Module 2

Relational Model and E-R Modeling

Relational Model: Candidate Keys, Primary Keys, Foreign Keys - Integrity Constraints - Handling of Nulls - Entity Relationship Model: Types of Attributes, Relationships, Structural Constraints, Relational model Constraints – Mapping ER model to a relationalschema – Extended ER Model - Generalization – Specialization – Aggregations.

Reference :

- *R. Elmasri & S. B. Navathe, Fundamentals of Database Systems, Addison Wesley, 7th Edition, 2016*
- *A. Silberschatz, H. F. Korth & S. Sudarshan, Database System Concepts, McGraw Hill,7th Edition 2019.*

Relation Schema

- A relation schema ‘R’ denoted by R(A1 , A2 , . . . An) is made up of a relation name R and a list of attributes A1 , A2 , . . . An .
- Each Attribute Ai is the name of role played by some domain D in the relation schema R.
- D is called the domain of Ai and is denoted by dom(Ai).
- A relation schema is used to describe a relation and R is called the name of this relation.
- The degree of a relation is the number of attributes ‘n’ of its relation schema.

Relation Schema -Example

EMPLOYEE(Name: String, EId: INT, HomePhone: INT, Address: String, OfficePhone: String, Age: Real, Salary: INT)
Degree of relation = **7** (i.e., 7 attributes)

Relation (r) of Schema (R):
A **relation r(R)** is a **set of n-tuples**.
Denoted as: **$r = \{t_1, t_2, ..., t_m\}$**

Tuple Structure:
Each tuple **$t = <V_1, V_2, ..., V_n>$**
Each value **V_i** belongs to **dom(A_i)** or is **NULL** (unknown or inapplicable).

Tuple Value Reference:
The value of attribute **A_i** in tuple **t** is denoted **$t[A_i]$**

Relation as a Table:
Rows = Tuples (each representing an EMPLOYEE)
Columns = Attributes (e.g., Name, EId)
Null values indicate missing or non-existent data

Relation Schema -Example

Employee						
Name	Eld	Home Phone	Address	Office Phone	Age	Salary
Mahesh	30-01	870-223366	Warangal	NULL	35	40 k
Ramesh	30-02	040-226633	Hyderabad	NULL	36	40 k
Suresh	30-03	040-663322	Kolkata	040-331123	35	42 k
Dinesh	30-04	040-772299	Bangalore	040-321643	36	40 k

Characteristics of Relations

Relations Are Sets of Tuples:

A relation is a **set**, not a list—**no inherent order** among tuples (rows).

Ordering of Tuples Is Irrelevant:

- The position of a tuple (e.g., first, second) is **not meaningful** in a relation.
- Unlike files, which store records in a specific **physical order**, relational tuples have **no fixed order**.

Logical vs Physical View:

- **Logically**, the relation has no ordering.
- **Physically**, when implemented or displayed (as a table or file), ordering **may exist**—but it doesn't affect the relation's definition.

Multiple Logical Orders Possible:

- Tuples can be logically sorted **by any attribute** (e.g., Name, EId, Age), depending on query needs.

NULL in Tuples

Atomic Values Only:

- Each value in a tuple must be **atomic** (indivisible).
- No **composite** or **multivalued** attributes are allowed —hence called the **flat relational model**.

Handling Multivalued & Composite Attributes:

- **Multivalued attributes** → stored in **separate relations**.
- **Composite attributes** → represented by their **simple components** only.

Use of NULL:

- Represents values that are either **unknown** or **not applicable**.

Examples of NULL Meaning:

- **OfficePhone = NULL** → Not applicable (e.g., student has no office).
- **HomePhone = NULL** → Either the student **has no home phone** or it is **unknown**.

Interpretation Matters:

The **meaning of NULL** depends on context:

"Not applicable" (no value exists)

"Unknown" (value exists but is not known)

Relational Model Constraints

Inherent Model-Based Constraints:

Built into the relational model itself.

Example: **No duplicate tuples allowed** in a relation.

