Learner Guide

366917666 / Sashkin / Shutterstock.com

ICTDBS506

Design databases

tafe.qld.edu.au

MAKE GREAT HAPPEN

tafe Queensland

Unit version	Document version	Release date	Comments/actions
ICTDBS506	V2	April, 2022	

Copyright

Disclaimer

S113P

Introduction

This Learner Guide has been developed to support you as a resource for your study program. It contains key information relating to your studies including all the skills and knowledge required to achieve competence.

What will I learn?

This unit describes the skills and knowledge required to establish client needs and technical requirements and to design a database that meets identified requirements.

It applies to those who are employed in roles including database administrators or designers who are required to design databases.

Where can I get more information?

For further information on your qualification, accredited course or Unit/s of Competency, please go to: ICTDBS506 - Design databases

TAFE Queensland student rules

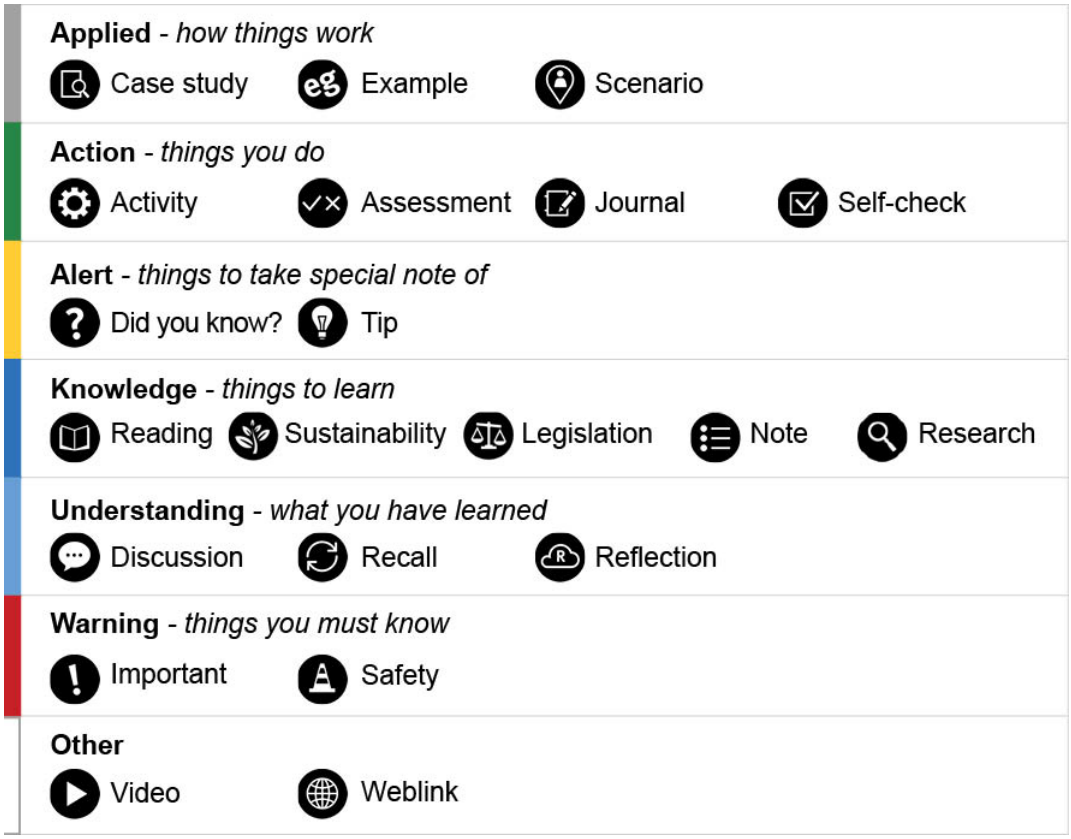
TAFE Queensland student rules are designed to ensure that learners are aware of their rights as well as their responsibilities. All learners are encouraged to familiarise themselves with the TAFE Queensland student rules, specifically as they relate to progress of study and assessment guidelines. A full guide to student rules can be found at Student rules[1].

Information to support your learning and assessment

There's always someone to help you. Undertaking further study can bring both excitement and challenges. Our Student Services, Learning Support and Library staff can help you make the most of your time at TAFE.

Callout panels

A number of panels have been designed to help guide you to important information and actions throughout this Learner Guide. The full choice of panels you are likely to encounter to support you in your studies are included below. NB: not all the panels will be used in every learner guide.



[1] http://tafeqld.edu.au/current-students/student-rules/

Contents

Welcome

Welcome to the *ICTDBS506 - Design databases,* unit.

By completing this unit, you will be able to:

- Determine database requirements
- Develop logical data model
- Design data structure
- Design queries, screens and reports
- Design access and security systems
- Confirm database design

Fundamentals of database design

Introduction

Throughout this unit of study, we will be taking a logical progressive approach to database design, starting from identifying the user requirements and ending with client acceptance and sign-off of the design.

In this first topic we introduce the fundamentals of database design. More specifically we will look at **relational databases**. We explore various progressive models of design including conceptual, physical, and logical. By progressive we mean that the design process follows a distinct logical sequence.

Objectives

By completing this topic it is expected that you will be able to:

- understand what a relational database is
- understand the differences between conceptual, logical, and physical design models.

366917666 / Sashkin / Shutterstock.com

Relational databases and database design

In simple terms, a database is a collection of related and meaningful data that is stored with a particular purpose in mind. A collection of random unrelated data does not constitute a database.

Examples

There are examples of databases all around us that we interact with, sometimes unconsciously, like:

- a pile of business cards kept in a wallet
- school records storing student grades and comments
- bank accounts holding financial information
- police files containing criminal records
- insurance company databases keeping track of clients' claims history
- stock exchange data tracking live and historical stock prices.

Data is stored in a database in a logical manner, that is, in a way that is accessible to the user of the database. You do not necessarily need a computer and software package to design a database. A database can be a simple list of names and addresses written on a collection of Post-It Notes. It could be a filing system used in an office that stores paper in filing cabinets. Most commonly databases store data electronically (e.g. bank account records).

260198537 / Africa Studio / Shutterstock.com

Database Management System (DBMS)

A **DBMS** is a general-purpose software we use to create and maintain a database by facilitating the tasks of defining, constructing, and manipulating the database. Defining a database involves creating a structure and providing details of each data type. As we will see, there are common data types used to store data.

Example

When designing the database for a banking organisation we may need to store account balances (as currency values), and account names (as string values).

Once the database structure has been designed, we can then store the data within the structure.

Database table - Colours and their meaning for the table.

Each column has an assigned name and data type that is determined during design.
Each column has an assigned name and data type that is determined during design. In this column, we are storing string values using the VARCHAR data type.
This is a row or record.
This is a field. A 'field' is where the column and row intersect and each 'field' holds a data value.
Notice the 'NULL' value in the table. This means that a value has not yet been set.

accountID (Number – Integer)	accountName (String – VARCHAR)	accountBalance (Currency – Decimal)	lastLogin (Date – Time)
20340	Simon Miles	100.10	2016-02-08-16:54:23
20341	Daryl Norton	3400.00	2016-02-01-08:15:12
20342	Margarita Daniels	253.85	2016-02-04-12:11:44
20343	Cora Rogers	0.00	NULL

When we design a database, in consultation with the client, we determine:

- how many tables our database will contain
- how many columns each table will have
- the data types for each column.

What is a relational database?

197803430 / Marynchenko Oleksandr / shutterstock.com

The **relational database** model was developed by an IBM research scientist, Dr Edgar F. Codd in the late 1960s. Codd was dissatisfied with existing database models, which are discussed further below (these are the hierarchical and network models). As a mathematician, Codd developed an alternative database model that would be based on data sharing relations. The revised model was published in June 1970 in an academic paper titled: 'A Relational Model of Data for Large Shared Banks'. In this new model relations are represented as tables, which are created, manipulated, and managed by database administrators. Each table contains a number of 'fields' (or columns). Rows of data are stored in a database table where each field for a given row holds a specific value.

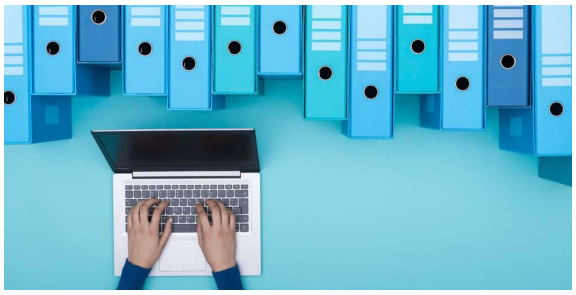
Think of a database as a container. The container holds a number of tables. Data is stored in a table where each row in the table represents a single record. Manipulating a database consists of being able to access, update, and retrieve the desired information from the database.

Some relational database characteristics include:

- the use of special key fields, often referred to as primary and foreign keys, which define relationships between the tables in the database which enable users to retrieve corresponding data from multiple tables and then use this data in meaningful ways
- the ability to store data across multiple tables
- the ability to avoid duplication of data by a process called normalisation where data is efficiently organised by implementing a set of rules referred to as normal forms.

The most popular relational database management systems (RDBMS) today include:

- Oracle
- MySQL
- Microsoft SQL Server
- PostgreSQL
- MariaDB
- Informix
- SQLite.



1029020020 / Stokkete / Shutterstock.com

Database architecture

Database design typically consists of a sequential series of six phases. The need to design a database normally comes as a result of a problem that has been identified. Perhaps the client's organisation is looking to systemise an existing process conducted manually, or it could be that the organisation has grown or changed, and the database needs to be changed as a result.

Video

The following video by Prescott Computer Guy (2011) provides the basic information in regard to Relational Database Concepts[2]

by Prescott Computer Guy (5:25 mins) – hosted on YouTube (2011)

The six phases of database design

Requirements, Collection and Analysis

Conceptual Design - Will the end users be able to view the database model from a high vantage point (conceptually)?

Choice of DBMS

Logical Design – Can the database designers view the database model from a logical vantage point? This is also called Data Model Mapping.

Physical Design – Can the programmers and database administrators view the model from a physical perspective?

Implementation

[2] https://www.youtube.com/embed/NvrpuBAMddw

While it is important to be aware of all the above six phases, most database designs can be considered to occur in three major phases:

- **Conceptual design** – End users can view the database model from a high vantage point (conceptually).
- **Logical design** (also called data model mapping) - Database designers are able to view the database model from a logical vantage point.
- **Physical design** - Programmers and database administrators (DBAs) are able to view the model from a physical perspective.

These three phases are often referred to as "Three-Schema Architecture" and help to evaluate a relational database from different vantage points. End users can view the database model from a high vantage point (conceptually).

Database designers are able to view the database model from a logical vantage point, and lastly, programmers and database administrators (DBAs) are able to view the model from a physical perspective.

It is imperative that adequate time is spent preparing and planning the database through each of the phases. To commence, we look at the first phase, requirements collection and analysis, in more detail.

Topic summary and review

In this first topic, we learnt what a **relational database** is, what a **DMBS** is, and how the design process is organised into six phases.

More broadly, however, database designers like to focus on three main phases – the conceptual design phase, the logical design phase, and the physical design phase.

Complete the activity below to help memorise the phases of a database design project.

Self-check – Phases of database design

Match the correct phase to its corresponding number to check your understanding of the six phases of database design.

Answers: 1, 2, 3, 4, 5, 6

Phase name	Phase number
Implementation	
Conceptual Design - Will the end users be able to view the database model from a high vantage point (conceptually)?	
Choice of DBMS	
Requirements, Collection and Analysis	
Physical Design – Can the programmers and database administrators view the model from a physical perspective?	
Logical Design – Can the database designers view the database model from a logical vantage point? This is also called Data Model Mapping.	

Check your answers at the end of this Learner Guide

Self-check – Fundamentals of database design

Complete the questions below to check your understanding.

Complete the sentence.

A relational database is based on data sharing _____.

- ☐ tables
- ☐ relations
- ☐ fields
- ☐ values

Relations between data are represented as _____.

- ☐ tables
- ☐ relations
- ☐ fields
- ☐ values

Complete the sentence by choosing the correct words.

Words: *relations, tables, *fields, values, relations, tables, fields, *values*

Tables contain columns which are…… and rows which hold specific ……

What are special key fields? (answer consists of 4 words)

…… and …… ……

What process is used to avoid data duplication (also known as redundancy)?

.....

Note: *How to follow the normalisation process will be learnt in Phase 4.*

Check your answers at the end of this Learner Guide

Phase 1 - Requirements Collection and Analysis

Introduction

This phase is also known as the pre-design phase. It involves a process of discovery and facts finding. We will discuss models used to identify user requirements. What is the purpose of the database? The models we learn will help both us and the client, identify, and categorise answers to this question.

At an introductory level, we will learn how to determine the business rules that will guide the database project, identify the key stakeholders and learn about certain diagrams that can help us plan our design using **UML (the Unified Modelling Language)**.

Lastly, we discuss some of the types of documentation, including **use case diagrams** and checklists that will be produced in the design process.

Objectives

By completing this topic it is expected that you will be able to:

- gain skills in performing user-needs analysis
- gain an introductory understanding of the use case UML diagram
- learn about different documentation types used in database design.

64558985 / Rawpixel / stock.adobe.com

The methods of requirements collection and analysis

There are many methods we can employ to help in requirements collection:

- questionnaires
- researching
- site visits
- observation in the work environment
- prototyping (rapid iterative development of specific functionality)
- brainstorming sessions
- Joint Requirements Planning (JRP) session
- sampling existing documentation, forms, and databases.

We use different techniques to communicate and enquire with the aim of discovering information about the organisation and its current processes, to gather as much detail as possible.

Discussion

Research examples of user-needs analysis.

What should you avoid when conducting user needs analysis? (Give 5 examples.)

Discuss your findings in the classroom.

Take the necessary time to:

- understand what problems there are with the current system (if applicable)
- identify all stakeholder requirements to clarify expectations relating to the new system
- review existing artefacts (documentations, forms, databases, organisational policies, and rules)
- understand all the processes of the current system (that is, what it does, and how it is done?)
- understand the business rules of the organisation in relation to the project.

Business rules

What are business rules?

They are short point-form facts that describe specific objects, actions, and relationships that the organisation deems to be important and that relate to the project.

Business rules always impose a ruling or constraint on the data, and they may include the type of data allowed in the database or how the data is collected or displayed.

373686427 / Shift Drive / shutterstock.com

Examples

Some database related business rules could be as follows:

- A customer can only register once.
- Unique email addresses must be used for each customer.
- A customer is flagged as inactive if no orders have been placed within the past three months.
- If a customer orders more than $50 in products, they will receive free delivery.
- A customer cannot purchase more than 1000 items in one single transaction.
- A customer must register with a valid mobile phone number.
- To register, the customer must specify both a first and a last name.

All the business rules pertaining to the project should be collated in this phase one (requirements collection and analysis). Many of these rules will be programmed into the database after the databases schema is created (phase 5).

Database designers should consider interviewing key stakeholders within the organisation, at various stages throughout the project's life cycle where a set of predefined questions can be asked, and responses collated.

Depending on the size of the project and the organisation in question, there may be numerous stakeholders involved with the system, who may include:

- project sponsors
- management (senior, middle, operational)
- subject matter experts
- technical staff
- end users
- client
- users

Once the designer has gathered and analysed all relevant information about the organisation and the system, a statement of user requirements or a requirements document as it is also called, should be presented to the client for approval. This is a formal document that will form the basis for the contract or agreement between client and database developer. Additional documentation may include use case (UML) diagrams preliminary ERD and other diagrams. Computer-aided techniques such as CASE tools (Computer Aided Software Engineering) can also be used.

Determining database client requirements

Determining client requirements includes using a number of methods and techniques to gather information and specifically to answer one question - why does the client need the database? The answer to this question usually falls into one of three categories:

The client needs to solve an existing problem.

