Data Redundancy & Normalization

CS 3200/5200: Databases

What is wrong with this table?

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

Data redundancy of the branch address

Data redundancy leads to anomalies

Relations that contain redundant information may potentially suffer from *update anomalies*.

Insertion anomaly: Tuple being inserted may be inconsistent with data in other tuples in the table

Deletion anomaly: Deleting a tuple leads to loss of information other than the tuple

Update anomaly: Update of one tuple is dependent on the update of other tuples. (Some texts call this a modification anomaly.)

Insertion 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

- Data may become inconsistent as we add new staff. We might assign different addresses to the same branch id.

- How do we add a new branch? Null values for the primary key are not allowed!

Deletion 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

We might lose information about the London branch if we delete SL21 and SL41.

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

If we update the branch address for SG37 but not both SG14 and SG5 we have inconsistencies with our database.

A Better Relational Data Model

Branch

branchNo	street	city	postCode
B005	22 Deer Rd	London	SW1 4EH
B007	16 Argyll St	Aberdeen	AB2 3SU
B003	163 Main St	Glasgow	G11 9QX
B004	32 Manse Rd	Bristol	BS99 1NZ
B002	56 Clover Dr	London	NW10 6EU

Staff

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

What is Normalization?

Normalization is a technique for producing a set of relations (tables) with desirable properties, given the data requirements of the enterprise.

What are “desirable” properties?

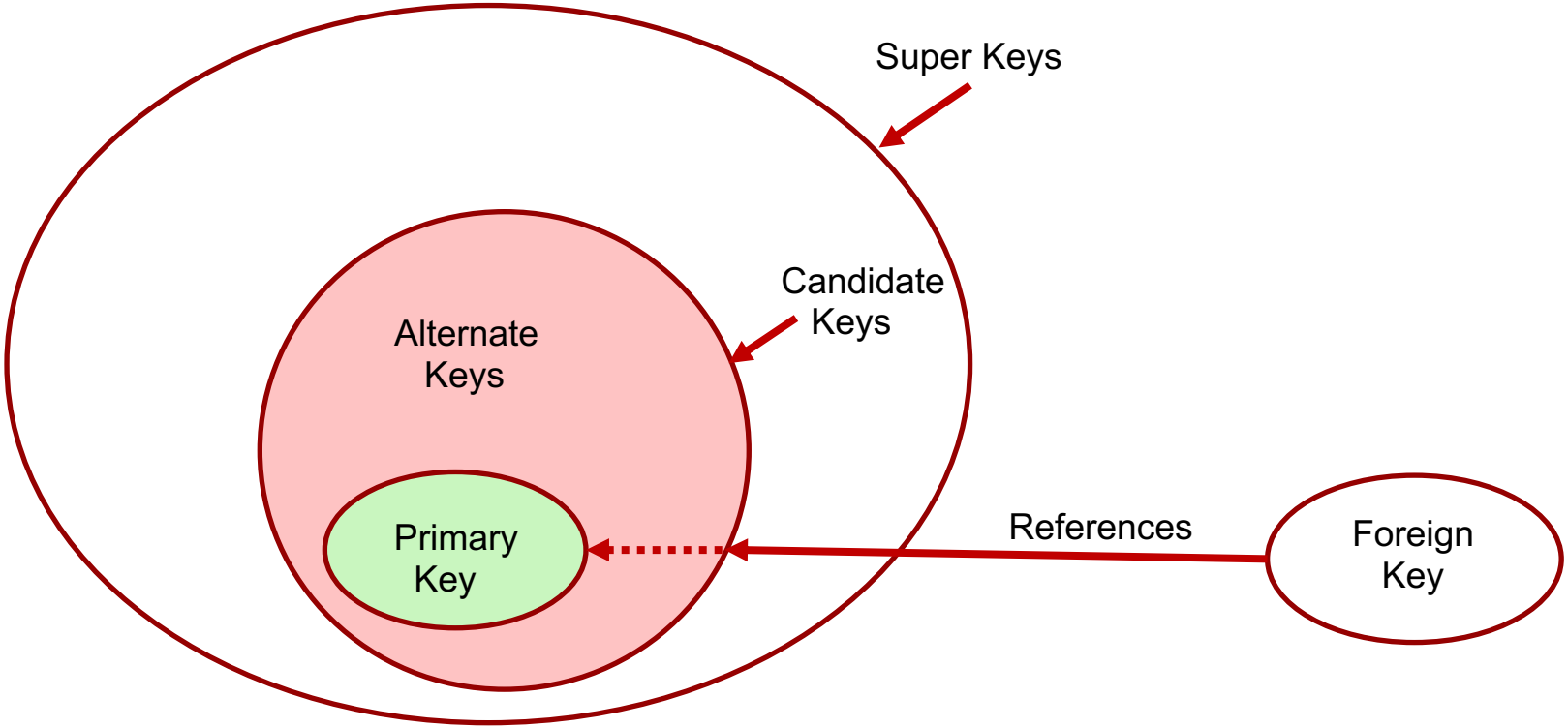
- Include the *minimal number of attributes* to support the requirements
- Attributes that are are closely related or *functionally dependent* should be grouped together into their own distinct table
- Redundancy should be *minimized* by ensuring that particular attributes occur only once across tables (excluding foreign key attributes that are essential for establishing relationships between tables.)

