Dr Simon D’Alfonso

THE UNIVERSITY OF MELBOURNE

INFO90002 Database Systems & Information Modelling

Week 05

Normalisation

This lecture

- Normalisation
  - 1st Normal Form (1NF)
  - 2nd Normal Form (2NF)
  - 3rd Normal Form (3NF)
  - Boyce Codd Normal Form (BCNF)

Normalisation is...

- A theoretical foundation for the relational model
- Application of a series of rules that improve db design
- Last phase of logical design
- Normalisation rules are designed to
  - Prevent data redundancy
  - Prevent update anomalies
- Normalisation is biased towards optimizing *Updates*
  - i.e. it assumes non-key columns are updated frequently
  - thus it can slow down *Selects* because of the need to join tables
  - thus designers sometimes “de-normalise” some tables to improve performance
- Every attribute must be functionally dependent on “The key, the whole key, and nothing but the key.”

Normalisation: a process

- Normal Form
  - the state of a relation resulting from applying rules about the functional dependency of some attributes upon others
- Design progresses in stages from 1NF via 3NF, possibly as far as 6NF
  - in this course we only cover 1NF to 3.5NF (Boyce-Codd NF)
  - this is often sufficient in practice

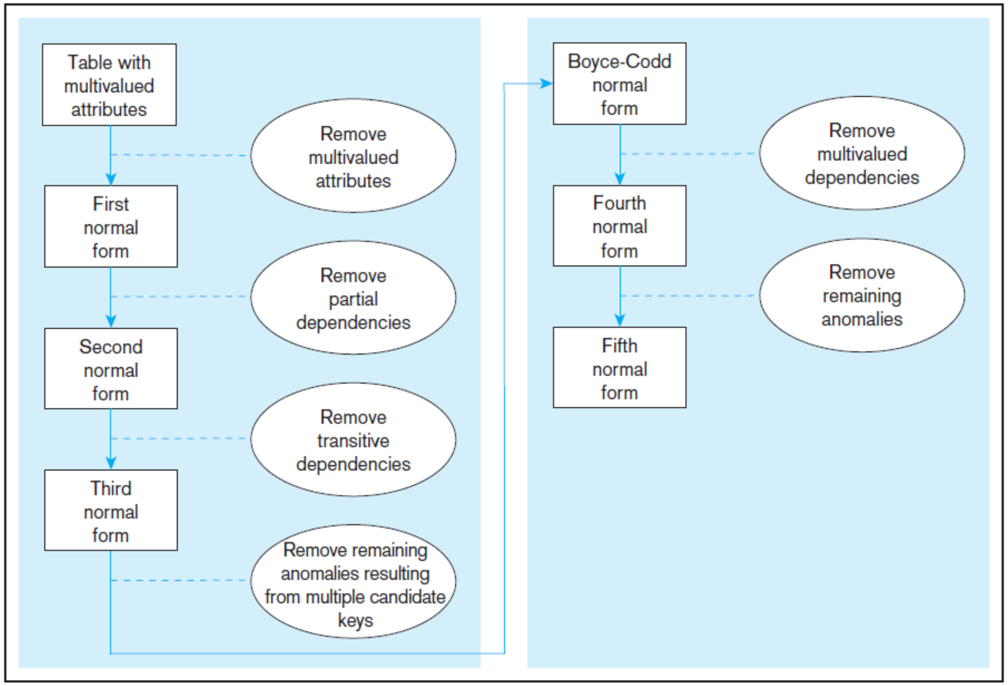


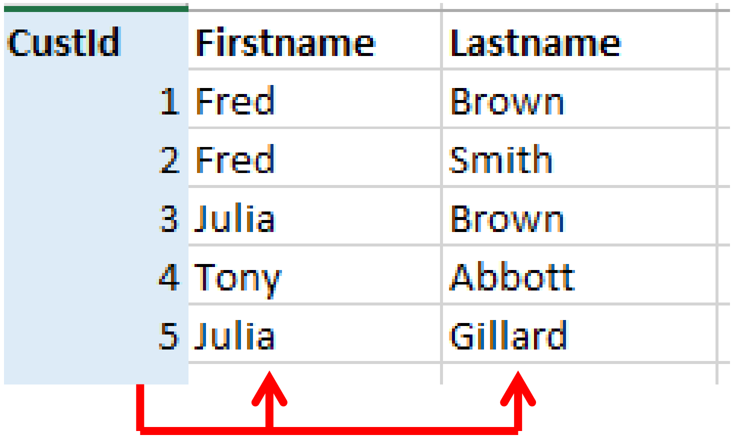
Table with multivalued attributes
Remove multivalued attributes
First normal form
Remove partial dependencies
Second normal form
Remove transitive dependencies
Third normal form
Remove remaining anomalies resulting from multiple candidate keys
Boyce-Codd normal form
Remove multivalued dependencies
Fourth normal form
Remove remaining anomalies
Fifth normal form

flowchart from Hoffer et al. *Modern Database Management*