The client wants to exploit a business opportunity.

It is a business directive.

Once the answer and the reason for the database are known, it becomes easier to identify the key stakeholders we need to communicate with and request information from to determine the requirements. It is at this stage that models, methods, and techniques for elucidating requirements will be introduced. Different models, or frameworks have been proposed to identify requirements. For the purpose of demonstrating one model in action, we are concentrating on the PIECES problem-solving framework proposed by James Wetherbe in 1994.

The PIECES framework

This problem-solving framework is useful for categorising user requirements. It can also help us identify problems (or potential problems) relating to a project or system which may have been otherwise overlooked. Using this framework, problems can be identified which can lead to solutions, new opportunities can be exploited, and directives (new requirements set by management) can be fulfilled.

Each of the categories for PIECES is discussed below where we look at this framework from the perspective of identifying user requirements. **PIECES** is an acronym for:

P	Performance	The purpose of this category is to identify the need to improve performance in a way that it can be quantified. This covers issues and considerations relating to system performance. How will the system need to perform for the user? What will be the expected throughput? How fast will the system be expected to perform, that is, what are the expected response times?
I	Information	This category relates to the need to improve information and data gathering processes and analysis relevant to the system. It looks at collecting accurate information with respect to input data, program outputs and data storage. What information will be input into the system? What are most appropriate / efficient input forms and layouts? How does the output need to be presented? What data needs to be stored? What input and output devices are required?
E	Economy	This category looks at improving the economic side of the project by controlling and reducing expenditure to maximise profits. Here we identify issues and considerations relating to initial outlay, operational costs, and perceived financial benefits of the system. Relevant questions include: What is the budget of the system? What are some of the anticipated cost savings? Are there current manual activities where an automated system may generate cost savings?
C	Control	This category aims to improve system security by enabling the necessary controls. Issues and considerations for system security and data entry questions include: What accounting controls are required? What input and/or output controls are required? What security is needed for the system?
E	Efficiency	Issues and considerations relating to method correctness and appropriateness. How can current operations be improved by the system? Will tasks be more efficiently completed with the introduction of the new system? What value will the system be to the working environment?
S	Services	The purpose of this category is to improve the quality of the service the organisation delivers to customers and other stakeholders. Issues and considerations relating to the functional requirements of the system questions include: What does the system need to do to solve the problem? What processes does the system need to perform? Will the system be easy to use and maintain? What ongoing support, training, and maintenance will be required?

Reading

For a more in-depth understanding of the PIECES problem-solving framework, read this research paper: PIECES Framework and Importance Performance Analysis Method to Evaluate the Implementation of Information Systems from:

- E3S Web of Conferences – ICENIS 2020[3]
- Using the PIECES Framework[4]

The **PIECES framework** checklist is an invaluable tool to use to prepare for your first client requirements gathering meeting. Meeting preparation is the key for a successful and productive meeting. Plan the meeting carefully considering what you want to achieve from it. A good starting point is to look at the PIECES checklist and prepare a list of questions relevant to the project.

The checklist below (from CSU) demonstrates a sample of categorised questions to ask from the perspective of problem identification and user-needs analysis.

The purpose of the checklist is to identify what the client needs in terms of solving a problem with an existing database, exploit some unique opportunities, or follow a business directive.

Weblink

California State University has its own example.

The PIECES Problem-Solving Framework and Checklist[5]

Research

Now search the internet for another example.

[3] https://www.e3s-conferences.org/articles/e3sconf/pdf/2020/62/e3sconf_icenis2020_15007.pdf

[4] https://www.academia.edu/9107585/Using_the_PIECES_framework

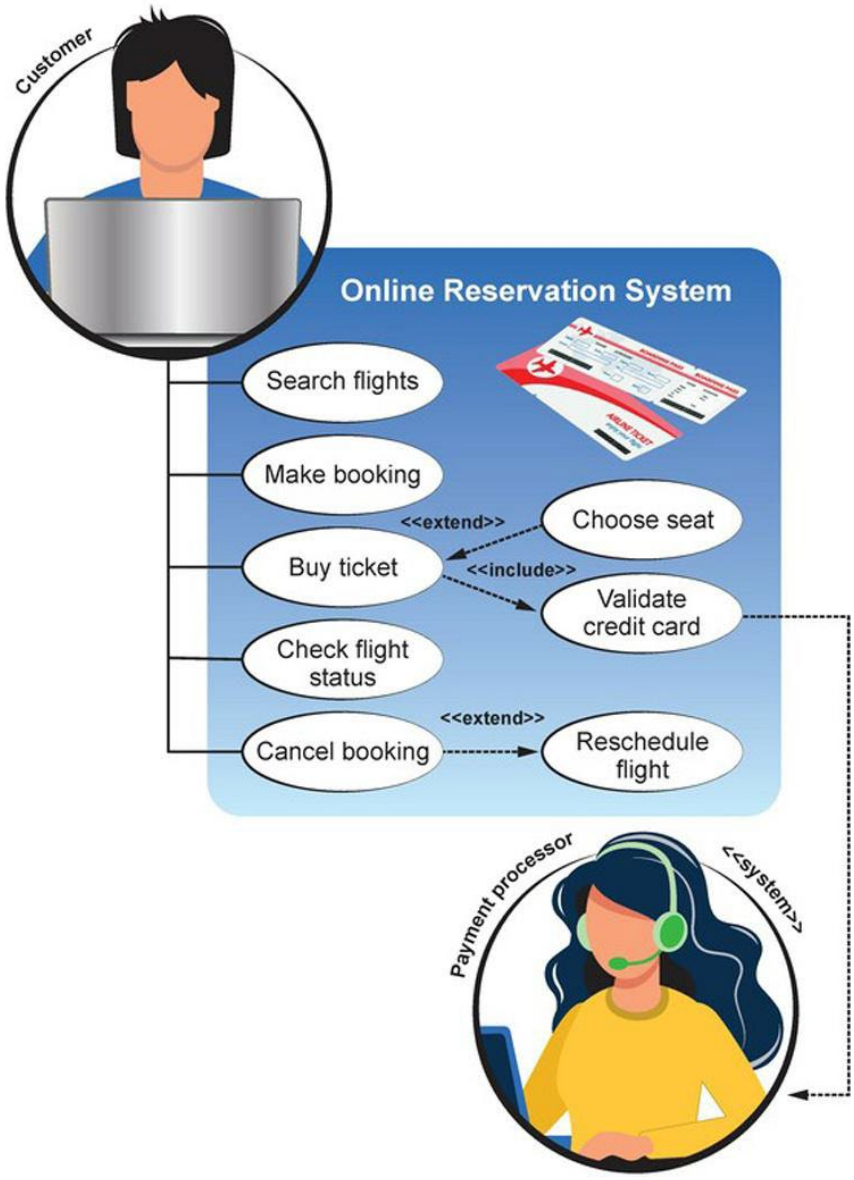
[5] https://www.csus.edu/indiv/m/merchants/pieces.pdf

Use case diagrams

A **use case diagram** is one of several types of **UML (Unified Modelling Language)** diagrams. UML is a graphical object-oriented language commonly used in systems analysis and design. It is a language used to describe models of real-world systems and processes in ways that are meaningful and easily understood.

Use case diagrams help the designer and key stakeholders understand the steps involved in particular system processes. They present these processes graphically at a conceptual high level. Consequently, these diagrams are ideal when consulting with management who tend to avoid detail and like information that is both meaningful and succinct. As an example, the figure below displays a use case diagram that describes the processes involved in using or interacting with an online reservation system.

Here is an example of a use case diagram and its separate elements.



385414132 / Golden Sikorka / shutterstock.com, modified by TAFE Queensland

USE CASE Diagram Notation Symbols

The symbols used in the use case diagram are

- actors
- use case
- system
- inclusions
- extensions

Weblink

See How to write effective Use Case Diagrams[6]

Video

Watch these two videos to help you understand more. They will play one after the other but if you want to watch them at separate times, use the playlist menu in the top right corner of the video.

The first video below presents a comical look (KH-3.0, 2020) at client/vendor interviews where stakeholders are trying to understand the system requirements, and each other. Perhaps you may relate.

The second video from BA-Experts (2021) is an introduction to use cases.

ict50220-ictdbs506 - Use case diagrams[7]

by KH-3.0 (7:37 mins) – hosted on YouTube (2020), BA-EXPERTS (8:21 mins) – hosted on YouTube (2021)

[6] https://www.visual-paradigm.com/tutorials/writingeffectiveusecase.jsp

[7] https://www.youtube.com/embed/videoseries?list=PLr24-CFnxcCwTsuu-BTMJRecK_ZOA_9-A

Key deliverables for Phase One

The key deliverables for this phase are

- the **statement of user requirements** or requirement document
- a **use case diagram/s**.

These documents contain the input that will be used in phase two to continue the process of designing a database.

Key stakeholders will analyse the deliverables produced to-date and determine whether it is still feasible to proceed.

Feasibility is based on three aspects:

- Operational feasibility - Will the new system operate in a way that is compatible with the way that the organisation operates?
- Technical feasibility - Will it be technically possible to achieve what is being asked?
- Financial feasibility - Will the new system fit within the allocated budget?

At this point, alternative options may be investigated.

Alternatives may involve 'tweaking' or expanding the existing system rather than replacing it completely.

Stakeholders have looked at the deliverables produced and have begun to question:

- "Why do we process orders this way?
- “Wouldn't it be more efficient to instead do it this way?"

This could lead to changes in business processes which will lead to changes to the proposed statement of requirements and use cases.

If stakeholders agree and the project is deemed to be feasible, then phase two commences – the conceptual data model will be designed in this phase.

Discussion

In groups of 2 to 3, review a given scenario and collaboratively create a list of 50 questions using one of the methods to identify user needs. A representative from each group should present to the class a summary of questions gathered.

Topic summary and review

In this topic you have been introduced to **relational databases** and associated **database management system (DBMS)** software. We have looked at a database project life cycle that included six phases - requirements collection and analysis, conceptual phase, choice of DBMS, logical design, physical design, and implementation.

This topic has primarily focused on the first phase – that of identifying user requirements. You have learnt about the **PIECES problem-solving framework** and its corresponding checklist. Finally, you have been provided with an introduction to **UML** use case diagrams.

To gain a deeper understanding of the concepts introduced in this topic a number of links to external resources, some relevant readings and a video have been provided.

In the next topic we introduce the second phase of the database life cycle - Conceptual Design.

Self-check - Fundamentals of database design and Phase 1

Complete the questions below.

What does UML stand for?

.....

In a use case diagram, what is not part of the system, but interacts with it?

- ☐ inclusions
- ☐ use case
- ☐ actor

What is the PIECES framework used to do? Check the four that apply.

- ☐ identifying the performance expectations of the database project
- ☐ identifying the expectations of the client's customers
- ☐ identifying the data types that will need to be included in the database
- ☐ identifying any security concerns and therefore any security measures that might be needed
- ☐ helping the client's whole business run as smoothly as possible
- ☐ improving the customer interface for the client
- ☐ improving the quality of service that the client delivers to its customers

What is the best data type suited to storing monetary values?

.....

Complete this sentence about the key outcomes. Choose from the following Words:

- a statement of user requirements
- a list of budget constraints
- a list of stakeholders and their expectations
- use case diagrams
- a mock-up of the whole project
- staffing requirements

The key deliverables for phase one are …….. and .....

Check your answers at the end of this Learner Guide

Phase 2 - Conceptual Database Design

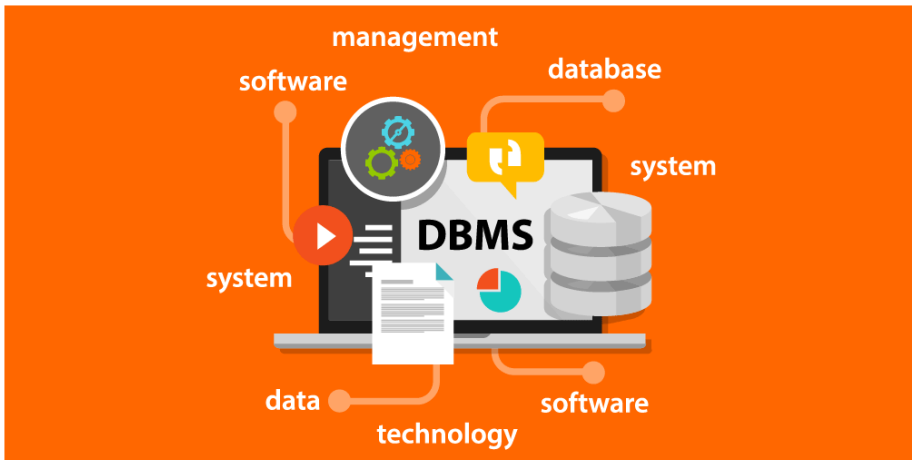
Introduction

This topic introduces the second phase to database design – the Conceptual Design. The purpose of this phase is to conceptualise our database design in an **entity relationship diagram (ERD)**. This is a high-level abstract representation and avoids too much detail but will be the starting point to design the database structure. During this phase we produce a conceptual model (ERD or schema) based on the information from the previous phase; requirements, collection, and analysis.

Objectives

By completing the topic it is expected that you will be able to:

- understand the conceptual data model purpose
- understand and know how to create an ERD
- understand entity and attributes concepts within an ERD
- understand primary and foreign keys and how to work with them
- understand cardinality of relationships.



395830441 / Bakhtiar Zein / shutterstock.com. Modified by TAFE Queensland.

What is a conceptual data model?

What is a data model?

A data model is a plan of the database, or blueprint which describes the system that will be built. All stakeholders can refer to the model and gain a collective understanding about the proposed system. It should be able to provide answers to all the questions raised by stakeholders about how the system will operate. If this is not the case, then the model will need to be revised. Data model design is an iterative process that should be repeated until all questions relating to the proposed system have been answered.

290589185 / Monkey Business Images / Shutterstock.com

In this phase we examine the database requirements and produce high-level specifications for these requirements.

The design in this phase is not related to any specific DBMS. We intentionally avoid making the model dependent on any specific DBMS. The notation used within the model must also be independent of any specific DBMS.

ERD Notation

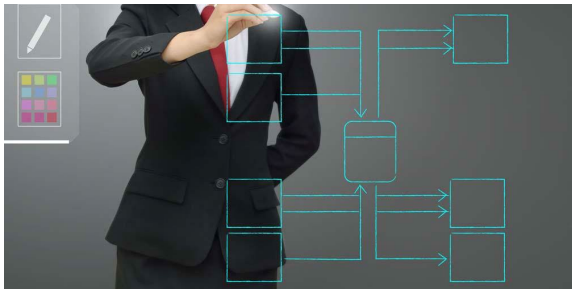
Weblink

Diagrams.net[8] is an open-source technology stack for building diagramming applications, and the world's most widely used browser-based end-user diagramming software.

Read more About diagrams.net[9]

> *Our mission statement is to provide free, high quality diagramming software for everyone.*
>
> Copied Under s113P, https://www.diagrams.net/about, 03/02/2022

The notations below are just a sample, but they represent the basic building block for an ERD. The diagrams in the table below and the ERD have been created using the online drawing tools on diagrams.net.



178242635 / Singkham / shutterstock.com
Modified by TAFE Queensland.

Recall

ERD – an Entity Relationship Diagram in a database design

Reading

Learn the ERD notations from this table[10], then do the activity that follows to check your understanding. (PDF 116 KB) used at TAFE Queensland. See a copy next page.