A Taxonomy of Keys

A **Key** is a set of attributes whose values can be used to identify a single row in a table.

Key	Definition
Super Key	Any set of attributes that uniquely identifies a row in a table (a tuple within a relation.)
Candidate Key	A Super Key that has a minimal number of attributes. (No subset of attributes is also a super key.)
Primary Key	The Candidate Key you chose for your table.
Alternate Key	The Candidate Key(s) you *didn't choose* for your table.
Foreign Key	The attributes that point to a key, usually in another table. Usually the key referenced is a *Primary Key* but it could be a Candidate Key. The Foreign Key is what lets you link up data in two different tables.
Composite Key	Any type of key that consists of two or more attributes.

A Taxonomy of Keys



Types of Relational Keys

Superkey: An attribute, or set of attributes, that uniquely identifies a tuple within a relation.

Staff

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

Can you identify some superkeys?

{sex, branchNo}?
{DOB}?
{position, branchNo}?

Types of Relational Keys

Candidate Key: A superkey such that there is no proper subset of attributes that is also a superkey.

Staff

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

Can you identify some Candidate Keys?

{DOB, position} ?
{DOB} ?
{staffNo} ?
{salary, branchNo?}

Types of Relational Keys

Primary Key: A selected candidate key used to uniquely identify a row in a table. Non chosen candidate keys are called **Alternate Keys**.

Staff

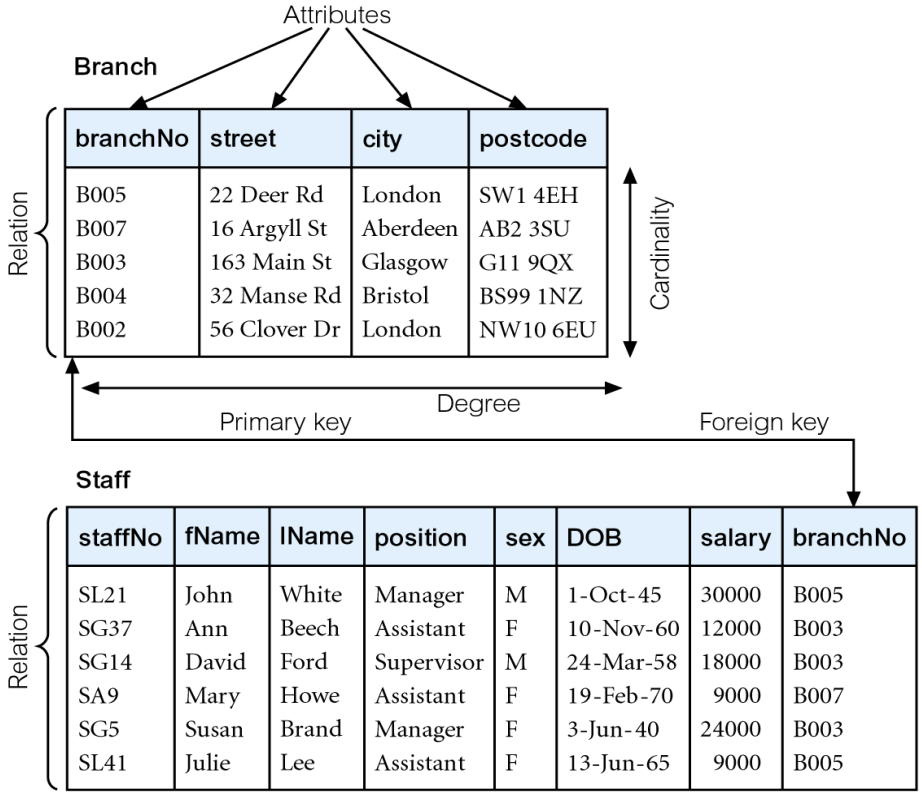
staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

