**Chapter 13 – Objectives**

* How functional dependencies can be used to group attributes into relations that are in a known normal form.
* How to undertake process of normalization.
* How to identify most commonly used normal forms, namely 1NF, 2NF, 3NF, and Boyce–Codd normal form (BCNF).

**Normalization**

**A technique for producing a set of relations with desirable properties, given the data requirements of an enterprise.**

The purpose of normalization is to identify a suitable set of relations that support the data requirements of an enterprise. The characteristics of a suitable set of relations include the following:

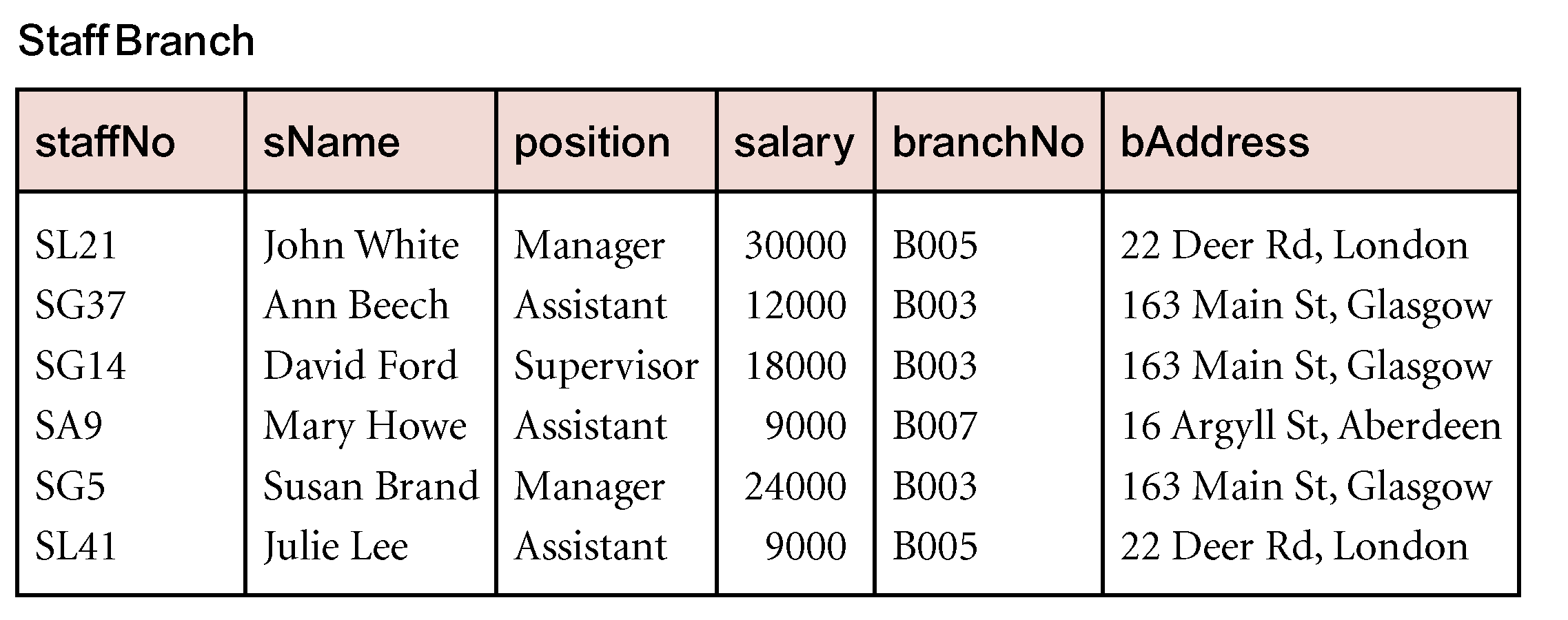
* the *minimal* number of attributes necessary to support the data requirements of the enterprise;
* attributes with a close logical relationship (described as functional dependency) are found in the same relation;
* *minimal* redundancy with each attribute represented only once with the important exception of attributes that form all or part of foreign keys, which are essential for the joining of related relations.

The benefits of using a database that has a suitable set of relations is that the database will be easier for the user to access and maintain the data, and take up minimal storage space on the computer.

