# Normalization

**ITM 692**

**Sanjay Goel**

# Normalization

## Definition

- This is the process which allows you to winnow out redundant data within your database.
  - The results of a well executed normalization process are the same as those of a well planned E-R model
- This involves restructuring the tables to successively meeting higher forms of Normalization.
- A properly normalized database should have the following characteristics
  - Scalar values in each fields
  - Absence of redundancy.
  - Minimal use of null values.
  - Minimal loss of information.

(Note: Winnow(Webster): To get rid of / eliminate inferior material

# Normalization

## Process

- Eliminate Repeating Groups
  - Make a separate table for each set of related attributes and give each table a primary key.
- Eliminate Redundant Data
  - If an attribute depends on only part of a multivalued key, remove it to a separate table.
- Eliminate Columns not dependent on key
  - If attributes do not contribute to a description of the key, remove them to a separate table.

# Normalization

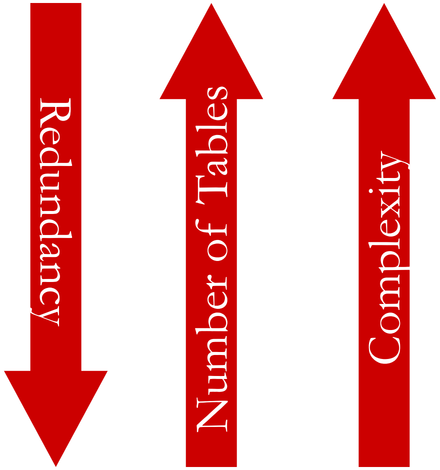
## Process

- Isolate Independent multiple relationships
  - No table may contain two or more 1:n or n:m relationships that are not directly related.
- Isolate Semantically Related Multiple Relationships
  - There may be practical constraints on information that justify separating logically related many-to-many relationships.

# Normalization

## Levels

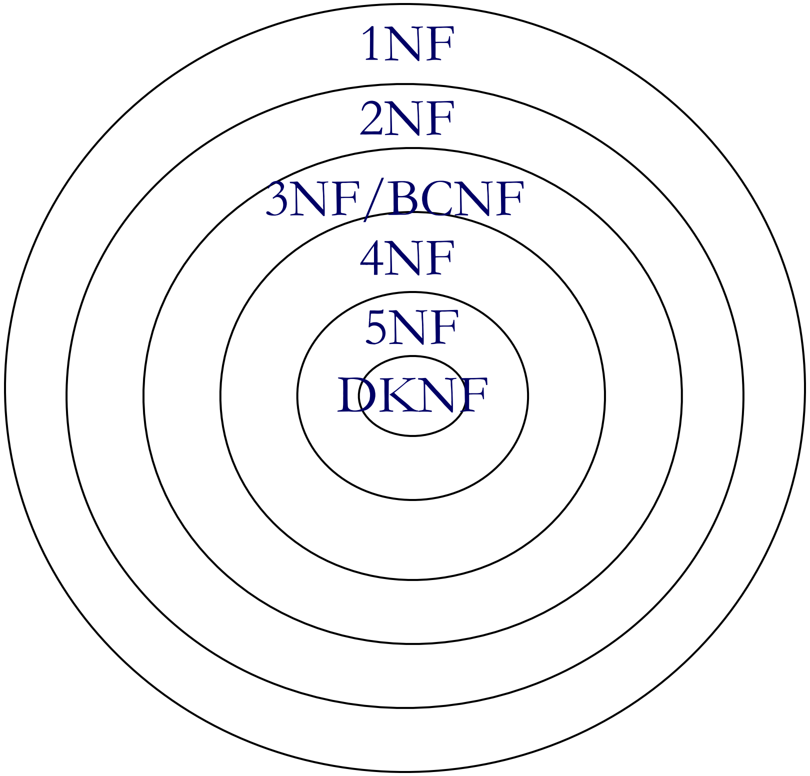
- Levels of normalization based on the amount of redundancy in the database.
- Relational theory defines a number of structure conditions called Normal Forms that assure that certain data anomalies do not occur in a database.
- Various levels of normalization are:
  - First Normal Form (1NF)
  - Second Normal Form (2NF)
  - Third Normal Form (3NF)
  - Boyce-Codd Normal Form (BCNF)
  - Fourth Normal Form (4NF)
  - Fifth Normal Form (5NF)
  - Domain Key Normal Form (DKNF)



Most databases should be 3NF or BCNF in order to avoid the database anomalies.

