

Normalisation

- A technique for producing a set of relations with desirable properties, given the data requirements of the applications

□ Outline

- Data redundancy and anomalies
- Spurious information
- Functional dependencies
- Normalisation
 - 1st normal form
 - 2nd normal form
 - 3rd normal form
 - Boyce-Codd normal form
 - 4th normal form
 - 5th normal form

Data Redundancy and Anomalies

- Beware of keeping multiple versions of information

Emp-Dept

<u>NI#</u>	Name	DateOfBirth	Dept#	Dname	Manager
21	AA	-	5	CS	91
22	BB	-	5	CS	91
23	CC	-	6	TS	93
24	DD	-	7	PSV	94
25	EE	-	7	PSV	94

- **Insertion**
 - a) How do we insert a new department with no employees yet? (keys?)
 - b) Entering employees is difficult as department information must be entered correctly
- **Deletion**

What happens when we delete CC's data - do we lose department 6!
- **Modification**

If we change the manager of department 5, we must change it for tuples with Dept# = 5

Spurious Information

- Avoid breaking up relations in such a way that spurious information is created

PROJECT

NI#	Name	ProjName	ProjLocation
123	XX	Accounts	London
123	XX	Analysis	Paris
124	YY	PI	London

may be broken into:

NI#	Name	ProjName
123	XX	Accounts
123	XX	Analysis
124	YY	PI

NI#	ProjLocation
123	London
123	Paris
124	London

joining them back together, we get NEW TUPLES!

NI#	Name	ProjName	ProjLocation
123	XX	Accounts	Paris
123	XX	Analysis	London

Functional Dependencies [1]

- Formal concepts that may be used to exhibit “goodness” and “badness” of individual relational schemas, and describe relationships between attributes

Examples of Functional Dependency

- Name, DateOfBirth, Dept# all depend on NI#
- Dname and Manager depend on Dept#
- ProjLocation depends on ProjName

Functional Dependencies [2]

- An attribute, X, of a relation is **functionally dependent** on attributes A, B, ..., N if the same values of A, ..., N are always associated with the same value of X

$\{A, \dots, N\} \rightarrow X$

- A, ..., N is called the **determinant** of the functional dependency
- This is a property of the **meaning** of the data, not a property that emerges from the values of the data
i.e. if you happen to have no two employees with the same name, you may indeed be able to infer age from name but this would not constitute genuine dependence

$NI\# \rightarrow \{Name, DateOfBirth\}$

$\{NI\#, Pnum\} \rightarrow Hours$

Full Functional Dependency

- X **fully depends** on A , ..., N if it is not dependent on any subset of A , ..., N; otherwise we talk of **partial dependency**

e.g. age is dependent on NI# and Name but only fully dependent

Normalisation


Process

- taking a set of relations and decomposing them into more relations satisfying some criteria

Decomposition

- essentially a series of projections so that the original data can be reconstituted using joins

Normal form

- form of the relations which satisfy the criteria
 - number of these of increasing stringency
- 

First Normal Form [1]

- A relation is in **first normal form** (1NF) if all values are **atomic**, i.e. single values - small strings and numbers
- Identify and remove repeating groups (multi-valued attributes)

First Normal Form [2]

DEPARTMENT

Dnumber	Dname	Locations
5	C.S.	{Paris, London}

Two ways of **normalising** this:

- Have a tuple for each location of each department

Dnumber	Dname	Locations
5	C.S.	Paris
5	C.S.	London

- Have a separate relation for (Dnumber, Locations) pairs

Dnumber	Dname
5	C.S.

