Data Integrity

The term data integrity refers to the accuracy and consistency of data.

When creating databases, attention needs to be given to data integrity and how to maintain it. A good database will enforce data integrity whenever possible.

For example, a user could accidentally try to enter a phone number into a date field. If the system enforces data integrity, it will prevent the user from making these mistakes.

Maintaining data integrity means making sure the data remains intact and unchanged throughout its entire life cycle. This includes the capture of the data, storage, updates, transfers, backups, etc. Every time data is processed there's a risk that it could get corrupted (whether accidentally or maliciously).

Risks to Data Integrity

Some more examples of where data integrity is at risk:

- A user tries to enter a date outside an acceptable range.
- A user tries to enter a phone number in the wrong format.
- A bug in an application attempts to delete the wrong record.
- While transferring data between two databases, the developer accidentally tries to insert the data into the wrong table.
- While transferring data between two databases, the network went down.
- A user tries to delete a record in a table, but another table is referencing that record as part of a relationship.
- A user tries to update a primary key value when there's already a foreign key in a related table pointing to that value.

Types of Data Integrity

In the database world, data integrity is often placed into the following types:

- Domain Constraints
- Entity integrity
- Referential integrity
- Key Constraint

1. Domain constraints

  - Domain constraints can be defined as the definition of a valid set of values for an attribute.

- The data type of domain includes string, character, integer, time, date, currency, etc. The value of the attribute must be available in the corresponding domain.

Example:

ID	NAME	SEMENSTER	AGE
1000	Tom	1st	17
1001	Johnson	2nd	24
1002	Leonardo	5th	21
1003	Kate	3rd	19
1004	Morgan	8th	A

Not allowed. Because AGE is an integer attribute

2. Entity integrity constraints

- The entity integrity constraint states that primary key value can't be null.
- This is because the primary key value is used to identify individual rows in relation and if the primary key has a null value, then we can't identify those rows.
- A table can contain a null value other than the primary key field.

Example:

EMPLOYEE

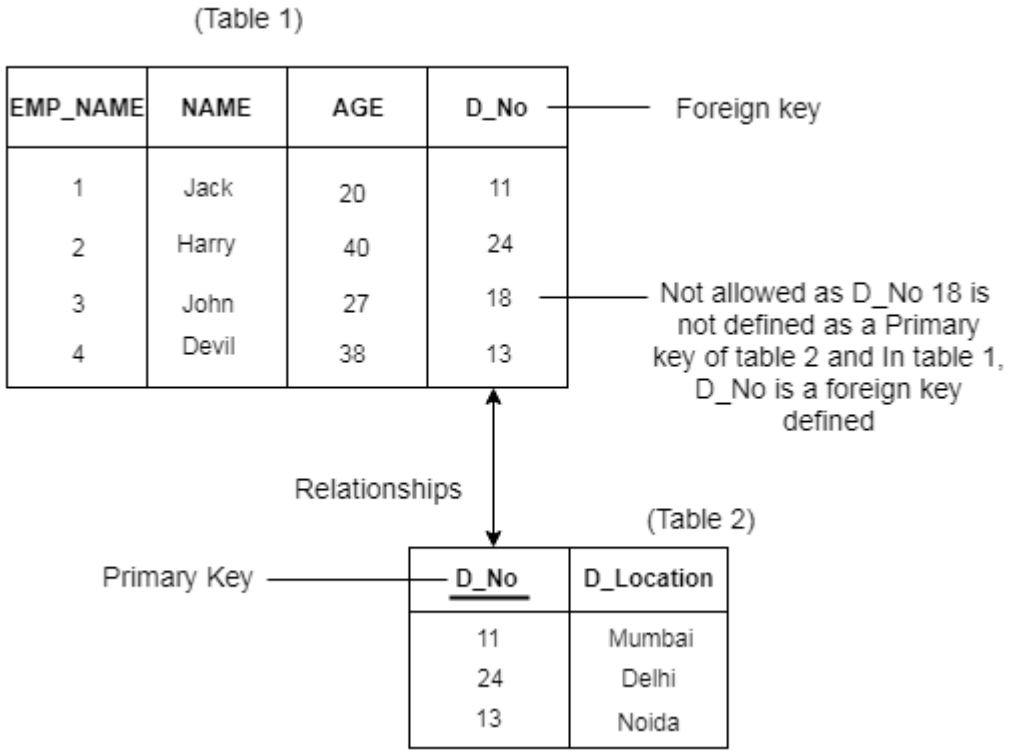
EMP_ID	EMP_NAME	SALARY
123	Jack	30000
142	Harry	60000
164	John	20000
	Jackson	27000

Not allowed as primary key can't contain a NULL value

3. Referential Integrity Constraints

- A referential integrity constraint is specified between two tables.
- In the Referential integrity constraints, if a foreign key in Table 1 refers to the Primary Key of Table 2, then every value of the Foreign Key in Table 1 must be null or be available in Table 2.

