IF -
  - all non key attributes are functionally dependent on the primary key (simple definition)
    - used by the textbook in examples
  - all non key attributes are functionally dependent on **any candidate key** (general definition)
    - see textbook section 6-3 (last paragraph immediately below table 6.2), same as *simple* if only one candidate key
    - **General is the requirement for our unit**

Q3: 1NF to 2NF

RESTOCK (part_no, vendor_no, restock_date_purchased, vendor_name, restock_costpu, restock_qtysupplied, restock_payment)

a. **vendor_no -> vendor_name, remove partial**

Removing the partial dependency yields:

A

RESTOCK (part_no, restock_date_purchased, restock_costpu, restock_qtysupplied, restock_payment)
VENDOR (vendor_no, vendor_name)

B

RESTOCK (part_no, vendor_no, restock_date_purchased, vendor_name, restock_costpu, restock_qtysupplied, restock_payment)
VENDOR (vendor_no, vendor_name)

C

RESTOCK (part_no, vendor_no, restock_date_purchased, restock_costpu, restock_qtysupplied, restock_payment)
VENDOR (vendor_no, vendor_name)

D

None of these

1NF to 2NF

Note show only transitive dependencies at 2NF

1NF

PART (part_no, part_name, cat_code, cat_name, part_stock, part_sell)

RESTOCK (part_no, vendor_no, restock_date_purchased, vendor_name, restock_costpu, restock_qtysupplied, restock_payment)

- vendor_no -> vendor_name, remove partial

2NF

PART (part_no, part_name, cat_code, cat_name, part_stock, part_sell)

RESTOCK (part_no, vendor_no, restock_date_purchased, restock_costpu, restock_qtysupplied, restock_payment)

VENDOR (vendor_no, vendor_name)

Transitive Dependencies:
cat_code -> cat_name

2NF to 3NF

- A RELATION IS IN 3NF IF -
  - all transitive dependencies have been removed - check for ***non key attribute dependent on another non key attribute***

- Move from 2NF to 3NF by removing transitive dependencies
  - Remove the attributes with transitive dependency into a new relation.
  - The determinant will be an attribute in both the original and new relations (it will become a PK / FK relationship)
  - Assign the determinant to be the PK of the new relation.

2NF to 3NF

2NF
PART (part_no, part_name, cat_code, cat_name, part_stock, part_sell)

RESTOCK (part_no, vendor_no, restock_date_purchased, restock_costpu, restock_qtysupplied, restock_payment)

VENDOR (vendor_no, vendor_name)

Transitive Dependencies:
cat_code -> cat_name

3NF
PART (part_no, part_name, cat_code, part_stock, part_sell)

RESTOCK (part_no, vendor_no, restock_date_purchased, restock_costpu, restock_qtysupplied, restock_payment)

VENDOR (vendor_no, vendor_name)

CATEGORY (cat_code, cat_name)

Relations in 3NF

Note show ALL full dependencies at 3NF

PART (part_no, part_name, cat_code, part_stock, part_sell)

RESTOCK (part_no, vendor_no, restock_date_purchased, restock_costpu, restock_qtysupplied, restock_payment)

VENDOR (vendor_no, vendor_name)

CATEGORY (cat_code, cat_name)

Full Dependencies:
part_no -> part_name, cat_code, part_stock, part_sell
part_no, vendor_no, restock_date_purchased -> restock_costpu, restock_qtysupplied, restock_payment
vendor_no -> vendor_name
cat_code -> cat_name

Q4. The final set of 3NF relations

PART (part_no, part_name, cat_code, part_stock, part_sell)

RESTOCK (part_no, vendor_no, restock_date_purchased, restock_costpu, restock_qtysupplied, restock_payment)

VENDOR (vendor_no, vendor_name)

CATEGORY (cat_code, cat_name)

has:

A. 4 PKs and 2 FKs
B. 6 PKs and 3 FKs
C. 4 PKs and 3 FKs
D. None of these is correct

The full process UNF to 3NF

UNF
PART (part_no, part_name, cat_code, cat_name, part_stock, part_sell (vendor_no, vendor_name, restock_date_purchased, restock_costpu, restock_qtysupplied, restock_payment))

1NF
PART (part_no, part_name, cat_code, cat_name, part_stock, part_sell)
RESTOCK (part_no, vendor_no, restock_date_purchased, vendor_name, restock_costpu, restock_qtysupplied, restock_payment)

Partial Dependencies:
vendor_no -> vendor_name

2NF
PART (part_no, part_name, cat_code, cat_name, part_stock, part_sell)
RESTOCK (part_no, vendor_no, restock_date_purchased, restock_costpu, restock_qtysupplied, restock_payment)
VENDOR (vendor_no, vendor_name)

Transitive Dependencies:
cat_code -> cat_name

3NF
PART (part_no, part_name, cat_code, part_stock, part_sell)
RESTOCK (part_no, vendor_no, restock_date_purchased, restock_costpu, restock_qtysupplied, restock_payment)
VENDOR (vendor_no, vendor_name)
CATEGORY (cat_code, cat_name)

Full Dependencies:
part_no -> part_name, cat_code, part_stock, part_sell
part_no, vendor_no, restock_date_purchased -> restock_costpu, restock_qtysupplied, restock_payment
vendor_no -> vendor_name
cat_code -> cat_name

Q5. Normalisation Exercise - Prepare UNF for the Drone Service Report

DRONE SERVICE REPORT
Drone ID: 100
Drone Type: DMA2

Drone Date Purchased: 13th Jan 2020
Drone Date Decommissioned: 10 Aug 2022

Date Serviced	Employee Who Serviced the Drone			
	Employee No	**Employee First Name**	**Employee Last Name**	**Employee Type**
13th Jan 2020	10	Trevor	Harewood	FullTime
1st June 2020	12	Gwynne	Sisley	Casual
12th Sep 2020	10	Trevor	Harewood	FullTime

