



**CE246: Database Management System**  
Dec – May 2021-22

# Database Security



**Devang Patel Institute of Advance Technology and Research**

# What are the Views?

- Views in SQL are considered as a virtual table
- View also has set of records in the form of rows and columns. But it is created based on the records in one or more tables. **A query is written on table/s and is given a name. Such named query is called as a view.**
- A table will have large number of data and table will be fired with specific frequently. In such case, instead of rewriting the query again and again, a name is given to the query and it will be called whenever it is required.
- Hence view is also called as named query or stored query.
- **Unlike ordinary base tables in a relational database, a view does not form part of the physical schema: as a result set, it is a virtual table computed or collated dynamically from data in the database when access to that view is requested.**
- Changes applied to the data in a relevant underlying table are reflected in the data shown in subsequent invocations of the view.

# What are the Views?

- When we create a view, only the query used to create the view is stored with view name.
- It does not create any copy of data in a separate file.
- In other words, views are virtual tables.
- When we fire query on these views, the underlying view query will be executed.
- Hence it does not consume any space in memory to store the data nor does it create same copies of data in the database. It simply shows the records from the table itself.
- To create the view, we can select the fields from one or more tables present in the database.
- A view can either have specific rows based on certain condition or all the rows of a table.

# View

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY
100	Steven	King	SKING	515.123.4567	17-JUN-87	AD_PRES	24000
101	Neena	Kochhar	NKOCHHAR	515.123.4568	21-SEP-89	AD_VP	17000
102	Lex	De Haan	LDEHAAN	515.123.4569	13-JAN-93	AD_VP	17000
103	Alexander	Hunold	AHUNOLD	590.423.4567	03-JAN-90	IT_PROG	9000
104	Bruce	Ernst	BERNST	590.423.4568	21-MAY-91	IT_PROG	6000
107	Diana	Lorentz	DLORENTZ	590.423.5567	07-FEB-99	IT_PROG	4200
124	Kevin	Mourgos	KMOURGOS	650.123.5234	16-NOV-99	ST_MAN	5800
141	Trenna	Rajs	TRAJS	650.121.8009	17-OCT-95	ST_CLERK	3500
142	Curtis	Davies	CDAVIES	650.121.2994	29-JAN-97	ST_CLERK	3100
143	Randall	Matos	RMATOS	650.121.2874	15-MAR-98	ST_CLERK	2600
EMPLOYEE_ID	LAST_NAME		SALARY		JUL-98	ST_CLERK	2500
149	Zlotkey		10500		JAN-00	SA_MAN	10500
174	Abel		11000		MAY-96	SA_REP	11000
176	Taylor		8600		MAR-98	SA_REP	8600
170	Kimberely	Grant	KGRANT	515.124.1044	24-MAY-99	SA_REP	7000
200	Jennifer	Whalen	JWHALEN	515.123.4444	17-SEP-87	AD_ASST	4400
201	Michael	Hartstein	MHARTSTE	515.123.5555	17-FEB-96	MK_MAN	13000
202	Pat	Fay	PFAY	603.123.6666	17-AUG-97	MK_REP	6000
205	Shelley	Higgins	SHIGGINS	515.123.8080	07-JUN-94	AC_MGR	12000
206	William	Gietz	WGIETZ	515.123.8181	07-JUN-94	AC_ACCOUNT	8300

# Advantage of View

- The basic advantage of view is it simplifies the query.
- It replaces the large and frequently used query as single table and allows querying it as any other table.
- In addition to this, we can give different accesses to different users on this view like we do it on table.
- When we give control access on the view, users will have access on only the portion of the table.
- Suppose we have created Design\_Emp and Test\_Emp view and given SELECT grants to the Design employees on Design\_Emp and Test employees on Test\_Emp. Now design employees will not be able to see the content of Test\_Emp view and vise versa.
- Even though users are given access on the views, they will not have access on the EMPLOYEE and DEPT tables. They will have access only on the columns and rows which represent the view. For example, ADDRESS column in EMPLOYEE table is not part of any view and Design or Testing employees will not be able see the ADDRESS column value.

# Advantage of View

- Views can represent a subset of the data contained in a table.
- Views can join and simplify multiple tables into a single virtual table.
- Views can act as aggregated tables, where the database engine aggregates data (sum, average, etc.) and presents the calculated results as part of the data.
- Views can hide the complexity of data.
- For example, a view could appear as Sales2000 or Sales2001, transparently partitioning the actual underlying table.
- Views take very little space to store; the database contains only the definition of a view, not a copy of all the data that it presents.
- Depending on the SQL engine used, views can provide extra security

# Creating a View

- A view can be created using the CREATE VIEW statement. We can create a view from a single table or multiple tables.

- **Syntax:**

```
CREATE VIEW view_name AS
```

```
SELECT column1, column2.....
```

```
FROM table_name
```

```
WHERE condition;
```

# Creating a View from a single table

**Student\_Detail**

STU_ID	NAME	ADDRESS
1	Stephan	Delhi
2	Kathrin	Noida
3	David	Ghaziabad
4	Alina	Gurugram

**Student\_Marks**

STU_ID	NAME	MARKS	AGE
1	Stephan	97	19
2	Kathrin	86	21
3	David	74	18
4	Alina	90	20
5	John	96	18



# Creating a View from a single table

## Example:

```
CREATE VIEW Details_View AS  
SELECT NAME, ADDRESS  
FROM Student_Detail  
WHERE STU_ID < 4;
```

- Just like table query, we can query the view to view the data.

# Creating a View from a single table

- To display the details of a view, execute following query.

```
SELECT * FROM Details_View;
```

Output:

**Output:**

NAME	ADDRESS
Stephan	Delhi
Kathrin	Noida
David	Ghaziabad

# Creating a View from multiple tables

- View from multiple tables can be created by simply include multiple tables in the SELECT statement.
- In the given example, a view is created named MarksView from two tables Student\_Detail and Student\_Marks.

