Cauvery College for Women (Autonomous)
Nationally Accredited (III Cycle) with 'A' Grade by NAAC
Annamalai Nagar, Tiruchirappalli-18.

Name of the Faculty : A.ANANDHAVALLI

Designation : Asst Professor

Department : Computer Applications

Contact Number : 9445181304

Programme : BCA

Batch : 2018-2019 Onwards

Semester : IV

Course : Database Management Systems

Course Code : 16SCCCA4

Unit : IV, V

Topics Covered : Relational Calculus, Entity- Relationship Model, Mapping &Key Constraints , E-R design issues, Normalizations , Funtional dependencies.

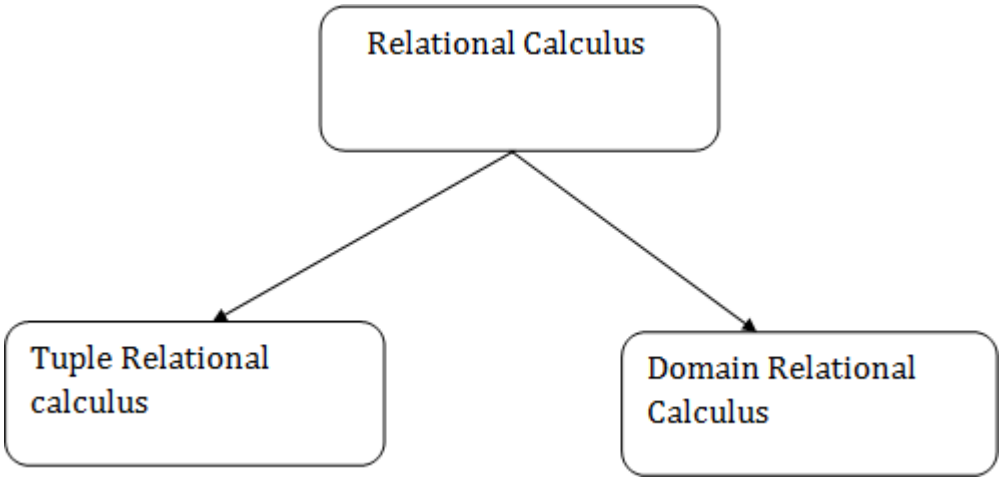
UNIT IV

Relational Calculus

- Relational calculus is a non-procedural query language. In the non-procedural query language, the user is concerned with the details of how to obtain the end results.

- The relational calculus tells what to do but never explains how to do.

Types of Relational calculus:



1. Tuple Relational Calculus (TRC)

- The tuple relational calculus is specified to select the tuples in a relation. In TRC, filtering variable uses the tuples of a relation.
- The result of the relation can have one or more tuples.

Notation:

1. {T | P (T)} or {T | Condition (T)}

Where

T is the resulting tuples

P(T) is the condition used to fetch T.

For example:

1. { T.name | Author(T) AND T.article = 'database' }

OUTPUT: This query selects the tuples from the AUTHOR relation. It returns a tuple with 'name' from Author who has written an article on 'database'.

TRC (tuple relation calculus) can be quantified. In TRC, we can use Existential (∃) and Universal Quantifiers (∀).

2. Domain Relational Calculus (DRC)

- The second form of relation is known as Domain relational calculus. In domain relational calculus, filtering variable uses the domain of attributes.
- Domain relational calculus uses the same operators as tuple calculus. It uses logical connectives ∧ (and), ∨ (or) and ¬ (not).
- It uses Existential (∃) and Universal Quantifiers (∀) to bind the variable.

Notation:

1. { a1, a2, a3, ..., an | P (a1, a2, a3, ... ,an)}

Where

a1, a2 are attributes
P stands for formula built by inner attributes

For example:

{< article, page, subject > | ∈ javatpoint ∧ subject = 'database'}

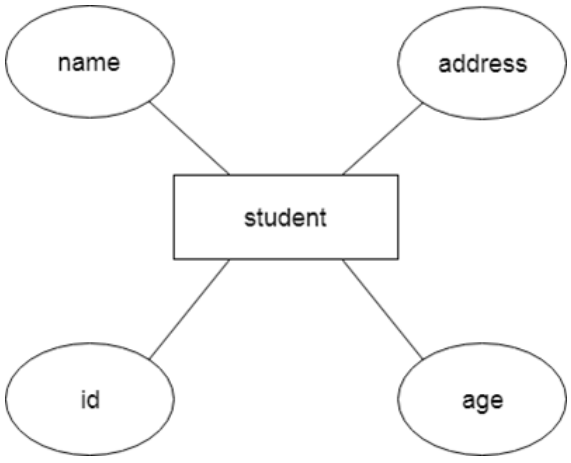
Output: This query will yield the article, page, and subject from the relational javatpoint, where the subject is a database.