Definitions

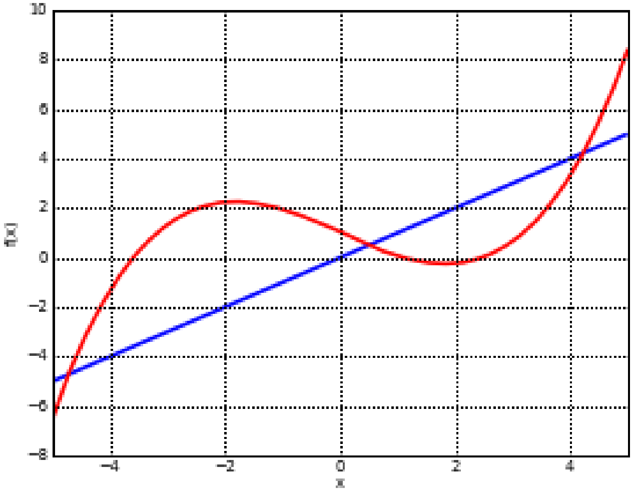
- Functional dependency
  - a set of attributes X *determines* another set of attributes Y iff each value of X is associated with only one value of Y
  - written X → Y (similar to y = f(x))
- Determinants
  - the attribute(s) on the *left hand side* of the arrow
- Key and Non-Key attributes
  - each attribute is either part of the primary key or it is not
- Partial functional dependency
  - a functional dependency of one or more non-key attributes upon *part (but not all)* of the primary key
- Transitive dependency
  - a functional dependency between 2 (or more) non-key attributes

Functions and functional dependence

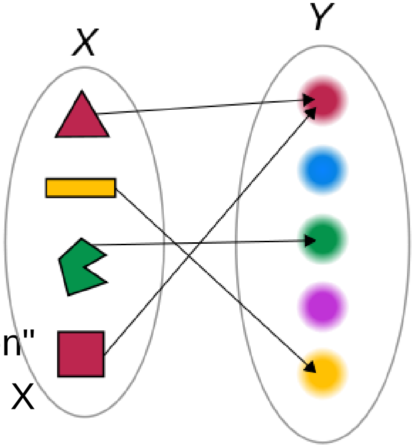


CustId	Firstname	Lastname
1	Fred	Brown
2	Fred	Smith
3	Julia	Brown
4	Tony	Abbott
5	Julia	Gillard

CustId determines Firstname and Lastname
neither Firstname nor Lastname determine anything



y = f(x)
for each x there is no more than one y



the "colour-of-the-shape function"
X determines Y, but Y does not determine X

Functional dependency

- A relationship between the attributes of one relation
  - one or more attributes determine the value of another attribute
- A primary key must functionally determine all non-key attributes of an entity
  - stock exchange company code → company name and details
    - If we know the company code, we know the value of company name
    - (http://www.asx.com.au/prices/understanding-asx-ticker-codes.htm)
  - studentId, subjectCode → dateCompleted, resultObtained
    - for any combination of studentId and subjectCode,
      there is only one dateCompleted and one resultObtained
- Determinant
  - only primary keys should be determinants