# Creating a View from multiple tables

## Example:

```
CREATE VIEW MarksView AS  
  
SELECT Student_Detail.NAME, Student_Detail.ADDRESS, Student_Marks.MARKS  
  
FROM Student_Detail JOIN Student_Marks  
  
WHERE Student_Detail.NAME = Student_Marks.NAME;
```

- **To display data from a view**
- `SELECT * FROM MarksView;`

# Creating a View from multiple tables

Output:

NAME	ADDRESS	MARKS
Stephan	Delhi	97
Kathrin	Noida	86
David	Ghaziabad	74
Alina	Gurugram	90

# Deleting a View

- A view can be deleted using the Drop View statement.

**DROP VIEW view\_name;**

- Example:
- If we want to delete the View **MarksView**, we can do this as:

**DROP VIEW MarksView;**

# CREATE or REPLACE a View (Update View)

- We can use the **CREATE OR REPLACE VIEW** statement to add or remove fields from a view.
- Syntax:

```
CREATE OR REPLACE VIEW view_name AS  
SELECT column1,coulmn2,..  
FROM table_name  
WHERE condition;
```

# CREATE or REPLACE a View

- **Example:** if we want to update the view MarksView and add the field AGE to this View from Student\_Marks Table, we can do this as:

```
CREATE OR REPLACE VIEW MarksView AS
```

```
SELECT  Student_Detail.NAME,  Student_Detail.ADDRESS,  Student_Marks.MARKS,  
Student_Marks.Age
```

```
FROM Student_Details JOIN Student_Marks
```

```
WHERE Student_Detail.NAME = Student_Marks.NAME;
```

- This will replace previously created view according to the condition mentioned in the select statement



# CREATE or REPLACE a View

If we fetch all the data from MarksView now as:

```
SELECT * FROM MarksView;
```

# Inserting a row in a view

- We can insert a row in a View in a same way as we do in a table.
- We can use the INSERT INTO statement of SQL to insert a row in a View.

- Syntax:

**INSERT INTO view\_name(column1, column2 , column3,..)**

**VALUES(value1, value2, value3..);**

# Inserting a row in a view

- **Example:**
- **In the below example we will insert a new row in the View Details\_View which we have created above in the example of “creating views from a single table”.**

```
INSERT INTO Details_View(NAME, ADDRESS)  
VALUES("Suresh","Gurgaon");
```

- **If we fetch all the data from DetailsView now as,**  

```
SELECT * FROM DetailsView;
```

# Deleting a row from a View

- Deleting rows from a view is also as simple as deleting rows from a table.
- We can use the DELETE statement of SQL to delete rows from a view.
- Also deleting a row from a view first delete the row from the actual table and the change is then reflected in the view

- Syntax:

**DELETE FROM view\_name WHERE condition**

- Example:

**DELETE FROM Details\_View**

**WHERE NAME="Suresh"**

# Questions

- **Question: Does the Oracle View exist if the table is dropped from the database?**
- Answer: Yes, in Oracle, the VIEW continues to exist even after one of the tables (that the Oracle VIEW is based on) is dropped from the database. However, if you try to query the Oracle VIEW after the table has been dropped, you will receive a message indicating that the Oracle VIEW has errors.
- If you recreate the table (the table that you had dropped), the Oracle VIEW will again be fine.

# Materialized View

- A Materialized View persists the data returned from the view definition query and automatically gets updated as data changes in the underlying tables.
- It improves the performance of complex queries (typically queries with joins and aggregations) while offering simple maintenance operations.
- With its execution plan auto matching capability, a materialized view does not have to be referenced in the query for the optimizer to consider the view for substitution.
- **This capability allows data engineers to implement materialized views as a mechanism for improving query response time, without having to change queries.**

# Materialized View

- **Materialized view in SQL is also a logical structure which is stored physically on the disc.**
- **Like a view in Materialized views in SQL we are using simple select statement to create it.**
- You should have create materialized views privileges to create a Materialized views.
- Definition of Materialized views(called as MV) has been stored in databases.
- **Materialized views are useful in Data-warehousing concepts.**
- When you create a Materialized view, Oracle Database creates one internal table and at least one index, and may create one view, all in the schema of the materialized views. Oracle Database uses these objects to maintain the materialized views in SQL data. You must have the privileges necessary to create these objects.

# Materialized View Syntax

- “Materialized views are also know as snapshots..”
- Snapshots acts like a physical table because data from snapshots are storing in to physical memory. Snapshot retrieves data very fast. So for performance tuning Snapshots are used.
- Following is the syntax of materialized view:

```
Create materialized view View_Name  
Build [Immediate/Deffered]  
Refresh [Fast/Complete/Force]  
on [Commit/Demand]  
as Select .....;
```



# Materialized View Syntax

- Using above syntax you can create materialized views. The Syntax includes some different optional fields:
- **Build Immediate:** Means materialized views(mv) created immediately.
- **Build Defferred:** Means materialized views(mv) created after one refresh.
- **Refresh on commit:** This option committed the data in materialized views in SQL immediately after data inserted and committed in table. This option is known as incremental refresh option. View is not fully refreshed with this option
- **Refresh on Demand:** Using this option you can add the condition for refreshing data in materialized views.