Entity–relationship model (ER model)

ER model

- ER model stands for an Entity-Relationship model. It is a high-level data model. This model is used to define the data elements and relationship for a specified system.
- It develops a conceptual design for the database. It also develops a very simple and easy to design view of data.
- In ER modeling, the database structure is portrayed as a diagram called an entity-relationship diagram.

For example, Suppose we design a school database. In this database, the student will be an entity with attributes like address, name, id, age, etc. The address can be another entity with attributes like city, street name, pin code, etc and there will be a relationship between them.



Symbols in the ER diagram:

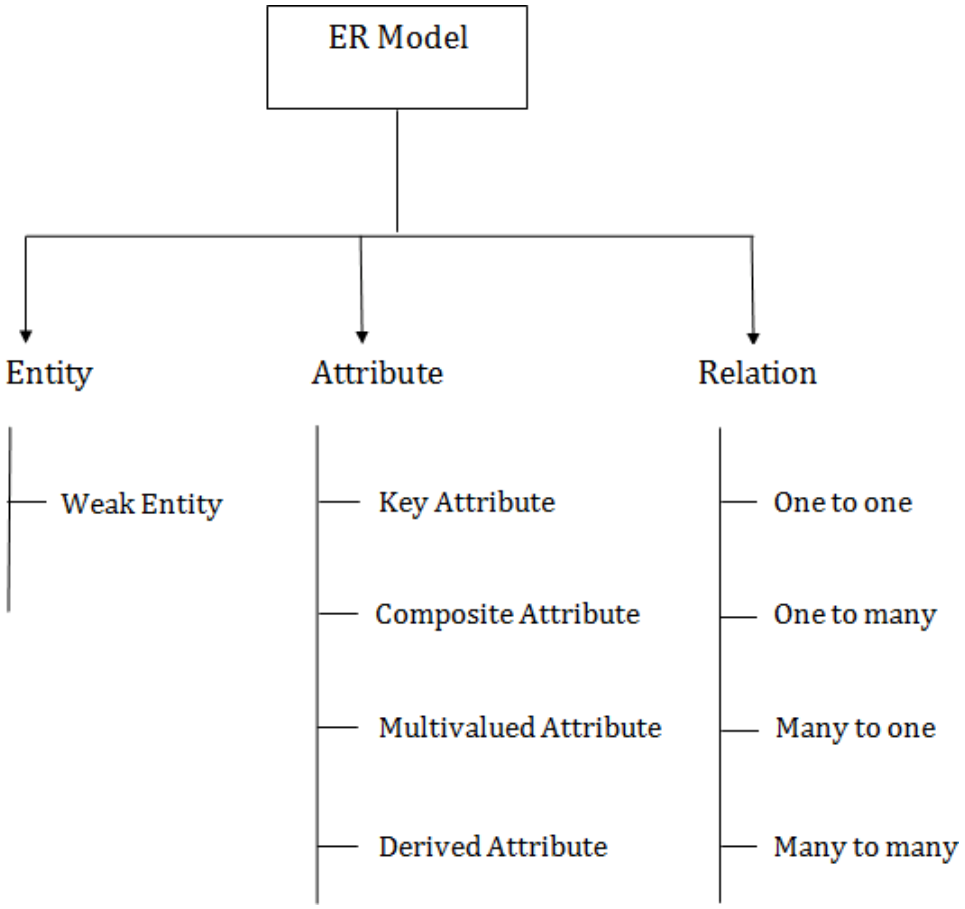
Rectangle: Represents Entity sets.
Ellipses: Attributes
Diamonds: Relationship Set
Lines: They link attributes to Entity Sets and Entity sets to Relationship Set

Double Ellipses: Multivalued Attributes
Dashed Ellipses: Derived Attributes
Double Rectangles: Weak Entity Sets
Double Lines: Total participation of an entity in a relationship set

Components of ER Diagram

An ER diagram has three main components:

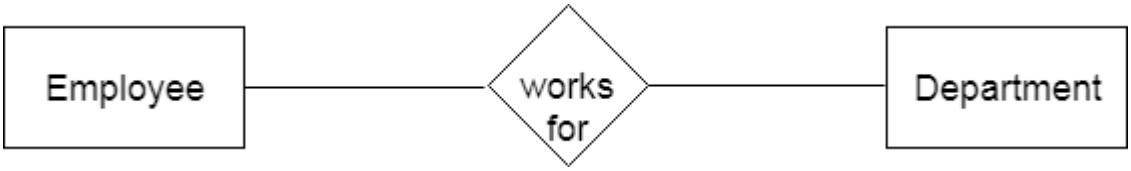
1. Entity
2. Attribute
3. Relationship



1. Entity:

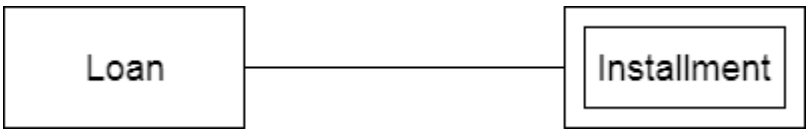
An entity may be any object, class, person or place. In the ER diagram, an entity can be represented as rectangles.

Consider an organization as an example- manager, product, employee, department etc. can be taken as an entity.



a. Weak Entity

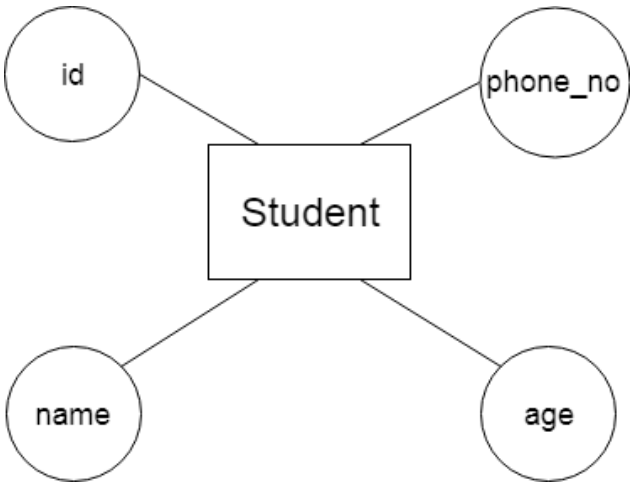
An entity that depends on another entity called a weak entity. The weak entity doesn't contain any key attribute of its own. The weak entity is represented by a double rectangle.



2. Attribute

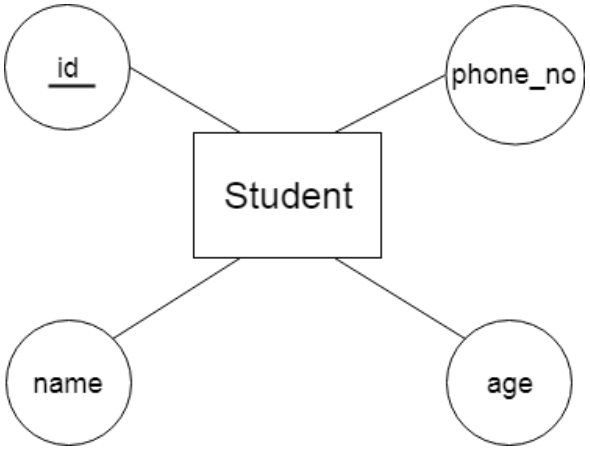
The attribute is used to describe the property of an entity. Eclipse is used to represent an attribute.

For example, id, age, contact number, name, etc. can be attributes of a student.



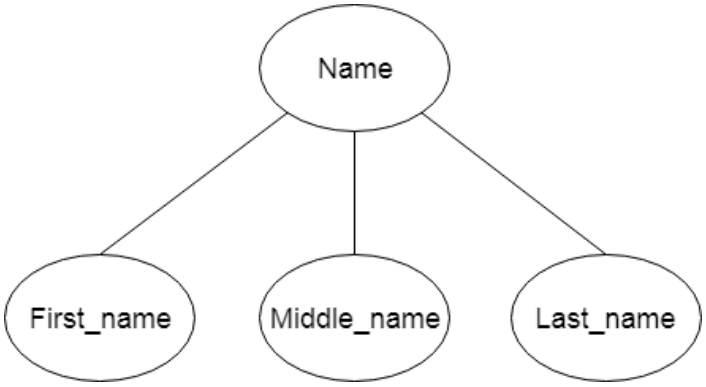
a. Key Attribute

The key attribute is used to represent the main characteristics of an entity. It represents a primary key. The key attribute is represented by an ellipse with the text underlined.