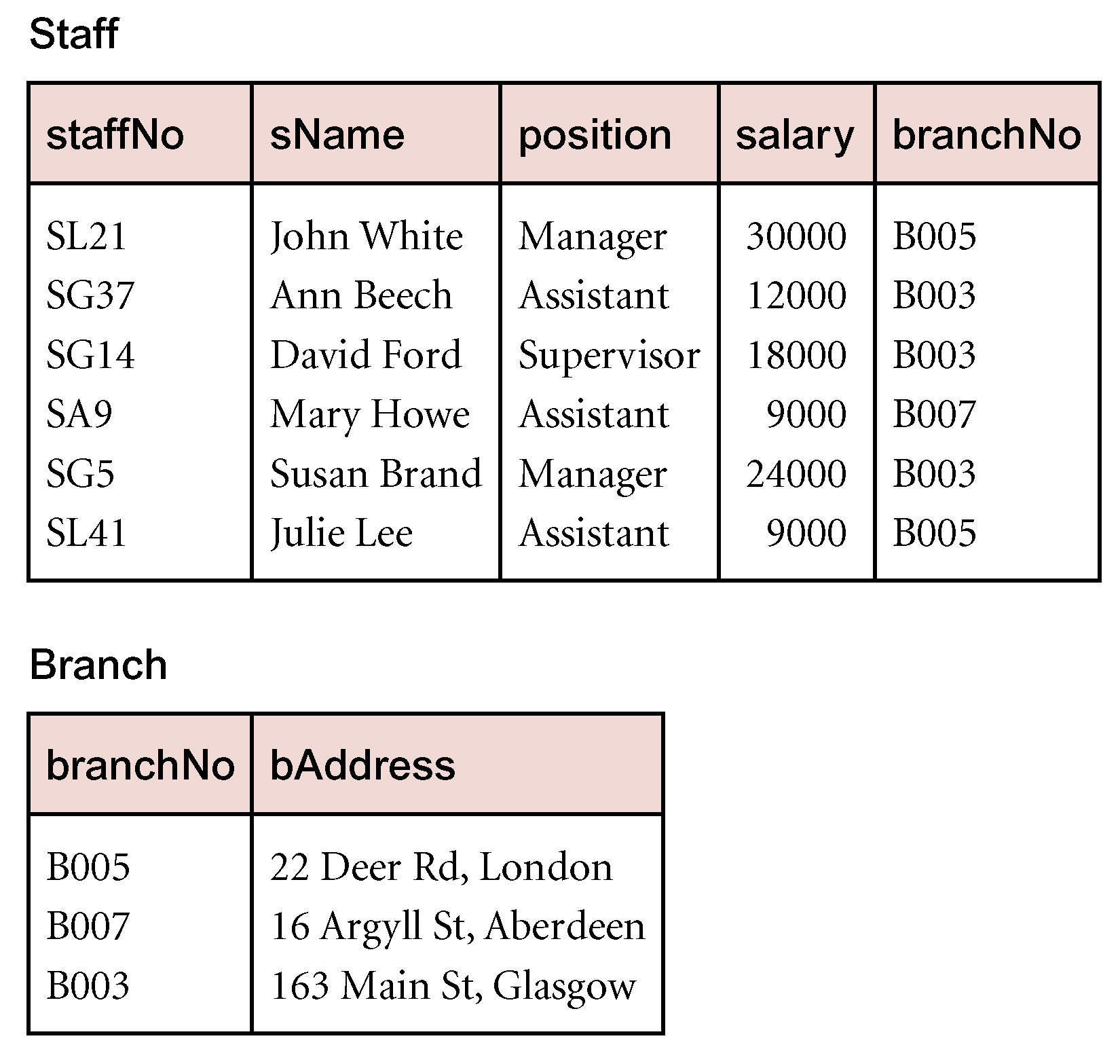
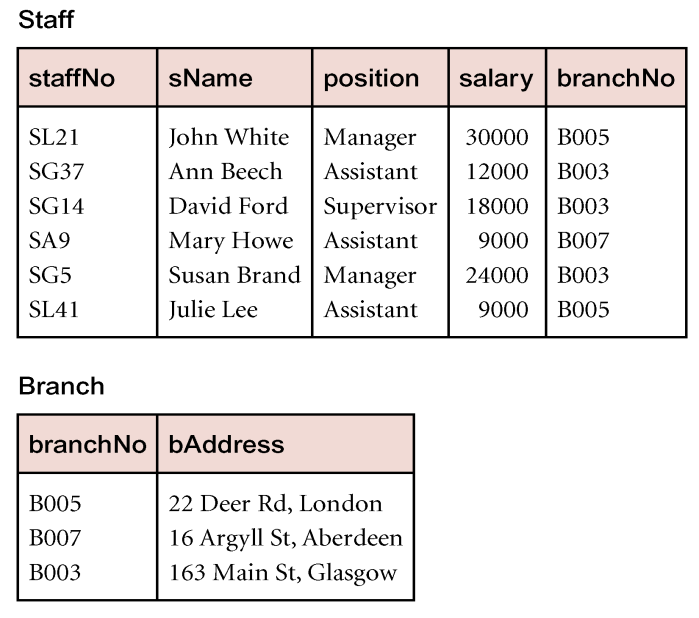
* Four most commonly used normal forms are first (1NF), second (2NF) and third (3NF) normal forms, and Boyce–Codd normal form (BCNF).
* Based on functional dependencies among the attributes of a relation.
* A relation can be normalized to a specific form to prevent possible occurrence of update anomalies.