# Normalization

## Levels



| | |
|---|---|
| 1NF | *Keys; No repeating groups or multi-valued* |
| 2NF | *No partial dependencies* |
| 3NF | *No transitive dependencies* |
| BCNF | *Determinants are candidate keys* |
| 4NF | *No multivalued dependencies* |
| 5NF | *No multivalued dependencies* |
| 4NF | *No multivalued dependencies* |

Each higher level is a subset of the lower level

# Normalization

## First Normal Form (1NF)

A table is considered to be in 1NF if all the fields contain only scalar values (as opposed to list of values).

### Example (Not 1NF)

| ISBN | Title | AuName | AuPhone | PubName | PubPhone | Price |
|---|---|---|---|---|---|---|
| 0-321-32132-1 | Balloon | Sleepy,<br>Snoopy,<br>Grumpy | 321-321-1111,<br>232-234-1234,<br>665-235-6532 | Small House | 714-000-0000 | $34.00 |
| 0-55-123456-9 | Main Street | Jones, Smith | 123-333-3333,<br>654-223-3455 | Small House | 714-000-0000 | $22.95 |
| 0-123-45678-0 | Ulysses | Joyce | 666-666-6666 | Alpha Press | 999-999-9999 | $34.00 |
| 1-22-233700-0 | Visual Basic | Roman | 444-444-4444 | Big House | 123-456-7890 | $25.00 |

Author and AuPhone columns are not scalar

# Normalization

## 1NF: Decomposition

1. Place all items appearing in the repeating group in a new table
2. Designate a primary key for each new table produced.
3. Create a relationship between the two tables
   - For 1:N relation duplicate the P.K. from 1 side to many side
   - For M:N relation create a new table with P.K. from both tables

## Example (1NF)

| ISBN | Title | PubName | PubPhone | Price |
|---|---|---|---|---|
| 0-321-32132-1 | Balloon | Small House | 714-000-0000 | $34.00 |
| 0-55-123456-9 | Main Street | Small House | 714-000-0000 | $22.95 |
| 0-123-45678-0 | Ulysses | Alpha Press | 999-999-9999 | $34.00 |
| 1-22-233700-0 | Visual Basic | Big House | 123-456-7890 | $25.00 |

| ISBN | AuName | AuPhone |
|---|---|---|
| 0-321-32132-1 | Sleepy | 321-321-1111 |
| 0-321-32132-1 | Snoopy | 232-234-1234 |
| 0-321-32132-1 | Grumpy | 665-235-6532 |
| 0-55-123456-9 | Jones | 123-333-3333 |
| 0-55-123456-9 | Smith | 654-223-3455 |
| 0-123-45678-0 | Joyce | 666-666-6666 |
| 1-22-233700-0 | Roman | 444-444-4444 |

# Normalization

## Functional Dependencies

1. If one set of attributes in a table determines another set of attributes in the table, then the second set of attributes is said to be functionally dependent on the first set of attributes.

