

1 Inleiding

1.1 Wie gebruikt databank(technologie)?

We worden dagelijks overspoeld door een enorme hoeveelheid gegevens en informatie die ons continu wordt aangeboden via internet, e-mail, Facebook, Twitter, Instagram, WhatsApp, YouTube, radio en televisie. Niet alleen in ons privéleven worden we overspoeld, ook in onze contacten met de ‘zakelijke wereld’ worden we er volop mee geconfronteerd. Een groot deel van de aangeboden data ervaren we als niet-relevant. Maar er zijn ook zaken die we willen vastleggen. Soms omdat we daartoe verplicht zijn (bedrijven/instellingen/organisaties), soms omdat we voor onszelf zekerheid willen (afspraken in agenda, e-mails, contactgegevens) en soms omdat we het gewoon leuk vinden (reisverslag van de vakantie, kasboek met inkomsten en uitgaven, ...).

Op het werk, tijdens het winkelen, in onze vrije tijd, ... bijna overal wordt data verzameld en opgeslagen.

Voor het vastleggen van gegevens wordt tegenwoordig zelden nog gebruik gemaakt van pen en papier en vrijwel altijd van een computer, een tablet of een smartphone. Deze hardware alleen volstaat niet, er dient ook software aanwezig te zijn waarmee de gegevens kunnen worden ingevoerd en opgeslagen. In het huidige informatietijdperk, is een databank daarom nooit veraf. Alle bedrijven/instellingen/organisaties registreren hun inkoopgegevens, productiegegevens, verkoopgegevens, financiële gegevens, patiëntgegevens, studentgegevens, gegevens over verkeersstromen, ... in een databank.



Enkele voorbeelden van het gebruik van databanken in het dagelijks leven:

- Betalen aan de kassa: De barcode van de producten worden gescand en in de databank wordt de prijs van het product opgezocht. In de databank wordt het aantal items in stock met één

verminderd. Als de stock lager wordt dan een bepaalde drempelwaarde, kan automatisch een order geplaatst worden.

- Bibliotheek: De lokale bibliotheek heeft een databank die de details van de boeken, lezers, reservaties, ... bijhoudt. Je kan een boek opzoeken in de databank op basis van titel, auteurs of onderwerp.
- Vakantie boeken: Wanneer je een vakantie boekt, moeten er wijzigingen aangebracht worden in de databanken om te voorkomen dat eenzelfde kamer of eenzelfde zitje op het vliegtuig twee keer geboekt zou worden.
- Bankautomaat: Als je geld opvraagt aan een bankautomaat, moet er eerst gecontroleerd worden of er nog voldoende geld staat op je bankrekening. Indien dat het geval is, moet vervolgens bijgehouden worden in de databank hoeveel geld je hebt opgenomen en moet het bedrag bijgewerkt worden.

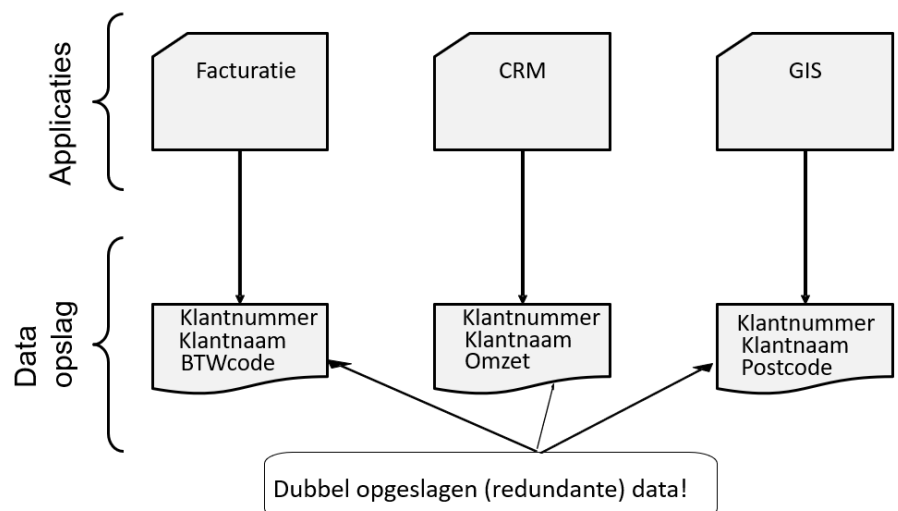
1.2 Gegevensmanagement via bestanden

Voor er gebruik gemaakt werd van databanken zoals we ze vandaag kennen om data in op te slaan, werd die data opgeslagen in bestanden. Deze aanpak is grotendeels verouderd maar toch niet helemaal verdwenen. Zo maakt de HP-NonStop mainframe gebruik van Enscribe, wat een soort van bestandsgebaseerd databankmanagement systeem is.



In een bestandsgebaseerde oplossing, definieert elke toepassing zijn eigen bestanden. Er wordt dus gebruik gemaakt van verschillende bestanden zonder relaties tussen de bestanden. Het is in de applicatieprogramma's dat de relaties tussen de data uit de verschillende bestanden moeten worden gelegd/gevonden.

Stel dat we een traditionele facturatie-applicatie hebben, geschreven in een programmeertaal zoals COBOL of C, die gebruik maakt van klantgegevens zoals klantnummer, klantnaam, BTW-code, ..., opgeslagen in een apart bestand. Een andere toepassing, zoals een CRM-systeem (Customer Relationship Management), maakt gebruik van een ander bestand met dezelfde gegevens. Ten slotte slaat een derde toepassing (GIS) informatie zoals klantnummer, klantnaam en postcode op in weer een ander bestand. De gegevensbestanden bevatten alleen de gegevens zelf; de gegevensdefinities en -beschrijvingen worden in elke toepassing afzonderlijk opgenomen. Een toepassing kan gebruik maken van één of meer bestanden. Naarmate er meer applicaties worden ontwikkeld met overeenkomstige gegevensbestanden, zal deze bestandsgebaseerde benadering van gegevensbeheer ernstige problemen veroorzaken.



1.2.1 Nadelen gegevensmanagement via bestanden

Er zijn echter een heleboel nadelen verbonden aan deze op bestanden gebaseerde werkwijze. Een aantal worden hieronder geïllustreerd. In het onderstaande bestand staan de certificaten van werknemers opgelijst.

Waarom zijn de rijen 9 en 10 leeg?		Hoe kan men een alfabetische lijst van de werknemers creëren?		Hoe kan je het aantal werknemers met het certificaat 'Basic Database Manipulation' tellen?		Is Basic Database Manipulation hetzelfde als Basic DB Manipulation?		Was als een werknemer een vierde certificaat verwerft?		
↓		↓		↓		↓		↓		
ID	Enum	Name	Title	HireDate	Skill1	Skill1Date	Skill2	Skill2Date	Skill3	Skill3Date
1	12345	Biran Oates	DBA	2/14/2005	Basic Database Management	2/14/2012	Advanced Database Management	2/14/2015	Basic Web Design	8/09/2018
2	18273	Marco Bienz	Analyst	7/28/2016	Basic Web Design	3/08/2019	Advance Process Modeling	8/19/2022		
3	16234	Jasmine Patel	Programmer	8/10/2015	Basic Web Design	8/10/2017	Advanced C# programming	8/10/2017	Basic DB manipulation	1/29/2022
4	13373	Franklin Johnson, Jr.	Purchasing Agent	3/15/2012	Advanced Spreadsheets	6/20/2021				
5	13567	Almond, Robert	Analyst	9/30/2022	Basic Process Modeling	9/30/2023	Basic Database Design	5/23/2024		
6	10282	Richardson, Amanda	Clerk	4/11/2021						
7	19382	Susan Mathis	Database Programmer	8/02/2020	Basic DB Design	8/02/2022	Basic Database Manipulation	8/02/2022	Advanced DB Manipulation	5/01/2023
8	14311	Duon, Lee	Programmer	9/01/2024	Basic Web Design	9/02/2024				
9					Master Database Programming					
10					Basic Spreadsheets					
11	19002	Wade Gaither	Clerk	5/20/2020	Advanced Spreadsheets	5/16/2023	Basic Web Design	5/16/2023		
12	13383	Raymond F. Matthews	Programmer	3/12/2022	Basic C# Programming	3/12/2024				
13	19283	Chavez, Juan	Clerk	7/04/2020						
14	14893	Patricia Richards	DBA	6/11/2014	Advanced Database Management	6/11/2016	Advanced Database Manipulation	9/20/2022		
15	13932	Lee, Megan	Programmer	9/29/2023						

Problemen die in de afbeelding hierboven geïllustreerd worden:

- Er zijn een aantal lege rijen gesloten in het bestand.
- Als je wil een alfabetische lijst van werknemers geven, moet je dit programmeren.
- Als je wil tellen hoeveel werknemers een certificaat Basic Databank Manipulation hebben, moet je dit ook programmeren.
- Is Basic Databank Manipulation hetzelfde als Basic DB Manipulation?
- Nu is er enkel plaats voorzien voor 3 certificaten, wat als een werknemer een extra certificaat behaalt. Moeten er dan twee extra kolommen toegevoegd worden?
- Wat als zowel de HR-dienst als de CEO dit bestand in hun bezit willen hebben en er wijzigingen moeten aangebracht worden?

Uit het bovenstaande voorbeeld komen duidelijk een aantal problemen naar voren, als er gewerkt wordt met een bestandsgebaseerde oplossing. De problemen worden hieronder in detail besproken.

Nadeel 1: Verspreiding en isolatie van gegevens

Wanneer gegevens in afzonderlijke bestanden worden opgeslagen, is het moeilijker om toegang te krijgen tot de gewenste gegevens.

Stel, men wil de gegevens over de verworven certificaten combineren met de salarisgegevens van de werknemers om bijvoorbeeld de volgende vraag te beantwoorden: 'Hoeveel verdient een werknemer met het certificaat 'Basic Database Management' en minstens 5 jaar anciënniteit?'.

Als we deze vraag willen beantwoorden, moeten we het bestand met de certificaten regel per regel overlopen en controleren of de werknemer het certificaat 'Basic Database Management' heeft behaald. Vervolgens moeten we het bestand met de salarisgegevens regel per regel overlopen om het bijhorende salaris van de betrokken werknemer te kennen. Deze data wordt daarna weggeschreven naar een tijdelijk bestand dat een oplijsting bevat van de salarisgegevens van alle werknemers met het gevraagde certificaat en minstens 5 jaar anciënniteit. Een dergelijke verwerking is moeilijk.

Nadeel 2: Gegevensredundantie

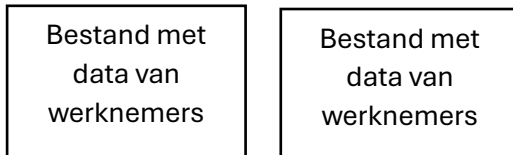
Wanneer gegevens in afzonderlijke bestanden worden opgeslagen, is het mogelijk dat dezelfde gegevens meer dan eens worden opgeslagen. We spreken over **gegevensredundantie**.

