

Module 1: Introduction to databases



Welcome to Week 1 of Database Fundamentals. This week, we will introduce databases – specifically relational databases. We will look into the concepts and characteristics of a database and a relational database. We will also have a look at the basic modelling approach of relational databases using entity-relationship modelling. Entity-relationship modelling is the standard modelling technique used by database practitioners worldwide to model a particular database problem. The concepts you learn this week will provide you with a foundation for the subject and be critical as you complete your first assessment.

Outcomes

By the end of this week, having completed the readings and all activities, you will be able to:

Topic 1: Introduction to database technologies

- Understand what a database is
- Appreciate why we should use databases
- Understand why database systems are essential.

Topic 2: Relational database model

- Understand different components of relational databases
- Appreciate why we should learn about relational databases
- Understand essential components, i.e. entity, attribute of an ER diagram.



Estimated learning hours

You will need to spend around 10 hours in total studying and completing activities this week.



Drop-in sessions and practice classes

The online practice class and drop-in sessions allow you to interact with faculty, subject matter experts and your student peers in real-time. These sessions are offered throughout the subject and are an essential part of the learning activities.

Practice class (2 hours)

In this 2-hour synchronous session, you will know the importance of databases and their design and implementation. For this exercise, the class will be divided into groups. Each group will be allocated questions from the exercises below. One member from each group will then provide the group's answers to the rest of the class for general discussion.

Drop-in session (1 hour)

You can attend this drop-in session if you have any questions about the materials from topics 1–2. This drop-in session can also help you prepare for practice class 1, based on the materials from the topics.

We recommend that you attend the sessions. However, if you cannot do so, it is essential that you view and listen to the recording, if available, and undertake the activities. Recordings and session notes are typically available on this page after the final online session for the week has taken place.
































In advance of the session, please post any questions or concerns you may have about the weekly learning or assessments on to the subject forums (see 'Announcements and forums' page) so that we can discuss them in the synchronous session.



Below are links to the various topics in this week of learning. Click on Topic 1 to get started with the learning for this week. If you have already started this week of learning, you can resume your studies at any of the below topic links.

Arrows at the bottom of the topic pages will take you to the next page within a topic or the next topic. Otherwise, you can return to this weekly landing page or the home page of this subject by using the tiles on the navigation bar on the left of this site.

Good luck with your studies this week!

	Your progress 
 Topic 1: Introduction to database technologies	
 Why use a database?	
 Benefits of a database approach	
 Quiz: Check your understanding	
 Discussion: What is a database?	
 Topic 2: Relational database model	
 Why Oracle?	
 Relational model	
 Data structure	
 Relations and attributes	
 Data integrity	
 Preservation of referential integrity	
 Quiz: Check your understanding	
 Activity: Data integrity	
 Summary and checklist	



Topic 1: Introduction to database technologies

Welcome to the first topic of **Database Fundamentals**.

We will learn about the necessity of using a database approach when building real-world applications in this topic.

About Oracle Academy - Free Resources for Educators and Their Students [0:55 mins]

About Oracle Academy - Free Resources for Educators and Their Students



Oracle Academy Cloud explainer video [1:50 mins]

Oracle Academy Cloud explainer video



Recommended reading

Before we start, please note the recommended reading for this topic is the first two chapters (1 and 2) of [R. Elmasri and S.B. Navathe \(2017\) *Fundamentals of Database Systems*, 7th edn., Pearson](#). The information in these two chapters sets the tone for the significance of using a database. It also introduces you to a brief history of database technologies developed over the last few decades.



History of databases [05:29 mins]

History of Databases



What is a database?

A **database** is:

A database can be defined as a shared collection of interrelated data designed to meet the varied information needs of an organisation.

(McFadden and Hoffer)

There are two distinct but related terms used in database technology:

Click on the below headings to reveal further information.

Database

A database is:

- a representation of some aspect of the real world or, perhaps, a collection of data elements (facts) representing real-world information.
- logically coherent and internally consistent
- designed, built and populated with data for a specific purpose.

Database system

A database system is a software system for managing databases: definition and creation, population, querying.

A file system

A database system is not a file system or vice-versa. Please watch the short video below to understand the differences between them better.

A file system explanation video [06:00 mins]



0:00 / 6:01

- [A file system explanation video transcript](#)

Last modified: Wednesday, 17 August 2022, 8:10 PM



Why use a database?

Imagine owning a business, say a streaming service such as Netflix – What sort of data would you like to keep?

- Current collection (i.e. movies, series)
- Customer and subscription details
- Information about a title (i.e. release year, maturity category, duration, genre, story synopsis, etc.)
- Customer account details, direct debit information
- Relationship between titles in the same genre (so that if a user is interested in a title, you can recommend related titles).

What would be the best approach to store all that data in a structured manner? The answer: a database.