**Data Redundancy**

* Major aim of relational database design is to group attributes into relations to minimize data redundancy and reduce file storage space required by base relations.
* Problems associated with data redundancy are illustrated by comparing the following Staff and Branch relations with the StaffBranch relation.

****

* StaffBranch relation has redundant data: details of a branch are repeated for every member of staff.
* In contrast, branch information appears only once for each branch in Branch relation and only branchNo is repeated in Staff relation, to represent where each member of staff works.



**Update Anomalies**

* Relations that contain redundant information may potentially suffer from update anomalies.
* Types of update anomalies include:
  + Insertion,
  + Deletion,
  + Modification.

**Lossless-join and Dependency Preservation Properties**

* Two important properties of decomposition:
  + Lossless-join property enables us to find any instance of original relation from corresponding instances in the smaller relations.
  + Dependency preservation property enables us to enforce a constraint on original relation by enforcing some constraint on each of the smaller relations.

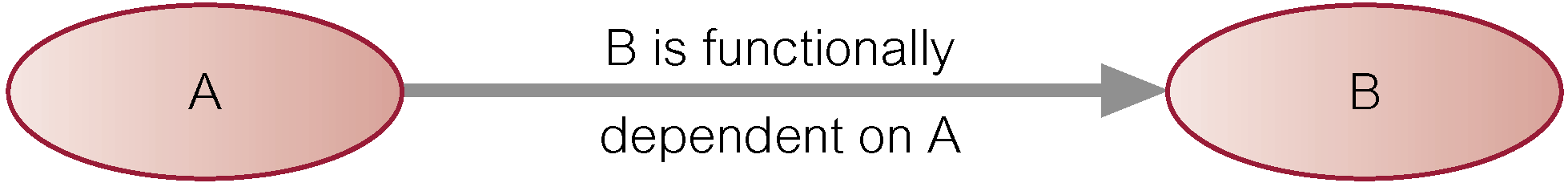
**Functional Dependencies**

* Important concept associated with normalization.
* Functional dependency describes relationship between attributes.
* For example, if A and B are attributes of relation R, B is functionally dependent on A (denoted A → B), if each value of A in R is associated with exactly one value of B in R.

**Characteristics of Functional Dependencies**

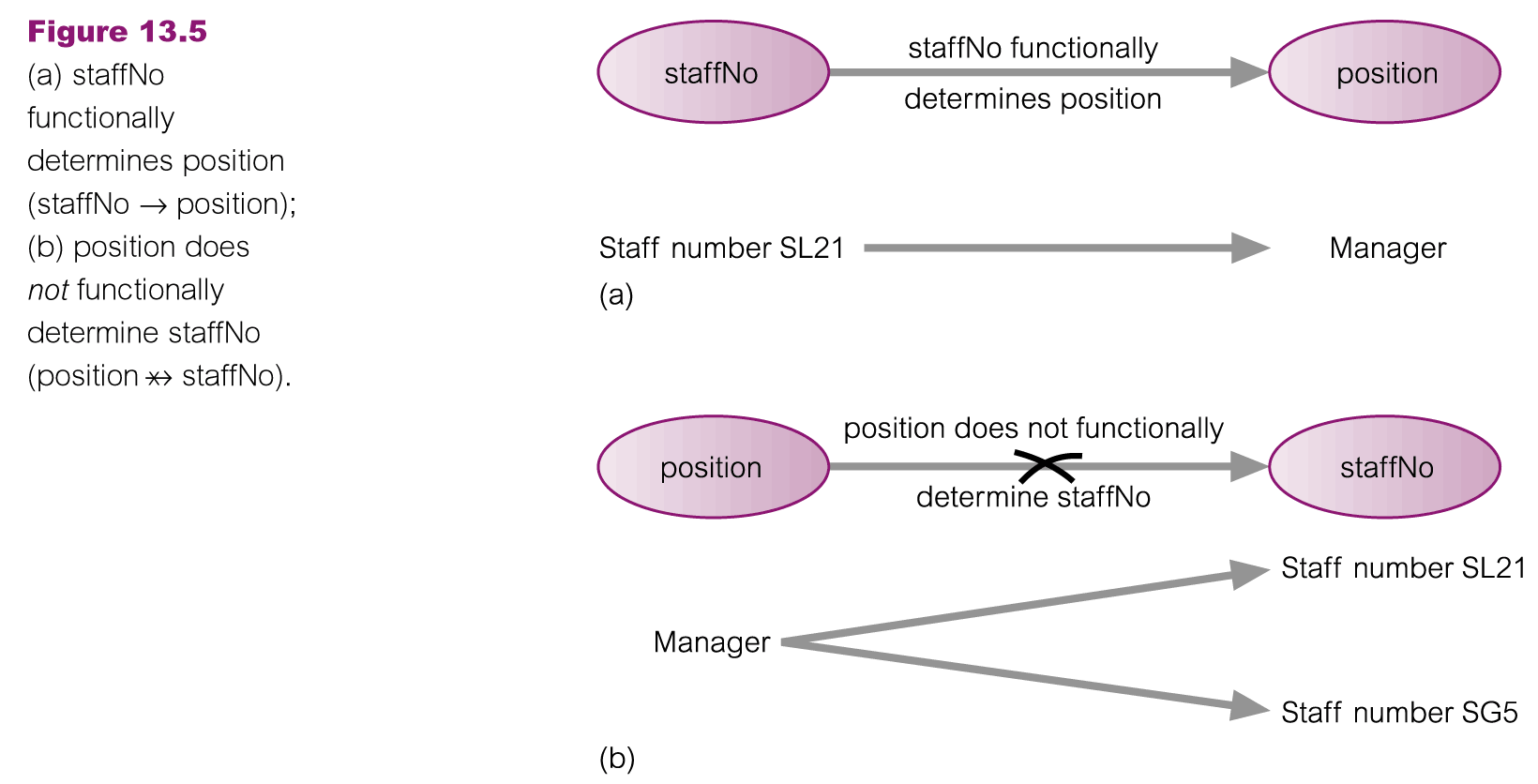
Property of the meaning or semantics of the attributes in a relation.