Er zijn heel wat nadelen verbonden aan het veelvuldig opslaan van dezelfde data:

- Gegevensredundantie is verspilling. Het kost tijd en geld om de gegevens meer dan eens in te voeren.
- Het neemt extra opslagruimte in beslag, met bijbehorende kosten.
- Het belangrijkste nadeel is dat gegevensredundantie kan leiden tot het verlies van **data-integriteit**: de data is niet meer consistent. Dit wordt hieronder geïllustreerd.

HR – afdeling

Payroll - afdeling



Stel dat de HR – afdeling van een bedrijf beschikt over een bestand met de data van werknemers (voornaam, familienaam, geboortedatum, adres, woonplaats, loon, ...) en dat de Payroll – afdeling van dit bedrijf ook beschikt over een analoog bestand met de data van werknemers. Als een werknemer een loonsverhoging krijgt, dan moeten de beide bestanden aangepast worden, anders is de data niet meer consistent.

Er zijn verschillende mogelijkheden om ervoor te zorgen dat de data consistent blijft, maar alle manieren zijn tijdrovend en mogelijk onderhevig aan (menselijke) fouten.

- De wijziging wordt onmiddellijk gecommuniceerd naar de beide departementen zodat beide departementen elk afzonderlijk hun bestand kunnen aanpassen. Hier schuilt natuurlijk een groot gevaar op fouten.
- De wijziging wordt gecommuniceerd naar één van de departementen, bijvoorbeeld het HR-departement, en 's nachts wordt het bestand van de HR – afdeling gekopieerd naar het bestand van de Payroll – afdeling. Dan is er tijdelijk wel geen up-to-date informatie bij de Payroll – afdeling.
- Er wordt een synchronisatieprogramma geschreven zodat de beide bestanden vergeleken worden en up-to-date gebracht worden. Zolang de synchronisatie niet gestart is, is de data in beide bestanden verschillend.

Nadeel 3: Data afhankelijkheid

Er is een zeer grote verwevenheid tussen het programma dat gebruik maakt van het bestand en de structuur van de data in het bestand.

Stel dat één lijn van het bestand met data van werknemers er als volgt uitziet

id	voornaam	familienaam	adres	pc	woonplaats	tel	statuut	loon	
0	5	35	90	145	155	200	215	260	280

In dat geval weet je dat het loon van de werknemer terug te vinden is van karakter 260 tot karakter 280. Om het gemiddelde loon te vinden van alle werknemers, schrijf je een programma dat alle regels één voor één inleest, het loon ophaalt van karakter 260 tot karakter 280 en de som van al deze lonen maakt. Tenslotte wordt de totale, berekende som gedeeld door het aantal werknemers.

Stel dat je echter vanaf een bepaald moment bijvoorbeeld ook de tweede naam van elk personeelslid (of het gsm – nummer of het emailadres) wil opslaan, dan zal de bovenstaande structuur bijvoorbeeld de volgende wijziging ondergaan:

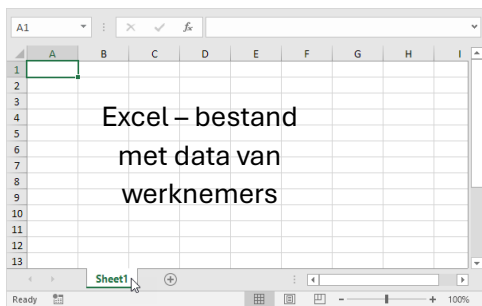
id	voornaam	tweede naam	familienaam	adres	pc	woonplaats	tel	statuut	loon	
0	5	35	65	120	175	185	230	245	290	310

Niet enkel moet het bestand met alle data van de werknemers aangepast worden, maar bovendien moeten ook alle programma's die gebruik maken van deze data gewijzigd worden. Het programma dat het gemiddelde loon berekent voor alle werknemers, moet vanaf nu karakter 290 tot karakter 310 ophalen.

Nadeel 4: Incompatibiliteit

Laten we terugkeren naar het voorbeeld waarbij dezelfde data van werknemers in twee verschillende departementen wordt bijgehouden, namelijk het HR – departement en het Payroll – departement. Het is heel goed mogelijk dat het Payroll – departement gebruik maakt van een COBOL – programma om de data in het bestand te verwerken, en dus een COBOL – bestand nodig heeft, terwijl het HR – departement een Excel – bestand met dezelfde data gebruikt. Er is dus een conversieprogramma nodig dat het Excel – bestand kan converteren naar een COBOL – bestand en vice versa.

HR – afdeling



Payroll – afdeling

COBOL -
bestand met
data van
werknemers

Nadeel 5: Fixed queries

Toen men overging van klassieke fichebakken naar een bestandsgebaseerd databankmanagement systeem, was dat een enorme verbetering. Het resultaat was echter dat gebruikers plots ook heel andere vragen wilden kunnen stellen. Omdat er echter een zeer grote verwevenheid is tussen de applicatie en de manier waarop data is opgeslagen in het bestand, was het niet mogelijk voor eindgebruikers om onmiddellijk deze vragen te stellen. Stel dat de applicatie enkel toelaat om het gemiddelde loon van alle werknemers te berekenen, dan is het niet mogelijk om het maximum- of het minimumloon op te vragen, tenzij de applicatie eerst uitgebreid wordt. En dat terwijl de nodige data in se wel beschikbaar is.



Uit al het voorgaande blijkt dat de bestandsgebaseerde aanpak verre van ideaal was en heel veel beperkingen met zich mee bracht. In essentie kunnen deze beperkingen toegeschreven worden aan voornamelijk twee factoren:

- De datadefinitie is ingebed in de applicaties die gebruik maken van de data, in plaats van afzonderlijk opgeslagen te worden.
- De data kan enkel opgevraagd en gewijzigd worden via de applicaties.

Er was dus een nieuwe aanpak vereist, wat resulteerde in het ontstaan van databanken en databankmanagementsystemen (DBMSen).

1.3 Basisbegrippen

Databank

Een databank is een gedeelde verzameling van logisch met elkaar verbonden gegevens en hun beschrijving, ontworpen om aan de informatienoden van een organisatie te voldoen.

Om een databank te kunnen opzetten, beheren en efficiënt gebruiken, zijn computerprogramma's nodig. De belangrijkste software in een databanksysteem is het databankmanagementsysteem.

Databankmanagementsysteem

Het databankmanagementsysteem (DBMS) is een verzameling van programma's waarmee een databank kan worden gecreëerd en beheerd en waarmee gegevens in de databank kunnen worden geladen, gewijzigd en opgevraagd. Het is de interface tussen de gebruikersprogramma's en de databank. Alle interactie met een databank gebeurt via het DBMS: als een gebruiker data wil opzoeken, toevoegen, wijzigen of verwijderen, wordt dit ingevoerd in het DBMS, dat vervolgens zorgt voor de correcte afhandeling van de gevraagde operatie.

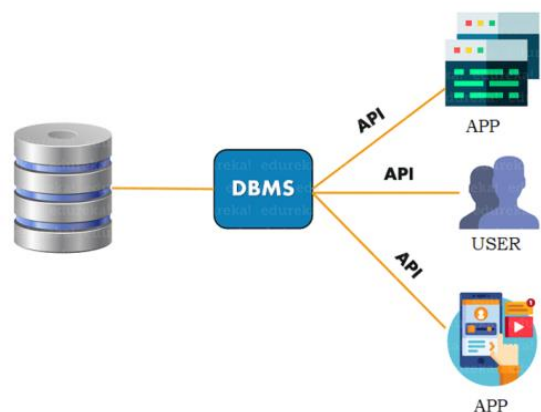
Voorbeeld

Onderstel een applicatie, bijvoorbeeld de software in de kassa van een winkelketen, of een medewerker van een bedrijf, die data opvraagt aan de databank of wegschrijft naar de databank. De gebruiker of de applicatie heeft GEÉÉN rechtstreekse toegang tot de fysieke databank. De toegang tot de effectieve data wordt geregeld door de DBMS! Dit is ook logisch. Stel je voor wat er zou gebeuren als alle kassasystemen van de winkelketen rechtstreeks in de databank wijzigingen zouden aanbrengen. Je hebt een soort van georchestreerde toegang tot de databank nodig.



Een databankmanagementsysteem laat met andere woorden onder andere het volgende toe:

- het definiëren van gegevens
- het manipuleren van gegevens
- het bewaken van de integriteit
- de beveiliging ondersteunen, zodat gebruikers en groepen alleen toegang hebben tot de databanken en gegevens waarvoor ze geautoriseerd zijn.
- back-up en recovery van data
- beheerstools om de prestaties van de databank te monitoren en te optimaliseren



Voordelen van DBMSen

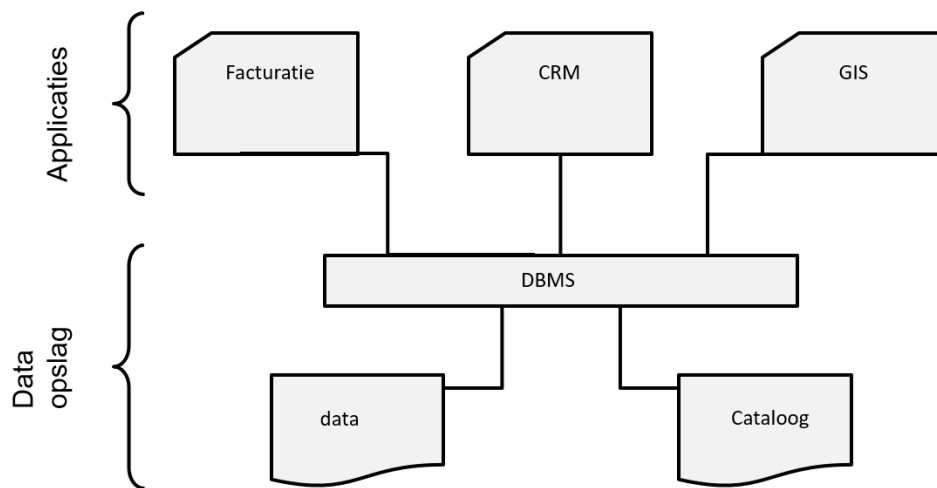
- Beheersen van dataredundantie: Bij het gebruik van een databank wordt er getracht redundantie te vermijden.
- Data consistentie: Door het proberen elimineren van redundantie, wordt automatisch het risico op inconsistente data verkleind. Wanneer een data item maar één keer wordt opgeslagen in de databank, is één update voldoende en is de nieuwe data ineens beschikbaar voor alle gebruikers van de databank.

- Meer informatie uit dezelfde data: Het gebruik van een DBMS laat toe om data op andere manieren te bevragen, waardoor we meer informatie kunnen halen uit dezelfde data.
- Data delen: Bestanden behoren typisch toe aan één gebruiker. Een databank echter behoort toe aan een organisatie en kan gedeeld worden door alle gebruikers die er toegang toe hebben. Op die manier kunnen meerdere gebruikers de data delen.
- Verbeterde data-integriteit: Databank – integriteit verwijst naar de validiteit en consistentie van opgeslagen data. Aan een databank kunnen er constraints toegevoegd worden, dit zijn eisen die opgelegd worden aan de data die niet mogen geschonden worden wanneer nieuwe data toegevoegd wordt of bestaande data aangepast wordt. Er kan bijvoorbeeld afgedwongen worden dat een geboortedatum van een werknemer nooit in de toekomst ligt, of dat de prijs van een product nooit negatief kan zijn. Deze constraints kunnen zowel slaan op individuele records als op relaties tussen records in een databank.
- Verbeterde beveiliging: Databankbeveiliging is het afschermen van de data in de databank tegen ongeoorloofde gebruikers. Dit kan afgedwongen worden door het gebruik van gebruikersnamen en paswoorden voor alle gebruikers van de databank. Sommige gebruikers kunnen toegang hebben tot meer of minder data.
- ...

