8.1 Database Concept Notes

Engr M Kashif
O/A level Computer Science

Compiled By Engr M Kashif

Objective:

- Show understanding of **limitations** of using file based approach for data storage and retrieval.
- Describe **features** of **relational database** that address **limitations** of file-based approach.
- Understand and use terminology associated with relational database model Including entity, table, record, field, tuple, attribute, primary key, candidate key, secondary key, foreign key, relationship (one-to-many, one-to-one, many-to many), referential integrity, indexing.
- Use an entity-relationship (E-R) diagram to document a database design.
- Show understanding of normalisation process. 1NF, 2NF and 3NF.
- Explain why given set of database tables are, or are not, in 3NF.
- Produce normalised database design for description of database, a given set of data, or a given set of tables.

What is Database?

Database is **structured** collection of **Related Data** stored in an **Efficient** and **Compact Manner** that can be accessed by different applications programs.

Word "**Efficient**" means that stored data can be **accessed** very easily and quickly. Word "**Compact**" means that stored data takes up as **little space** as possible.

EMK

"**Related Data**" means that database contains data about **particular topic** such as:

- Database of employees that contains data of employees of an organization.
- Database of students that contains data of students of college/university etc.

- **Entity**: It is anything that can have data stored about it ,e.g a person, place, event.
- **Table**: It is a group of **similar data**, in a database, with rows and columns. Entity is implemented as table in relational database. For each entity there is one table created.
- **Record**: Row in a table in a database is known as **record**. It is also known as **Tuple** or an **instance**. It is one instance of an entity, which is represented by a row in a table.
- **Field**: A column in a table in a database is known as **field** or **attribute**. Example STUDENT entity may have attributes like Roll Number, Name, Grade, Address, age etc.

When there is only a single table in database, this is called a **'Flat File Database'.**

File Based Approach

In file base approach, data is stored in one or more flat files.
Explanation: One way to keep information on computer is to store it in files. Company has application programs, which is designed to manipulate data files.
Organization data was **duplicated** in separate files for use of individual departments.
Example:
HR department would hold details on name, address, qualifications etc. of each employee, while payroll department would hold details of name, address and salary of each employee.

Each department had its own set of application programs to process data in these files.

Limitation of File-Based Approach

- Data can be altered by one application and not by another; it then becomes **inconsistent** and it leads to **Data Integrity**.
- In computer File-Based System, data can be **duplicated**. Duplication is wasteful as it **Costs Time** and **Money**. Data has to be entered more than once, therefore it takes up time and more **Storage Space**. This lead to **Data Redundancy**.
- **Enquiries** available can depend on **structure of data** and **software** used, so data is not independent. **Data formats** are defined in application programs. If there is a need to change any of these formats, whole programs have to be changed. Different applications may hold data in different forms, again causing a problem which leads to **Data Dependency**.
- Computer file-based processing system do not provide **Data Privacy**. Data privacy would be properly handled by a database system.
- In file based system, it is **difficult** to **update records**. Same information is stored at different places so never sure that everywhere changes have been made.
- Computer file-based system do not provide proper **security** system against **illegal** access of data. Anyone can easily change or delete data stored in file. This lead to no **Data Security**.

Advantage of Relational Database

Problems that occurred using file-based approach have been solved by database.

- Storage space is not wasted as data items are only stored once, meaning little or **no Redundant** data.
- Data altered in one application is available in another application, so data is **Consistent**.
- **Enquiries** available are not dependent on **structure of data** and **software** used, so data is **independent**.
- Complex **queries** are easier to run.
- Reduces **program-data dependency** because data is separate from software so changes to the data do not require programs to be re-written.

EMK

Relational Database

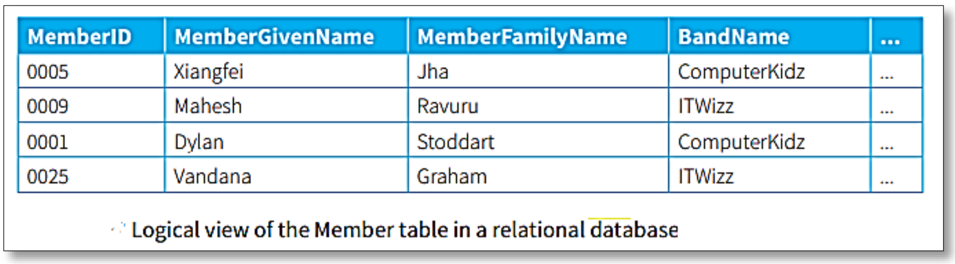
Relational Database is a collection of relational tables. In relational database model, each item of data is stored in a **relation** which is a special type of **table**.

