



UNIT III – RELATIONAL DATABASE MODEL

4 HRS

Structure of RDBMS

- ✓ A **relational database** is a type of database. It uses a structure that allows us to identify and access data in relation to another piece of data in the database. Often, data in a relational database is organized into tables.
- ✓ A **relational database management system (RDBMS)** is a program that allows you to create, update, and administer a relational database. Most relational database management systems use the SQL language to access the database.
- ✓ MySQL, SQL-Server, MS-Access, Oracle are some of popular RDBMS.
- ✓ Examples of RDBMS structure:

<u>CUSTOMERS</u>	
customer_id	customer_name
101	John Doe
102	Bruce Wayne

<u>ORDERS</u>			
order_id	customer_id	order_date	amount
555	101	12/24/09	\$156.78
556	102	12/25/09	\$99.99
557	101	12/26/09	\$75.00

<u>STUDENT</u>					
STUD_NO	STUD_NAME	STUD_PHONE	STUD_STATE	STUD_COUNT RY	STUD_AG E
1	RAM	9716271721	Haryana	India	20
2	RAM	9898291281	Punjab	India	19
3	SUJIT	7898291981	Rajsthan	India	18
4	SURESH		Punjab	India	21

Table 1

<u>STUDENT_COURSE</u>		
STUD_NO	COURSE_NO	COURSE_NAME
1	C1	DBMS
2	C2	Computer Networks
1	C2	Computer Networks

Table 2

Database Schema

- ✓ A **database schema** is the skeleton structure that represents the logical view of the entire database. It defines how the data is organized and how the relations among them are associated. It formulates all the constraints that are to be applied on the data.
- ✓ A database schema defines its entities and the relationship among them. It contains a descriptive detail of the database, which can be depicted by means of schema diagrams. It's the database designers who design the schema to help programmers understand the database and make it useful.
- ✓ A database schema can be divided broadly into two categories –
- ✓ **Physical Database Schema** – This schema pertains to the actual storage of data and its form of storage like files, indices, etc. It defines how the data will be stored in a secondary storage.
- ✓ **Logical Database Schema** – This schema defines all the logical constraints that need to be applied on the data stored. It defines tables, views, and integrity constraints.

Schema Diagram for Employee Database

EMPLOYEE

Fname	Minit	Lname	<u>Ssn</u>	Bdate	Address	Sex	Salary	Super ssn	Dno
-------	-------	-------	------------	-------	---------	-----	--------	-----------	-----

DEPARTMENT

Dname	<u>Dnumber</u>	Mgr ssn	Mgr start date
-------	----------------	---------	----------------