**Diagrammatic representation.**



The *determinant* of a functional dependency refers to the attribute or group of attributes on the left-hand side of the arrow.

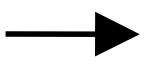
**An Example Functional Dependency**

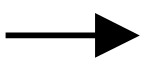


**Example Functional Dependency that holds for all Time**

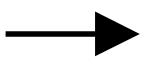
Consider the values shown in staffNo and sName attributes of the Staff relation.

Based on sample data, the following functional dependencies appear to hold.

staffNo sName 

sName staffNo 

However, the only functional dependency that remains true for all possible values for the staffNo and sName attributes of the Staff relation is:

staffNo sName 

**Full Functional Dependency**

* An additional characteristic of functional dependencies that is useful for normalization is that their determinants should have the minimal number of attributes necessary to maintain the functional dependency with the attribute(s) on the right hand-side. This requirement is called *full functional dependency.*
* Full functional dependencyindicates that if A and B are attributes of a relation, B is fully functionally dependent on A, if B is functionally dependent on A, but not on any proper subset of A.

**Example Full Functional Dependency**

Exists in the Staff relation.

staffNo, sName → branchNo

True - each value of (staffNo, sName) is associated with a single value of branchNo.

However, branchNo is also functionally dependent on a subset of (staffNo, sName), namely staffNo. Example above is a *partial dependency.*

**Characteristics of Functional Dependencies**

Main characteristics of functional dependencies used in normalization:

* + There is a *one-to-one* relationship between the attribute(s) on the left-hand side (determinant) and those on the right-hand side of a functional dependency.
  + Holds for *all* time.
  + The determinant has the *minimal* number of attributes necessary to maintain the dependency with the attribute(s) on the right hand-side.

**Transitive Dependencies**

* Important to recognize a transitive dependency because its existence in a relation can potentially cause update anomalies.
* Transitive dependency describes a condition where A, B, and C are attributes of a relation such that if A → B and B → C, then C is transitively dependent on A via B (provided that A is not functionally dependent on B or C).

**Example Transitive Dependency**

* Consider functional dependencies in the StaffBranch relation (see Slide 12).

staffNo → sName, position, salary, branchNo, bAddress

branchNo → bAddress

* Transitive dependency, branchNo → bAddress exists on staffNo via branchNo.

**Identifying Functional Dependencies**

* Identifying all functional dependencies between a set of attributes is relatively simple if the meaning of each attribute and the relationships between the attributes are well understood.
* This information should be provided by the enterprise in the form of discussions with users and/or documentation such as the users’ requirements specification.
* However, if the users are unavailable for consultation and/or the documentation is incomplete then depending on the database application it may be necessary for the database designer to use their common sense and/or experience to provide the missing information.

