GEGEVENSBANKEN EXAM 2023-2024

Prof B. Baessens

Liam Luys

KU Leuven

Veel succes met het examen!
Ik heb geprobeerd een samenvatting te maken die alle info bevat uit de cursustekst/slides.
Geen garantie dat alles er in staat en/of juist is, doe zeker de boeken ook open.
Foutje gevonden? Laat het zeker weten, dan pas ik het aan!

Les 1 - Fundamental Concepts

Liam Luys – r0934449 - 2e BA Informatica
Gegevensbanken

dinsdag 13 februari 2024 17:19

Fundamental Concepts of Database Management

Applications of Database Technology

Toepassingen Database
- Opslaan en verkrijgen van (alfa)numerieke data in inventarisprogramma
- Multimedia Programma's (YouTube, Spotify, …)
- Biomedische Applicaties (Vingerafdrukken, Scans, …)
- …

Key Definitions

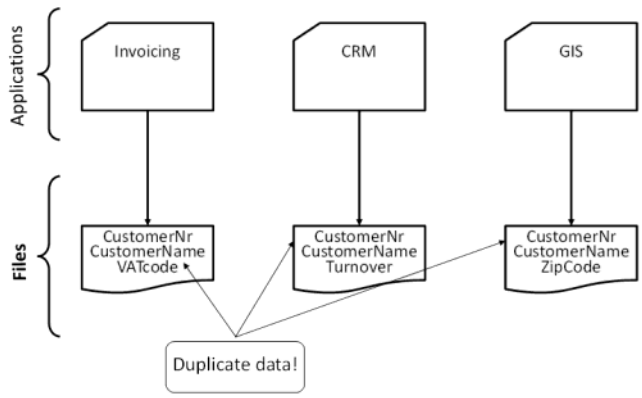
Database
- Verzameling van gerelateerde data-items binnen een specifiek bedrijfsproces of probleemomgeving
- Wordt gebruikt door bepaalde groep mensen

Database Management System DBMS
- Softwarepakket gebruikt om een database te ontwikkelen, onderhouden en definieren
- Opgebouwd uit verschillende modules

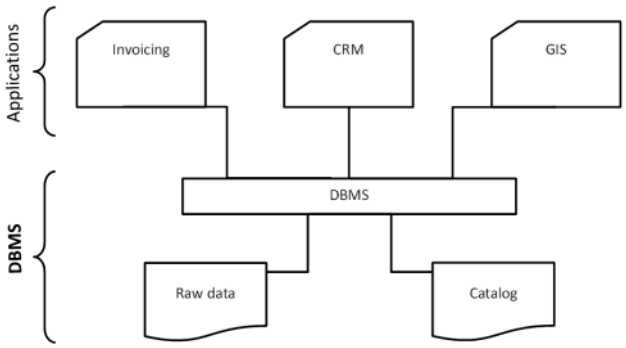
Database Systeem
- Combinatie database en DBMS

Bestand vs Database Approach to Data Management

File Approach
- Gegevens opslaan in aparte bestanden
- Duplicate informatie wordt bewaart
  - Inconsistentie gevaar!
- Sterke koppeling tussen applicaties en data
- Moeilijk om gelijktijdigheid te controleren
- Moeilijk om met meerdere gebruikers te werken



Database Approach
- Gegevens gezamenlijk opslaan
- Efficiënter en consistenter dan file approach
- Zwakke koppeling tussen applications en data
- Mogelijkheden voor data querying (SQL)



Elements of a Database System

Database Systeem Onderdelen
- Database model / Instances
- Data Model
- Three Layer Architecture
- Catalog
- Database User
- Database Language

Database Model
- Beschrijving van database gegevens vanuit bepaald perspectief
    - Eigenschappen, relaties, voorwaarden, opslagdetails
- Opgesteld tijdens database design, wijzigt niet
- Opgeslagen in Catalog

Soorten Modellen
- Conceptueel Model
    - Hoog-niveau beschrijving van relaties en eigenschappen
    - Implementatie-onafhankelijk, gebruiksvriendelijk
    - Opgesteld in EER (Enhanced-Entity Relationship) of object-georienteerd model

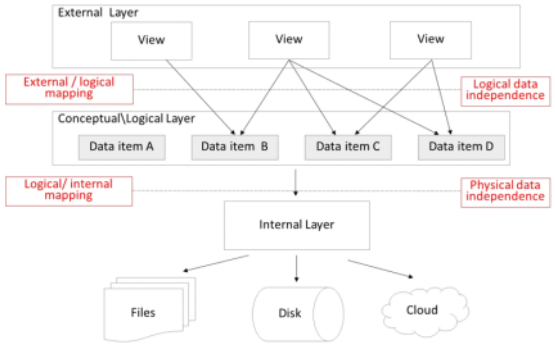
- Logisch Model
    - Vertaling van conceptueel model naar implementatie-omgeving
    - Hiërarchisch, CODASYL, relationeel, noSQL, …

- Intern Model
    - Stelt fysieke opslagdetails van data voor
    - Welke data wordt waar opgeslagen? Welk formaat?
    - Specifiek voor DBMS

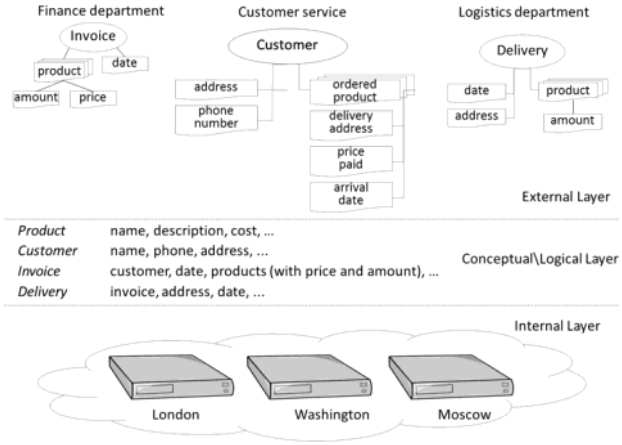
- Extern Model
    - Deelverzameling van data in logisch model
    - Views voor specifieke applicaties/gebruikers

Database State
- De data in de database op een specifiek moment
- Instance / Current set / Snapshot van data

Three Layer Architecture
- Interne Laag
- Conceptuele/Logische Laag
- Externe Laag



physical data + logical data independence!



Catalog

- Hart van DBMS
- Bevat metadata
  - Data definities
  - Definities van views, logische en interne datamodel
- Synchronisatie tussen 3 datamodellen voor consistentie

Database Users

- Informatie-architect
  - Design conceptueel model
  - Dicht bij bedrijf en klant interactie

- Database Designer
  - Vertaling conceptueel naar logisch en intern model

- Database Administrator DBA
  - Implementatie en beheer van database

- Application Developer
  - Ontwikkelen database applicaties

- Bedrijfsgebruiker
  - Applicaties gebruiken voor specifieke database operaties

Database Languages

- Data Definition Language DDL
  - Gebruikt door DBA om externe, logische en intern datamodel te beschrijven

- Data Manipulation Language DML
  - Ontvangen, bewerken en invoegen van data

- Structured Query Language SQL
  - Zowel DDL als DML voor relationele databases

Advantages of Database Systems and Database Management

Data Onafhankelijkheid
- Wijzigingen in data hebben weinig impact op de applicaties

  Fysieke data onafhankelijkheid
  - Applicaties, views of logisch datamodel moet niet worden aangepast bij veranderingen in data-opslag of intern model
  - DBMS biedt interface tussen logisch en intern model

  Logische data onafhankelijkheid
  - Software applicaties minimaal beinvloed door wijzigingen in conceptueel of logisch model
  - Views blijven gelijk
  - DBMS biedt interface tussen conceptueepl/logisch model en extern model

Database Modelling
- Belangrijk om alle modeleigenschappen goed te documenteren

Ongestructureerde Data
- Tekstdocument
- Video, foto

Semi-Gestructureerd
- Geen bepaalde structuur, maar structuur is onregelmatig of vluchtig
- CV, sociale media

Gestructureerde Data
- Data beschreven in logisch model
- Data die je in een tabel/records kan opslaan
- Integriteitsregels mogelijk
  - Beperkingen op datamodellen

Data Redundantie
- Back-up systemen
- Verantwoordelijkheid DBMS
  - Synchroniseren van data consistency
  - Biedt correctheid van data (niet bij file-approach)

Integriteitsregels
- Deel van conceptueel/logisch model
- Opgeslagen in Catalog

- Syntactisch
  - Hoe wordt data voorgesteld en opgelsagen?
  - Veld is een integer, vorm waarin data wordt opgeslagen

- Semantisch
  - Wat betekent de data, is het correct?
  - Regels waaraan data voldoet

Concurrency Control
- DBMS zorgt voor parallelle uitvoeringsmogelijkheden
- Database transacties
  - Atomische eenheden van schrijf/leesoperaties
- DBMS vermijdt inconsistentie

- ACID
  - Atomicity

    - Consistency
    - Isolation
    - Durability

Backup and Recovery

- Omgaan met verlies van data
    - Netwerkfouten, bugs in systeem, hardwareproblemen

- Volledige of deels backup
- Data herstellen naar vorige state

Data Security

- Verschillende veiligheidsniveaus
- Logins, user accounts, …
- Opgeslagen in catalog

Key Performance Indicators KPIs DBMS

- Response Time
- Throughput Rate
- Space Utilization

Nagekeken en vervolledigd op 23/05/2024

Les 2 - Architecture

zondag 18 februari 2024 21:44

Liam Luys – r0934449 - 2e BA Informatica
Gegevensbanken

KU LEUVEN

Architectuur DBMS

DBMS

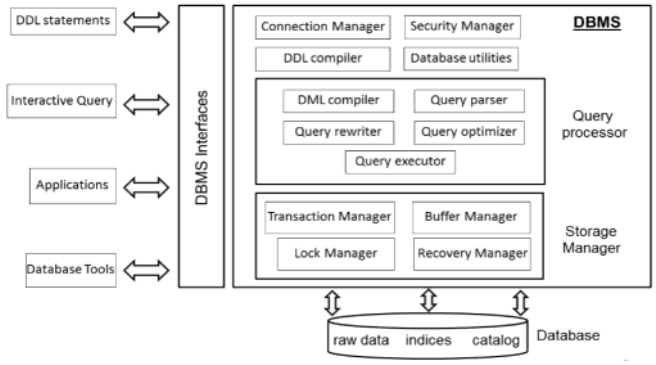
- DataBase Management System
- Moet verschillende data-gerelateerde activiteiten ondersteunen
  - Querying, Storage, …
- Interfaces bieden voor de omgeving