If you have such a database that stores data relevant to your streaming service, what sort of questions might you want the database to answer?

- What are the current English movies in the collection?
- How many of these are animations?
- What is the most-watched movie in Australia today?
- How many seasons does 'Friends' have?
- Which state of Australia has the lowest number of subscribers?

These are just example questions. Imagine the endless questions you might be asking to make sound business decisions if you are running a business. For example, the last question on subscribers might help you target specific areas for a new advertising campaign to promote the streaming service.

These questions can be translated into queries, written in a form that a database system can understand. A database enables the storage of data and assists the generation of information.

What makes a good database?

Businesses need to clearly understand what data they need to store (requirements). These requirements need to be clearly articulated and described by a model. A model is a simplified representation of reality. The requirements are analysed and represented using an entity-relationship diagram (ER diagram) for relational databases. We will learn about ER diagrams in more detail later. Designing the database using a formal approach, such as ER modelling, is critical to having a good database. That way, the captured data is clear and non-ambiguous.

Once the database is adequately modelled and implemented, the stored data should be correct and sensible. In relational databases, several constraints ensure the integrity of the data.

How does the database enable special requirements?

Different businesses might have different or specific requirements related to their data storage. Consider the following examples:

- A retail business such as a supermarket (e.g. Coles, Woolworths) needs sales transaction data stored in real-time and fast. Otherwise, they might end up with inconsistent information in the database.
- A database that stores hospital records, such as treatments available, patients' information, doctors' information, ward and bed availability, billing data, etc., needs to ensure that the privacy of these data is protected.
- Mass data storage, such as weather records with thousands of observation sites, many observations per day, many values per observation and many years of records, requires persistent and efficient storage.
- Social Media data, such as Facebook posts and Tweets, require real-time, persistent, and efficient storage since data can grow exponentially.

Databases provide a uniform framework for fulfilling these very different requirements.

Last modified: Wednesday, 17 August 2022, 8:10 PM



Benefits of a database approach

Databases provide several significant features that make data storage and organisation easy and efficient. Three of these benefits are discussed below, with examples.

Click on the below headings to reveal further information.

Control data redundancy

A database helps us control data redundancy. Data redundancy is the unnecessary repetition of the same data in the database. Data redundancy leads to three major problems:

- duplicate effort
- storage waste
- inconsistency of data.

Consider the table below:

A table with redundant data

Student Number	Student Name	Subject Code	Grade
11	John Smith	CSE1PE	C
11	John Smith	CSE2DBF	A
11	John Smith	CSE10OF	B
12	James Brown	CSE2DBF	B
12	James Brown	CSE10OF	A

In the table above, we have inserted the same information multiple times. Such as the name John Smith which appears three times in the table, and James Brown, which is repeated to give two entries. This can also lead to wastage of storage.

If James Brown suddenly decides to change their last name to Orange, and we mistakenly update only one row, we might end up with something like this: An example of inconsistent data below.


An example of inconsistent data

Student Number	Student Name	Subject Code	Grade
12	James Brown	CSE2DBF	B
12	James Orange	CSE10OF	A

As you can see, the information in the above table is inconsistent. One row says student number 12 is named 'James Brown' and the next row conveys different information.

A Database provides us with relationship features through keys, where you can avoid such unnecessary redundancies.

Using these features will assist you to control the redundancy and ensure there are no duplicates or inconsistent information in your database.



Maintain data integrity and control access

Data Integrity: Most database applications have certain Integrity constraints that must hold for the data. A Database Management System (DBMS) provides capabilities for defining and enforcing these constraints. The simplest type of integrity constraint involves specifying a data type for each data item. For example, while using a database, you can define the type of data a column can take. If you set an appropriate type, such as real numbers for employee's 'Salary' column, you ensure that a salary cannot be 'abcd'. Several other integrity constraints ensure data integrity.

Restricting unauthorised access: A DBMS provides a security and authorisation subsystem, which the Database Administrator uses to create accounts and specify account restrictions. The DBMS should then enforce these restrictions automatically.

In simple words, databases let you control who gets access to which data.

Programming advantage and recovery of data

Programming advantages:

- **Persistent storage for the program:** The application data must be persistent across multiple sessions when developing computer applications. For example, if you change your address in Facebook, logout and login again to your profile, you would expect the updated address to be there still. This is known as *persistency*. A database provides us with the best possible way to make our applications persistent.
- **Multiple user interface:** A database provides an interface with flexibility such that the same database can be used for multiple applications. For example, when you log in to our LMS and Student Online applications using your browser, the two applications look different (different user interface) and do different things (different functionalities). However, underneath, they might use the same student database that stores student information.

Providing backup and recovery:

Suppose the computer system fails in the middle of a complex update program. In that case, the recovery subsystem ensures that the database is restored to the state it was in before the program started executing. In other words, databases enable backing up our data and recovering stored data in case of a system failure.