Fundamental principle of a relational database is that a tuple is a set of **'atomic'** values; each attribute has **one** value or **no** value.

Example: Design for a database for **Theatrical Agency** might contain table definitions as:
Member (MemberID, MemberGivenName, MemberFamilyName, BandName, ...)
Band (BandName, AgentID, ...)

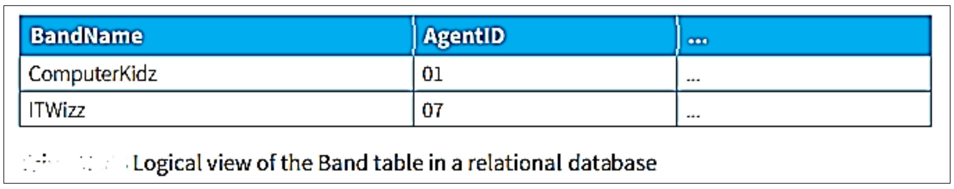
Relational Database is a database in which the data items are linked by internal Pointers.

8.1 Database Concept Notes



MemberID	MemberGivenName	MemberFamilyName	BandName	...
0005	Xiangfei	Jha	ComputerKidz	...
0009	Mahesh	Ravuru	ITWizz	...
0001	Dylan	Stoddart	ComputerKidz	...
0025	Vandana	Graham	ITWizz	...

Logical view of the Member table in a relational database



BandName	AgentID	...
ComputerKidz	01	...
ITWizz	07	...

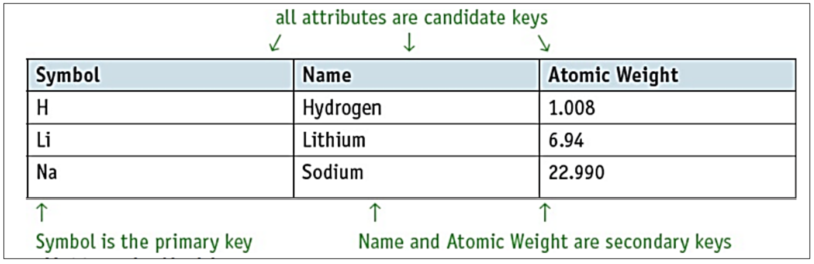
Logical view of the Band table in a relational database

Important feature of **Relational Database** concept is **Primary Key**.

- **Primary Key** may be a **single attribute** or a **combination** of attributes. If primary key consist of more than one field then it is called **composite primary key**.
  Every table must have a primary key and each **tuple** in table must have a value for primary key and that value must be **unique**.

Note: Primary key is underlined in database design. Primary key ensures **integrity** within table. DBMS will not allow to insert a value for a primary key when that value already exists. Therefore, each tuple automatically becomes **unique**.

- **Candidate Key:** In some cases there may be more than one attribute for which **unique** values are guaranteed. In this case, each one is a candidate key and one will be selected as **primary key**.
- **Secondary Key:** A candidate key that is not selected as primary key is then referred to as a secondary key.
- **Foreign key:** An attribute in one table that refers to primary key in another table.



all attributes are candidate keys

Symbol	Name	Atomic Weight
H	Hydrogen	1.008
Li	Lithium	6.94
Na	Sodium	22.990

Symbol is the primary key

Name and Atomic Weight are secondary keys

Referential Integrity is a concept in **relational database** design that ensures **consistency** and **accuracy** of relationships between tables. Use of a **foreign key** to ensure that a value can only be entered in one table when same value already exists in referenced table.

Explain reasons why Referential Integrity is important in a database. 9618/M/J/23

Referential Integrity makes sure data is **consistent.**

Referential Integrity makes sure all data is **up-to-date**.

Referential integrity ensures that every foreign key has a **corresponding** primary key.

Referential Integrity **prevents** records from being added/deleted/modified incorrectly.