# Materialized View Syntax

- Example

```
CREATE MATERIALIZED VIEW MV_Employee
```

```
BUILD immediate
```

```
REFRESH complete
```

```
on commit SELECT * FROM Employee;
```

# Materialized View Real life Example

- Suppose there are 2 tables named Employee and Department. The Employee table contains 1 million records and department table contains 20 records. We need to fetch the Employees associated with that department.
- **Step -1 :**
- To Perform above scenario we basically create view:

```
CREATE View V_Employee AS
```

```
SELECT E.Employee_num,E.Employee_name,D.Department_Name
```

```
FROM Employee E JOIN Department D where E.Dept_no=D.Dept_no;
```

# Materialized View Real life Example

- **Step - 2:**
- Fetch the records from the View

Select \* from V\_Employee;

- It will fetch 10 million records with associated department. But to fetch that records check the time. Let us consider it will take 2 Mins means 120 secs to fetch records

# Materialized View Real life Example

- Step 3 :
- Let us Create materialized view which will refresh automatically.

```
CREATE or REPLACE Materialized view MV_Employee AS  
SELECT E.Employee_num,E.Employee_name,D.Department_Name  
FROM Employee E , Department D where E.Dept_no=D.Dept_no  
REFRESH AUTO on COMMIT SELECT * FROM DEPARTMENT;
```

- We have created materialized views in sql for that.and lets check performance.

```
Select* from MV_Employee;
```

- It will fetch 1 million records in 60 secs. So performance is improved double when you use materialized view.

# Difference between View and Materialized View

View	Materialized Views(Snapshots)
1.View is nothing but the logical structure of the table which will retrieve data from 1 or more table.	1.Materialized views(Snapshots) are also logical structure but data is physically stored in database.
2.You need to have Create view privileges to create simple or complex view	2.You need to have create materialized view 's privileges to create Materialized views
3.Data access is not as fast as materialized views	3.Data retrieval is fast as compare to simple view because data is accessed from directly physical location
4.There are 2 types of views:  1.Simple View  2.Complex view	4.There are following types of Materialized views:  1.Refresh on Auto  2.Refresh on demand
5.In Application level views are used to restrict data from database	5.Materialized Views are used in Data Warehousing.

# Database Security

- Data security is the protection of the data from unauthorized users.
- Only the authorized users are allowed to access the data.
- Most of the users are allowed to access a part of database i.e., the data that is related to them or related to their department.
- Mostly, the DBA or head of department can access all the data in the database.
- Some users may be permitted only to retrieve data, whereas others are allowed to retrieve as well as to update data.

# Database Security

- Database security is a broad area that addresses many issues, including the following:
  1. **Various legal and ethical issues regarding the right to access certain information**

**For example:** some information may be deemed to be private and cannot be accessed legally by unauthorized organizations or persons.

2. **Policy issues at the governmental, institutional, or corporate level regarding what kinds of information should not be made publicly available**

**For example:** credit ratings and personal medical records.



# Database Security

**3. System-related issues such as the system levels at which various security functions should be enforced**

**For example:** whether a security function should be handled at the physical hardware level, the operating system level, or the DBMS level.

**4. The need in some organizations to identify multiple security levels and to categorize the data and users based on these classifications**

**For example:** top secret, secret, confidential, and unclassified. The security policy of the organization with respect to permitting access to various classifications of data must be enforced.

# Threats to Database

- Threats to databases can result in the loss of some or all of the following commonly accepted security goals: **integrity, availability, and confidentiality**.
- **Loss of integrity:** Database integrity refers to the requirement that information be protected from improper modification.
- Modification of data includes creating, inserting, and updating data; changing the status of data; and deleting data.
- Integrity is lost if unauthorized changes are made to the data by either intentional or accidental acts.
- If the loss of system or data integrity is not corrected, continued use of the corrupted data could result in inaccuracy, fraud, or erroneous decisions.

# Threats to Database

- **Loss of availability:** Database availability refers to making objects available to a human user or a program who/which has a legitimate right to those data objects.
- Loss of availability occurs when the user or program cannot access these objects.
- **Loss of confidentiality:** Database confidentiality refers to the protection of data from unauthorized disclosure.
- The impact of unauthorized disclosure of confidential information can range from violation of the Data Privacy Act to the jeopardization of national security.
- Unauthorized, unanticipated, or unintentional disclosure could result in loss of public confidence, embarrassment, or legal action against the organization.

# Threats to Database

- An example of the Confidentiality, Integrity and Availability
- Logging into an e-commerce site to check your orders and make an additional purchase. The e-commerce site uses the three principles of the CIA triad in the following ways:
- **Confidentiality:** When you log in, you're asked for a password. If it's been a while since your last log-in, you may be asked to input a code that's been sent to you or some other form of two-factor authentication.
- **Integrity:** Data integrity is provided by making sure your purchases are reflected in your account and allowing you to contact a representative if there's a discrepancy.
- **Availability:** You can log into your account whenever you want, and you may even be able to contact customer support at any time of the day or night.

# Database Integrity

- Data integrity in the database is the correctness, consistency and completeness of data.
- Data integrity is enforced using the following three integrity constraints:
- **Entity Integrity** - This is related to the concept of primary keys. All tables should have their own primary keys which should uniquely identify a row and not be NULL.
- **Referential Integrity** - This is related to the concept of foreign keys. A foreign key is a key of a relation that is referred in another relation.
- **Domain Integrity** - This means that there should be a defined domain for all the columns in a database.

# Why Data Integrity is important?

- Data integrity is essential because it is a necessary constituent of data integration.
- If data integrity is maintained, data values stored within the database are consistent about the data model and type.
- **Here are some examples of data integrity at risk:**
- An attempt to enter a phone number in the wrong format.
- A developer accidentally tries to insert the data into the wrong table while transferring data between two databases.
- An attempt to delete a record in a table, but another table references that record as part of a relationship.
- A user accidentally tries to enter a phone number into a date field.

# How to Ensure Data Integrity in a Database

- **The common methods used for data integrity check include:**
- Limit access to data and change permissions to constrain modifications to data by unapproved parties.
- Focus on data validation to ensure the accuracy of data when collected or integrated.
- Maintain a regular backup of data.
- Use logs to monitor when data is entered, altered, or erased.
- Conduct systematic internal audits to ensure that information is up to date.