Nadelen van DBMSen

- Complexiteit: Alle functionaliteit die verwacht wordt van een goed DBMS zorgt ervoor dat dit om een ingewikkeld stuk software gaat.
- Grootte: Om dezelfde reden is een DBMS ook een omvangrijk systeem. Een DBMS vergt heel veel schijfruimte en geheugenruimte om efficiënt zijn werk te kunnen doen.
- Kost: De kost van een DBMS kan heel erg variëren. Een open-source DBMS hoeft niet veel te kosten, maar een professionele DBMS voor heel veel gebruikers kan ook extreem duur zijn. Bovendien is er ook hardware nodig waarop dit DBMS draait, wat op zich ook een kost met zich meebrengt.
- Grote impact bij het falen van het systeem: Het centraal opslaan van data verhoogt de kwetsbaarheid van het systeem. Omdat alle gebruikers en applicaties afhankelijk zijn van de beschikbaarheid van de DBMS, kan het uitvallen van bepaalde componenten een enorme impact hebben op de rest van het systeem.
- Performantie: In het geval van een bestandsgebaseerde databank, worden applicaties geschreven die specifiek gebruik maken van die bestanden, waardoor de performantie vaak zeer goed is. Het doel van een DBMS is meer algemeen, om door heel uiteenlopende applicaties te kunnen gebruikt worden, waardoor de performantie vaak minder goed is.






De voorgaande figuur waarin elke applicatie gebruik maakte van afzonderlijke bestanden, verandert naar de volgende figuur waarbij er gebruik gemaakt wordt van een DBMS. De toepassingen werken nu rechtstreeks met de DBMS in plaats van met hun eigen bestanden. De DBMS levert de gewenste gegevens op verzoek van elke toepassing. De DBMS bewaart en beheert twee soorten gegevens: ruwe data en metadata. Metadata verwijst naar de datadefinities die nu zijn opgeslagen in de catalogus van de DBMS.



RDBMS

Een RDBMS of relationeel databasebeheersysteem (= Relational Database Management System = RDBMS) is een systeem voor het beheren van **relationele** databanken. Een RDBMS moet een gebruiker minimaal in staat stellen om één of meerdere relationele databanken aan te maken en ook de objecten die bij die databank horen, zoals tabellen, relaties, views en queries (waarover later veel meer) te creëren en vlot te gebruiken.

Bekende voorbeelden van RDBMS'en

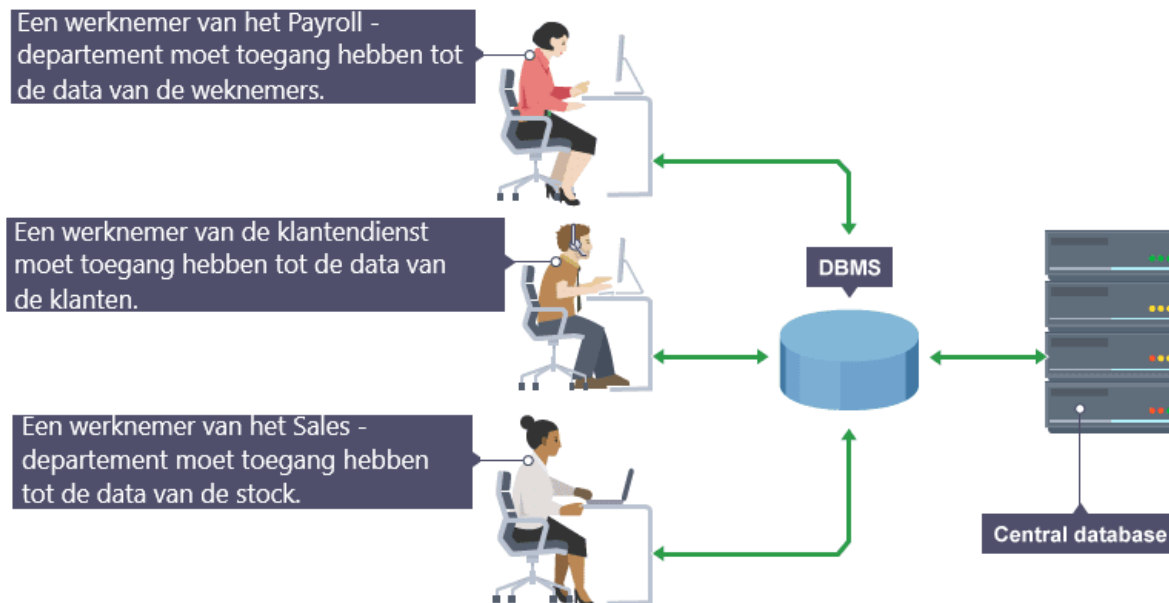
ORACLE		Het grootste en eerste commerciële RDBMS. Wordt gebruikt in veel van 's werelds grootste bedrijven.
MS SQL Server		RDBMS-product van Microsoft. Leverbaar in vele versies voor verschillende bedrijfsbehoeften.
DB 2		RDBMS van IBM. Wordt gebruikt als databank bij mainframes.
MySQL		Het populairste open source RDBMS. Sinds 2010 wordt het ontwikkeld, gedistribueerd en ondersteund door Oracle Corporation.
PostgreSQL		Ook een gratis, open source RDBMS. Sommigen zouden zeggen krachtiger dan MySQL

Databanksysteem

De combinatie van een DBMS en een databank wordt vaak een **databanksysteem** genoemd.

Toepassingsprogramma's

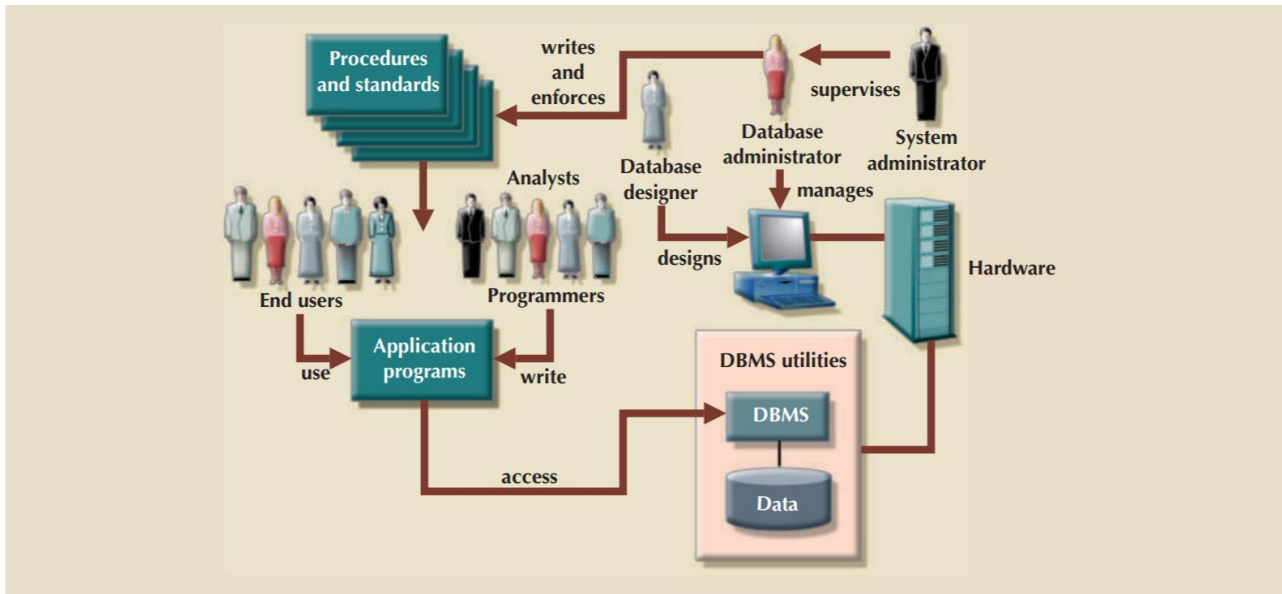
Je kunt op indirecte wijze werken met een databank via toepassingssoftware. De toepassingssoftware zorgt eerst voor de verbinding met het DBMS en deze verzorgt de toegang tot de relevante databanken. Daarna zal het de gebruikersacties die betrekking hebben op de databank vertalen naar databankoperaties, die dan voor verwerking worden doorgestuurd naar het DBMS. Na uitvoering van de operaties stuurt het DBMS de eventuele resultaten terug naar het toepassingsprogramma, dat de resultaten toont aan de gebruiker.



Databankgebruikers

- De data-administrators (DA) zijn die personen die in een onderneming centraal verantwoordelijk zijn voor de data en die meestal een hogere kaderfunctie hebben. De DA beslist onder meer welke data, in welk formaat, in welke databank moet worden opgeslagen. De DA speelt een belangrijke rol bij het ontwerp van een databank. Hij ontwikkelt de modellen, bepaalt de relaties en legt de constraints vast waaraan de databank moet voldoen. Deze functie is cruciaal bij grote conversies of aanpassingen aan de databanken, zoals die rond het millenium en die bij de invoering van de euro.
- De databankontwerper vertaalt het conceptueel model naar logisch en intern model (waarover verder meer).
- De databank administrators (DBA) zijn die personen binnen een onderneming die technisch verantwoordelijk zijn voor de implementatie en het onderhoud van de databanken, in nauw overleg met de DA's. De DBA zorgt ook voor een voldoende hoge performantie van de databank voor de applicaties van de gebruikers en is onder meer verantwoordelijk voor het herstel na falen (back-up en restore).
- De applicatieontwikkelaar staat in voor de ontwikkeling van de toepassingsprogramma's die het toelaten om op een interactieve, gebruiksvriendelijke manier te werken met een databanksysteem. Om de ontwikkelde software te kunnen testen, moeten de toepassingsontwikkelaars kunnen beschikken over een werkkopie van de databanken.

- De eindgebruikers kunnen worden opgedeeld in twee groepen. Enerzijds zijn er de gewone eindgebruikers die zich helemaal niet bewust zijn van de databank. Zij maken gebruik van applicaties die achterliggend de databank gebruiken, bijvoorbeeld de kassierster die een product scant. Anderzijds zijn er de geavanceerde eindgebruikers die bekend zijn met de structuur van de databank en die vrijwel altijd met de databank werken via complexere instructies uit de databanktaal of via verfijnde data.



1.3.1 Elementen van een databanksysteem

Databankmodel

Het is belangrijk een onderscheid te maken tussen de beschrijving van de data, ook nog de data definities genoemd, en de effectieve data. Het **databankmodel** of databank schema beschrijft de data op verschillende detailniveaus en specificeert de verschillende data items die worden opgeslagen in de databank, hun kenmerken, hun onderlinge relaties en details in verband met de opslag. Het databankmodel wordt vastgelegd bij het ontwerpen van de databank en verandert doorgaans niet dikwijls. Het databankmodel wordt opgeslagen in de **catalogoog**, wat het hart is van de DBMS.

