SQL Databases

Name: ___________________________ (   ) Class: _________ Date: _________

Lesson 1: Introduction to Databases

Instructional Objectives:

By the end of this task, you should be able to:

- Draw entity-relationship (ER) diagrams to show the relationship between tables.
- Determine the attributes of a database: table, record and field.
- Explain the purpose of and use primary, secondary, composite and foreign keys in tables.
- Explain with examples, the concept of data redundancy and data dependency.
- Reduce data redundancy to third normal form (3NF).

SQL Databases

What is a database?

A **database** is a collection of data stored in an organised or logical manner. Storing data in a database allows us to access and manage the data. Examples of databases in real-life include:

- Patient medical records
- Supermarket inventory
- Contact list

Relational database

In the relational database model, data is stored in relations and represented in the form of tuples (rows). A relational database is a collection of relational tables.

A table is a two-dimensional representation of data stored in rows and columns.

A table description can be expressed as:

```
TableName (Attribute1, Attribute2, Attribute3, …)
```

For example,

```
Student (RegNo, Name, Gender, CivicsClass, CivicsTutor)
```

A table is a relation if it fulfils the following conditions:

- Values are atomic
- Column values are of the same kind
- Rows are unique
- The order of columns is insignificant
- Each column must have a unique name

[https://www.cs.wcupa.edu/~zjiang/RDB_table.htm]

Below is an example of a table `Student18S12`, showing data for students from civics class 18S12. Field

RegNo	Name	Gender	MobileNo
1	Adam	M	92313291
2	Adrian	M	92585955
3	Agnes	F	83324112
4	Aisha	F	88851896
5	Ajay	M	94191061
6	Alex	M	98675171
7	Alice	F	95029176
8	Amy	F	98640883
9	Andrew	M	95172444
10	Andy	M	95888639

Record

SQL Databases

Tables are made up of records and fields.

A **record** is a complete set of data about a single item. For this example, a record refers to the complete set of data of a particular student.

> **Test your understanding**
> How many records does the table `Student18S12` have?

A **field** refers to one piece of data about a single item. For this example, the table has 4 fields – `RegNo`, `Name`, `Gender` and `MobileNo`.

Candidate keys

A **candidate key** is defined as a minimal set of fields which can uniquely identify each record in a table. It tells a particular record apart from another record. A candidate key should never be NULL or empty. There can be more than one candidate key for a table. A candidate key can also be a combination of more than one field.

> **Test your understanding**
> Which of the fields in table `Student18S12` are candidate keys?

Primary keys

A **primary key** is a candidate key that is most appropriate to become the main key for a table. It uniquely identifies each record in a table and should not change over time. That is, a primary key tells a particular record apart from another record.

> **Test your understanding**
> Which of the candidate key is a suitable primary key for the table `Student18S12`?

Secondary keys

Candidate keys that are not selected as primary key are known as secondary keys.

Composite keys

Sometimes, more than one field is needed to uniquely identify a record. A **composite key** is a combination of two or more fields in a table that can be used to uniquely identify each record in a table. Uniqueness is only guaranteed when the fields are combined. When taken individually, the fields do not guarantee uniqueness.

Take a look at the table `Student` on the next page.

SQL Databases

RegNo	Name	Gender	CivicsClass
1	Adam	M	18S12
2	Adrian	M	18S12
3	Agnes	F	18S12
4	Aisha	F	18S12
5	Ajay	M	18S12
6	Alex	M	18S12
7	Alice	F	18S12
8	Amy	F	18S12
9	Andrew	M	18S12
10	Andy	M	18S12
1	Adam	M	18A10
2	Bala	M	18A10
3	Bee Lay	F	18A10
4	Ben	M	18A10
5	Boon Kiat	M	18A10
6	Boon Lim	M	18A10
7	Charles	M	18A10
8	Chee Seng	M	18A10
9	Cher Leng	F	18A10
10	Choo Tuan	M	18A10

CivicsClass 18S12

CivicsClass 18A10

Now, a single field is not able to uniquely identify a record. A composite key composing 2 fields is needed uniquely identify a record.

Test your understanding
Which 2 fields would you choose to form a composite key for the above table `Student`?