# Types of Integrity

- The integrity refers to the accuracy, consistency and correctness of data present in the database.
- Data integrity is imposed during the database design to make sure that data residing in the database remains complete, accurate and reliable



# Types of Integrity

- Domain integrity
- It ensures that column data in the table adhere to the permissible set of values.
- For example, in the **ORDERS** table, the domain integrity on the **ORDER\_DATE** column ensures that data in this column is always in DATE format.



ORDER_ID	ORDER_TOTAL	ORDER_DATE
Z22345	342	28-07-2020
Z62998	543	30-07-2020
Z56990	431	28-07-2020
Z56902	6743	29-07-2020
Z99781	443	27-07-2020
Z56112	889	30-07-2020

*The ORDER\_DATE column follows domain integrity as the format of date is fixed.*

# Types of Integrity

- Entity integrity
- It ensures that each row in a table is unique. This integrity is achieved using the primary key.
- For example, the ORDERS table has a primary key as ORDER\_ID, which cannot be duplicated throughout the table.

ORDER_ID	ORDER_TOTAL	ORDER_DATE
Z22345	342	28-07-2020
Z62998	543	30-07-2020
Z56990	431	28-07-2020
Z56902	6743	29-07-2020
Z99781	443	27-07-2020
Z56112	889	30-07-2020

*The ORDER\_ID column follows the entity integrity. The value in ORDER\_ID column should be unique and act as a primary key for the table.*

# Types of Integrity

## Referential integrity

- It ensures that data integrity is maintained between primary and foreign key.
- For example, ORDERS table has a primary key as ORDER\_ID and foreign key as TRANSACTION\_ID which refers to the TRANSACTIONS table.

ORDER_ID	TRANSACTION_ID	ORDER_DATE
Z22345	ITX4489	23-10-2020
Z62998	ITX4311	21-10-2020
Z56902	ITX3120	26-10-2020

*The TRANSACTION\_ID column follows the referential integrity. This column is foreign key in ORDERS table and primary key in TRANSACTIONS table.*

TRANSACTION_ID	TRANS_AMT	TRANSACTION_STATUS
ITX4489	1128	PAID
ITX4311	2318	PAID
ITX3120	88956	UNPAID

# Security Vs Integrity

Security	Integrity
Data security deals with protection of data.	Data integrity deals with the validity of data.
Data security is making sure that only the people who should have access to the data are the only ones who can access the data.	Data integrity is making sure that the data is correct and not corrupt.
Data security avoids from unauthorized access of data.	Data integrity avoids from human errors, when data is entered.
Data security is implemented through user account (passwords).	Data integrity is implemented through constraints such as Primary key, Foreign key, Check constraints etc.

# Authentication and Authorization

- **Authentication**

- User authentication is to make sure that the person accessing the database is who he claims to be.
- Authentication can be done at the operating system level or even the database level itself.
- It is used by both server and client.
- The server uses authentication when someone wants to access the information, and the server needs to know who is accessing the information. The client uses it when he wants to know that it is the same server that it claims to be.
- The authentication by the server is done mostly by using the username and password. Other ways of authentication by the server can also be done using cards, retina scans, voice recognition, and fingerprints.
- Authentication does not ensure what tasks under a process one person can do, what files he can view, read, or update. It mostly identifies who the person or system is actually.
- Many authentication systems such as retina scanners or bio-metrics are used to make sure unauthorized people cannot access the database.

# Authentication and Authorization

- As per the security levels and the type of application, there are different types of Authentication factors:
- **Single-Factor Authentication**
- Single-factor authentication is the simplest way of authentication. It just needs a username and password to allow a user to access a system.
- **Two-factor Authentication**
- As per the name, it is two-level security; hence it needs two-step verification to authenticate a user. It does not require only a username and password but also needs the unique information that only the particular user knows, such as first school name, a favorite destination. Apart from this, it can also verify the user by sending the OTP or a unique link on the user's registered number or email address.
- **Multi-factor Authentication**
- This is the most secure and advanced level of authorization. It requires two or more than two levels of security from different and independent categories. This type of authentication is usually used in financial organizations, banks, and law enforcement agencies. This ensures to eliminate any data exposure from the third party or hackers.

# Authentication and Authorization

- **Example:**
- When you enter your ATM card into the ATM machine, the machine asks you to enter your pin.
- After you enter the pin correctly, the bank then confirms your identity that the card really belongs to you and you're the rightful owner of the card.
- By validating your ATM card pin, the bank actually verifies your identity, which is called authentication.
- It merely identifies who you are, nothing else.

# Authentication and Authorization

- **Authorization:**

- Authorization, on the other hand, occurs after your identity is successfully authenticated by the system, which ultimately gives you full permission to access the resources such as information, files, databases, funds, locations, almost anything.
- In simple terms, authorization determines your ability to access the system and up to what extent.
- Once your identity is verified by the system after successful authentication, you are then authorized to access the resources of the system.
- Authorization is the process of granting someone to do something.
- The authorization usually works with authentication so that the system could know who is accessing the information.
- Authorization is not always necessary to access information available over the internet. Some data available over the internet can be accessed without any authorization, such as you can read about any technology from here.



# Authentication and Authorization

- **The different permissions for authorizations available are:**
- **Primary Permission** - This is granted to users publicly and directly.
- **Secondary Permission** - This is granted to groups and automatically awarded to a user if he is a member of the group.
- **Public Permission** - This is publicly granted to all the users.