Once we learn more about the structures and integrity constraints of relational databases, you will be able to appreciate the benefits of databases. So, be excited!

Last modified: Wednesday, 17 August 2022, 8:10 PM



Quiz: Check your understanding



Quiz

Let's check your understanding of what we have just covered with a quick quiz. Can you score 100 per cent?

Your task

1. Complete all the questions.
2. Use the results to reflect on any areas of the lessons you may need to revisit.

Guidelines

- This quiz is not graded but is an essential part of your learning.
- Please try to complete this activity as early as possible.

Grading method: Highest grade

Attempt quiz now



Discussion: What is a database?



Discussion

What is a database? Why use a database? Is there a time when you tried to use a database, and it was broken? How did you feel about this?

In this discussion, you will list and describe 1 to 3 examples of database types.

Your task

1. Search and find some popular types of databases. These may be things like Oracle. You can add more information, like the differences between each database type (if you can find this type of information).
2. Try and find different types from your classmates; you will be surprised at how many there are.
3. Write at least 50 words per answer.
4. Post your response on this page.
5. Review your classmates' responses and comment constructively on at least two at the end of the week. Is there anything unusual in their response (or similar)?

Guidelines

- This activity is not graded but is an essential part of your learning.
- You should spend at least 20 minutes on this activity.
- Submit your first post by the end of the week to allow your classmates time to respond constructively.
- Once you have posted, respond to at least two of your colleagues' posts.
- Complete this activity as earlier as possible.

Add a new discussion topic

Subject 


Message 












Post to forum

Cancel

Advanced



There are required fields in this form marked  .

Discussion ↓	Started by	Last post	Replies	Subscribe
☆ Using databases without thinking about it	 Walmiria Woodliff 8 Sep 2022	Casey Lee Ram... 13 Sep 2022	3	<input checked="" type="checkbox"/> ⋮
☆ Tableau / PostgreSQL	Christina Ycasi... 8 Sep 2022	Christina Ycasi... 8 Sep 2022	0	<input checked="" type="checkbox"/> ⋮
☆ Relational vs Non-relational database	Fernando Marti... 8 Sep 2022	Fernando Marti... 8 Sep 2022	0	<input checked="" type="checkbox"/> ⋮
☆ Previous Financial Markets Database Project	 Cameron Jame... 4 Sep 2022	 Callan Hiho 7 Sep 2022	4	<input checked="" type="checkbox"/> ⋮
☆ Popularity Contest - MySQL vs PostgreSQL	Catherine Bower 5 Sep 2022	Casey Lee Ram... 13 Sep 2022	3	<input checked="" type="checkbox"/> ⋮
☆ Painting Database	Vera Ann Cook 6 Sep 2022	 Hai Thanh Truo... 8 Sep 2022	1	<input checked="" type="checkbox"/> ⋮
☆ NetSuite database	 Hai Thanh Truo... 8 Sep 2022	Carmelita Jade ... 11 Sep 2022	1	<input checked="" type="checkbox"/> ⋮
☆ MySQL in Healthcare Research	Carmelita Jade ... 11 Sep 2022	Carmelita Jade ... 11 Sep 2022	0	<input checked="" type="checkbox"/> ⋮
☆ Mongoose Database	 Yu-Jen Cheng 8 Sep 2022	Carmelita Jade ... 11 Sep 2022	1	<input checked="" type="checkbox"/> ⋮
☆ Inertia of Workplace Database Systems	 Callan Hiho 7 Sep 2022	 Callan Hiho 7 Sep 2022	0	<input checked="" type="checkbox"/> ⋮
☆ Databases in research	Casey Lee Ram... 13 Sep 2022	Casey Lee Ram... 13 Sep 2022	0	<input checked="" type="checkbox"/> ⋮
☆ Database products	 Kundai Gongga 8 Sep 2022	Fernando Marti... 12 Sep 2022	1	<input checked="" type="checkbox"/> ⋮
☆ Commercial vs Open Source Databases	 Hajira Bilal 8 Sep 2022	Christina Ycasi... 8 Sep 2022	1	<input checked="" type="checkbox"/> ⋮
☆ Cloud Database	 Amanpreet Kaur 10 Sep 2022	Fernando Marti... 13 Sep 2022	1	<input checked="" type="checkbox"/> ⋮



Topic 2: Relational database model

Welcome to our second topic. In this topic, we will learn about the basic concepts of relational databases. We will first look into the history of databases, followed by the current popularity of Oracle DBMS, and then look into the fundamental concepts on which relational databases are built.



Recommended reading

- The recommended reading for this topic is Chapters 3 and 5 of [R. Elmasri and S.B. Navathe \(2017\) *Fundamentals of Database Systems*, 7th edn., Pearson.](#)

The information in these two chapters explain the concepts of relational databases in a clear and organised manner, which you might find very useful.