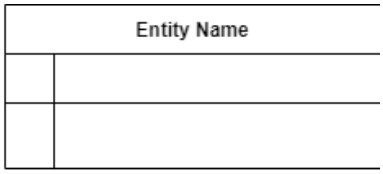
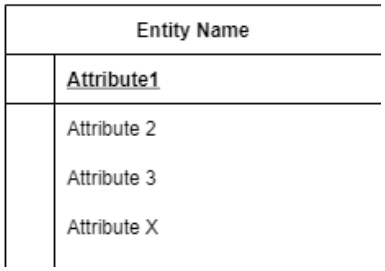
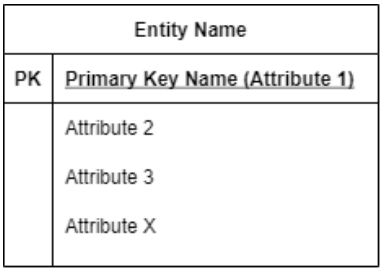
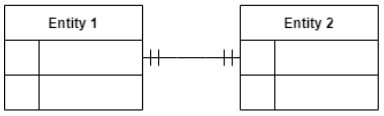
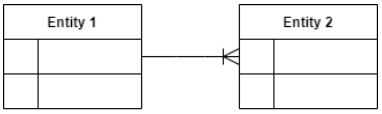
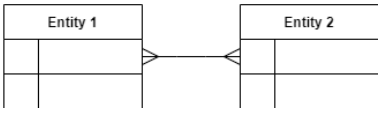
[8] https://www.diagrams.net/

[9] https://www.diagrams.net/about

10 https://connect.tafeqld.edu.au/content/enforced/401047-MO_ICT50220_ICTDBS506_TQM_02/ict50220-ictdbs506/media/ICTDBS506_ERD_notation_table_LHO_TQM_v1.pdf?_&d2lSessionVal=dv6RXOKjVX19zwlIXBPdousqy&ou=401047

ERD notation table

ICTDBS506 - Design databases
ICT50220 - Diploma of Information Technology

Notation Symbol Name	ERD Notation
Entity An entity is an object about which the database needs to record information. For example, in a school database student and teacher are entities. Entities may become tables after normalisation.	
Attributes Attributes are the data that we record about entities. For example, for the entity student we need to record Student Number, Student Name, and Student Surname and so on.	
Primary Key The primary key is the key that uniquely identifies a table. The value of the primary key cannot be repeated in another row of the table. For example, no two students can have the same Student Number.	
Relations – Cardinality Ratio - The term used to describe the relationship between one table and another table is called **cardinality**. The notation is known as fork or crow's feet	
One-to-One (1:1) This identifies the degree of participation of one entity into another entity. For example, if entity one is Doctor and entity two is Patient. It will not make a good business sense to have one doctor treating only 1 patient and 1 patient being treated by only one doctor.	
One-to-Many (Many-to-One the reverse) (1:N or (N:1) If we continue with the Doctor- Patient example, having one doctor visiting many patients is an improvement but we still have one patient being treated by only one doctor.	
Many-to-Many (N:M) Continuing with the Doctor- Patient example, a N:M relation allows a doctor to treat multiple patients and a patient to be treated by multiple doctors – while this is good for the hospital is not correct in terms of database design. The solution will be provided in later topics. At this stage understanding of the cardinality ratio is sufficient.	

Self-check - ERD notation.

Match the symbol names to their correct ERD notation.

Answers:

entity, attribute; primary key; cardinality ratio; fork or crow's feet ; 1:1; 1:N or N:1; M:N

ERD notation	Notation symbol name
This ratio is used to identify the degree of participation between one entity and multiple other entities, and the reverse. E.g. one doctor treats many patients, and one patient is treated by many doctors.	
This is the object about which the database needs to record information. They may become tables.	
This uniquely identifies a table. So the value of this cannot be repeated in another row of the table.	
This is the term used to describe the relationship between one table and another table.	
This term describes the notation symbol between tables.	
This ratio is used when one entity relates to multiple other entities. E.g. one doctor treats many patients	
This refers to the data that we record about entities. E.g. name, student number	
This ratio indicates the degree of participation of one entity to another entity. E.g. doctor to patient	

Check your answers at the end of this Learner Guide

Entity relationship diagram (ERD) draft

Note

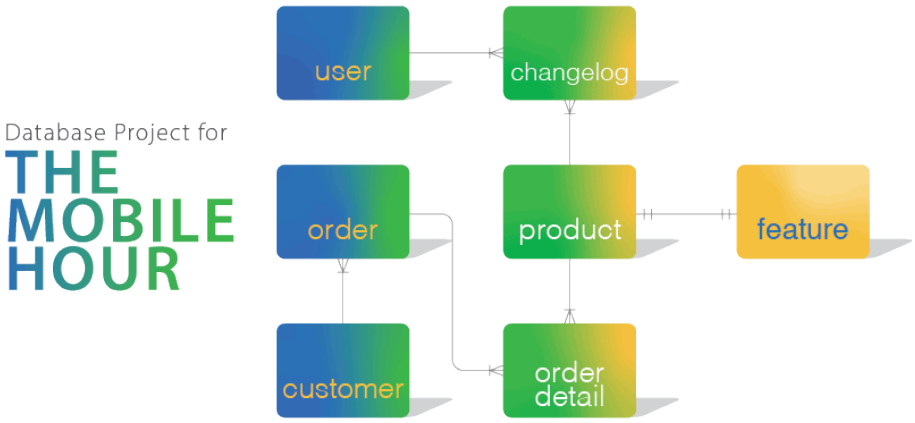
In this phase, Conceptual Database Design, we will refer to the objects that the database needs to record information about as **entities** and the data that will be recorded about each entity as **attributes**.

Later in Phase 4: Logical Database Design, after the ERD has been normalised, we will call entities **tables** and attributes **columns**. This distinction is made to keep a separation between the conceptual design and the logical design stages after a specific DBMS has been selected.

The conceptual data model provides a high-level of detail. It is a simple and readable diagram, primarily designed for business stakeholders, that shows all entities and the relationship between these entities expressed as **cardinality ratio**. The below model is a conceptual example of an ERD diagram, for "The Mobile Hour" a fictitious online shop.

247822480 / ideaweb / shutterstock.com

Example of First Draft ERD for The Mobile Hour



Cardinality ratio shown between

- user and changelog
- changelog and product
- product and order detail
- order detail and order
- order and customer

And the relationship between product and feature has a different relationship

Note

At this stage we are expecting that you can create a simple ERD draft. More details to refine the ERD will be provided in the next three topics.

The key deliverable for this phase is the **conceptual data model.** Once the design team has produced a model (based on deliverables obtained from the first phase), they submit it to the client (stakeholders) for review and feedback.

Again, this is an iterative process, and some stakeholders may request several alterations and additions be made. When feedback has been received, the design team makes any requested changes and proceeds to the next phase.

In consultation with the key stakeholders of the project, the design team now have the following deliverables:

- User Requirements Statement or Requirement Document
- Use case diagram(s)
- Conceptual data model (in the form of a draft ER diagram)

Based on the deliverables produced to date, we move on to the next phase, Choice of DBMS, where an appropriate DBMS will be chosen.

Topic summary and review

In this topic you have learnt about the process involved in creating a conceptual model derived from the communication with key stakeholders, the requirement document, and supported by a use case diagram. The conceptual model has been presented as a draft ERD that displays the entities and the relationship between entities denoted by the cardinality ratio.

To gain a deeper understanding of the concepts introduced in this topic a number of links to external resources and a video have been provided.

Weblink

For more information and examples of ERD diagrams and diagramming tools visit:

- Tutorials Point – ERD Model[11]
- Guru99 - ERD Model: DBMS Example[12]
- Visual Paradigm – How to Model and ERD[13]

Video

View the video (Baldazzi 2013) for more training on

Entity Relationship Diagram (ERD) Training[14]

by Gina Baldazzi (15:03 mins) – hosted on YouTube (2013)

[11] https://www.tutorialspoint.com/dbms/er_diagram_representation.htm

[12] https://www.guru99.com/er-diagram-tutorial-dbms.html

[13] https://www.visual-paradigm.com/tutorials/how-to-model-relational-database-with-erd.jsp

[14] https://www.youtube.com/embed/-fQ-bRllhXc

Discussion

1. Research and describe the processes involved in creating a conceptual data model. Discuss your findings in the classroom.
2. Draw a first draft ERD for a local clinic that employs 7 doctors of different specialisations, 2 nurses, and 3 receptionists. The client wants to record:
   - employee's details
   - patients' personal details
   - patients' medical details and details of treatments received at the clinic
   - a patient/doctor visits record
   - appointment details (who, when etc.).

Self-check - Conceptual Database Design - Phase 2

Check your understanding of this topic by completing the following quiz.

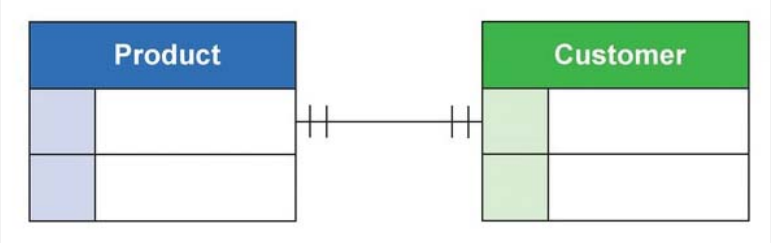
What is the term used to describe how an entity relates to another entity?

.....

What describing word is used for the data model that displays entities and their relationships?

.....

Look at the ERD sample presented below. It represents two entities: Product and Customer. What does the cardinality ratio notation describe?



- ☐ that many products can be bought by one customer?
- ☐ that one product can be bought by one customer and that one customer can buy one product?
- ☐ that one product can be stocked by many stores?
- ☐ that many products can be bought by many customers?

What acronym is used for the most relevant diagram notation to show relationships between entities?

.....

What is the key deliverable produced at the end of phase two?

- ☐ the use case diagram
- ☐ the database
- ☐ the conceptual data model
- ☐ the physical data

Check your answers at the end of this Learner Guide

Phase 3 – Choice of Database Management System (DMBS)

Introduction

This is phase three of the design process. Once the DBMS has been chosen the remaining two design phases will be dependent on the specific features of the database software selected.

In this topic we will look at several factors influencing the decision of choosing a DBMS. We will also focus on some of the technical considerations, with an emphasis on web-based technologies.

Objectives

By completing the topic it is expected that you will be able to:

- appreciate the importance of understanding the requirements of a database system
- understanding relational and non-relational database interfaces in order to make a well-informed decision
- understand the technical considerations of choosing a DBMS
- understand the economic considerations of choosing a DBMS
- understand the organisational considerations of choosing a DBMS.

Compliance, scalability and security

In this competency unit we are focused on **relational** databases but in the real world today NoSQL database interfaces are gaining popularity, however, the NoSQL approach is not an alternative to all SQL databases if data need to be structured and require **ACID** (atomicity, consistency, isolation, and durability) **compliance**. The ability to query data fast and with integrity is also a characteristic of SQL databases. Although NoSQL interfaces offer more flexibility and scalability (horizontally) they are not as strong in query efficiency. In terms of scalability, SQL scales vertically while NoSQL is better at horizontal **scalability**.

Open source versus proprietorship database software also ranks high in choosing a DMBS. Open-source databases (e.g. MariaDB) are appealing cost-effective solutions as the code is free and can be modified, while proprietary databases (e.g. Oracle and IBM SQL) are closed source software and come with a licensing price tag.

Database server and data security is also high on the agenda of many organisations in terms of choosing a DBMS. Database security includes protecting data from unauthorised access and/or theft.

Weblink

Check the following weblinks to learn more about Compliance[15] and Scalability[16].

290445947 / deepadesigns / shutterstock.com

[15] https://mariadb.com/resources/blog/acid-compliance-what-it-means-and-why-you-should-care/

[16] https://www.thorntech.com/sql-vs-nosql/

Main factors driving the choice of a DBMS

There are three main factors driving the choice of a DBMS – technical, economic and organisational.

Learn more about the important considerations when choosing a DMBS.

DBMS considerations

Technical considerations

Explanation - Multiple technical factors should be considered in relation to the functional requirements already identified in phase one. Each of the questions below can help clarify the technical requirements. Ideally, a technical subject matter expert should be part of the project team to provide advice on best fit based on the identified requirements.

274711127 / Aha-Soft / Shutterstock.com. Modified by TAFE Queensland.

Questions / Areas of concern

- What DBMS software should be purchased?
- How reliable is the hosting provider?
- Do they have an uptime guarantee?
- What type of security needs to be implemented for the server?
- Where will the DBMS be hosted in the cloud?
- Will there be a firewall installed to protect the DBMS?
- What are the backup requirements of the client?
- What will be the technical specifications of the server?
- Where will the DBMS be located?
- Will the DBMS be continually monitored?
- If the DBMS is to be hosted in-house, are the in-house people trained to satisfactorily manage the DBMS, considering the aspects mentioned above?
- Will the database be connected to a website?

Economic considerations

Explanation - From an economic perspective, the cost of acquiring and implementing a DBMS is significant and must be analysed in terms of suitability in relation to the identified user requirements.

Although a cost-benefit analysis can be performed to quantify costs, it is difficult to measure intangible benefits such as improved usability and increased speed in accessing information.

Other factors worth considering when selecting a DBMS are the quality of vendor service and support, which is guaranteed with proprietary databases.

278754269 / Aha-Soft / Shutterstock.com. Modified by TAFE Queensland

Questions / Areas of concern

- Relocating cost (if required)
- Software acquisition costs
- Hardware acquisition costs (if necessary)
- Maintenance costs
- Creation and conversion (from existing system) costs
- Personnel costs
- Training costs
- Ongoing operating costs

Organisational considerations

Explanation - As we have seen, there may be several factors influencing an organisation to implement a new DBMS or upgrade an existing DBMS.

149241191 / Blablo101 / Shutterstock.com. Modified by TAFE Queensland.

Questions / Areas of concern

- Is the data becoming increasingly complex?
- Is there a quantity of redundant information within the system due to previously bad design?
- Is the data volume growing?
- Is there an increasing frequency of ad hoc requests?
- Is there a need for greater control?
- Will there be a change in business processes?
- Will the database need to reflect a change management related issue (such as the organisation downsizing, or new acquisition and merger)?

Weblink

Cloud-based offerings are becoming more popular. IaaS, PaaS, and SaaS are cloud-based on-demand services offered by vendors that are scalable and cost-efficient, in that users can pay according to demand using a subscription-based method.

DBaaS – IaaS Database Cloud Comparison[17]

[17] https://www.navisite.com/blog/dbaas-or-iaas-database-cloud-comparison/

Topic summary and review

In this topic you have learnt about some of the considerations of choosing a new DBMS, including technical, economic, and organisational factors. An introduction of the characteristics and uses of non-relational databases has also been provided.

To gain a deeper understanding of the concepts introduced in this topic a number of links to external resources have been provided.

In the next topic we introduce the fourth phase of the database life cycle, Logical Database Design.

Research

Research Infrastructure-as-a-Service (IaaS) and Platform-as-a-Service (PaaS).

Discussion

Discuss your findings from this research in the classroom.

Self-check - Choice of DBMS - Phase 3

Complete the questions below to check your understanding.

What does the A stand for in ACID compliance?

.....

Which needs in the following list would make a SQL database preferable? Select all that apply.

- ☐ structured data
- ☐ horizontal scalability
- ☐ ACID compliance
- ☐ vertical scalability
- ☐ fast query
- ☐ flexibility

Which of the following needs would make a noSQL database preferable?

- ☐ structured data
- ☐ horizontal scalability
- ☐ ACID compliance
- ☐ vertical scalability
- ☐ fast query
- ☐ flexibility

Complete the sentences by choosing the correct word.

Words: *staffing; language; organisational*

Technical, economic, and …….. factors must be considered when choosing a DBMS.