Referential Integrity makes sure that if data is changed in one place, change is reflected in all related records.

Referential Integrity makes sure any queries return **accurate** and complete **results.**

Relationships

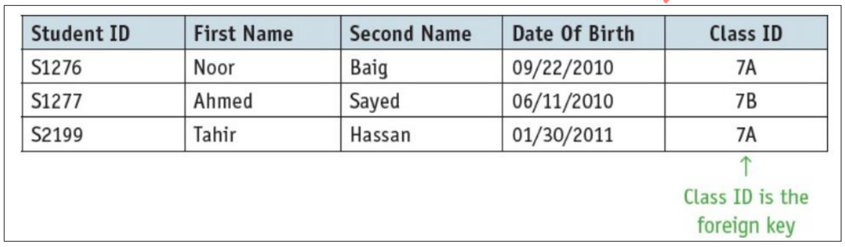
A **Relationship** is formed when one table in a database has a **Foreign Key** that refers to a **Primary Key** in another table in database.

In order to ensure **Referential Integrity** database must not contain any values of a **foreign** key that are not matched to corresponding **primary key**.

EMK

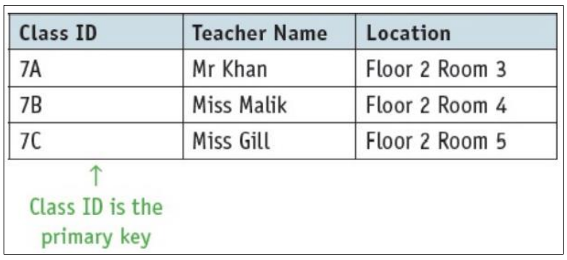
Example:

School Database contain table **Student** and table **Class** that contains Class ID, Teacher Name and Location of classroom. Only values for Class ID that are stored in **Class Table** can be used as foreign key in **Student table.**



Student ID	First Name	Second Name	Date Of Birth	Class ID
S1276	Noor	Baig	09/22/2010	7A
S1277	Ahmed	Sayed	06/11/2010	7B
S2199	Tahir	Hassan	01/30/2011	7A

Class ID is the foreign key



Class ID	Teacher Name	Location
7A	Mr Khan	Floor 2 Room 3
7B	Miss Malik	Floor 2 Room 4
7C	Miss Gill	Floor 2 Room 5

Class ID is the primary key

Relationships Types

1. **One-to-One Relationship:**

In a one-to-one relationship, each record in one table is associated with exactly one record in another table, and vice versa. This type of relationship is less common than other types, such as one-to-many or many-to-many relationships.

Example:

Country - capital city: Each country has exactly one capital city. Each capital city is capital of exactly one country.

2. **One-to-Many or Many-to-One Relationship**

In one-to-many relationship, each record in one table can be associated with one or more records in another table. This is common types of relationships in database

Example: Relationship between Student and Class is **Many-To-One**, as one value of attribute Class ID may appear many times in **Student Table** but only once in **Class Table**.

3. **Many-to-Many Relationship**

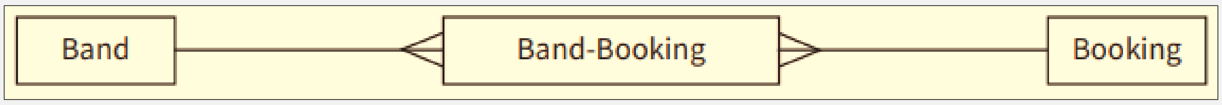
A many-to-many relationship is a type of relationship between two tables in a relational database where one record in first table can be associated with multiple records in second table, and vice versa.

Example: Consider a database that stores information about students and courses they are enrolled in. A student can be enrolled in multiple courses, and a course can have multiple students enrolled in it.

How to implement many-to-many relationship?

To implement a many-to-many relationship in a relational database, a **junction table** is used. A **junction table (**linking table**)** is a table that links two tables by having fields which are primary key of other two tables. For instance, in student-course example, a junction table would contain student ID and course ID, linking two tables together.

Problem relationship is M:M, where a foreign key cannot be used. A foreign key attribute can only have a single value, so it cannot handle many references required. **Solution** for M:M relationship is to create a link entity. For Band and Booking, logical entity model will contain link entity like this.