Why is staffNo the best Primary Key of all our Candidate Keys?

Types of Relational Keys

Foreign Key: An attribute that references a candidate key in another table.

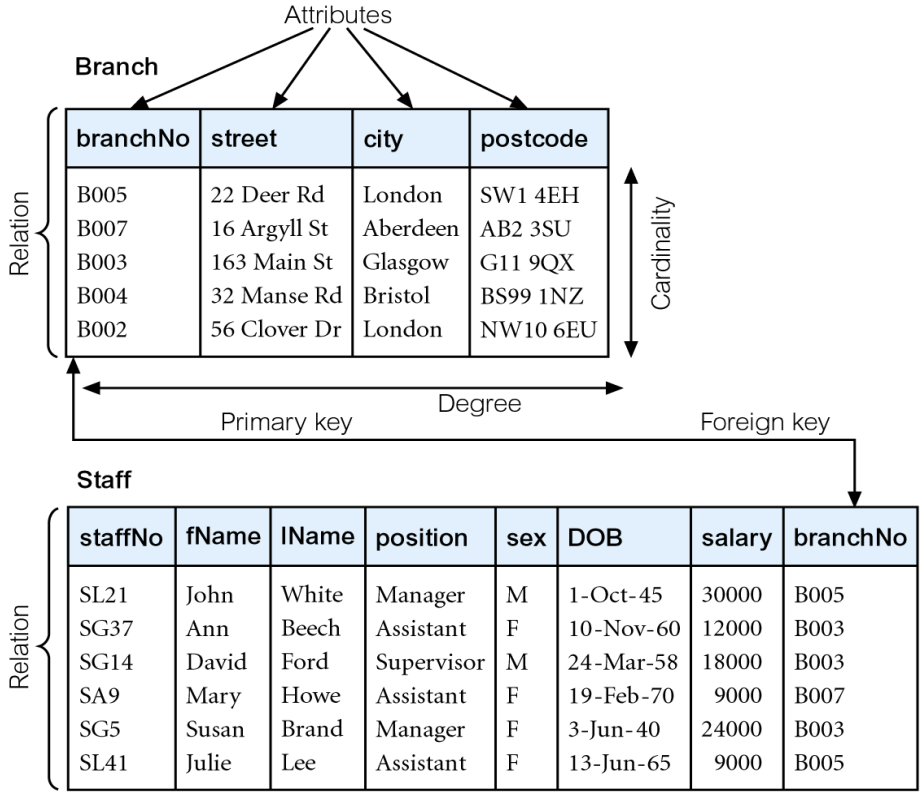
- Usually the referenced candidate key is a primary key.
- Technically, the referenced key could occur in the same table.



Referential Integrity

The value for the foreign key exists in the referenced table or the value is null (empty / unassigned).

Thus, foreign key constraints ensure referential integrity!