What is a data model?

The word model refers to something that is a simplified representation of a complex reality. A **data model** is the representation (description) of the structure of a database.

The database structure includes the data types, relationships, and constraints that should hold for the data. Essentially, a data model describes:

- the different subjects being stored
- the relationships that exist between these subjects
- the constraints or restrictions that must be true within the database context,

Different data models

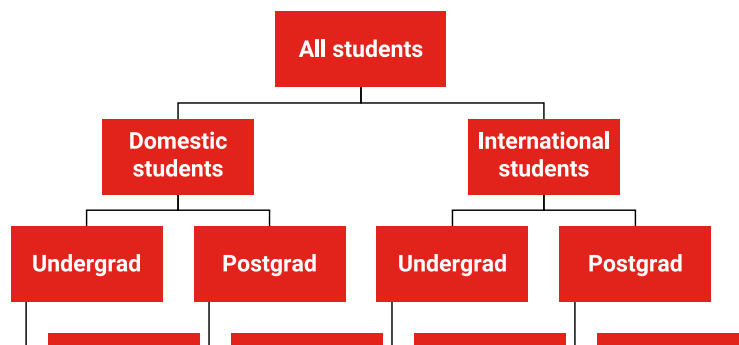
In the history of databases, several data models have been introduced over the last few decades. Three of these are described below.

Click on the below headings to reveal further information.

Hierarchical model

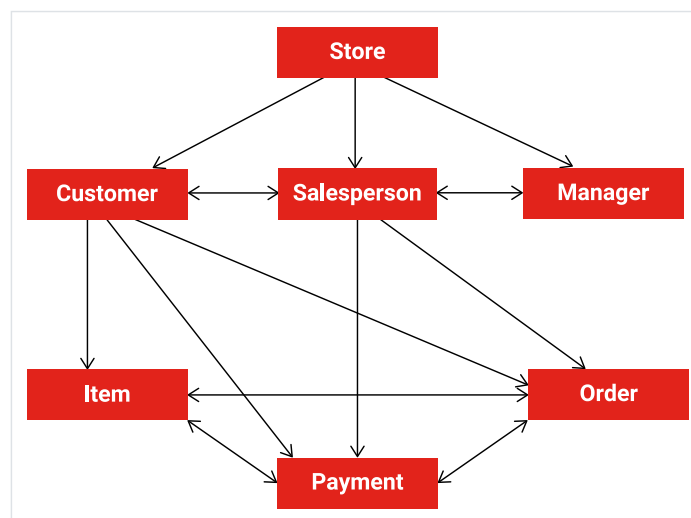
This was first introduced in the 1960s by IBM. The DBMS based on this model was known as an Information Management System (IMS), and it was used in the Apollo 11 project, a project responsible for sending a spacecraft to the Moon. The model represents the data in a database with a tree structure consisting of progressive parent-child relations. However, the tree-like structure had limitations in addressing real-life problems.





Network model

This was first introduced in the 1960s by General Electric (GE). The model represents the data in a database as interconnected nodes. It tried to address the limitations of the hierarchical model, but the success was limited.



Relational model

Later in the 1970s, the relational model was introduced. This model represents the data in a database as a collection of relations/tables. Interestingly, the model was not proposed by an IT company. Instead, it was proposed as part of an academic thesis by Edgar Codd. Soon, the model became very popular within the IT industry because of its structural simplicity. The table-based model was easy to comprehend and relatable to real-life scenarios.

Object-oriented model

Although the relational database model is still the most popular data model, the Object-oriented model was introduced in the 1990s to accommodate object-oriented concepts (e.g. class, interface, etc.) in a database. However, by that time, the popularity and use of relational databases were so invasive within the IT community that it remained the most-used database model to date.





Activity: The history of information management

Your task

Conduct an internet search on the Apollo 11 project and IMS using Google. Identify the driving forces that led to the creation of the first information management system. Identify the factors that remain at the heart of databases and database systems today.

Guidelines

- This activity is not graded but is an essential part of your learning.
- Spend around 20 minutes on this activity.
- If you have any questions, you can contact your tutor for guidance.
- Complete this activity by Monday of week 2 before 12:00 midday (AEST).

Last modified: Wednesday, 17 August 2022, 8:10 PM



Why Oracle?

Given the massive number of relational database products out there, we have chosen to introduce and practice the concepts of relational databases using Oracle DBMS. First, you need to understand why.



Activity: Database rankings

Your task

Head to the [database ranking site](#), a website that lists the popularity of database management systems by scrolling the entire web and counting the number of times a particular DBMS was mentioned on the web. Also, click on the trend chart (towards the top right of the page).

- Which database holds the top ranking?
- Which database has held the top ranking most consistently over time?