DEPT_LOCATIONS

<u>Dnumber</u>	<u>Dlocation</u>
----------------	------------------

PROJECT

Pname	<u>Pnumber</u>	Plocation	Dnum
-------	----------------	-----------	------

WORKS_ON

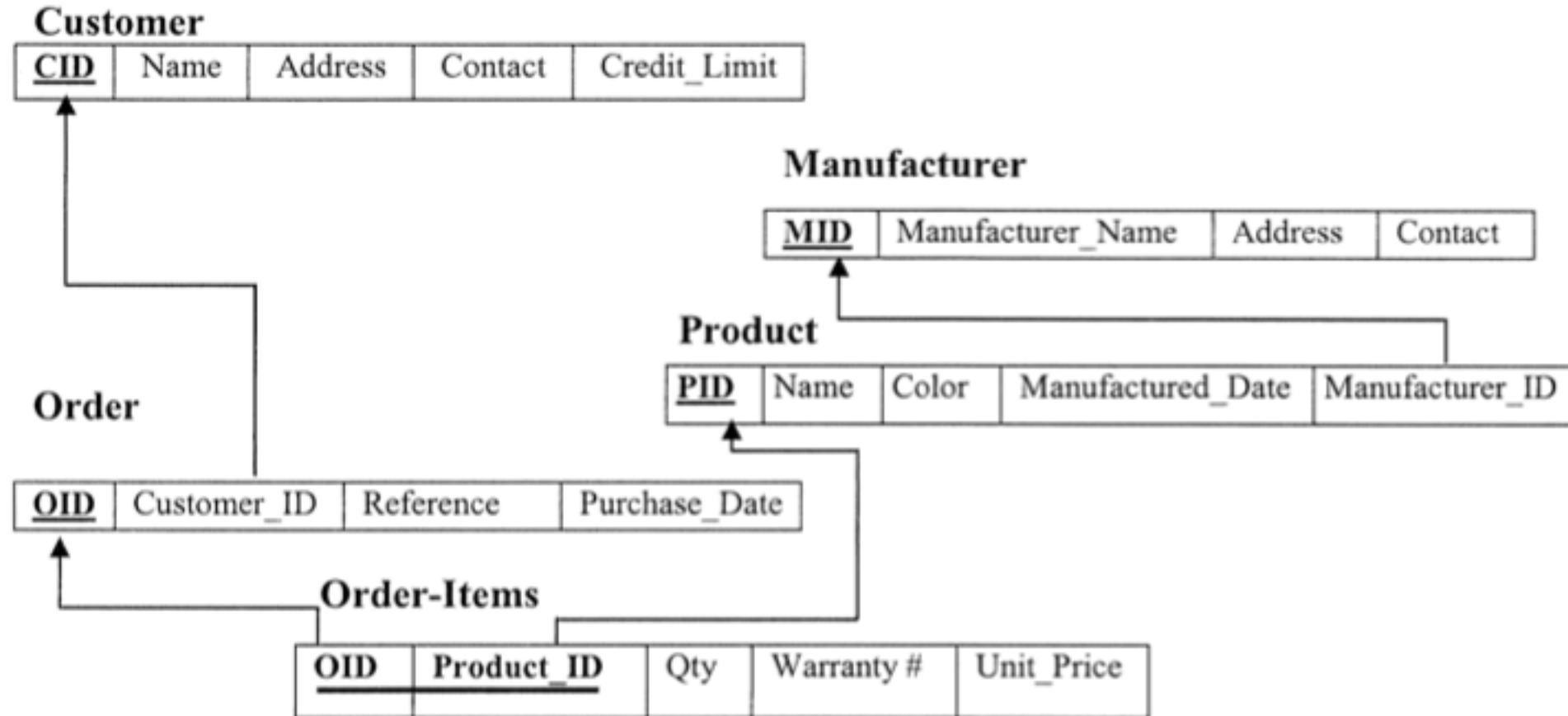
<u>Essn</u>	<u>Pno</u>	Hours
-------------	------------	-------

DEPENDENT

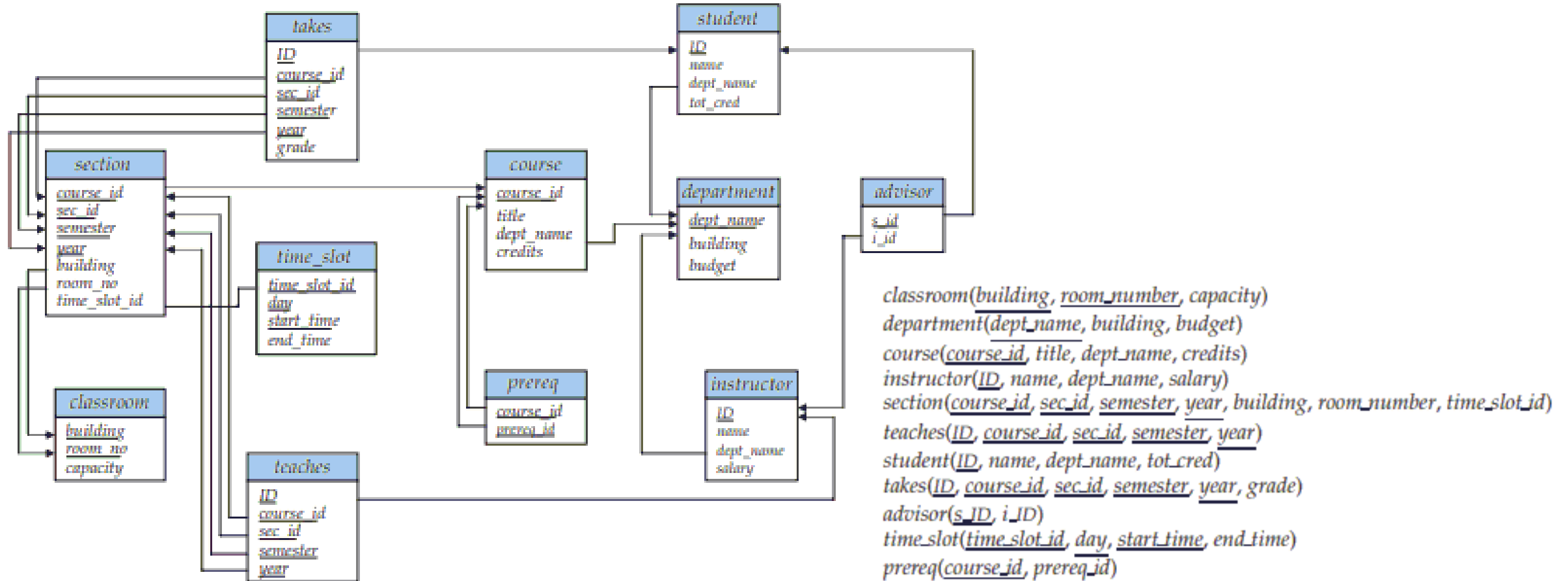
<u>Essn</u>	<u>Dependent name</u>	Sex	Bdate	Relationship
-------------	-----------------------	-----	-------	--------------