Hierarchy of Normal Forms

- 1NF
  - Any *multivalued* attributes and repeating groups have been removed. There is a SINGLE value (possibly null) at the intersection of each row and column of the table.
- 2NF
  - Any *partial* functional dependencies have been removed (ie all non-key attributes are identified by the WHOLE key)
- 3NF
  - Any *transitive* dependencies have been removed (ie non-key attributes are identified by ONLY the PRIMARY key)
- BCNF
  - Any remaining anomalies that result from functional dependencies have been removed (ie because there was more than one candidate primary key for the same non-keys)

Normalisation example

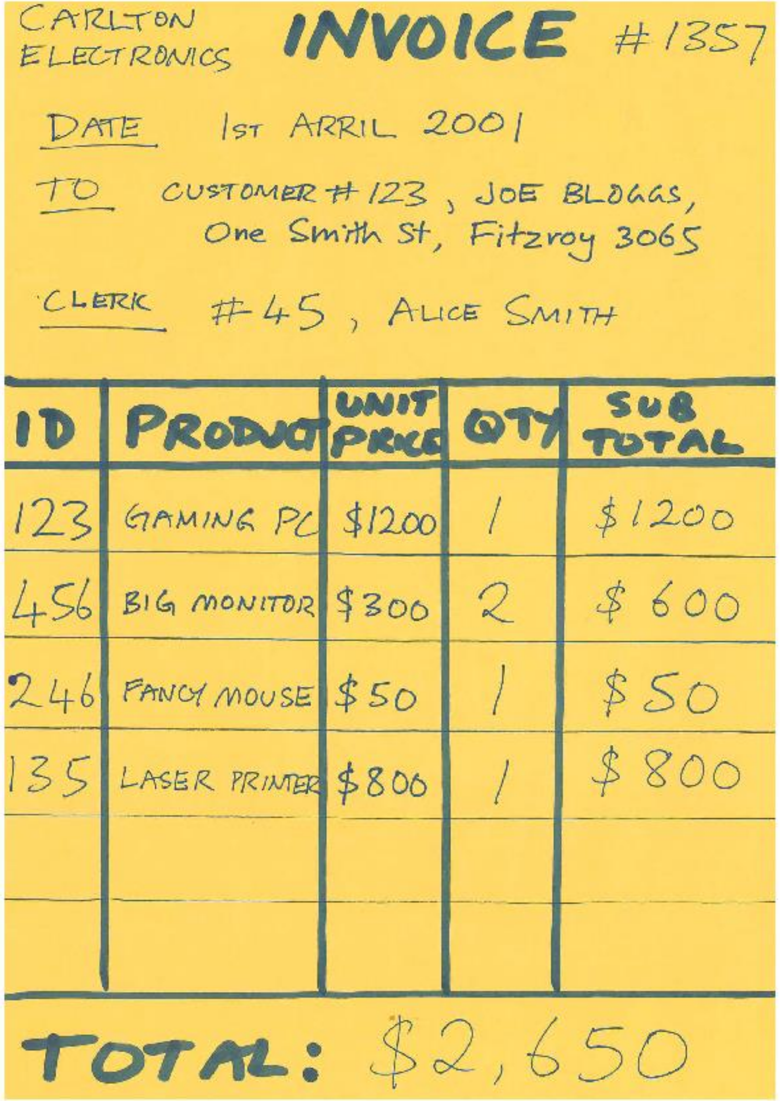
- To begin:
  - Write in relational notation
    List all the attributes that must be recorded, as one big relation
  - Don’t include anything that you can derive (e.g. total sales)
  - Use parentheses “()” to indicate repeating groups
  - Underline the primary key

  - INVOICE (InvoiceNo, Date, CustomerNo, CustomerName, CustomerAddress, ClerkNo, ClerkName, (ProductNo, ProductDesc, UnitPrice, Qty))

First normal form (1NF)

- Separate repeating groups into new tables
  - if an attribute is multi-valued,
  - or a group of attributes is repeated several times for one entity,
  - create a new table containing the repeating data

example: paper invoices typically include a repeating group which represents each product sold in that sale