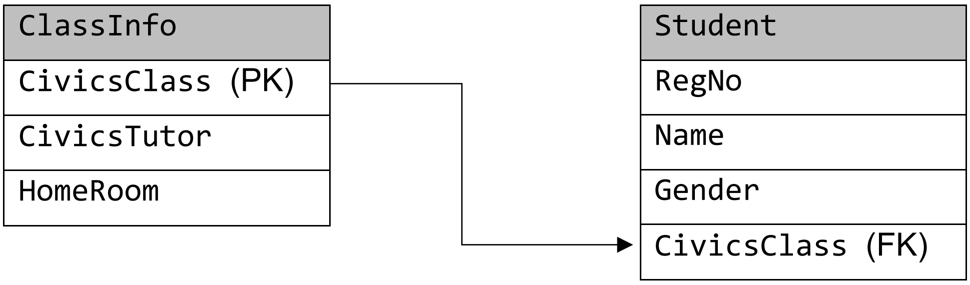
Foreign keys

Another table, `ClassInfo`, as shown below, stores the information about each civics class.

CivicsClass	CivicsTutor	HomeRoom
18S12	Peter Lim	BlkA-L2-3
18A10	Pauline Lee	BlkC-L1-4

`CivicsClass` is chosen to be the primary key (PK) for table `ClassInfo`.

Notice that the primary key in table `ClassInfo` (i.e., the `CivicsClass` field) is related or linked to the `CivicsClass` field in table `Student`. This makes `CivicsClass` field in table `Student` a foreign key (FK).



A **foreign key** is an attribute (field) in one table that refers to the primary key in another table.

Data redundancy

Data redundancy refers to the same data being stored more than once.

Below is an example of `Student` table.

RegNo	Name	Gender	CivicsClass	CivicsTutor
1	Adam	M	18S12	Peter Lim
2	Adrian	M	18S12	Peter Lim
3	Agnes	F	18S12	Peter Lim
4	Aisha	F	18S12	Peter Lim
5	Ajay	M	18S12	Peter Lim
6	Alex	M	18S12	Peter Lim
7	Alice	F	18S12	Peter Lim
8	Amy	F	18S12	Peter Lim
9	Andrew	M	18S12	Peter Lim
10	Andy	M	18S12	Peter Lim

As we can see, data for `CivicsClass` and `CivicsTutor` are repeated for students who are in the same civics class. This is data redundancy.

This will cause issues when inserting, updating and deleting data from the database. See the table below for examples.

Inserting data	A new student cannot be inserted unless a civics class and a civics tutor have been assigned.
Updating data	If Peter Lim decide to leave the college, all the records in the `Student` table would need to be updated. If we miss any record, it will lead to inconsistent data.
Deleting data	If all the records of the `Student` table are deleted, information on civics class and civics tutor will be lost.

Data dependency

In order to normalise data, we need to understand data dependencies, specifically functional and transitive dependencies.

SQL Databases

Functional dependency

Attribute Y is **functionally dependent** on attribute X (usually the primary key), if for every valid instance of X, the value of X uniquely determines the value of Y (X → Y).

For example, suppose we have a `Student` table with the following attributes: `MatricNo`, `Name`, `Gender`, `CivicsClass` and `CivicsTutor`. `MatricNo` is a unique number assigned to every student in the college.

`MatricNo` uniquely identifies the `Name` because if we know the `MatricNo`, we can know the `Name` associated with it. Therefore, we can say `Name` is functionally dependent on `MatricNo` (`MatricNo` → `Name`).

Transistive dependency

A functional dependency is said to be **transitive** if it is indirectly formed by two functional dependencies. Z is transitively dependent on X if Y is functionally dependent on X but X is not functionally dependent on Y, and Z is functionally dependent on Y. In other words, X → Z is a transitive dependency if the following hold true:

- X → Y
- Y does not → X
- Y → Z

For example, `CivicsClass` is functionally dependent on `MatricNo` (`MatricNo` → `CivicsClass`) but `MatricNo` is not functionally dependent on `CivicsClass`. On the other hand, `CivicsTutor` is functionally dependent on `CivicsClass` (`CivicsClass` → `CivicsTutor`). Therefore, `CivicsTutor` is transitively dependent on `MatricNo` (`MatricNo` → `CivicsTutor`).

Normalisation

Normalisation is the process of organising the tables in a database to reduce data redundancy and prevent inconsistent data. There are at least three normal forms associated with normalisation: first normal form (1NF), second normal form (2NF), and third normal form (3NF).

First Normal Form

For a table to be in 1NF, all columns must be atomic. This means there can be no multi-valued columns – i.e., columns that would hold a collection such as an array or another table. In another words, the information in each column cannot be broken down further.

Consider the following table:

MatricNo	Name	Gender	CivicsClass	CivicsTutor	HomeRoom	CCAInfo
1	Adam	M	18S12	Peter Lim	TR1	Tennis Teacher IC = Adrian Tan
2	Adrian	M	18S12	Peter Lim	TR1	Choir Teacher IC = Adeline Wong, Student Council Teacher IC = David Leong
3	Adam	M	18A10	Pauline Lee	TR2	Rugby Teacher IC = Andrew Quah
4	Bala	M	18A10	Pauline Lee	TR2	Badminton Teacher IC = Lilian Lim
5	Bee Lay	F	18A10	Pauline Lee	TR2	Choir Teacher IC = Adeline Wong, Chess Club Teacher IC = Edison Poh

It is assumed that every civics class only has one civics tutor, and each CCA has only one teacher in-charge (IC).

