DBMS Notes

Database:

- Database can be defined as the structured form of data storage from which data can be retrieved and managed based on requirements.

Database Management System

- DBMS is a software that allows to store, modify and retrieve data from a database.
- It acts as an interface between data and applications.

Relational Database Management System

- It is a type of database that stores and provides access to data points that are related to one another.

Relational Database Basics

- Collection of tables
- **Heading:** table name and column names
- **Body:** rows, occurrences of data

Student

StdNo	StdFirstName	StdLastName	StdCity	StdState	StdZip	StdMajor	StdClass	StdGPA
123-45-6789	HOMER	WELLS	SEATTLE	WA	98121-1111	IS	FR	3.00
124-56-7890	BOB	NORBERT	BOTHELL	WA	98011-2121	FIN	JR	2.70
234-56-7890	CANDY	KENDALL	TACOMA	WA	99042-3321	ACCT	JR	3.50

Sample Table with Matching Values

Student

StdNo	StdFirstName	StdLastName	StdCity	StdState	StdZip	StdMajor	StdClass	StdGPA
123-45-6789	HOMER	WELLS	SEATTLE	WA	98121-1111	IS	FR	3.00
124-56-7890	BOB	NORBERT	BOTHELL	WA	98011-2121	FIN	JR	2.70
234-56-7890	CANDY	KENDALL	TACOMA	WA	99042-3321	ACCT	JR	3.50

Offering

OfferNo	CourseNo	OffTerm	OffYear	OffLocation	OffTime	FacNo	OffDays
1111	IS320	SUMMER	2020	BLM302	10:30 AM		MW
1234	IS320	FALL	2019	BLM302	10:30 AM	098-76-5432	MW
4321	IS320	FALL	2019	BLM214	3:30 PM	098-76-5432	TTH

Enrollment

OfferNo	StdNo	EnrGrade
1234	123-45-6789	3.3
1234	234-56-7890	3.5
4321	123-45-6789	3.5
4321	124-56-7890	3.2

Module-03: Data Integrity

- Data integrity refers to the **accuracy and consistency** of data stored in a database.
- It can be maintained using constraints.
- These constraints define the rules according to which the operations like updation, insertions, etc. must be performed to maintain the data integrity.
- There are mainly four types of Data Integrity:
    - Domain Integrity.
    - Entity Integrity.
    - Referential Integrity.
    - User defined Integrity.

1. **Domain Integrity**
    - Domain refers to the range of acceptable values.
    - It refers to the range of values that we are going to accept and store in a particular column within a database.
    - Data integrity ensures that all data in a field contains valid values.

- Example – Any column contains only integer values so it can't store any other values. (string/ double/...)

2. **Entity Integrity**
    - It says that the value of the primary key should not be NULL. If the value of the primary key is NULL then we can't uniquely identify the rows if all other fields are the same.
3. **Referential Integrity**
    - It is used to maintain the data consistency between two tables.
    - A referential integrity constraint is also known as foreign key constraint.
    - The table containing the foreign key is called the child table, and the table containing the Primary/Candidate key is called the Parent table.
4. **User Defined Integrity**
    - Sometimes these three integrity i.e domain, referential and entity integrity are not enough to maintain the data integrity. Such integrity is typically implemented through triggers and stored procedures.
    - It involves the rules and constraints created by the user to fit their needs.

Module-04: SQL (Structured Query Language)

It is a programming language for storing, manipulating and retrieving data in a relational database.

SQL SELECT Statement

- The SELECT statement is used to select data from a database.
- The data returned is stored in a result table, called the result-set.

SELECT Syntax:

SELECT col1, col2, col3, … FROM table_name;

```
59 •  select * from Student;
60
61
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

StdNo	StdFirstName	StdLastName	StdCity	StdState	StdZip	StdMajor	StdClass	StdGPA
123-45-6789	HOMER	WELLS	SEATTLE	WA	98121-1111	IS	FR	3.00
124-56-7890	BOB	NORBERT	BOTHELL	WA	98011-2121	FIN	JR	2.70
234-56-7890	CANDY	KENDALL	TACOMA	WA	99042-3321	ACCT	JR	3.50
345-67-8901	WALLY	KENDALL	SEATTLE	WA	98123-1141	IS	SR	2.80
456-78-9012	JOE	ESTRADA	SEATTLE	WA	98121-2333	FIN	SR	3.20
567-89-0123	MARIAH	DODGE	SEATTLE	WA	98114-0021	IS	JR	3.60
678-90-1234	TESS	DODGE	REDMOND	WA	98116-2344	ACCT	SO	3.30
789-01-2345	ROBERTO	MORALES	SEATTLE	WA	98121-2212	FIN	JR	2.50
876-54-3210	CRISTOPHER	COLAN	SEATTLE	WA	98114-1332	IS	SR	4.00
890-12-3456	LUKE	BRAZZI	SEATTLE	WA	98116-0021	IS	SR	2.20
901-23-4567	WILLIAM	PILGRIM	BOTHELL	WA	98113-1885	IS	SO	3.80
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Using WHERE Clause

- The WHERE clause is used to filter records.
- It is used to extract only those records that fulfill a specified condition.

```
9
0 •  select * from Student where StdClass = 'JR';
1
2
```

sult Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

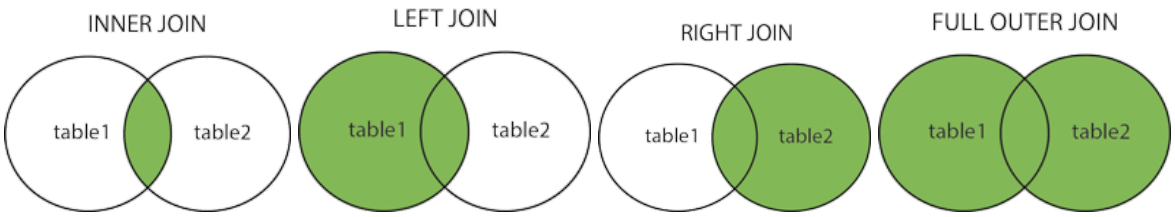
StdNo	StdFirstName	StdLastName	StdCity	StdState	StdZip	StdMajor	StdClass	StdGPA
124-56-7890	BOB	NORBERT	BOTHELL	WA	98011-2121	FIN	JR	2.70
234-56-7890	CANDY	KENDALL	TACOMA	WA	99042-3321	ACCT	JR	3.50
567-89-0123	MARIAH	DODGE	SEATTLE	WA	98114-0021	IS	JR	3.60
789-01-2345	ROBERTO	MORALES	SEATTLE	WA	98121-2212	FIN	JR	2.50
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

SQL Joins Operator

- Joins Operator is used to retrieve data from two or more tables.
- It can be applied only when there is a logical relationship between those tables.

Different Types of SQL JOINs

- **(INNER) JOIN:** Returns records that have matching values in both values.
- **LEFT (OUTER) JOIN**: Returns all records from the left table, and the matched records from the right table.
- **RIGHT (OUTER) JOIN:** Returns all records from the right table and the matched records from the left table.
- **FULL (OUTER) JOIN:** Returns all records when there is a match in either the left or right table.