CARLTON ELECTRONICS

INVOICE #1357

DATE 1ST ARRIL 2001

TO CUSTOMER #123, JOE BLOGGS,
One Smith St, Fitzroy 3065

CLERK #45, ALICE SMITH

ID	PRODUCT	UNIT PRICE	QTY	SUB TOTAL
123	GAMING PC	$1200	1	$1200
456	BIG MONITOR	$300	2	$600
246	FANCY MOUSE	$50	1	$50
135	LASER PRINTER	$800	1	$800

TOTAL: $2,650

Our example (1NF)

- INVOICE (InvoiceNo, Date, CustomerNo, CustomerName, CustomerAddress, ClerkNo, ClerkName, (ProductNo, ProductDesc, UnitPrice, Qty))

- Repeating group is (ProductNo, ProductDesc, UnitPrice, Qty)
- The new table is as follows:
  - LINEITEM (*InvoiceNo*, ProductNo, ProductDesc, UnitPrice, Qty)
    (foreign keys get italics or dotted underline)
- The repeating fields will be removed from the original relation, leaving the following.
  - INVOICE (InvoiceNo, Date, CustomerNo, CustomerName, CustomerAddress, ClerkNo, ClerkName)
- These two relations together comprise a database in 1NF

Second normal form (2NF)

- Relevant only for relations whose primary key is a COMPOSITE key
- Remove partial functional dependencies.
  - an attribute is dependent on only *part of* the primary key
  - i.e. dependent on *one* of the columns in a composite primary key
- Create a separate table, containing the functionally dependent data, and the part of the key on which it depends.

Our example (2NF)

- INVOICE (InvoiceNo, Date, CustomerNo, CustomerName, CustomerAddress, ClerkNo, ClerkName)
- LINEITEM (*InvoiceNo*, ProductNo, ProductDesc, UnitPrice, Qty)

- INVOICE doesn’t have a composite key, so it is already in 2NF
- LINEITEM is not in 2NF, because ProductDesc and UnitPrice are determined by ProductNo, which is *part of* the primary key. We need to create a new table from these fields

- The new table will contain the following fields:
  - PRODUCT (ProductNo, ProductDesc, UnitPrice)

Our example (2NF)

- All of these fields except the primary key will be removed from the original table. The primary key will be left in the original table to allow linking of data:
  - LINEITEM (*InvoiceNo*, *ProductNo*, Qty)

- Our database is now in second normal form:
  - INVOICE (InvoiceNo, Date, CustomerNo, CustomerName, CustomerAddress, ClerkNo, ClerkName)
  - LINEITEM (*InvoiceNo*, *ProductNo*, Qty)
  - PRODUCT (ProductNo, ProductDesc, UnitPrice)

Third normal form (3NF)

- Remove *transitive dependencies*
  - a functional dependency where one non-key attribute is functionally dependent on another non-key attribute
  - thus its value is only indirectly determined by the primary key.

- Create a separate table containing the determinant attribute and the fields that are functionally dependent on it.
  - keep a copy of the key in the original table.

Our example 3NF

- INVOICE (InvoiceNo, Date, CustomerNo, CustomerName, CustomerAddress, ClerkNo, ClerkName)
- LINEITEM (*InvoiceNo*, *ProductNo*, Qty)
- PRODUCT (ProductNo, ProductDesc, UnitPrice)

- LINEITEM and PRODUCT are in 3NF as there are not any transitive dependencies
  - All the attributes only depend on the key
- INVOICE is not in 3NF
  - Transitive dependencies exist
    - CustomerName and CustomerAddress depend on CustomerNo
    - ClerkName depends on ClerkNo