This table is not in 1NF because the `CCAInfo` column contains multiple values.

In order for the table to be in 1NF, we split `CCAInfo` into two single-value columns `CCAName` and `CCATeacherIC`. Notice that the students with `MatricNo` 2 and `MatricNo` 5 have multiple CCAs. We keep this information intact by splitting their records into multiple records, each corresponding to a different CCA. The resulting table is shown below.

MatricNo	Name	Gender	CivicsClass	CivicsTutor	HomeRoom	CCAName	CCATeacherIC
1	Adam	M	18S12	Peter Lim	TR1	Tennis	Adrian Tan
2	Adrian	M	18S12	Peter Lim	TR1	Choir	Adeline Wong
2	Adrian	M	18S12	Peter Lim	TR1	Student Council	David Leong
3	Adam	M	18A10	Pauline Lee	TR2	Rugby	Andrew Quah
4	Bala	M	18A10	Pauline Lee	TR2	Badminton	Lilian Lim
5	Bee Lay	F	18A10	Pauline Lee	TR2	Choir	Adeline Wong
5	Bee Lay	F	18A10	Pauline Lee	TR2	Chess Club	Edison Poh

The values for `CCAName` and `CCATeacherIC` are now **atomic** for each row. The primary key for the above table will be the composite key formed by `MatricNo` and `CCAName`.

Second Normal Form

For a table to be in 2NF, it must satisfy two conditions:

1) The table should be in 1NF
2) Every non-key attribute must be fully dependent on the entire primary key. This means no attribute can depend on part of the primary key only.

`Name`, `Gender`, `CivicsClass`, `CivicsTutor` and `HomeRoom` is dependent on only part of the primary key, `MatricNo`. `CCATeacherIC` is dependent only on `CCAName`. Thus, we decompose into three tables as shown below.

Student

MatricNo	Name	Gender	CivicsClass	CivicsTutor	HomeRoom
1	Adam	M	18S12	Peter Lim	TR1
2	Adrian	M	18S12	Peter Lim	TR1
3	Adam	M	18A10	Pauline Lee	TR2
4	Bala	M	18A10	Pauline Lee	TR2
5	Bee Lay	F	18A10	Pauline Lee	TR2

StudentCCA

MatricNo	CCAName
1	Tennis
2	Choir
2	Student Council
3	Rugby
4	Badminton
5	Choir
5	Chess Club

CCAInfo

CCAName	CCATeacherIC
Tennis	Adrian Tan
Choir	Adeline Wong
Student Council	David Leong
Rugby	Andrew Quah
Badminton	Lilian Lim
Chess Club	Edison Poh

The primary key for table `Student` will be `MatricNo`. `MatricNo` and `CCAName` will form a composite key for table `StudentCCA` and `CCAName` will be the primary key for table `CCAInfo`.

Third Normal Form

For a table to be in 3NF, it must satisfy two conditions:

1) The table should be in 2NF
2) The table should not have transitive dependencies

Looking at table `Student`, `CivicsTutor` and `HomeRoom` are dependent on `CivicsClass` and `CivicsClass` is dependent on `MatricNo`. Therefore, `CivicsTutor` and `HomeRoom` are transitively dependent on `MatricNo`. To remove the transitive dependency, we decompose the `Student` table as follows:

Student

MatricNo	Name	Gender	CivicsClass
1	Adam	M	18S12
2	Adrian	M	18S12
3	Adam	M	18A10
4	Bala	M	18A10
5	Bee Lay	F	18A10

Civics

CivicsClass	CivicsTutor	HomeRoom
18S12	Peter Lim	TR1
18A10	Pauline Lee	TR2

MatricNo remains the primary key for table Student and CivicsClass will be the primary key of the newly formed table Civics.

The final design after normalisation is represented below:
Student (MatricNo, Name, Gender, CivicsClass)

Civics (CivicsClass, CivicsTutor, HomeRoom)

StudentCCA (MatricNo, CCAName)

CCAInfo (CCAName, CCATeacherIC)

The primary key for each table is indicated by underlining one or more attributes. Foreign keys are indicated by using a dashed underline.

This design could also be represented using an Entity-Relationship diagram, also known as an E-R diagram.

Entity-Relationship diagram

Designers of a large database will normally construct a diagram of the planned database. An example of this is entity-relationship (E-R) diagram.

