Normalization

Database Normalization

* Process to organize data to minimize redundancy (data duplication), which results in data consistency
* There are 6 normal forms

**Normalization 1.**

**Table is in 1NF:**

1. The data in each column should be atomic. **No multiple values, separated by comma**.
2. The table does **not contain** any **repeating column groups**
3. **Identify** each record **uniquely** using **primary key**

**Problems:**

* Disk space wastage
* Data inconsistency
* DML queries can be very slow (***CRUD - Create, read, update & delete***)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| employee\_name | gender | salary | dept\_name | dept\_head | dept\_location |
| Sam | Male | 4500 | IT | John | London |
| Pam | Female | 2300 | HR | Mike | Sydney |
| Simon | Male | 1345 | IT | John | London |
| Mary | Female | 2567 | HR | Mike | Sydney |
| Todd | Male | 6890 | IT | John | London |

**Solution:**

* Smaller size (example: John takes 4 bytes, while 1 takes 1 byte)
* Simple replacing (example: dept\_head)
* Non repeating data (example: dept\_head & dept\_location)

|  |  |  |  |
| --- | --- | --- | --- |
| dept\_id | dept\_name | dept\_head | dept\_location |
| 1 | IT | John | London |
| 2 | HR | Mike | Sydney |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| employee\_id | employee\_name | gender | salary | dept\_id |
| 1 | Sam | Male | 4500 | 1 |
| 2 | Pam | Female | 2300 | 2 |
| 3 | Simon | Male | 1345 | 1 |
| 4 | Mary | Female | 2567 | 2 |
| 5 | Todd | Male | 6890 | 1 |

**Problem (non atomic solumns):**

* It is not possible to SELECT, INSERT, UPDATE & DELETE just one employee

|  |  |
| --- | --- |
| dept\_name | employee |
| IT | Sam, Mike, Shaun |
| HR | Pam |

**Problem (repeating column groups):**

* It is not possible to SELECT, INSERT, UPDATE & DELETE just one employee
* Non effective storage, lots of 0 data (waste disc space)

|  |  |  |  |
| --- | --- | --- | --- |
| dept\_name | employee\_1 | employee\_2 | employee\_3 |
| IT | Sam | Mike | Shaun |
| HR | Pam |  |  |

**Solution:**

* Apply rules of the 1NF

|  |  |
| --- | --- |
| dept\_id | dept\_name |
| 1 | IT |
| 2 | HR |

|  |  |
| --- | --- |
| dept\_id | employee |
| 1 | Sam |
| 1 | Mike |
| 1 | Shaun |
| 2 | Pam |

**Normalization 2.**

**Table is in 2NF**

1. Table meets all the condition s of 1NF
2. Move **redundant data to a separate table**
3. Create **relationships** between tables **using foreign keys**

**Problems:**

* Disk space wastage
* Data inconsistency
* DML queries can be very slow (***CRUD - Create, read, update & delete***)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| emp\_id | employer\_name | gender | salary | dept-name | dept\_head | dept\_location |
| 1 | Sam | Male | 4500 | IT | John | London |
| 2 | Pam | Female | 2300 | HR | Mike | Sydney |
| 3 | Simon | Male | 1345 | IT | John | London |
| 4 | Mary | Female | 2567 | HR | Mike | Sydney |
| 5 | Todd | Male | 6890 | IT | John | London |

**Solution:**

* dept\_id (primary\_key & foreing\_key)

|  |  |  |  |
| --- | --- | --- | --- |
| dept\_id | dept\_name | dept\_head | dept\_location |
| 1 | IT | John | London |
| 2 | HR | Mike | Sydney |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| employee\_id | employee\_name | gender | salary | dept\_id |
| 1 | Sam | Male | 4500 | 1 |
| 2 | Pam | Female | 2300 | 2 |
| 3 | Simon | Male | 1345 | 1 |
| 4 | Mary | Female | 2567 | 2 |
| 5 | Todd | Male | 6890 | 1 |

**Normalization 3.**

**Table is in 3NF:**

1. Meets all conditions of 1NF & 2NF
2. Does not contain columns (attributes) that are not **fully dependent** upon the **primary key**
3. Improves database processing while minimizing storage costs

**Problems:**

* Column annual\_salary (we can not compute annual salary)
* emp\_id is fully dependent on dept\_name & department head

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| emp\_id | employer\_name | gender | salary | annual\_salary | dept\_id |
| 1 | Sam | Male | 4500 | 54000 | 1 |
| 2 | Pam | Female | 2300 | 27600 | 2 |
| 3 | Simon | Male | 1345 | 16140 | 1 |
| 4 | Mary | Female | 2567 | 30804 | 2 |
| 5 | Todd | Male | 6890 | 82680 | 1 |