Words: *technical, economic, organisational*

Backup requirements come under the category of ……... considerations.

Check your answers at the end of this Learner Guide

Phase 4 – Logical Database Design

Introduction

In this topic we discuss phase four of the database design process, Logical Database Design. At this point, we will have produced a statement of user requirements, several use case diagrams, and a basic entity relationship diagram (ERD) showing entities and relationships. We will also have analysed some of the technical, economic, and organisational considerations in choosing a database management system (DBMS).

We will now take our ER diagram designed in phase two and proceed to convert it from a conceptual data model to a logical data model.

Objectives

By completing the topic it is expected that you will be able to:

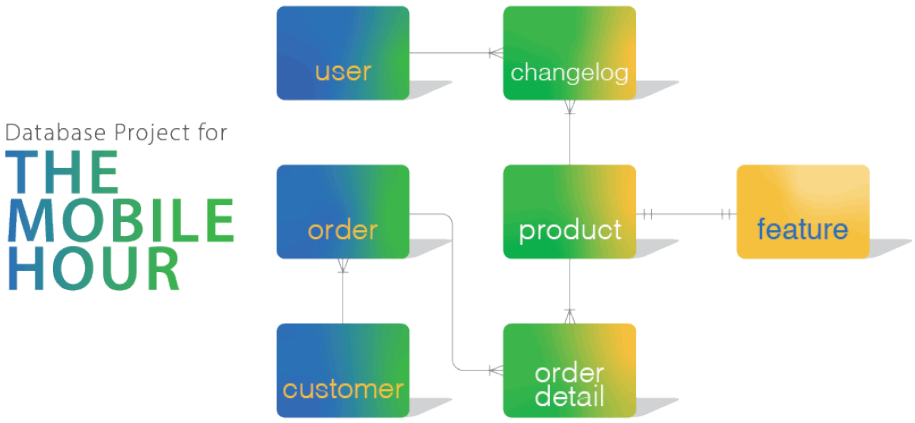
- understand the purpose of using primary and foreign keys
- understand how to set attributes for each of the model entities
- understand the importance of resolving many-to-many relationships
- understand the difference between identifying and non-identifying relationships
- understand the process of normalisation to the third normal form.

92126581 / Semisatch / shutterstock.com

Understanding relations and cardinality ratio

Once we have identified each of the entities required in our system, we can begin to think about what attributes each entity will require. If we look at the diagram presented below (originally included in phase two), we can see that the database for the online store The Mobile Hour will require seven entities – product, feature, order, order detail, customer, user, and changelog.

Database Project for The Mobile Hour



Database Project for The Mobile Hour

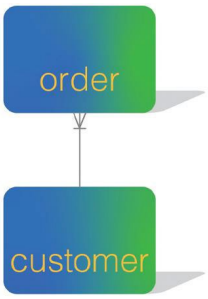
- user
- order
- customer
- changelog
- product
- order detail
- feature

What information do we need to store about each product? How much information do we need to retain about our customers? What information is stored in the changelog?

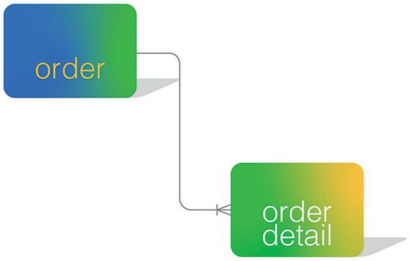
We should know the answers to these questions based on the research conducted in phase one, but for any questions raised we are unclear about, we need to return to the client for clarification.

Our original first draft ERD (in entity relationship diagram (ERD) draft page) started by identifying entities and the connections between them. From this diagram and the relation cardinality ratio we can infer the following holds true for The Mobile Hour store.

One customer (Customer ID) can place more than one order, but each order number refers to only one customer as indicated by the **one-to-many** relation.



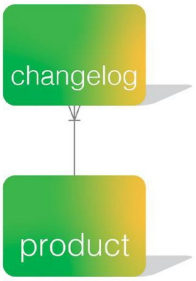
One order number appears many times in the order details entity, but each order details ID refers to only one order number as indicated by the **one-to-many** relation.



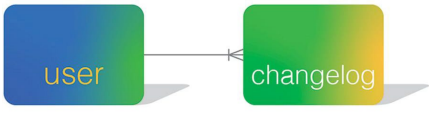
One product ID appears many times in the order details entity, but each order details ID refers to only one product ID as indicated by the **one-to-many** relation.

One product ID corresponds exactly one feature ID in the feature entity. This is a **one-to-one** relation. The client explained that due to the specialised features no feature is shared by two phones.

One product ID appears many times in the order details entity, but each changelog ID refers to only one product ID as indicated by the **one-to-many** relation.

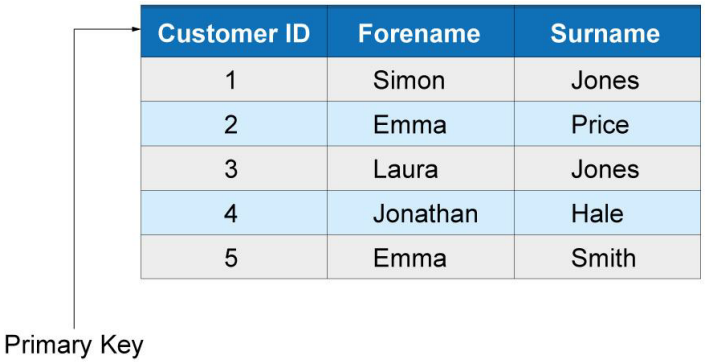


One user ID appears many times in the changelog entity, but each changelog ID refers to only one user ID as indicated by the **one-to-many** relation.



Each organisation sets their own data rules. The database analyst asks for the business rules and incorporates them into the ERD, then presents the diagram to the client for feedback and approval.

Assigning primary keys to entities



Primary Key

Customer ID	Forename	Surname
1	Simon	Jones
2	Emma	Price
3	Laura	Jones
4	Jonathan	Hale
5	Emma	Smith

Next, we identify the primary keys.

A primary key is a column that will uniquely identify each record that is stored in a table. Referring to the ER diagram (In Understanding relations and cardinality ratio) you can see that the productID attribute in the product entity is a candidate. This ID will be used to uniquely identify all products in our database.

In some situations, there may be a need to define more than one column as a primary key. Using just one column in some instances may not uniquely identify a particular row, in which case two (or more) columns are set as the primary key to establish uniqueness. This is referred to as a **composite primary key**.

Weblink

However, a single column primary key is the most common type and will be used in most cases. Composite Primary Key[18]

[18] https://www.geeksforgeeks.org/composite-key-in-sql/

Adding attributes to entities

The attributes that are added to the entities are dictated by the business needs and the data operations that the organisation carries out.

Example

For the customer entity the organisation might need to record

- name
- date of birth
- contact details
- tax file number
- and so on.

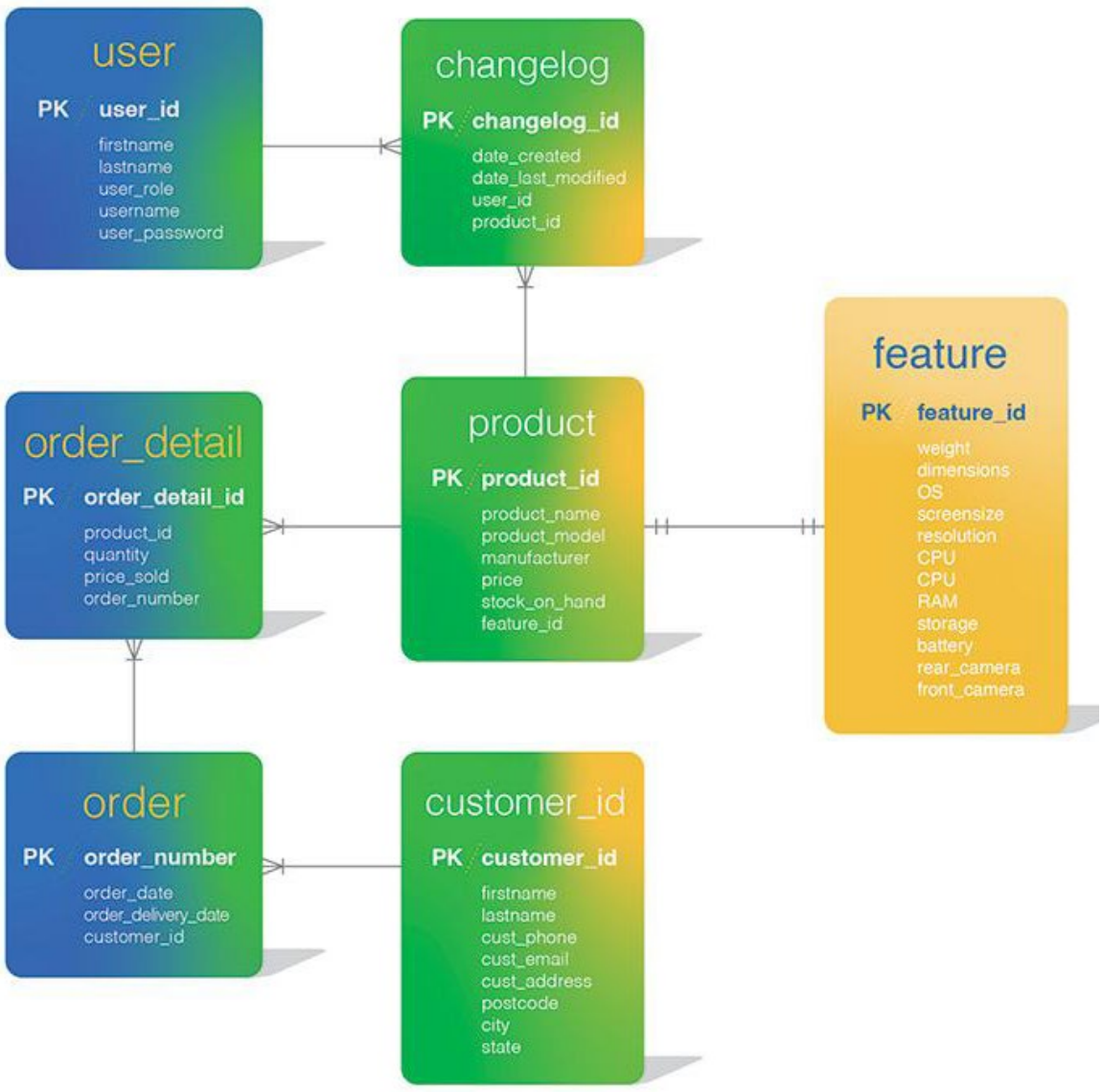
Some of that data may be legal business requirements like the tax file number. The inclusion or exclusion of attributes needs to be discussed with the stakeholders.

368997863 / rawpixel / Shutterstock.com

Adding foreign keys to entities

Foreign keys are used to establish a relationship between entities. The primary key, in one entity, is added to a linked adjacent entity as the foreign key. This will ensure that later, when the database is implemented, we can use JOIN statements to query multiple tables in a single query. In fact, we use the foreign key to join two tables.

The diagram below displays a revised version of the draft ERD. It includes entities, attributes, relations (cardinality ratio) and primary and foreign keys:



user

- **PK** user_id
- firstname
- lastname
- user_role
- username
- user_password

changelog

- **PK** changelog_id
- date_created
- date_last_modified
- user_id
- product_id

product

- **PK** oriduct_id
- product_name
- product_model
- manufacture
- price
- stock_on_hand
- feature_id

order_detail

- **PK** order_detail_id
- product_id
- quantity
- price_sold
- order_number

order

- **PK** order_number
- order_date
- order_delivery_date
- customer_id

customer_id

- **PK** customer_id
- firstname
- lastname
- cust_phone
- cust_email
- cust_address
- postcode
- city
- state

feature

- **PK** feature_id
- weigh
- dimensions
- OS
- screensize
- resolution
- CPU
- RAM
- storage
- battery
- rear_camera
- front_camera

Rules for resolving many-to-many relationships

Can we have many-to-many relationships?

The answer is no. If we allow a many-to-many relation, it means that the primary key in both tables is compromised (repeated). A relational database is strictly structured and has many rules to ensure the integrity of the data within the database.

Important

A primary key (single or composite) value can **never** be repeated in the column. Foreign key values can be repeated as many times as necessary.

Before we start resolving the many-to-many problem, we need to understand how foreign keys are assigned. The general instruction is that the primary key of one entity must be added as an extra attribute to the entity it is related to. The problem is, which primary key goes to which table? You need to know the foreign key assignment rules.

Learn the three foreign key assignment rules here.

In a one-to-one relation, the foreign key can be applied to either entity. It will not make the primary key repeat.

In a one-to-many relation, the primary key of the entity with the one (ratio) must go to the table with the many (ratio) Why? Because if we do it the other way around we repeat the primary key in the first entity.

In a many-to-many relation, a third table/entity must be created to destroy the many-to-many and avoid primary repeats. In consequence, between the three tables we will have safe one-to-many relations. In which case we need to apply rule number 2.

98796455 / ThreeCups / Shutterstock.com. Modified by TAFE Queensland.

Next we practise applying rules to resolve many-to-many relationships.

Examples of resolving many-to-many relationships

Remember Rule 2.

In a one-to-many relation, the primary key of the entity with the one (ratio) must go to the table with the many (ratio).

<table>
<tr><th>Correct or incorrect assignment of a foreign key?</th><th>Explanation</th><th>Tables</th></tr>
<tr>
<td>  </td>
<td>This example correctly displays a one-to-many relation between Doctor and Patient (1:N) – the one (1) is on the Doctor side and the many (N) is on the Patient side.

The Doctor_ID has been correctly added to the Patient table and both tables maintain their respective primary keys unique (no repeats).</td>
<td>
<table>
<tr><th colspan="3">Doctor</th></tr>
<tr><th>Doctor_ID (PK)</th><th>Name</th><th>...</th></tr>
<tr><td>D1</td><td></td><td></td></tr>
<tr><td>D2</td><td></td><td></td></tr>
</table>
<table>
<tr><th colspan="4">Patient</th></tr>
<tr><th>Doctor_ID (PK)</th><th>Name</th><th>...</th><th>Doctor_ID (FK)</th></tr>
<tr><td>P1</td><td></td><td></td><td>D1</td></tr>
<tr><td>P2</td><td></td><td></td><td>D1</td></tr>
<tr><td>P3</td><td></td><td></td><td>D2</td></tr>
<tr><td>P4</td><td></td><td></td><td>D1</td></tr>
</table>
</td>
</tr>
<tr>
<td>  </td>
<td>Adding the Patient _D to the Doctor table is incorrect.

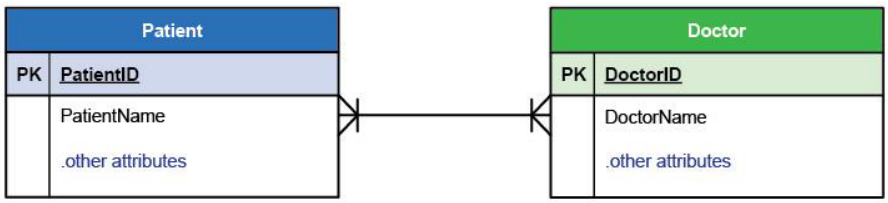
The Doctor primary key has been compromised (repeated).