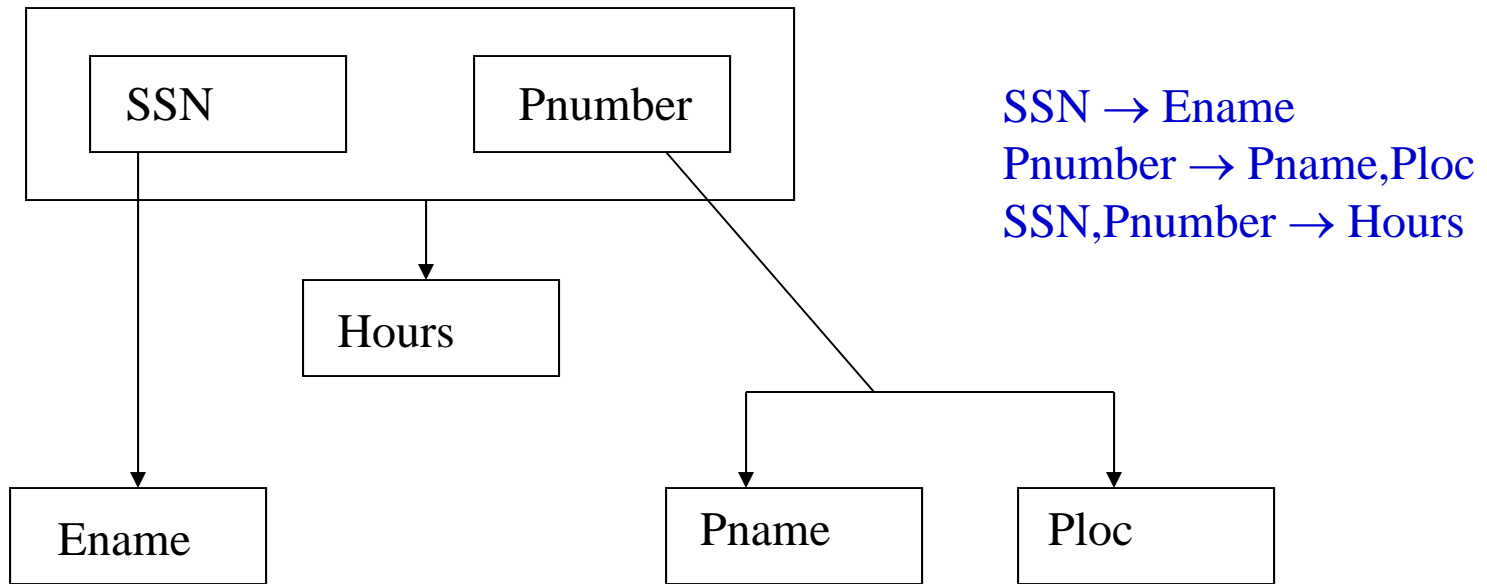
Dnumber	Locations
5	Paris
5	London

- The latter is better as it avoids redundancy

Second Normal Form

- By the definition of the primary key, every other attribute is functionally dependent on it
- If all the other attributes are **fully** functionally dependent then the relation is in **Second Normal Form (2NF)**
- Clearly, **any relation with a single primary key will be 2NF**
- If there are two primary key attributes, A and B, then each other attribute is either
 - dependent on A alone
 - dependent on B alone
 - or dependent on both
- 2NF consists of creating a separate relation for each of the three cases

EMP_PROJ(SSN, Pnumber, Hours, Ename, Pname, Ploc)



Decomposition into three 2NF relations:

Work(SSN, Pnumber, Hours)

EMP(SSN, Ename)

Project(Pnumber, Pname, Ploc)

Third Normal Form

- **Third Normal Form** eliminates **transitive dependencies**, i.e. those dependencies which hold only because of some intermediary
- An attribute is transitively dependent on the primary key if there is some other attribute which it is dependent on and which is, in turn, dependent on the key

<u>NI#</u>	Dnumber	Dname
1234	5	C.S.
1234	5	C.S.