Schema-Based Constraints (Defined using DDL):

These are specified at the schema level.

1. **Domain Constraint** – Values must come from a predefined domain.
2. **Key Constraint** – Attributes defined as keys must be unique.
3. **NOT NULL Constraint** – Attribute must always have a value.
4. **Entity Integrity** – Primary key cannot be NULL.
5. **Referential Integrity** – Foreign key must match a primary key in another table or be NULL.
6. **UNIQUE Constraint** – Attribute values must be unique (but can be NULL).
7. **CHECK Constraint** – Custom condition that attribute values must satisfy.

Domain Constraints - Supported Data Types

Each attribute must have a value that belongs to its defined **domain**:

Numeric Types:

- Short Integer
- Integer
- Long Integer
- Real (Float, Double-Precision Float)

Textual Types:

- Characters
- Fixed-Length Strings
- Variable-Length Strings

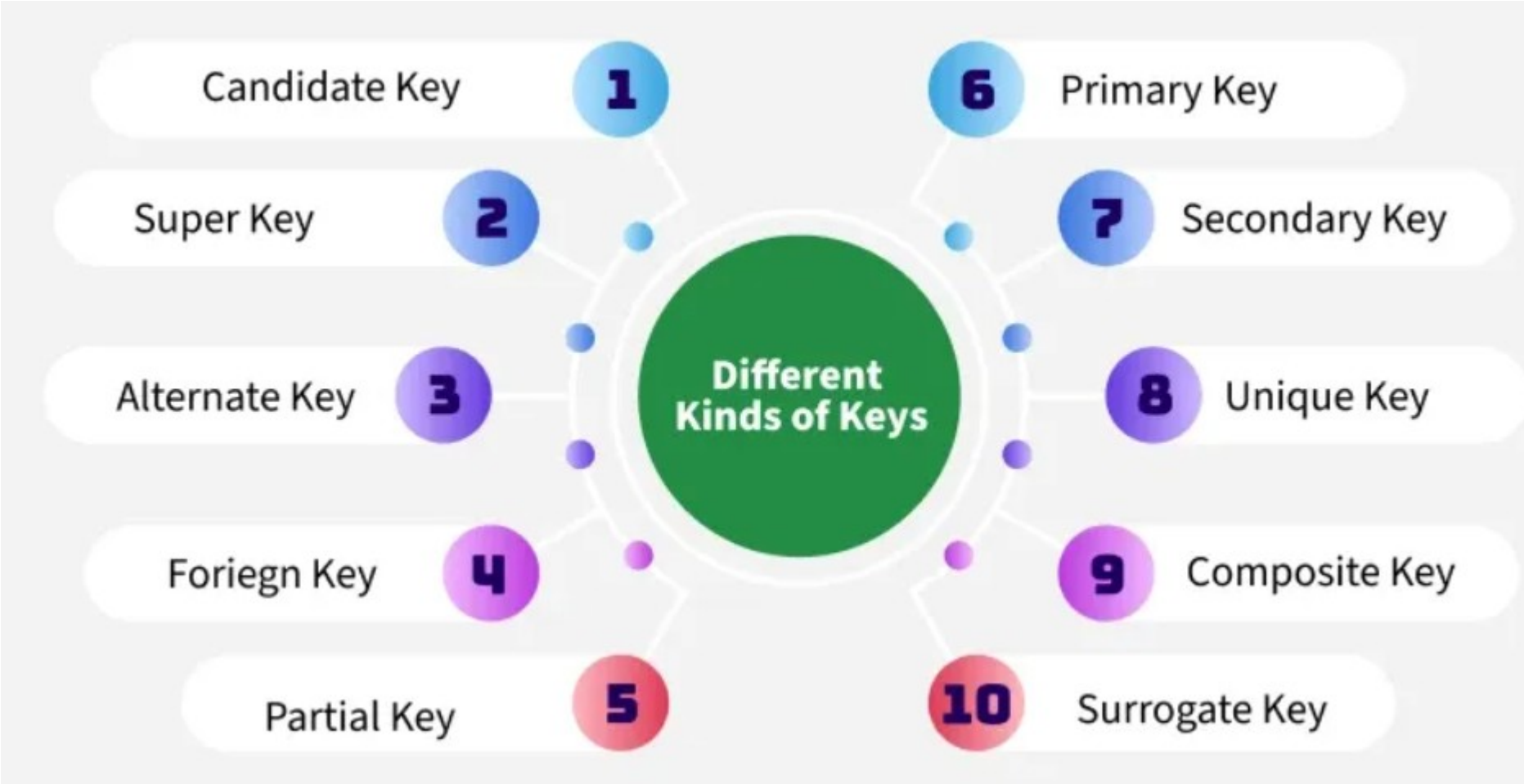
Other Types:

- Boolean
- Date, Time,

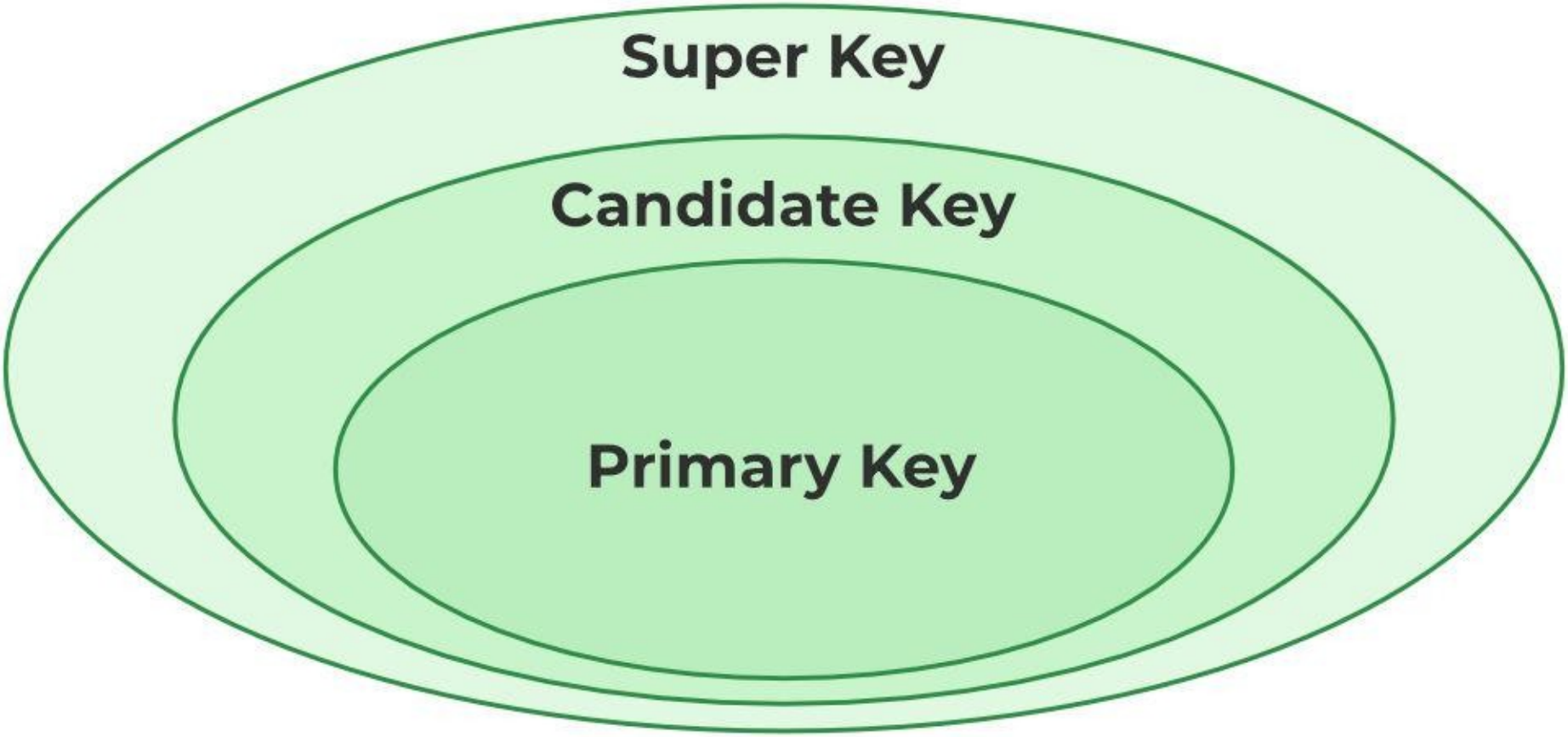
Domain Constraints - Supported Data Types