b. Composite Attribute

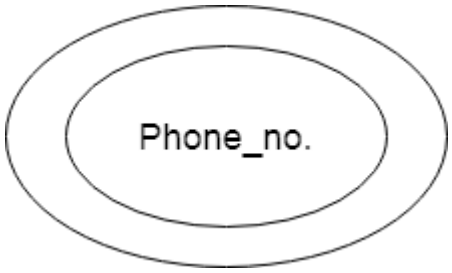
An attribute that composed of many other attributes is known as a composite attribute. The composite attribute is represented by an ellipse, and those ellipses are connected with an ellipse.



c. Multivalued Attribute

An attribute can have more than one value. These attributes are known as a multivalued attribute. The double oval is used to represent multivalued attribute.

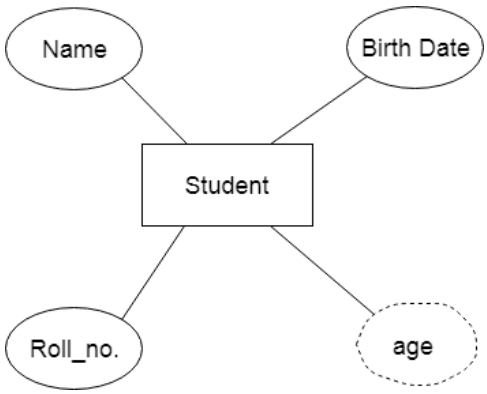
For example, a student can have more than one phone number.



d. Derived Attribute

An attribute that can be derived from other attribute is known as a derived attribute. It can be represented by a dashed ellipse.

For example, A person's age changes over time and can be derived from another attribute like Date of birth.



3. Relationship

A relationship is used to describe the relation between entities. Diamond or rhombus is used to represent the relationship.

Types of relationship are as follows:

a. One-to-One Relationship

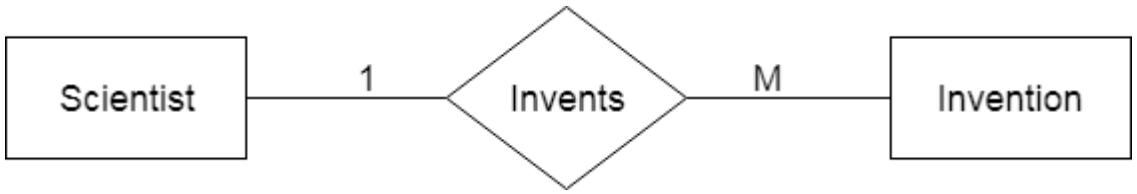
When only one instance of an entity is associated with the relationship, then it is known as one to one relationship.

For example, A female can marry to one male, and a male can marry to one female.

b. One-to-many relationship

When only one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then this is known as a one-to-many relationship.

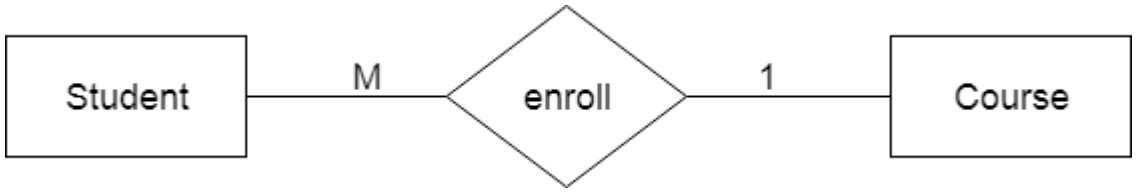
For example, Scientist can invent many inventions, but the invention is done by the only specific scientist.



c. Many-to-one relationship

When more than one instance of the entity on the left, and only one instance of an entity on the right associates with the relationship then it is known as a many-to-one relationship.

For example, Student enrolls for only one course, but a course can have many students.



d. Many-to-many relationship

When more than one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then it is known as a many-to-many relationship.

For example, Employee can assign by many projects and project can have many employees.