Schema diagram for the
COMPANY relational
database schema.

Schema Diagram for Order Database



Schema Diagram for University Database



Schema of the university database.

Keys

- ✓ A **DBMS key** is an attribute or set of an attribute which helps you to identify a row(tuple) in a relation(table).
- ✓ They allow you to find the relation between two tables. Keys help you uniquely identify a row in a table by a combination of one or more columns in that table.

1. Super key

- ✓ A **superkey** is a group of single or multiple keys which identifies rows in a table. A Super key may have additional attributes that are not needed for unique identification.

Example:

EmpSSN	EmpNum	Empname
9812345098	AB05	Shown
9876512345	AB06	Roslyn
199937890	AB07	James

- ✓ In the above-given example, **EmpSSN** and **EmpNum** name are **superkeys**.

Keys

2. Primary Key

- ✓ A column or group of columns in a table which helps us to uniquely identifies every row in that table is called a primary key.
- ✓ This DBMS can't be a duplicate. The same value can't appear more than once in the table.

Rules for defining Primary key:

- Two rows can't have the same primary key value
- It must for every row to have a primary key value.
- The primary key field cannot be null.
- The value in a primary key column can never be modified or updated if any foreign key refers to that primary key.

StudID	Roll No	First Name	LastName	Email
1	11	Tom	Price	abc@gmail.com
2	12	Nick	Wright	xyz@gmail.com
3	13	Dana	Natan	mno@yahoo.com

- ✓ Here, **StudID** is a Primary Key.

Keys

3. Candidate Key

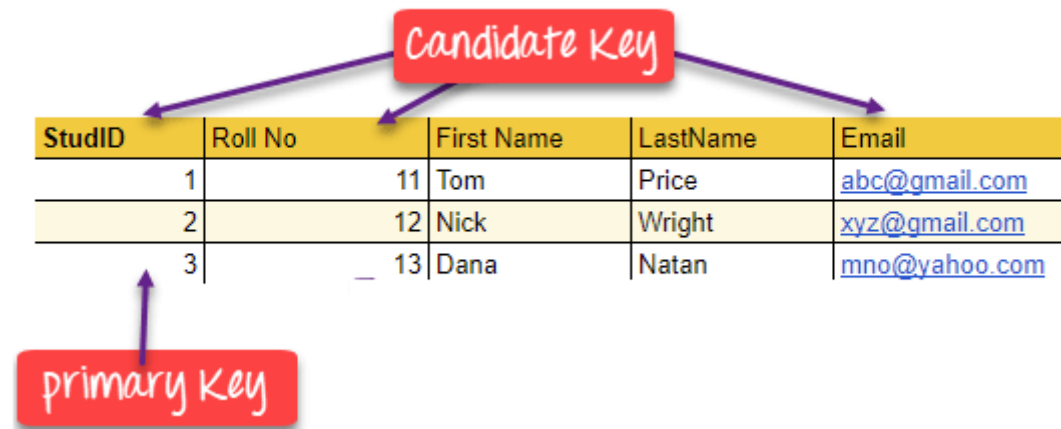
- ✓ A super key with no repeated attribute is called **candidate key**.
- ✓ The Primary key should be selected from the candidate keys. Every table must have at least a single candidate key.

Properties of Candidate key:

- ✓ It must contain unique values
- ✓ Candidate key may have multiple attributes
- ✓ Must not contain null values
- ✓ It should contain minimum fields to ensure uniqueness
- ✓ Uniquely identify each record in a table

Example: In the given table Stud ID, Roll No, and email are candidate keys which help us to uniquely identify the student record in the table.