In order to **Speed up Searching** for data, an **index** can be used. Index is a **data structure** built from one or more columns in a database table. Student table could be indexed on Class, Second Name and First Name to provide class lists in alphabetical order of Second Name.

Entity–Relationship Modelling

An **E-R** diagram can be used to document **design** of database. This provides understandable **visual representation** of how **entities** in a database are related.

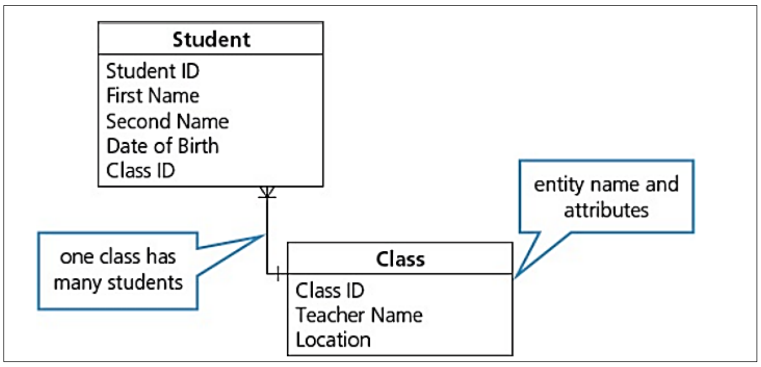
- **Optional Relationship:**

Relationships may be optional. For example, in a workroom with desks, each employee has one desk, but there could be spare desks. Relationship between desk and employee is **zero or one**, so this relationship is optional.

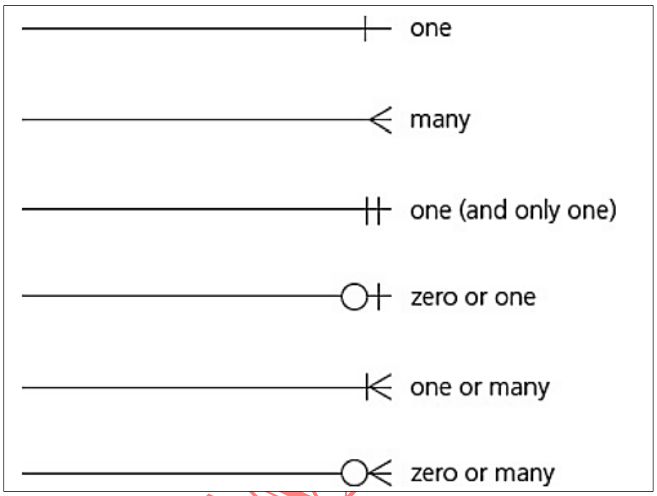
- **Mandatory Relationship:**

Relationship between mother and a child is **mandatory** because every mother must have at least one child, so relationship is one or many.

Note: Type of relationship and whether it is mandatory or optional gives **cardinality** of relationship. **Cardinality** determines how many records relate to each other.



E-R Diagram for School Database

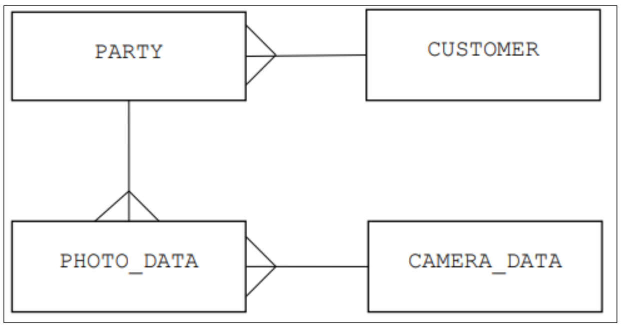


Example: A photographer creates a relational database to store data about photographs taken at birthday parties. **P11 Nov 22**
Database, PHOTOGRAPHS, stores details of customer, the party, the photographs taken and the cameras used.
Photographer has several cameras that are used for taking photographs at parties.
Each camera has a specific lens type (for example, XY32Z) and lighting type (example, F1672).
Data about each photograph is stored in database including party at which it was taken, the time it was taken and camera used. Database has these four tables:

```
CUSTOMER(CustomerID, FirstName, LastName, Telephone)

PARTY(PartyID, CustomerID, PartyDate, StartTime)

PHOTO_DATA(PhotoID, PartyID, TimeTaken, CameraID)

CAMERA_DATA(CameraID, LensType, LightingType)
```