```
54 • select offering.OffTerm, course.CrsUnits
55   from offering INNER JOIN course
56   ON offering.CourseNo = course.CourseNo;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

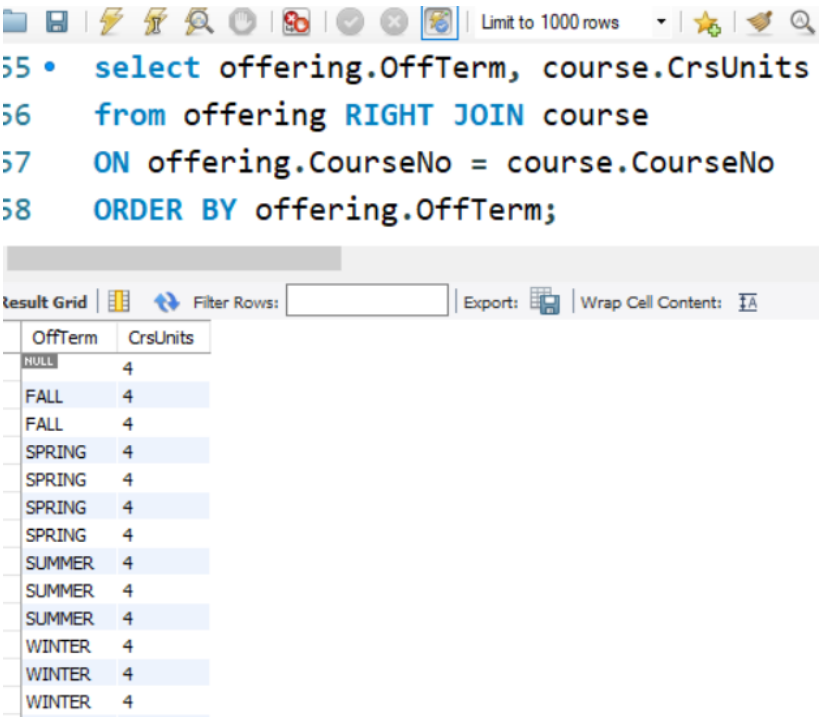
OffTerm	CrsUnits
WINTER	4
WINTER	4
SPRING	4
SUMMER	4
FALL	4
SPRING	4
FALL	4
WINTER	4
SUMMER	4
SUMMER	4
SPRING	4
WINTER	4
SPRING	4

Limit to 1000 rows

```
54 • select offering.OffTerm, course.CrsUnits
55   from offering LEFT JOIN course
56   ON offering.CourseNo = course.CourseNo
```

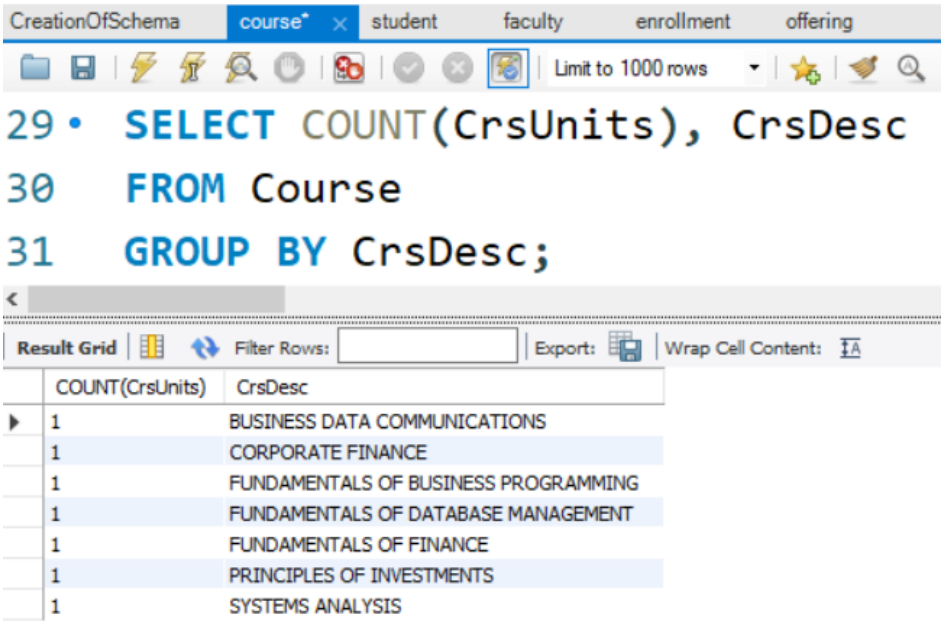
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

OffTerm	CrsUnits
FALL	4
FALL	4
SPRING	4
SPRING	4
SPRING	4
SPRING	4
SUMMER	4
SUMMER	4
SUMMER	4
WINTER	4



SQL GROUP BY Statement

- The GROUP BY statement groups rows that have the same values into summary rows, like “find the number of customers in each country”.
- It is often used in aggregate functions (COUNT(), MAX(), MIN(), SUM(), AVG()) to group the result-set by one or more columns.



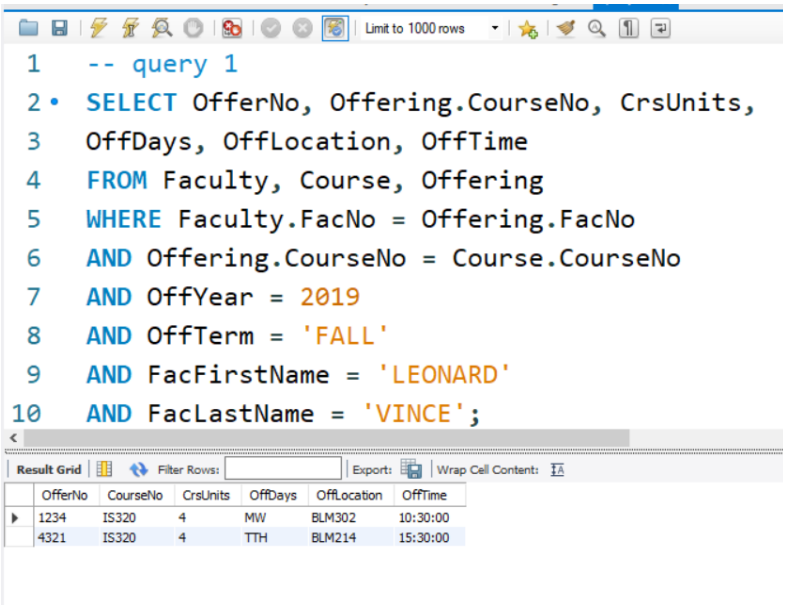
SQL HAVING Clause

- The HAVING clause was added to SQL because the WHERE keyword cannot be used with aggregate functions. **Syntax:**

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
HAVING condition
ORDER BY column_name(s);
```

Module-05: Query Formulation Process

Example 1: List Leonard Vince's teaching schedule in fall 2019. For each course, list the offering number, course number, number of units, days, location, and time.



```
1   -- query 1
2•  SELECT OfferNo, Offering.CourseNo, CrsUnits,
3   OffDays, OffLocation, OffTime
4   FROM Faculty, Course, Offering
5   WHERE Faculty.FacNo = Offering.FacNo
6   AND Offering.CourseNo = Course.CourseNo
7   AND OffYear = 2019
8   AND OffTerm = 'FALL'
9   AND FacFirstName = 'LEONARD'
10  AND FacLastName = 'VINCE';
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

OfferNo	CourseNo	CrsUnits	OffDays	OffLocation	OffTime
1234	IS320	4	MW	BLM302	10:30:00
4321	IS320	4	TTH	BLM214	15:30:00