De **toestand** van een databank is de data die de databank op dat ogenblik bevat en wijzigt voortdurend, door het toevoegen, verwijderen of bijwerken van data.

Voorbeeld van een databankmodel:

Kunstenaar (naam, geboorteplaats, geboortedatum)

Museum (naam, stad)

Kunstwerk (naam, museum, jaar)

Toestand van een databank:

KUNSTENAAR

naam	geboorteplaats	geboortedatum
Michelangelo	Caprese	06/03/1475
Rembrandt	Leiden	15/07/1606
Picasso	Malaga	25/08/1881

MUSEUM

naam	stad
Sint-Pietersbasiliek	Rome
Museo Reina Sofia	Milaan
Gemäldegalerie van de Staatliche Museen	Berlijn

KUNSTWERK

naam	museum	jaar
Guernica	Museo Reina Sofia	1937
Christuskop	Gemäldegalerie van de Staatliche Museen	1648
Pieta	Sint-Pietersbasiliek	1499

Er zijn 3 data entiteiten: KUNSTENAAR, MUSEUM en KUNSTWERK. Een KUNSTENAAR wordt gekarakteriseerd door een naam, geboorteplaats en geboortedatum. Een MUSEUM wordt gekarakteriseerd door een naam en stad en een KUNSTWERK wordt gekarakteriseerd door een naam, museum en jaar.

datamodel

Een databankmodel bestaat uit verschillende datamodellen, die elk de data vanuit een ander perspectief beschrijven. Een goed datamodel zorgt voor een duidelijke en ondubbelzinnige beschrijving van de data items, de relaties tussen de data items en de constraints die van toepassing zijn.

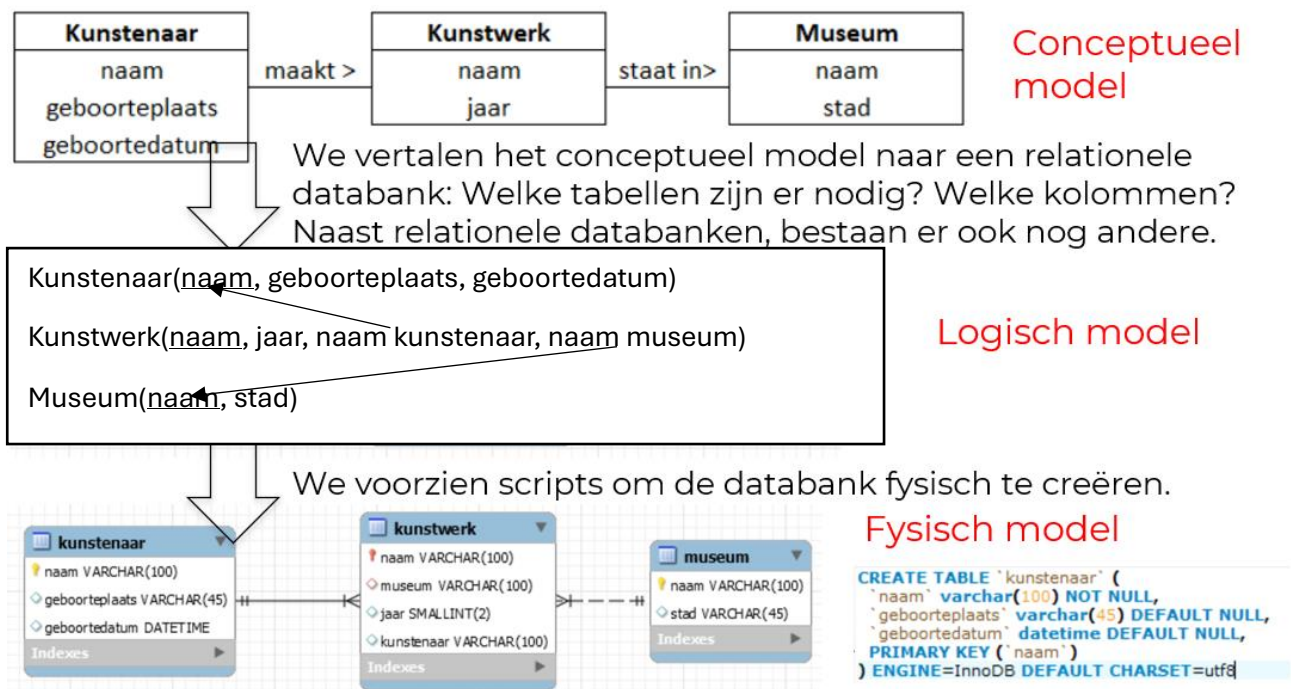
Er zullen in de cursus verschillende datamodellen aan bod komen bij het databankdesign.

Het **conceptueel datamodel** voorziet een beschrijving van de data op hoog niveau (bijvoorbeeld LEVERANCIER, PRODUCT), samen met de karakteristieken (bijvoorbeeld leveranciersnaam, productnummer) en hun onderlinge relaties (bijvoorbeeld een LEVERANCIER kan een PRODUCT leveren). Het is een instrument om te communiceren tussen de informatie architect en de opdrachtgever om zeker te zijn dat de eisen waaraan de data moet voldoen correct worden begrepen en voorgesteld. Het conceptueel model is daarom onafhankelijk van een concrete implementatie van de databank, het moet gebruiksvriendelijk zijn en dicht aanleunen bij hoe de 'business' de data ziet. Het meest gebruikte conceptueel gegevensmodel is het ER model. Het ER model wordt meestal voorgesteld met behulp van een ERD (Entity Relationship Diagram). Het ERD is met andere woorden de basis blauwdruk van de databank. Het ERD wordt gebruikt om het conceptueel gegevensmodel grafisch weer te geven.

Het conceptueel datamodel biedt een aantal belangrijke voordelen. Ten eerste biedt het een beeld van de volledige data dat relatief eenvoudig te begrijpen is. Ten tweede is het conceptueel datamodel onafhankelijk van zowel software als hardware. Software onafhankelijkheid betekent dat het model niet afhankelijk is van de DBMS-software die gebruikt wordt om het model te implementeren. Onafhankelijkheid van hardware betekent dat het model niet afhankelijk is van de hardware die gebruikt wordt om het model te implementeren.

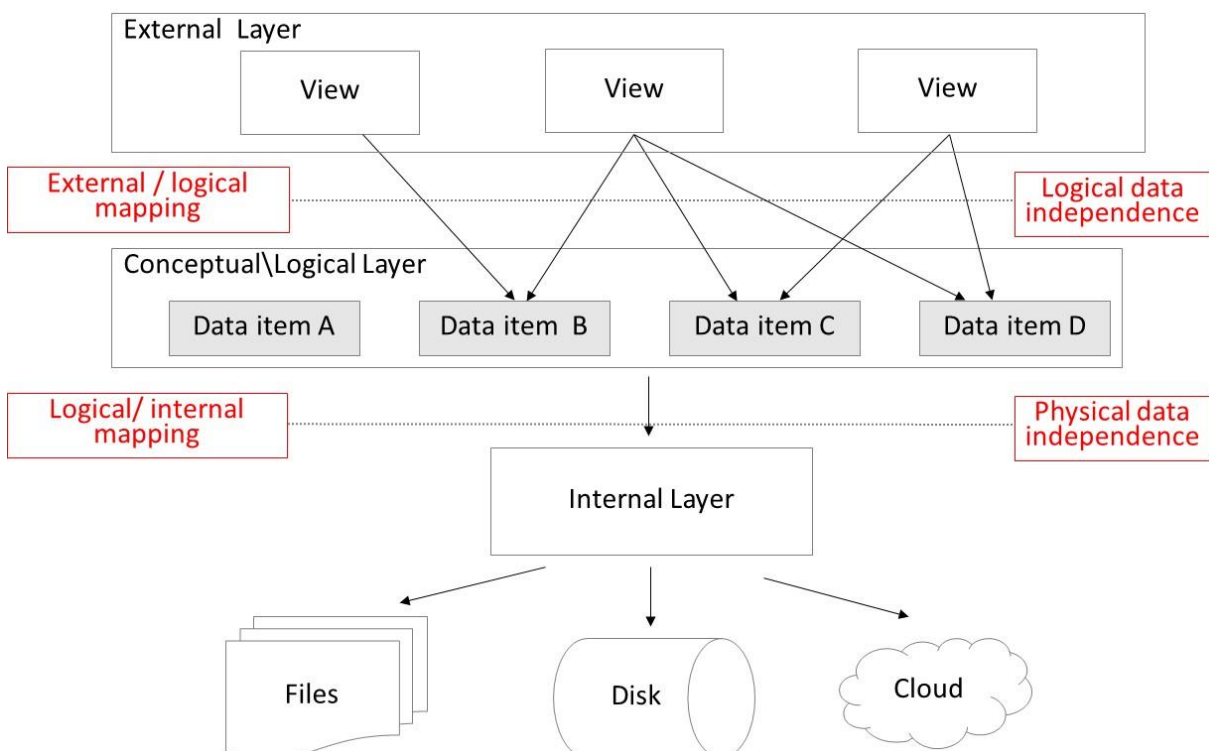
Het **logische databankmodel** is een vertaling ('mapping') van het conceptuele model naar een specifieke omgeving. De data items in het logische model zijn doorgaans nog altijd verstaanbaar voor niet IT-ers, maar leunen al dichter aan bij hoe de data fysiek zal opgeslagen worden. Hier wordt bijvoorbeeld de keuze gemaakt of men een relationele databank zal gebruiken om de data in op te slaan, of een object-georiënteerde databank, of een NoSQL databank.

Het **fysieke databankmodel** beschrijft hoe de data in de databank zal worden opgeslagen: welke data waar zullen worden opgeslagen, wat de grootte is van de datavelden, welke indexen er worden voorzien om het zoeken van de data te vereenvoudigen, ... Deze details hangen zeer nauw samen met de gekozen DBMS.



drielagen architectuur

De hedendaagse databasemanagementsystemen zijn vrijwel allemaal gebouwd volgens een drielagenarchitectuur. Dit betekent echter niet dat er geen andere architecturen bestaan en worden gebruikt. De belangrijkste reden voor een meerlagenarchitectuur is het verkrijgen van dataonafhankelijkheid. De drie lagen van de drielagenarchitectuur noemen we de externe laag, de logische laag en de interne laag.