Entity-relationship (E-R) diagram for database PHOTOGRAPHS is,

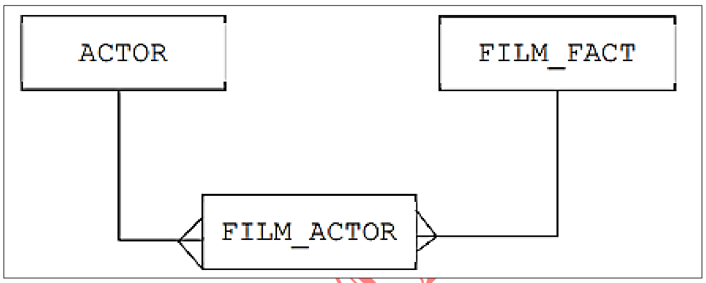
- 1:M between CUSTOMER and PARTY
- 1:M between PARTY and PHOTO_DATA
- 1:M between CAMERA_DATA and PHOTO_DATA



Example: Database, **FILMS**, stores information about films and actors. Part of database is shown.

```
ACTOR(ActorID, FirstName, LastName, DateOfBirth)
FILM_FACT(FilmID, FilmTitle, ReleaseDate, Category)
FILM_ACTOR(ActorID, FilmID)
```

E-R diagram will be look like this. **9618/12/M/J/22**



A **Composite Primary** key consists of two or more attributes that together form the primary key. Table **FILM_ACTOR** has a composite primary key because neither key uniquely identifies each tuple by itself and One actor cannot appear in same film twice so together, they are unique.

ESQ: Relational database, TECHNOLOGY, stores data about staff in company and computer devices used by staff. Database has following tables: **9618/13/M/J/22**

```
STAFF(StaffID, FirstName, LastName, DateOfBirth, JobTitle)

DEVICE(DeviceID, Type, DatePurchased, StaffID)
```

Describe relationship between two tables.

Ans: **Primary** key **StaffID** in STAFF links to **foreign** key StaffID in DEVICE. One staff member can have many devices. Each device can only be with one member of staff. (One to Many Relationship).

Normalization

Normalization is used to construct **Relational Database** that has **integrity** and in which **Data Redundancy** is reduced.

Tables that are not normalized will be **larger**. As more data is stored, it will be harder to **update** database when changes are made and more **difficult** to extract required data to answer **queries**.

Normal Forms

Normalization Process is carried out through a set of guidelines known as **normal forms.** Each normal form represents a level of organization, and higher the normal form, the more **refined** and **normalized** the database is.

First Normal Form

To achieve First Normal Form (1NF) in a database, following rules must be met for each table:

- **No Repeating Groups or repeating attributes:** Eliminate columns with repeated or similar data. Avoid having repeating groups within a table.
- **Atomic Values:** Ensure that each data item cannot be further divided (atomic values). Avoid using multi-valued or composite attributes.

  Examples of Atomic Data: • NY344599 • ISBN # e.g.1-931841-62-4 • First name, 'John' • A surname, 'Hunt' • Telephone number • Book name: 'CS MADE EASY ' • Complete description ”A fountain pen is a writing instrument’’.
- **Unique Rows:** Ensure that each row is unique by having a **primary key**. This means there are no duplicate rows in table.
- **Unique Field Names:** Provide each field with a **unique name** within table to avoid ambiguity and ensure clarity in **data representation**.

MK

Repeating groups can be categorized into two types:

Repeating Similar data: This occurs when same type of data is repeated across multiple fields within a single record. This violates principles of 1NF because same kind of information should not be stored in separate fields.

Example, having fields like Phone1, Phone2, in a table where each field stores phone number.

Repeating groups of Attributes: This occurs when a group of attributes is repeated for different instances within a single record. It violates principles of 1NF because each attribute should have a single occurrence in a well-normalized table.

Example: In Student table, if the RollNo, FirstName, LastName, and DOB are repeated for each subject entry, it represents a repeating group of attributes.