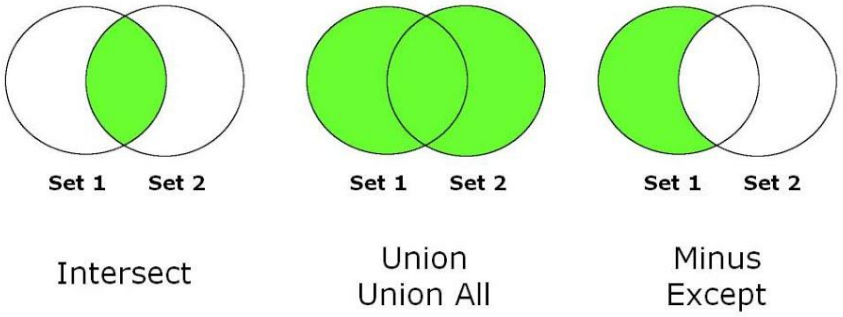
Example 2: List Bob Norbert's course schedule in spring 2020. For each course, list the offering number, course number, days, location, time, course units, and faculty name.

```
11
12 •  SELECT Offering.OfferNo, Offering.CourseNo, OffDays,
13     OffLocation, OffTime, CrsUnits, FacFirstName, FacLastName
14     FROM Faculty, Offering, Enrollment, Student, Course
15     WHERE Faculty.FacNo = Offering.FacNo
16     AND Offering.OfferNo = Enrollment.OfferNo
17     AND Student.StdNo = Enrollment.StdNo
18     AND Offering.CourseNo = Course.CourseNo
19     AND OffYear = 2020
20     AND OffTerm = 'SPRING'
21     AND StdFirstName = 'BOB'
22     AND StdLastName = 'NORBERT';
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

OfferNo	CourseNo	OffDays	OffLocation	OffTime	CrsUnits	FacFirstName	FacLastName
5679	IS480	TTH	BLM412	15:30:00	4	CRISTOPHER	COLAN
9876	IS460	TTH	BLM307	13:30:00	4	LEONARD	FIBON

Traditional SET Operators



Example for UNION: Retrieve basic data about all university people.

```
25    -- query 3
26    SELECT FacNo AS PerNo, FacFirstName AS FirstName,
27    FacLastName AS LastName, FacCity AS City,
28    FacState AS State
29    FROM Faculty
30    UNION
31    SELECT StdNo AS PerNo, StdFirstName AS FirstName,
32    StdLastName AS LastName, StdCity AS City,
33    StdState AS State
34    FROM Student;
```

Result Grid | Filter Rows: | Export: | Wrap Cell C

PerNo	FirstName	LastName	City	State
098-76-5432	LEONARD	VINCE	SEATTLE	WA
543-21-0987	VICTORIA	EMMANUEL	BOTHELL	WA
654-32-1098	LEONARD	FIBON	SEATTLE	WA
765-43-2109	NICKI	MACON	BELLEVUE	WA
876-54-3210	CRISTOPHER	COLAN	SEATTLE	WA
987-65-4321	JULIA	MILLS	SEATTLE	WA
123-45-6789	HOMER	WELLS	SEATTLE	WA
124-56-7890	BOB	NORBERT	BOTHELL	WA
234-56-7890	CANDY	KENDALL	TACOMA	WA
345-67-8901	WALLY	KENDALL	SEATTLE	WA
456-78-9012	JOE	ESTRADA	SEATTLE	WA
567-89-0123	MARIAH	DODGE	SEATTLE	WA

Result 4

Example for INTERSECT: Show teaching assistants, faculty who are students. Only show the common columns in the result.

```
37    SELECT FacNo AS PerNo, FacFirstName AS FirstName,
38    FacLastName AS LastName, FacCity AS City,
39    FacState AS State
40    FROM Faculty
41    INTERSECT
42    SELECT StdNo AS PerNo, StdFirstName AS FirstName,
43    StdLastName AS LastName, StdCity AS City,
44    StdState AS State
45    FROM Student;
46
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