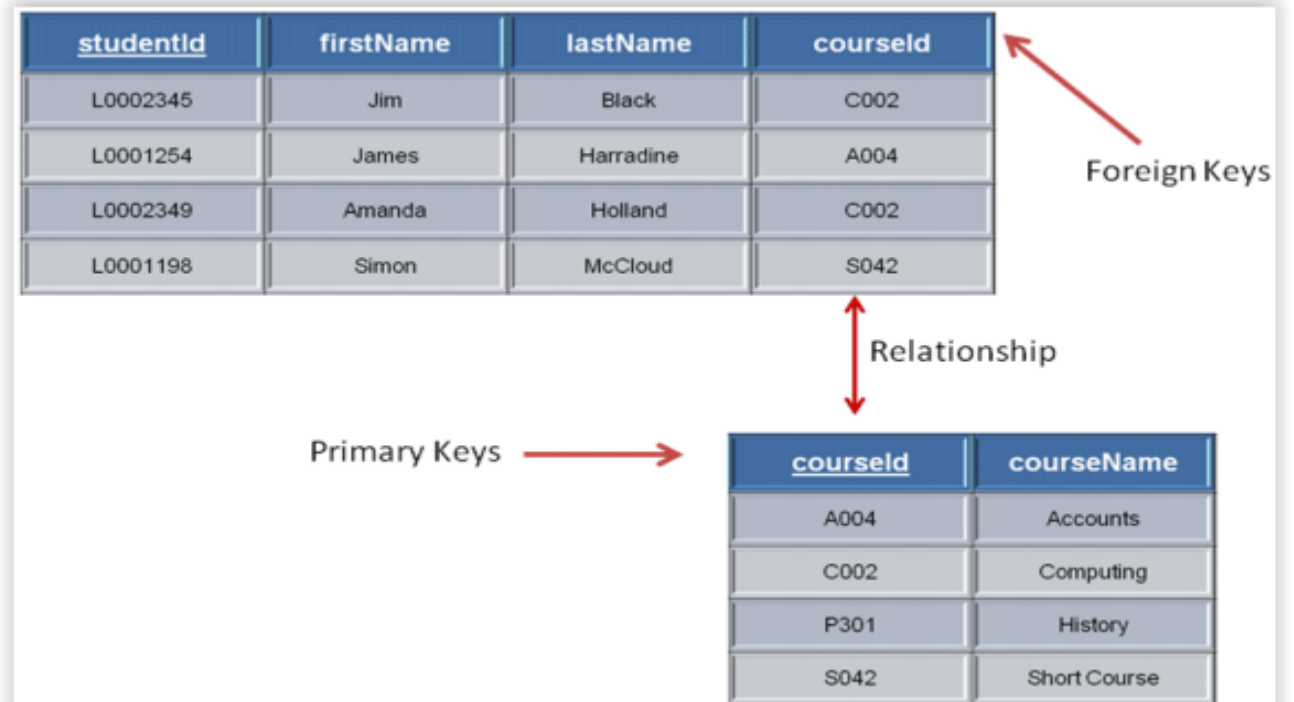
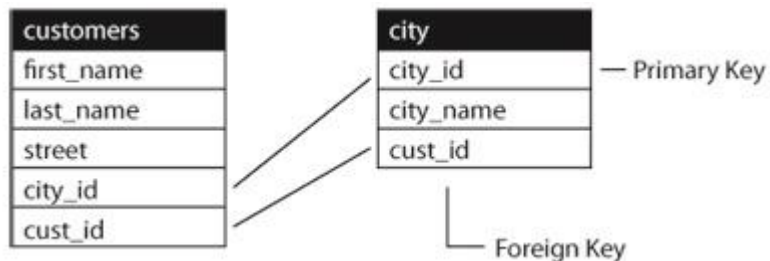
StudID	Roll No	First Name	LastName	Email
1	11	Tom	Price	abc@gmail.com
2	12	Nick	Wright	xyz@gmail.com
3	13	Dana	Natan	mno@yahoo.com



Keys

4. Foreign key

- ✓ A foreign key is a column which is added to create a relationship with another table. Foreign keys help us to maintain data integrity and also allows navigation between two different instances of an entity.
- ✓ Every relationship in the model needs to be supported by a foreign key.



Keys

5. Composite key

- ✓ A key which has multiple attributes to uniquely identify rows in a table is called a composite key.

6. Alternate key

- ✓ All the keys which are not primary key are called an alternate key. It is a key which is currently not the primary key.
- ✓ However, A table may have single or multiple choices for the primary key.