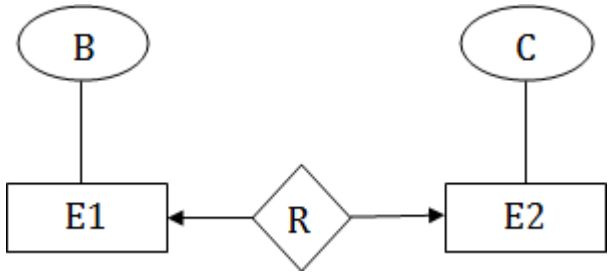
Mapping Constraints

- A mapping constraint is a data constraint that expresses the number of entities to which another entity can be related via a relationship set.

- It is most useful in describing the relationship sets that involve more than two entity sets.
- For binary relationship set R on an entity set A and B, there are four possible mapping cardinalities. These are as follows:
    1. One to one (1:1)
    2. One to many (1:M)
    3. Many to one (M:1)
    4. Many to many (M:M)

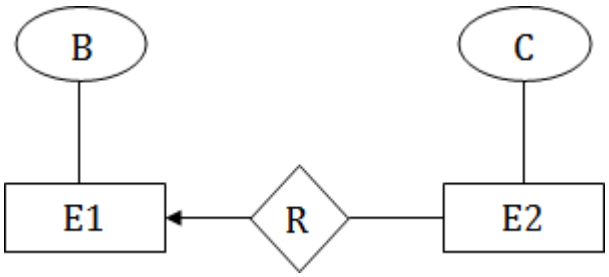
One-to-one

In one-to-one mapping, an entity in E1 is associated with at most one entity in E2, and an entity in E2 is associated with at most one entity in E1.



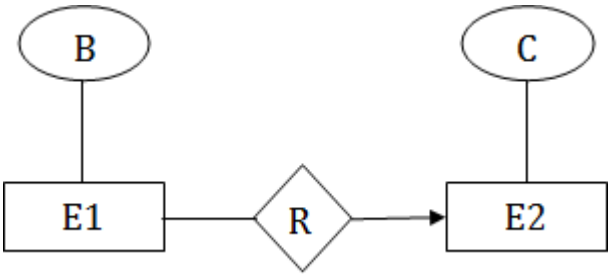
One-to-many

In one-to-many mapping, an entity in E1 is associated with any number of entities in E2, and an entity in E2 is associated with at most one entity in E1.



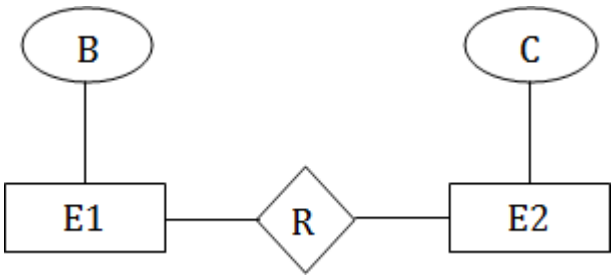
Many-to-one

In one-to-many mapping, an entity in E1 is associated with at most one entity in E2, and an entity in E2 is associated with any number of entities in E1.



Many-to-many

In many-to-many mapping, an entity in E1 is associated with any number of entities in E2, and an entity in E2 is associated with any number of entities in E1.



Key constraints

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

Example:

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

Not allowed. Because all row must be unique

Entity Relationship Design Issues

However, users often mislead the concept of the elements and the design process of the ER diagram. Thus, it leads to a complex structure of the ER diagram and certain issues that does not meet the characteristics of the real-world enterprise model.

The basic design issues of an ER database schema in the following

1) Use of Entity Set vs Attributes

The use of an entity set or attribute depends on the structure of the real-world enterprise that is being modelled and the semantics associated with its attributes. It leads to a mistake when the user use the primary key of an entity set as an attribute of another entity set. Instead, he should use the relationship to do so. Also, the primary key attributes are implicit in the relationship set, but we designate it in the relationship sets.

2) Use of Entity Set vs. Relationship Sets

It is difficult to examine if an object can be best expressed by an entity set or relationship set. To understand and determine the right use, the user need to designate a relationship set for describing an action that occurs in-between the entities. If there is a requirement of representing the object as a relationship set, then its better not to mix it with the entity set.