PerNo	FirstName	LastName	City	State
876-54-3210	CRISTOPHER	COLAN	SEATTLE	WA

Example for Except/Minus: Show faculty who are not students (only faculty). Only show the common columns in the result.

```
48 •  SELECT FacNo AS PerNo, FacFirstName AS FirstName,
49    FacLastName AS LastName, FacCity AS City,
50    FacState AS State
51    FROM Faculty
52    EXCEPT
53    SELECT StdNo AS PerNo, StdFirstName AS FirstName,
54    StdLastName AS LastName, StdCity AS City,
55    StdState AS State
56    FROM Student;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

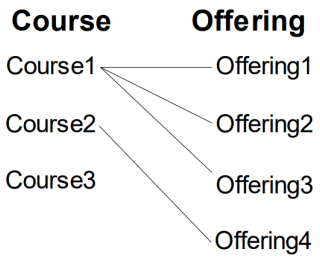
PerNo	FirstName	LastName	City	State
098-76-5432	LEONARD	VINCE	SEATTLE	WA
543-21-0987	VICTORIA	EMMANUEL	BOTHELL	WA
654-32-1098	LEONARD	FIBON	SEATTLE	WA
765-43-2109	NICKI	MACON	BELLEVUE	WA
987-65-4321	JULIA	MILLS	SEATTLE	WA

Module-06: Entity Relationship Diagram

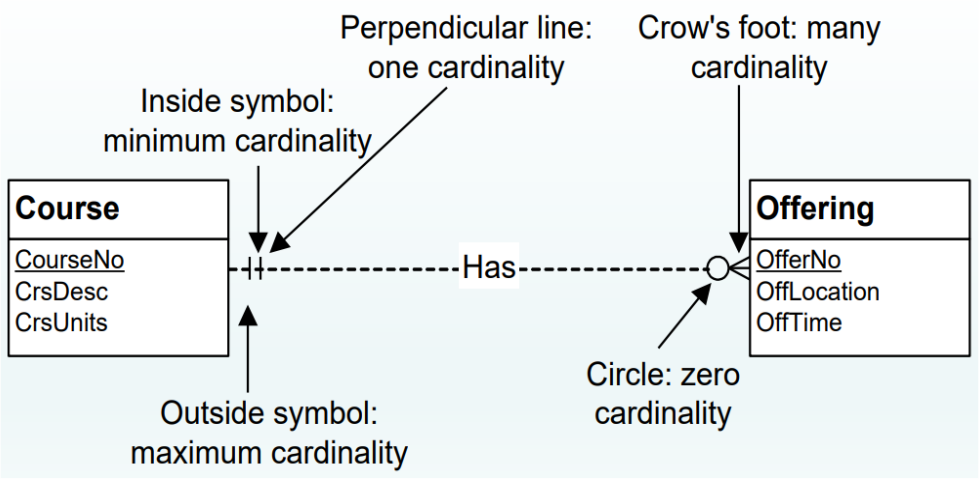
- ERD stands for Entity Relationship Diagram.
- It is a visual representation that illustrates the relationship between different entities in a database.

Cardinality in SQL

- It is a uniqueness of data values contained in a particular column (attribute) of a database table.

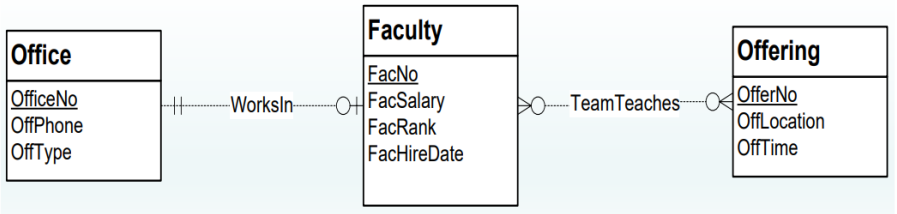


Cardinality Notation

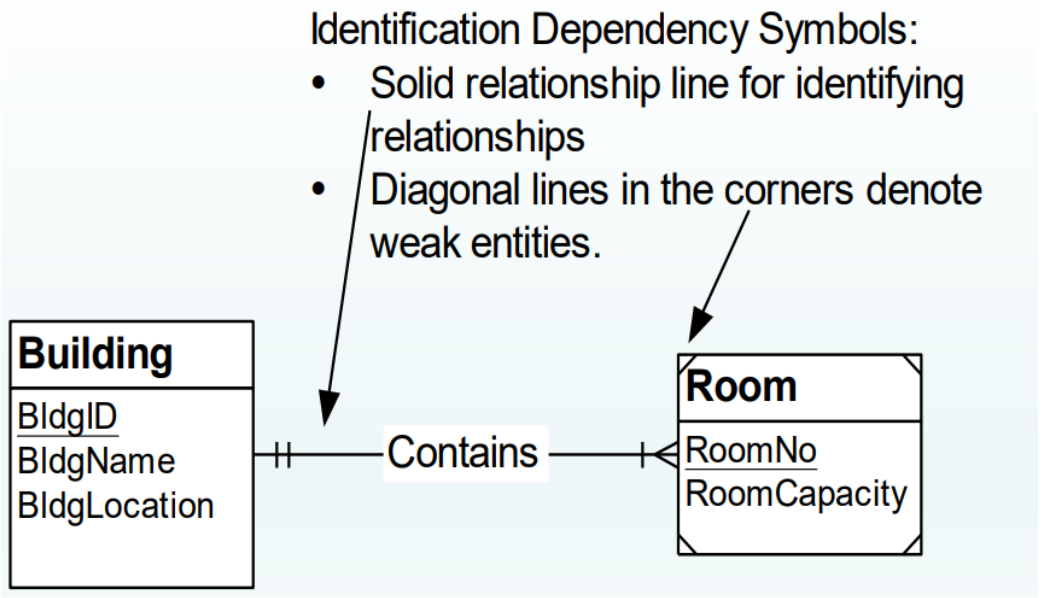


Classification	Cardinality Restrictions
Mandatory	Minimum cardinality ≥ 1
Optional	Minimum cardinality = 0
Functional or single-valued	Minimum cardinality = 1
1-M	Maximum cardinality = 1 in one direction; maximum cardinality > 1 in the other direction
M-N	Maximum cardinality > 1 in both directions
1-1	Maximum cardinality = 1 in both directions

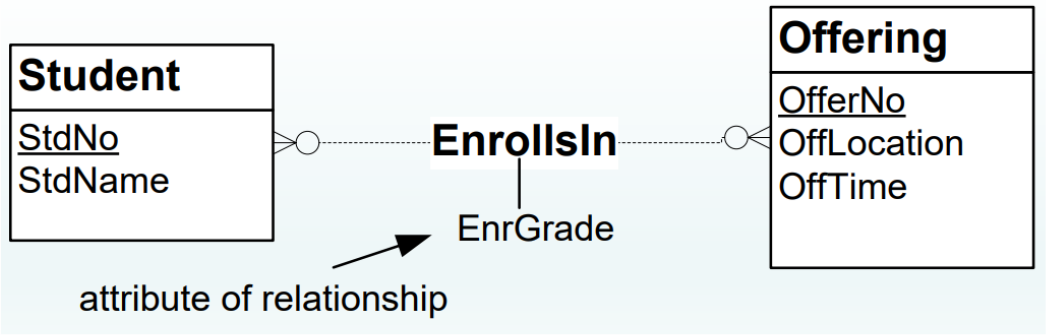
More Relationship Examples



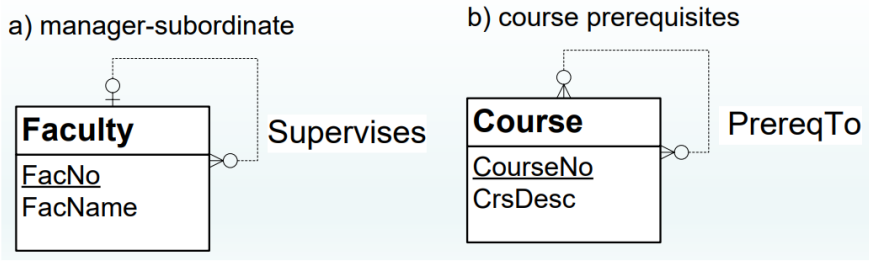
Identification Dependency



M - N Relationships with Attributes



ERD Notation for Self - Referencing Relationships



Module-07: ERD Rules and Problem Solving

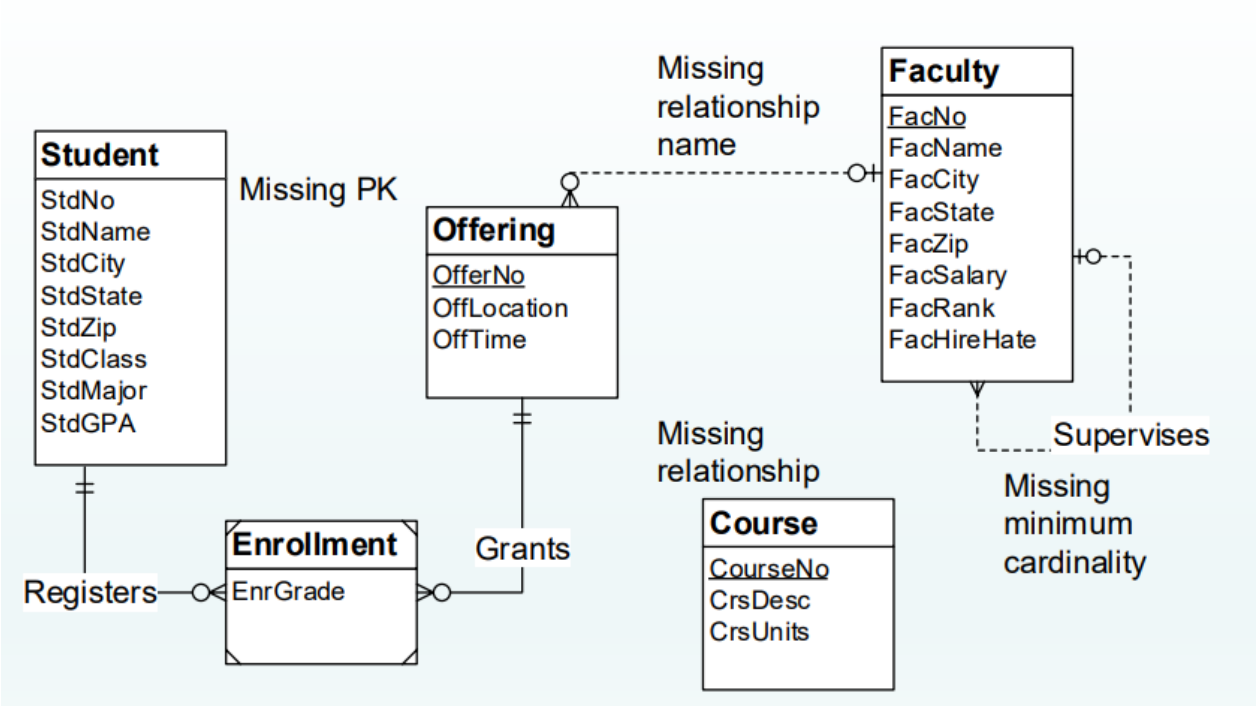
Diagram Rules

- Ensure that ERD notation is correctly used.
- Similar to syntax rules for a computer language.
- **Completeness rules:** no missing specifications.
- Supported by the ER Assistant.

Completeness Rules

- **Primary Key Rule:** All entity types have a PK (direct or indirect).
- **Naming Rule:** All entity types, relationships, and attributes have a name.
- **Cardinality Rule:** Cardinality is specified in both directions for each relationship.
- **Entity Participation Rule:** All entity types participate in at least one relationship.

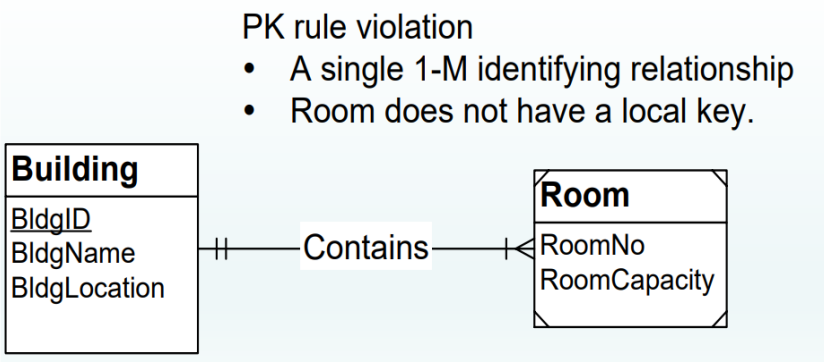
Completeness Rule Violations



Primary Key Rule Issue

- The PK Rule is simple in most cases.
- For some weak entity types, the PK rule is subtle
    - Weak entity type with only one 1-M identifying relationship.
    - Weak entity types must have a local key to argument the borrowed PK from the parent entity type.
    - Violation of PK Rule if the local key is missing.

PK Rule Violation Example



Naming Consistency Rules

- **Entity Name Rule:** Entity type names must be unique.
- **Attribute Name Rule:** Attribute names must be unique within each type and relationship.

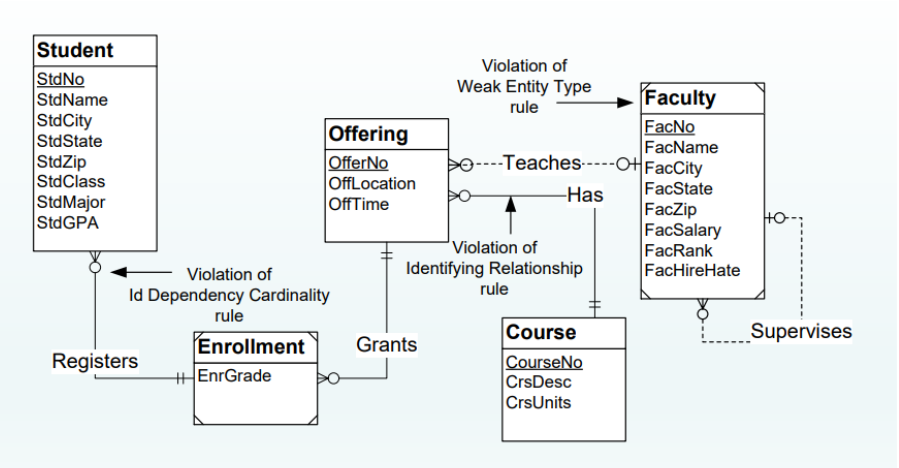
Connection Consistency Rules

- **Relationship/Entity Connection Rule:** Relationships connect two entity types (not necessarily distinct).