### Example 1

| ISBN | Title | Price |
|---|---|---|
| 0-321-32132-1 | Balloon | $34.00 |
| 0-55-123456-9 | Main Street | $22.95 |
| 0-123-45678-0 | Ulysses | $34.00 |
| 1-22-233700-0 | Visual Basic | $25.00 |

Table Scheme: {ISBN, Title, Price}

Functional Dependencies: {ISBN} → {Title}

{ISBN} → {Price}

# Normalization

## Functional Dependencies

## Example 2

| PubID | PubName | PubPhone |
|---|---|---|
| 1 | Big House | 999-999-9999 |
| 2 | Small House | 123-456-7890 |
| 3 | Alpha Press | 111-111-1111 |

Table Scheme: {PubID, PubName, PubPhone}

Functional Dependencies: {PubId} → {PubPhone}
{PubId} → {PubName}
{PubName, PubPhone} → {PubID}

## Example 3

| AuID | AuName | AuPhone |
|---|---|---|
| 1 | Sleepy | 321-321-1111 |
| 2 | Snoopy | 232-234-1234 |
| 3 | Grumpy | 665-235-6532 |
| 4 | Jones | 123-333-3333 |
| 5 | Smith | 654-223-3455 |
| 6 | Joyce | 666-666-6666 |
| 7 | Roman | 444-444-4444 |

Table Scheme: {AuID, AuName, AuPhone}

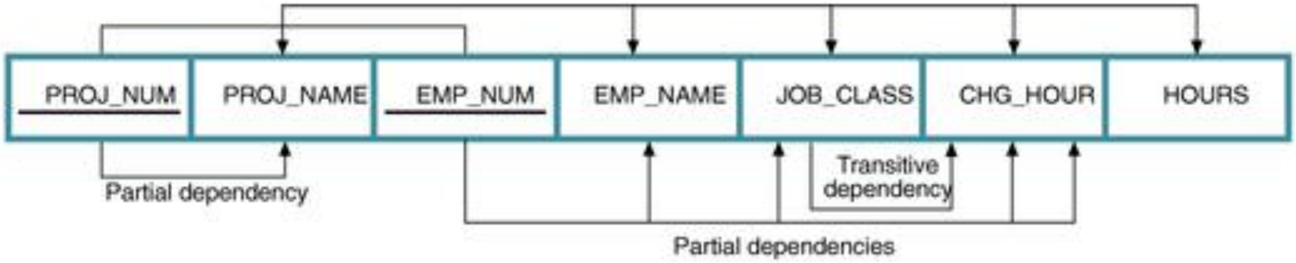
Functional Dependencies: {AuId} → {AuPhone}
{AuId} → {AuName}
{AuName, AuPhone} → {AuID}

# Normalization

## Dependency Diagram

- The primary key components are bold, underlined, and shaded in a different color.
- The arrows above entities indicate all desirable dependencies, i.e., dependencies that are based on PK.
- The arrows below the dependency diagram indicate less desirable dependencies -- partial dependencies and transitive dependencies

**Example:**



# Normalization

## Functional Dependencies: Example

Database to track reviews of papers submitted to an academic conference. Prospective authors submit papers for review and possible acceptance in the published conference proceedings. Details of the entities:

- Author information includes a unique author number, a name, a mailing address, and a unique (optional) email address.
- Paper information includes the primary author, the paper number, the title, the abstract, and review status (pending, accepted,rejected)
- Reviewer information includes the reviewer number, the name, the mailing address, and a unique (optional) email address
- A completed review includes the reviewer number, the date, the paper number, comments to the authors, comments to the program chairperson, and ratings (overall, originality, correctness, style, clarity)

# Normalization

## Functional Dependencies: Example

Functional Dependencies

- AuthNo → AuthName, AuthEmail, AuthAddress
- AuthEmail → AuthNo
- PaperNo → Primary-AuthNo, Title, Abstract, Status
- RevNo → RevName, RevEmail, RevAddress
- RevEmail → RevNo
- RevNo, PaperNo → AuthComm, Prog-Comm, Date, Rating1, Rating2, Rating3, Rating4, Rating5

# Normalization

## Second Normal Form (2NF)

For a table to be in 2NF, there are two requirements:

- The database is in first normal form
- All **nonkey** attributes in the table must be functionally dependent on the entire primary key