# Authentication and Authorization

- **The categories of authorization that can be given to users are:**
- **System Administrator** - This is the highest administrative authorization for a user. Users with this authorization can also execute some database administrator commands such as restore or upgrade a database.
- **System Control** - This is the highest control authorization for a user. This allows maintenance operations on the database but not direct access to data.
- **System Maintenance** - This is the lower level of system control authority. It also allows users to maintain the database but within a database manager instance.
- **System Monitor** - Using this authority, the user can monitor the database

# Authentication and Authorization

- **Example:**
- The process of verifying and confirming employees ID and passwords in an organization is called authentication, but determining which employee has access to which floor is called authorization.
- Let's say you are traveling and you're about to board a flight.
- When you show your ticket and some identification before checking in, you receive a boarding pass which confirms that the airport authority has authenticated your identity. But that's not it. A flight attendant must authorize you to board the flight you're supposed to be flying on, allowing you access to the inside of the plane and its resources.
- Access to a system is protected by both authentication and authorization.
- Any attempt to access the system might be authenticated by entering valid credentials, but it can only be accepted after successful authorization. If the attempt is authenticated but not authorized, the system will deny access to the system.

# Authentication and Authorization

- In the digital world, authentication and authorization accomplish these same goals.
- Authentication is used to verify that users really are who they represent themselves to be.
- Once this has been confirmed, authorization is then used to grant the user permission to access different levels of information and perform specific functions, depending on the rules established for different types of users.

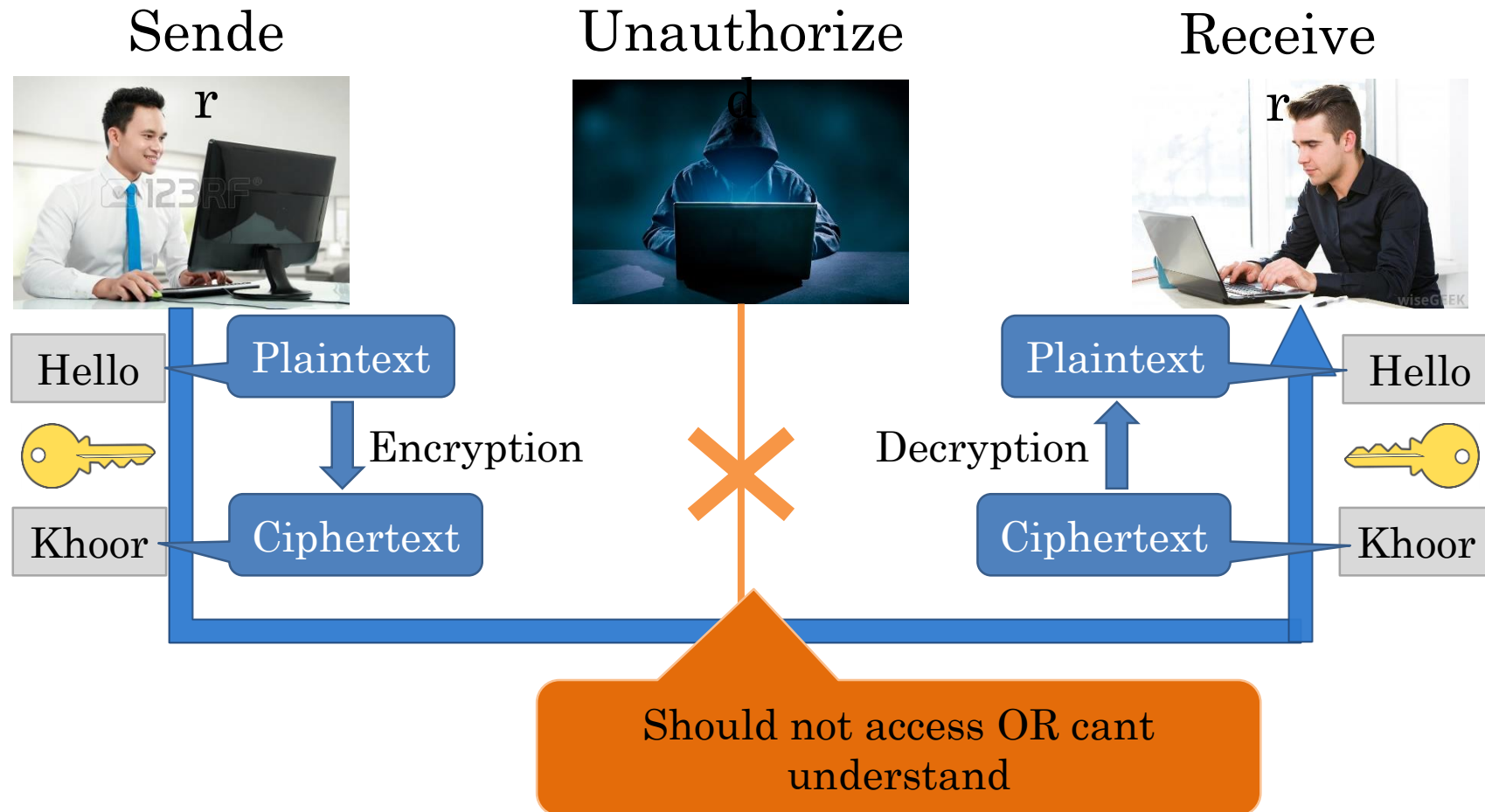
# Authentication Vs Authorization

Authentication	Authorization
Authentication confirms your identity to grant access to the system.	Authorization determines whether you are authorized to access the resources.
It is the process of validating user credentials to gain user access.	It is the process of verifying whether access is allowed or not.
It determines whether user is what he claims to be.	It determines what user can and cannot access.
Authentication usually requires a username and a password.	Authentication factors required for authorization may vary, depending on the security level.
Authentication is the first step of authorization so always comes first.	Authorization is done after successful authentication.
Example: Entering Login details is necessary for the employees to authenticate themselves to access the organizational emails or software.	Example: After employees successfully authenticate themselves, they can access and work on certain functions only as per their roles and profiles.
Authentication credentials can be partially changed by the user as per the requirement.	Authorization permissions cannot be changed by the user. The permissions are given to a user by the owner/manager of the system, and he can only change it.