**Example - Identifying a set of functional dependencies for the StaffBranch relation**

* Examine semantics of attributes in StaffBranch relation (see Slide 12). Assume that position held and branch determine a member of staff’s salary.
* With sufficient information available, identify the functional dependencies for the StaffBranch relation as:

staffNo → sName, position, salary, branchNo, bAddress

branchNo → bAddress

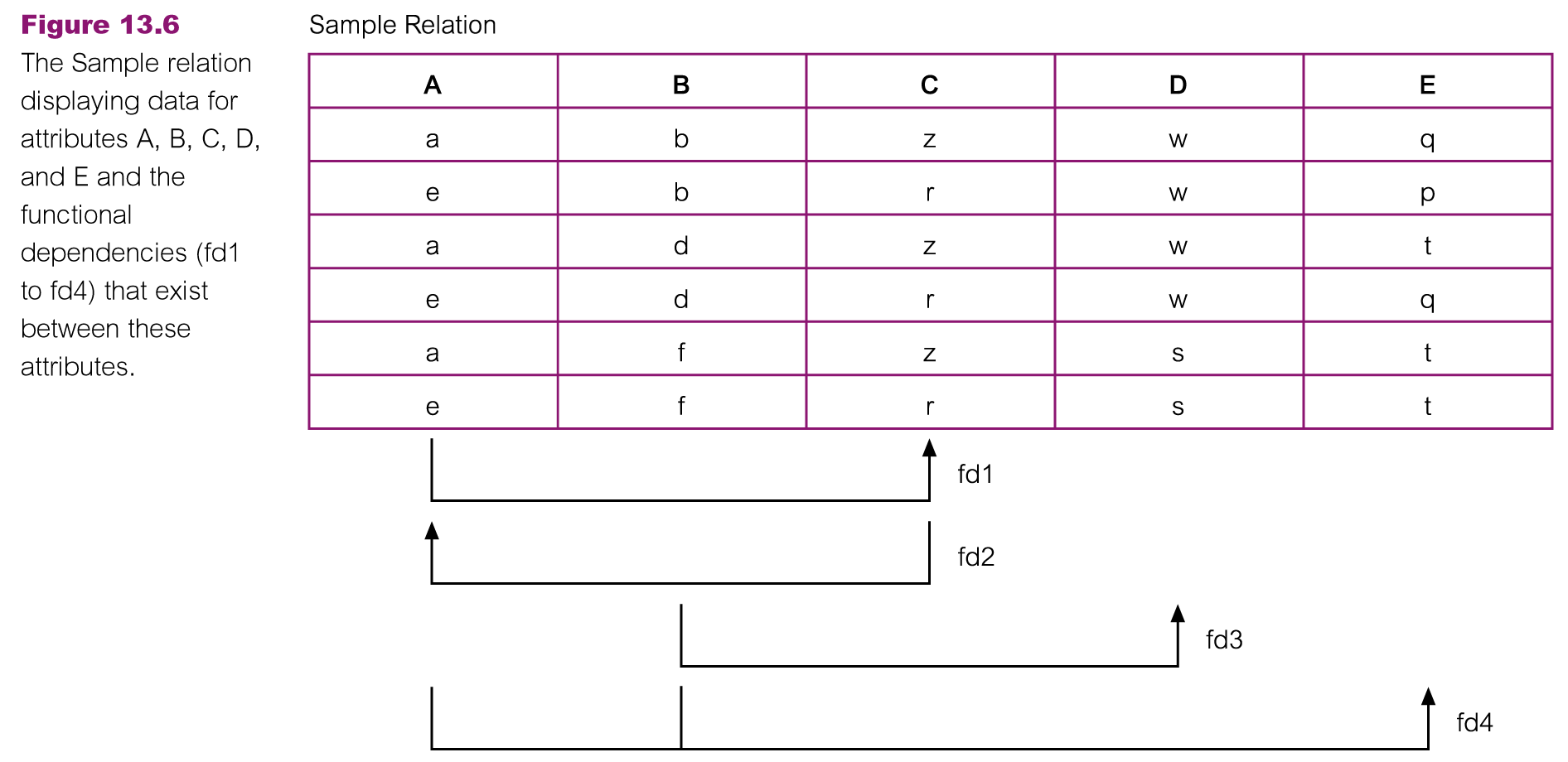
bAddress → branchNo

branchNo, position → salary

bAddress, position → salary

**Example - Using sample data to identify functional dependencies.**

* Consider the data for attributes denoted A, B, C, D, and E in the Sample relation (see Slide 33).
* Important to establish that sample data values shown in relation are representative of all possible values that can be held by attributes A, B, C, D, and E. Assume true despite the relatively small amount of data shown in this relation.



Function dependencies between attributes A to E in the Sample relation.

