Unit: 4
Lecture: 3
Types of Keys in Relational Model and Codd Rules

STUDENT

STUD_NO	STUD_NAME	STUD_PHONE	STUD_STATE	STUD_COUNTRY	STUD_AGE
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

STUDENT_COURSE

STUD_NO	COURSE_NO	COURSE_NAME
1	C1	DBMS
2	C2	Computer Networks
1	C2	Computer Networks

Table 2

Candidate Key:
The minimal set of attributes that can uniquely identify a tuple is known as a candidate key.
e.g. STUD_NO in STUDENT relation.

- The value of the Candidate Key is unique and non-null for every tuple.
- There can be more than one candidate key in a relation e.g. Stud_No or Stud_Phone is the candidate key for relation Student.
- The candidate key can be simple (having only one attribute) or composite as well. e.g. {STUD_NO, COURSE_NO} is a composite candidate key for relation STUDENT_COURSE.
- Number of candidate keys in a relation are $^{n}C_{(floor(n/2))}$ e.g. if a relation have 5 attributes i.e. R(A,B,C,D,E) then total number of candidate keys are $^{5}C_{(floor(5/2)}$ =10.

Super Key: The set of attributes that can uniquely identify a tuple is known as Super Key e.g. STUD_NO, (STUD_NO, STUD_NAME), etc,

- Adding zero or more attributes to the candidate key generates the super key.
- A candidate key is a super key but vice versa is not true.

Primary Key: There can be more than one candidate key in a relation out of which one can be chosen as the Primary Key.
e.g. STUD_NO as well as STUD_PHONE both are candidate keys for relation STUDENT but STUD_NO can be chosen as the primary key (only one out of many candidate keys)

Alternate Key: The candidate key other than the primary key is called an alternate key. e.g. STUD_NO as well as STUD_PHONE both are candidate keys for relations STUDENT but STUD_PHONE will be an alternate key.

Foreign Key: If an attribute can only take the values which are present as values of some other attribute, it will be a foreign key to the attribute to which it refers.
e.g. STUD_NO in STUDENT_COURSE is a foreign key to STUD_NO in STUDENT relation.

Codd Rules

Codd rules were proposed by E.F. Codd which should be satisfied by relational model:

Rule 1: Information Rule: The data stored in a database, may it be user data or metadata, must be a value of some table cell. Everything in a database must be stored in a table format.

Rule 2: Guaranteed Access Rule: Every single data element (value) is guaranteed to be accessible logically with a combination of table-name, primary-key (row value), and attribute-name (column value). No other means, such as pointers, can be used to access data.

Rule 3: Systematic Treatment of NULL Values: The NULL values in a database must be given a systematic and uniform treatment. This is a very important rule because a NULL can be interpreted as one the following − data is missing, data is not known, or data is not applicable.

Rule 4: Active Online Catalog: The structure description of the entire database must be stored in an online catalog, known as **data dictionary**, which can be accessed by authorized users. Users can use the same query language to access the catalog which they use to access the database itself.

Rule 5: Comprehensive Data Sub-Language Rule: A database can only be accessed using a language having linear syntax that supports data definition, data manipulation, and transaction management operations. This language can be used directly or by means of some application. If the database allows access to data without any help of this language, then it is considered as a violation.

Rule 6: View Updating Rule: All the views of a database, which can theoretically be updated, must also be updatable by the system.

Rule 7: High-Level Insert, Update, and Delete Rule:A database must support high-level insertion, updation, and deletion. This must not be limited to a single row, that is, it must also support union, intersection and minus operations to yield sets of data records.

Rule 8: Physical Data Independence: The data stored in a database must be independent of the applications that access the database. Any change in the physical structure of a database must not have any impact on how the data is being accessed by external applications.

Rule 9: Logical Data Independence: The logical data in a database must be independent of its user's view (application). Any change in logical data must not affect the applications using it. For example, if two tables are merged or one is split into two different tables, there should be no impact or change on the user application. This is one of the most difficult rule to apply.

Rule 10: Integrity Independence: A database must be independent of the application that uses it. All its integrity constraints can be independently modified without the need of any change in the application. This rule makes a database independent of the front-end application and its interface.

Rule 11: Distribution Independence: The end-user must not be able to see that the data is distributed over various locations. Users should always get the impression that the data is located at one site only. This rule has been regarded as the foundation of distributed database systems.

Rule 12: Non-Subversion Rule: If a system has an interface that provides access to low-level records, then the interface must not be able to subvert the system and bypass security and integrity constraints.