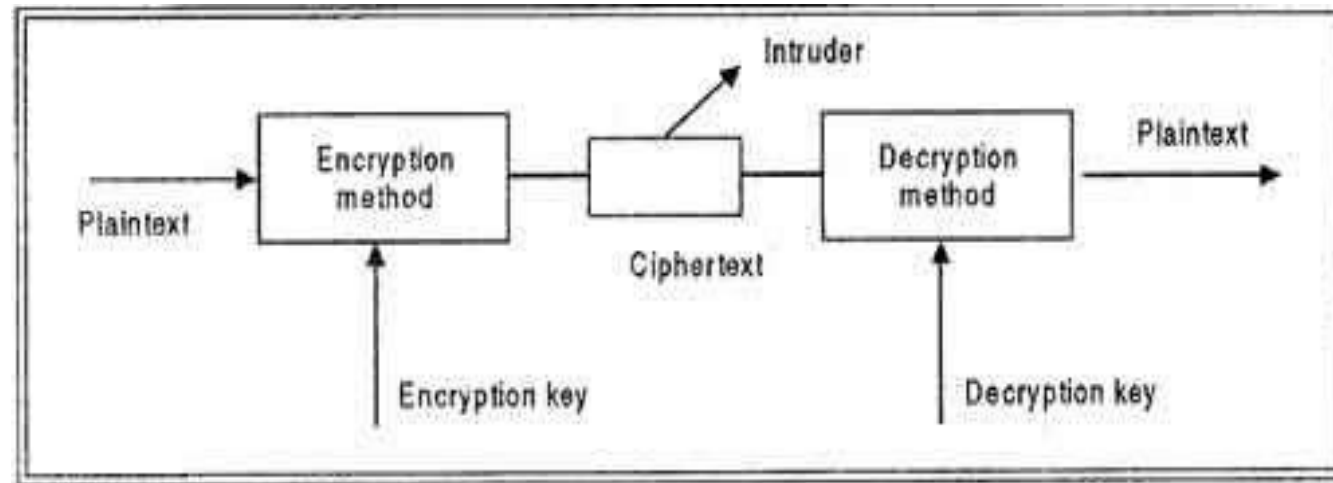
# Data Encryption

- Encryption is a security method in which information is encoded in such a way that only authorized user can read (understand) it.
- It uses encryption algorithm to generate ciphertext that can only be read if decrypted
- A DBMS can use encryption to protect information in certain situations where the normal security mechanisms of the DBMS are not adequate.
- For example, an intruder may steal tapes containing some data or tap a communication line. By storing and transmitting data in an encrypted form, the DBMS ensures that such stolen data is not intelligible to the intruder. Thus, encryption is a technique to provide privacy of data.

# Data Encryption



# Data Encryption





# Data Encryption

- In encryption, the message to be encrypted is known as plaintext.
- The plaintext is transformed by a function that is parameterized by a key.
- The output of the encryption process is known as the cipher text.
- Ciphertext is then transmitted over the network.
- The process of converting the plaintext to ciphertext is called as Encryption and process of converting the ciphertext to plaintext is called as Decryption.
- Encryption is performed at the transmitting end and decryption is performed at the receiving end.
- For encryption process we need the encryption key and for decryption process we need decryption key as shown in figure.
- Without the knowledge of decryption key intruder cannot break the ciphertext to plaintext.
- This process is also called as Cryptography.

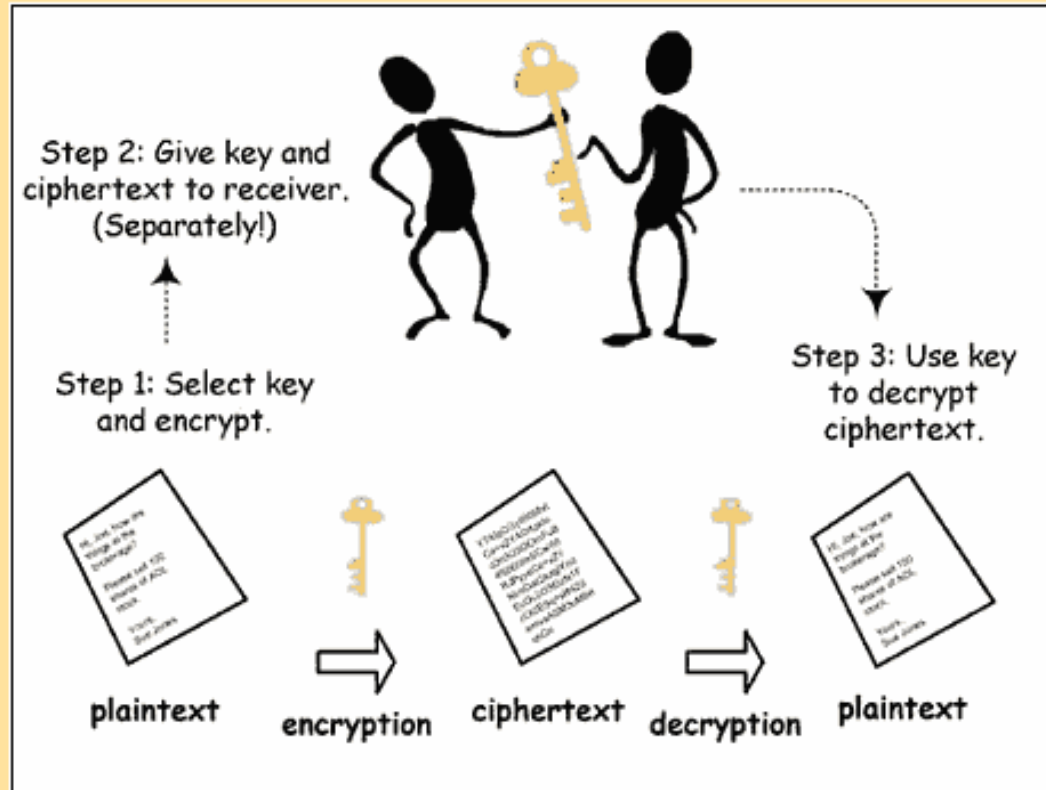
# Data Encryption

- Data encryption is the process of encoding (translating) a message or information in such a way that only authorized persons can access it and those who are not authorized cannot.
- Encryption is the process of translating plaintext data (plaintext) into something that appears to be meaningless (ciphertext).
- Decryption is the process of converting ciphertext back to plaintext.
- Types of Encryption
  - Symmetric key encryption / Private key encryption
  - Asymmetric key encryption / Public key encryption