3) Use of Binary vs n-ary Relationship Sets

Generally, the relationships described in the databases are binary relationships. However, non-binary relationships can be represented by several binary relationships. For example, we can create and represent a ternary relationship 'parent' that may relate to a child, his father, as well as his mother. Such relationship can also be represented by two binary relationships i.e, mother and father, that may relate to their child. Thus, it is possible to represent a non-binary relationship by a set of distinct binary relationships.

4) Placing Relationship Attributes

The cardinality ratios can become an affective measure in the placement of the relationship attributes. So, it is better to associate the attributes of one-to-one or one-to-many relationship sets with any participating entity sets, instead of any relationship set. The decision of placing the specified attribute as a relationship or entity attribute should possess the charactestics of the real world enterprise that is being modelled.

For example, if there is an entity which can be determined by the combination of participating entity sets, instead of determining it as a separate entity. Such type of attribute must be associated with the many-to-many relationship sets.

Thus, it requires the overall knowledge of each part that is involved in designing and modelling an ER diagram. The basic requirement is to analyse the real-world enterprise and the connectivity of one entity or attribute with other.

UNIT V

What is Normalization?

Database Normalization is a technique of organizing the data in the database. Normalization is a systematic approach of decomposing tables to eliminate data redundancy(repetition). It is a multi-step process that puts data into tabular form, removing duplicated data from the relation tables.

Write the uses of Normalization.

Normalization is used for mainly two purposes,

- Eliminating redundant(useless) data.

- Ensuring data dependencies make sense i.e data is logically stored.

What do you mean by keys?

A key is a value used to identify a record in a table uniquely.

Normalization are divided into the following normal forms:

First Normal Form

Second Normal Form

Third Normal Form

BCNF

Fourth Normal Form

First Normal Form (1NF)

The first normal form is the most basic among the normal forms. It imposes a very basic requirement on relations. A domain is atomic if elements of the domain are considered to be indivisible units. Examples of non-atomic domains are address (consists of street name, pin code etc), roll numbers in our institute (first two digits tell the year of joining).Integers are assumed to be atomic as they cannot be divisible. A schema is in First normal form if the domain of every attribute is atomic.

Example Relation Student= (StudentID, Department, YearOfJoining, SName) is in first normal form as every attribute is indivisible.

However the relation Student= (StudentID, Department, Sname) with student id of the form 07 012345 or 08 123456 is not in first normal form as student ID is used to find the year of joining.

Second Normal Form (2NF)

For a table to be in the Second Normal Form,

It should be in the First Normal form.

And, it should not have Partial Dependency.

Third Normal Form (3NF)

A table is said to be in the Third Normal Form when,

It is in the Second Normal form.

And, it doesn't have Transitive Dependency.

Boyce and Codd Normal Form (BCNF)

Boyce and Codd Normal Form is a higher version of the Third Normal form. This form deals with certain type of anomaly that is not handled by 3NF. A 3NF table which does not have multiple overlapping candidate keys is said to be in BCNF.

Rules for BCNF

For a table to satisfy the Boyce-Codd Normal Form, it should satisfy the following two conditions:

1. It should be in the **Third Normal Form**.
2. And, for any dependency A → B, A should be a **super key**.

In simple words, it means, that for a dependency A → B, A cannot be a **non-prime attribute**, if B is a **prime attribute**.

Functional Dependencies

Functional Dependencies are very important in distinguishing good database designs from bad database designs.

They are constraints on the set of Legal relations.

A legal instance r(R) of Schema R satisfies Functional Dependency α →β if

1. Both α and β are subsets of R

2. For all pairs of tuples t1 and t2 in the instance if t1[α]=t2[α] then t1[β]=t2[β]

In schema Courses=(CourseID, Instructor, Credits) if α=CourseID and β=Credits then α ->β forms a functional dependency.

Trivial Case: If β is a subset of α then α ->β is a functional dependency.

Super, Primary and Candidate Keys are types of functional dependencies.

That is if any of these keys are present in alpha then α ->β holds for any β in the relation.

A single functional dependency on a relation R will not be very useful but if we have all the functional dependencies they are very useful in checking normal forms. Given a set of functional dependencies F we can find a larger set of functional dependencies using that set.

Axioms allow us to reason about functional dependencies with ease.

The following 3 axioms known as Armstrong's axioms are used repeatedly: α, β and γ are set of attributes.

1. Reflexivity: if β ⊆ α then α →β holds

2. Augmentation: If α →β then γα → γβ holds

3. Transitivity: If α →β holds and β → γ holds, then α → γ hold

.