Second Normal Form (2 NF)

Rule for Second Normal Form (2NF) are as follows:

- **No Partial Dependencies:** Ensure that **non-key attributes** are fully dependent on every part of primary key. There should be **no partial dependencies**, where only a part of the primary key determines the value of certain non-key fields. In simple word, All attributes must be fully dependent on (composite) primary key // No partial dependencies.

- **Already in First Normal Form (1NF):** Table must already satisfy requirements of First Normal Form. This includes having atomic values, no repeating groups, unique rows (with a primary key), and unique field names.

Note: Non-key fields or **Non-key attributes** refer to the attributes (or columns) in a table that are not part of primary key. These attributes provide additional details about entity being represented but are not used for unique identification.

A **key field** is a field (or set of fields) that uniquely identifies a record in a table. In a relational database, a primary key is a specific type of key that uniquely identifies each record.

Example: Table: StudentSubjectChoices .

<u>StudentName</u>	<u>Subject</u>	Level	SubjectTeacher
Tom	Physics	A	SAN
Tom	Chemistry	A	MEB
Tom	Gen Studies	AS	DIL
Joe	Geography	AS	ROG
Joe	French	AS	HEN
Samir	Computing	A	VAR
Samir	Chemistry	A	MEB

StudentName + Subject is **composite primary key** and student have unique names.

EMK

Why above table is Not in 2 NF?

SubjectTeacher (non-key attribute) dependent only on part of primary key i.e. **Subject** so it is partial dependency. **SubjectTeacher** is not dependent on **StudentName**.

For one student (e.g. Tom we have more than one values of subject teacher (i.e. SAN, MEB, DIL)).

Attribute **Level** is fully dependent on both part of Primary Key – means value of **Level** can only be obtained using both fields of PK.

Third Normal Form (3 NF)

Rules for Third Normal Form (3NF) are as follows:

- **Already in 2NF:** Table must already satisfy requirements of Second Normal Form (2NF). This includes having **no partial dependencies**, meaning that non-key attributes are fully dependent on every part of primary key.
- **No Transitive Dependencies**: Ensure that there are **no non-key attributes** that depend on another **non-key attribute**. In other words, eliminate transitive dependencies, where a **non-key attribute** depends on another **non-key attribute**, which itself depends on primary key.

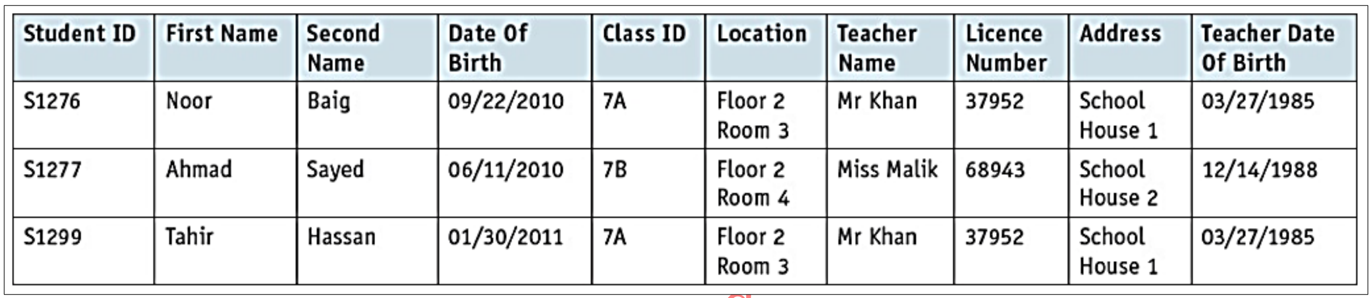
Example: Staff (StaffID, StaffName, City, Country)

StaffID	StaffName	City	Country
56	ALI	Lahore	Pakistan
78	Ahmed	Islamabad	Pakistan
52	Jon	New York	USA

Above table is Not in 3 NF because Country (non-key attribute) depends upon City (non-key attribute). If we know the value of City we can find the value of Country.