Guidelines

- This activity is not graded but is an essential part of your learning.
- Spend around 20 minutes on this activity.
- If you have any questions, you can contact your tutor for guidance.
- Complete this activity by Monday of week 2 before 12:00 midday (AEST).

As you can see from the database ranking website, Oracle is the most popular relational database management system among database practitioners, and its popularity has been steady for the last seven years. Because of this steady popularity trend, what better DBMS can you think of for learning relational databases?

Now that we are familiar with the history of database models and the popularity of Oracle DBMS, let us get started on the fundamental concepts of a relational database model.

Last modified: Wednesday, 17 August 2022, 8:11 PM



Relational model

There are three major aspects of a relational data model.

Click on the below headings to reveal further information.

Data structure

The structural part mainly outlines how the data in a relational model should be structured. The structure generally consists of two dimensions (rows, columns) tables/relations. The values of a column (also known as attributes) come from a particular domain and are of atomic types.

Data integrity

The integrity part of the relational model outlines the different integrity constraints that must hold true in a relational database. Data integrity mainly consists of two general integrity rules, namely entity integrity constraint and referential integrity constraint.

Data manipulation

The manipulation part of the relational data model sets out how the data should be manipulated (creation of data and extraction of data). The manipulation part consists of a set of algebraic operators for data manipulations. (We will learn this in a later topic.)

Database case study

To introduce the concepts of **Data structure** and **Data integrity** of a relational data model, let's look at a case study. When you work through the case study, you will be asked to contribute your ideas.

You have been tasked with developing a Melbourne's public transport tram system database. We will call the database the 'Tram Database'.



Activity: Tram Database data types

Your task

Your first task is to itemise the different types of data that need to be stored in the Tram Database. Think about catching a tram and write down all the data types needed to make this happen. Here are some main headings to get you started. For each main heading, list the data types that would need to be captured.

- Tram vehicle
- Tram route
- Track
- Driver
- Shift (for a driver)
- Timetable.

Once you have come up with a list, compare it with the case study example by clicking on the solution button.

Click on the below headings to reveal further information.

Solution

- Tram vehicle: number, type, capacity, status, maintenance record, current depot . . .



- Route: number, start, end, tracks . . .
- Track: streets, sections . . .
- Driver: name, rating, personal details . . .
- Shift: driver, tram, route, times . . .
- Timetable: routes and times.

Guidelines

- This activity is not graded but is an essential part of your learning.
- Spend around 10 minutes on this activity.
- If you have any questions, you can contact your tutor for guidance.
- Complete this activity by Monday of week 2 before 12:00 midday (AEST).

Many different types of data are involved in running trams.

Information about each kind of 'thing' is presented in a table with rows and columns in a relational database. Of course, these tables need more columns to be realistic!

For simplicity's sake, let's assume that the columns listed below are the only columns we need to deal with.

VEHICLE			ROUTE		
No	Type	Seats	No	Start	End
181	W	66	69	Hawthorn	St Kilda
182	W	64	75	City	East Burwood
622	LR	102	48	Pt Melbourne	North Balwyn

DRIVER		SHIFT			
Name	Rating	Driver	Route	Tram	Day
Adams	A1	Bernard	69	181	Wed
Bernard	C1	Bernard	69	181	Thu
		Adams	48	622	Thu

Tram database

Next, we need to set the context and outline some real-life rules that must hold true in a tram database. Fill in the blanks below to establish these rules more clearly.



Activity: Tram database rules

Referring to the Tram Database tables above, fill in the missing words.

- Every vehicle must have a .
- It is implicit that driver's are unique, or the data won't make sense.
- A column in the Shift table should correspond to a row in the Vehicle table.
- Suburb entries should be valid suburbs.
- Day entries should be of the week.

☒ Check



Last modified: Wednesday, 17 August 2022, 8:11 PM



Data structure

The structural part of a relational database model is founded on three major concepts derived from set theory.

They include:

- Domain
- Relation
- Attribute.


First, we will look into these concepts using the Tram Database example. We will then look into the concept of keys before discussing the integrity part.

Domains

Domain refers to the scope of data values in a particular column of a table, i.e. the values that a single row in that column can take.

For example, we say:

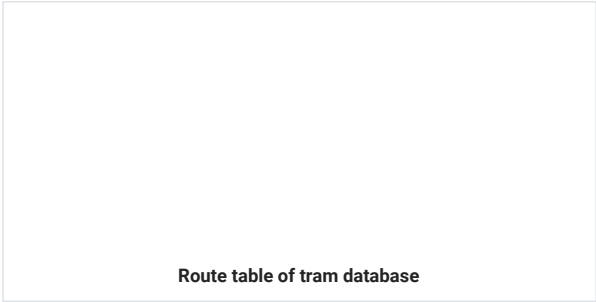
- The domain of the column 'day' in the 'shift' table is the set of all possible days: {Mon, Tue, Wed, Thu, Fri, Sat, Sun}



Shift table of tram database

So, the column 'Day' can take the value 'Mon' but should never take the value 'Egg'! It will not make any sense.