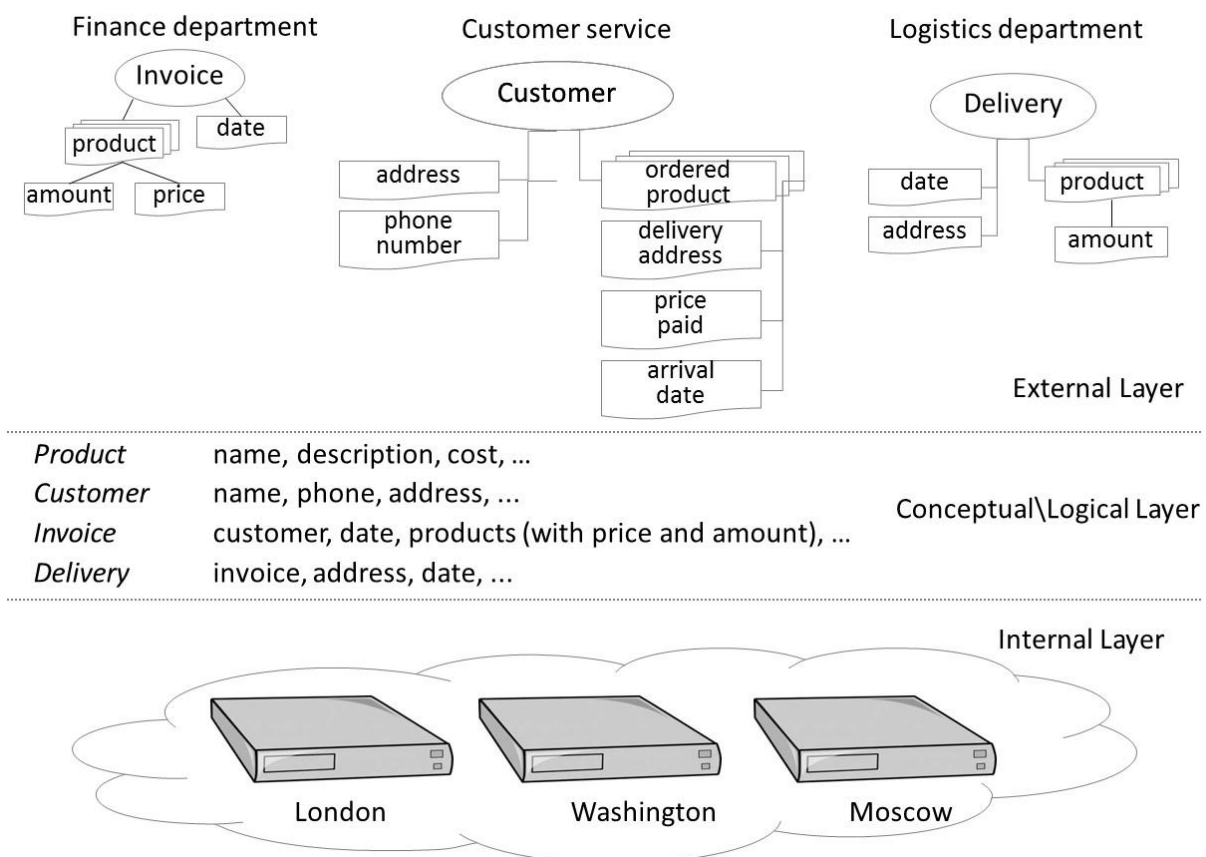
De middelste laag is de conceptuele / logische laag. Hier vinden we het conceptueel en logisch datamodel terug. Beide datamodellen focussen op de karakteristieke eigenschappen van de data items en de onderliggende relaties, zonder te focussen op de fysieke implementatie die afhankelijk is van de gekozen DBMS. Het conceptuele datamodel moet een gebruiksvriendelijk, implementatie-onafhankelijk en transparant datamodel zijn, gebouwd in nauwe samenwerking tussen de informatiearchitect en de zakelijke gebruiker(s). Het wordt verfijnd tot een logisch gegevensmodel op basis van de gekozen DBMS.

De logische laag houdt alle fysieke opslagdetails verborgen, zodat alle aandacht kan gaan naar de abstracte beschrijving van alle in de databank voorkomende entiteiten, verwantschappen tussen entiteiten, gebruikersgedefinieerde operatoren en integriteitsbeperkingen.

In de externe laag hebben we het externe datamodel, dat views bevat. Een view is een venster dat toegang biedt op een zorgvuldig geselecteerd deel van het logische datamodel voor een specifieke groep gebruikers. Een view beschrijft het deel van de databank waarin een bepaalde applicatie of gebruikersgroep geïnteresseerd is en verbergt de rest van de databank. Het wordt gebruikt om de toegang tot gegevens te controleren en beveiliging af te dwingen. De views worden afgestemd op de gegevensbehoeften van een applicatie of (groep van) gebruiker(s). Een view kan één of meerdere applicaties bedienen.

De interne laag bevat het interne datamodel, dat vastlegt hoe data fysisch georganiseerd en opgeslagen wordt.

In het ideale geval, leiden wijzigingen in de ene laag tot slechts minimale wijzigingen in een andere laag. Het zou moeten mogelijk zijn om data fysisch anders te organiseren zonder dat dit een grote impact heeft op de conceptuele / logische laag of de externe laag. Wijzigingen aan de conceptuele / logische laag zouden op hun beurt weinig veranderingen in de externe laag tot gevolg mogen hebben.



De bovenstaande figuur illustreert de drie-lagen architectuur voor een klassiek bedrijfsproces. De conceptuele / logische laag definieert data items zoals PRODUCT, CUSTOMER, INVOICE en DELIVERY. De interne laag bevat de details over hoe en waar de data fysiek opgeslagen worden. De externe laag biedt 3 views aan: een view voor het finance department, voor de customer service en voor het logistics department.

Gegevensonafhankelijkheid

Gegevensonafhankelijkheid wil zeggen dat wijzigingen aan de gegevensbeschrijving weinig tot geen impact hebben op de applicaties.

- Fysieke gegevensonafhankelijkheid: wijzigingen van de opslagspecificaties hebben geen invloed op het logisch model noch op de applicatie. Deze worden opgevangen door het DBMS
- Logische gegevensonafhankelijkheid: minimale aanpassingen aan de applicaties bij wijzigingen aan het logisch model

Gestructureerde, ongestructureerde en semi-gestructureerde gegevens

Het is belangrijk op te merken dat niet alle soorten data kunnen worden beschreven volgens een logisch datamodel. Dit is alleen mogelijk voor **gestructureerde gegevens**. Dit was het enige soort gegevens waar de vroegste DBMS-implementaties zich op focusten. Met gestructureerde gegevens kunnen individuele kenmerken van gegevensitems worden geïdentificeerd en formeel worden gespecificeerd, zoals het nummer, de naam, het adres en het e-mailadres van een student of het nummer en de naam van een cursus. Het voordeel is de mogelijkheid om integriteitsregels uit te schrijven en zo de correctheid van de gegevens af te dwingen. Het vergemakkelijkt ook het zoeken, verwerken en analyseren van gegevens, omdat zowel het DBMS als de toepassingsprogramma's sterke controle over de gegevens hebben.

Bij **ongestructureerde gegevens** zijn er geen onderliggende subcomponenten die op een zinvolle manier geïnterpreteerd kunnen worden door een DBMS of applicatie. Neem een lang tekstdocument met de biografieën van beroemde inwoners van New York. In deze platte tekst is het mogelijk om te zoeken naar de termen "naam", "student" en "New York" die dicht bij elkaar voorkomen, maar het is onmogelijk om te beoordelen of ze betrekking hebben op studenten die in New York woonden, studenten die in New York geboren zijn of misschien zelfs studenten waarvoor de tekst uitlegt dat ze altijd dezelfde trui droegen met de opdruk "New York" erop. Veel recente databasemanagementsystemen bieden faciliteiten om dergelijke full-text documenten efficiënt op te slaan en te doorzoeken. Dit is vooral belangrijk omdat de hoeveelheid ongestructureerde data in de meeste organisaties die van gestructureerde data ruimschoots overtreft. Deze ongestructureerde gegevens kunnen veel nuttige informatie bevatten, als ze efficiënt kunnen worden geëxtraheerd. Denk aan het verbeteren van de interactie met klanten door het opslaan en analyseren van klachtenbrieven, het classificeren van juridische documenten op basis van hun inhoud of het beoordelen van het marktsentiment ten opzichte van een nieuw product door het analyseren van tweets die naar het product verwijzen. Bovendien beperken moderne DBMS'en zich niet alleen tot het opslaan en beheren van ongestructureerde tekstuele gegevens, maar ook van andere soorten gegevens, zoals stilstaande beelden, video en audio.

Tot slot zijn er ook semi-gestructureerde gegevens. Dit zijn gegevens die een bepaalde structuur hebben, maar de structuur kan zeer onregelmatig zijn. Typische voorbeelden zijn de webpagina's van individuele gebruikers op een groot social media platform, of cv-documenten in een personeelsdatabank, die losjes dezelfde structuur kunnen vertonen, maar die niet volledig voldoen aan een enkel, rigide formaat.

Datareduntantie beheren

Één van de belangrijkste nadelen van de bestandsgebaseerde benadering van gegevensbeheer was ongewenste redundantie van gegevens, wat gemakkelijk kan leiden tot inconsistente gegevens.

Het dupliceren van data kan echter wenselijk zijn in gedistribueerde omgevingen omwille van de veiligheid en om de prestaties bij het ophalen van gegevens te verbeteren door lokale toegang tot gegevens mogelijk te maken, in plaats van gebruik te maken van (tijdrovende) netwerkverbindingen. Het DBMS is dan verantwoordelijk voor het beheer van de redundantie door synchronisatiefaciliteiten te bieden om de consistentie van de data te waarborgen. Een update van een lokale gegevenskopie wordt bijvoorbeeld automatisch doorgevoerd naar alle duplicaatgegevens die op andere locaties zijn opgeslagen.

Integriteitsregels

Integriteitsregel kunnen expliciet worden gedefinieerd. Deze regels kunnen gebruikt worden om de correctheid van de gegevens af te dwingen. Syntactische regels specificeren hoe de gegevens opgeslagen moeten worden. Bijvoorbeeld, customerID moet worden weergegeven als een geheel getal (bijv. 100, 125 en 200 zijn correct, maar 1.20 of 2a niet); geboortedatum moet worden opgeslagen als maand, dag en jaar (bijv. 02/27/1975 is correct, maar 27/02/1975 niet). Semantische regels richten zich op de semantische correctheid of betekenis van de gegevens. Bijvoorbeeld, klantID moet uniek zijn; rekeningsaldo moet groter zijn dan 0; en een klant kan niet worden verwijderd als hij/zij nog openstaande facturen heeft. Deze integriteitsregels worden gespecificeerd als onderdeel van het conceptuele/logische datamodel en centraal opgeslagen in de catalog. Dit verbetert de efficiëntie en onderhoudbaarheid van de applicaties aanzienlijk omdat de integriteitsregels direct worden afgedwongen door de DBMS wanneer er iets wordt bijgewerkt.

Catalog

De catalog is het hart van de DBMS. Het bevat de datadefinities, of metadata, van je databasetoepassing. Het slaat de definities op van de views, logische en interne datamodellen en synchroniseert deze drie datamodellen om hun consistentie te garanderen. Het is een opslagplaats voor integriteitsregels en andere informatie zoals gebruikers, ...