Normalisation Process

If **School Database** is held in a single table it could look like this:



Student ID	First Name	Second Name	Date Of Birth	Class ID	Location	Teacher Name	Licence Number	Address	Teacher Date Of Birth
S1276	Noor	Baig	09/22/2010	7A	Floor 2 Room 3	Mr Khan	37952	School House 1	03/27/1985
S1277	Ahmad	Sayed	06/11/2010	7B	Floor 2 Room 4	Miss Malik	68943	School House 2	12/14/1988
S1299	Tahir	Hassan	01/30/2011	7A	Floor 2 Room 3	Mr Khan	37952	School House 1	03/27/1985

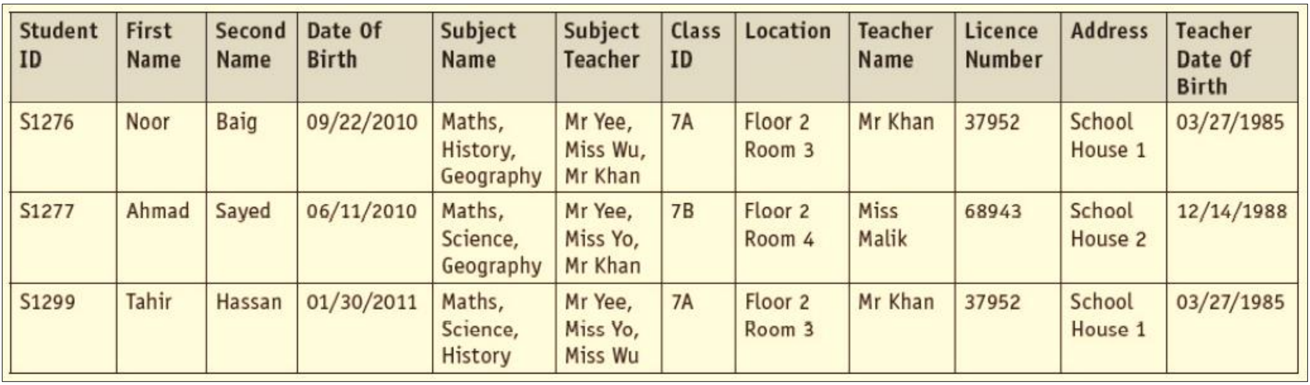
Problem With School Database:

- Every time a new student is added, teacher's name, address, licence number, date of birth, and location of classroom need to be added as well.
- If Mr Khan leaves school and is replaced by another teacher, then every record containing his name and other details needs to be changed.
- If all students from Class 7B leave, then all the details about Class 7B will be lost.

EMK

Worked Example

School database to be normalised, the process is:



Student ID	First Name	Second Name	Date Of Birth	Subject Name	Subject Teacher	Class ID	Location	Teacher Name	Licence Number	Address	Teacher Date Of Birth
S1276	Noor	Baig	09/22/2010	Maths, History, Geography	Mr Yee, Miss Wu, Mr Khan	7A	Floor 2 Room 3	Mr Khan	37952	School House 1	03/27/1985
S1277	Ahmad	Sayed	06/11/2010	Maths, Science, Geography	Mr Yee, Miss Yo, Mr Khan	7B	Floor 2 Room 4	Miss Malik	68943	School House 2	12/14/1988
S1299	Tahir	Hassan	01/30/2011	Maths, Science, History	Mr Yee, Miss Yo, Miss Wu	7A	Floor 2 Room 3	Mr Khan	37952	School House 1	03/27/1985

First Normal Form

Un-Normalised School database can be represented as follows.

STUDENT(<u>StudentID,</u> FirstName, SecondName, DateOfBirth, SubjectName, SubjectTeacher, SubjectName, SubjectTeacher, SubjectName, SubjectTeacher, ClassID, Location, TeacherName, LicenceNumber, Address, TeacherDateOfBirth).

STUDENT is table name; attributes are listed in order and **primary key** is underlined.

Note: student's subjects and **subject teacher** are **repeating attributes**. For database to be in first normal form, these need to be removed to a separate table and linked to original table with a **foreign key**.

School database can now be represented in 1NF as follows.

STUDENT (StudentID, FirstName, SecondName, DateOfBirth, ClassID, Location, TeacherName, LicenceNumber, Address, TeacherDateOfBirth).