- **Relationship/Relationship Connection Rule:** Relationships are not connected to other relationships.
- **Redundant Foreign Key Rule:** Foreign keys are not used.

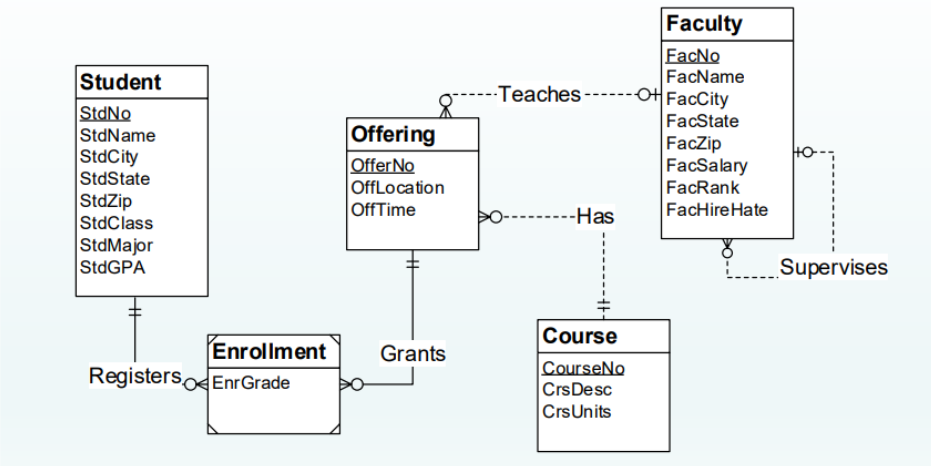
Identification Dependency Rules

- **Weak Entity Type Rule:** Weak entity types have at least one identifying relationship.
- **Identifying Relationship Rule:** At least one participating entity type must be weak for each identifying relationship.
- **Identification/Dependency Cardinality Rule:** The minimum and maximum cardinality must equal 1 for a weak entity type in all identifying relationships.

Identification Dependency Violations

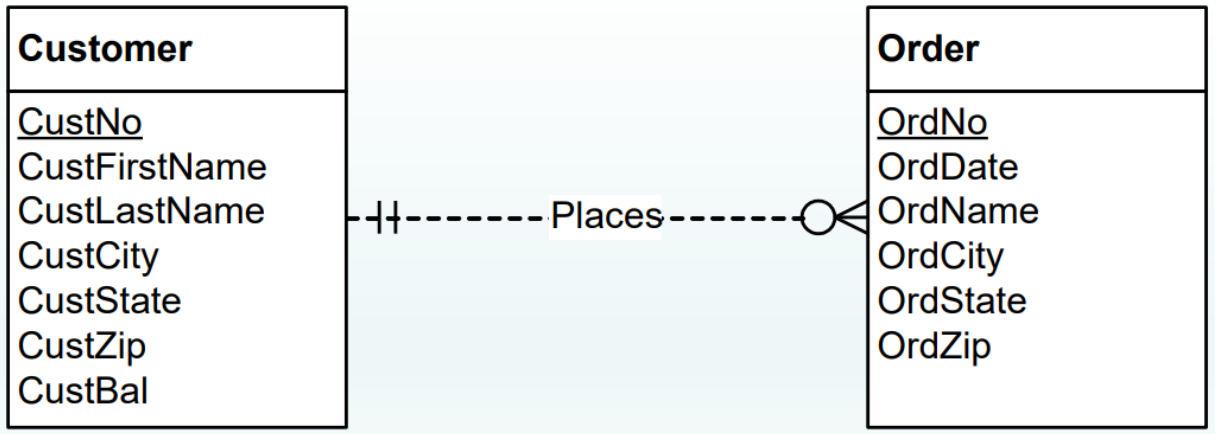


Corrected ERD



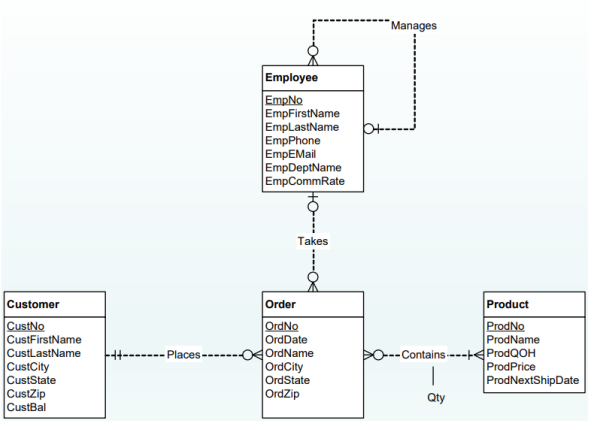
ERD Notation Problem 1

- Draw an ERD containing Order and Customer entity types
    - CustNo (PK), CustFirstName, CustLastName, address, attributes, CustBal
    - OrdNo (PK), OrdDate, OrdName, address, attributes
- Connect with a 1-M relationship
- Order optional for a customer
- Customer mandatory for an order



ERD Notation Problem 3

- Product entity type
    - ProdNo (PK), ProdName, ProdQOH, ProdPrice, ProdNextShipDate
- M-N relationship between product and order with order quantity attribute
- Order optional for product.
- Product mandatory for order



Module-08: Developing Business Data Models

Data Modeling

- Data Modeling in software engineering is the process of simplifying the diagram or data model of a software system by applying certain formal techniques.
- It involves expressing data and information through text and symbols. The data model provides the blueprint for building a new database or reengineering legacy applications.

Broad Goals of Database Development

- Develop a common vocabulary.
- Define business rules.
- Ensure data quality.
- Provide efficient implementation.

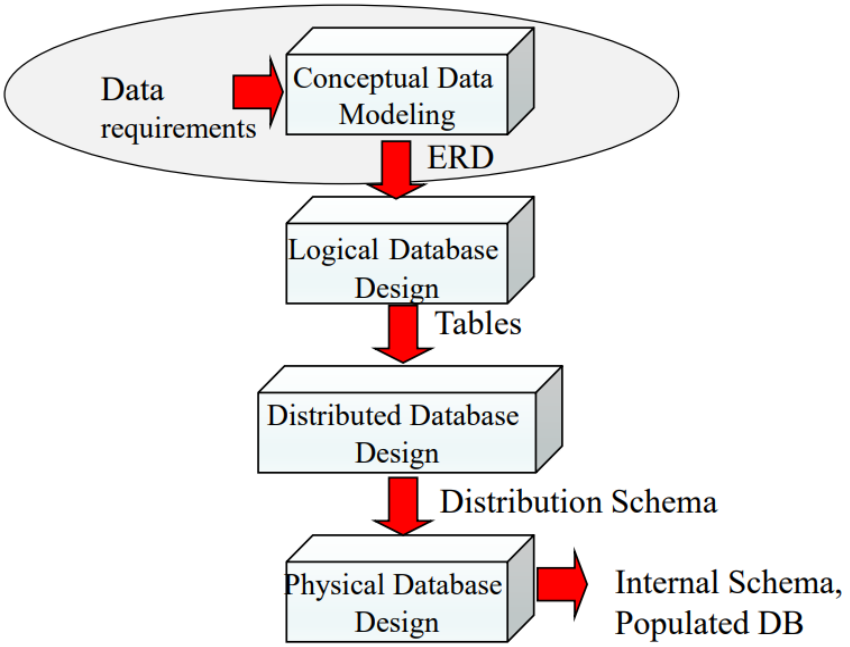
Develop a Common Vocabulary

- Diverse groups of users.
- Difficult to obtain acceptance of a common vocabulary.
- Compromise to find the least objectionable solution.
- Unify organization by establishing a common vocabulary.

Define Business Rules

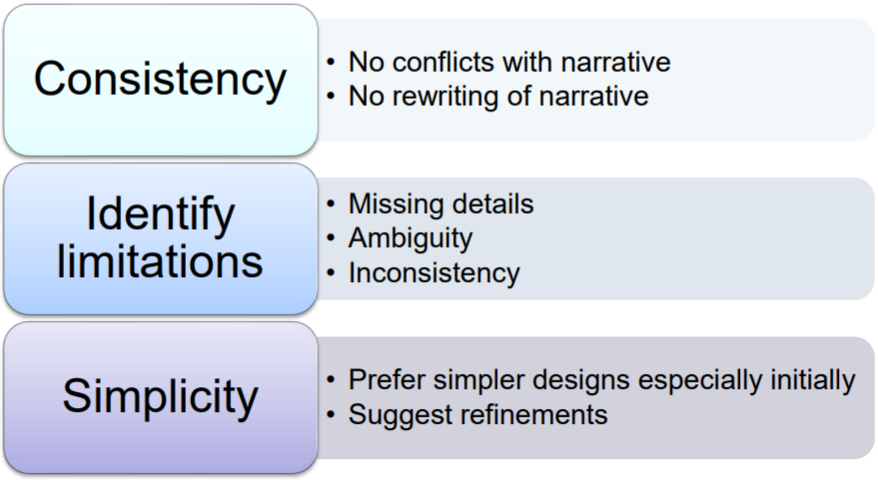
- Support organizational policies
- Determine restrictiveness
    - **Too restrictive**: reject valid business interactions.
    - **Too loose:** allow erroneous business interactions.
- Provide exceptions for flexibility.

Database Development Phases

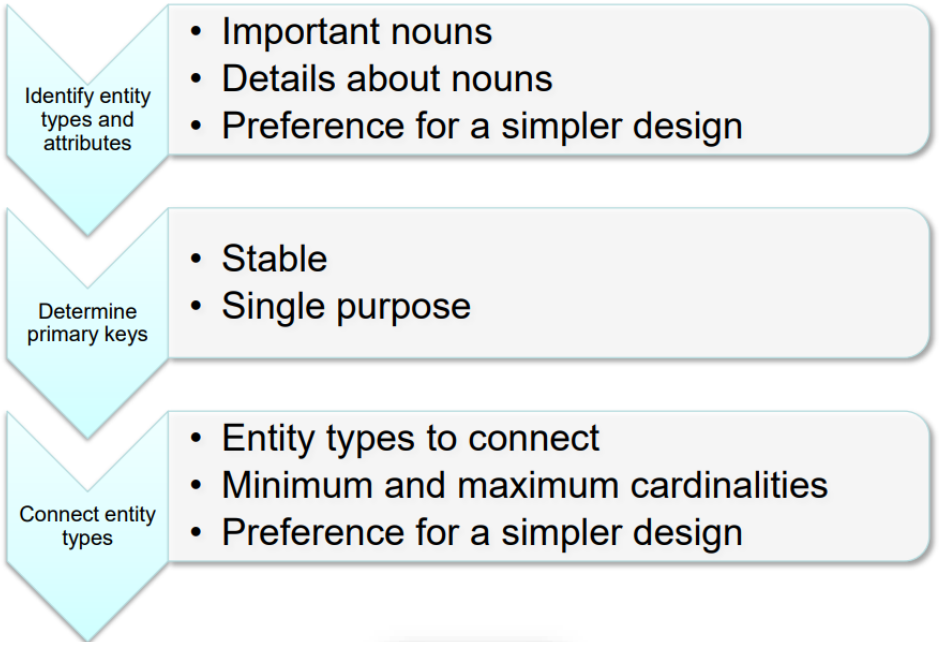


Goals of Narrative Problem Analysis

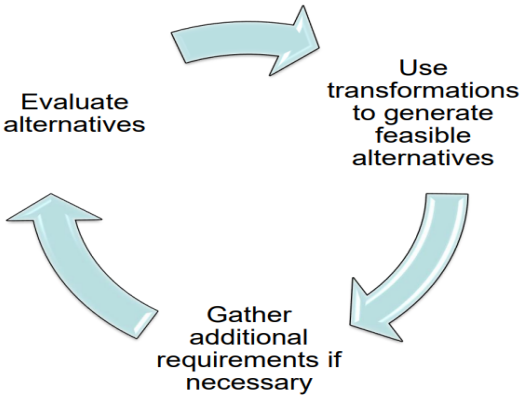
- Narrative analysis is a form of qualitative research in which the researcher focuses on a topic and analyzes the data collected from case studies, surveys, observations or other similar methods. The researchers write their findings, then review and analyze them.



Steps of Narrative Problem Analysis



Data Refinements



Module11: Normalization Concepts and Practice

Functional dependency

- Functional dependency in DBMS refers to a relationship that is present between attributes of any table that are dependent on each other.

FD Definition

- X (Functionally) determines Y or X -> Y
- For each X value, there is at most one Y value
- StdNo -> StdCity if each StdNo value has at most one StdCity value