1.4 'Goede' databank

Voorwaarden voor een goede databank:

- Vanuit het perspectief van de gebruikers:
 - betrouwbaar: de gegevens moeten up-to-date en juist zijn.
 - volledig: de databank moet alle gegevens bevatten die de applicatie(s) nodig hebben.
 - efficiënt: een gebruiker moet antwoord krijgen binnen een 'redelijke' termijn.
 - verstaanbaar: de gegevens moeten voldoen aan vooraf bepaalde eisen voor verstaanbaarheid. Voor bepaalde specialistische doeleinden kan het zijn dat gebruikers scholing nodig hebben, maar in principe zouden ze zonder scholing in hun eigen taal met de databank moeten kunnen werken.
- Vanuit het perspectief van de ontwerpers, bouwers en beheerders:
 - testbaar: de applicaties op een databank moeten goed te debuggen zijn, er moeten hulpmiddelen zijn om de software te ontdoen van fouten.
 - aanpasbaar: er kunnen altijd wijzigingen in gegevensstructuren en applicaties nodig zijn. Het mag niet onnodig moeilijk zijn deze wijzigingen aan te brengen.
- Vanuit het perspectief van de organisatie als geheel:
 - apparatuuronafhankelijk: organisaties kopen nieuwe computertypes, of fuseren, met de regelmaat van de klok. Databanken moeten daartegen bestand zijn: het database

management systeem moet op allerlei hardware en onder allerlei besturingssystemen kunnen draaien.

- organisatieonafhankelijk: wanneer er fusies of samenwerkingsverbanden ontstaan, worden dikwijls gegevens gedeeld tussen de betrokken organisaties. Het is dan erg waardevol wanneer de gegevensdefinities die men hanteert met elkaar overeenstemmen.

1.5 Geschiedenis

Bronnen

- <https://www.youtube.com/watch?v=KG-mqHoXOXY> → Zeker de moeite om dit filmpje eens te bekijken!
- <https://www.timetoast.com/timelines/dbms>

In de voorgaande tekst kwam reeds aan bod hoe bestandsgebaseerde systemen de voorloper zijn van DBMS'en. Het is echter niet zo dat er een specifiek tijdstip is waarop de bestandsgebaseerde aanpak is opgehouden te bestaan en de DBMS'en zijn ontstaan. Welintegendeel, zelfs tot op vandaag worden er nog bestandsgebaseerde systemen gebruikt, zoals bij de HP-NonStop mainframe.

Er wordt gesuggereerd dat de DBMS zijn oorsprong vindt in het Apollo maanlandingsproject uit de jaren 1960. Dit project werd gestart in het kader van de wens van President Kennedy om tegen het einde van het decennium een eerste mens te laten landen op de maan. Op dat ogenblik was er nog geen systeem voor handen dat zou toelaten om de enorme hoeveelheid informatie afkomstig van dit project, te beheren.

Tot dan toe werd data opgeslagen in individuele, platte bestanden: ze hielden een eenvoudige, sequentiële lijst bij van records. Voor elke zoekopdracht moest de lijst van records één voor één overlopen worden. Dit was een trage manier om alle data te doorzoeken en het was niet gemakkelijk om grote hoeveelheden data te onderhouden. Er was nood aan databanken die snel, efficiënt en betrouwbaar waren.

In 1964 ontwikkelde IBM voor North American Rockwell (de belangrijkste contractant van het Apollo project) GUAM, wat staat voor Generalized Update Access Methode. GUAM is een hiërarchisch model en was gebaseerd op het concept dat kleinere componenten samengebracht kunnen worden als onderdelen van grotere componenten, ... totdat het uiteindelijke product volledig is samengesteld. Deze structuur stemt overeen met de vorm van een boom ondersteboven.

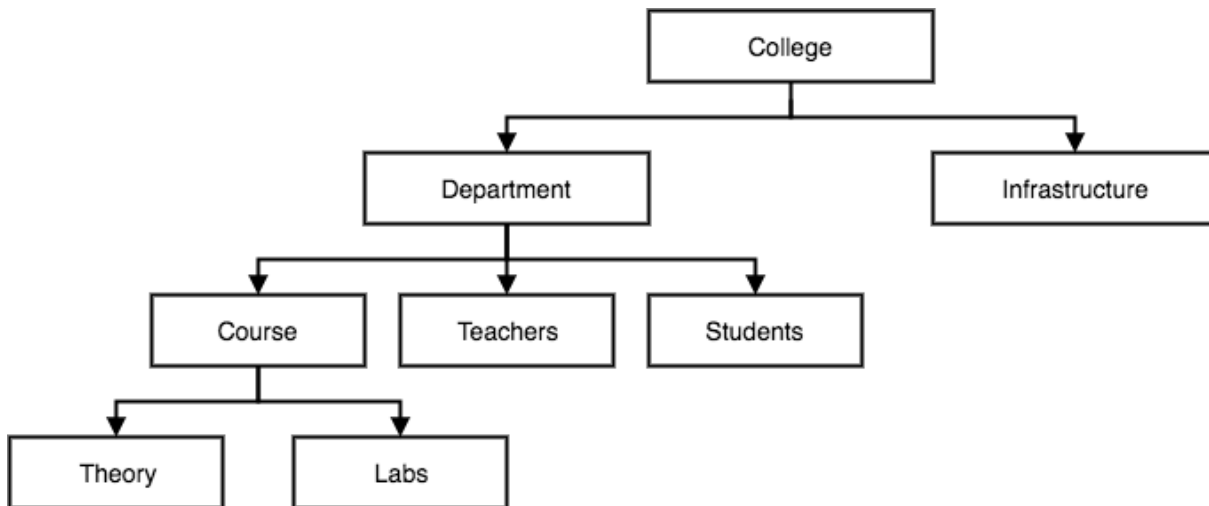
Kenmerken van een hiërarchisch model:

- Elk record in een databank (parent) kan verwijzen naar een n-aantal andere records (children).
- Elk recordtype heeft één en niet meer dan één eigenaar (owner).
- Het hiërarchische model kent maar één boomstructuur per databank.
- De takken hebben zijdelings geen samenhang (alleen parent en children).
- De enige ingang (root) van de boomstructuur is van bovenaf.

Nadelen:

- Er bestaan enkel één-op-veel verbanden.
- De gegevens zijn niet direct toegankelijk. Navigatie is enkel mogelijk via de parent-child-relaties.

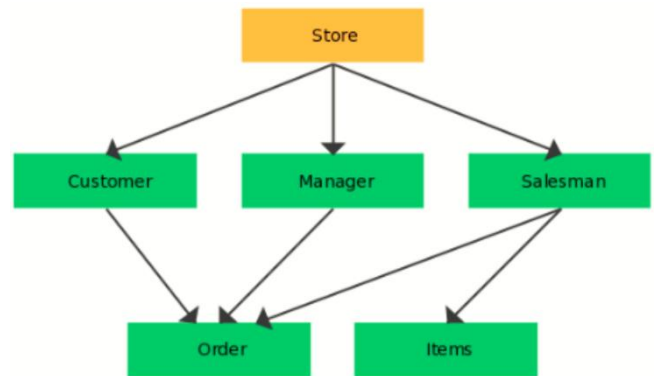
IBM ontwikkelde GUAM verder tot IMS (Information Management System). De reden waarom IBM IMS beperkte tot het gebruik van hiërarchieën van records was omdat dit toeliet seriële opslagapparaten te gebruiken, meestal magnetische tape.



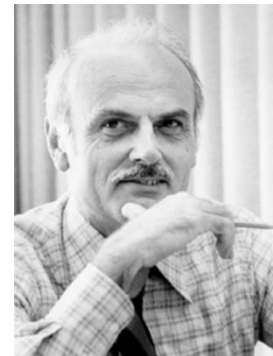
Chales Bachman, die werkzaam was bij General Electric, ontwierp een meer flexibel model: de hiërarchische structuur werd vervangen door een netwerkmodel en de kinderen in de boom konden meerdere ouders hebben. Eens de databank ingewikkeld werd, werd het echter moeilijk om al deze pointers te beheren. Er kon bijvoorbeeld ook een probleem ontstaan wanneer ergens in de databank een pointer corrupt raakte. Hierdoor kunnen ganse delen van de databank opeens niet meer bereikbaar zijn.



In 1969 legt de CODASYL (Conference on Data Systems Languages) Databank Task Group (DBTG) de standaarden vast voor netwerkdatabanken. De nevenstaande figuur, toont een voorbeeld van een netwerkddatabank.



E.F. (Ted) Codd, een computerwetenschapper die aan de slag was bij IBM, had een beter idee. Hij schreef in 1970 een paper over het relationeel datamodel ('A relational model of data for large shared data banks'). Daarin paste hij concepten uit de relationele algebra toe op het opslaan van grote hoeveelheden data. In een relationeel datamodel wordt data georganiseerd met behulp van eenvoudige tabellen die gerelateerde informatie bevatten. Er wordt niet langer gebruik gemaakt van pointers, maar de tabellen zijn verbonden via overeenkomstige datavelden. Er wordt gebruik gemaakt van **vreemde sleutels** om de logische in plaats van fysieke verbanden tussen de gegevens bij te houden. Dit maakt het gemakkelijker om data terug te vinden, samen te voegen en te wijzigen.



Relational Model

Movies		
Movie Name	Studio ID	Actor ID
Towering Inferno	20F	31
The Godfather	PP	22
Chinatown	PP	14
Straw Dogs	ABC	57

Actors		
Actor ID	Actor Name	Born
57	Dustin Hoffman	1937
42	Al Pacino	1940
14	Jack Nicholson	1937
22	Diane Keaton	1946

Studios			
Studio ID	Studio Name	Founded	Headquarters
PP	Paramount Pictures	1912	Hollywood
UA	United Artists	1919	Los Angeles
20F	20th Century Fox	1935	Century City

Doorheen de jaren '70 werd er veel onderzoek gedaan met betrekking tot het relationeel model. Initieel was er veel weerstand aangaande dit model.

C.J. Date, een IBM instructeur en auteur, was echter overtuigd van de relationele databank als het betere model. Codd en Date schreven meerdere papers, gaven lezingen en debatteerden over de voordelen van het relationeel model. Maar Codd's relationele model stond toen in concurrentie met een ander IBM product, namelijk het winstgevende IMS. IBM was daardoor initieel niet zo te vinden voor dit nieuwe, relationeel model.



How Relational Databases Work

Computerized databases help people store and track huge amounts of information. The smallest unit of information in a database is called a **field**. Fields are grouped together to form **records**. Records are then grouped together to form **tables**.

Flat-file databases take all the information from all the records and store everything in one table. This works fine when you have a small number of records related to a single topic, such as a person's name and phone number, but if you have hundreds or thousands of records, each with a number of fields, the database quickly becomes difficult to use.

Relational databases separate this mass of information into numerous tables. All the columns in each table should be about one topic, such as "student information," "class information," or "trainer information."

The tables for a relational database are linked to each other through the use of keys. Each table may have one **primary key** and any number of **foreign keys**. A foreign key is simply a primary key from one table that has been placed in another table.

The most important rules for designing relational databases are called **Normal Forms**.