Example:



4. Key constraints

- Keys are the entity set that is used to identify an entity within its entity set uniquely.
- An entity set can have multiple keys, but out of which one key will be the primary key. A primary key can contain a unique and null value in the relational table.

Example:

ID	NAME	SEMENSTER	AGE
1000	Tom	1st	17
1001	Johnson	2nd	24
1002	Leonardo	5th	21
1003	Kate	3rd	19
1002	Morgan	8th	22

Not allowed. Because all row must be unique

Database security

Database security encompasses a range of security controls designed to protect the Database Management System (DBMS). The types of database security measures your business should use include protecting the underlying infrastructure that houses the database such as the network and servers), securely configuring the DBMS, and the access to the data itself.

Database security controls

Database security encompasses multiple controls, including system hardening, access, DBMS configuration, and security monitoring. These different security controls help to manage the circumventing of security protocols.

System hardening and monitoring

The underlying architecture provides additional access to the DBMS. It is vital that all systems are patched consistently, hardened using known security configuration standards, and monitored for access, including insider threats.

DBMS configuration

It is critical that the DBMS be properly configured and hardened to take advantage of security features and limit privileged access that may cause a misconfiguration of expected security settings. Monitoring the DBMS configuration and ensuring proper change control processes helps ensure that the configuration stays consistent.

Authentication

Database security measures include authentication, the process of verifying if a user's credentials match those stored in your database, and permitting only authenticated users access to your data, networks, and database platform.

Access

A primary outcome of database security is the effective limitation of access to your data. Access controls authenticate legitimate users and applications, limiting what they can access in your database. Access includes designing and granting appropriate user attributes and roles and limiting administrative privileges.

Database auditing

Monitoring (or auditing) actions as part of a database security protocol delivers centralized oversight of your database. Auditing helps to detect, deter, and reduce the overall impact of unauthorized access to your DBMS.

Backups

A data backup, as part of your database security protocol, makes a copy of your data and stores it on a separate system. This backup allows you to recover lost data that may result from hardware failures, data corruption, theft, hacking, or natural disasters.

Encryption

Database security can include the secure management of encryption keys, protection of the encryption system, management of a secure, off-site encryption backup, and access restriction protocols.

Application security

Database and application security framework measures can help protect against common known attacker exploits that can circumvent access controls, including SQL injection.

Why is database security important?

Safeguarding the data your company collects and manages is of utmost importance. Database security can guard against a compromise of your database, which can lead to financial loss, reputation damage, consumer confidence disintegration, brand erosion, and non-compliance of government and industry regulation.

Database security safeguards defend against a myriad of security threats and can help protect your enterprise from:

- Deployment failure
- Excessive privileges
- Privilege abuse
- Platform vulnerabilities
- Unmanaged sensitive data
- Backup data exposure
- Weak authentication
- Database injection attacks

Views Management:

D*atabase view* is a subset of a database and is based on a query that runs on one or more database tables. Database views are saved in the database as named queries and can be used to save frequently used, complex queries.

There are two types of database views: dynamic views and static views. Dynamic views can contain data from one or two tables and automatically include all of the columns from the specified table or tables. Dynamic views are automatically updated when related objects or extended objects are created or changed. Static views can contain data from multiple tables and the required columns from these tables must be specified in the SELECT and WHERE clauses of the static view. Static views must be manually updated when related objects or extended objects are created or changed.

When you create a dynamic view with data from two tables, you must ensure that both tables have the same PRIMARYKEYCOLSEQ columns or contain unique indexes with the same column name in the same order.

Database views are populated depending on the object on which they are based. For example, if you add or remove an attribute from the WORKORDER object, the attribute is either added or removed from the dynamic view that is based on the object. When you change an attribute, not all changes are applied to the associated database view. For example, if you change the data type of an attribute, the change is applied to the database view. However, if you change or add a domain to the default value of the WORKORDER object, the change is not automatically applied to the database view. Instead, you must apply this change to the database view.