- The domain of the 'Start' and 'End' columns of the 'Route' table should be the set of valid suburbs of Melbourne.



Route table of tram database

If you had an Employee table in our institutional database, the domain of 'salary' might be the set of all floating-point numbers greater than 0 and less than 200 000 (say).

So, put simply, a **domain** is the set of values from which a simple row of a column takes its value.



Activity: Tram database domains

Your task

Consider the tram timetable:

1. Make a list of the data types that should appear on a timetable.



2. Define the set of acceptable values (domain) for each data type.

Once you have come up with a list and domain values, compare it with the case study example by clicking on the solution button.

Click on the below headings to reveal further information.

Solution

Data types may vary but will generally include:

- Route (start and endpoints (suburbs))
- Tram stops (which could be the stop number and location)
- Days of the week
- Time of the day.

Guidelines

- This activity is not graded but is an essential part of your learning.
- Spend around 10 minutes on this activity.
- If you have any questions, you can contact your tutor for guidance.
- Complete this activity by Monday of week 2 before 12:00 midday (AEST).

Last modified: Wednesday, 17 August 2022, 8:11 PM



Relations and attributes

Formally, a **relation** is a subset of the Cartesian product of a list of domains characterised by a name. A relation is a two-dimensional table composed of **tuples** (also known as rows) and **columns** (also known as attributes). Each tuple in the relation represents an instance of the 'thing' represented by the table. A name also identifies each column.

VEHICLE		
No	Type	Seats
181	W	66
182	W	64
622	LR	102

Vehicle table

For example, in the 'Vehicle' table/relation above, there are:

- three attributes/columns (i.e. No, Types, Seats)
- three rows/tuples
- the relation represents the 'thing' (vehicle)
- one row in the relation (let us say the second) represents a vehicle's instance (one real-life object). The **instance** is identified by vehicle number 182, vehicle type of 'W' and capacity of 64.

Note: In a relational database, the terms 'relation' and 'relationship' mean two things. **Relation** refers to the concept we have just learned, which is nothing more than a two-dimensional table. However, **relationship** refers to the connection between two or more tables. Soon, we will see examples of relationships.

Keys

It is often desirable to extract a particular tuple from a relation. For example, with a 'bank database', each client wants the balance of a particular account.

That is, there must be some way of uniquely identifying each tuple. This is usually achieved by defining some attribute as a key or a unique label. For example:

- The route number is a key in the 'Tram Database' – different routes have different numbers.
- In banks, each account number is a key. Name and address is not a key – there can be more than one person of the same name at the same address.
- By definition, keys are unique, so no two tuples can have the same key. A set of attributes can also be a key. For example, (Start, End} is a key for the route.

So, generally, a set of attributes (a set might contain one or more elements) forms a **key**, which can uniquely identify a **tuple** (or row) in a relation.

There are two types of keys

Click on the below headings to reveal further information.

Primary key

A **primary key** is the most efficient (i.e., smallest and most meaningful) set of attributes that can uniquely identify a tuple in a relation.

In relational databases, primary keys are underlined.

Primary key example table

--



Primary key

Vehicle

No	Type	Seats
181	W	66
182	W	64
622	LR	102

Foreign key

An attribute's value for one relation can be drawn from another relation.

For example, Shift:Tram values must be from Vehicle:No. (but not vice versa – vehicles are not necessarily scheduled to a shift).

'Shift' table

Vechcle			Shift			
No	Type	Seats	Driver	Route	Tram	Day
181	W	66	Bernard	69	181	Wed
182	W	64	Bernard	69	181	Thu
622	LR	102	Adams	48	622	Thu

Foreign key

In the 'Shift' table, the value in the 'Tram' column for each row refers to the value in the 'No' column of the 'Vehicle' table. The 'Tram' column in the 'Shift' table is to identify a particular vehicle from the 'Vehicle' table to be associated with a particular shift. The 'Tram' column in the 'Shift' table refers to values drawn from another table (i.e. vehicle).

These sorts of columns are known as **foreign keys in a relational database**.

Note: foreign keys establish the relationship (i.e., connection) between two or more relations. So, in the above scenario, there is a relationship between the 'Shift' and 'Vehicle' relations, and it is being formed by the 'Tram' and 'No' columns of the respective tables.



Activity: Primary and foreign keys

Your task

Consider a university database containing the following relations:



Student

Student ID	First Name	Last Name

Subject

Subject ID	Subject Name	Lecture ID

Enrolment

Subject ID	Student ID	Grade	Start date

Lecturer

Lecturer ID	Lecture Name	Contact details

Academic record

Academic record ID	Student ID	Subject ID	Grade	Completion date

For each relation list the:

- Private key(s)
- Foreign key(s).