ITEM

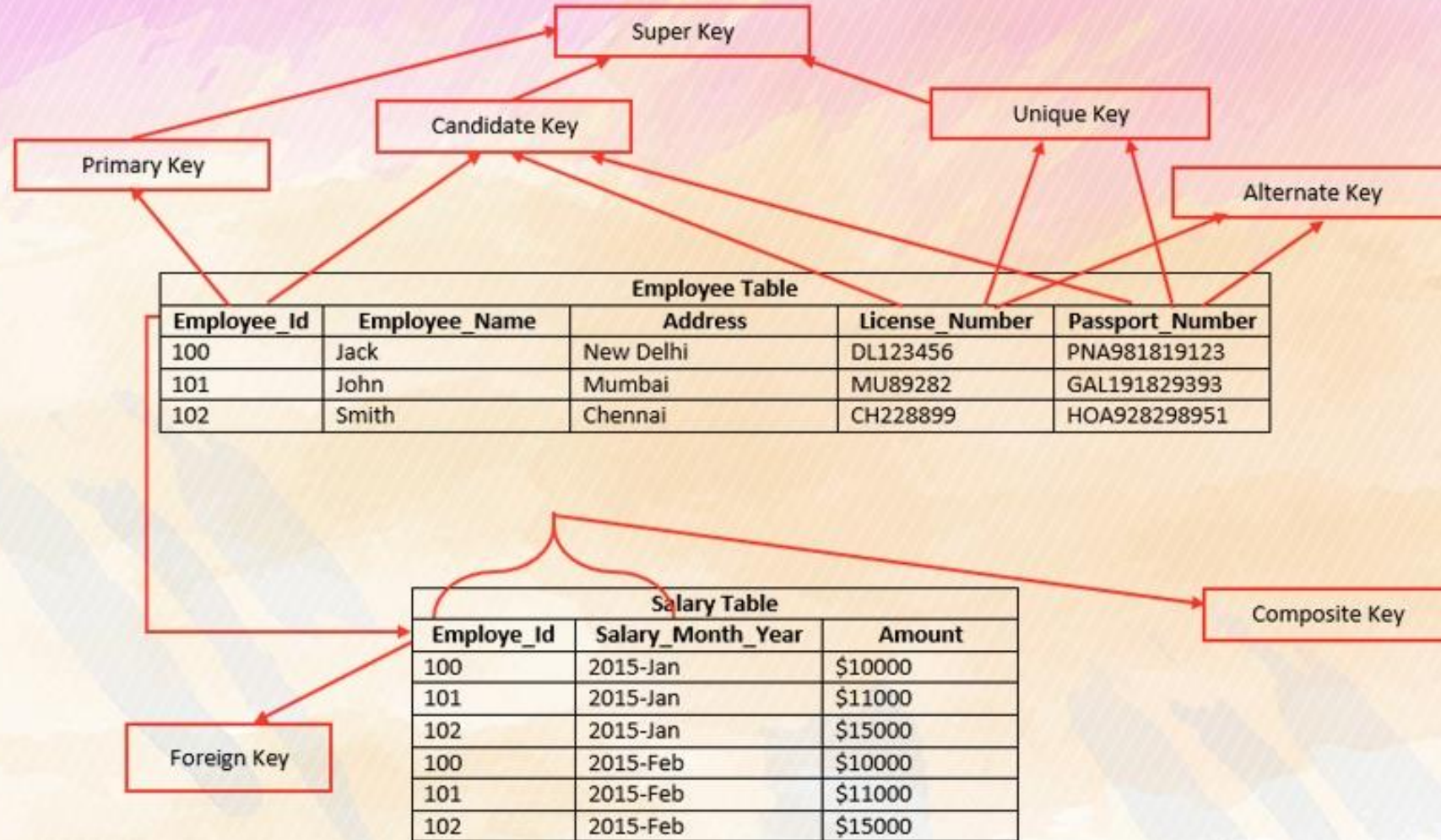
Composite Key

Supplier_ID	Item_ID	Item_Name	Quantity
S ₁	I ₁	AC	5
S ₁	I ₂	Inverter	8
S ₂	I ₂	Inverter	4
S ₂	I ₃	UPS	15
S ₂	I ₄	Generator	5
S ₃	I ₃	UPS	10

StudID, Roll No, Email are qualified to become a primary key. But since StudID is the primary key, Roll No, Email becomes the alternative key.