A → C (fd1)

C → A (fd2)

B → D (fd3)

A, B → E (fd4)

**Identifying the Primary Key for a Relation using Functional Dependencies**

* Main purpose of identifying a set of functional dependencies for a relation is to specify the set of integrity constraints that must hold on a relation.
* An important integrity constraint to consider first is the identification of candidate keys, one of which is selected to be the primary key for the relation.

**Example - Identify Primary Key for StaffBranch Relation**

* StaffBranch relation has five functional dependencies (see Slide 31).
* The determinants are staffNo, branchNo, bAddress, (branchNo, position), and (bAddress, position).
* To identify all candidate key(s), identify the attribute (or group of attributes) that uniquely identifies each tuple in this relation.
* All attributes that are not part of a candidate key should be functionally dependent on the key.
* The only candidate key and therefore primary key for StaffBranch relation, is staffNo, as *all* other attributes of the relation are functionally dependent on staffNo.

**Example - Identifying Primary Key for Sample Relation**

* Sample relation has four functional dependencies (see Slide 31).
* The determinants in the Sample relation are A, B, C, and (A, B). However, the only determinant that functionally determines all the other attributes of the relation is (A, B).
* (A, B) is identified as the primary key for this relation.

**The Process of Normalization**

* Formal technique for analyzing a relation based on its primary key and the functional dependencies between the attributes of that relation.
* Often executed as a series of steps. Each step corresponds to a specific normal form, which has known properties.
* As normalization proceeds, the relations become progressively more restricted (stronger) in format and also less vulnerable to update anomalies.

**Unnormalized Form (UNF)**

* A table that contains one or more repeating groups.
* To create an unnormalized table
  + Transform the data from the information source (e.g. form) into table format with columns and rows.

**First Normal Form (1NF)**

A relation in which the intersection of each row and column contains one and only one value.

**UNF to 1NF**

* Nominate an attribute or group of attributes to act as the key for the unnormalized table.
* Identify the repeating group(s) in the unnormalized table which repeats for the key attribute(s).
* Remove the repeating group by
  + Entering appropriate data into the empty columns of rows containing the repeating data (‘flattening’ the table).
  + Or by
  + Placing the repeating data along with a copy of the original key attribute(s) into a separate relation.

**Second Normal Form (2NF)**

* Based on concept of full functional dependency:
  + A and B are attributes of a relation,
  + B is fully dependent on A if B is functionally dependent on A but not on any proper subset of A.
* 2NF - A relation that is in 1NF and every non-primary-key attribute is fully functionally dependent on the primary key.

**1NF to 2NF**

* Identify primary key for the 1NF relation.
* Identify functional dependencies in the relation.
* If partial dependencies exist on the primary key remove them by placing them in a new relation along with copy of their determinant.

**Third Normal Form (3NF)**

* Based on concept of transitive dependency:
  + A, B and C are attributes of a relation such that if A 🡪 B and B 🡪 C,
  + then C is transitively dependent on A through B. (Provided that A is not functionally dependent on B or C).
* 3NF - A relation that is in 1NF and 2NF and in which no non-primary-key attribute is transitively dependent on the primary key.

**2NF to 3NF**

* Identify the primary key in the 2NF relation.
* Identify functional dependencies in the relation.
* If transitive dependencies exist on the primary key remove them by placing them in a new relation along with copy of their determinant.