Remember the rules!</td>
<td>
<table>
<tr><th colspan="4">Doctor</th></tr>
<tr><th>Doctor_ID (PK)</th><th>Name</th><th>...</th><th>PatientID (FK)</th></tr>
<tr><td>D1</td><td></td><td></td><td>P1</td></tr>
<tr><td>D1</td><td></td><td></td><td>P1</td></tr>
<tr><td>D1</td><td></td><td></td><td>P1</td></tr>
<tr><td>D2</td><td></td><td></td><td>P2</td></tr>
</table>
<table>
<tr><th colspan="3">Patient</th></tr>
<tr><th>Patient_ID (PK)</th><th>Name</th><th>...</th></tr>
<tr><td>P1</td><td></td><td></td></tr>
<tr><td>P2</td><td></td><td></td></tr>
<tr><td>P3</td><td></td><td></td></tr>
<tr><td>P4</td><td></td><td></td></tr>
</table>

</td>
</tr>
</table>

Now let's use rule 3 to fix a many-to-many problem.

Before fixing the many-to-many problem:



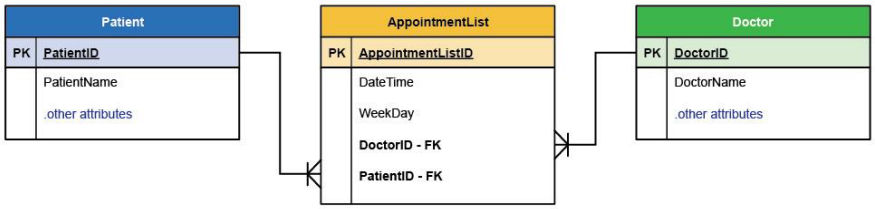
Patient

- **PK** PartientID
- PatientName
- .other attributes

Doctor

- **PK** DoctorID
- DoctorName
- .other attributes

After fixing the many-to-many problem by adding a third entity we have the following ERD adjustment:



Patient

- **PK** PartientID
- PatientName
- .other attributes

AppointmentList

- **PK** AppointmentListID
- DateTime
- WeekDay
- **DoctorID - FK**
- **PatientID - FK**

Doctor

- **PK** DoctorID
- DoctorName
- .other attributes

The addition of the third entity has resolved the many-to-many problem, now we have two one-to-many relations. The primary keys of the three (3) entities are safe and the repetitions will occur in the foreign keys, which is correct.

Recall

Rule 1: In a one-to-one relation, the foreign key can be applied to either entity. It will not make the primary key repeat.

Rule 2: In a one-to-many relation, the primary key of the entity with the one (ratio) must go to the table with the many (ratio) Why? Because if we do it the other way around we repeat the primary key in the first entity.

Rule 3: In a many-to-many relation, a third table/entity must be created to destroy the many-to-many and avoid primary repeats. In consequence, between the three tables we will have safe one-to-many relations. In which case we need to apply rule number 2.

From this section onwards, we will refer to entities as **tables** and to attributes as **columns**.

Normalisation before normalisation

Before we delve into the normalisation process it would be beneficial to know what can be done to simplify the process. Normalisation can be complex if the design of the database has not started considering some fundamentals of relational databases:

Fundamentals of relational databases

- Simplicity of design
- Efficiency of storage
- No redundancy or duplication of data
- Efficiency of queries

Steps to follow before normalisation.

Step 1 : Tables, columns, and primary keys names

The name of the table is important as it represents the building block of a relational database. It is the basic data structure of relational databases. If we need to store information about customers, then **customer** is an appropriate name for the table. Next the columns. A table should only have columns that represent or describe the table (table name). For example, customer name, last name, phone, email, address, DOB, tax file number and so on. Each column must be directly related to the primary key of the table e.g. the customer emails correspond to the specific customer identified by the primary key and the same is true for DOB, address and all the other attributes in the example. This is important because normalisation will remove any non-table related column.

The primary key of the table should take on the table name (within reason) to reduce confusion e.g., if the table name is **customer** and you use client_id as the primary, it would confuse database users - **customer_id** is a better choice.

Step 2 : Foreign keys

Foreign keys are added to the table to enable the linking between tables to perform multi-table join queries. That means that they are not covered under the rule of columns having to describe the table's name as they are just a linking tool and their inclusion on any table depends on the cardinality ratio rules mentioned above.

The foreign key in an entity refers to and always corresponds to an existing primary key in other entity and must fulfil two conditions:

- The foreign key column must have the same declaration as the primary key.
- A value in the foreign key column must be equal to a value in the primary key column or set to NULL.

This is called **referential integrity constraint**.

The attribute in the foreign key must have the same domain as the primary key that it references. A value of the foreign key is either equal to a value in the primary key of the referenced entity or is set to NULL.

Step 3 : Data redundancy and duplication

Relational databases forbid entering the same column into more than one table unless it is the foreign key. CustomerName will only be stored in the customer table and the same goes for all the other columns in the table (unless it is the primary key converted into foreign key in a second table).

In the next section we are introducing the formal normalisation process using first normal for (1NF), second normal form (2NF) and third normal form (3NF). A brief description will be provided on other normal forms beyond the 3NF.

Normalisation and normal forms

Normalisation is a systematic process of ensuring that our database will not contain redundant (duplicate) data. It also ensures that there will be no data anomalies (problems with missing or inconsistent data) after performing certain SQL queries involving INSERT, UPDATE, or DELETE.

The purpose behind the normalisation process is to ensure that we do not store more data than we need to, and also to ensure that our data is stored logically.

If updating the address details of a customer stored in our database, we should only need to update once – there should not be multiple instances in our database where we store the same information.

There are five progressive normal forms. We start with first normal form and progress on to second and then third. We rarely need to go beyond third normal form to fourth, or fifth form.

Important

Normalisation rules, or normal form, must be applied to every table. If a table has new columns added after being normalised, they need to be normalised again as the new columns may not comply with the normal forms.

397574752 / SFIO CRACHO / Shutterstock.com

How to apply normalisation rules to the tables

Learn how normalisation rules must be applied to every table and every reiteration of a table.

First Normal Form (1NF)

1NF Rules

For a table to be in the 1NF it must comply with the following rules.

✔ Rule 1: A primary key (PK) exists – meaning that there are no repeating values in the column

✖ Rule 2: All values in each column are atomic – single indivisible values

Can you find the problem in the following table using these rules?

Products			
ProductID	**ProductName**	**ProductDescription**	**Specs**
1	Samsung Universe 42	Latest addition to the Samsung family	Lithium batteries, Phantom grey colour, 280 grams, 16.7 x 8.8 x 3.4 cm, Android 10.0
2	Nokia Smart 10	A new era of mobile phones starts with Nokia Smart 10	Lithium polymer, Nordic Blue colour, 220 grams, 7.97 x 91 x16.89 cm, Android 11

There is an atomicity problem in the Specs column.

Explanation –

✔ The primary key exists and has no repeating values.

✖ The ‘Specs’ column has multiple values, not a single one. Values separated by commas cannot be individually accessed by the database in searches or queries. The database considers the whole entry a single value (a single feature).

✔ The product name is fine – even if it has more than one word it is still one single value, the name.

✔ The product description is also considered a single value, the description.

What is the solution? –

Split the table into two tables.

- Products
- Specs

Table 1:

Products		
Product ID	**ProductName**	**ProductDescription**
1	Samsung Universe 42	Latest addition to the Samsung family
2	Nokia Smart 10	A new era of mobile phones start with Nokia Smart 10

Table 2

Specs					
SpecsID	**Battery**	**Colour**	**Weight**	**Dimensions**	**OS**
1	Lithium Glass	Phantom Grey	280	16.7 x 8.8 x 3.4	Android 10.0
2	Lithium Polymer	Nordic Blue	220	7.97 x 91 x 16.89	Android 11

For a table to be in the 1NF it must comply with the following rules.

- Table 1: Products (ProductID (PK), ProductName, ProductDescription, SpecsID (FK))
- Table 2: Specs (SpecsID (PK), Battery, Colour, Weight, Dimensions, OS)

Second Normal Form (2NF)

2NF Rules

For a table to be in the 2NF it must comply with the following rules.

- Rule 1: The table is in the 1NF
- Rule 2: All columns, that are not part of the primary key, are fully functionally dependent on the primary key.

Important

2NF is only relevant if using a composite primary key, e.g., a primary key that is made up of two or more columns. If you are not using any composite primary keys, then proceed to 3NF.

Fully functionally dependent means that the columns that are not part of the primary key must NOT be uniquely identifiable from a subset (from a part) of the primary key. This is also known in the literature as partial dependency. Meaning that all non-key columns must be fully dependent of the primary key.

2NF

We are looking at a sample table with a composite primary key (OrderNo, ProductID).

Can you find the problem in this table by following these rules?

- The table is in the 1NF
- All columns, that are not part of the primary key, are fully functionally dependent on the primary key.

OrderNo	ProductID	DateTime	Quantity	PriceSold	CustomerID
1	314	2021-01-20 02:45:20	3	200	123
1	21	2021-01-20 02:45:20	14	20	123
1	177	2021-01-20 02:45:20	234	45	123
1	11	2021-01-20 02:45:20	11	60	123
2	21	2021-03-17 23:52:44	80	20	34
3	177	2021-01-27 20:01:20	25	45	123
3	11	2021-01-27 20:01:20	34	60	123
Composite primary Key		Non-key (PK) columns			

As you can see, OrderNo has repeating values and so does ProductID but the combination of the two keys is never repeated.

Now check the 2NF rules

✔ The table is in the 1NF

✘ All columns, that are not part of the primary key, are fully functionally dependent on the primary key.

Explanation -

Looking at the table we can see that Customer number 123 has placed an order (order 1) and has purchased 4 items on the same day. A few days later the same customer placed another order for 2 items.

The question to identify partial dependencies is – Is there a column that is more dependent on one column of the PK than on the other? The answer is, yes there are two, Quantity and PriceSold. Quantity is dependent on the order number (in a different order the customer may purchase a different quantity). PriceSold is more dependent on ProductNo (price changes on the product purchased not on the order number).

There is also redundancy, but this will be fixed in the 2NF solution.

Solution:

Split the table into two tables.

- Orders
- OrderLine

Table 1

Orders		
OrderNo	DateTime	CustomerID
1	2021-01-20 02:45:20	123
2	2021-03-17 23:52:44	34
3	2021-01-27 20:01:20	123

Table 2

OrderLine				
OrderLineNo	OrderNo (fk)	ProductID	Quantity	PriceSold
1	1	314	3	200
2	1	21	14	20
3	1	177	234	45
4	1	11	11	60
5	2	21	80	20
6	3	177	25	45
7	3	11	34	60

Primary keys in both tables have unique values and the only repetition is in the foreign key column, which is correct. Both tables now have single primary keys.

- **Table 1:** Orders(orderNo (PK), DateTime, Customer)
- **Table 2:** OrderLineNo (OrderLineNo (PK), ProductID, Quantity, PriceSold, OrderNo (FK))

Note

We are assuming a one-to-one relation as the client considered that no two mobile phones will have the same specs – they change every season!

Third normal form (3NF)

3NF Rules

For a table to be in the 3NF it must comply with the following rules.

- **Rule 1:** The table is in the 2NF
- **Rule 2:** There exists no transition dependency amongst the non-key columns.

No transition dependency means that all columns must be directly dependent on the primary key. To break the 3NF, we are looking at one non-key column being dependent on another non-key column.

Can you find the problem in this table by following these rules?

EmpID	EmpName	...	Address	Postcode	Suburb
1	Joyce		12 Cambridge Street	4497	Balonne
2	Alesha		88 Jacolite Street	4497	Balonne
3	Brayden		69 Devon Street	4497	Balonne
4	Rodney		18 Bayfield Street	3020	Alvion
5	Clyde		7 Glen William Road	3020	Sunshine
6	William		93 Edgewater Close	4163	Redland
7	Shanai		54 Tooraweenah Road	4497	Balonne
PK	Non-key (PK) columns				

✔ The table is in the 2NF

✖ There exists no transition dependency amongst the non-key columns.

Now check the 3NF rules

Explanation:

✔ The "…" indicate other columns like middle initials, surname, and email – these will not affect the normalisation (and there is only so much space on a page).

✖ In terms of dependency, we can say that the postcode depends on the suburb one lives (if I know the suburb, I can find the postcode) also, if we know the postcode we can identify the state – so there is **some** degree of dependency between these non-key columns

What is the solution?

Split the table into two tables.

- Employee
- PostalInfo

Table 1

Employee					
EmpID	EmpName	EmpSurname	EmpEmail	Address	PostalInfoID (FK)
1	Joyce	Tyson	jt@jt.com	12 Cambridge Street	1
2	Alesha	Wise	aw@aw.com	88 Jacolite Street	1
3	Brayden	Alcock	ba@ba.com	69 Devon Street	1
4	Rodney	Avalos	ra@ra.com	18 Bayfield Street	2
5	Clyde	Wall	cw@cw.com	7 Glen William Road	3
6	William	Mccoy	wm@wm.com	93 Edgewater Close	4
7	Shanai	Norris	sn@sn.com	54 Tooraweenah Road	1

Table 2

PostalInfo			
PostalInfoID (PK)	Postcode	Suburb	State
1	4497	Balonne	QLD
2	3020	Alvion	VIC
3	3020	Sunshine	VIC
4	4163	Redland	QLD

Note

Primary keys in both tables have unique values and the only repetition is in the foreign key column, which is correct. Both tables now have single primary keys.

The address must stay in the employee table or there will be no efficiency. If an organisation has 20 000 employees and they come from only 20 postcodes, we do not want 2 tables of 20 000 rows. In addition, the address is specific to the employee (like their name or DOB), the suburb, postcode and state are not.

- **Table 1:** Employee (EmpID (PK), EmpName, EmpSurname, EmpEmail, EmpAddress, PostalInfoID(FK)))
- **Table 2:** PostalInfo (PostalInfoID (PK), Postcode, Suburb, State)

Note

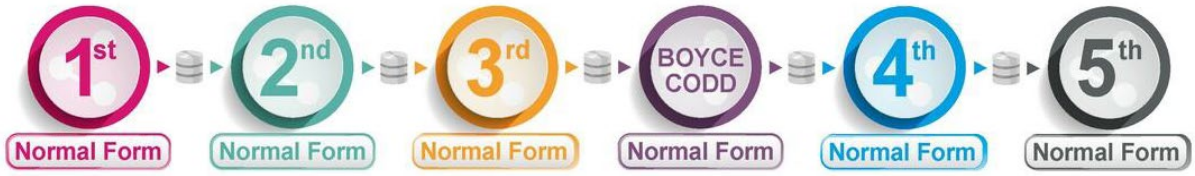
Foreign key for a **many-to-one** relation was added to the Employee table following the rules explained earlier.

Should we blindly follow all normalisation rules?

No – the database analyst will follow the rules as long as the rules provide efficiencies for the database.

Well done! You have now learned to apply the 1NF, 2NF and 3NF rules. They are also called normalisation rule or normal form rules.

Progressing beyond 3NF



Progressing beyond 3NF is beyond the scope of this unit. Too much decomposition (granularity) can result in unnecessary complexity in a relational database and can lead to performance issues. Why is this so? When we continue to decompose our data to atomic form (the point where our data cannot be broken down or divided any further), it means that we will continue to create additional tables to separate our data. The more tables our database has, the more complex our SQL queries become, as we must JOIN more tables to retrieve the data we are looking for. In most cases, this is an unnecessary complexity and results in inefficient database performance as our queries become far more complex and therefore take longer for the database to process and execute our queries.

- Boyce-Cood Normal Form (BCNF) - BCNF is a reinforcement of the 3NF (also called 3.5NF) rather than an independent normal form.
- Fourth Normal Form (4NF) or MultiValued Dependencies (MVD) - The 4NF was introduced by Ronald Fagin in 1977.
- Fifth Normal Form (5NF) or Projection/Join Normal Form (PJ/NF) - 5NF splits tables into the smallest possible pieces to eliminate all redundancy within a table.