# Data Encryption

## Symmetric-key Cryptography

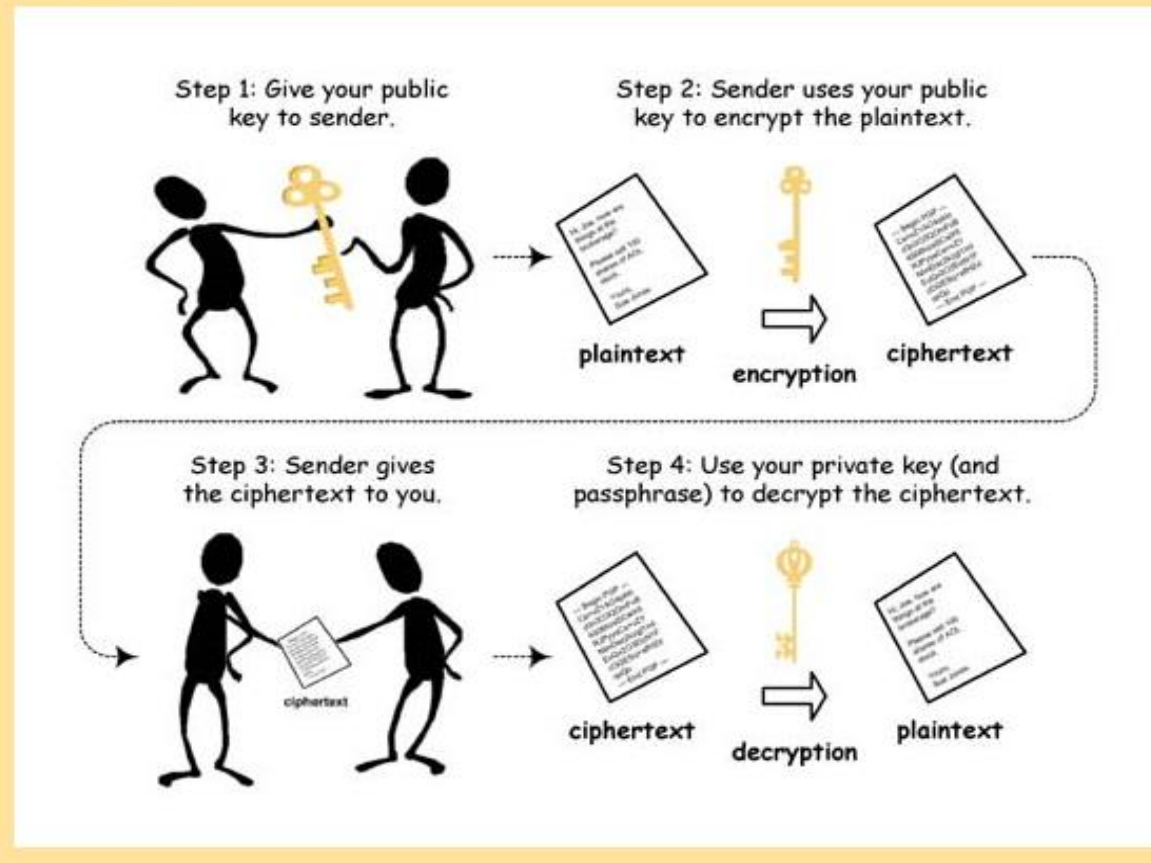
**Symmetric-key cryptography** refers to encryption methods in which both the sender and receiver share the same key.



# Data Encryption

## Asymmetric-key Cryptography

Public-key is a type of asymmetric key cryptography in which two different keys are used. There is a public key and a private key. A public key system is sort of like a mail slot - any one can **encrypt** (put mail in your mail slot), but only the holder of the private key can **decrypt** (the owner of the mail box who has its key).



# Missing Information

- The standard method of representing missing information is to set the “value” to null.
- A null represents missing or unknown information at the column level.
- The column must be defined to be able to accept a null; a column defined as “not null” cannot be set to null.
- A null is not the same as 0 or blank.
- Null means no entry has been made for the column and it implies that the value is either unknown or not applicable.
- A 0 or a blank are actual values stored for the column.

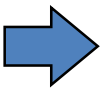
# Continue..

- SQL supports a special value NULL in place of a value in a tuple's component
- Null's can be used for multiple purposes value exists, but we do not know what it is.
- It is an information that does not exist
- **Example:**
- boris registered for pass/fail and thus has no project assigned (nulls used to represent information is inapplicable)
- stefan registered for letter grade but has no current project assignment (nulls used to represent unavailability of information)

Cs184 projects	student	proj_title	date
	boris	null	null
	stefan	null	null

# Continue..

- Using nulls for missing values may result in loss of information

Cs184 projects	student	proj_title	date		Cs184 projects	student	proj_title	date
	boris	oodb	11/16/95			boris	null	null
	stefan	oodb	11/16/95			stefan	null	null
	monica	par dbms	11/21/95			monica	null	null

- Information that boris and stefan are part of the same project team , and that monica is a team by herself is lost by using nulls!

