**Primary Key:**

* A field that is used to uniquely identify each row in a table.
* It uniquely identifies each record
* **Logical Primary Key:**

The key we use in queries to look up records from ad-hoc queries.

Eg: SELECT \* FROM users WHERE email = $email;

**Foreign Key:**

* A primary key from another table, used to reference a unique record in that other table.

**Composite Primary Key:**

* Composite key, or composite primary key, refers to cases where more than one column is used to specify the primary key of a table.
* In such cases, all foreign keys will also need to include all the columns in the composite key. ­­

**Surrogate Primary Key**

1. Auto incrementation
2. Arbitrary value
3. Generally, Integer
4. Has no real relation with the other attributes of the table

**Candidate Keys:**

* A Candidate Key can be any column or a combination of columns that can qualify as unique key in database.

**Natural Key**

* A key that is formed of attributes that already exist in the real world.
* For example, U.S. citizens are issued a Social Security Number (SSN) that is unique to them (this isn't guaranteed to be true, but it's pretty darn close in practice).  SSN could be used as a natural key, assuming privacy laws allow it, for a *Person* entity (assuming the scope of your organization is limited to the U.S.).

**Insert Anomaly:**

* An **Insert Anomaly** occurs when certain attributes cannot be **inserted** into the database without the presence of other attributes. For example, this is the converse of delete **anomaly** - we can't add a new course unless we have at least one student enrolled on the course.

**Update Anomaly:**

* We have an update anomaly when we cannot update a single value without having to update in multiple places.

**Delete Anomaly:**

* We have a delete anomaly when we cannot delete one piece of data without advertently deleting other, unrelated data (that we may not want to delete).
* For example, consider what happens if Student S30 is the last student to leave the course - All information about the course is lost.

**Normalization:**

* Is the process of decomposing larger tables into smaller related tables, for the purpose of reducing anomalies.
* It was first proposed by Edgar F. Codd
* He defined what are the stages of normalization

1 NF

2 NF

3 NF

4 NF (Boyce Codd Normal Form)

* Lossless – reconstitute entire original table with queries

**First Normal Form:**

* All Values must be atomic, i.e a single value
* Example:

|  |  |  |  |
| --- | --- | --- | --- |
| **Book\_ID** | **Title** | **Author** | **Num\_Pages** |
| 1 | Dune | Frank Herbert | 566 |
| 2 | Dune World | Frank Herbert,  John Williams | 722 |
| 3 | Space | Mike Moune | 501 |

After first normalization:

|  |  |  |  |
| --- | --- | --- | --- |
| **Book\_ID** | **Title** | **Author** | **Num\_Pages** |
| 1 | Dune | Frank Herbert | 566 |
| 2 | Dune World | Frank Herbert,  John Williams | 722 |
| 3 | Dune World | John Williams | 722 |
| 4 | Space | Mike Moune | 501 |

**Second Normal Form:**

* No partial dependencies
* **Functional Dependency:**

When one attribute determines the value of another attribute

Example: Title -> Author

* **Example of 2 NF:**

|  |  |  |
| --- | --- | --- |
| Name | Email | Service Provider |
| John | [john@hotmail.com](mailto:john@hotmail.com) | Hotmail |
| Peter | [peter@gmail.com](mailto:peter@gmail.com) | Gmail |
| John | [johns@gmail.com](mailto:johns@gmail.com) | Gmail |
| Sally | [johns@gmail.com](mailto:johns@gmail.com) | Gmail |

* In this above table, to convert it into 2 NF we remove the service provider from the mail table and create another service provider table.

**Third Normal Form:**

* 2 NF and no transitive dependencies
* Transitive dependency is A -> B, B->C then A->C
* emp( [emp\_id], emp\_email, dept, dept\_mgr)
* In the above example,
* emp\_id -> dept, dept -> dept\_mgr, emp\_id -> dept\_mgr via dept

**In-class Work:**

* ([ibsn], title, num\_pages, price, author, author\_country, publisher, publisher\_phone)
* All values are forced to be atomic
* Author can write many books
* One book can be written by one author
* Publisher can publish many books
* Book is published by one publisher

**1NF**

* Already in 1NF

**2NF**

* Already in 2NF

**3NF**

* Book([isbn], title, num\_pages, price, author, publisher)

Author ([author], author\_country)

Publisher ([publisher], publisher\_phone)

* Book ([isbn], title, num\_pages, price, author\_id, publisher\_id)

Author ([author\_id], author, author\_country)

Publisher ([publisher\_id], publisher, publisher\_phone)

**BCNF:**

* Employee (name, email, dept, dept\_mgr, dept\_mgr\_email)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Email** | **Dept** | **Dept\_Mgr** | **Dept\_Mgr\_email** |
| Tom Jones | [tom@x.com](mailto:tom@x.com) | Sales | Bill Henry | [bill@x.com](mailto:bill@x.com) |
| Peter Firth | [peter@x.com](mailto:peter@x.com) | Sales | Bill henry | [bill@x.com](mailto:bill@x.com) |
| Tom Jones | [peter@x.com](mailto:peter@x.com) | Accounting | Frank Moore | [frank@x.com](mailto:frank@x.com) |

**PK = [pname, email, dept]**

* **This table is in 1 NF**
* **Is it in 2 NF?** 
  + No, Dept\_mgr can be determined by department alone, so we have a partial dependency
* **So, we decompose the primary table into the below tables**

Employee ([name, email, dept])

Dept ([dept], dept\_mgr, dept\_mgr\_email)

* The department table **isn’t in 3 NF** because dept -> **dept\_mgr\_email** via **dept\_mgr**
* **So, we decompose it into two different tables**

1. Dept([dept], dept\_mgr)
2. Dept\_mgr ([dept\_mgr], dept\_mgr\_email)

* **All tables are now in 3 NF**

**BCNF:**

* There can be no multi valued dependencies

**Intro DBA:**

* Create DB
* Create tables
* Joins
* Queries
* Alter Tables

**Intermediate DBA:**

* Security
* Grant privileges
* Revoke privileges

**Advanced DBA:**

* Normalize
* Intuitive
* Stored Procedures
* Functions
* Variables