- **Data Dependencies (Inherent Constraints)**
  - **Functional Dependencies**
  - **Multivalued Dependencies**
- Used in **normalization** to assess and improve schema quality

Key Constraints



Relation between Primary Key, Candidate Key, and Super Key



Key Constraints

1.Uniqueness of Tuples:

1. A **relation is a set** → all **tuples must be unique**.
2. No two tuples can have **identical values for all attributes**.

2. Super Key (SK)

Definition:

A **super key** is any set of attributes that **uniquely identifies** tuples in a relation.

Property:

For any two distinct tuples $\mathbf{t_1}$ and $\mathbf{t_2}$ in relation **r**:

→ $\mathbf{t_1[SK] \neq t_2[SK]}$

Every relation has **at least one super key** — the set of **all its attributes**.

3. Key (Minimal Super Key)

A **key (K)** is a **minimal super key**:

It **uniquely identifies tuples**, like a super key.

But **no proper subset** of its attributes can still be a super key.

Two Conditions for a Key:

Uniqueness: No two tuples can share the same values for all attributes in the key.

Minimality: No attribute in the key can be removed without losing uniqueness.

Key Constraints

- {EId} is a **key**:
  - It uniquely identifies each employee.
  - No two tuples can have the same Eid

Super Keys Examples

All of the following are **super keys** (contain EId, which is already unique):

1. {EId, HomePhone, Name}
2. {EId, Age, Salary}
3. {Name, EId, Address}

These are **not minimal**, so they are not candidate keys.

Non-Key Super Key Example

- {EId, Name, Age} is a **super key**, **but not** a candidate **key**, because:
  - You can remove Name or Age and it still remains a super key (due to EId)

Key Characteristics

- A **key formed from a single attribute** (like EId) is automatically **minimal**, so it **is a key**.
- A **multi-attribute key** must require **all attributes** to ensure

Candidate Keys

- **Candidate Keys**: All **minimal super keys** in a relation schema.

  Example Candidate Keys in EMPLOYEE:
  - {EId}
  - {Name, HomePhone} *(Assuming this combo is unique for all employees)*

  The **actual number** of candidate keys depends on **data constraints**.

Primary Key

- One candidate key is selected as the **primary key**.
  - For EMPLOYEE, typically: **{EId}**

Keys -Example

Number of super keys = 2^{n-k} *where "n" is the number of attributes in the Relation R and "k" is the number of attributes in the Candidate Key*

Example-1: Let a Relation R have attributes {a1,a2,a3} and a1 is the candidate key. Then how many super keys are possible?

Here, any superset of a1 is the super key.
Super keys are = {a1, a1 a2, a1 a3, a1 a2 a3}
Thus we see that 4 Super keys are possible in this case.

In general, if we have 'N' attributes with one candidate key then the number of possible superkeys is $2^{(N - 1)}$.

Keys -Example

Example-2 : Let a Relation R have attributes {a1, a2, a3,...,an}. Find Super key of R.

Maximum Super keys = 2^n - 1.

Proof: There are n attributes in R. So the total number of possible subsets/combination of attributes of R is 2^n

Now to be a Super key, there should be **at least one** attribute present i.e. the NULL set or the set with no attribute can't be a super key.