# Continue..

- 98% of employees have a fax number and a query accessing office number and fax number is very common.
- Storing information using a different schema (employee, office num) and (employee, fax number) will cause all such queries to perform a join!
- Instead using nulls is a better idea in this case.

office relation	employee	office num	fax number
	boris	DCL 2111	333-2400
	stefan	DCL 3001	null
	monica	DCL 3444	333-0067



# Continue..

- When a DBMS supports nulls you can distinguish between a deliberate entry of 0 (for numerical columns) or a blank (for character columns) and an unknown or inapplicable entry (NULL for any data type).
- Null indicates that the user did not explicitly make an entry or has explicitly chosen <NULL> for the column in that specific row.
- For example, a null in the Price column of the ITEM table does not mean that the item is being given away for free; instead it means that the price is not known or has not yet been set.
- Whereas a Price set to 0 would seem to indicate that the item is available free of charge.

# Continue..

- A null is not a value it is not greater than, less than, or equal to any other value. And one null does not equal another null.
- **It is invalid to test if a column is = NULL, < NULL, <= NULL, > NULL, or >= NULL.**
- To test for the existence of nulls, use the special predicate IS NULL or IS NOT NULL in the WHERE clause of the SELECT statement.

# Continue...

- Example:
- `SELECT * FROM Visited;`

id	site	dated
619	DR-1	1927-02-08
622	DR-1	1927-02-10
734	DR-3	1930-01-07
735	DR-3	1930-01-12
751	DR-3	1930-02-26
752	DR-3	-null-
837	MSK-4	1932-01-14
844	DR-1	1932-03-22

# Continue...

- `SELECT * FROM Visited WHERE dated < '1930-01-01';`

id	site	dated
619	DR-1	1927-02-08
622	DR-1	1927-02-10

- `SELECT * FROM Visited WHERE dated >= '1930-01-01';`

id	site	dated
734	DR-3	1930-01-07
735	DR-3	1930-01-12
751	DR-3	1930-02-26
837	MSK-4	1932-01-14
844	DR-1	1932-03-22

- Both the results have not included the record with null values.

# Continue...

- The reason is that `null < '1930-01-01'` is neither true nor false.
- Null means, “We don’t know,” and if we don’t know the value on the left side of a comparison, we don’t know whether the comparison is true or false.
- Since databases represent “don’t know” as null, the value of `null < '1930-01-01'` is actually null. `Null >= '1930-01-01'` is also null because we can’t answer to that question either.
- And since the only records kept by a WHERE are those for which the test is true, record #752 isn’t included in either set of results.
- Comparisons aren’t the only operations that behave this way with nulls. `1 + null` is null, `5 * null` is null, `log(null)` is null, and so on.
- In particular, comparing things to null with `=` and `!=` produces null:

# Continue...

- To check whether a value is null or not, we must use a special test IS NULL:
- `SELECT * FROM Visited WHERE dated IS NULL;`

id	site	dated
752	DR-3	-null-

- `SELECT * FROM Visited WHERE dated IS NOT NULL;`

id	site	dated
619	DR-1	1927-02-08
622	DR-1	1927-02-10
734	DR-3	1930-01-07
735	DR-3	1930-01-12
751	DR-3	1930-02-26
837	MSK-4	1932-01-14
844	DR-1	1932-03-22

# Continue...

- For example, suppose we want to find all the salinity measurements that weren't taken by Lake.
- `SELECT * FROM Survey WHERE quant = 'sal' AND person != 'lake';`

taken	person	quant	reading
619	dye	sal	0.13
622	dye	sal	0.09
752	roe	sal	41.6
837	roe	sal	22.5

- but this query filters out the records where we don't know who took the measurement.
- Once again, the reason is that when person is null, the `!=` comparison produces null, so the record isn't kept in our results.

# Continue..

- Any arithmetic operation on Null and any other value results in Null.
- E.g.,  $x + 3 = \text{Null}$ , if  $x$  is Null
- Comparison of Null with any value (including other Null) results in a value UNKNOWN
- E.g.,  $x > 3$  results in UNKNOWN, if  $x$  is Null
- The results of comparison was always TRUE or FALSE.
- Logical operators: AND, OR, NOT combined these truth values in a natural way to return a TRUE or a FALSE.
- However, comparison of Null to a value produces a third truth value -- UNKNOWN



# Continue..

- We can use three-valued logic instead of classical two-valued logic to evaluate conditions where result should be unknown.
- When there are no NULLs around, conditions evaluate to true or false, but if a null is involved, a condition will evaluate to the third value ('undefined', or 'unknown').
- This is the idea behind testing conditions in WHERE clause of SQL SELECT: only tuples where the condition evaluates to true are returned.

# 3VL – THREE VALUED LOGIC

- To understand how AND, OR, and NOT work in 3-valued logic, think of TRUE = 1; FALSE = 0, and UNKNOWN = 1/2.

AND = MIN

OR = MAX

NOT = 1 - X

- Example:

TRUE AND (FALSE OR NOT( UNKNOWN ))

= MIN(1, MAX(0, (1 -  $\frac{1}{2}$  )))

= MIN(1, MAX(0,  $\frac{1}{2}$  ))

= MIN(1,  $\frac{1}{2}$  )

=  $\frac{1}{2}$ .

# 3VL – THREE VALUED LOGIC

- The three-valued logic defines that a boolean expression can return one of these values:
- Truth Table

X	Y	X AND Y	X OR Y	NOT X
TRUE	TRUE	TRUE	TRUE	FALSE
TRUE	UNKNOWN	UNKNOWN	TRUE	FALSE
TRUE	FALSE	FALSE	TRUE	FALSE
UNKNOWN	TRUE	UNKNOWN	TRUE	UNKNOWN
UNKNOWN	UNKNOWN	UNKNOWN	UNKNOWN	UNKNOWN
UNKNOWN	FALSE	FALSE	UNKNOWN	UNKNOWN
FALSE	TRUE	FALSE	TRUE	TRUE
FALSE	UNKNOWN	FALSE	UNKNOWN	TRUE
FALSE	FALSE	FALSE	FALSE	TRUE

Thank You