Weblink

For more information on the difference between the fourth and fifth normal forms go to.
Difference between 4NF and 5NF[19]

[19] https://www.geeksforgeeks.org/difference-between-4nf-and-5nf/

Topic summary and review

In this topic you have learnt about the logical data model which is a model that is dependent on the chosen DBMS. We have discussed identifying attributes based on the user requirements gathered in phase one. We have also looked at primary keys, composite primary keys, and foreign keys. We explained the reasoning behind the need to resolve many-to-many relationships and the concept of normalisation was discussed.

Self-check - Logical database design and normalisation - Phase 4

Check your understanding of this topic by completing the following quiz.

Which normal form do we achieve when we decompose attributes containing multiple values?

- ☐ First normal form (1NF)
- ☐ Second normal form (2NF)
- ☐ Third normal form (3NF)
- ☐ One beyond 3NF

What primary key contains two or more columns?

.....

What is used to uniquely identify rows in a specific entity?

- ☐ foreign key
- ☐ decomposition key
- ☐ candidate key
- ☐ primary key

What term describes the systematic process of ensuring that our entities will not contain redundant or duplicate data?

.....

What situation occurs when the same attribute name and values are stored in different entities throughout the database?

.....

Check your answers at the end of this Learner Guide

You worked through an activity to see how to apply the 1NF, 2NF and 3NF rules. Examples were shown how to normalise to the third normal form (3NF). This process helps to prevent data redundancy.

Self-check - Normalisation

Match the rules into their correct normalisation (Normal Form) rules.

Answers: *NF1; NF2; NF3*

Normalisation form rules.	**Match the rule from the above answers**
Rule 1: A primary key (PK) exists – meaning that there are no repeating values in the column	
Rule 2 All values in each column are atomic – single indivisible values	
Rule 1: The table is in the 1NF	
Rule 2: All columns, that are not part of the primary key, are fully functionally dependent on the primary key.	
Rule 1: The table is in the 2NF	
Rule 2: There exists no transition dependency amongst the non-key columns.	

Check your answers at the end of this Learner Guide

Phase 5 - Physical Database Design

Introduction

In this topic we continue to develop our data model – converting the logical design model to a physical design model. Once the normalisation process is completed, the resulting ERD is final. This means that the entities in the final ERD will become the database tables and their attributes will become the table columns. During the normalisation process some entities may have been split into two tables, if dictated by the normal form rules. In this topic we introduce a **manual data dictionary**.

Lastly, we will again revisit the client's business requirements and apply any relevant constraints to our database accordingly. The aim is to maintain data integrity. Constraints place checks on the data being entered and ensure that data stored in our database is accurate and has integrity.

514021990 / LeoWolfert / shutterstock.com

Objectives

By completing this topic it is expected that you will be able to:

- understand differences between the logical data model and physical data model
- understand common data types specific to the chosen DBMS
- understand the importance of the data dictionary as the blueprint for the database
- revise and analyse business requirements and apply constraints.

The steps involved in converting our logical data model into a physical data model are as follows:

1. Complete the normalisation process up to the 3NF.
2. Create a manual data dictionary and assign data types and precision parameters to each column in each table.
3. Identify business constraints in the data dictionary for each column in which they need to be applied. These can be implemented with CHECK constraints or triggers.
4. Identify indexes in the data dictionary – if relevant. By default, the DBMS creates indexes on the primary and foreign keys.

Data types

Some data types are common to the most popular databases, while others will slightly differ. Refer to the documentation of the DBMS that you have chosen for implementation for a complete list of available data types. In the table below, the data types displayed are for MySQL. Only the most basic data types are listed to demonstrate some of the characteristics and ranges.

Notice the **signed** and **unsigned** keyword for integers. Signed data types accept positive and negative numbers. Unsigned data type integers only accept positive numbers.

Data type	Description
CHAR(size)	A FIXED length string from 0 to 255 characters
VARCHAR(size)	A VARIABLE length string from 0 to 65535 characters
TINYTEXT	String from 0 to 255 characters
TEXT(size)	String from 0 to 65,535 bytes
SMALLINT(size)	Signed – Range from -32768 to 32767 Unsigned – Range from 0 to 65535.
MEDIUMINT(size)	Signed – Range from -8388608 to 8388607 Unsigned Range from 0 to 16777215
INT(size)	Signed – Range from-2147483648 to 2147483647 Unsigned – Range from 0 to 4294967295
FLOAT(p) or DOUBLE	If p is from 25 to 53, the data type becomes DOUBLE()
DECIMAL(size, d)	An exact fixed-point number e.g., DECIMAL (10,2)
DATE	Format: YYYY-MM-DD. The supported range is from '1000-01-01' to '9999-12-31'

Weblink

For more detailed information about data types for MySQL and MySQL Server visit:

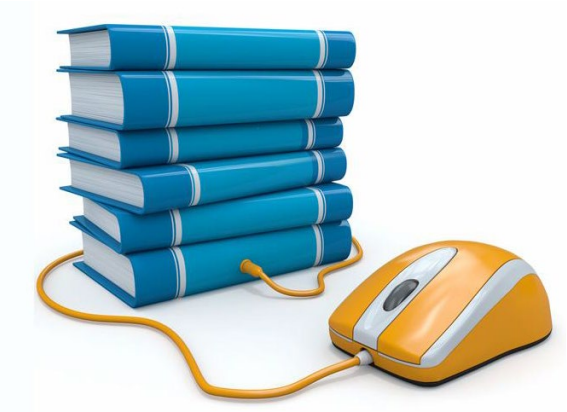
- MySQL Data Types[20] SQL Server Data Types[21]

[20] https://www.w3schools.com/mysql/mysql_datatypes.asp

[21] https://docs.microsoft.com/en-us/sql/t-sql/data-types/data-types-transact-sql?view=sql-server-ver15

Data dictionary

A **data dictionary** contains a description of database tables, column declarations and other procession parameters. In general, it contains:



114147685 / Maxx-Studio / Shutterstock.com. Modified by TAFE Queensland.

- Table and column definitions
- Data types, data length – space allocation
- Default values
- Key constraints
- Semantic constraints
- Comments and any other relevant data about the database structure
- Access rights for schemas on each of the objects
- Last updated and last accessed information about the object
- Any other database information

Here is a sample manual data dictionary created after normalisation to guide the database implementation.

DATA DICTIONARY for <<Database Name>>											
Table 1	Column	Data Type & Length	Signed Unsigned	Max/Min Values Or Range	Primary Key	Auto-Increment	Foreign Key	Required Field Null - Not Null	Unique Field Identified As Key	Business Data Constraints Data Validation Triggers	Comments

Example

Go to Appendix A at the end of this learner guide to see a larger landscape version of the Data Dictionary table above.

However, most DBMS can generate a data dictionary from the database schema. It is important to keep the data dictionary up-to-date, especially if multiple users work on the same database. The comments column can be used to keep track of changes e.g. what the changes were, who made them, and when they were made.

Weblink

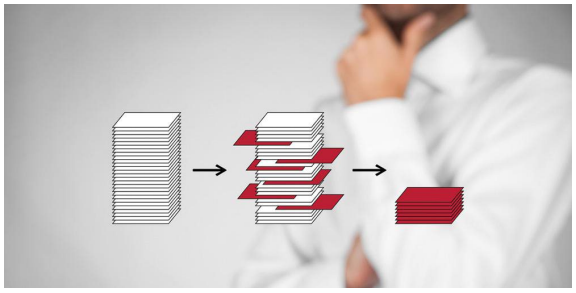
For more information and examples on data dictionaries visit:

- What is a Data Dictionary?[22]
- Visual Paradigm – Derive a Data Dictionary from Textual Analysis[23]

Data structures

A data structure is a formal method of organising specific types of data so we can access, process (manipulate) and retrieve them. In programming we have data structure such as arrays, queues, graphs, and stacks. Databases organise data differently. Common data structures used by SQL-based databases include:

- table - basic data organisation in a database
- index structures
- trees and R-trees
- record organisation
- page organisation
- file organisation.



254322622 / Jirsak / shutterstock.com

Reading

To understand the functionality of a database management system and query processing, read this paper. Research paper on Data structures for Databases[24]

[22] https://analystanswers.com/what-is-a-data-dictionary-a-simple-thorough-overview/

[23] https://www.visual-paradigm.com/tutorials/builddatadicfromtextual.jsp

[24] https://www.cise.ufl.edu/~mschneid/Research/papers/HS05BoCh.pdf

Database constraints

What is a constraint?

A constraint is a restriction that can be placed on database columns.

There are several types of constraints. In phase one (requirements collection and analysis) there were a number of business rules identified that relate to the project, and more specifically, that relate to our database. We can now analyse the database related business requirements and apply constraints to our database to ensure that the data that is stored remains valid.

If the project involves the design of an associated web application or website that will utilise the database, additional constraints can be placed in the code (whether using Java, PHP, ASP.NET or another language) to ensure data integrity and validity are maintained. Techniques can be used to ensure that we escape any output and validate and sanitise any input to protect against cross-site scripting (XSS). We can also use prepared statements or parameterised queries to protect our data against SQL Injection attacks.

282766454 / stockfour / Shutterstock.com

Types of constraints

- **Semantic constraints -** When we set our data types for each of the columns, this provides a basic constraint.

Example

If we set a column that will contain a company name, and set the length to 100, e.g. varchar(100), the database will not let us enter a string that is greater than 100 characters long. Similarly, if we have defined a column as having an integer data type, the database will not accept a string value.

- **NULL constraints -** When you create a table, you can identify if a specific column can accept NULL (no value). We can also set a database column to not accept a NULL value. This means that for this column a value must be provided when entering a new row into the database table.
- **CHECK constraints -** Check constraints allow you to provide a more refined level of validation for certain columns.

Example

You could place a constraint on a product price column so that it only accepts values greater than zero. These are also called semantic constraints.

Triggers can be used as an alternative to this constraint. Triggers allow us to carry out the data validation by placing one or more conditions on a particular database action. If the conditions are not met, the insert/update/delete operation is cancelled and the system outputs an error message.

Triggers can be used:

- Before an insert operation
- After an insert operation
- Before an update operation
- After an update operation
- Before a delete operation
- After a delete operation.

Primary Key constraints - As we have already seen, setting a column as a primary key ensures that each row will be uniquely identifiable using the key – there will not be any rows that contain exactly the same data. This is what is called entity integrity.

Foreign Key actions - Setting a foreign key constraint in our database ensures that related data in associated tables cannot be accidentally deleted. Now, if an attempt is made to delete a Customer from the Customer table, and there is a corresponding record for this Customer in the Address table, the database will act according to the referential action that has been set at the time of setting the constraint.

357517214 / Antonio Guillem / Shutterstock.com

The possible actions are:

- **Restrict (No action) -** This action will reject the delete or update operation for the parent table and return an error message.
- **Set null -** This action will delete or update the row in the parent table and set any corresponding rows in the child table to null. (This means that not null must not be set in the column of the child table.)
- **Cascade -** This action will delete or update any corresponding rows in the child table when a row in the parent table is updated or deleted.

Unique constraints

A unique constraint is placed on a superkey to ensure that the combination of columns in the superkey is always unique.

What is a superkey?

A superkey is the combination of one or more columns that identify a row. A primary key can be a superkey. A unique key can be a superkey that is used to constrain data. Once set, the database will prevent you from adding two rows with the same customer's name and email address.

Indexes

Indexing is done to speed up a database search. However, indexing is only efficient on tables with many rows. In a small table, indexing will not speed up searches. Indexing needs extra storage requirements per table to store the index information. Performance degradation can occur if a table has too many indexes.

Most of the database processing is searching (select statements) to retrieve and display data. Without indexing, inserts (write) can be fast because the database engine just adds a row to the table. When you add an index, the engine must update each index while performing the insert. Indexing works better on integer (INT) type fields that it does on CHAR or VARCHAR fields. In general, indexing will be slower for INSERT, UPDATE and DELETE (write actions) actions and faster for searches (read actions).

You can create compound indexes (an index that includes multiple columns). These compound indexes will work well in instances where the WHERE condition is likely to use multiple columns.

Indexes work well for columns that are frequently used in:

- WHERE clauses
- sorting (ORDER BY) clauses
- comparisons that use the =, >, >=, <, <=, or BETWEEN operators.

136964447 / Eliana Elisseeva / Shutterstock.com

Topic summary and review

In this topic you have learnt about the physical data model, where we converted our existing logical data model. This process involved updating entities to tables, and attributes to columns. We looked at data types that are common and unique to different DBMS including MySQL and Microsoft SQL Server. We also examined different constraint types and looked at using indexing in our database where appropriate.

Use this checklist and quiz to check what you have learnt in this topic.

- Do I understand the differences between the logical data model and physical data model?
- Do I understand common data types specific to the chosen DBMS?
- Do I understand the importance of the data dictionary as the blueprint for the database?
- Do I know how to revise and analyse business requirements and then apply constraints?

Self-check - Phase 5 - data types and constraints

Complete the questions below to check your understanding.

Do signed or unsigned data type integers ONLY accept positive numbers?

Complete the sentence with the correct missing word.

Words: *rows; integers; fields; columns*

A constraint is a restriction that can be placed on database ……...

What type of constraint is one that identifies a specific column that can accept no value?

- ☐ semantic
- ☐ NULL
- ☐ CHECK
- ☐ primary key
- ☐ unique
- ☐ foreign key

What type of constraint is placed on a superkey to make sure that the combination of columns is always unique?

- ☐ semantic
- ☐ NULL
- ☐ CHECK
- ☐ primary key
- ☐ unique
- ☐ foreign key

What type of constraint is set to provide a more refined level of validation for certain columns?

- ☐ semantic
- ☐ NULL
- ☐ CHECK
- ☐ primary key
- ☐ unique
- ☐ foreign key

Check your answers at the end of this Learner Guide

Weblink

For more information on how to use Visual Paradigm visit:

- Visual Paradigm tutorial[25]

Further information and examples of database constraints can be found at:

- GeeksforGeeks Tutorial - Constraints[26]

To learn more about constraints, follow the links below.

- General SQL constraints[27]
- MySQL specific constraints[28]
- Microsoft SQL Server specific constraints[29]
- SQL Indexing[30]

[25] https://knowhow.visual-paradigm.com/database-engineering/generate-ddl-without-db-connection/

[26] https://www.geeksforgeeks.org/difference-between-entity-constraints-referential-constraints-and-semantic-constraints/

[27] https://www.w3schools.com/sql/sql_constraints.asp

[28] https://zetcode.com/mysql/constraints/

[29] https://docs.microsoft.com/en-us/previous-versions/sql/sql-server-2008-r2/ms189862(v=sql.105)?redirectedfrom=MSDN

[30] https://docs.microsoft.com/en-us/previous-versions/sql/sql-server-2008-r2/ms189862(v=sql.105)?redirectedfrom=MSDN

Post-Physical Model Considerations

Introduction

Now that the conceptual, logical, and physical data models have been completed, and the stakeholders of the project have confirmed the design, we can move on to consider other design aspects:

- Database access and security
- User interface
- Backup and restore strategies
- Scalability

This topic will wrap up the database design requirements. Although there are several DBMS that you could use to implement the database, the concepts presented in the topic are common to most of them.

Objectives

By completing this unit it is expected that you will be able to understand:

- database access techniques
- database security requirements
- the importance of the user interface and reporting considerations
- the need to have a backup and recovery plan in place
- horizontal and vertical scalability and their application to databases.

152663261 / watcharakun / shutterstock.com

Access and security - design considerations

System access and security

301954255 / chinnarach / stock.adobe.com

It is crucial to understand how the system will be secured and what steps can be taken to ensure that it will remain that way. There are several security aspects that the designer of the database system should be mindful of. The designer should have a good understanding of the business security plan (if the organisation has produced one), as this will be the definitive reference used in designing the security features of the database system. It will dictate which security policies are implemented and how they are maintained.

