Normalization II

1 Construction of the Relational Data model

- Bottom-up approach: Normalization
  - start with the initial tables and attributes (from the ER model)
  - analyse the relationships among the attributes
  - re-design the tables and attributes in a ”better” way:
    - decompose the tables into more tables (schema refinement)
    - ensure entity and referential integrity
- Purpose of normalization
  - every relation represents a ”real world” entity
  - single-valued columns
  - avoid redundancy (i.e. repetitions)
    - Minimise the amount of space required
    - Simplify maintenance of the database
  - Data can be updated correctly to avoid update anomalies

2 Data update anomalies

Staff Branch

staffNo	sName	position	salary	branchNo	bAddress
SL21	John White	Manager	30000	B005	22 Deer Rd, London
SG37	Ann Beech	Assistant	12000	B003	163 Main St, Glasgow
SG14	David Ford	Supervisor	18000	B003	163 Main St, Glasgow
SA9	Mary Howe	Assistant	9000	B007	16 Argyll St, Aberdeen
SG5	Susan Brand	Manager	24000	B003	163 Main St, Glasgow
SL41	Julie Lee	Assistant	9000	B005	22 Deer Rd, London

- Modification anomaly:
  - we want to change the address of branch B003
  - we must update it in many places (due to redundancy)
  - problem: if we do not update one of them ⇒ inconsistent data
- Deletion anomaly:
  - we want to delete staff with staffNo SA9 from the database
  - this is the last staff member in branch B007
  - Problem: we loose the details of this branch ⇒ incomplete data
- Insertion anomaly
  - We want to add a new branch, which has no staff yet
    ⇒ we must add Null into the attributes related to staff
  - staffNo is aa private key ⇒ violation of entity integrity

3 Functional data dependencies

The fundamentals of normalization theory:

- Functional data dependency:
  - Let A and B be two sets of attributes; we say that

    "B is functionally dependent on A" (denoted $A \rightarrow B$)

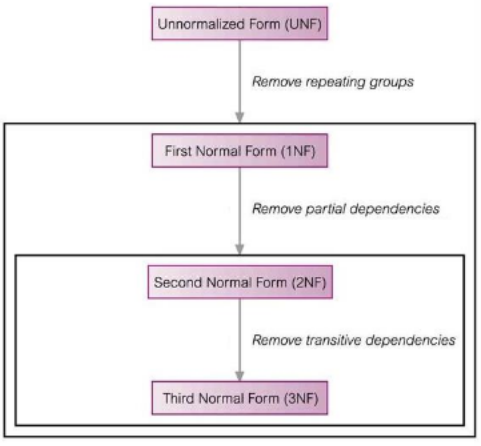
    if each value of A determines exactly one value of B
- In a functional data dependency ($A \rightarrow B$):
  - determinant: the set of all attributes on the left hand side (i.e. A)
  - dependent: the set of all attributes on the right hand side (i.e. B)
- By the definition of relational keys:
  - a candidate key is a minimal set of attributes, which functionally determine all attributes in a relation
  - among all candidate keys, we choose (any) one of them to serve as the primary key

4 Functional dependencies

- **Full** functional dependency $A \rightarrow B$
  - B is functionally dependent on A
  - B is not functionally dependent on any proper subset of A
  - Example: staffNo→ sName
- Partial functional dependency $A \rightarrow B$:
  - B is functionally dependent on A
  - B remains functionally dependent on at least one proper subset of A
  - Example: staffNo,sName → branchNo
- Transitive functional dependency:
  - Functional dependencies $A \rightarrow B$ and $B \rightarrow C$
  - Then the functional dependency $A \rightarrow C$ is called **transitive**

5 Normalization process

- Normalization is a multi-stage process
  - The result of each stage is called a Normal form
  - at each stage: check whether specific criteria are satisfied. If not: re-organise the data
- We study the first 3 normal forms which are most important for practical applications



6 First Normal Form (1NF)

- Repeating group:
  - an attribute (or group of attributes) that occurs with multiple values for a single occurence of the primary key
- A table is in the un-normalised form (UNF):
  - when it contains one or more repeating groups
  - This does not conform with the definition of a relation
- A table is in the First Normal Form (1NF) if it has:
  - no repeating groups (every cell has one value)
  - No identical rows

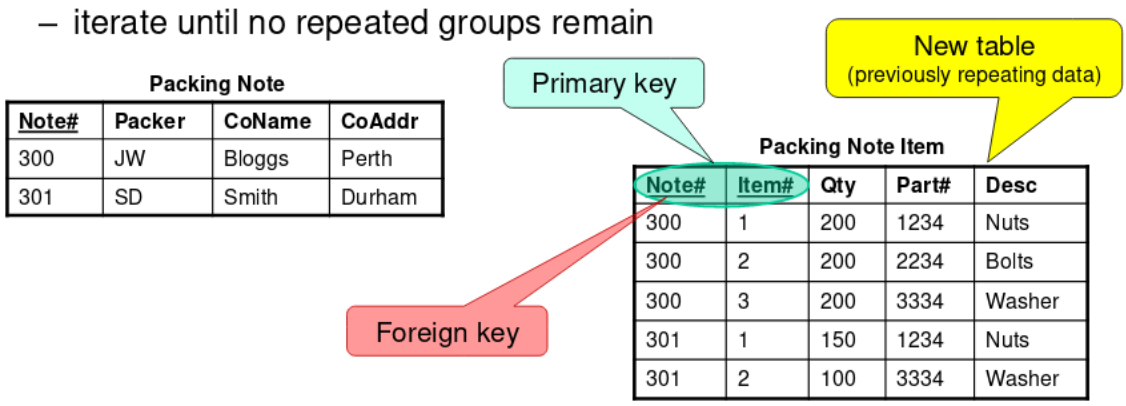
How to bring a table into 1NF

- One alternative would be:
  - Repeat the appropriate columns horizontally
  - Problem: a table must have a fixed number of columns
  - We need a fixed (large) upper limit on the number of repetitions
  - Many of these new columns would be empty ⇒ waste of space
  - Complicated querying: we need to search many columns to find the right item
- Another alternative would be:
  - For every multi valued attribute: one long string containing the whole list of items
  - the same problems: long strings, difficult querying
- First method: one table solution
  - enter appropriate data into the empty columns (by repeating data)

Packing note table

Note #	Packer	CoName	CoAddr	Item#	Qty	Part#	Desc
300	JW	Bloggs	Perth	1	200	1234	Nuts
300	JW	Bloggs	Perth	2	200	2234	Bolts
300	JW	Bloggs	Perth	3	200	3334	Washer
301	SD	Smith	Durham	1	150	1234	Nuts
301	SD	Smith	Durham	2	100	3334	Washer

  - The resulting table is in 1NF, but still: we introduced a lot of redundancy (by repeating data)
- Second method: two tables solution
  - place the repeating data in a separate relation
  - in the new relation place a copy of the original primary key
  - this key now becomes a foreign key (to refer to the original relation)
  - iterate until no repeated groups remain



- The resulting tables are in 1NF with much less redundancy than before

7 Second normal form (2NF)

- A table is in the Second Normal Form (2NF) if:
  - it is in 1NF and
  - there are no partial functional dependencies i.e. every non key attribute is dependent on the whole primary key
- Non-key attributes: all attributes that are not a part of the primary key
- 2NF applies to relations with composite keys
- When the primary key only has one attribute (single key) if the table is in 1NF $\Rightarrow$ it is also in 2NF
- How to bring a table into 2NF
  - Remove the partially dependent attributes
  - place them in a new relation, along with the copy of their determinant