MySQL Supported Constraints

CONSTRAINT	DESCRIPTION
NOT NULL	In MySQL NOT NULL constraint allows to specify that a column can not contain any NULL value. MySQL NOT NULL can be used to CREATE and ALTER a table.
UNIQUE	The UNIQUE constraint in MySQL does not allow to insert a duplicate value in a column. The UNIQUE constraint maintains the uniqueness of a column in a table. More than one UNIQUE column can be used in a table.
PRIMARY KEY	A PRIMARY KEY constraint for a table enforces the table to accept unique data for a specific column and this constraint creates a unique index for accessing the table faster.
FOREIGN KEY	A FOREIGN KEY in MySQL creates a link between two tables by one specific column of both tables. The specified column in one table must be a PRIMARY KEY and referred by the column of another table known as FOREIGN KEY.
CHECK	A CHECK constraint controls the values in the associated column. The CHECK constraint determines whether the value is valid or not from a logical expression.
DEFAULT	In a MySQL table, each column must contain a value (including a NULL). While inserting data into a table, if no value is supplied to a column, then the column gets the value set as DEFAULT.

Doesn't work!

Creating a foreign key constraint in a table

```
CREATE TABLE Branch (
  branch_id INT PRIMARY KEY AUTO_INCREMENT,
  city VARCHAR(50)
 );
```

```
CREATE TABLE Staff (
  staff_id INT PRIMARY KEY AUTO_INCREMENT,
  name VARCHAR(50),
  branch_id INT, -- Which branch does the staff member work in?
  CONSTRAINT staff_fk_branch FOREIGN KEY (branch_id) REFERENCES Branch ( branch_id)
 );
```

Foreign key but no constraint? Yes!

What are the database design tradeoffs?

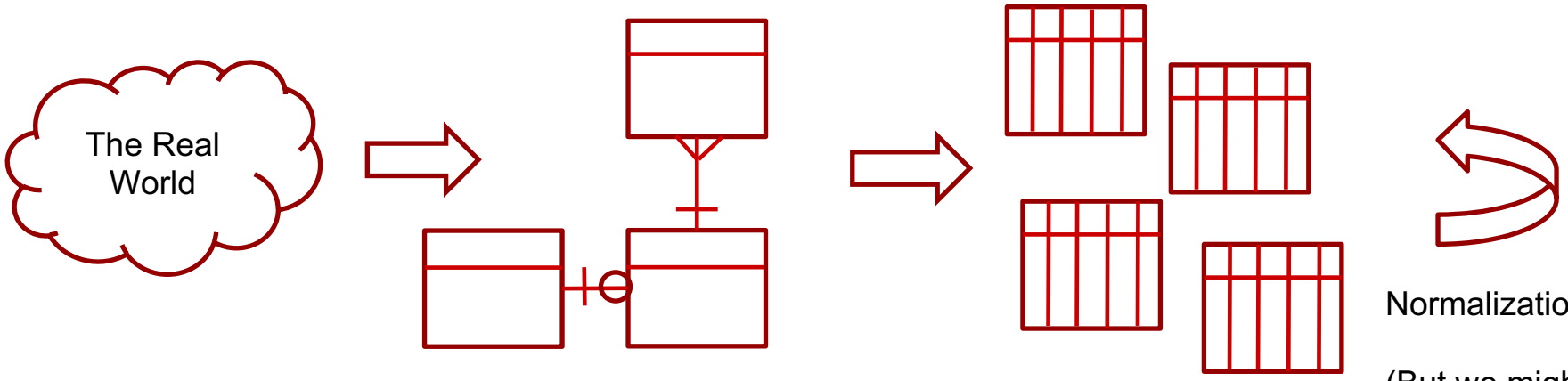
```
CREATE TABLE Branch (
  branch_id INT PRIMARY KEY AUTO_INCREMENT,
  city VARCHAR(50)
 );
```

```
CREATE TABLE Staff (
  staff_id INT PRIMARY KEY AUTO_INCREMENT,
  name VARCHAR(50),
  branch_id INT -- Which branch does the staff member work in?
  );
```

Properties of *normalized* Relations (Tables)

- Each table has a name distinct from all other tables in the database.
- Each cell of table contains exactly one atomic (single) value – no sets!
- Each attribute or column has a distinct name.
- Values within a given attribute are all from the same domain, i.e., they all have the same data type.
- Each tuple (record / row) is distinct
- Column order / Row order makes no difference

Where is normalization in the “big picture”?