***Note:*** *Remember that we are dealing with non-key attributes*

### Example 1 (Not 2NF)

**Scheme → {StudentId, CourseId, StudentName, CourseTitle, Grade}**

1. **Key → {StudentId, CourseId}**
2. **{StudentId} → {StudentName}**
3. **{CourseId} → {CourseTitle}**
4. **{StudentId, CourseId} → {Grade}**
5. **StudentName depends on a subset of the key I.e. StudentId**
6. **CourseTitle depends on a subset of the key. i.e. CourseId**

# Normalization

## Second Normal Form (2NF)

### Example 2 (Not 2NF)

Scheme → {City, Street, HouseNumber, HouseColor, CityPopulation}

1. key → {City, Street, HouseNumber}
2. {City, Street, HouseNumber} → {HouseColor}
3. {City} → {CityPopulation}
4. CityPopulation does not belong to any key.
5. CityPopulation is functionally dependent on the City which is a proper subset of the key

### Example 3 (Not 2NF)

Scheme → {studio, movie, budget, studio_city}

1. Key → {studio, movie}
2. {studio, movie} → {budget}
3. {studio} → {studio_city}
4. studio_city is not a part of a key
5. studio_city functionally depends on studio which is a proper subset of the key

# Normalization

## 2NF: Decomposition

1. If a data item is fully functionally dependent on only a part of the primary key, move that data item and that part of the primary key to a new table.
2. If other data items are functionally dependent on the same part of the key, place them in the new table also
3. Make the partial primary key copied from the original table the primary key for the new table.

(Place all items that appear in the repeating group in a new table)

### Example 1 (Convert to 2NF)

**Old Scheme → {StudentId, CourseId, StudentName, CourseTitle, Grade}**

**New Scheme → {StudentId, StudentName}**

**New Scheme → {CourseId, CourseTitle}**

**New Scheme → {StudentId, CourseId, Grade}**

# Normalization

## 2NF: Decomposition

### Example 2 (Convert to 2NF)

**Old Scheme → {StudioID, Movie, Budget, StudioCity}**

**New Scheme → {Movie, StudioID, Budget}**

**New Scheme → {Studio, City}**

### Example 3 (Convert to 2NF)

**Old Scheme → {City, Street, HouseNumber, HouseColor, CityPopulation}**

**New Scheme → {City, Street, HouseNumber, HouseColor}**

**New Scheme → {City, CityPopulation}**

# Normalization

## Third Normal Form (3NF)

- This form dictates that all **non-key** attributes of a table must be functionally dependent on a candidate key such that there are no interdependencies among non-key attributes i.e. there should be no transitive dependencies

- For a table to be in 3NF, there are two requirements
  - The table should be second normal form
  - No attribute is transitively dependent on the primary key

### Example (Not in 3NF)

**Scheme → {Title, PubID, BookType, Price }**

1. **Key → {Title, PubId}**
2. **{Title, PubId} → {BookType}**
3. **{BookType} → {Price}**
4. **Both Price and BookType depend on a key hence 2NF**
5. **Transitively {Title, PubID} → {Price} hence not in 3NF**

| Title | PubID | BookType | Price |
|---|---|---|---|
| Moby Dick | 1 | Adventure | 34.95 |
| Giant | 2 | Adventure | 34.95 |
| MobyDick | 2 | Adventure | 34.95 |
| Iliad | 1 | War | 44.95 |
| Romeo &Juliet | 1 | Love | 59.90 |

# Normalization

## Third Normal Form (3NF)

### Example 2 (Not in 3NF)

Scheme → {StudioID, StudioCity, CityTemp}

1. Primary Key → {StudioID}
2. {StudioID} → {StudioCity}
3. {StudioCity} → {CityTemp}
4. {StudioID} → {CityTemp}
5. Both StudioCity and CityTemp depend on the entire key hence 2NF
6. CityTemp transitively depends on Studio hence violates 3NF