So, maximum possible number of Super keys of R = 2^n - 1.

Keys -Example

Example-3: Let a Relation R have attributes {a1, a2, a3,..., an} and the candidate key is "a1 a2 a3" then the possible number of super keys?

Following the previous formula, we have 3 attributes instead of one. So, here the number of possible super keys is $2^{(N-3)}$.

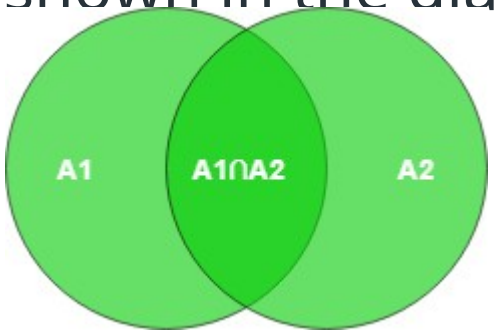
Example-4: Let a Relation R have attributes {a1, a2, a3,..., an} and the candidate keys are "a1", "a2" then the possible number of super keys?

This problem now is slightly different since we now have two different candidate keys instead of only one. Tackling problems like these is shown in the diagram below:

Keys -Example

Example-4: Let a Relation R have attributes {a1, a2, a3,..., an} and the candidate keys are "a1", "a2" then the possible number of super keys?

This problem now is slightly different since we now have two different candidate keys instead of only one. Tackling problems like these is shown in the diagram below:



→ $|A1 \cup A2| = |A1| + |A2| - |A1 \cap A2|$

= (super keys possible with candidate key A1) + (super keys possible with candidate key A2) - (common superkeys from both A1 and A2)

$= 2(n-1) + 2(n-1) - 2(n-2)$

Keys -Example

Example-5: Let a Relation R have attributes {a1, a2, a3,..., an} and the candidate keys are "a1", "a2 a3" then the possible number of super keys?

Super keys of (a1) + Super keys of (a2 a3) - Super keys of (a1 a2 a3)

$\Rightarrow 2(n - 1) + 2(n - 2) - 2(n - 3)$

Example-6: Let a Relation R have attributes {a1, a2, a3,..., an} and the candidate keys are "a1 a2", "a3 a4" then the possible number of super keys?

Super keys of(a1 a2) + Super keys of(a3 a4) - Super keys of(a1 a2 a3 a4)

$\Rightarrow 2(n - 2) + 2(n - 2) - 2(n - 4)$

Keys -Example

Example-5: Let a Relation R have attributes {a1, a2, a3,..., an} and the candidate keys are "a1", "a2 a3" then the possible number of super keys?

Super keys of (a1) + Super keys of (a2 a3) - Super keys of (a1 a2 a3)

$\Rightarrow 2(n - 1) + 2(n - 2) - 2(n - 3)$

Example-6: Let a Relation R have attributes {a1, a2, a3,..., an} and the candidate keys are "a1 a2", "a3 a4" then the possible number of super keys?

Super keys of(a1 a2) + Super keys of(a3 a4) - Super keys of(a1 a2 a3 a4)

$\Rightarrow 2(n - 2) + 2(n - 2) - 2(n - 4)$

Keys -Example

Example-5: Let a Relation R have attributes {a1, a2, a3,..., an} and the candidate keys are "a1", "a2 a3" then the possible number of super keys?

Super keys of (a1) + Super keys of (a2 a3) - Super keys of (a1 a2 a3)

$\Rightarrow 2(n - 1) + 2(n - 2) - 2(n - 3)$

Example-6: Let a Relation R have attributes {a1, a2, a3,..., an} and the candidate keys are "a1 a2", "a3 a4" then the possible number of super keys?