**Solution:**

|  |  |  |
| --- | --- | --- |
| dept\_id | dept\_name | dept\_head |
| 1 | IT | John |
| 2 | HR | Mike |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| employee\_id | employee\_name | gender | salary | dept\_id |
| 1 | Sam | Male | 4500 | 1 |
| 2 | Pam | Female | 2300 | 2 |
| 3 | Simon | Male | 1345 | 1 |
| 4 | Mary | Female | 2567 | 2 |
| 5 | Todd | Male | 6890 | 1 |

**Normalization 4.**

**Table is in 4NF:**

1. has more general type of dependency known as a multivalued dependency
   * multivalued dependency:
     + full constraint between two sets of attributes in a relation
     + requires that certain finite ordered list of elements be present in a relation

**Problems:**

|  |  |  |
| --- | --- | --- |
| restaurant | pizza\_variety | delivery\_area |
| A1 Pizza | Thick Crust | Springfield |
| A1 Pizza | Thick Crust | Shelbyville |
| A1 Pizza | Thick Crust | Capital City |
| A1 Pizza | Stuffed Crust | Springfield |
| A1 Pizza | Stuffed Crust | Shelbyville |
| A1 Pizza | Stuffed Crust | Capital City |
| Elite Pizza | Thin Crust | Capital City |
| Elite Pizza | Stuffed Crust | Capital City |
| Vincenzo's Pizza | Thick Crust | Springfield |
| Vincenzo's Pizza | Thick Crust | Shelbyville |
| Vincenzo's Pizza | Thin Crust | Springfield |
| Vincenzo's Pizza | Thin Crust | Shelbyville |

**Solution:**

|  |  |
| --- | --- |
| restaurant | pizza\_variety |
| A1 Pizza | Thick Crust |
| A1 Pizza | Stuffed Crust |
| Elite Pizza | Thin Crust |
| Elite Pizza | Stuffed Crust |
| Vincenzo's Pizza | Thick Crust |
| Vincenzo's Pizza | Thin Crust |

|  |  |
| --- | --- |
| restaurant | delivery\_area |
| A1 Pizza | Springfield |
| A1 Pizza | Shelbyville |
| A1 Pizza | Capital City |
| Elite Pizza | Capital City |
| Vincenzo's Pizza | Springfield |
| Vincenzo's Pizza | Shelbyville |

**Normalization 5.**

**Table is in 5NF:**

1. every non-trivial join dependency in it is implied by the candidate keys
   * level of database normalization designed to reduce redundancy in relational databases recording multi-valued facts by isolating semantically related multiple relationships

**Problems:**

|  |  |  |
| --- | --- | --- |
| traveling\_salesman | brand | product\_type |
| Jack Schneider | Acme | Vacuum Cleaner |
| Jack Schneider | Acme | Breadbox |
| Mary Jones | Robusto | Pruning Shears |
| Mary Jones | Robusto | Vacuum Cleaner |
| Mary Jones | Robusto | Breadbox |
| Mary Jones | Robusto | Umbrella Stand |
| Louis Ferguson | Robusto | Vacuum Cleaner |
| Louis Ferguson | Robusto | Telescope |
| Louis Ferguson | Acme | Vacuum Cleaner |
| Louis Ferguson | Acme | Lava Lamp |
| Louis Ferguson | Nimbus | Tie Rack |

**Solution:**

|  |  |
| --- | --- |
| traveling\_salesman | product\_type |
| Jack Schneider | Vacuum Cleaner |
| Jack Schneider | Breadbox |
| Mary Jones | Pruning Shears |
| Mary Jones | Vacuum Cleaner |
| Mary Jones | Breadbox |
| Mary Jones | Umbrella Stand |
| Louis Ferguson | Telescope |
| Louis Ferguson | Vacuum Cleaner |
| Louis Ferguson | Lava Lamp |
| Louis Ferguson | Tie Rack |

|  |  |
| --- | --- |
| traveling\_salesman | brand |
| Jack Schneider | Acme |
| Mary Jones | Robusto |
| Louis Ferguson | Robusto |
| Louis Ferguson | Acme |
| Louis Ferguson | Nimbus |

|  |  |
| --- | --- |
| brand | product\_type |
| Acme | Vacuum Cleaner |
| Acme | Breadbox |
| Acme | Lava Lamp |
| Robusto | Pruning Shears |
| Robusto | Vacuum Cleaner |
| Robusto | Breadbox |
| Robusto | Umbrella Stand |
| Robusto | Telescope |
| Nimbus | Tie Rack |

**Normalization 6.**

Not actually necessary