Requirements Gathering

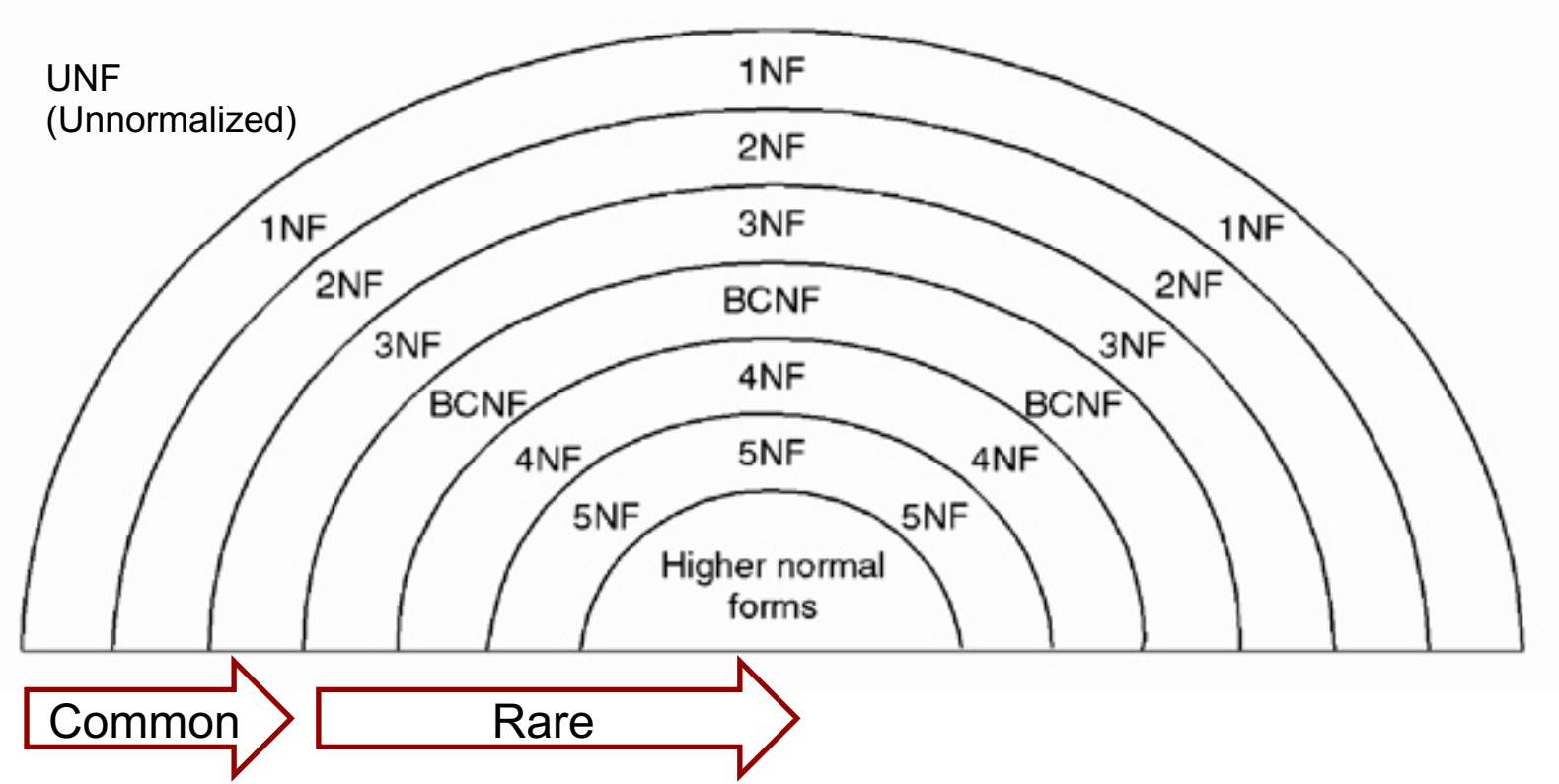
Conceptual Modeling

Logical Modeling

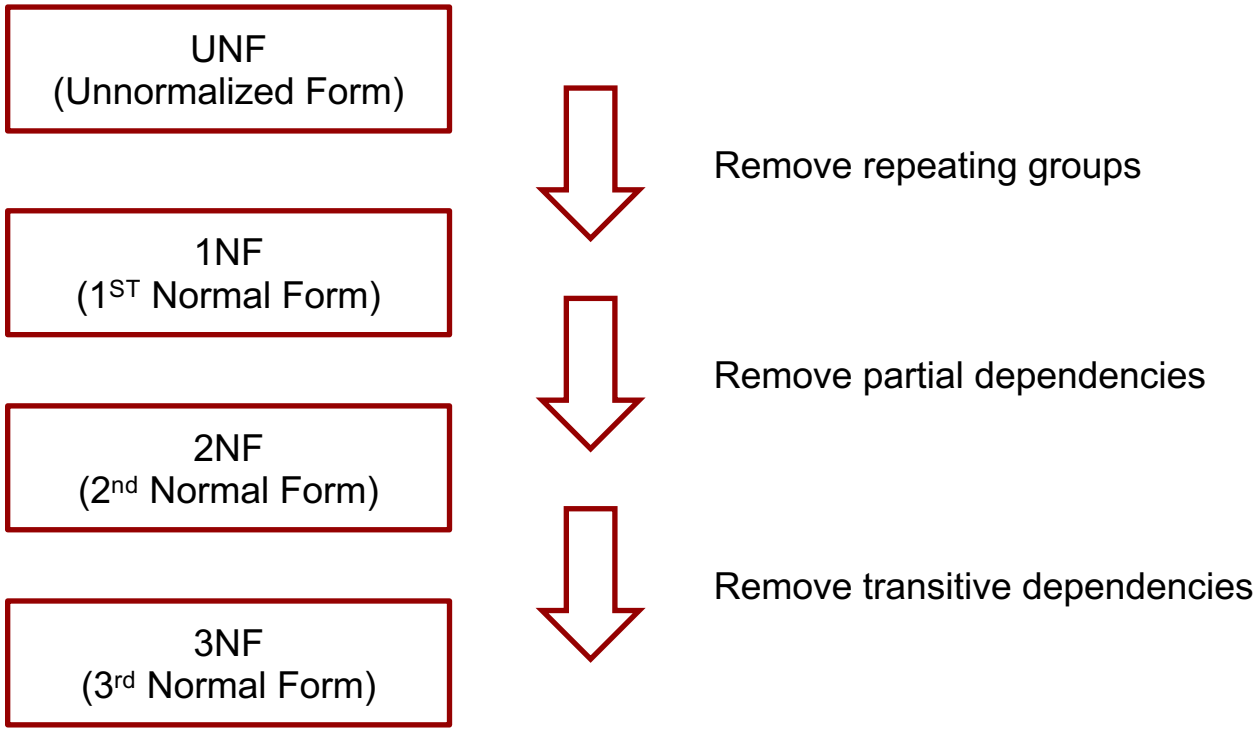
Normalization

(But we might also apply it during conceptual modeling as well)

Normal Forms



Normalization Stages



Unnormalized form

UNNORMALIZED FORM (UNF) – DUPLICATES ENTITIES					
Mother Id	Mother Name	Child1	Child2	Child3	Child4
1	Elsa	Mary	Alice	NULL	NULL
2	Golda	George	Fred	NULL	NULL
3	Viola	Ava	NULL	NULL	NULL
4	Iris	Kayla	NULL	NULL	NULL
5	Daisy	Harry	NULL	NULL	NULL

Table to track mother to child

First normal form

- Tuples in a relation must contain the same number of fields
- The domain of each attribute must be the same
- The value of each attribute contains only a single value. (No attributes are sets!)

1st Normal Form

Mother Id	Mother Name
1	Elsa
2	Golda
3	Viola
4	Iris
5	Daisy

Child Id	Name	Mother
11	Mary	1
12	Alice	1
13	George	2
14	Fred	2
15	Ava	3
16	Kayla	4
17	Harry	5

Decompose table, remove repeating attributes

Another example from Biology

Managing gene nomenclature

gene_id	official_name	Synonyms
123	abc	abc1, abc2, 3xab
456	sec	Foo22, sec12, afoo, xfoo

NO!

gene_id	official_name	Syn1	Syn2	Syn3	Syn4	Syn5	Syn6
123	abc	abc1	abc2	3xab	null	null	null
456	sec	foo2	sec12	afoo	xfoo	null	null

NO!