### Example 3 (Not in 3NF)

Scheme → {BuildingID, Contractor, Fee}

1. Primary Key → {BuildingID}
2. {BuildingID} → {Contractor}
3. {Contractor} → {Fee}
4. {BuildingID} → {Fee}
5. Fee transitively depends on the BuildingID
6. Both Contractor and Fee depend on the entire key hence 2NF

| BuildingID | Contractor | Fee |
|---|---|---|
| 100 | Randolph | 1200 |
| 150 | Ingersoll | 1100 |
| 200 | Randolph | 1200 |
| 250 | Pitkin | 1100 |
| 300 | Randolph | 1200 |

# Normalization

## 3NF: Decomposition

1. Move all items involved in transitive dependencies to a new entity.
2. Identify a primary key for the new entity.
3. Place the primary key for the new entity as a foreign key on the original entity.

## Example 1 (Convert to 3NF)

**Old Scheme → {Title, PubID, BookType, Price }**

**New Scheme → {BookType, Price}**

**New Scheme → {Title, PubID, BookType}**

# Normalization

## 3NF: Decomposition

### Example 2 (Convert to 3NF)

**Old Scheme → {StudioID, StudioCity, CityTemp}**

**New Scheme → {StudioID, StudioCity}**

**New Scheme → {StudioCity, CityTemp}**

### Example 3 (Convert to 3NF)

**Old Scheme → {BuildingID, Contractor, Fee}**

**New Scheme → {BuildingID, Contractor}**

**New Scheme → {Contractor, Fee}**

| BuildingID | Contractor |
|---|---|
| 100 | Randolph |
| 150 | Ingersoll |
| 200 | Randolph |
| 250 | Pitkin |
| 300 | Randolph |

| Contractor | Fee |
|---|---|
| Randolph | 1200 |
| Ingersoll | 1100 |
| Pitkin | 1100 |

# Normalization

## Boyce-Codd Normal Form (BCNF)

- BCNF does not allow dependencies between attributes that belong to candidate keys.
- BCNF is a refinement of the third normal form in which it drops the restriction of a non-key attribute from the 3rd normal form.
- Third normal form and BCNF are not same if following conditions are true:
  - The table has two or more candidate keys
  - At least two of the candidate keys are composed of more than one attribute
  - The keys are not disjoint i.e. The composite candidate keys share some attributes

**Example 1 - Address (Not in BCNF)**

**Scheme → {City, Street, ZipCode}**

1. **Key1 → {City, Street }**
2. **Key2 → {ZipCode, Street}**
3. **No non-key attribute hence 3NF**
4. **{City, Street} → {ZipCode}**
5. **{ZipCode} → {City}**
6. **Dependency between attributes belonging to a key**

# Normalization

## Boyce-Codd Normal Form (BCNF)

### Example 2 - Movie (Not in BCNF)

**Scheme → {MovieTitle, StudioID, MovieID, ActorName, Role, Payment }**

1. **Key1 → {MovieTitle, StudioID, ActorName}**
2. **Key2 → {MovieID, ActorName}**
3. **Both role and payment functionally depend on both candidate keys thus 3NF**
4. **{MovieID} → {MovieTitle}**
5. **Dependency between MovieID & MovieTitle Violates BCNF**

### Example 3 - Consulting (Not in BCNF)

**Scheme → {Client, Problem, Consultant}**

(Only one consultant works on a specific client problem)

1. **Key1 → {Client, Problem}**
2. **Key2 → {Client, Consultant}**
3. **No non-key attribute hence 3NF**
4. **{Client, Problem} → {Consultant}**
5. **{Client, Consultant} → {Problem}**
6. **Dependency between attributes belonging to keys violates BCNF**

# Normalization

## BCNF: Decomposition

1. Place the two candidate primary keys in separate entities
2. Place each of the remaining data items in one of the resulting entities according to its dependency on the primary key.