Super keys of(a1 a2) + Super keys of(a3 a4) - Super keys of(a1 a2 a3 a4)

$\Rightarrow 2(n - 2) + 2(n - 2) - 2(n - 4)$

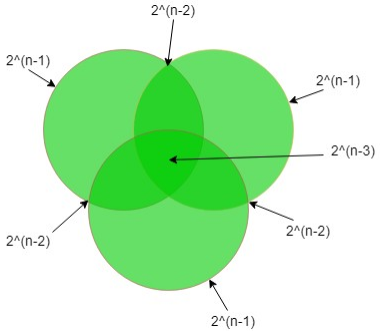
Keys -Example

Example-7: Let a Relation R have attributes {a1, a2, a3,..., an} and the candidate keys are "a1 a2", "a1 a3" then the possible number of super keys?

Super keys of (a1 a2) + Super keys of (a1 a3) - Super keys of(a1 a2 a3)

$\Rightarrow 2(n - 2) + 2(n - 2) - 2(n - 3)$

Keys -Example



Let a Relation R have attributes {a1, a2, a3,...,an} and the candidate keys are "a1", "a2", "a3" then the possible number of super keys?

In this question, we have 3 different candidate keys. Tackling problems like these are shown in the diagram below.

→ $|A1 \cup A2 \cup A3| = |A1| + |A2| + |A3| - |A1 \cap A2| - |A1 \cap A3| - |A2 \cap A3| + |A1 \cap A2 \cap A3|$

= (super keys possible with candidate key A1) + (super keys possible with candidate key A2) + (super keys possible with candidate key A3) - (common super keys from both A1 and A2) - (common super keys from both A1 and A3) - (common super keys from both A2 and A3) + (common super keys from both A1, A2, and A3)

= 2(n-1) + 2(n-1) + 2(n-1) - 2(n-2) - 2(n-2) - 2(n-2) + 2(n-3)

Keys -Example

Example-9: A relation R (A, B, C, D, E, F, G, H)and set of functional dependencies are

CH → G,
A → BC,
B → CFH,
E → A,
F → EG
Then how many possible super keys are present?

Step 1:- First of all, we have to find what the candidate keys are:-
as we can see in the given functional dependency D is missing but in relation, D is given so D must be a prime attribute of the Candidate key.

A+ = E+ = B+ = F+ = all attributes of a relation except D
So, Candidate keys are = AD, BD, ED, FD

Keys –Candidate key

Step 2:-Find super keys due to a single candidate key there is a two possibilities of attribute either we select or not hence there will be 2 chances so,

A_ _D_ _ _ _ = _ B_ D_ _ _ _ = _ _ _ DE _ _ _ = _ _ _ D_F_ _ = 26

Step 3:-Find superkeys due to a combination of two Candidate Keys. So,

n(AD ∩ BD) = n(AD ∩ ED) = n(AD ∩ FD) = n(BD ∩ ED) = n(BD ∩ FD) = n(ED ∩ FD) = 25

Step 4:-Find super keys due to a combination of 3 Candidate Keys

So,

n(AD ∩ BD ∩ ED) = n(AD ∩ ED ∩ FD) = n(ED ∩ BD ∩ FD) = n(BD ∩ FD ∩ AD) = 24

Step 5:-Find super keys due to all. So,

n(AD ∩ BD ∩ ED ∩ FD) = AB_DEF_ _ = 23

So, According to the inclusion-exclusion principle :-

|W ∪ X ∪ Y ∪ Z| = |W| + |X| + |Y| + |Z| - |W ∩ X| - |W ∩ Y| - |W ∩ Z| - |X ∩Y| - |X ∩ Z| - |Y ∩ Z| + |W ∩ X ∩ Y| + |W ∩ X ∩ Z| + |W ∩ Y ? Z| + |X ∩ Y ∩ Z|
- |W ∩ X ∩ Y ∩ Z|

Super keys = 4 * 26 - 6 * 25 + 4 * 24 - 23 = 120

Keys –Candidate key

Example 10 : Let a Relation R have attributes {a1,a2,a3_____ an} and {a1a2a3____ak} as the candidate key where k<=n. Then how many super keys are possible?
The possible number of superkeys is $2^{(n-k)}$.