Our example 3NF

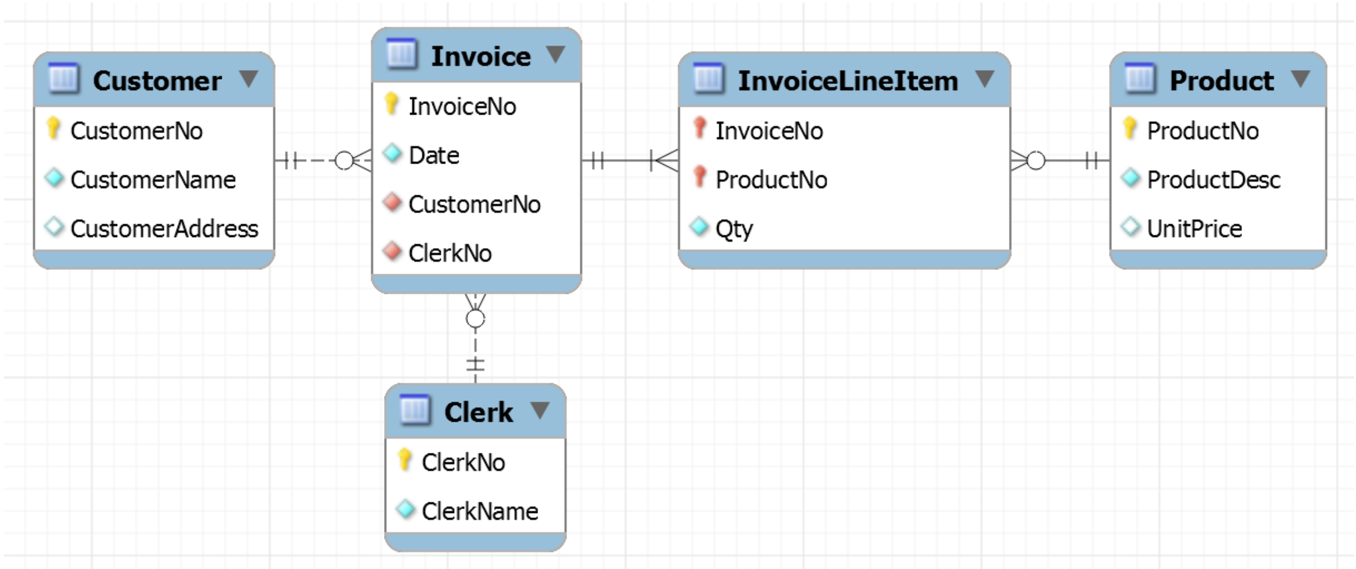
- The new tables will be:
  - CUSTOMER (CustomerNo, CustomerName, CustomerAddress)
  - CLERK (ClerkNo, ClerkName)
- All of these fields except the primary keys will be removed from the original table. The PKs will be left in the original table (as FKs) to allow linking of data, as follows:
  - INVOICE (InvoiceNo, Date, *CustomerNo*, *ClerkNo*)
- Together with the unchanged tables below, these tables make up the database in third normal form.
  - LINEITEM (*InvoiceNo*, *ProductNo*, Qty)
  - PRODUCT (ProductNo, ProductDesc, UnitPrice)

What if we did *not* normalize ?

- Repetition of data
  - Individual facts are stored many times.
- Update anomalies
  - To change a fact, we must change it many times.
- Deletion anomalies
  - Information about entity A is stored inside entity B.
    If we delete all B rows, we lose our record of A.
- Insertion anomalies
  - To record a new A, we must insert a B.

Our example schema in 3NF

- CUSTOMER (CustomerNo, CustomerName, CustomerAddress)
- CLERK (ClerkNo, ClerkName)
- PRODUCT (ProductNo, ProductDesc, UnitPrice)
- INVOICE (InvoiceNo, Date, *CustomerNo, ClerkNo*)
- LINEITEM (*InvoiceNo, ProductNo*, Qty)



Beyond 3NF

- It is often sufficient to stop normalizing at 3NF
- BCNF, 4NF, 5NF and even 6NF have been defined
  - we’ll touch on BCNF
  - it is suggested that you read about the others, see
  - http://en.wikipedia.org/wiki/Database_normalization#Normal_forms
  - but we will not assess beyond 3NF