DBMS Architectuur

- Bestaande uit verschillende afzonderlijke componenten
- Afhankelijk van Functionaliteiten en Ontwerp

DBMS Onderdelen

- DDL Statements
  - Data-defenities in een catalogus
- Interactive Query
  - Uitgevoerd van interface: command line, graphical user, form, …
- Applications
  - Interactie via DML-instructies
- Database Tools
  - Voor onderhoud en beheer van DBMS door database manager



DBMS Onderdelen

- Connection Manager
  - Voorzien mogelijkheden voor connectie met database
  - Lokaal vs netwerk, authenticatie, proces vs threads
- Security Manager
  - Beheren van toegangsrechten (read/write/execute)
  - Catalog bevat toegangen
- DDL Compiler
  - Compileren van data-definities in DDL (Data Definition Language)
  - Ideaal zijn 3 DDL's: Intern, Logisch, Extern datamodel
  - Compileren Stappen
    - DDL Definities, syntactisch

        - Vertaling naar intern formaat, error handler
        - DDL definities in catalog opslaan

- Query Processor
    - Belangrijk onderdeel
    - Zorgt voor operaties/queries op database (Insert, update, remove, …)
    - Bestaat uit verschillende onderdelen

        - DML Compiler
            - Compileren van DML (Data Manipulation Language) instructies naar programmeertaal (host language)
            - Procedurele DML / Record at a time DML
                - Geven exact aan hoe navigeren en aan te passen in de database
                - Geen query-processor
                - Gebruikt in hiërarchische DBMS

            - Declaratieve DML / Set at a time DML
                - Geven aan wat moet gebeuren of welke aanpassingen moeten worden gedaan
                - Implementatie door query-processor
                  Voorbeeld: SQL (Sequential Query Language)

              Impedance Mismatch Problem
                - Mapping tussen object-georiënteerd (vb Java) en relationeel concept (vb SQL)
                - Hoe verschillende datastructuren op elkaar mappen?
                    - Gebruik maken van dezelfde datastructuren in OO en relationele concepten
                    - Object Relational Matters, Middleware die omzetting doet

            - Stappen DML Compiler
                - DML instructies uit host language halen
                - Samen met Query Parser, Rewriter, Optimizer en Executor de instructies uitvoeren
                - Errors genereren indien nodig

        - Query Parser
            - Query omzetten naar interne representatie
            - Afstemming met catalogus

        - Query Rewriter
            - Optimaliseren van query, onafhankelijk van huidige database-staat
            - Vergemakkelijken van query's, hervormen, uitschrijven, …

        - Query Optimizer
            - Query optimaliseren adhv huidige database-staat
            - Query-execution plans vergelijken qua kosten (responstijd schatting) via info uit catalog
            - Maken van goede schattingen en zoeken naar optimale uitvoeringspad (search problem)

        - Query Executer
            - Zorgt voor werkelijke uitvoering
            - Roept storage manager op

- Storage Manager
    - Bewaakt fysieke bestandstoegangen en correctheid/efficiëntie van de opslag ervan
    - Bestaande uit
        - Transaction Manager
            - Uitvoering van database transacties (sequentie read/write operaties in atomische eenheid)
            - Opstellen van schema, interleavings, ACID
            - Commit/Rollback bij succesvolle/gefaalde uitvoering -> Vermijden van inconsistentie

        - Buffer Manager
            - Buffergeheugen beheren
            - Data Locality of 20/80 wet: 80% van transacties gaat over 20% van de data

        - Replacement Policy

    - Lock Manager
        - Zorgt voor concurrency/gelijktijdigheid
        - Locks opzetten/afzetten
        - Read/Write locks

    - Recovery Manager
        - Correcte uitvoering nagaan via log-bestand
        - Herstellen bij crash/uitval

- DBMS Utilities
    - Loading: Laden van database met verschillende bronnen
    - Reorganization: Automatisch reorganisatie voor betere uitvoering
    - Performance: KPI (Key Performance Indicators) om prestatie te evalueren
    - User management: Maken van gebruikersgroepen en toegangsrechten
    - Backup en recovery

- DBMS Interface
    - Communiceren met anderen (gebruikers/programma's)
    - Veel varianten interface
        - Web, stand-alone, command line, graphical user, admin, network, …

Soorten DMBS

Categoriseren adhv
- Data Model
- Aantal gelijktijdige toegangen
- Architectuur
- Gebruik

DBMS kan in meerdere/andere categorieën vallen

Data Model

Datamodellen
- Hiërarchische DBMS
    - Boomstructuur datamodel
    - Procedurale DML, record-gericht
    - Geen query-processor
      Vb: IMS (IMS)

- Netwerk DBMS
    - Netwerk datamodel, meer flexibel (CODASYL DBMS)
    - Procedurale DML, record-gericht
    - Geen query-processorµ
      Vb: CA-IDMS

- Relationele DBMS
    - Relationeel datamodel
    - Meest gebruikt!
    - SQL (declaratief, set-gericht)
    - Query-processor
    - Scheiding tussen logische en interne datamodel
      Vb: MySQL

- Objectgerichte DBMS
    - OO datamodel
    - Geen impedance mismatch

    - Niche markten populair, niet in industrie
      Vb: db4o

- Object-Relationeel DBMS (ORDBMS)
    - Relationeel model uitgebreid met objectgerichte functies
    - SQL, relationeel model

- XML DBMS
    - XML als dataopslag
    - XML-structuur mappen naar fysieke opslag

- NoSQL DBMS
    - Big data, ongestructureerd
    - Key-value, tuple, graph, column-oriented, …
    - Gemakkelijker fraude opsporen
      Vb: Neo4j (grafen)

Aantal gelijktijdige toegangen

Aantal gelijktijdige toegangen

- Single User
- Multiple User (threads, concurrency)

Architectuur

Architecturen

- Gecentraliseerd
    - Data op één centrale server

- Client Server
    - Actieve clients vragen service van passieve servers

- n-tier
    - Client met GUI, applicatie server, web servers
    - Via middleware communiceren

- Cloud
    - Hosted door third-party cloud provider

- Federated
    - Uniforme interface voor meerdere databronnen
    - Verbergen van onderliggende opslagdetails

- In-Memory
    - Alle data in geheugen plaatsen
    - Voor real-time doeleinden

Gebruik

Gebruiksvormen

- Online Transaction Processing (OLTP)
    - Operationeel of transacties beheren
    - Server moet veel transacties aankunnen tegelijk
    - Goede support voor grote hoeveelheid korte, simpele queries
      Vb: POS-systeem supermarkt

- Online Analytical Processing (OLAP)
    - Operationele data gebruiken voor strategische beslissingen
    - Beperkte hoeveelheid gebruikers
    - Kleinere hoeveelheid, complexe queries

- Big Data & Analytics
    - NoSQL Databases

    - Flexibele structuren
    - Ongestructureerde data: e-mails, teksten, sociale media

- Multimedia
    - Opslag voor multimedia data
    - Inhoud-gerelateerde query-mogelijkheden

- Spatial
    - Spatial dataopslag (2D en 3D)
    - Geografische Informatiesystemen (GIS)

- Sensoring
    - Sensoren gebruiken als biometrische data Wearables, telematische data, …

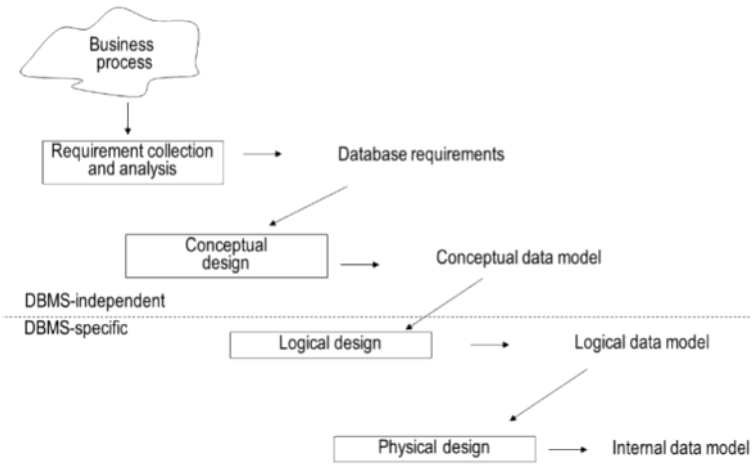
- Mobile
    - DBMS op smartphones, tablets
    - Elk moment online, synchronisatie, batterijduur en opslag beperkt, …

- Open Source
    - Code voor open source DBMS, publiek beschikbaar
    - MySQL, Twig

Conceptual Data Modeling using (E)ER model and UML Class Diagram

Phases of Database Design

Fases in design
- Business Process
- Database Requirements
- Conceptueel datamodel
- Logisch Datamodel
- Intern Datamodel



Entity Relationship (ER) Model

Entiteit Types
- Stelt bedrijfsconcept voor met éénduidige betekenis voor gebruikers
  Vb. studenten

Entiteit
- Eén instantie van een entity type

Vb. een student

Attribuut Types
- Stelt eigenschap van entity type voor
  Vb. naam van student

Attribuut
- Eén instantie van attribuut type
  Vb. Liam

Domein
- Mogelijke waardes voor attribuut verbonden aan bepaalde entiteit
- Kan NULL-waarde bevatten
- Niet zichtbaar in ER model
  Vb. meerderjarig: Ja / Neen

Sleutelattribuut
- Specifiek en uniek attribuut, voor elke entiteit verschillend
- Beschrijft bepaalde entiteit
- Kan combinatie zijn van andere attribuuttypes
  Vb. studentennummer