Example 11: Let a relation R have attributes {a1,a2,a3_____ an} such that any k of the attributes at a time determines all other attributes. Find the value of k such that the number of candidate keys in the relation will be maximum.

Any k attributes at a time constitute one candidate key. These k attributes are randomly chosen from the n attributes. So for some k, the possible no of candidate keys is nCk,i.e, n!/(n-k)!k!. For the number of members to be maximum k must be ⌊n/2⌋ so that nCk is the maximum for that value.

Referential and entity integrity constraint

Entity Integrity Constraint – Key Points

Primary Key Cannot Be NULL:

Every tuple must have a **non-null value** for the **primary key**.

NULL values in a primary key would make a tuple **unidentifiable**, violating entity integrity.

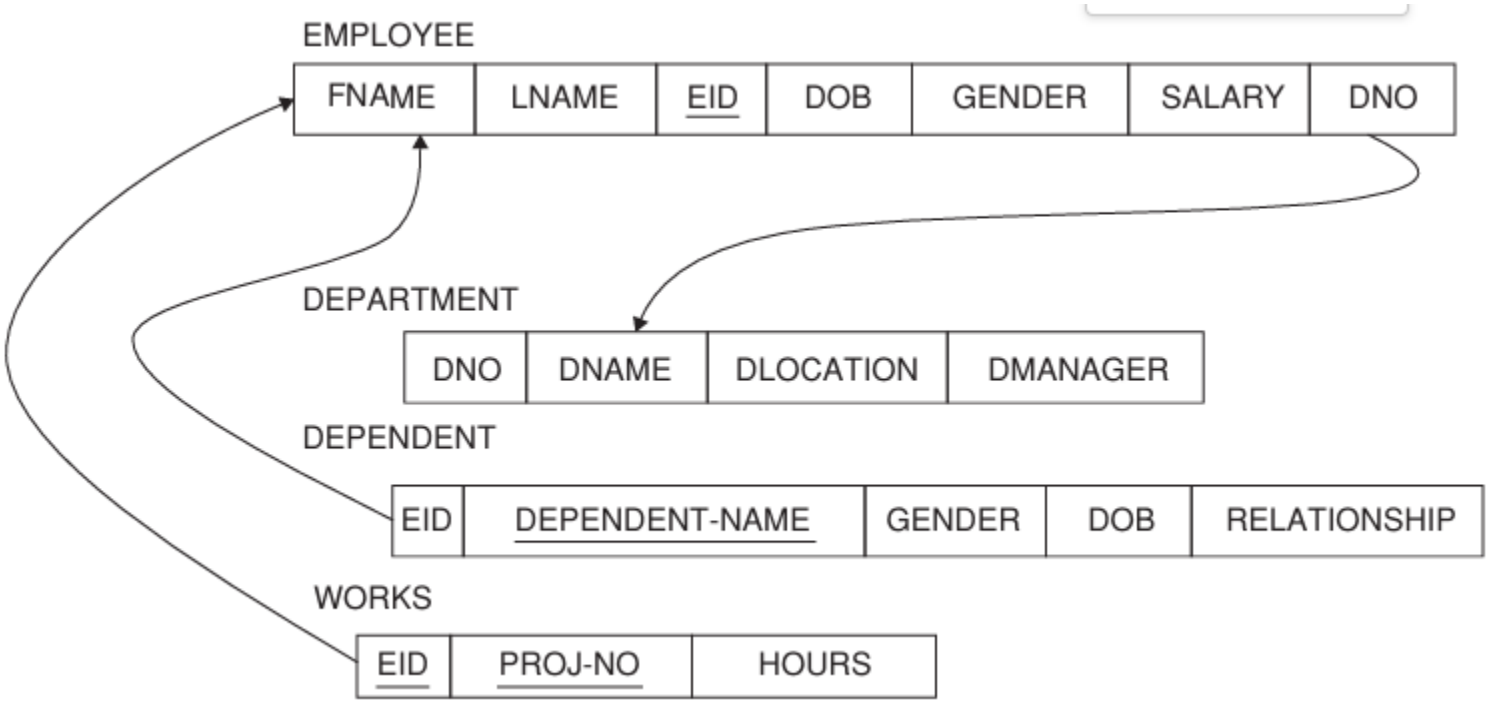
Applies to Individual Relations:

This is a **relation-level** constraint (not between relations).