Securing a web site to protect against cross-site scripting (XSS) and SQL injection can be achieved through a combination of techniques such as input validation, output escape, sanitation and parameterised queries and prepared statements.

Password and database access

106052978 / Jmiks / Shutterstock.com

What is the current organisational policy regarding passwords? How are they to be stored? It is not recommended that passwords be stored as plain text. If the database system is compromised, a hacker can easily retrieve all users' plain text passwords if they are stored this way. When the user attempts to login to the database server, an SQL query will attempt to retrieve the row based on the username and password entered. How can we protect that password?

Old MD5 and SHA hash algorithms alone are no longer a safe option for today's technologies. Developers try to keep ahead developing new stronger algorithms and strategies. If we are using PHP, you can use **password_hash()** to protect the user password. It can be configured to include SALT and COST. It supports the following algorithms: CRYPT, B-CRYPT, BLOWFISH, and Argon2. Then we need to use **password_verify()** to verify that the hash matches the entered password.

Salt - Salt is a unique random string of characters that is added to the password before it is hashed. It is stored in the site. Even if two users choose the same password the salt will always be different.

Cost - Cost can control the time needed to calculate a single hash round.

We can also encrypt data on the server side as a way of increasing security and protecting data from cyber-attacks. Popular methods for database encryption are as follows.

- Transparent or External Database Encryption (TDE) e.g., DBDefence for MSQL.
- Column-Level Encryption.
- Symmetric Encryption e.g., AES.
- Asymmetric Encryption.
- Application-Level Encryption.

Weblink

For information and syntax example to use PASSWORD_HASH() visit:

- Password Hashing in PHP[31]
- Symmetric and Asymmetric Encryption[32]
- Transparent Data Encryption (TDE)[33]
- Column Level Encryption[34]

Research

Research the password hashing algorithms used by two (2) different programming languages (not PHP) when connecting to a database. Share your findings with the class.

Reading

See a detailed overview on Application Encryption White Paper[35]

[31] https://www.php.net/manual/en/function.password-hash.php

[32] https://cyware.com/news/exploring-the-differences-between-symmetric-and-asymmetric-encryption-8de86e8a

[33] https://www.comparitech.com/net-admin/database-encryption-tools/

[34] https://blog.devart.com/column-level-sql-server-encryption-example-using-sql-complete.html

[35] https://www.futurex.com/application-level-encryption/

Access and security - user Permissions

The database administrator has total privileges on the server and databases but there are other roles that require permissions. As a rule, users should only be granted the permissions they need to carry out their work. This will keep security in check and misuse of privileges will be avoided. When an access right or privilege is granted, it is normally stored in the database with the current user as the grantor. When GRANTED BY is used with REVOKE, the privilege (registered as) granted by the named user will be removed.

519867919 / ibreakstock / shutterstock.com

Identify multiple user requirements

The business rules and user requirements we have gathered to date should give us a clear understanding of the user roles we will be required to define as part of the new system. Each user role will have specific permissions to perform specific functions.

Example

A general user may be able to add new orders and look up and edit customer details, however they may not be able to add new products to the database or generate reports. An administrator user may be given these additional privileges.

There may also be several levels of administrator.

Example

A 'user manager' may have permissions to be able to add new users and remove existing users from the system. Or a 'super administrator' may have permissions to perform every single function that is available in the system.

The system must be written in such a way as to distinguish between the different user roles that are authenticated, so that various functions (such as menu options or buttons) may be added or removed from the user interface according to the user role. This is achieved programmatically by the code during the loading of the page.

The client business model should be used as a reference to help define each of the user profiles that relate to system access. This includes clarifying details around:

- What a particular user role can access?
- What a particular user role is not able to access?
- When a particular user must change their password?
- How long the user can stay inactively logged in before time-out?
- Number of concurrent users of the system?

132796727/wavebreakmedia/shutterstock.com

Before moving onto the next consideration after the physical design, check your understanding of access and security considerations.

Self-check - access and security

Read the questions and choose the correct answer.

What configuration options are available with PASSWORD_HASH() when using the BCRYPT algorithm?

- ☐ Salt and checksum
- ☐ Redundancy check
- ☐ Salt and Cost
- ☐ Cost only

Complete this sentence to make it correct.

Words: *should; should not; can*

Users …… be able to set their own passwords without restrictions – choosing something that they can easily remember.

Complete the sentence to make it correct.

Words*: responsibilities; orders; permissions; wishes; functions; entities; attributes; keys*

As a database designer, it is important to know which users have specific …… to perform certain …….. in the database.

For login attempts that repeatedly fail, which tactic/s could be used?

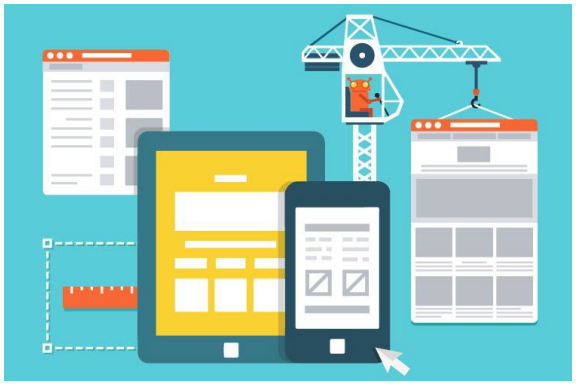
- ☐ Display a CAPTCHA input.
- ☐ Lock the account for a period of time.
- ☐ Ask the user to contact support for help.
- ☐ All of the above.

Check your answers at the end of this Learner Guide

Designing the user interface (UI)

Designing the user interface for our database follows a similar design process to the one we have just followed when designing our database. It starts by examining the user requirements.

- Who is the target audience?
- What will the user achieve by using the system? Or what are the user goals?
- What is the reason the system is being created? Or what are the business goals?



228317128 / Max Griboedov / shutterstock.com. Modified by TAFE Queensland.

When designing the interface, we need to consider all the potential user groups that will be using the system.

The user interface includes:

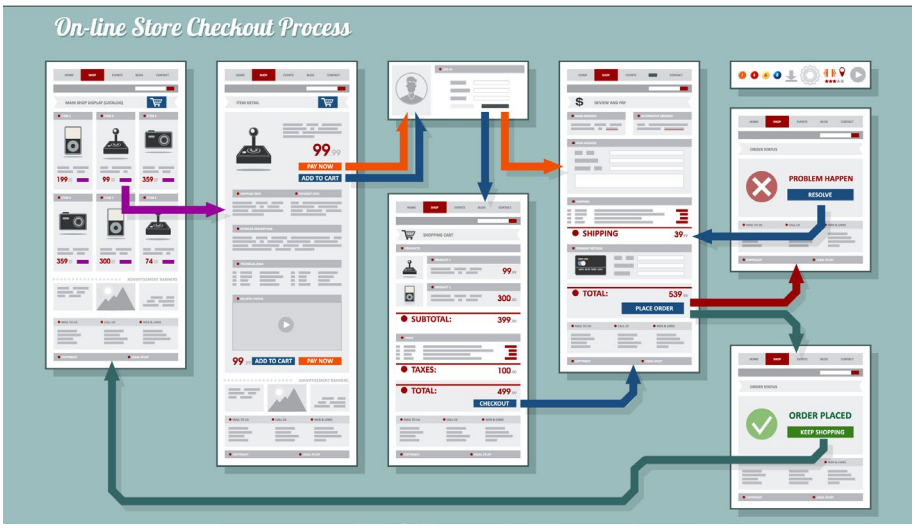
- the navigation (buttons and menus)
- input forms
- output forms

What we are really designing is an experience – the user experience (UX). A good UX design is one that is transparent to the user. The user only really notices the UX if it has been poorly designed. A superior design just works. The user can easily navigate around the design and can easily achieve what they need to get done.

Important

It is important to ask a lot of questions. Obtaining feedback from our target audience on UX is critical. But how can we do this before the user interface has been created?

Diagramming software can be used to draw wireframe diagrams of all screen designs.



395529733 / Viktorus / shutterstock.com. Modified by TAFE Queensland.

Wireframe diagrams should be presented to representatives from the various user groups you have identified, who can provide feedback on the logic and intuitiveness of the proposed design. A user might be able to identify how a particular function could be achieved more efficiently perhaps using less clicks. The greater number of users that can provide feedback the better. Asking a lot of questions will obtain subjective responses (meaning that quite often a response may be based on a particular person's preference or opinion). When there are many subjective responses, we can begin to understand more about the process being critiqued.

Gathering user requirements about the design is an approach used when designing our database.

Three different techniques.

Questionnaires

Interviews

Observations

228625591 / PixMarket, 117695041 / A-R-T, 265065383 / Leone_V / shutterstock.com. Modified by TAFE Queensland.

We can also create use case diagrams to describe a particular process to help the design team think through specific details of a sequential process flow.

Popular software to design UX solution prototypes include:

- Adobe XD
- Arvel, Axure RP
- inVision
- Figma.

Self-check - designing the user interface

Complete the questions to check your understanding.

Select all the methods below that are suitable for gathering data for analysis.

- ☐ user interviews
- ☐ questionnaires
- ☐ using your intuition only
- ☐ none of the above

What does the acronym UX stand for?

What does the acronym UI stand for?

What does the user interface not include? Select all that apply.

- ☐ target audience
- ☐ buttons for navigation
- ☐ menus for navigation
- ☐ feedback from the target audience
- ☐ input forms
- ☐ output forms

Complete the sentence.

Words: *customer, user, owner, customer, owner, business*

To fully appreciate the requirements underpinning a UI design, the designer must understand the target audience, and both the …….. and the ……..... goals.

Check your answers at the end of this Learner Guide

Identify backup and recovery requirements

Backup Requirements

During the requirements gathering phase, requirements would have been gathered pertaining to backup and recovery of the system.

321043403 / dizain / shutterstock.com

Backup Plan

These requirements should outline:

- the frequency of backup (how often)
- the type of backup
- medium (where to store the backups).

When you work with a database that stores important data, you should have a plan for backing up that database regularly. That way, if the hard drive/s that stores the database fails, you can restore the database and minimise the amount of data that is lost.

Backup Strategies

You should also have a step-by-step restore strategy to assist you if you need to use your backup files.

Three common types of backup strategies are:

- full backup
- differential backup
- incremental.

Weblink

Demystifying data backup types[36]

Another technique used to backup data is to export or dump the database. This allows you to either export or dump the data or the structure or both. Once you have your export or dump you use it to import the data (and structure) to restore the database. It is mostly used during development to keep track of changes.

[36] https://parablu.com/demystifying-data-backups-types-of-backups/

In terms of database recovery options, each DBMS offers several strategies to support the recovery process, for example, SQL server has three recovery models that work in conjunction with their backup strategies:

- simple
- full
- backlogged.

Your backup plan should also identify where the data is going to be backed up. The answer to this question will depend on the size of your database and on how important this data is to your organisation. Some organisations outsource the backup of their system to specialised backup companies. The most common options are:

- onsite backups
- offside backups
- cloud backups.

These are managerial decisions that as the database designer you cannot make without consultation with the stakeholders.

215905031 / onephoto / stock.adobe.com

Database Scalability

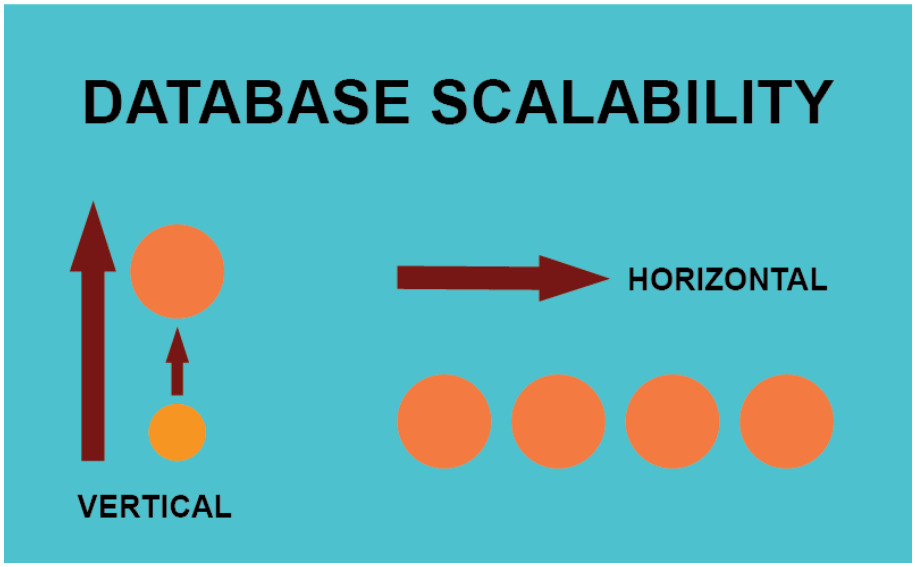
In DBMS, **scalability** refers to the strategy that we can use to handle a greater data volume of data in the database by adding the necessary processing power to perform the extra tasks in the same amount of time.

We can use **horizontal scalability** also referred to as **scale out** and add more machines (in parallel) to distribute the data and data processing. SQL-based DBMS are well-suited for horizontal scalability.

Another strategy is to use **vertical scalability** also referred to as **scale up** and add more and better resources to existing machines. NoSQL databases are better suited for vertical scalability.

Reading

For more information about scalability see the following Database scalability[37]



Database Scalability © TAFE Queensland

[37] https://www.red-gate.com/simple-talk/databases/sql-server/performance-sql-server/designing-highly-scalable-database-architectures/

Final submission to client

The final submission of the database design should be submitted to the client before the implementation of the database takes place (phase six). Final submission artefacts will include:

- approved user requirements documentation
- approved use case diagrams
- technological considerations
- conceptual, logical, and physical data model designs (ER diagrams)
- data dictionary
- user interface wireframe diagrams (navigation, input forms, reports)
- access and security considerations
- backup and recovery plans.

Once approval has been granted by the client, the design team may proceed to implement the database.

520788541 / fizkes / shutterstock.com

Topic summary and review

In this topic you learnt about security strategies including encryption methods, and user access privileges. You have learnt about concepts relating to user interface design and the importance of consulting with stakeholders involved in the project, to appreciate and understand the requirements relating specifically to the user interface. You also learnt about the importance of identifying backup and recovery requirements and preparing a backup and recovery strategy in the event of a catastrophic system failure. Finally, you were introduced to the concept of database scalability to increase performance when data volumes increase.

Weblink

To gain a deeper understanding of the concepts presented see the links provided to access more information and examples. This one explains data backup types.

5 of the Best Cloud Backup Solutions for Small Businesses[38]

519935605 / Alexander Supertramp / shutterstock.com

[38] https://www.eiresystems.com/best-cloud-backup-solutions-for-small-businesses/

Activity

In your own notebook, use this table to create an appropriate backup strategy for a local kindergarten database. Add rows, as necessary.

Back up Type	What data?	Backup Method	Time Schedule	Notes

Self-check - Post-physical design – backup and scalability

Complete the questions below to check your understanding.

Complete the definition of scalability.

***Words**: similar; smaller; greater; adding; changing; removing; company; time, computer*

Scalability is the strategy used to handle a …….. volume of data by …….. the necessary processing power to perform extra tasks in the same ……..

What is absolutely essential in every database management project?

.....

Select all that apply to horizontal scalability.

- ☐ scaling up
- ☐ scaling out
- ☐ adding more machines
- ☐ adding to the existing machine(s)
- ☐ suitable for noSQL
- ☐ suitable for SQL

Select all that apply to vertical scalability.

- ☐ scaling up
- ☐ scaling out
- ☐ adding more machines
- ☐ adding to the existing machine(s)
- ☐ suitable for noSQL
- ☐ suitable for SQL

Check your answers at the end of this Learner Guide

Self-check answers

Self-check – Phases of database design

Match the correct phase to its corresponding number to check your understanding of the six phases of database design.