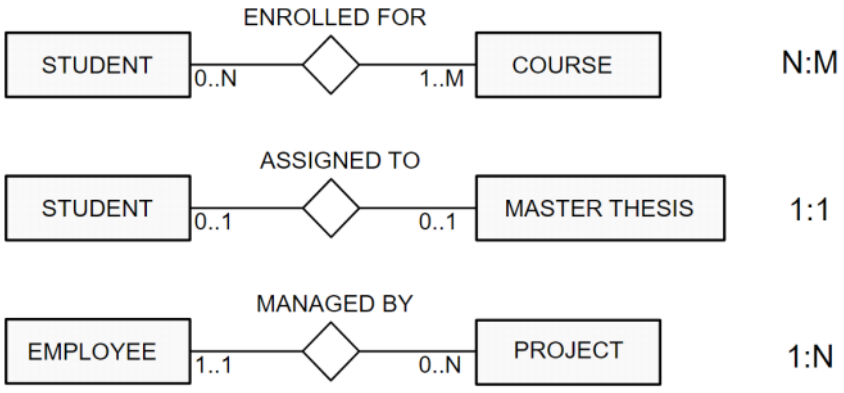
Attributen Soorten
- Simpel vs. Samengesteld
  - Bestaat attribuut uit meerdere onderdelen?
    Vb. Adres

- Enkel vs. Meerdere waardes
  - Kan een attribuut meerdere waardes bevatten?
    Vb. emailadres

- Afgeleid attribuuttype
  - Waarde kan je afleiden uit ander attribuuttype
    Vb. leeftijd (geboortedatum)

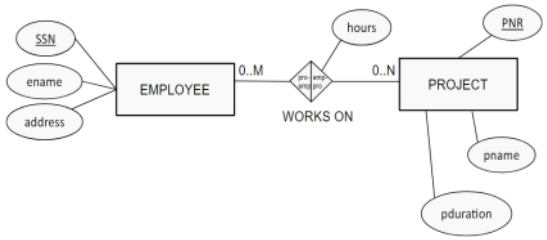
Relatie
- Stelt associatie voor tussen meerdere entiteiten
- Graad: Aantal entiteitstypes die een relatie ondersteunen
- Rol: In welke richting interpreteren
- Cardinaliteit: Minimale en maximale aantal relaties waaraan entiteit deelneemt

<u>Minimumcardinaliteit van 1 --> Bestaansafhankelijkheid in model!</u>



Relatie Attribuut Type

- Attribuuttype van een relatie
- Informatie/Waardes van de relatie tussen entiteiten

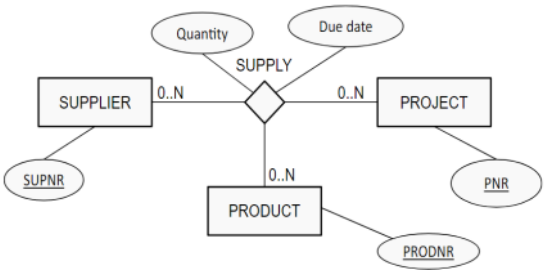


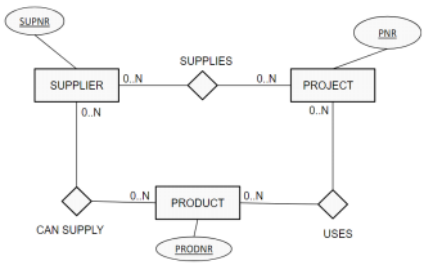
Zwakke Entiteit Type

- Heeft geen sleutel-attribuut op zichzelf
- Moet sleutel-attribuut van moeder-entiteit lenen om sleutel te maken
- Altijd bestaansafhankelijk!
  Vb. kamers in hotel

Ternaire Relatietypes

- Opletten voor verlies van semantiek!





Note: loss of semantics!

Beperkingen ER Model

- Stelt tijdelijk snapshot voor, geen tijdsafhankelijke voorwaarden mogelijk
- Geen garantie op consistentie overheen relatietypes
- Domeinen worden niet meegegeven in model
- Functies zijn niet verbonden in ER model

Nagekeken en vervolledigd op 23/05/2024

Conceptual Data Modelling using the (E)ER model and UML Class Diagrams

Enhanced Entity Relationship (EER) Model

Specialisatie
- Top-Down
- Definiëren van subklassen van een entiteitstype
- "IS A" relatie
  Vb. Artist superklasse met zanger, acteur subklasse

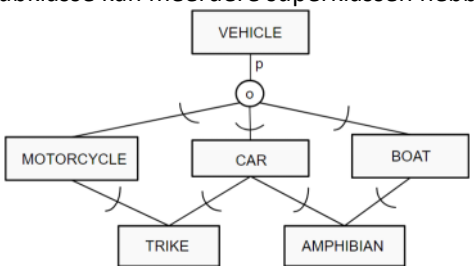
Generalisatie / Abstractie
- Bottom-Up
- Van subklassen naar superklasse

Voorwaarden Generalisatie/Specialisatie
- Disjoint/Overlap
  - Entiteit maximaal in één subklasse --> DISJOINT
  - Entiteit kan in meerdere subklassen --> OVERLAP

- Completeness
  - Volledige specialisatie: elke entiteit is deel van een subklasse
  - Partiele specialisatie: entiteit is niet noodzakelijk deel van subklasse

Specialisatie Hiërarchie
- Elke subklasse één superklasse
- Overerving superklasse specificaties

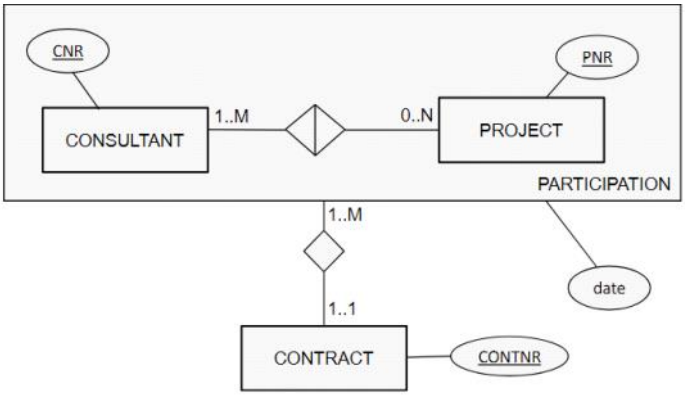
Specialisatie Rooster
- Subklasse kan meerdere superklassen hebben



Categoriseren
- Onderverdelen in subklasse met meerdere mogelijke superklassen
- Elke superklassen een verschillend entiteitstype
- Categorie stelt verzameling entiteiten voor in de unie van superklassen

- Totale Categorisatie: Alle entiteiten van superklassen behoren tot subklasse
- Partiele Categorisatie: Niet alle entiteiten van superklassen behoren tot subklasse

Aggregatie
- Samenvoegen van entiteiten in een onderlinge relatie tot nieuw entiteitstype
- Nuttig als nieuw entitetstype eigen attribuuttype en/of relatietype heeft



Design EER Model

1. Identificeer entiteitstypes
2. Identificeer relaties en bijhorende graden
3. Stel cardinaliteiten en voorwaarden op
4. Identificeer attribuuttypes en eigenschappen
5. Link elk attribuuttype aan entiteitstype of relatietype
6. Sleutelattribuuttype van elk entiteitstype duidelijk maken
7. Identificeer zwakke entiteitstypes en partiele sleutels
8. Voeg abstracties toe (generalisatie/specialisatie/categoriseren/aggregatie)
9. Voeg specificaties toe aan abstracties (disjoint, overlap, totaal, partieel)

UML Class Diagrams

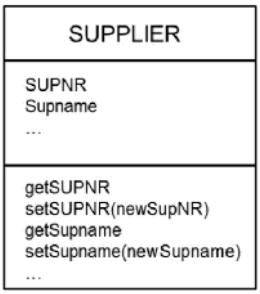
Oorsprong UML

- Unified Modeling Language
- ISO Standaard vanaf 2005
- Visualisatie, constructie en documentatie van softwaresystemen

UML: bestaande uit klassen *(Entiteitstypes)*
Klasse: set van objecten *(Entiteiten)*
Object: Gekarakteriseerd door variabelen *(attribuuttypes)* en methodes

Information Hiding: Encapsulatie
Inheritance: methodes/variabelen van superklasse voor meerdere subklassen gebruiken
Method Overloading: Methodes in dezelfde klasse met gelijke naam maar andere argumenten

Klassen



Variabelen

- Geen support voor unieke waardes (sleutelattribuuttypes) in UML
- Bepaalde types beschikbaar: string, int, boolean

Access Modifiers

- Instellen wie toegang heeft tot een variabele of methode
- 3 Types
    - Privaat (-): enkel bij eigen klasse --> Voorkeur (encapsulatie)
    - Publiek (+): alle klassen en subklassen
    - Beschermd (#): klasse en subklasse

Associaties

- Relatietype ER
- Meerdere associaties binnen dezelfde klasse mogelijk
- Link = Bepaald voorkomen van associatie
- Multipliciteit bepaald (gelijk aan cardinaliteit)

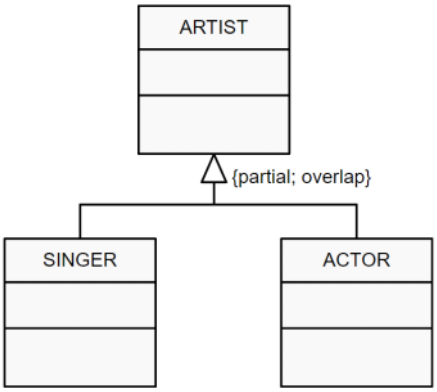
UML class diagram multiplicity	ER model cardinality
*	0..N
0..1	0..1
1..*	1..N
1	1..1

- Associatieklasse
    - Variabelen en methoden van associatie zelf
    - Associatie zien als nieuwe klasse

- Richting Associatie
    - Unidirectional: Bepaalde richting in associatie (pijl)
    - Bidirectional: Geen richting in associatie (rechte lijn)

- Qualified Associatie
    - Type associatie die qualifier gebruikt
    - Gebruikt één/meerdere variabelen als index voor navigatie van qualified klasse naar doelklasse
    - Gebruikt om zwakke entiteiten voor te stellen