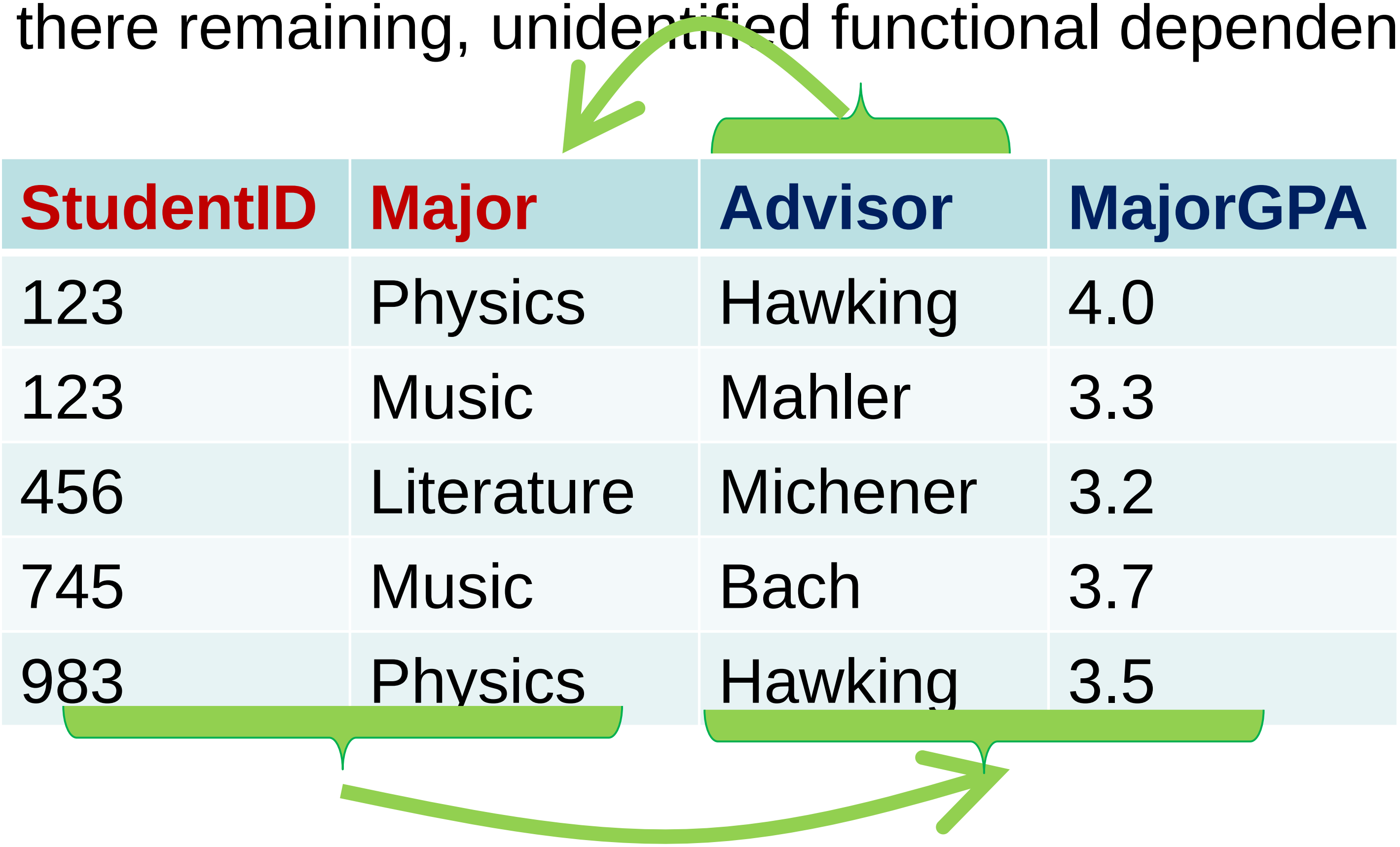
Boyce-Codd normal form (BCNF)

- Can arise when …
  - not every determinant in the relation is a candidate key

- A simple test is to check whether …
  - "Every non-key attribute must provide a fact about the key, the whole key, and nothing but the key.”

BCNF example

- Consider the following relation
  - Student (StudentID, Major, Advisor, MajorGPA)
- It is in 3NF … but …
- Are there remaining, unidentified functional dependencies?