### Example 1 (Convert to BCNF)

**Old Scheme → {City, Street, ZipCode }**

**New Scheme1 → {ZipCode, Street}**

**New Scheme2 → {City, Street}**

- **Loss of relation {ZipCode} → {City}**

**Alternate New Scheme1 → {ZipCode, Street }**

**Alternate New Scheme2 → {ZipCode, City}**

# Normalization

## Decomposition: Loss of Information

1. If decomposition does not cause any loss of information it is called a **lossless** decomposition.
2. If a decomposition does not cause any dependencies to be lost it is called a **dependency-preserving** decomposition.
3. Any table scheme can be decomposed in a lossless way into a collection of smaller schemas that are in BCNF form. However the dependency preservation is not guaranteed.
4. Any table can be decomposed in a lossless way into 3rd normal form that also preserves the dependencies.
   - 3NF may be better than BCNF in some cases

Use your own judgment when decomposing schemas

# Normalization

## BCNF: Decomposition

### Example 2 (Convert to BCNF)

Old Scheme → {MovieTitle, StudioID, MovieID, ActorName, Role, Payment }

New Scheme → {MovieID, ActorName, Role, Payment}

New Scheme → {MovieTitle, StudioID, ActorName}

- **Loss of relation {MovieID} → {MovieTitle}**

New Scheme → {MovieID, ActorName, Role, Payment}

New Scheme → {MovieID, MovieTitle}

- **We got the {MovieID} → {MovieTitle} relationship back**

### Example 3 (Convert to BCNF)

Old Scheme → {Client, Problem, Consultant}

New Scheme → {Client, Consultant}

New Scheme → {Client, Problem}

**Loss or Relation {Consultant, Problem}**

New Schema → {Client, Consultant}

New Schema → {Consultant, Problem}

# Normalization

## Fourth Normal Form (4NF)

- Fourth normal form eliminates independent many-to-one relationships between columns.
- To be in Fourth Normal Form,
  - a relation must first be in Boyce-Codd Normal Form.
  - a given relation may not contain more than one multi-valued attribute.

## Example (Not in 4NF)

Scheme → {MovieName, ScreeningCity, Genre)

Primary Key: {MovieName, ScreeningCity, Genre)

1. All columns are a part of the only candidate key, hence BCNF
2. Many Movies can have the same Genre
3. Many Cities can have the same movie
4. Violates 4NF

| Movie | ScreeningCity | Genre |
|---|---|---|
| Hard Code | Los Angles | Comedy |
| Hard Code | New York | Comedy |
| Bill Durham | Santa Cruz | Drama |
| Bill Durham | Durham | Drama |
| The Code Warrier | New York | Horror |

# Normalization

## Fourth Normal Form (4NF)

### Example 2 (Not in 4NF)

**Scheme → {Manager, Child, Employee}**

1. **Primary Key → {Manager, Child, Employee}**
2. **Each manager can have more than one child**
3. **Each manager can supervise more than one employee**
4. **4NF Violated**

| Manager | Child | Employee |
|---|---|---|
| Jim | Beth | Alice |
| Mary | Bob | Jane |
| Mary | NULL | Adam |

### Example 3 (Not in 4NF)

**Scheme → {Employee, Skill, ForeignLanguage}**

1. **Primary Key → {Employee, Skill, Language }**
2. **Each employee can speak multiple languages**
3. **Each employee can have multiple skills**
4. **Thus violates 4NF**

| Employee | Skill | Language |
|---|---|---|
| 1234 | Cooking | French |
| 1234 | Cooking | German |
| 1453 | Carpentry | Spanish |
| 1453 | Cooking | Spanish |
| 2345 | Cooking | Spanish |