Note only some rows shown ……

Q6. Normalisation Exercise - Continue the process to 3NF

UNF

DRONE (drone_id, dt_code, drone_pur_date, drone_decom_date (ds_date_serviced, emp_no, emp_fname, emp_lname, emp_type))

INF

DRONE (drone_id, dt_code, drone_pur_date, drone_decom_date)

DRONE_SERVICE (drone_id, ds_date_serviced, emp_no, emp_fname, emp_lname, emp_type)

No partial dependency

2NF

DRONE (drone_id, dt_code, drone_pur_date, drone_decom_date)

DRONE_SERVICE (drone_id, ds_date_serviced, emp_no, emp_fname, emp_lname, emp_type)

Transitive Dependency

emp_no -> emp_fname, emp_lname, emp_type

3NF

DRONE (drone_id, dt_code, drone_pur_date, drone_decom_date)

DRONE_SERVICE (drone_id, ds_date_serviced, emp_no)

EMPLOYEE (emp_no, emp_fname, emp_lname, emp_type)

Full dependencies
drone_id -> dt_code, drone_pur_date, drone_decom_date
drone_id, ds_date_serviced -> emp_no
emp_no -> emp_fname, emp_lname, emp_type

Q7. Normalisation Exercise - Repeat process for the Drone Rental Details Report

DRONE RENTAL DETAILS
Drone ID: 180
Drone Type: SWPS
Drone Manufacturer: SwellPro

Drone Date Purchased: 15th Feb 2021

Rental Number	Rented	Booked Out By Employee No	Returned	Booked in By Employee No	Drone Returned Damaged
123	13th Mar 2021 9:30 AM	10	13th Mar 2021 4:30 PM	10	No
256	1st April 2021 1:00 PM	15	3rd April 2021 4:00 PM	12	No
312	12th April 2021 10:00 AM	12	19th April 2021 12 Noon	15	Yes

UNF - 1NF

UNF

DRONE (drone_id, dt_code, dt_manufacturer, drone_pur_date
(rent_no, rent_out_dt, emp_no_out, rent_in_dt, emp_no_in,
rent_returned_damaged))

INF

DRONE (drone_id, dt_code, dt_manufacturer, drone_pur_date)

RENTAL (rent_no, rent_out_dt, emp_no_out, rent_in_dt, emp_no_in,
rent_returned_damaged, drone_id)

No partial dependency

2NF - 3NF

2NF

DRONE (drone_id, dt_code, dt_manufacturer, drone_pur_date)

RENTAL (rent_no, rent_out_dt, emp_no_out, rent_in_dt, emp_no_in, rent_returned_damaged, drone_id)

Transitive Dependency:
dt_code -> dt_manufacturer

3NF

DRONE_TYPE (dt_code, dt_manufacturer)
DRONE (drone_id, dt_code, drone_pur_date)
RENTAL (rent_no, rent_out_dt, emp_no_out, rent_in_dt, emp_no_in, rent_returned_damaged, drone_id)

Full Dependency
dt_code -> dt_manufacturer
drone_id -> dt_code, drone_pur_date
rent_no -> rent_out_dt, emp_no_out, rent_in_dt, emp_no_in, rent_returned_damaged, drone_id

Synthesis - which relations represent the "same" thing

Service:
DRONE (drone_id, dt_code, drone_pur_date, drone_decom_date)
DRONE_SERVICE (drone_id, ds_date_serviced, emp_no)
EMPLOYEE (emp_no, emp_fname, emp_lname, emp_type)

Rental:

DRONE_TYPE (dt_code, dt_manufacturer)
DRONE (drone_id, dt_code, drone_pur_date)
RENTAL (rent_no, rent_out_dt, emp_no_out, rent_in_dt, emp_no_in, rent_returned_damaged, drone_id)

FINAL:

DRONE_TYPE (dt_code, dt_manufacturer)

DRONE (drone_id, dt_code, drone_pur_date, drone_decom_date)

DRONE_SERVICE (drone_id, ds_date_serviced, emp_no)

EMPLOYEE (emp_no, emp_fname, emp_lname, emp_type)

RENTAL (rent_no, rent_out_dt, emp_no_out, rent_in_dt, emp_no_in, rent_returned_damaged, drone_id)

Post Workshop Task - answer available Sunday 5 PM

Normalise this form:

EMPLOYEE ON-BOARDING FORM

Employee Number	**1123 (office use only)**			
First Name	Ada	**Last Name**	Lovelace	
DOB	1-Jan-1990			
Address	*Street No*	*Street*	*Suburb*	*Postcode*
	900	Dandenong Rd	Caulfield East	3145
Phone	04113344556 (M), 99031000 (OFFICE)			
Qualifications				
	Degree Name	**Institution**	**Year**	
	Bachelor of Computer Science	MIT	2011	
	Master of Information Technology	Monash	2013	
Family Members				
	No	**Name**	DOB	
	1	Albert Einstein	02-Jan-1992	
	2	Grace Hopper	12-May-1994	
SKILL (tick selected)				
	Skill name			
	Java			
	SQL			
	SPARK			
	Python			

Summary

- Things to remember
  - Represent form as presented, no interpretation, to yield starting point (UNF)
  - Functional dependency
  - Process of removing attributes in relations based on the concept of 1NF, 2NF and 3NF.
    - UNF to 1NF define PK & remove repeating group.
    - 1NF to 2NF remove partial dependency.
    - 2NF to 3NF remove transitive dependency.