Dname is dependent on NI# as required by 2NF, but only because it is dependent on Dnumber which is, in turn, dependent on NI#

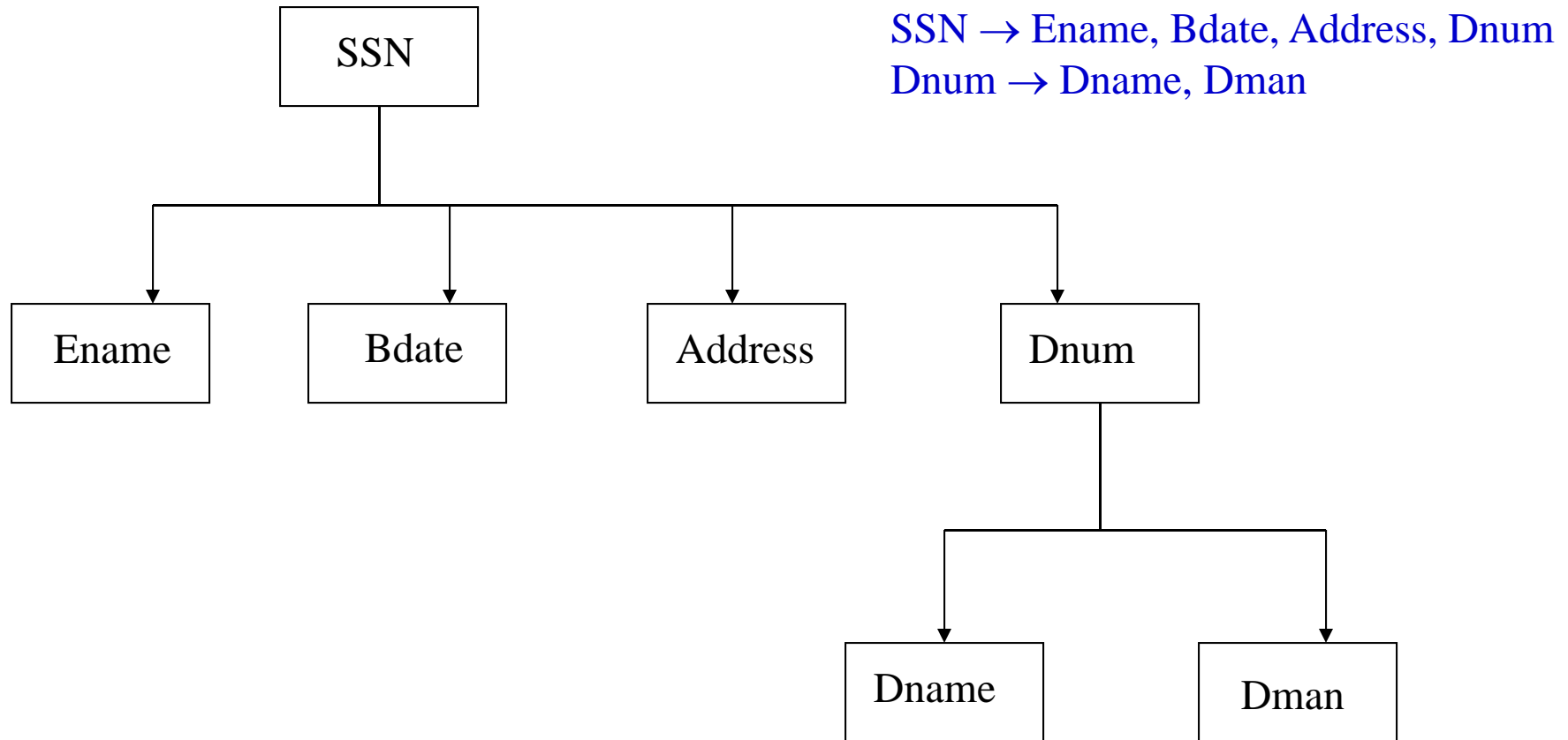
Normalising this would create:

<u>NI#</u>	Dnumber
1234	5
1234	5

<u>Dnumber</u>	Dname
5	C.S.