- X: left-hand side (LHS) or determinant
- Y: right-hand side (RHS)

Normalization

- It is the process of organizing and structuring a database design to minimize redundancy and dependency issues.
- It involves breaking down a database schema into smaller, more manageable and well-structured tables to ensure data integrity and optimize query performance.
- The process of normalization helps in
    - Reducing data redundancy
    - Improving data integrity
    - Facilitating efficient data management
    - Enabling easier maintenance and modification of the database structure.
- It **aims** to achieve a well-structured and optimized database schema that accurately represents the real-world entities and their relationships while avoiding data anomalies and inconsistencies.
- The normalization process typically involves applying a set of rules, known as normal forms, to eliminate data redundancy and anomalies.

Redundancy

Row level duplicity: To reduce row level duplicity use the concept of primary key (Unique + Not Null). Example -

Emp_id	**Emp_name**	**Emp_dept**
32	Aryan	HR
47	Devyansh	Software Developer
32	Aryan	HR

Solution: By applying Primary key constraint on Employee_id it will not allow duplicate entry.

Column Level Duplicity: It refers to the presence of duplicate values within a specific column of a database table. It means that multiple rows share the same value in a particular column. Three types of problem we are facing in this duplicity:

- **Insertion anomaly:** It occurs when certain attributes can't be inserted into a table without the presence of other attributes.
- **Deletion anomaly:** It exists when certain attributes are lost because of the deletion of the other attributes.
- **Updating anomaly**: It exists when one or more instances of duplicated data is updated but not all.

Example:

Emp_id	Emp_name	DeptId	DeptName	Dept_location
1	Aryan	101	HR	Delhi
2	Rahul	102	Developer	Hyderabad
3	Nithin	101	HR	Delhi
4	Priyanshu	101	HR	Delhi

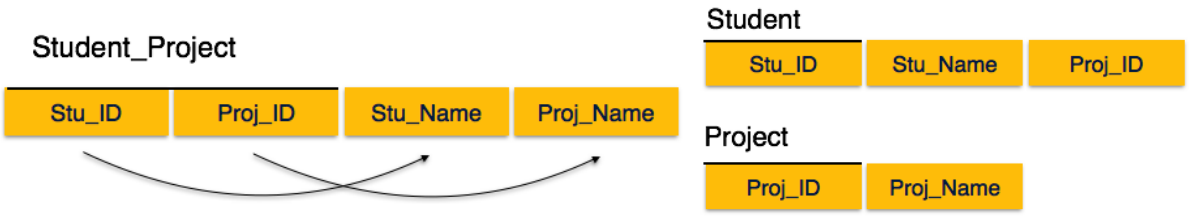
Types of Normal Forms

- **First Normal Form (1NF):** Table should not contain any multivalued attribute.

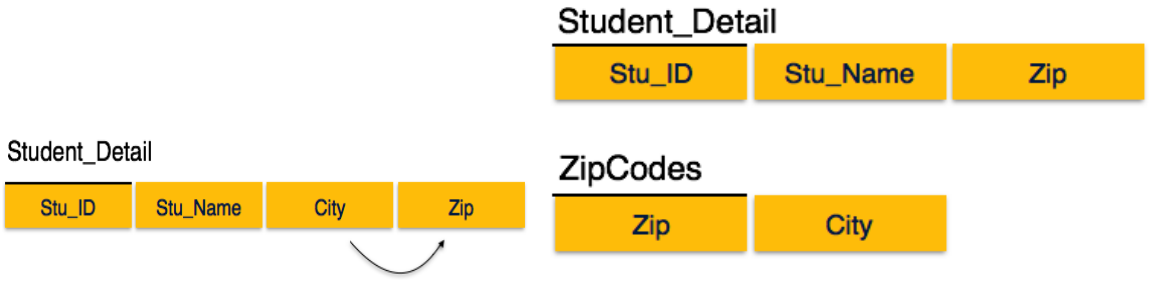
Course	Content
Programming	Java, c++
Web	HTML, PHP, ASP

Course	Content
Programming	Java
Programming	c++
Web	HTML
Web	PHP
Web	ASP

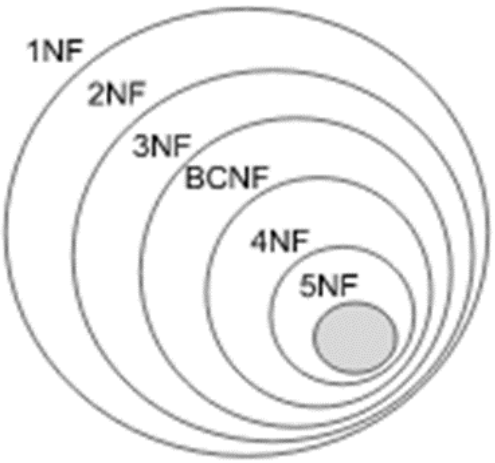
- **Second Normal Form:** 1NF + No Partial Dependencies by ensuring that non-prime attributes depend on the entire primary key.



- **Third Normal Form:** 2NF + No Transitive Dependencies by ensuring that non-key attributes depend only on the primary key and not on other non-key attributes.



- **Boyce-Codd Normal Form (BCNF):** It is a stricter form of 3NF that ensures that each non-key attribute is dependent only on the candidate key.
- **Fourth Normal Form (4NF):** BCNF + No multivalued dependencies.
- **Fifth Normal Form (5NF):** 5NF is the highest level of normalization and involves decomposing a table into smaller tables to remove data redundancy and improve data integrity.



Nested Query: Query inside a query, also known as non-correlated or independent nested query.

- Use in WHERE and HAVING conditions.
- No reference to outer query.

Example 1: Students with a high grade