Answers: 1, 2, 3, 4, 5, 6

Phase name	Phase number
Requirements, Collection and Analysis	1
Conceptual Design - Will the end users be able to view the database model from a high vantage point (conceptually)?	2
Choice of DBMS	3
Logical Design – Can the database designers view the database model from a logical vantage point? This is also called Data Model Mapping.	4
Physical Design – Can the programmers and database administrators view the model from a physical perspective?	5
Implementation	6

Self-check – Fundamentals of database design

Complete the questions below to check your understanding.

Complete the sentence.

A relational database is based on data sharing…….

- ☐ tables
- ☑ relations
- ☐ fields
- ☐ values

Relations between data are represented as …….

- ☑ tables
- ☐ relations
- ☐ fields
- ☐ values

Complete the sentence by choosing the correct words.

Words: *relations, tables, fields, values, relations, tables, fields, values*

Tables contain columns which are fields and rows which hold specific values.

What are special key fields? (answer consists of 4 words)

primary and foreign keys

What process is used to avoid data duplication (also known as redundancy)?

normalisation

Note: *How to follow the normalisation process will be learnt in Phase 4.*

Self-check - Fundamentals of database design and Phase 1

Complete the questions below.

What does UML stand for?

Unified modelling language

In a use case diagram, what is not part of the system, but interacts with it?

- ☐ inclusions
- ☐ use case
- ☑ actor

What is the PIECES framework used to do? Check the four that apply.

- ☑ identifying the performance expectations of the database project
- ☐ identifying the expectations of the client's customers
- ☑ identifying the data types that will need to be included in the database
- ☑ identifying any security concerns and therefore any security measures that might be needed
- ☐ helping the client's whole business run as smoothly as possible
- ☐ improving the customer interface for the client
- ☑ improving the quality of service that the client delivers to its customers

What is the best data type suited to storing monetary values?

.....

Complete this sentence about the key outcomes. Choose from the following Words:

- a statement of user requirements
- a list of budget constraints
- a list of stakeholders and their expectations
- use case diagrams
- a mock-up of the whole project
- staffing requirements

The key deliverables for phase one are a statement of user requirements and use case diagrams

Self-check - ERD notation.

Match the symbol names to their correct ERD notation.

Answers:

entity, attribute; primary key; cardinality ratio; fork or crow's feet ; 1:1; 1:N or N:1; M:N

ERD notation	Notation symbol name
This is the object about which the database needs to record information. They may become tables.	entity
This refers to the data that we record about entities. E.g. name, student number	attribute
This uniquely identifies a table. So the value of this cannot be repeated in another row of the table.	primary key
This is the term used to describe the relationship between one table and another table.	cardinality ratio
This term describes the notation symbol between tables.	fork or crow's feet
This ratio indicates the degree of participation of one entity to another entity. E.g. doctor to patient	1:1
This ratio is used when one entity relates to multiple other entities. E.g. one doctor treats many patients	1:N or N:1
This ratio is used to identify the degree of participation between one entity and multiple other entities, and the reverse. E.g. one doctor treats many patients, and one patient is treated by many doctors.	M:N

Self-check - Conceptual Database Design - Phase 2

Check your understanding of this topic by completing the following quiz.

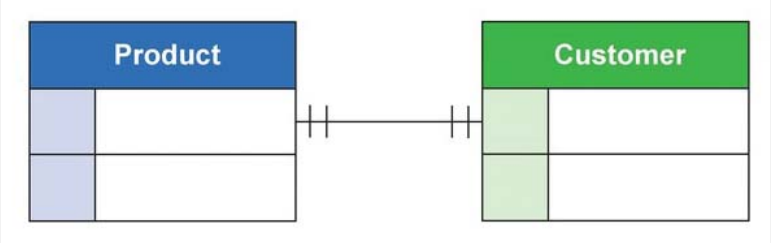
What is the term used to describe how an entity relates to another entity?

cardinality ratio

What describing word is used for the data model that displays entities and their relationships?

conceptual

Look at the ERD sample presented below. It represents two entities: Product and Customer. What does the cardinality ratio notation describe?



- ☐ that many products can be bought by one customer?
- ☑ that one product can be bought by one customer and that one customer can buy one product?
- ☐ that one product can be stocked by many stores?
- ☐ that many products can be bought by many customers?

What acronym is used for the most relevant diagram notation to show relationships between entities?

ERD

What is the key deliverable produced at the end of phase two?

- ☐ the use case diagram
- ☐ the database
- ☑ the conceptual data model
- ☐ the physical data

Self-check - Choice of DBMS - Phase 3

Complete the questions below to check your understanding.

What does the A stand for in ACID compliance?

atomicity

Which needs in the following list would make a SQL database preferable? Select all that apply.

- ☑ structured data
- ☐ horizontal scalability
- ☑ ACID compliance
- ☑ vertical scalability
- ☑ fast query
- ☐ flexibility

Which of the following needs would make a noSQL database preferable?

- ☐ structured data
- ☑ horizontal scalability
- ☐ ACID compliance
- ☐ vertical scalability
- ☐ fast query
- ☑ flexibility

Complete the sentences by choosing the correct word.

Words: *staffing; language; organisational*

Technical, economic, and organisational factors must be considered when choosing a DBMS.

Words: *technical, economic, organisational*

Backup requirements come under the category of technical considerations.

Self-check - Logical database design and normalisation - Phase 4

Check your understanding of this topic by completing the following quiz.

Which normal form do we achieve when we decompose attributes containing multiple values?

- [x] First normal form (1NF)
- [ ] Second normal form (2NF)
- [ ] Third normal form (3NF)
- [ ] One beyond 3NF

What primary key contains two or more columns?

Composite

What is used to uniquely identify rows in a specific entity?

- [ ] foreign key
- [ ] decomposition key
- [ ] candidate key
- [x] primary key

What term describes the systematic process of ensuring that our entities will not contain redundant or duplicate data?

Normalisation

What situation occurs when the same attribute name and values are stored in different entities throughout the database?

data redundancy

Self-check - Normalisation

Match the rules into their correct normalisation (Normal Form) rules.

Answers: *NF1; NF2; NF3*

Normalisation form rules.	**Match the rule from the above answers**
Rule 1: A primary key (PK) exists – meaning that there are no repeating values in the column	NF1
Rule 2 All values in each column are atomic – single indivisible values	NF1
Rule 1: The table is in the 1NF	NF2
Rule 2: All columns, that are not part of the primary key, are fully functionally dependent on the primary key.	NF2
Rule 1: The table is in the 2NF	NF3
Rule 2: There exists no transition dependency amongst the non-key columns.	NF3

Self-check - Phase 5 - data types and constraints

Complete the questions below to check your understanding.

Do signed or unsigned data type integers ONLY accept positive numbers?

unsigned

Complete the sentence with the correct missing word.

Words: *rows; integers; fields; columns*

A constraint is a restriction that can be placed on database columns

What type of constraint is one that identifies a specific column that can accept no value?

- ☐ semantic
- ☑ NULL
- ☐ CHECK
- ☐ primary key
- ☐ unique
- ☐ foreign key

What type of constraint is placed on a superkey to make sure that the combination of columns is always unique?

- ☐ semantic
- ☐ NULL
- ☐ CHECK
- ☐ primary key
- ☑ unique
- ☐ foreign key

What type of constraint is set to provide a more refined level of validation for certain columns?

- ☐ semantic
- ☐ NULL
- ☑ CHECK
- ☐ primary key
- ☐ unique
- ☐ foreign key

Self-check - access and security

Read the questions and choose the correct answer.

What configuration options are available with PASSWORD_HASH() when using the BCRYPT algorithm?

- ☐ Salt and checksum
- ☐ Redundancy check
- ☑ Salt and Cost
- ☐ Cost only

Complete this sentence to make it correct.

Words: *should; should not; can*

Users should not be able to set their own passwords without restrictions – choosing something that they can easily remember.

Complete the sentence to make it correct.

Words*: responsibilities; orders; permissions; wishes; functions; entities; attributes; keys*

As a database designer, it is important to know which users have specific permissions to perform certain functions in the database.

For login attempts that repeatedly fail, which tactic/s could be used?

- ☐ Display a CAPTCHA input.
- ☐ Lock the account for a period of time.
- ☐ Ask the user to contact support for help.
- ☑ All of the above.

Self-check - designing the user interface

Complete the questions to check your understanding.

Select all the methods below that are suitable for gathering data for analysis.

- [x] user interviews
- [x] questionnaires
- [ ] using your intuition only
- [ ] none of the above

What does the acronym UX stand for?

user experience

What does the acronym UI stand for?

user interface

What does the user interface not include? Select all that apply.

- [x] target audience
- [ ] buttons for navigation
- [ ] menus for navigation
- [x] feedback from the target audience
- [ ] input forms
- [ ] output forms

Complete the sentence.

Words: *customer, user, owner, customer, owner, business*

To fully appreciate the requirements underpinning a UI design, the designer must understand the target audience, and both the user and the business goals.

Self-check - Post-physical design – backup and scalability

Complete the questions below to check your understanding.

Complete the definition of scalability.

***Words**: similar; smaller; greater; adding; changing; removing; company; time, computer*

Scalability is the strategy used to handle a greater volume of data by adding the necessary processing power to perform extra tasks in the same time.

What is absolutely essential in every database management project?

Data backup

Select all that apply to horizontal scalability.

- ☐ scaling up
- ☑ scaling out
- ☑ adding more machines
- ☐ adding to the existing machine(s)
- ☐ suitable for noSQL
- ☑ suitable for SQL

Select all that apply to vertical scalability.

- ☑ scaling up
- ☐ scaling out
- ☐ adding more machines
- ☑ adding to the existing machine(s)
- ☑ suitable for noSQL
- ☐ suitable for SQL

References

(2021) Scribd.com. Available at: https://www.scribd.com/doc/7298188/PIECES-Framework#from_embed (Accessed: 21 October 2021).

Constraints (2012). Available at: https://docs.microsoft.com/en-us/previous-versions/sql/sql-server-2008-r2/ms189862(v=sql.105)?redirectedfrom=MSDN (Accessed: 20 October 2021).

Economic factors (Choice of DBMS) - GeeksforGeeks (2018). Available at: https://www.geeksforgeeks.org/economic-factors-choice-of-dbms/ (Accessed: 17 October 2021).

Full vs. incremental vs. differential: Comparing backup types (2021). Available at: https://searchdatabackup.techtarget.com/tip/Data-backup-types-explained-Full-incremental-differential-and-incremental-forever-backup (Accessed: 20 October 2021).

Problem Solving: Unveiling the Multiple Faces of the PIECES Framework — Business Analyst Learnings (2021). Available at: https://www.businessanalystlearnings.com/blog/2013/6/23/problem-solving-unveiling-the-multiple-faces-of-the-pieces-framework (Accessed: 17 October 2021).

Student rules and policies (2021). Available at: https://tafeqld.edu.au/about-us/policy-and-governance/policies-and-procedures/student-rules-and-policies/index.html (Accessed: 20 October 2021).

Glossary

Term	Meaning
Attributes	In database design an attribute represents an entity's property. For example, in the entity Employee the attribute EmployeeID is an attribute. Other attributes could be EmployeeName or EmployeeEmail. After normalisation, the attribute became the table's columns.
Cardinality ratio	Cardinality ratio represents the degree of participation of one entity in another entity. It is commonly represented by one-to-one, one-to-many and many-to-many or in their shorthand notation 1:1, 1:N and N:M.
Conceptual data model	The conceptual model is high-level or abstract representation of a database system in the real world. It is presented as a diagram often called entity relationship (ER) or conceptual schema.
Data structures	Data structures are the name given to organised ways of organising data for a specific purpose. The purpose can be to store, manipulate, or retrieve data or how the data is stored physically or in memory. In databases, this includes from a table to indexes, files, and pages.
Data types	Data types are the classification given to distinct types of data, including their ranges and format. Typical data types in databases are integers (INT(),) decimal (DEC(x.y)), fix character length(CHAR(), variable length (VARCHAR()) and date.
Data dictionary	In a database, the data dictionary is a document that collates the database table definitions and as a minimum includes table name, column names and datatype (with ranges) and other precision parameters such as NULL, UNIQUE, and so on.
Data redundancy and duplication	Any data that is unnecessarily duplicated is redundant and should be deleted. The process to identify and solve this problem is called normalisation.
Entities	In a database an entity represents an object about which the database needs to store information. After normalisation, the entities will become the database tables.
Entity relationship diagram (ERD)	The ERD is a graphical model that represents a database for a specific domain. The main components of an ERD are entities, attributes, and relations (cardinality ratio). There are a few notations and symbols that can be used to draw an ERD.

Foreign key	A foreign key is defined in a second table which relates to the primary key in the first table. The primary and foreign keys share the same values and establish a link between two related tables. Foreign key constraints can be applied to the relationship which can prevent related records from being removed.
Horizontal scalability or **Scaling out**	Scalability is the solution that you try to address database performance load issues due to increased data volumes. Adding more machines (in parallel) to distribute the data and data processing. SQL databases are better suited for horizontal scalability.
Logical data model	The logical data model is derived from the conceptual model, but it contains DBMS-specific information in relation to entities, attributes, primary and feign keys not present (or not necessary) in the conceptual model. The conceptual model is usually produced before the DBMS has been selected, often as a communication tool with stakeholders.
Index	An index can be defined on one or more columns, allowing database users to quickly locate records, in a comparable way to using an index in a book. For example, in a customer table, we could index the last name of each customer. Retrieving a customer using their last name can be achieved quickly using an index, compared to not using one. This would be akin to paging through a book starting at the front, to find the information, rather than using the index.
Normalisation	Normalisation is the process used in relational databases to reduce redundancy and duplication and ensure efficient storage of the data. The process is implemented with three main normal forms or set of normalisations rules, 1NF, 2NF, and 3NF. The process can be extended with other normal forms.
Physical data model	The physical data model is derived from the logical data model and represents the system as it is going to be implemented.
Primary key	A primary key can be defined on one or more columns, to uniquely identify a particular row. A primary key defined across multiple columns is called a composite key.
Referential integrity constraint	The referential integrity constraint checks that the foreign key in an entity refers to and always corresponds to an existing primary key in other entity and must fulfil two conditions: • The foreign key column must have the same declaration as the primary key. • A value in the foreign key column must be equal to a value in the primary key column or set to NULL.

Relational database	An organised and structured set of data held in interrelated tables. Each table typically shares data that is common with other tables, in columns that are defined in a primary/foreign key relationship.
Table	A table contains a series of rows (or records), and columns (or fields). Each column is of a particular data type, for example, text-based, numeric, or date-based. Each row typically holds unique data. For example, a table of customers.
UML	UML (unified modelling language) is a standard way used to visualise the design of a system. There are several UML diagrams that can be used to achieve this goal, including the use case diagram, class diagram, activity diagram, and sequence diagram.
User needs analysis	User needs analysis is the research, communication, and data gathering activities carried out to identify the needs and expectations that users have regarding products and services.
Validation rules or constraints	A validation constraints check and restrict input into a table. This is often achieved using CHECK constraints, semantic constraints, and triggers.
Vertical scalability or **Scaling up**	Scalability is the solution that you try to address database performance load issues due to increased data volumes. Adding more and better resources to existing machine. NoSQL databases are better suited for vertical scalability.

Appendix 1

DATA DICTIONARY for <<Database Name>>											
Table 1	**Column**	**Data Type & Length**	**Signed Unsigned**	**Max/Min Values Or Range**	**Primary Key**	**Auto-Increment**	**Foreign Key**	**Required Field Null - Not Null**	**Unique Field Identified As Key**	**Business Data Constraints Data Validation Triggers**	**Comments**