StudentID	Major	Advisor	MajorGPA
123	Physics	Hawking	4.0
123	Music	Mahler	3.3
456	Literature	Michener	3.2
745	Music	Bach	3.7
983	Physics	Hawking	3.5

Problems with this relation

- Update anomaly
  - – Hawking is replaced by Einstein
- Insertion anomaly
  - – Babbage advises comp. sci., but has no students
- Deletion anomaly
  - – e.g. delete the only student with a particular advisor (456)

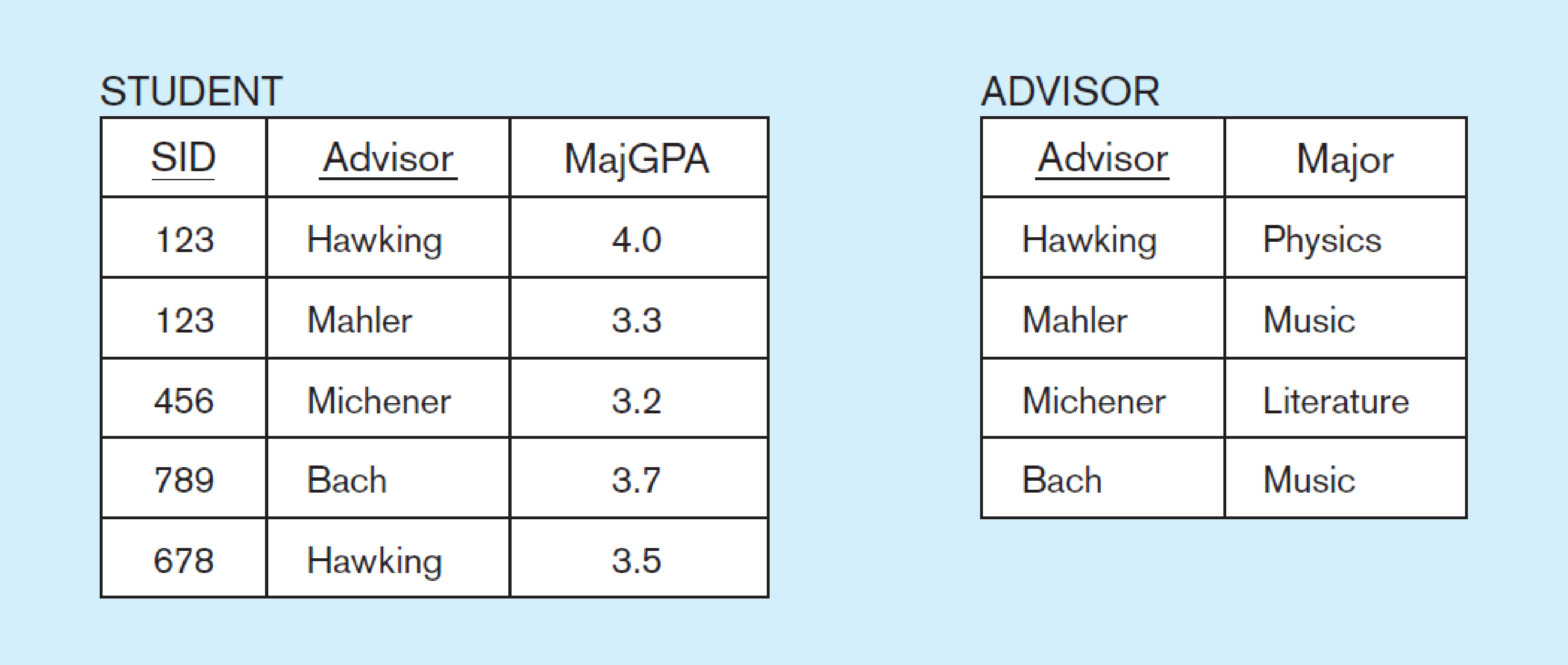
StudentID	Major	Advisor	MajorGPA
123	Physics	Hawking	4.0
123	Music	Mahler	3.3
456	Literature	Michener	3.2
745	Music	Bach	3.7
983	Physics	Hawking	3.5

Normalise to BCNF

- Modify the relation so that
  - the determinant in the relation that is not part of the key becomes a component of the primary key of the revised relation
  - the attribute that is functionally dependant on that determinant becomes a non key attribute
- The relation becomes
  - Student (StudentID, Advisor, Major, MajorGPA)
- Can this be normalized further?
  - Its not in 2NF (partial functional dependency) – so normalise it
  - Student (StudentID, *Advisor*, MajorGPA)
  - Advisor (Advisor, Major)