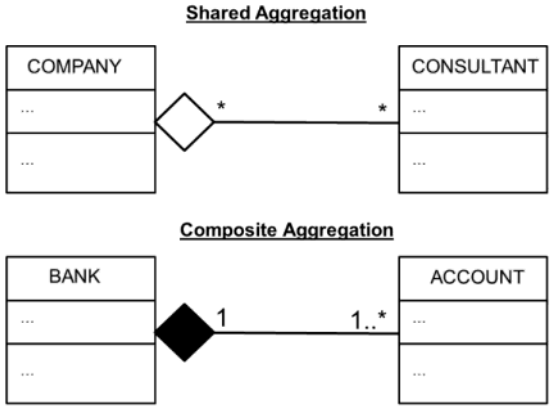
Specialisatie / Generalisatie



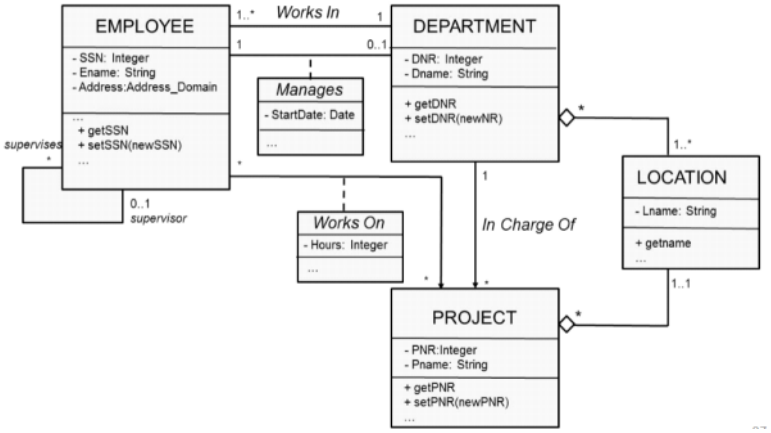
Aggregatie

- Samengestelde relatie waarbij samengestelde klasse een deelklasse bevat
- Twee types
    - Gedeelde Aggregatie (Aggregatie)
        - Object kan behoren tot meerdere samengestelde objecten
        - Maximum multipliciteit is ondergedetermineerd
        - Deelobject kan ook voorkomen zonder samengesteld object
        - Zwakke koppeling

    - Samengestelde Aggregatie (Samenstelling)
        - Deelobject behoort bij één samengesteld object
        - Maximum multipliciteit samengesteld object is 1
        - Mimimum multipliciteit is 0 of 1
        - Sterke koppeling



UML Class Diagram Example



Advanced Modeling Concepts

- Wijziging Eigenschap
  - Specifieert type operaties toegestaan op variabele-waardes of links
  - 3 keuzes
    - Default: elk type toegestaan
    - addOnly: geen verwijderen, wel toevoegen
    - Frozen: geen wijzigingen

- Object Constraint Language OCL
  - Gebruikt om specifieke voorwaarden uit te drukken in declaratieve manier
  - Invarianten (pre/post, klasse-invariant)

- Dependency Relationship
  - Afhankelijkheid tussen relaties
  - Wijzigingen in UML model heeft effect op andere concepten die het gebruiken

UML Class Diagram vs. EER

UML class diagram	EER model
Class	Entity type
Object	Entity
Variable	Attribute type
Variable value	Attribute
Method	-
Association	Relationship type
Link	Relationship

UML class diagram		EER model	
Qualified Association		Weak entity type	
Specialisation/Generalisation		Specialisation/Generalisation	
Aggregation		Aggregation (Composite/Shared)	
OCL		-	
Multiplicity	*	Cardinality	0..N
	0..1		0..1
	1..*		1..N
	1		1..1

Nagekeken en vervolledigd op 23/05/2024

Organizational Aspects of Data Management

Data Management

Metadata

- Data over data
- Net als gewone data ook belangrijk om te modelleren
  - Concepten van data ook toepassen op metadata

- Bij DBMS
  - Metadata opgeslagen in catalog ('hart van database systeem')

- Biedt antwoorden op vele vragen
  - Welke data bevat de database?
  - Wie heeft toegang?
  - Welke transacties werken met welke data?
  - ….

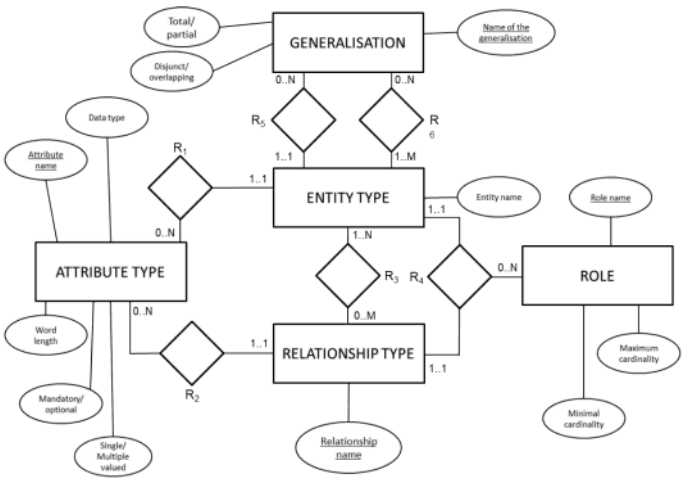
Catalog

- Geïntegreerd of los van DBMS
- Voorkeur voor geïntegreerde versie
- Belangrijke info voor gebruikers
  - Gebruiken van metadata voor queries op DB
  - Import/export mogelijkheden
  - Onderhoud en hergebruik van metadata
  - Info over data en gebruikers/toegangen/…

Metamodel

- Datamodel voor metadata
- Geeft weer welke metadata wordt opgeslagen
- Kan opgebouwd worden door database design
  - Conceptueel Model
    - EER
    - UML

- Voorbeeld EER conceptueel metamodel



Data Quality

- "Juistheid voor een bepaald gebruik"
  - Handigheid/Kwaliteit hangt af van gebruikscontext

- Bepaald waarde van data voor business
  - GIGO (Garbage in, garbage out)
  - Slechte data leidt tot slechte beslissingen

- Heeft veel invloed op organisatie
  - Operationeel Level
    - Gebruikerstevredenheid
    - Jobtevredenheid
  - Strategisch Level
    - Beslissingen beinvloeden

- Wordt uitgedrukt door framework
  - Data Quality Framework categoriseert verschillende dimensies voor kwaliteit *vb Wang et al.*

Category	DQ dimensions	Definitions
Intrinsic	Accuracy	The extent to which data is certified, error-free, correct, flawless and reliable
	Objectivity	The extent to which data is unbiased, unprejudiced, based on facts and impartial
	Reputation	The extent to which data is highly regarded in terms of its sources or content

Category	DQ dimensions	Definitions
Contextual	Completeness	The extent to which data is not missing and covers the needs of the tasks and is of sufficient breadth and depth of the task at hand
	Appropriate-amount	The extent to which the volume of data is appropriate for the task at hand
	Value-added	The extent to which data is beneficial and provides advantages from its use
	Relevance	The extent to which data is applicable and helpful for the task at hand
	Timeliness	The extent to which data is sufficiently up-to-date for the task at hand

Category	DQ dimensions	Definitions
Representation	Interpretable	The extent to which data is in appropriate languages, symbols and the definitions are clear
	Easily-understandable	The extent to which data is easily comprehended
	Consistency	The extent to which data is continuously presented in the same format
	Concisely-represented (CR)	The extent to which data is compactly represented, well-presented, well-organized, and well-formatted
	Alignment	The extent to which data is reconcilable (compatible)

Category	DQ dimensions	Definitions
Access	Accessibility	The extent to which data is available, or easily and swiftly retrievable
	Security	The extent to which access to data is restricted

Category	DQ dimensions	Definitions
Access	Accessibility	The extent to which data is available, or easily and swiftly retrievable
	Security	The extent to which access to data is restricted appropriately to maintain its security
	Traceability	The extent to which data is traceable to the source

Belangrijke Data Quality Dimensies

- Accuracy
  - Correctheid van de waarde, hangt vaak samen met andere dimensies

- Completeness
  - Schema Completeness: graad van missende entiteits- en attribuuttypes in schema
  - Column Completeness: graad van missende waardes in kolom van tabel
  - Population Completeness: graad van leden van populatie die worden voorgesteld

- Consistency
  - Is alle data consistent met elkaar?
  - Verschillende perspectieven
    - Dubbele data op meerdere plaatsen
    - Twee gerelateerde elementen (vb postcode en plaats)
    - Zelfde element met andere notatie

Oorzaken Slechte Data Quality

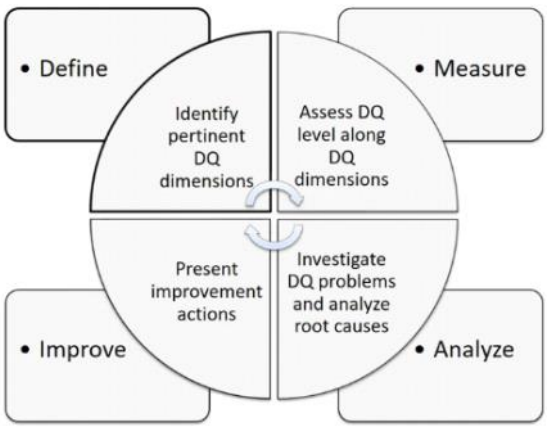
- Meerdere databronnen
- Subjectieve Beslissingen
- Beperkte computermogelijkheden
- Volume van data
- …

Data Governance / Gegevensbeheer

- Om data te beheren en datakwaliteit te garanderen
- Gegevens beheren als een bezit ipv een verplichting
- Vaak ad-hoc en reactieve moeite voor databeheer

Veel data quality problemen worden niet aangepakt, of pas enkel wanneer echt nodig

- Verschillende frameworks
  vb Wang



Roles in Data Management

Welke jobmogelijkheden in Data Management?

Rollen in Databeheer

- Information Architect
  - Conceptueel model maken
  - Overgang business en IT-omgeving

- Database Designer
  - Conceptueel naar logisch datamodel

- Data Owner
  - Beheert toegangen, data, gebruik, …

- Data Steward
  - Waakt over data kwaliteit

- Database Administrator
  - Verantwoordelijk voor implementatie

- Data Scientist
  - Analyse van data

Relational Databases

Relational Model

Basisconcepten

- Databasemodel voorgesteld door relaties tussen elementen
- Relatie als records/tuples, geordende lijst van attribuut-waardes
- Geen grafische representatie
- Bouwen van logische en interne datamodellen

EER model	Relational Model
Entity type	Relation
Entity	Tuple
Attribute type	Column name
Attribute	Cell

Domein

- Mogelijke waardes dat een attribuut kan aannemen
- Enkelvoudige attributen, **geen geneste/samengestelde waardes!**
- NULL-waarde: missende waarde / onbelangrijk / Niet acceptabel

Relatie

- Stelt een set voor
- Geen ordening / duplicaten
- Cartesisch product tussen domeinen

Types van Keys

- Superkey

    - (Meerdere) attributen die een record uniek bepalen
    - Kan redundante/overbodige attribuuttypes bevatten
      Studentnummer + naam

- Key
    - Kleinst aantal attributen om record uniek te bepalen
    - Minimale superkey (zonder redundante attribuuttypes)
      studentnummer

- Kandidaat Keys
    - Indien relatie meer dan 1 key heeft

- Primaire Key
    - Om record in relatie te identificeren
    - Mag nooit NULL-waarde kunnen hebben!