Referential and entity integrity constraint

1. **Ensures Consistency Between Relations**:
   - A tuple in one relation must **refer to an existing tuple** in another relation.
2. **Requires a Foreign Key (FK)**:
   - A **foreign key** in relation **R1** references the **primary key (PK)** in relation **R2**.
3. **Conditions for Referential Integrity**:
   - Attributes in **FK** must match the **domain** of attributes in **PK** of R2.
   - For any tuple t1 in R1:
     - t1[FK] must either **match** t2[PK] in R2
     - Or be **NULL** (if optional reference is allowed)
4. **Terminology**:
   - **R1**: Referencing relation
   - **R2**: Referenced relation
   - **t1[FK] = t2[PK]** → t1 references t2
5. **Self-referencing Foreign Keys**:
   A **foreign key can refer to the same relation** (e.g., an employee referencing a manager in the same table).
6. **Diagrammatic Representation**:
   - Use a **directed arc (arrow)** from the **foreign key** in R1 → to the **primary key** in R2.
   - Arrow indicates the **referential dependency**.
7. **Referential integrity rule:** The database must not contain any unmatched foreign key values. If 'R' References 'A', then A must exist.

Referential and entity integrity constraint



UNIQUE Constraint

1. Ensures **no duplicate values** in the column.
2. Can be applied on **any column except the primary key**.
3. Can be applied at **column level or table level**.
4. A **primary key** automatically implies uniqueness, so **UNIQUE** is not needed there

```
CREATE TABLE student (
  RNo INT,
  Name VARCHAR(60) UNIQUE,
  Grade CHAR(1)
);
```

If this table already contains:
Name = 'Anu'
Then this query:
sql

```
INSERT INTO student VALUES (14, 'Anu', 'B');
```

Will **violate the UNIQUE constraint** on Name.

Check Constraint

1. Used to **restrict values** within a certain condition/range.
2. Value is **checked before insertion**.
3. Can be applied to **enforce conditions** like positive numbers, limited range, etc.

Example:

```
CREATE TABLE student (
  RNo INT CHECK (RNo > 0),
  Name VARCHAR(60),
  Dept VARCHAR(4)
)
INSERT INTO student VALUES (-4, 'Bhanu', 'CS');
```

Will violate the CHECK constraint because RNo must be > 0.

Creating a View (Virtual Table)

1. A **view** is a **virtual table** created from one or more existing tables.
2. It **does not store data physically**, only the query definition.
3. Useful for **security**, **simplifying queries**, or **showing partial data**.

- SQL

```
CREATE VIEW sales_view AS
SELECT * FROM Sales
WHERE customer = 'Ana';
```

To see contents of the view:

sql

```
SELECT * FROM sales_view;
```

DELETE Statement

- DELETE * FROM TableName — removes **all rows** (not valid syntax in standard SQL; correct is DELETE FROM TableName).
- To delete **specific rows**, use a **WHERE clause**.

Example:

sql

DELETE FROM Sales WHERE PreviousBalance = 3000;

- This removes **only those rows** in Sales where PreviousBalance = 3000.

Output:

Order Id	Order Name	Previous Balance	Customer
22	Order 4	1000	Adam
24	Order 6	2000	John
25	Order 7	2000	Ana
26	Order 8	4000	Ana

Referential actions:

- Referential Integrity can be violated if the value of any foreign key refers to a tuple that does not exist in the referenced relation.
- When certain violations occur, we need to perform some alternate action.
- Those actions are as follows:

1. ON DELETE CASCADE
2. ON UPDATE CASCADE
3. ON DELETE SET NULL
4. ON DELETE SET DEFAULT