```
-- student details with a high grade
select StdNo, StdFirstName, StdLastName, StdMajor, StdGPA
from Student
where Student.StdNo IN
( select StdNo from Enrollment where EnrGrade >= 3.5);
```

StdNo	StdFirstName	StdLastName	StdMajor	StdGPA
123-45-6789	HOMER	WELLS	IS	3.00
124-56-7890	BOB	NORBERT	FIN	2.70
234-56-7890	CANDY	KENDALL	ACCT	3.50
567-89-0123	MARIAH	DODGE	IS	3.60
789-01-2345	ROBERTO	MORALES	FIN	2.50
890-12-3456	LUKE	BRAZZI	IS	2.20
901-23-4567	WILLIAM	PILGRIM	IS	3.80
NULL	NULL	NULL	NULL	NULL

Example 2: Student details and grade for students with a high grade in a fall 20219 offering.

```
-- Student details and grade for students with a high grade in a fall 2019 offering
SELECT StdFirstName, StdLastName, StdCity, EnrGrade
FROM Student INNER JOIN Enrollment
ON Student.StdNo = Enrollment.StdNo
WHERE EnrGrade >= 3.5
AND Enrollment.OfferNo IN
( SELECT OfferNo FROM Offering
WHERE OffTerm = 'FALL'
AND OffYear = 2019 );
```

StdFirstName	StdLastName	StdCity	EnrGrade
CANDY	KENDALL	TACOMA	3.50
MARIAH	DODGE	SEATTLE	3.80
HOMER	WELLS	SEATTLE	3.50
ROBERTO	MORALES	SEATTLE	3.50

Before Deletion of Offering Table:

OfferNo	CourseNo	OffTerm	OffYear	OffLocation	OffTime	FacNo	OffDays
1111	IS320	SUMMER	2020	BLM302	10:30:00	NULL	MW
1234	IS320	FALL	2019	BLM302	10:30:00	098-76-5432	MW
2222	IS460	SUMMER	2019	BLM412	13:30:00	NULL	TTH
3333	IS320	SPRING	2020	BLM214	08:30:00	098-76-5432	MW
4321	IS320	FALL	2019	BLM214	15:30:00	098-76-5432	TTH
4444	IS320	WINTER	2020	BLM302	15:30:00	543-21-0987	TTH
5555	FIN300	WINTER	2020	BLM207	08:30:00	765-43-2109	MW
5678	IS480	WINTER	2020	BLM302	10:30:00	987-65-4321	MW
5679	IS480	SPRING	2020	BLM412	15:30:00	876-54-3210	TTH
6666	FIN450	WINTER	2020	BLM212	10:30:00	987-65-4321	TTH
7777	FIN480	SPRING	2020	BLM305	13:30:00	765-43-2109	MW
8888	IS320	SUMMER	2020	BLM405	13:30:00	654-32-1098	MW
9876	IS460	SPRING	2020	BLM307	13:30:00	654-32-1098	TTH
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Example 3: Delete offerings taught by Leonard Vince.

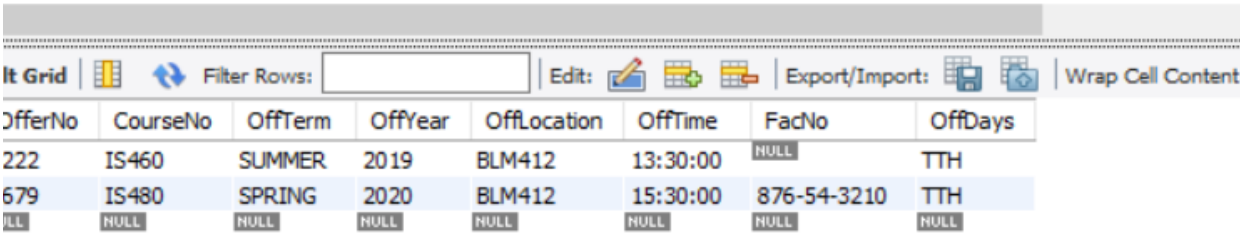
```
-- Delete offerings taught by Leonard Vince
DELETE FROM Offering
WHERE Offering.FacNo IN
( SELECT FacNo FROM Faculty
WHERE FacFirstName = 'JUDY'
AND FacLastName = 'CHAN' );
```

OfferNo	CourseNo	OffTerm	OffYear	OffLocation	OffTime	FacNo	OffDays
1111	IS320	SUMMER	2020	BLM302	10:30:00	NULL	MW
1234	IS320	FALL	2019	BLM302	10:30:00	098-76-5432	MW
2222	IS460	SUMMER	2019	BLM412	13:30:00	NULL	TTH
3333	IS320	SPRING	2020	BLM214	08:30:00	098-76-5432	MW
4321	IS320	FALL	2019	BLM214	15:30:00	098-76-5432	TTH
4444	IS320	WINTER	2020	BLM302	15:30:00	543-21-0987	TTH
5555	FIN300	WINTER	2020	BLM207	08:30:00	765-43-2109	MW
5678	IS480	WINTER	2020	BLM302	10:30:00	987-65-4321	MW
5679	IS480	SPRING	2020	BLM412	15:30:00	876-54-3210	TTH
6666	FIN450	WINTER	2020	BLM212	10:30:00	987-65-4321	TTH
7777	FIN480	SPRING	2020	BLM305	13:30:00	765-43-2109	MW
8888	IS320	SUMMER	2020	BLM405	13:30:00	654-32-1098	MW
9876	IS460	SPRING	2020	BLM307	13:30:00	654-32-1098	TTH
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Example 4: Update the location of offerings taught by Leonard Vince.