Now in BCNF

- Student (StudentID, *Advisor*, MajorGPA)
- Advisor (Advisor, Major)



STUDENT

SID	Advisor	MajGPA
123	Hawking	4.0
123	Mahler	3.3
456	Michener	3.2
789	Bach	3.7
678	Hawking	3.5

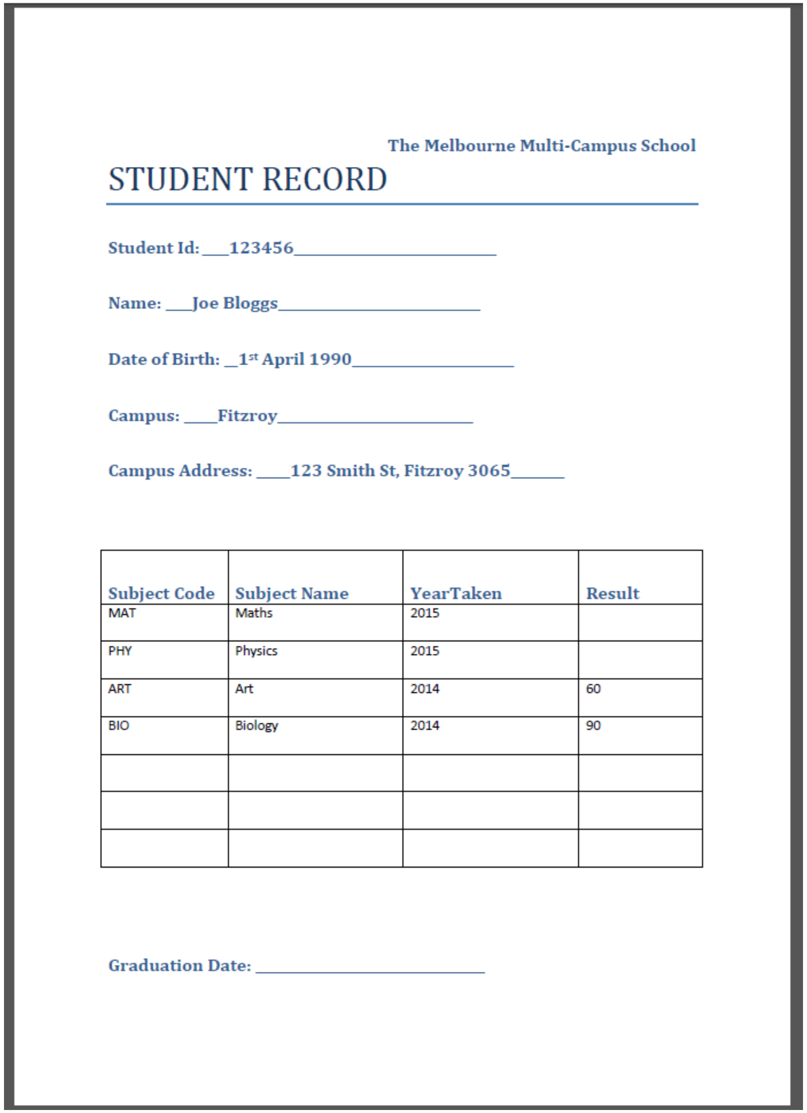
ADVISOR

Advisor	Major
Hawking	Physics
Mahler	Music
Michener	Literature
Bach	Music

Normalisation Exercise

Now it is your turn!

Normalization example



The Melbourne Multi-Campus School

STUDENT RECORD

Student Id: __123456_____________________

Name: ___Joe Bloggs_____________________

Date of Birth: _1st April 1990________________

Campus: ___Fitzroy_____________________

Campus Address: ___123 Smith St, Fitzroy 3065______

Subject Code	Subject Name	YearTaken	Result
MAT	Maths	2015	
PHY	Physics	2015	
ART	Art	2014	60
BIO	Biology	2014	90

Graduation Date: ________________________

- The *Melbourne Multi-Campus School* has been using a paper-based information system.
- Student data is recorded on these sheets of paper.
- Now they are going to computerize.
- How shall we store the data in a relational database?