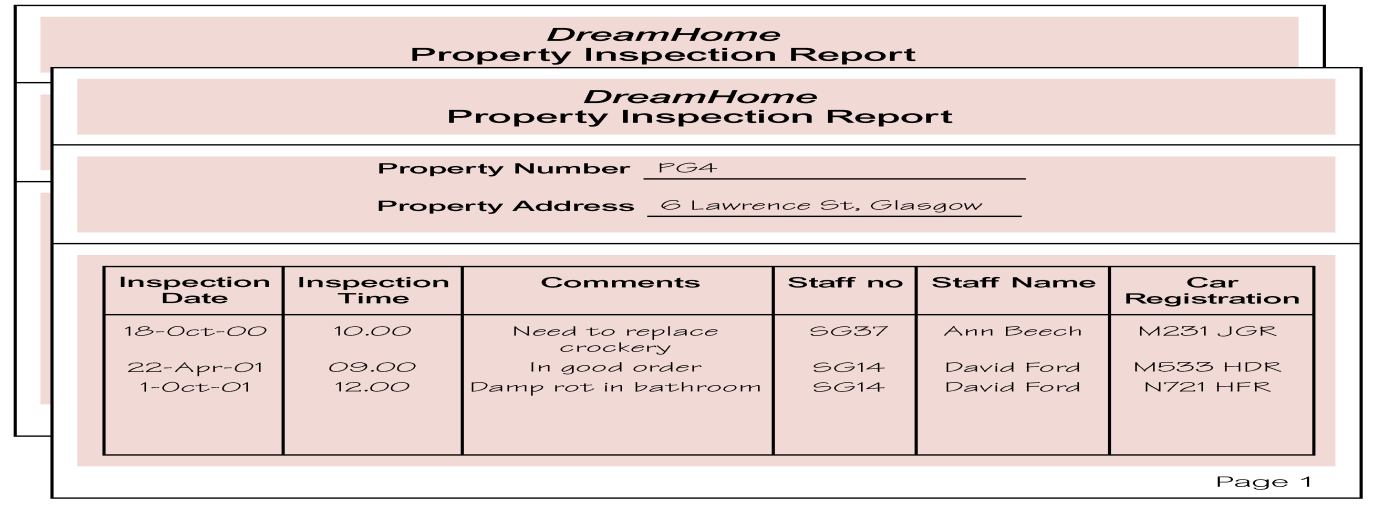
**General Definitions of 2NF and 3NF**

* Second normal form (2NF)
* A relation that is in 1NF and every non-primary-key attribute is fully functionally dependent on any candidate key.
* Third normal form (3NF)
* A relation that is in 1NF and 2NF and in which no non-primary-key attribute is transitively dependent on any candidate key.

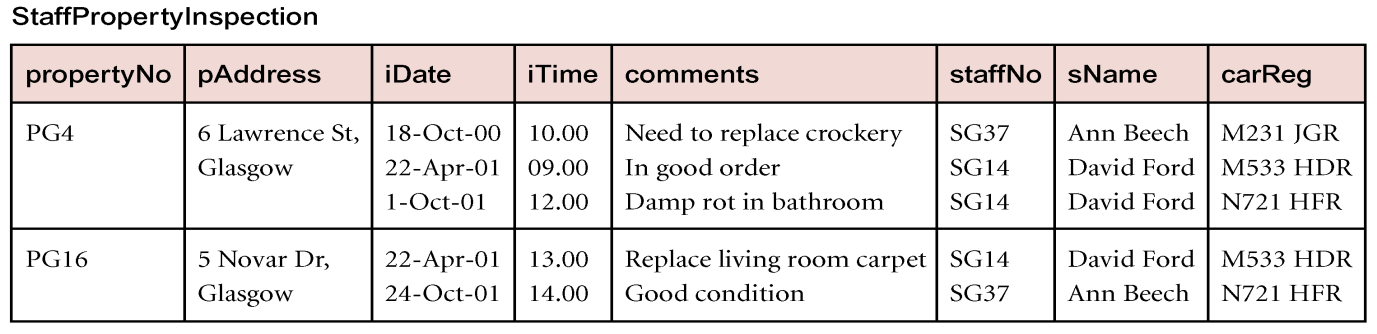
**Boyce–Codd Normal Form (BCNF)**

* Based on functional dependencies that take into account all candidate keys in a relation, however BCNF also has additional constraints compared with general definition of 3NF.
* BCNF - A relation is in BCNF if and only if every determinant is a candidate key.
* Difference between 3NF and BCNF is that for a functional dependency A → B, 3NF allows this dependency in a relation if B is a primary-key attribute and A is not a candidate key.
* Whereas, BCNF insists that for this dependency to remain in a relation, A must be a candidate key.
* Every relation in BCNF is also in 3NF. However, relation in 3NF may not be in BCNF.
* Violation of BCNF is quite rare.
* Potential to violate BCNF may occur in a relation that:
* contains two (or more) composite candidate keys;
* the candidate keys overlap (i.e. have at least one attribute in common).