```
-- Update the location of offerings taught by Leonard Vince.
UPDATE Offering
SET OffLocation = 'BLM412'
WHERE OffYear = 2020
AND FacNo IN
( SELECT FacNo FROM Faculty
WHERE FacFirstName = 'JUDY'
AND FacLastName = 'CHAN');

select * from Offering where OffLocation = 'BLM412';
```



Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content

OfferNo	CourseNo	OffTerm	OffYear	OffLocation	OffTime	FacNo	OffDays
222	IS460	SUMMER	2019	BLM412	13:30:00	NULL	TTH
679	IS480	SPRING	2020	BLM412	15:30:00	876-54-3210	TTH
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

View

- View is a database object which is a kind of virtual table based on the result-set of an SQL statement.
- View does not store any data but it will contain only the structure of the table.
- A view contains rows and columns, just like a real table.
- The fields in a view are fields from one or more real tables in the database.
- You can add SQL statements and functions to a view and present the data as if the data were coming from one single table. A view is created with the CREATE VIEW statement.
- It is a powerful tool for data abstraction and simplifying complex queries.

  Syntax:

```
CREATE VIEW view_name AS
SELECT column1, column2, ...
FROM table_name
WHERE condition;
SELECT * FROM view_name;
```

Update a View: In some cases, you can update the underlying tables through a view. If the view's defined with the proper conditions and allows for updates, you can modify the data in the underlying tables by updating or inserting rows through the view.

UPDATE SaleEmployees SET FirstName = 'John' WHERE EmployeeID = 123;

DROP VIEW view_name;

Example:

```
-- View
-- creation of view
 create view std_view as select * from student;

 -- Using the view
select * from std_view;
```

StdNo	StdFirstName	StdLastName	StdCity	StdState	StdZip	StdMajor	StdClass	StdGPA
123-45-6789	HOMER	WELLS	SEATTLE	WA	98121-1111	IS	FR	3.00
124-56-7890	BOB	NORBERT	BOTHELL	WA	98011-2121	FIN	JR	2.70
234-56-7890	CANDY	KENDALL	TACOMA	WA	99042-3321	ACCT	JR	3.50
345-67-8901	WALLY	KENDALL	SEATTLE	WA	98123-1141	IS	SR	2.80
456-78-9012	JOE	ESTRADA	SEATTLE	WA	98121-2333	FIN	SR	3.20
567-89-0123	MARIAH	DODGE	SEATTLE	WA	98114-0021	IS	JR	3.60
678-90-1234	TESS	DODGE	REDMOND	WA	98116-2344	ACCT	SO	3.30
789-01-2345	ROBERTO	MORALES	SEATTLE	WA	98121-2212	FIN	JR	2.50
876-54-3210	CRISTOPHER	COLAN	SEATTLE	WA	98114-1332	IS	SR	4.00
890-12-3456	LUKE	BRAZZI	SEATTLE	WA	98116-0021	IS	SR	2.20
901-23-4567	WILLIAM	PILGRIM	BOTHELL	WA	98113-1885	IS	SO	3.80

ALTER VIEW in SQL

- The ALTER VIEW statement modifies an existing view by altering a reference type column to add a scope.

```
8     -- Create Product Table
9     CREATE TABLE Product (
10    pro_id int PRIMARY KEY AUTO_INCREMENT,
11    pro_name VARCHAR(100),
12    pro_price int );
13
14    -- Inserting a data in product table
15    INSERT INTO Product(pro_name, pro_price)
16    VALUES ("Milk", 54), ("Rice", 70);
17
18    select * from Product;
```

Result Grid | Filter Rows: | Edit:

pro_id	pro_name	pro_price
1	Milk	54
2	Rice	70
NULL	NULL	NULL

```
20      -- View Creation
21 •    CREATE VIEW product_details
22      AS SELECT * FROM Product;
23
24 •    SELECT * FROM product_details;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content

pro_id	pro_name	pro_price
1	Milk	54
2	Rice	70

```
26 •  ALTER TABLE Product ADD COLUMN pro_expiry DATE;
27
28 •  SELECT * FROM Product;
```

Result Grid | Filter Rows: | Edit: | Export/Import:

pro_id	pro_name	pro_price	pro_expiry
1	Milk	54	NULL
2	Rice	70	NULL
NULL	NULL	NULL	NULL

Note: when the record in table is changed it will automatically reflect those in view but, if we change the table structure then, we should include the REPLACE clause to show the updated structure.

```
34 •  CREATE OR REPLACE VIEW product_details
35    AS SELECT * FROM Product;
36 •  SELECT * FROM product_details;
37
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

pro_id	pro_name	pro_price	pro_expiry
1	Milk	54	NULL
2	Rice	70	NULL

Triggers

- A trigger is a user-defined SQL command that resides in system memory and it will be invoked automatically in response to an event such as insert, delete, or update. Each trigger is always associated with a table.

Syntax:

```
create trigger [trigger_name]
[before | after]
{insert | update | delete}
on [table_name]
[for each row]
[trigger_body]
```

Delimiter: The delimiter is changed to // to enable the entire definition to be passed to the server as a single statement, and then restored to; before invoking the procedure. This enables the delimiter used in the procedure body to be passed through to the server rather than being interpreted by MYSQL itself.