- Alternatieve Keys
    - Kandidaat keys die niet als (primary) key worden gebruikt

- Vreemde Key
    - Attribuuttype set FK vreemde sleutel als
        - FK heeft zelfde domein als primary key attribuuttype PK van andere relatie
        - Waarde FK in record van huidige state komt ook voor als waarde van PK in huidige state of is NULL

Domain constraint	The value of each attribute type A must be an atomic and single value from the domain dom(A).
Key constraint	Every relation has a key that allows to uniquely identify its tuples.
Entity integrity constraint	The attribute types that make up the primary key should always satisfy a NOT NULL constraint.
Referential integrity constraint	A foreign key FK has the same domain as the primary key PK attribute type(s) it refers to and either occurs as a value of PK or NULL.

Normalization

Normalisatie
- Inconsistentie tegengaan
- Alles normaliseren geeft goed relationeel model
- Voordelen
    - Logisch model: gebruiker begrijpt data en formuleert goede queries
    - Implementatie: Storage wordt efficiënt gebruikt, geen inconsistentie

Richtlijnen Normalisatie
- Geef duidelijke benamingen aan methodes
- Attribuuttypes van meerdere entiteitstypes niet combineren in enkele relatie
- Vermijden van veel NULL-waardes in relaties

Functionele Afhankelijkheid $X \rightarrow Y$

- Indien 2 attributen X en Y functioneel afhankelijk, dan bepaald X een unieke waarde voor Y
- Niet perse in twee richtingen!
- Soorten
    - Volledig Functioneel Afhankelijk
        - Als verwijderen van een attribuuttype van X ervoor zorgt dat de afhankelijkheid niet meer geldt
    - Partieel Functioneel Afhankelijk
        - Als een attribuuttype verwijderd kan worden van X en de afhankelijkheid blijft bestaan

Triviale Functionele Afhankelijkheid $X \rightarrow Y$

- Als Y een deelverzameling is van X
  Vb. $SSN, NAME \rightarrow SSN$

Multi-valued Afhankelijkheid $X \rightarrow\rightarrow Y$

- Als elke X waarde exact een set van Y waardes bepaald, onafhankelijk van andere attribuuttypes

Transitieve Afhankelijkheid $X \rightarrow Y$

- Als een set van attribuuttypes Z bestaat dat geen kandidaat key of subset van een key is, en $X \rightarrow Z$ en $Z \rightarrow Y$ geldt

Prime Attribuuttype

- Attribuuttype dat deel uitmaakt van kandidaat key
  Vb. $R1(\boldsymbol{SSN}, \boldsymbol{PNUMBER}, PNAME, HOURS) \rightarrow SSN\ en\ PNUMBER$ prime attribuuttypes

Normaalvormen

- Eerste Normaalvorm
    - Elk attribuuttype van relatie atomisch en enkele waarde
    - Geen samenstellingen of multi-valued attribuuttypes
- Tweede Normaalvorm
    - Eerste Normaalvorm
    - Elke niet-prime attribuuttype A in de relatie is volledig functioneel afhankelijk van elke key in de relatie
- Derde Normaalvorm
    - Tweede Normaalvorm
    - Geen enkele niet-prime attribuuttype is transitief afhankelijk van de primaire key
- Boyce-Codd Normaalvorm
    - Elke niet-triviale functionele afhankelijkheid $X \rightarrow Y$ als X een superkey is, is X een kandidaat key of een superset ervan
    - Strikter dan Derde Normaalvorm (elke hier is in 3NF, niet andersom)
- Vierde Normaalvorm
    - In Boyce-Codd Normaalvorm
    - Elke niet-triviale multi-valued afhankelijkheden $X \rightarrow\rightarrow Y$, als X een superkey is, is X een kandidaat key of een superset ervan

Nagekeken en vervolledigd op 23/05/2024

Relational Databases

Mapping Conceptueel ER Model naar Relationeel Model

Mapping Entiteitstypes


```
EMPLOYEE(SSN, address, first name, last name)
PROJECT(PNR, pname, pduration)
```

Mapping Relatietypes

- Binaire 1:1 relatietype
  - Twee relaties maken: één voor elk entiteitstype
  - Vreemde sleutel van de ene en primaire sleutel van de andere toevoegen (connectie maken)
  - Bestaansafhankelijkheid --> vreemde sleutel in bestaansafhankelijke relatie en NOT NULL

- Binaire 1:N relatietype
  - Vreemde sleutel (verwijzend naar andere entiteitstype) toevoegen bij entiteitstype met N als multipliciteit
  - Afhankelijk van cardinaliteit: NOT NULL of NULL ALLOWED

- Binaire M:N relatietype
  - Nieuwe relatie R introduceren die twee entiteitstypes mapped
  - Primaire key R is combinatie van vreemde sleutels die verwijzen naar entiteitstypes

- Unaire relatietype
  - 1:1 -> Vreemde key verwijzend naar primaire key van zelfde relatie toevoegen
  - M:N -> Nieuwe relatie R aanmaken

- n-rij relatietype
  - Relaties aanmaken voor elk entiteitstype
  - Extra relatie R aanmaken voor combinatie van entiteitstypes
  - Primary key R is combinatie van alle vreemde NOT-NULL keys

Mapping Multivalued attribuuttypes

- Voor elk multivalued attribuuttype een nieuwe relatie aanmaken
- Samengestelde multivalued attribuuttypes ontleden naar componenten

Mapping Zwakke Entiteitstypes

- Mappen in relatie R met alle corresponderende attirbuuttypes
- Vreemde key toevoegen die naar primaire key van relatie van ouder-entiteit wijst
- Bestaansafhankelijkheid --> vreemde key is NOT NULL

Alles Samenvoegen

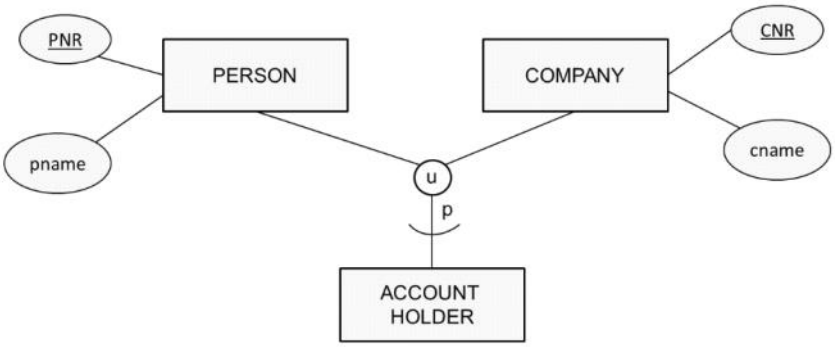
ER Model	Relational model
Entity type	Relation
Weak entity type	Foreign key
1:1 or 1:N relationship type	Foreign key
M:N relationship type	New relation with two foreign keys
N-ary relationship type	New relation with N foreign keys

Mapping Conceptueel EER Model naar Relationeel Model

Mapping Specialisatie

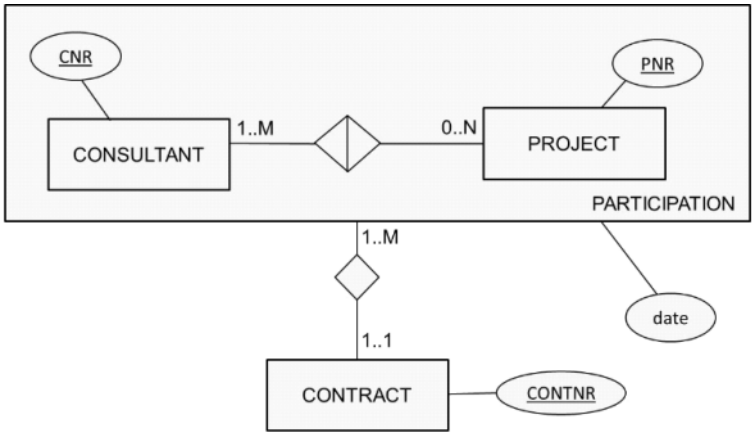
- 3 Opties
  - Relatie maken voor superklasse en elke subklasse linken met vreemde sleutel
  - Relatie voor elke subklasse en geen voor superklasse
  - Eén relatie met alle attribuuttypes van super- en subklassen en voeg speciaal attribuuttype toe

Mapping Categorisatie



PERSON(PNR, …, *CustNo*)
COMPANY(CNR, …, *CustNo*)
ACCOUNT-HOLDER(CustNo, …)

Mapping Aggregatie



CONSULTANT(CNR, …)
PROJECT(PNR, …)
PARTICIPATION(*CNR*, *PNR*, *CONTNR*, date)
CONTRACT(CONTNR, …)

Nagekeken en vervolledigd op 23/05/2024

Structured Query Language

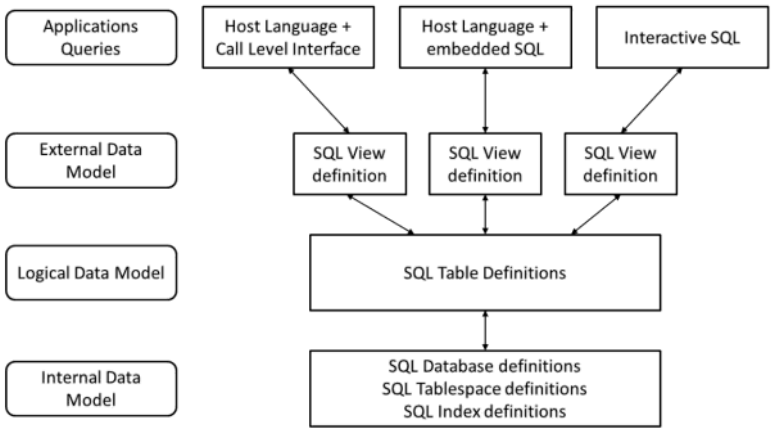
Relation Database Management and SQL

Oorsprong SQL
- ISO Standaard 1987
- Dialecten en eigen ontwikkelingen

Eigenschappen SQL
- Set-Georienteerd
  - Werken met datasets
  - Elk resultaat van SQL is een set
- Declaratief
  - SQL gebruiken om aan te geven welke data we willen (aanpassen)
  - Niet specifieren hoe de data gezocht moet worden
  - Geen proceduraal programeren!
- Dialecten
  - Veel verschillende implementaties van SQL
    - PostgreSQL, MySQL, …
- Free Form Language