1st Normal Form

Gene_Id	Official Name
123	abc
456	foo

Syn_Id	Name	Gene_id
11	abc1	123
12	abc2	123
13	3xab	123
14	foo22	456
15	foobar	456
16	afoo	456
17	foox	456

Functional Dependencies

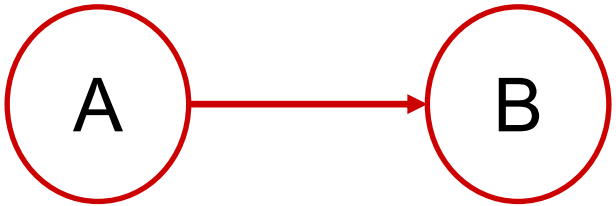
SSN	Name	Level	Rate Code	Hourly Wage	Hours
S	N	L	R	W	H
123-22-3666	Attishoo	48	8	10	40
231-31-5368	Smiley	22	8	10	30
131-24-3650	Smethurst	35	5	7	30
434-26-3751	Guldu	35	5	7	32
612-67-4134	Madayan	35	8	10	40

Can you find examples where, if you know X you can determine Y?
(X and Y are one *or more* attributes)

Functional Dependency

A **functional dependency** describes a relationship between attributes within a relation / table.

A and B are sets of one *or more* attributes.
B is functionally dependent on A (denoted A → B) if each value of A is associated with exactly one value of B.



Determinant

Identifying functional dependencies

In our prior example, we had two *functional dependencies* (FDS)

S → {S,N,L,R,W,H}
Each relation is dependent on the primary key since the primary key identifies the values for the other attributes

R → {W}
Each value of R is associated with exactly 1 value of W
(R is the *determinant*)

Full –vs- Partial Functional Dependencies

Full functional dependency: If A and B are attribute sets, B is fully dependent on A if B is functionally dependent on A but not on any proper subset of A. E.g. (order_id, seq_no) → product_id

Partial functional dependency: If there is an attribute of A that can be removed while still maintaining the dependency.

(staffNo, sName) → branchNo (partial)
staffNo → branchNo (full)

Second normal form

- Applies only to tables in 1NF that have a composite key (i.e., the primary key is composed of two or more attributes).
- A relation with a single-attribute primary key is automatically in at least 2NF
- A table is in 2NF if it is in 1NF **and every non-primary key attribute is fully functionally dependent on the (entire) primary key**
- 1NF → 2NF: remove partial key dependencies by table decomposition

Example 1NF with a composite key

On August 18, 2017, I bought a Macbook Pro, the Murach MySQL book, and a new briefcase
On August 20, 2017, I bough a toothbrush

Order_id	Order_date	Item_Seq	Item_Description
10001	18-Aug-2017	1	Macbook Pro
10001	18-Aug-2017	2	Murach MySQL
10001	18-Aug-2017	3	Briefcase
10002	20-Aug-2017	1	Toothbrush

Composite Primary Key: (order_id, item_seq)
(order_id, item_seq) → order_date is a **partial dependency** because order_id → order_date

2NF via decomposition

Order

order_id	order_date
10001	18-Aug-2017
10002	20-Aug-2017

order_id. → order_date

LineItem

order_id	item_seq	item_description
10001	1	Macbook Pro
10001	2	Murach MySQL
10001	3	Briefcase
10002	1	Toothbrush

(order_id, item_seq). → item_description

Third normal form

- Table is in first and second normal form
- No dependencies between 2 non-key attributes
- No non-key attribute is transitively dependent on the primary key
- Solution: decompose the table so that the offending attribute is in a separate table

Example 2NF that is not in 3NF

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

staffNo → branchNo → bAddress

Branch address is transitively dependent on the primary key, staffNo.
Therefore, we need to move branch address to another table.