Once you have listed the keys, click on the solution button below to compare your answer

Click on the below headings to reveal further information.

Solution

Relation	Private key(s)>	Foreign key(s)
Student	Student ID	Nil
Subject	Subject ID	Nil
Enrolment	Nil	Subject ID, Student ID
Lecturer	Lecturer ID	
Academic record	Academic record ID	Student ID, Subject ID



Guidelines

- This activity is not graded but is an essential part of your learning.
- Spend around 10 minutes on this activity.
- If you have any questions, you can contact your tutor for guidance.
- Complete this activity by Monday of week 2 before 12:00 midday (AEST).

Last modified: Wednesday, 17 August 2022, 8:11 PM



Data integrity

Now that we have learned about the structural part of the relational model and are also familiar with the concepts of keys, we will look into the concepts of Data Integrity. There are two integrity constraints in relational databases:

1. Entity Integrity Constraint
2. Referential Integrity Constraint .

Entity integrity constraint

According to an **entity integrity constraint**:

- the primary key value cannot be null
- the primary key must not be duplicated.

In database terms, 'null' refers to emptiness.

So, given the vehicle table:

VEHICLE		
No	Type	Seats
181	W	66
182	W	64
622	LR	102

'Vehicle' table

If the primary key is 'No' (Number), then according to the entity integrity constraint, you cannot have a vehicle in this table without a value for 'No' (cannot be null).

Moreover, if you already have a vehicle in the table with 'No' value 181, you cannot have a second one with the 'No' value 181 (cannot be duplicated).

Referential integrity constraint

The **referential integrity constraint** is specified between two relations and is used to maintain consistency among tuples of the two relations.

The constraint specifies that the value of a foreign key must be valid.

For example:

'Shift' table (1)						
Vechcle			Shift			
No	Type	Seats	Driver	Route	Tram	Day
181	W	66	Bernard	69	181	Wed
182	W	64	Bernard	69	181	Thu
622	LR	102	Adams	48	622	Thu

Foreign key

According to the referential integrity constraint, suppose 'Tram' in the 'Shift' table is a foreign key, which refers to 'No' in the 'Vehicle' table. In that case, you cannot have a row in the shift table that has the value for the tram column to be 200 (for example) if there is no vehicle in the vehicle table with a 'No' value equal to 200.



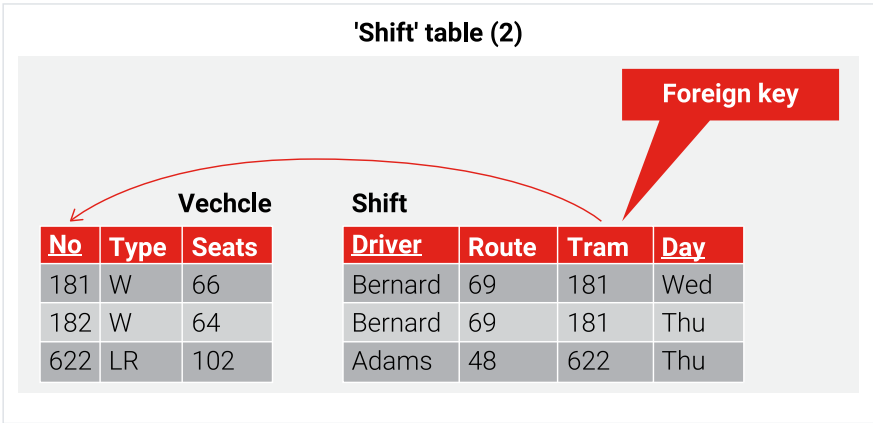
Food for thought: given the above data, if we now decide to remove tram number 622 from our database, how can we preserve the referential integrity constraint, given the third row of the shift table will no longer refer to something valid?

Last modified: Wednesday, 17 August 2022, 8:11 PM



Preservation of referential integrity

We can take three approaches to preserve referential integrity while updating or deleting the value of a target of a foreign key reference. Let us discuss the scenario where we want to delete vehicle number 622 from the vehicle table:



Click on the below headings to reveal further information.

RESTRICTED

The update/delete operation is restricted to the case where there is no such matching reference (it is rejected otherwise). So, if we take this approach, the database will reject our attempt to delete the last row of the vehicle table since the third row of the 'Shift' table refers to this particular vehicle.

CASCADE

The update/delete operation cascades to those matching references. If we take this approach, when we delete the third row of the 'Vehicle' table, the database will also delete the third row (or any other row that refers to vehicle number 622) of the 'Shift' table.

NULLIFIES

The foreign key is set to null in all such matching references, and the target is then updated/deleted. If we take this approach, after the third row of the 'Vehicle' table is successfully deleted, the value of the 'Tram' column of the third row of the 'Shift' table will be set to 'Null' (i.e., empty/meaningless value).