Gebruik SQL
- Interactief / Dynamisch
  - SQL instructie in terminal ingeven
  - Instructies worden direct uitgevoerd
- Embedeed / Statisch
  - SQL instructies in code van een programma
  - Instructies uitgevoerd tijdens uitvoering programma

Three Level Database Architecture



SQL Functies
- Definiëren van data

    - Maken van tabellen
    - Data Definition Language DDL

- Data manipulatie
    - Bekijken en beheren van bestaande data
    - Toevoegen, updaten, verwijderen, …
    - Data Manipulation Language DML

SQL Data Definition Language

Data Definition Language DDL
- SQL maakt nieuwe structuren aan
- Opzetten van tabellen met data, views, voorwaarden
- Implementeren van relationele model

Kolomvoorwaarden
    - Primaire Key
    - Vreemde Key
    - Uniek
    - NOT NULL
    - DEFAULT
    - Check

Referential Integrity Constraints
- Vreemde sleutel zelfde domein als primaire sleutel waar het naar verwijst
- Wat met vreemde sleutel als primaire sleutel wordt bijgewerkt?
    - ON UPDATE/DELETE CASCADE
    - ON UPDATE/DELETE RESTRICT
    - ON UPDATE/DELETE SET NULL
    - ON UPDATE/DELETE SET DEFAULT

DROP-Commando
- Weglaten of verwijderen van database objecten
- Combinatie mogelijk met CASCADE en RESTRICT

ALTER-Commando
- Aanpassen kolomdefinitie
- Bv instellen default waarde

SQL Data Manipulation Language

Data Manipulation Language DML
- Data beheren en bekijken
- Verschillende operatoren mogelijk
    - SELECT
    - INSERT
    - UPDATE
    - DELETE

SELECT Operation

SELECT Operatie
- Verzamelen van data uit relationele database
- Past geen gegevens aan!
- Hoofddoel: weergave van gegevens

- kiezen welke kolommen worden weergegeven, …

- Extra clauses mogelijk
  - FROM: tabellen die je nodig hebt
  - WHERE*: Condities op dataset
  - GROUP BY*: Kolommen om te groeperen
  - HAVING*: Conditie op gegroepeerde dataset
  - ORDERED BY*: Ordening van gegevens

    ** = optioneel*

FROM Clause

FROM Clause
- Verplicht in elke SQL instructie

- Eén Tabel
  - Logisch, tabel zelf gebruiken

- Meerdere Tabellen
  - Wordt vaak gebruikt (door genormaliseerde data in meerdere tabellen)
  - Tabellen vermelden en terug aan elkaar linken via sleutels

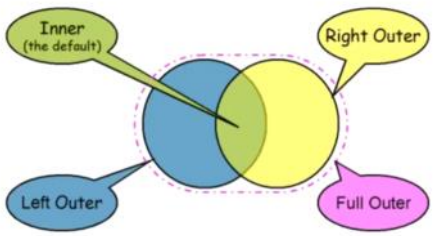
Tabellen linken in SQL
- SQL berekent kruisproduct van tabellen (alle mogelijkheden)
- Toepassen van INNER JOIN (voorkeur) of WHERE

JOIN Operaties
- INNER JOIN
  - Alle records met linken tussen tabellen worden weergegeven

- LEFT OUTER JOIN
  - Alle records van linkertabel met links naar rechtertabel (ook zonder link wordt weergegeven met NULL)

- RIGHT OUTER JOIN
  - Alle records van rechtertabel met links naar linkertabel (ook zonder link wordt weergegeven met NULL)

- FULL OUTER JOIN
  - Alle records uit beide tabellen



Aliasses
- Naam toewijzen naar kolom/tabel
- Wijzigt niets aan echte database!

e.g.: Alias for table – Show the name and telephone number of every person

```
SELECT p.name, p.telnr
FROM   Person AS p;
```

e.g.: Alias for table and JOIN – Overly complex way to show exactly the same

e.g.: Alias for table – Show the name and telephone number of every person

```
SELECT p.name, p.telnr
FROM   Person AS p;
```

e.g.: Alias for table and JOIN – Overly complex way to show exactly the same

```
SELECT a.name, a.telnr
FROM   Person AS a INNER JOIN Person AS b
       ON a.id = b.id;
```

e.g.: Alias for column – Change the **displayed** column name to Number

```
SELECT name, telnr AS Number
FROM   Person;
```

42

WHERE Clause

WHERE Clause
- Condities toevoegen op dataselectie
- Meestal gebruikt met FROM operatie

Boolean Operaties
- Combineren van condities
- 'AND' en 'OR'

In-Operator
- Specifiëren van meerdere waarden in een WHERE Clause
  WHERE Country IN ('Belgium', 'Mexico') === WHERE Country='Belgium' OR Country='Mexico'

Between-Operator
- Voorstellen van conditie met bepaald bereik
  WHERE ID Between 1 AND 100

LIKE-Operator
- Zoeken naar patroon in kolom
- Mogelijk gebruik van wildcards *(% of *)*
  WHERE Name LIKE '%Har%'

NULL-Waarde
- Stelt een onbekende waarde voor
- Kan je niet op filteren met WHERE
- Specifieke filters
  - WHERE … IS NULL
  - WHERE … IS NOT NULL

Miscellaneous Functions

Wildcards
- % of *
- Geeft alle kolommen terug

DISTINCT Clause
- Verwijderen van duplicaten in de query
- Verwijderd niets in de database! Enkel in de query

Aggregation Functions
- Functies gebaseerd op bepaalde waarden in de query
- Voorbeelden

    - Avg()
    - Count()
    - First()
    - Last()
    - Max()
    - ….

- Combinatie mogelijk in SELECT operatie

ORDER BY Clause

ORDER BY Clause

- Geeft aan in welke volgorde resultaten moeten worden getoond
- Indien niet gespecifieerd, wordt databasevolgorde weergegeven

- Meerdere ordeningen tegelijk mogelijk

GROUP BY Clause

GROUP BY Clause

- Groeperen van gegevens adhv bepaalde kolomwaarde
- Niet hetzelfde als SELECT DISTINCT!
- Resultaat zijn groeperingsnamen, maar achterliggend worden records in de groep bijgehouden
    - Je kan aggregation functions oproepen op de groepen

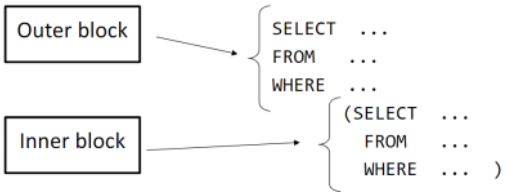
HAVING Clause

HAVING Clause

- Legt conditie op gegroepeerde data
- Niet hetzelfde als WHERE!
- Filteren op aggregation functions op de groepering van data

Structured Query Language

Nested Queries
- Inner Block
- Outer Block



Subquery vs join operatie
- Vaak zijn subquery/geneste queries efficiënter

Correlated Queries
- In inner block objecten uit outer block gebruiken
- Als conditie in WHERE clause van geneste query verwijst naar kolom van tabel gedeclareerd in outer query

SQL Operatoren

ALL
- Vergelijkt met alle elementen uit multiset
- Als geneste query geen waarde teruggeeft: TRUE return

ANY
- Vergelijkt en zoekt één element uit multiset die aan voorwaarde voldoet
- Als geneste query geen waarde teruggeeft: FALSE return

EXISTS
- Kijkt of een resultaat van een query leeg is of niet
- Boolean waarde

Set Operatoren

UNION
- Unie van 2 SELECT operaties
- SELECT operaties moeten zelfde structuur teruggeven

INTERSECT
- Doorsnede van 2 SELECT operaties
- SELECT operaties moeten zelfde structuur teruggeven

EXCEPT
- Verschil van 2 SELECT operaties
- SELECT operaties moeten zelfde structuur teruggeven

SQL Views

SQL Views
- Onderdeel externe datamodel
- Content gegenereerd op uitvoering/vraag van applicatie
- Onderdeel three-layer database architecure
    - Maakt logische data onafhankelijkheid mogelijk

View Materialization
- Fysieke tabel gegenereerd wanneer view eerste keer wordt opgevraagd

WITH CHECK
- Nakijken of record na de update nog tot de view behoort
- Condities vergelijken en fout geven indien negatief

SQL Privileges

SQL Privileges
- Recht om bepaalde SQL statements te gebruiken op database object(en)
- Rechten toekennen via GRANT/REVOKE

SQL for Metadata Management

Catalog
- Implementeren als relationele database

Examen
- Wat doet query?
- Voorbeeld slide 12
- Query oplossen door voorbeelden op te lossen op papier

Les 8 - Warehousing 1

maandag 15 april 2024 10:05

Liam Luys – r0934449 - 2e BA Informatica
Gegevensbanken

KU LEUVEN

Data Warehousing and Business Intelligence

Decision Making Levels

Niveaus Management / Beslissingen maken

- Operationeel Niveau
  - Dagelijkse beslissingen
  - Korte tijd
  - On-Line Transaction Processing OLTP

- Tactisch Niveau
  - Middle management beslissingen
  - Medium termijn focus (maanden tot jaar)

- Strategisch Niveau
  - Senior management
  - Jaren termijn focus (1-5 jaren)

Operationele Systemen

- Operationeel niveau
- Focus op simpele instructies
  - INSERT, UPDATE, DELETE, SELECT

Decision Support Systems

- Tactisch en strategisch niveau
- Focus op datakennis dmv complexe queries
- Analyses, data weergeven in meerdere dimensies

Data Warehouse Definition

Data Warehouse

- Subject Oriented
  - Data georganiseerd rond bepaald onderwerp: Klanten, Producten, verkoop, …

- Integrated
  - Data integreren vanuit verschillende bronnen in verschillende vormen

- Non-Volatile
  - Data is read-only, niet vaak bewerken of verwijderen
  - Data loading en data retrieval

- Time Variant
  - Data uit bepaalde tijd(speriode) / snapshots
  - Niet perse up-to-date (operationele data wel!)