StudID	Roll No	First Name	LastName	Email
1	11	Tom	Price	abc@gmail.com
2	12	Nick	Wright	xyz@gmail.com
3	13	Dana	Natan	mno@yahoo.com

Summary



Relational Algebra

- ✓ Every database management system must define a query language to allow users to access the data stored in the database.
- ✓ Relational Algebra is a procedural query language used to query the database tables to access data in different ways.
- ✓ In relational algebra, input is a relation (table from which data has to be accessed) and output is also a relation (a temporary table holding the data asked for by the user).

We can use Relational Algebra to fetch data from this Table(relation)

Select Name students with age less than 17

Output

Name
Ckon
Dkon

ID	Name	Age
1	Akon	17
2	Bkon	19
3	Ckon	15
4	Dkon	13

The output for query is also in form of a table(relation), with results in different columns

Relational Algebra Operations

- ✓ The primary operations that we can perform using relational algebra are: **Select, Project, Union, Set Difference, Cartesian Product and Join.**

a) Select Operation (σ)

- ✓ This is used to fetch rows(tuples) from table(relation) which satisfies a given condition.
- ✓ Let's take an example of the Student table we specified above in the Introduction of relational algebra, and fetch data for students with age more than 17.

$\sigma_{\text{age} > 17} (\text{Student})$

- ✓ This will fetch the tuples(rows) from table Student, for which age will be greater than 17.
- ✓ You can also use, and, or operators, to specify two conditions, for example,

$\sigma_{\text{age} > 17 \text{ and gender} = \text{'Male'}} (\text{Student})$

- ✓ This will return tuples(rows) from table Student with information of male students, of age more than 17.

Relational Algebra Operations

b) Project Operation (Π)

- ✓ Project operation is used to project only a certain set of attributes of a relation. In simple words, If you want to see only the names all of the students in the Student table, then you can use Project Operation.
- ✓ It will only project or show the columns or attributes asked for, and will also remove duplicate data from the columns.
- ✓ For example,

$\Pi_{\text{Name, Age}}(\text{Student})$

- ✓ Above statement will show us only the Name and Age columns for all the rows of data in Student table.
- Relational algebra expressions

SQL	Result	Relational algebra						
<pre>select name, salary from E where salary < 200</pre>	<table><tr><th>name</th><th>salary</th></tr><tr><td>John</td><td>100</td></tr><tr><td>Tom</td><td>100</td></tr></table>	name	salary	John	100	Tom	100	<p>PROJECT_{name, salary} (SELECT_{salary < 200}(E))</p> <p><i>or, step by step, using an intermediate result</i></p> <p>Temp <- SELECT_{salary < 200}(E)</p> <p>Result <- PROJECT_{name, salary}(Temp)</p>
name	salary							
John	100							
Tom	100							

Relational Algebra Operations

c) Union Operation (\cup)

- ✓ This operation is used to fetch data from two relations (tables).
- ✓ For this operation to work, the relations (tables) specified should have same number of attributes (columns) and same attribute domain. Also the duplicate rows are automatically eliminated from the result.
- ✓ Syntax: **$A \cup B$** where A and B are relations.
- ✓ For example, if we have two tables **RegularClass** and **ExtraClass**, both have a column student to save name of student, then,
$$\Pi_{\text{Student}}(\text{RegularClass}) \cup \Pi_{\text{Student}}(\text{ExtraClass})$$
- ✓ Above operation will give us name of Students who are attending both regular classes and extra classes, eliminating repetition.

Relational Algebra Operations

d) Set Difference (-)

- ✓ This operation is used to find data present in one relation and not present in the second relation. This operation is also applicable on two relations, just like Union operation.
- ✓ **Syntax: $A - B$** where A and B are relations.
- ✓ For example, if we want to find name of students who attend the regular class but not the extra class, then, we can use the below operation:

$$\Pi_{\text{Student}}(\text{RegularClass}) - \Pi_{\text{Student}}(\text{ExtraClass})$$

e) Cartesian Product (X)

- ✓ This is used to combine data from two different relations (tables) into one and fetch data from the combined relation. **Syntax: $A \times B$**
- ✓ For example, if we want to find the information for Regular Class and Extra Class which are conducted during morning, then, we can use the following operation:

$$\sigma_{\text{time} = \text{'morning'}} (\text{RegularClass} \times \text{ExtraClass})$$

- ✓ For the above query to work, both RegularClass and ExtraClass should have the attribute time.

Relational Algebra Operations

f) Join

✓ Here are the different types of the JOINS in SQL:

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