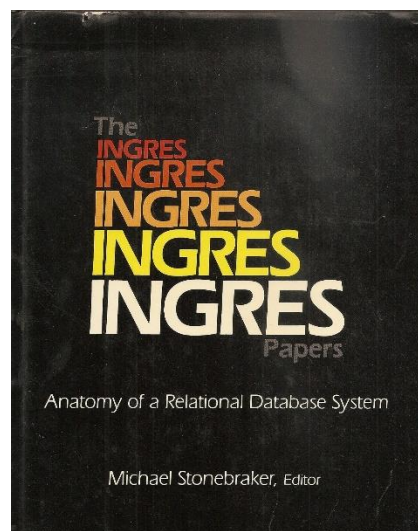
When databases are designed properly, huge amounts of information can be kept under control. This lets you query the database (search for information) and quickly get the answer you need.

Query: "What students are taking classes from trainer CHARLES HILL?"

Answer:

1	Mary	Hinkle	555.123.4567
2	Paul	Litz	555.258.8963

Copyright © Design by Fred Schneider



In 1973, maakte Michael Stonebraker's team aan UC Berkeley gebruik van Codd's idee om de INGRES relationele databank te creëren die tegen een beperkte vergoeding vrij verkrijgbaar was. Verschillende bedrijven gebruikten het als basis voor succesvolle commerciële producten.

Het duurde tot 1975 vooraleer IBM een experimentele, relationele databank had gecreëerd, met de naam System /R. Er werd gebruik gemaakt van SEQUEL, wat een taal is om op gestructureerde manier databank query's voor het opzoeken en wijzigen van data, te formuleren. SEQUEL is een acroniem voor Structured English Query Language en werd ontwikkeld door Don Chamberlin en Raymond Boyce, beiden IBM medewerkers.

Een jonge ondernemer, genaamd Larry Ellison, werd ook geïnspireerd door Codd's ideeën. In 1977 maakte hij gebruik van \$2000 van zijn eigen kapitaal om een nieuw software bedrijf met de naam Relational Software Laboratories op te richten, samen met Bob Miner en Ed Oates: een aantal ingenieurs uit California. In 1979 veranderde de naam naar Relational Software Inc. Het was hun doel om de eerste commercieel beschikbare relationele databank te ontwikkelen en te verkopen, die compatibel was met IBM's System /R. Hun belangrijkste product was 'Oracle', ontstaan in 1979. De eerste versie van Oracle draaide op minicomputers, maar tegen 1983 had het bedrijf Oracle herschreven om ook te kunnen draaien op andere computersystemen waaronder IBM PC's en mainframes. Oracle werd al snel winstgevend en in 1983 werd de naam van Relational Software Inc. omgedoopt naar Oracle Systems Corporation, het 'flagship' product van het bedrijf.

In 1980 verlieten een aantal academici Berkeley en richtten het bedrijf Relational Technology Inc. op met de bedoeling een commerciële versie van Ingres te bouwen. Ingres en Oracle waren aartsrivalen. En hoewel Ingres op verschillende vlakken superieur was tegenover Oracle, moest Ingres het afleggen tegen Oracle, onder meer door Oracle's agressieve marketingstrategie

In 1983 bracht IBM uiteindelijk een volwaardige commerciële relationele databank uit, namelijk DB2 voor mainframes, maar het was inmiddels te laat voor IBM om nog de markt van de minicomputers te kunnen domineren. Oracle verkocht zijn relationele databank toen al aan IBM klanten. Oracle had veel vroeger het concept van relationele databanken van Ted Codd omarmd, in tegenstelling tot de eigen werkgever van Ted Codd, namelijk IBM. Op die manier werd Oracle één van de grootste software bedrijven ter wereld. Daardoor werden relationele databanken de belangrijkste instrumenten om data op een computer te organiseren.

In de late jaren '80 wint SQL aan populariteit.

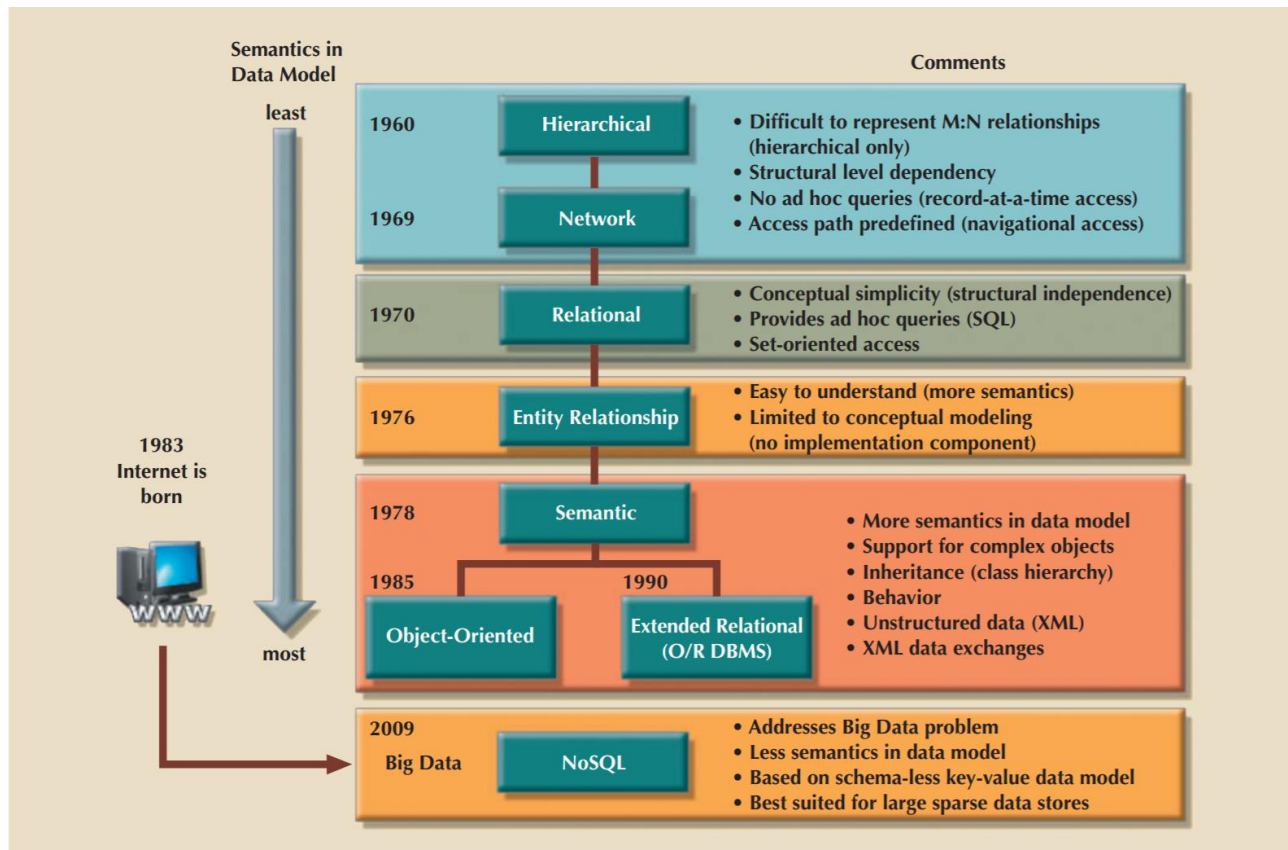
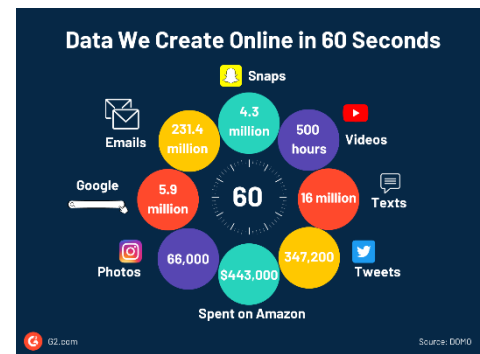
Begin de jaren '90 maakten gestaag verbeterde SQL-implementaties en drastische verbeteringen in processorsnelheden SQL tot een praktische oplossing voor transactieverwerkingstoepassingen. Tegelijk kende het object georiënteerd programmeren begin de jaren '90 een enorme opgang. Men verwachtte een parallelle beweging naar object-relationele databanken (dit zijn relationele DBMSen met objectgeoriënteerde faciliteiten) en object-georiënteerde databanken (OODBMS). SQL en de relationele databanken hebben de uitdaging tot nu toe echter meer dan doorstaan.

De enorme hoeveelheden data die de laatste jaren geproduceerd worden, o.a. door de sociale media, en de typische eigenschappen van deze data (foto's, video's, ...) hebben bijgedragen tot de groei van NoSQL (Not Only SQL) databanken. Het is belangrijk om in te zien dat de klassieke, relationele databanken zoals ze ontwikkeld werden begin de jaren '70 nooit bedoeld waren om de dagelijkse, gigantische stroom van Instagram-posts, WhatsApp-berichten, Facebook-updates, ... in op te slaan.

In 2024 was één minuut goed voor:

- Er worden 5,9 miljoen Google-zoekopdrachten uitgevoerd.
- Instagram-gebruikers delen 66.000 foto's.
- Facebook-gebruikers posten 1,7 miljoen stukjes content.

- Mensen versturen 231,4 miljoen e-mails.
- YouTubers uploaden 500 uur video's per minuut
- Snapchat gebruikers versturen 4,3 miljoen snaps.
- Twitter-gebruikers schrijven 347.200 tweets.
- Mensen versturen 16 miljoen sms'jes.
- Venmo-gebruikers maken \$ 437.600 over.
- Amazon shoppers geven 443.000 dollar uit.



1.6 Oefeningen

Oefening 1

Kies Waar of Vals. Indien Vals, verbeter de uitspraak dan op een niet triviale manier.

Uitspraak	Waar	Vals
Het werk van E.F. Codd in het begin van de jaren '70 leidde tot de ontwikkeling van de relationele databanken.		
Voor het conceptueel model is het niet belangrijk hoe het fysieke model zal geïmplementeerd worden.		
Het fysieke model is afhankelijk van de gebruikte hard – en software.		
In termen van het effect op een applicatieprogramma, is het fysiek model het meest belangrijke model.		
De DBMS fungeert als schakel tussen de gebruiker en de databankapplicatie.		
De middelste laag van de drie-lagen architectuur bestaat uit het conceptuele datamodel en het logische datamodel. Het logische datamodel wordt fysiek geïmplementeerd in de interne laag.		
De bovenste laag van de drie-lagen architectuur is de externe laag. Views voor één of meerdere applicaties bieden een zicht op het volledige logische model.		

Fysieke data-onafhankelijkheid betekent dat noch de applicaties, noch de views, noch het logisch datamodel moeten veranderd worden als er wijzigingen worden aangebracht aan de specifieke manier waarop data wordt opgeslagen in de interne laag.		
Logische data-onafhankelijkheid betekent dat applicaties slechts minimaal gewijzigd moeten worden wanneer er wijzigingen zijn in het conceptuele of logische datamodel.		

Oefening 2

Selecteer alle juiste antwoorden

Wat is het verschil tussen informatie en data?

- ☐ Data wordt omgezet naar nuttige informatie. Het wordt opgeslagen in een databank en is toegankelijk voor systemen en gebruikers.
- ☐ Er zijn geen verschillen tussen data en informatie. Het zijn 2 woorden voor hetzelfde.
- ☐ Data wordt bijgehouden en begrepen enkel door gebruikers.
- ☐ Informatie wordt bijgehouden en begrepen enkel door gebruikers.