Note: For the purposes of this syllabus, we will only cover a simplified convention for drawing E-R diagrams.

An **entity** is a specific object of interest. Collective nouns or nouns are usually used to name entities. Entities are represented by rectangles.

For example, Student

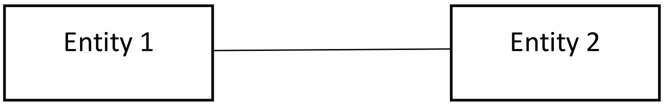
Relationship describes a link between two entities.

SQL Databases

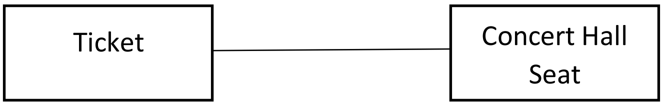
One of the following three relationships can exist between two entities.

1. One to One

   This can be represented by:

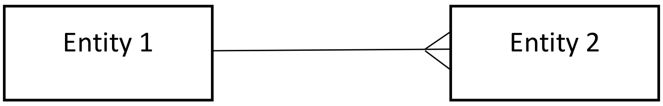


   For example, at a concert each ticket entitles you to a particular seat and each seat is linked to only one ticket.

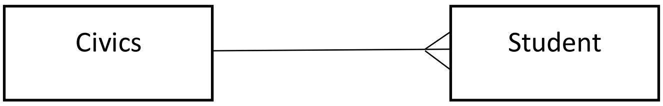


2. One to Many

   This can be represented by:

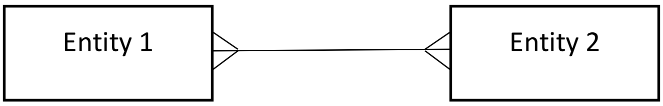


   For example, a student can belong to only one civics class, but a civics class can have many students.

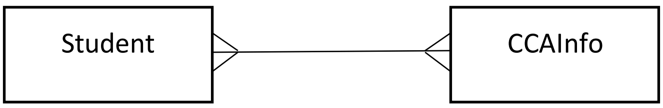


3. Many to Many

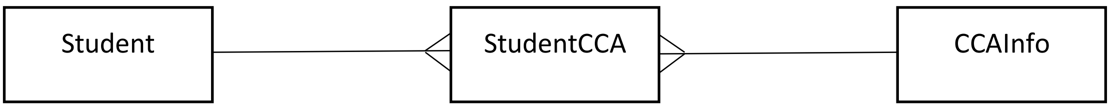
   This can be represented by:



   For example, a student can join many CCAs and one CCA can have many students.



   To implement a many-to-many relationship in a database system, we will usually decompose a many-to-many relationship into two (or more) one-to-many relationships. For example,



Representing the normalised tables,

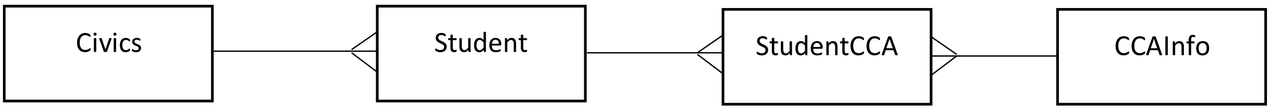
Student (MatricNo, Name, Gender, CivicsClass)

Civics (CivicsClass, CivicsTutor, HomeRoom)

StudentCCA (MatricNo, CCAName)

CCAInfo (CCAName, CCATeacherIC)

using an E-R diagram, we will have the following:



References

1. Burdett, A. (2016). *BCS glossary of computing*. BCS Learning and Development Limited.
2. Captain, F. A. (2015). *Six-step relational database design: A step by step approach to relational database design and development*. Place of publication not identified: Publisher not identified.
3. LANGFIELD, S. D. (2015). *Cambridge International AS and A Level Computer Science Coursebook*. Place of publication not identified: Cambridge University Press.
4. Archiveddocs. (n.d.). Lesson 2: Designing a Normalized Database. Retrieved from https://docs.microsoft.com/en-us/previous-versions/tn-archive/cc505842(v=technet.10)
5. Introduction to Database Keys. (n.d.). Retrieved from https://www.studytonight.com/dbms/database-key.php
6. Watt, A. (2014, October 24). Database Design - 2nd Edition. Retrieved from https://opentextbc.ca/dbdesign01/
7. Singh, C., & Prasad. (2018, December 14). Functional dependency in DBMS. Retrieved from https://beginnersbook.com/2015/04/functional-dependency-in-dbms/
8. Chapter 6. Entity-Relationship Modelling. (n.d.). Retrieved from https://www.cs.uct.ac.za/mit_notes/database/htmls/chp06.html#one-to-one-relationships-between-two-entities