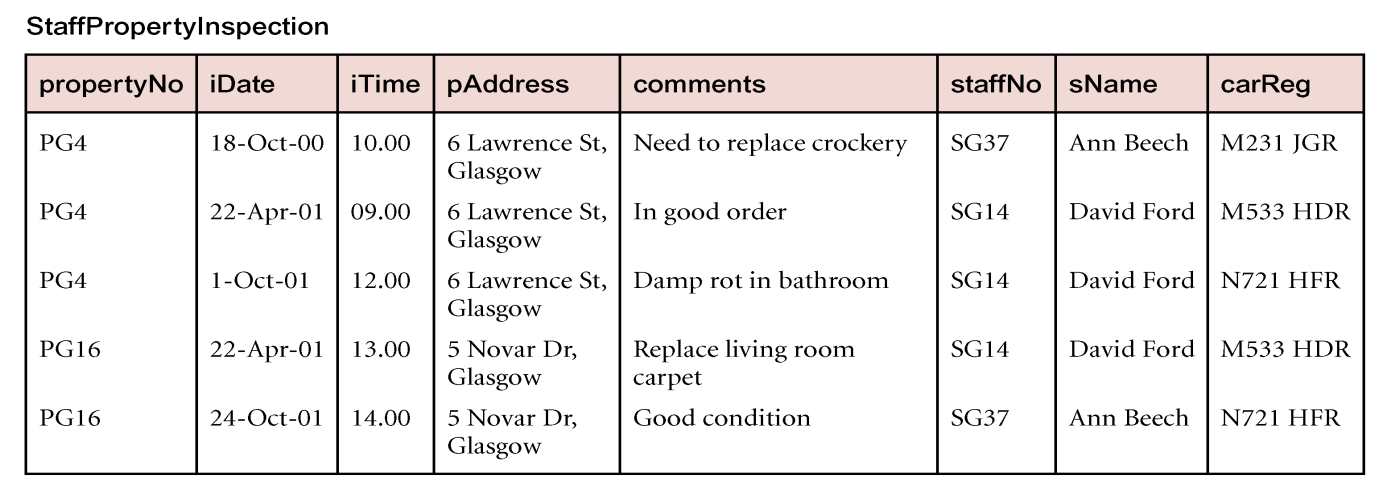
**Review of Normalization (UNF to BCNF)**



**UNF: Unnormalized form**



**INF: First Normal Form**

h

We use functional dependencies to identify candidate keys for the StaffPropertyInspection relation as being composite keys comprising (propertyNo, iDate), (staffNo, iDate, iTime) and (carReg,iDate,iTime). We select (propertyNo, iDate) as the primary key for this relation. The StaffPropertyInspection is now defined as:

**StaffPropertyInspection ( propertyNo,iDate,iTime,pAddress,comments,staffNo,sName,carReg )**

StaffPropertyInspection is in INF as there is a single value at the intersection of each row and column.

**2NF: Second Normal Form**

The normalization of INF relations to 2NF involves the removal of partial dependencies on the primary key. If a partial dependency exists, we remove the functionally dependent attributes from the relation by placing them in a new relation with a copy of their determinant.

The functional dependencies are:

Fd1: propertyNo,iDate 🡪 iTime,comments,staffNo,sName,CarReg,pAddress **(Primary Key)**

Fd2: propertyNo 🡪 pAddress **(Partial dependency)**

Fd3: staffNo 🡪 sName **(Transitive dependency)**

Fd4: staffNo,iDate 🡪 CarReg

Fd5: CarReg, iDate, iTime 🡪 propertyNo ,pAddress, comments,staffNo,sName **(Cadidate Key)**

Fd6: staffNo, iDate, iTime 🡪 propertyNo ,pAddress, comments **(Cadidate Key)**

By removing partial dependency from the relation, two new relations will be formed as follows:

**Property( propertyNo,pAddress )**

**PropertyInspection ( propertyNo,iDate,iTime,comments,staffNo,sName,carReg )**

**3NF: third normal Form:**

The normalization of 2NF relations to 3NF involves the removal of transitive dependencies. If transitive dependency exists, we remove the transitively dependent attributes from the relation by placing them in a new relation with a copy of their determinant.

Property Relation:

Fd2: propertyNo 🡪 pAddress

PropertyInspection Relation:

Fd1: propertyNo,iDate 🡪 iTime,comments,staffNo,sName,CarReg

Fd3: staffNo 🡪 sName

Fd4: staffNo,iDate 🡪 CarReg

Fd5’: CarReg, iDate, iTime 🡪 propertyNo , comments,staffNo,sName

Fd6’: staffNo, iDate, iTime 🡪 propertyNo , comments

**Property ( propertyNo,pAddress )**

**Staff ( staffNo, sName )**

**PropertyInspect ( propertyNo,iDate,iTime,comments,staffNo,carReg )**

**BCNF: Boyce Codd Normal Form:**

A relation is in BCNF if every determinant of a relation is a candidate key.

Property Relation:

Fd2: propertyNo 🡪 pAddress

Staff Relation

Fd3: staffNo 🡪 sName

PropertyInspect Relation

Fd1’: propertyNo,iDate 🡪 iTime,comments,staffNo ,CarReg

Fd4: staffNo,iDate 🡪 CarReg

Fd5’: CarReg, iDate, iTime 🡪 propertyNo , comments,staffNo

Fd6’: staffNo, iDate, iTime 🡪 propertyNo , comments

**Property ( propertyNo,pAddress )**

**Staff ( staffNo,sName )**

**StaffCar ( staffNo, iDate, carReg)**

**Inspection( propertyNo,iDate,iTime,comments,staffNo )**