- Non-3NF relations are likely to hold redundant information
- A relation is in 3NF if for any pair of attribute A & B such that $A \rightarrow B$, there is no attribute such that $A \rightarrow X$ and $X \rightarrow B$

EMP_DEPT(SSN, Ename, Bdate, Address, Dnum, Dname, Dman)



Decomposition into two 3NF relations:

EMPLOYEE(SSN, Ename, Bdate, Address, Dnum)

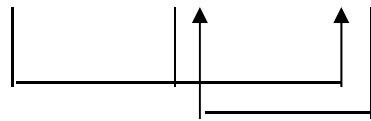
DEPT(Dnum, Dname, Dman)

Boyce-Codd Normal Form (BCNF) [1]

- 2NF + 3NF: no partial dependencies + no transitive dependencies
 - What about other candidate keys (if they exist)?
- A relation is in BCNF if and only if every determinant is a candidate key

Boyce-Codd Normal Form (BCNF) [2]

TEACH(Student, Course, Instructor)



Three possible decompositions

R1(Student, Instructor) and R2(Student, Course)

R1(Course, Instructor) and R2(Course, Student)

R1(Instructor, Course) and R2(Instructor, Student)

- Last is the best as it does not generate spurious tuples after join
- Sometimes better to stop at 3NF to not lose functional dependencies

Fourth Normal Form [1]

- Multi-valued dependencies (MVD)

<u>Branch</u>	<u>Staff</u>	<u>Client</u>
B3	Ann	Aline
B3	David	Aline
B3	Ann	Mike
B3	David	Mike

Branch \twoheadrightarrow Staff

Branch \twoheadrightarrow Client

- A MVD $A \twoheadrightarrow B$ is **trivial** if either (a) B is a subset of A , or (b) the union of A and B is the relation; otherwise it is said to be **non-trivial**
- In our case, both MVDs are non-trivial

Fourth Normal Form [2]

- A relation is in 4NF if it is in BCNF and contains no non-trivial MVDs
- This means decomposing our relation into

<u>Branch</u>	<u>Staff</u>
B3	Ann
B3	David

<u>Branch</u>	<u>Client</u>
B3	Aline
B3	Mike

Fifth Normal Form

- **Lossless-join Decomposition:** A property of a decomposition that ensures that no spurious rows are generated when relations are reunited through a natural join operation
- A relation is in 5NF if it has no join dependency

PIS(PropertyID, ItemDescription, SupplierID)

- **Join Dependency:** suppliers provide certain items to particular properties

decompose into three 5NF relations

R1(PropertyID, ItemDescription)

R2(ItemDescription, SupplierID)

R3(PropertyID, SupplierID)

Join all three only does not create spurious information

Example: Medical Records

We have:

- patients with unique NI#, each having a name and a GP
- each GP has an id.number, name & address
- a hospital appointment connects a patient, a date, a hospital and a consultant
- a consultant has a phone number and visits one hospital on any given day
- a hospital has an address

The Universal Relation:

U(NI# , Appdate, Apptime, PTname, GP# , GPaddress, GPname, Cname, Cphone, Hosp, HospAddress)

Example: Functional Dependencies

From a patient's name, we can determine the patient's name and GP

NI# \rightarrow {PTname, GP#}

From a GP's ID we determine the GP's name and address

GP# \rightarrow {GPname, GPaddress}

From a particular patient on a particular day we can determine the consultant information, the time of the appointment and which was the hospital

NI#, AppDate \rightarrow {Cname, AppTime, Hosp}

Each consultant has one phone number

Cname \rightarrow Cphone

Each hospital has only one address

Hosp \rightarrow HospAddress

Example: Normalisation

Starting with the Universal Relation we find all kinds of redundancy

So moving to 2NF split off those attributes only dependent on part of the primary key

Patient (NI# , PTname, GP# , GPaddress, GPname)

Appt (NI# , AppDate, AppTime, Cname, Cphone, Hosp, HospAddress)

Patient is 2NF but not 3NF since $NI\# \rightarrow GP\# \rightarrow \{GPaddress, GPname\}$

So split off the GP information

Patient2 (NI# , PTname, GP#)

GP (GP# , GPaddress, GPname)

Similarly, Appt becomes

App (NI# , AppDate, AppTime, Cname, Hosp)

Con (Cname, Cphone)

Hospital (Hosp, HospAddress)

Example: Normalisation

Final result

Patient2 (NI# , PTname, GP#)

GP (GP# , GPaddress, GPname)

App (NI# , AppDate, AppTime, Cname, Hosp)

Con (Cname, Cphone)

Hospital (Hosp, HospAddress)