# Normalization

## 4NF: Decomposition

1. Move the two multi-valued relations to separate tables
2. Identify a primary key for each of the new entity.

### Example 1 (Convert to 4NF)

**Old Scheme → {MovieName, ScreeningCity, Genre}**

**New Scheme → {MovieName, ScreeningCity}**

**New Scheme → {MovieName, Genre}**

| Movie | Genre |
|---|---|
| Hard Code | Comedy |
| Bill Durham | Drama |
| The Code Warrier | Horror |

| Movie | ScreeningCity |
|---|---|
| Hard Code | Los Angles |
| Hard Code | New York |
| Bill Durham | Santa Cruz |
| Bill Durham | Durham |
| The Code Warrier | New York |

# Normalization

## 4NF: Decomposition

### Example 2 (Convert to 4NF)

**Old Scheme → {Manager, Child, Employee}**

**New Scheme → {Manager, Child}**

**New Scheme → {Manager, Employee}**

| Manager | Child |
|---|---|
| Jim | Beth |
| Mary | Bob |

| Manager | Employee |
|---|---|
| Jim | Alice |
| Mary | Jane |
| Mary | Adam |

### Example 3 (Convert to 4NF)

**Old Scheme → {Employee, Skill, ForeignLanguage}**

**New Scheme → {Employee, Skill}**

**New Scheme → {Employee, ForeignLanguage}**

| Employee | Skill |
|---|---|
| 1234 | Cooking |
| 1453 | Carpentry |
| 1453 | Cooking |
| 2345 | Cooking |

| Employee | Language |
|---|---|
| 1234 | French |
| 1234 | German |
| 1453 | Spanish |
| 2345 | Spanish |

# Normalization

## Fifth Normal Form (5NF)

- Fifth normal form applies to M-Way relationships.
- In 5NF all tables are broken into as many tables as possible in order to avoid redundancy.
- Once it is in fifth normal form it cannot be broken into smaller relations without changing the facts or the meaning.

# Normalization

## Domain Key Normal Form (DKNF)

- A relation is in DKNF if all constraints and dependencies on the relation can be enforced by enforcing the domain constraints and key constraints on the relation.
  - A domain is the set of permissible values for an attribute.
- By enforcing key and domain restrictions, the database is assured of being freed from modification (insertion & deletion) anomalies.
- Designed to specify the "ultimate normal form" which uses all possible types of dependencies and constraints.
  - DKNF is the normalization level that most designers aim to achieve.
  - The practical utility of DKNF is limited, because it is difficult to specify general integrity constraints.
- It has been shown that a relation in DKNF is in 5NF and that DKNF is not always achievable.

# Normalization

## Domain Key Normal Form (DKNF)

- Example (Relations with complex constraints)
  - CAR = {MAKE, VIN#}, MANUFACTURE = {VIN#, COUNTRY} where COUNTRY is the country where the car was manufactured.
  - A complex constraint is For a Toyota or Lexus made in Japan, the first character of the VIN# is a "J"; for a Honda or Acura made in Japan, the second character of the VIN# is a "J".
- Example (Normalization)
  - R = {BRANCH, ACCTNUM, BALANCE}
  - Constraint: An ACCTNUM beginning with 9 is a special account which requires a minimum balance of $2,500.
  - R is not in DKNF.
  - Replace R by the decomposition D = {R1, R2} where R1 = {BRANCH, ACCTNUM, BALANCE} with the constraint that an ACCTNUM does not begin with 9 and R2 = {BRANCH, ACCTNUM, BALANCE} with the constraints that an ACCTNUM begins with 9 and the BALANCE is greater than or equal to 2500.
  - D is in DKNF.

# Normalization

## Summary

- Different Stages of Normalization

| | |
|---|---|
| **1NF** | Keys;  No repeating groups |
| **2NF** | No partial dependencies |
| **3NF** | No transitive dependencies |
| **BCNF** | Determinants are candidate keys |
| **4NF** | No multivalued dependencies |
| **5NF** | Remove m-way relationships |
| **DKNF** | Use domain constraints to enforce dependencies |