These strategies are handy in preserving the referential integrity constraint in a relational database. We will see these strategies in practice when learning structured query language (SQL).

Last modified: Wednesday, 17 August 2022, 8:12 PM



Quiz: Check your understanding



Quiz

Let's check your understanding of what we have just covered with a quick quiz. Can you score 100 per cent?

Your task

1. Complete all the questions.
2. Use the results to reflect on any areas of the lessons you may need to revisit.

Guidelines

- This quiz is not graded but is an essential part of your learning.
- Please try to complete this activity as early as possible.

Grading method: Highest grade

Attempt quiz now



Activity: Data integrity



Activity: Data integrity

Your task

Consider the university database from the previous activity, with data added. Assume the only data in the database is that which appears below:

Student

Student ID	First Name	Last Name
ABC-1	Chen	Ying
CDE-2	Ronaldo	Alves
	George	Washington
YDG-9	Chen	Ying

Subject

Subject ID	Subject Name	Lecture ID
BUA5MGT	Management	Q234
BUA5MFN	Financial Management	H748
BUA5FMA	Accounting	M038
BUA5HRM	Human Resources	P534

Enrolment

Subject ID	Student ID	Grade	Start date
BUA5MGT	CDE-2	C	2/2/20
BUA5MGT	YDG-9	D	4/6/21
BUA5MFN	CDE-7	HD	7/8/21
BUA5FMA	ABC-1	P	4/2/22

Lecturer

Lecturer ID	Lecture Name	Contact details
Q234	Max Plank	1234
H748	Tedd Codd	2345
M038	Carolyn Begg	3456



Lecturer ID	Lecture Name	Contact details
Q234	Tom Connolly	6789

Answer the following questions:

1. What entity integrity constraints exist in the data?
2. What referential integrity constraints exist in the data?
3. Assume the subject BUA5FMA is no longer offered. What will the relation tables look like if the preservation effect is 'cascade'?

Once you have arrived at an answer, click on the solution button below to compare your answer.

Click on the below headings to reveal further information.

Solution

1. Entity integrity constraints:

- In Student relation: Student ID value for George Washington – cannot be null.
- In Lecture relation: Lecturer ID for Tom Connolly – cannot be a duplicate.

2. Referential integrity constraints:

- In Subject relation: Lecturer ID 'P534' is unknown in the Lecturer relation.
- In Enrolment relation: Student ID 'CDE-7' is unknown in Student relation.

3. The Relation tables will look like the following:

Student

Student ID	First Name	Last Name
ABC-1	Chen	Ying
CDE-2	Ronaldo	Alves
	George	Washington
YDG-9	Chen	Ying

Subject

Subject ID	Subject Name	Lecture ID
BUA5MGT	Management	Q234
BUA5MFN	Financial Management	H748
BUA5HRM	Human Resources	P534

Enrolment

Subject ID	Student ID	Grade	Start date
BUA5MGT	CDE-2	C	2/2/20
BUA5MGT	YDG-9	D	4/6/21



Subject ID	Student ID	Grade	Start date
BUA5MFN	CDE-7	HD	7/8/21

Lecturer

Lecturer ID	Lecture Name	Contact details
Q234	Max Plank	1234
H748	Tedd Codd	2345
M038	Carolyn Begg	3456
Q234	Tom Connolly	6789

Guidelines

- This activity is not graded but is an important part of your learning.
- Spend around 10 minutes on this activity.
- Remember that if you have any questions, you can contact your tutor for guidance.
- Complete this activity by Monday of week 2 before 12:00 midday (AEST).

Last modified: Wednesday, 17 August 2022, 8:12 PM



Summary and checklist

This week you were introduced to databases and relational databases. You were introduced to the importance of data and entity-relationship modelling, how primary and foreign keys relate to different attributes of relations and the constraints used to preserve data integrity.

Here's a checklist of the activities you need to complete this week:

- **Complete all required reading**
- **Discussion:** what is a database, and your experience in using them?
- **Activity:** the history of information management
- **Activity:** Database rankings
- **Activity:** Tram database data types
- **Activity:** Tram database rules
- **Activity:** Tram database domains
- **Activity:** Primary and foreign keys
- **Activity:** Data integrity
- **Quiz:** check your understanding.

Now that we understand the basics of relational databases, we will look into modelling such databases next week. We will start with a top-down modelling and design approach, i.e., entity-relationship modelling.

Now that you have finished this week of learning use the navigation bar on the left of this site to navigate to the landing page for the next week of learning to the subject home page.

References

Elmasri, R. and Navathe, S.B. (2017) *Fundamentals of Database Systems*, 7th edn., Pearson.

Hoffer, J. A., Prescott, M. B., & McFadden, F. R. (2005). *Modern database management*. Upper Saddle River, N.J: Pearson/ Prentice Hall.

Last modified: Wednesday, 17 August 2022, 8:12 PM