Operationele Data

- Altijd de live up-to-date data

	Transactional system	Data Warehouse
Usage	Day to day business operations	Decision support at tactical/strategic level
Data latency	Real-time data	Periodic snapshots, incl. historical data
Design	Application oriented	Subject Oriented
Normalization	Normalized data	(Sometimes also) denormalized data
Data manipulation	Insert/Update/Delete/Select	Insert/Select
Transaction management	Important	Less of a concern
Type of queries	Many, simple queries	Fewer, but complex and ad-hoc queries

Data Warehouse Design

Modellen
- Conceptueel model: communicatie tussen partijen
- Logisch model: Implementatie

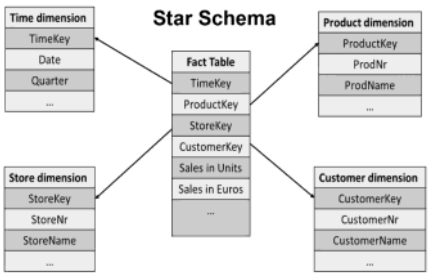
Datamodellen / Voorstellingen
- Transactionele Database
  - ER, UML, …

- Data Warehouses
  - Geen modellen beschikbaar!
  - Meestal direct designed op logisch level

Voorstelling Data Warehouse
- Star Schema
- Snowflake Schema
- Fact Cpnstellation

Star Schema

Star Schema
- 1 Basistabel (Fact Table) met links naar andere (Dimension) tabellen



Fact Table
- Tupel per transactie/event en bijhorende gegevens

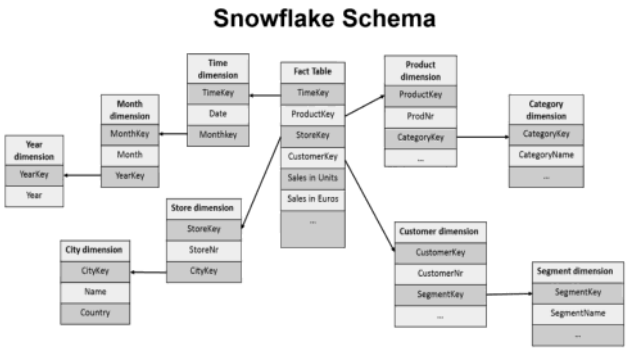
Dimension Table
- Informatie over links in de fact table (tijd, winkel, product, klant)
- Vaak niet-genormaliseerd

Snowflake Schema

Snowflake Schema

- 1 Basistabel (Fact Table) met gelinkte tabellen in hiërarchische orde
- Meer tabellen dan Star Schema
    - Meer SQL JOINS nodig!

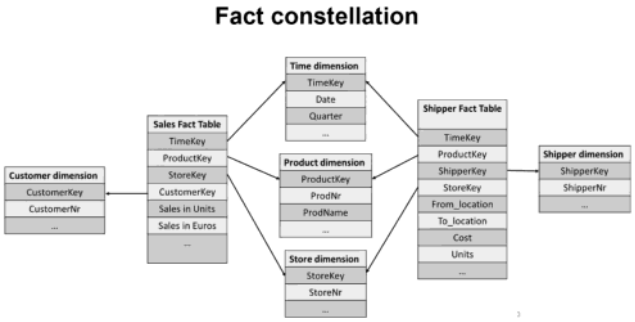
- Wanneer nuttig?
    - Als dimension tables te groot worden
    - Als queries niet (alle) dimension tables gebruiken



Fact Constellation

Fact Constellation

- 2 Fact Tables met gemeenschappelijke dimensions
- Galaxy / Meerdere Stars Schema



Problemen met Schema's

Surrogate Keys

- Gemaakt om te linken met andere tabellen
- Betekenisloze integers
- Bufferen data warehouse van operationeel systeem
- Niet gebruiken van business keys
    - Hebben een bepaalde betekenis binnen bedrijf
    - Zijn groter, worden vaak hergebruikt op langere termijn

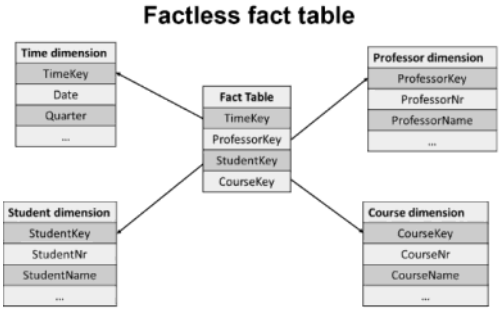
- Surrogate Keys besparen in plaats en efficiëntie

Granulariteit Fact Table

- Hoeveelheid details opgeslagen in fact table (korreligheid)
- Hogere granulariteit -> Meer rijen in fact table
- Balans tussen meer details en minder opslagcapaciteit

Factless Fact Table

- Fact Table met enkel Surrogate Keys (linken naar andere tabellen)



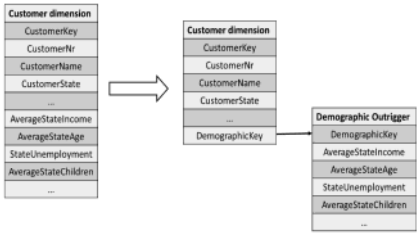
Junk Dimensions

- Omgaan met lage cardinaliteit attribuuttypes (flags/indicators)
- Boolean waarden, Yes/No
- Junk dimensions stelt combinaties van lage-cardinaliteit attribuuttype voor

JunkKey1	On-line purchase	Payment	Discount
1	Yes	Credit-card	Yes
2	Yes	Credit-card	No
3	No	Credit-card	Yes
4	No	Credit-card	No
5	No	Cash	Yes
6	No	Cash	No

Outtrigger Tables

- Set van attribuuttypes van dimensietabel met hoge verbondenheid, lage cardinaliteit en welke regelmatig wordt bijgewerkt



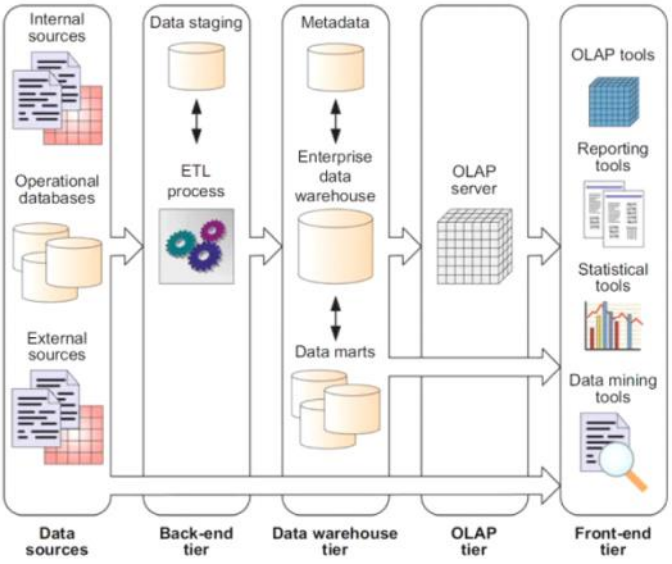
Slowly Changing Dimensions

- Dimensies die wijzigen overheen de tijd
- Hoe aanpakken in data?
    - Wijzigen van records naar nieuwe versie
    - Nieuw record toevoegen en oude als "niet-actief" markeren
    - Binnen record nieuw veld aanmaken met vernieuwde dimensie
    - Nieuw record toevoegen en bijhouden van start- en eindtijd van een record

Rapidly Changing Dimensions

- Dimensies die snel wijzigen op korte tijd
- Nieuwe records aanmaken zou snel zeer veel opslag vragen
- Oplossen door opsplitsen record in mini-dimension table

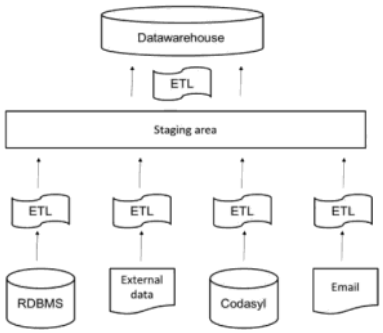
Data Warehouse Architecture



The Extracction Transformation and Loading ETL Process

Extraction Transformation and Loading ETL

- Data halen uit bepaalde bronnen, transformeren naar juiste formaat voor staging area
- Doet al 80% van het werk om een datawarehouse te creëren
- Per bron een andere ETL



- Extractie uit bron
  - Volledig: Alles uit bron halen
  - Incrementeel: Enkel wat gewijzigd is sinds vorige versie

- Transformatie naar Staging
  - Vorm/Formaat wijzigen
  - Cleansing: Omgaan met fouten/onvolledigheid
  - Samenvoegen/scheiden van data

- Laden naar data warehouse
  - Fact- en dimensiontabellen invullen

- Documentatie en metadata

Nagekeken en vervolledigd op 25/05/2024

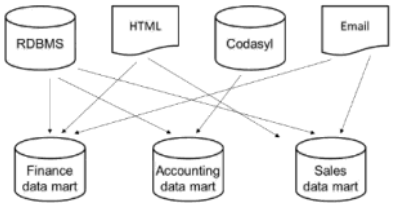
Data Warehousing and Business Intelligence

Data Marts

Data Mart
- Kleine vorm van data warehouse
- Bevat specifieke info voor bepaalde kleinere groep van eindgebruikers
- Gericht op gebruik en specifieke queries

- Afhankelijk
  - Directe data uit Data Warehouse

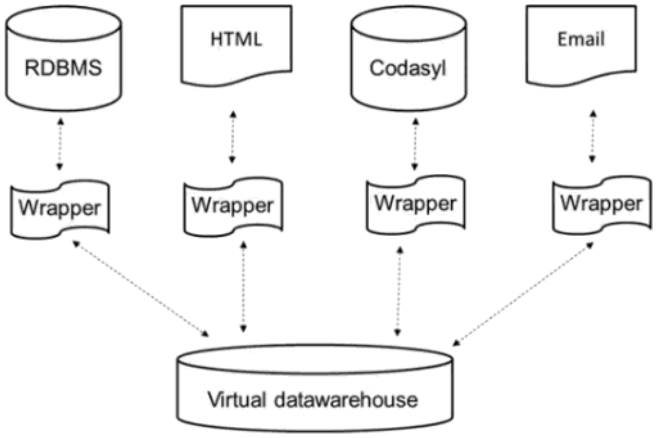
- Onafhankelijk
  - Data uit operationele systemen, externe bronnen of combinatie