A 3NF Relational Data Model

Branch

branchNo	street	city	postCode
B005	22 Deer Rd	London	SW1 4EH
B007	16 Argyll St	Aberdeen	AB2 3SU
B003	163 Main St	Glasgow	G11 9QX
B004	32 Manse Rd	Bristol	BS99 1NZ
B002	56 Clover Dr	London	NW10 6EU

Staff

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

Redundancy leads to anomalies

UPDATE ANOMALIES

Modification anomaly: Can we change W in just the first tuple?

Deletion anomaly: Can we delete tuple 3 and 4?

Insertion anomaly: What if we insert another tuple where the rating equals 8 but the wage is not equal to 10?
How do we track the wage associated with ratings not stored in the employee table?

S	N	L	R	W	H
123-22-3666	Attishoo	48	8	10	40
231-31-5368	Smiley	22	8	10	30
131-24-3650	Smethurst	35	5	7	30
434-26-3751	Guldu	35	5	7	32
612-67-4134	Madayan	35	8	10	40

There is a functional dependency between Rate and Wage. This functional dependency limits the operations I can do on my data if I want to keep my data consistent.

How to remove a functional dependency?

S	N	L	R	W	H
123-22-3666	Attishoo	48	8	10	40
231-31-5368	Smiley	22	8	10	30
131-24-3650	Smethurst	35	5	7	30
434-26-3751	Guldu	35	5	7	32
612-67-4134	Madayan	35	8	10	40

S	N	L	R	H
123-22-3666	Attishoo	48	8	40
231-31-5368	Smiley	22	8	30
131-24-3650	Smethurst	35	5	30
434-26-3751	Guldu	35	5	32
612-67-4134	Madayan	35	8	40

Decompose the original relation into 2 relations.

R	W
8	10
5	7

Wage

Bill Kent's quote (paraphrased):

Every non-key attribute must provide a fact :

about the key,

the *whole* key,

and nothing but the key,

so help me Codd!

Kent W. (1982) A Simple Guide to Five Normal Forms in Relational Database Theory

Boyce-Codd Normal Form (BCNF)

- Also called 3.5 NF because it is a more restrictive form of third normal form (3NF)

- Two conditions must be met:
  - ❑ The table is in 3NF
  - ❑ If we have a functional dependency A→B, then A should be a *Super Key.* In other words, attributes that are **not** part of the Primary Key should not functionally determine the value of attributes that **are** part of the Primary Key

Boyce-Codd Normal Form (BCNF)

Student_ID	Course	Professor
100	Database Systems	Rachlin
100	Graphics	Shah
101	Database Systems	Derbinsky
102	Theory	Gold
103	Database Systems	Rachlin

{Professor} is not a Super Key.

But we have the dependency {Professor} → {Course}

Therefore,
This table is **not** BCNF!

A student takes multiple courses
A course can be taught by multiple professors
We assume, in this example, that each professor teaches **one** course, i.e., {Professor} → {Course}
{Student_ID, Course} is the PRIMARY KEY, i.e., {Student_ID, Course} → {Professor}

Boyce-Codd Normal Form (BCNF)

Student_ID	Course	Professor
100	Database Systems	Rachlin
100	Graphics	Shah
101	Database Systems	Derbinsky
102	Theory	~~Gold~~ Rachlin? (No!)
103	Database Systems	Rachlin

As is, the table cannot enforce the desired business rule: that every professor teaches at most one class. Nothing prevents us from assigning professor Rachlin to teach Theory. (This would be a bad idea anyway!)

The problem is that the table is **NOT** in BCNF form! {Professor} → {Course} no longer true!

Boyce-Codd Normal Form (BCNF)

Student_ID	Professor_ID
100	200
100	205
101	210
102	215
103	200

Professor_ID	Professor	Course
200	Rachlin	Database Systems
205	Shah	Graphics
210	Derbinsky	Database Systems
215	Gold	Theory

Now {Professor_ID} → {Professor, Course}

{Professor} is an alternate key, so {Professor}→{Course} doesn't create a transitive dependency, and note, also, that the table design now enforces the business rule that a professor only teaches one course.