Wat zijn de voordelen van het gebruik van een DBMS

- ☐ Vermijden van gegevensredundantie en het voorkomen van inconsistenties
- ☐ Onafhankelijkheid van de gebruikte software
- ☐ Flexibele toegang tot gegevens die kunnen gedeeld worden door meerdere gebruikers
- ☐ Maakt een gecentraliseerde controle van de data mogelijk

Welke eigenschappen zijn van toepassing op hiërarchische databanken?

- ☐ Relaties tussen records vormen een hiërarchie of boomachtige structuur.
- ☐ Records zijn afhankelijk en gerangschikt in structuren op meerdere niveaus, bestaande uit één rootrecord en een willekeurig aantal ondergeschikte niveaus.
- ☐ Relaties tussen de records zijn één-op-veel, omdat elk data-element slechts gerelateerd is aan één element erboven.
- ☐ Data-element of record op het hoogste niveau van de hiërarchie wordt het root-element genoemd. Elk gegevenselement kan worden gevonden door vanaf de wortel en langs de takken van de boom geleidelijk naar beneden te bewegen totdat het gewenste record is gevonden.
- ☐ Redundantie is mogelijk

Welke is geen verantwoordelijkheid van de DBA?

- ☐ Backups voorzien
- ☐ Bepaalt de tabellen, attributen en relaties tussen de tabellen
- ☐ Stelt een noodherstelplan op voor de databank

Welke is geen eigenschap van data modellering?

- ☐ Het is een korte, eenvoudige en ondubbelzinnige beschrijving van een gebruik of een procedure die gangbaar is binnen een specifieke organisatie
- ☐ Het wordt gebruikt om de interactie tussen designers, programmeurs en eindgebruikers te vergemakkelijken.
- ☐ Het wordt gebruikt om discrepanties tussen het databankontwerp en de echte wereld waarin de data-objecten bestaan te elimineren of om zijn minst te verminderen.
- ☐ Het is een eenvoudige voorstelling - vaak grafisch - van objecten die bestaan in de echte wereld

Welk is geen voordeel van het gebruik van een DBMS?

- ☐ Data inconsistentie minimaliseren
- ☐ DBMS providers upgraden hun producten vaak door het voorzien van bijvoorbeeld nieuwe features. Dit impliceert dat de software moet geüpgraded worden of zelfs dat er hardware upgrades nodig zijn.
- ☐ Data kan gemakkelijker geïntegreerd worden
- ☐ Data is gemakkelijker toegankelijk
- ☐ Vendor dependency: er moet veel geïnvesteerd worden in technologie en opleiding, daarom zijn gebruikers zelden geneigd om te veranderen van DBMS provider
- ☐ Data kan gemakkelijker gedeeld worden door verschillende gebruikers.
- ☐ Een DBMS garandeert goed data management

Welke software wordt tegenwoordig vaak gebruikt als DBMS

- ☐ Oracle
- ☐ MS SQL Server
- ☐ Linux
- ☐ MySQL

E.F. Codd ontwikkelde het relationele model in de

- ☐ 1960s
- ☐ 1970s
- ☐ 1980s
- ☐ 1990s

Het conceptueel ontwerp

- ☐ is een manier om de databank te documenteren
- ☐ heeft nood aan cijfers met betrekking tot data volume en hoe frequent de data zal opgevraagd worden, om de grootte van de databank te bepalen
- ☐ omvat modellering onafhankelijk van het DBMS
- ☐ wordt enkel gebruikt in relationele databanken

Het conceptueel model is

- ☐ afhankelijk van de hardware
- ☐ afhankelijk van de software
- ☐ afhankelijk van de hardware en de software
- ☐ onafhankelijk van de hardware en de software

Voor welk van de volgende is er specifieke kennis van een DBMS nodig?

- ☐ Conceptueel ontwerp
- ☐ Logisch ontwerp
- ☐ Fysiek ontwerp
- ☐ Databank ontwerp

Het logisch ontwerp

- ☐ beschrijft de volledige databank
- ☐ is een standaard manier om informatie te organiseren in gemakkelijk toegankelijke delen
- ☐ beschrijft hoe de data effectief opgeslagen wordt op de schijf
- ☐ alle voorgaande

Welke van de volgende stellingen is niet correct?

- ☐ gegevensmanagement via bestanden zorgt ervoor dat dezelfde informatie voor verschillende applicaties apart opgeslagen wordt
- ☐ gegevensmanagement via bestanden zorgt ervoor verschillende applicaties tegelijk oudere en nieuwere versies van dezelfde data kunnen gebruiken
- ☐ bij gegevensmanagement via bestanden, kan een verandering in de structuur van de data in een bestand gemakkelijk opgelost worden aangezien elke applicatie over zijn eigen bestanden beschikt

Welke van de volgende stellingen is niet correct?

- ☐ wanneer er gebruik gemaakt wordt van een databank, beschikken applicaties niet langer over hun eigen bestanden, maar hebben alle applicaties toegang tot dezelfde versie van de data door middel van het DBMS
- ☐ wanneer er gebruik gemaakt wordt van een databank, is er doorgaans minder opslagruimte nodig dan in het geval van gegevensmanagement via bestanden
- ☐ wanneer er gebruik gemaakt wordt van een databank, is het onderhoud van de data en de metadata eenvoudiger

Welke van de volgende stellingen is niet correct?

- ☐ in het conceptueel datamodel, worden de vereisten voor de data van een bedrijf vastgelegd en gemodelleerd
- ☐ een conceptueel datamodel is implementatie-afhankelijk
- ☐ een logisch datamodel vertaalt het conceptueel model naar een specifieke implementatie-omgeving
- ☐ voorbeelden van de mogelijke implementaties van het logische datamodel zijn het hiërarchisch, relationeel, ... datamodel

Wat is de verzamelnaam voor de verschillende data items, hun karakteristieke eigenschappen, relaties, beperkingen, ... en wordt gespecificeerd tijdens het databank design?

- ☐ databank model
- ☐ catalogoog
- ☐ conceptueel model
- ☐ intern model
- ☐ geen enkel van de voorgaande

In de drie-lagen architectuur, tussen de externe laag en de conceptuele laag

- ☐ is er fysieke data onafhankelijkheid
- ☐ is er logische data onafhankelijkheid
- ☐ is er geen onafhankelijkheid: ze zijn in wezen hetzelfde
- ☐ bevindt zich de interne laag

Oefening 3

In de tekst kwam aan bod dat het gebruik van bestanden geen goed idee is om data bij te houden, maar dat het beter is om te kiezen voor een DBMS.

Iemand besluit dit advies te negeren en een tabelletje te creëren in Excel.

Illustreer 3 verschillende redenen waarom het gebruik van bestanden geen goed idee is om data op te slaan aan de hand van de onderstaande tabel.

PROJECT_CODE	PROJECT_MANAGER	MANAGER_PHONE	MANAGER_ADDRESS	PROJECT_BID_PRICE
21-5Z	Holly B. Parker	904-338-3416	3334 Lee Rd., Gainesville, FL 37123	16833460.00
25-2D	Jane D. Grant	615-898-9909	218 Clark Blvd., Nashville, TN 36362	12500000.00
25-5A	George F. Dorts	615-227-1245	124 River Dr., Franklin, TN 29185	32512420.00
25-9T	Holly B. Parker	904-338-3416	3334 Lee Rd., Gainesville, FL 37123	21563234.00
27-4Q	George F. Dorts	615-227-1245	124 River Dr., Franklin, TN 29185	10314545.00
29-2D	Holly B. Parker	904-338-3416	3334 Lee Rd., Gainesville, FL 37123	25559999.00
31-7P	William K. Moor	904-445-2719	216 Morton Rd., Stetson, FL 30155	56850000.00

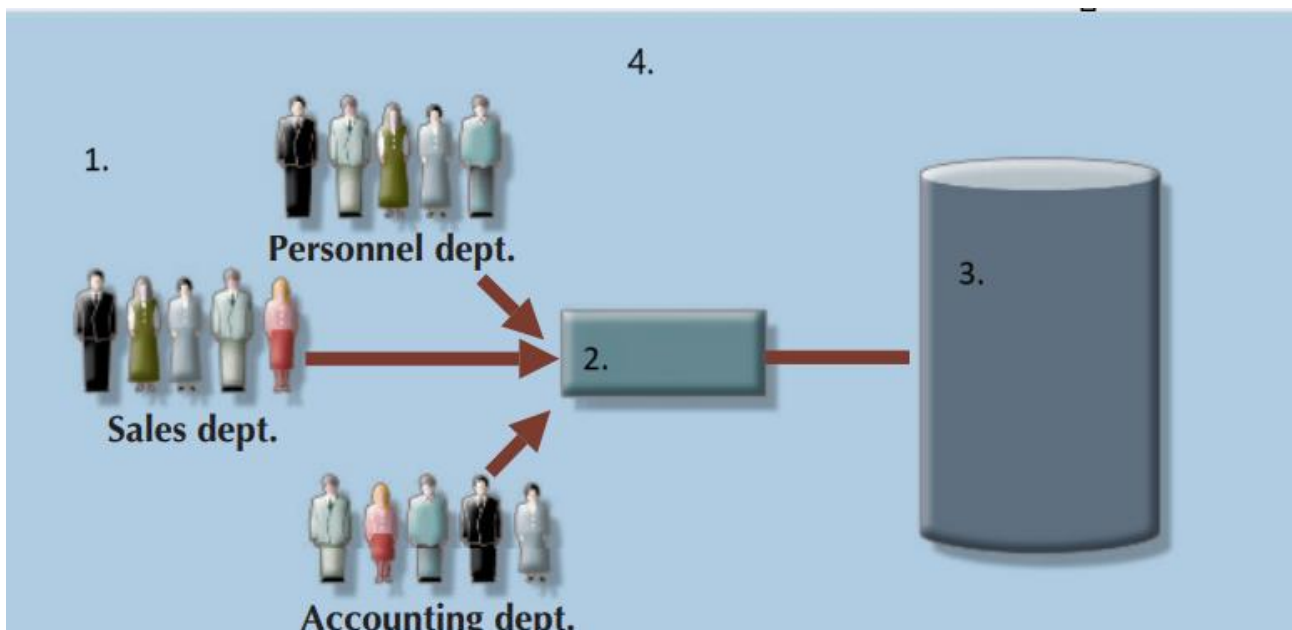
Oefening 4

Geef het correcte begrip

Een programma dat een service aanvraagt.	
Een persoon of programma dat gebruik maakt van de databank.	
Een programma dat een service aanbiedt aan aanvragende programma's.	
Er zijn geen inconsistenties in de gegevens en de data zal consistente resultaten opleveren.	
Een toestand waarin de toegang tot de data niet wordt beïnvloed door veranderingen in de fysieke gegevensopslag	

Oefening 5

Geef voor 1 + 2 + 3 + 4 telkens de juiste term.



1.7 Bronnen

Principles of DATABASE MANAGEMENT [Wilfried Lemahieu / Seppe Vanden Broucke / Bart Baesens]

Principes van databases [Guy De Tré]

DATABASE SYSTEMS Design, Implementation, and Management [Carlos Coronel / Steven Morris]