STUDENTSUBJECT (StudentID, SubjectName, SubjectTeacher)

Now table will look like this : **Student table**

Student ID	First Name	Second Name	Date Of Birth	Class ID	Location	Teacher Name	Licence Number	Address	Teacher Date Of Birth
S1276	Noor	Baig	09/22/2010	7A	Floor 2 Room 3	Mr Khan	37952	School House 1	03/27/1985
S1277	Ahmad	Sayed	06/11/2010	7B	Floor 2 Room 4	Miss Malik	68943	School House 2	12/14/1988
S1299	Tahir	Hassan	01/30/2011	7A	Floor 2 Room 3	Mr Khan	37952	School House 1	03/27/1985

STUDENTSUBJECT Table

Student ID	Subject Name	Subject Teacher
S1276	Maths	Mr Yee
S1276	History	Miss Wu
S1276	Geography	Mr Khan
S1277	Maths	Mr Yee
S1277	Science	Miss Yo
S1277	Geography	Mr Khan
S1299	Maths	Mr Yee
S1299	Science	Miss Yo
S1299	History	Miss Wu

Primary key for **STUDENTSUBJECT** table can be a **composite key** formed from two attributes **StudentID** and **SubjectName**; attribute StudentID is also a **foreign key** that links to **STUDENT** table.

Second Normal form (2NF)

There are now **two** tables; in STUDENTSUBJECT table, primary key is a composite key and **SubjectTeacher** is only dependent on **SubjectName** part of primary key. This is a **partial dependence** and needs to be removed by introducing a third table, **SUBJECT.**

School database can now be represented in **2NF** as follows.

STUDENT (StudentID, FirstName, SecondName, DateOfBirth, ClassID, Location, TeacherName, LicenceNumber, Address, TeacherDateOfBirth)

STUDENTSUBJECT (StudentID, SubjectName)

SUBJECT (SubjectName, SubjectTeacher)

Table will Look like;

Student ID	First Name	Second Name	Date Of Birth	Class ID	Location	Teacher Name	Licence Number	Address	Teacher Date Of Birth
S1276	Noor	Baig	09/22/2010	7A	Floor 2 Room 3	Mr Khan	37952	School House 1	03/27/1985
S1277	Ahmad	Sayed	06/11/2010	7B	Floor 2 Room 4	Miss Malik	68943	School House 2	12/14/1988
S1299	Tahir	Hassan	01/30/2011	7A	Floor 2 Room 3	Mr Khan	37952	School House 1	03/27/1985

Student ID	Subject Name
S1276	Maths
S1276	History
S1276	Geography
S1277	Maths
S1277	Science
S1277	Geography
S1299	Maths
S1299	Science
S1299	History

Subject Name	Subject Teacher
Maths	Mr Yee
History	Miss Wu
Geography	Mr Khan
Science	Miss Yo

School Database in 2Nf

Third Normal form (3NF)

There are now three tables.

In **STUDENT** table,

- Attributes **Location** and **TeacherName** depend upon attribute **ClassID**
- Attributes LicenceNumber, Address and TeacherDateOfBirth depend upon attribute TeacherName.

These are **Non-Key Dependencies** that need to be removed to ensure that database is in 3NF.

At this stage it is worth inspecting database and its contents to consider any other problems that could arise, such as the following:

- Teacher names might not be unique; therefore, it is better to use licence number as a primary key.
- Teachers can be both class teachers and subject teachers; these need to be combined in one table.

Improved School database can now be represented in 3NF as follows.

STUDENT(StudentID, FirstName, SecondName, DateOfBirth,)

CLASS(ClassID, Location, LicenceNumber)

TEACHER(LicenceNumber, TeacherName, Address, TeacherDateOfBirth)

STUDENTSUBJECT(StudentID, SubjectName)

SUBJECT(SubjectName, LicenceNumber).

EMK

Advantages of normalization Process:

- **Data consistency:** Each piece of information is stored in one place, which helps in data consistency by reducing redundancy and eliminating data anomalies.
- **Data integrity:** Normalization ensures data integrity by enforcing rules and constraints.
- **Efficient data storage:** by eliminating redundant information, it reduces the amount of disk space required and improves database performance.
- **Simplified updates:** Updates and modifications can be made more efficiently.

ESQ: Tick one box in each row to identify stage for each task. **9618/12/Nov 21**

Task	Normalisation stage		
	0NF to 1NF	1NF to 2NF	2NF to 3NF
Remove any partial key dependencies		✓	
Remove any repeating groups of attributes	✓		
Remove any non-key dependencies			✓

****