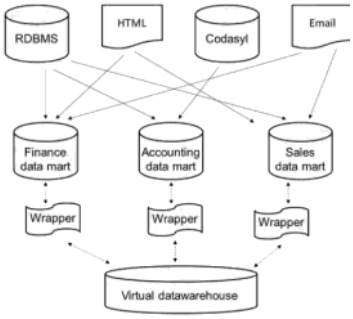
Mogelijk probleem met data mart?
- Inconsistente data
- Data moet steeds overal geüpdatet worden bij een aanpassing, hoe kan dat?
- Oplossing: Virtualisatie

Virtualisatie
- Middelware gebruiken om data uit data stores te halen en weer te geven
- Data blijft in originele bron en wordt enkel weergegeven in data mart
- Gebruikt geen fysieke opslag

Virtuele Data Warehouse
- Set van queries (SQL Views) die virtuele data warehouse genereren
  - Wrappers zorgen voor transformatie queries naar data warehouse



- Als extra laag op onafhankelijke fysieke data marts



Nadelen virtualisatie

- Extra processen nodig van de bronnen
- Geen geschiedenis van data, volledig afhankelijk van databronnen

Operational Data Storage ODS

Operational Data Store ODS

- Stagingomgeving met query mogelijkheden
- Goed voor analyses die real-time data nodig hebben
- Complexere analyses wel op data warehouse zelf

Data Lake

	Data warehouse	Data lake
Data	Structured	Often unstructured
Processing	Schema-on-write	Schema-on-read
Storage	Expensive	Low cost
Transformation	Before entering the DW	Before analysis
Agility	Low	High
Security	Mature	Maturing
Users	Decision makers	Data Scientists

Data Warehouse Usage

Business Intelligence (BI)

- Groepering van activiteiten, technieken en tools om patronen te herkennen in data en zo toekomstige voorspellingen te doen

- Garbage In, Garbage Out (GIGO)

Types BI

- Verificatie Gerichte BI
    - Queries en reporting
    - Pivot Tabellen
    - Online analytical Processing (OLAP)

- Ontdekkingsgerichte BI
    - Analyses

Query en Reporting

- Gebruiker kan grafisch en interactief vragen stellen en reports opstellen
- Serf Service BI (Je hoeft geen dataspecialist/ITer te zijn)
- Query By Example QBE
    - Query omgezet in een gebruiksvriendelijke vorm

- Report kan vernieuwd worden op elk moment
- Visuele technieken en weergaves

Pivot Table

- Data samenvatten in een tabel
- Kruistabellen, meerdimensionale tabellen voorstellen in tweedimensionale tabel

Sales		Quarter				Total
		Q1	Q2	Q3	Q4	
Region	Europe	100	200	50	100	450
	Africa	50	100	200	50	400
	Asia	20	50	10	150	230
	America	50	10	100	100	260
	Total	220	360	360	400	1340

On-Line Analytical Processing (OLAP)

- Interactieve analyse van data, samenvatten en visueel voorstellen
- Tool om queries te doen voor gebruikers
- Verschillende Types
    - MOLAP
        - Multidimensionaal OLAP
        - Multidimensionale data opslaan met multidimensionale DBMS (MDBMS)
        - Data in multidimensionale array-structuur
        - Snel in data opzoeken, heeft meer plaats nodig
        - Niet geoptimaliseerd voor operaties (invoegen/verwijderen)
        - Geen SQL-achtige mogelijkheid

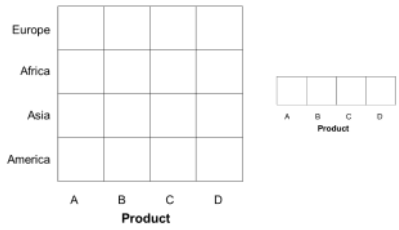
Array (key, value)	Q1	Q2	Q3	Q4	Total
Product A	(1,1) 10	(1,2) 20	(1,3) 40	(1,4) 10	(1,5) 80
Product B	(2,1) 20	(2,2) 40	(2,3) 10	(2,4) 30	(2,5) 100
Product C	(3,1) 50	(3,2) 20	(3,3) 40	(3,4) 30	(3,5) 140
Product D	(4,1) 10	(4,2) 30	(4,3) 20	(4,4) 20	(4,5) 80
Total	(5,1) 90	(5,2) 110	(5,3) 110	(5,4) 90	(5,5) 400

    - ROLAP
        - Relational OLAP
        - Data opslaan in relationele data warehouse dmv star, snowflake of fact constellation schema
        - Meer gestandardiseerd, SQL-mogelijkheid
        - Kan beter om met meerdere dimensies

    - HOLAP
        - Hybrid OLAP
        - Combinatie van MOLAP en ROLAP
        - Data in RDBMS in relationeel data warehouse
        - OLAP analyse start van multidimensionale database
        - Combinatie van performance van MOLAP en grootte¨/scalability van ROLAP

Operaties

- Roll-Up



(omgekeerde Drill-down)

- Drill-Across
    - Info van 2 of meer fact tables gebruiken

- Slicing
    - Bepaalde dimensie op een waarde zetten (filteren)

- Dicing
    - Selectie van range in één/meerdere dimensies

OLAP Queries in SQL

Extensies van GROUP BY in SQL

- CUBE
    - Berekent unie van elke subset van attribuuttype

- ROLLUP
    - Unie van elke prefix van attribuuttype, van detail naar algemeen

- GROUPING SETS
    - Gelijk aan UNION ALL van meerdere GROUP BY statements

Zie slides en oefeningen voor code!

Nagekeken en vervolledigd op 25/05/2024

Les 10 - Transactions

Liam Luys – r0934449 - 2e BA Informatica
Gegevensbanken

donderdag 23 mei 2024 14:38

Basics of Transaction Management

Transactions, Recovery andn Concrrency Control

Databases

- Meestal meerdere-gebruikers databases
- Gelijktijdige toegang tot dezelfde data
    - Leidt tot mogelijke inconsistentie

- Errors binnen DBMS of omgeving
- DBMS moet ACID richtlijnen opvolgen

Transactie

- Set van database-operaties gegeven door gebruiker of applicatie dat wordt gezien als ondeelbare eenheid van werk
- Operatie die in één stap atomair moet verlopen
- Faalt volledig of voert succesvol uit
- Behoudt consistentie van database

Recovery

- Bij problemen de database herstellen in consistente status
- Zonder dataverlies

Concurrency Control

- Gelijktijdige transacties beheren
- Voorkomen van inconsistentie

Transactions and Transaction Management

Transactie Grenzen

- Expliciet
    - $begin.transactie$ en $end.transactie$

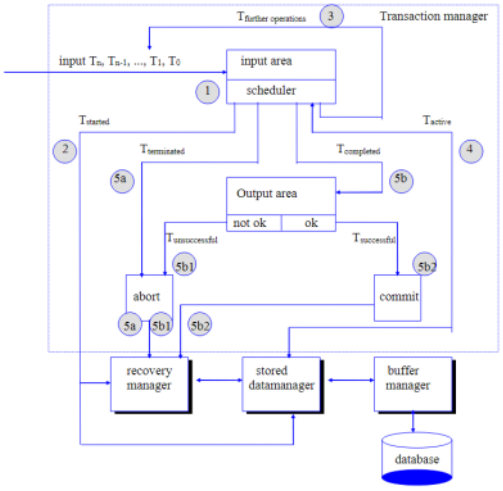
- Impliciet
    - Eerste uitvoerbaar SQL statement

  Transactie is actief eens de eerste operatie is uitgevoerd

Uitgevoerde Transactie

- Succesvol --> Commit
- Mislukt --> Roll back

DBMS Componenten voor Transactie Management



Logfile

- Registratie van data
  - Unieke log-id
  - Transactie-identifier
  - Info over transactie: startpunt, tijdsduur, records betrokken, …

- Before en After images van alle records betrokken in transactie
- Huidige status van transactie (Actief/Commited/Afgebroken)

- Write Ahead Log Strategie
  - Updates geregistreerd voor disk-aanpassingen

Recovery

Fouten in uitvoering

- Transactiefout
  - Door fout in logica van transactie-operaties en/of applicatie

- Systeemfout
  - Besturingssysteem van database crasht

- Mediafout
  - Secundaire opslag beschadigd of onaanspreekbaar

Oplossen van fouten

Systeem Herstel

- Bij systeemfout 2 mogelijke statussen uitvoering
  - Nog in Actieve Status
  - Al in de 'committed' status voor de fout optrad

- Kijken naar logfile
  - Welke transactie(s) waren in uitvoering?
  - UNDO en REDO

- Database buffer flushing strategy heet inpact op UNDO en REDO!

Media Herstel

- Gebaseerd op data redundantie

- Offline of online media (backups)

- Afweging tussen kost van redundante data en de tijd nodig om systeem te herstellen

- Twee Soorten Media Herstel
  - Disk Mirroring
    - Gelijktijdig schrijven naar 2 of meer fysieke disks
    - Hogere operatie-kost
    - Weinig negatieve impact op schrijf performantie

  - Archiving
    - Databasebestanden tijdelijk kopiëren naar ander medium
    - Afweging kost van backups en dataverlies
    - Volledig of incrementeel backup

Rollforward Recovery
- Combinatie Disk Mirroring en Archiving
- Database archiveren en logfile mirroring

Eventual Consistency
- Laten tijdelijke inconsistentie toe, in ruil voor verhoogte performantie
- Gebruikt in NoSQL Databases

Concurrency Control

Gelijktijdigheid
- Nodig voor efficiënt gebruik systeem
- Verantwoordelijkheid scheduler

Problemen Gelijktijdigheid
- Verlies van updates
  - Andere update overschrijft een update

- Afhankelijkheidsprobleem
  - Transactie leest data die wordt bewerkt door andere gebruiker (uncomitted)
  - "Dirty Read Problem"

- Inconsistentie analyses
  - Lezen van data die gelijktijdig wordt bewerkt

- Nonrepeatable Read
  - Transactie leest zelfde data meerdere keren maar krijgt telkens andere data door updates van andere transactie

- Phantom Reads
  - Insert/Delete operaties op data die een andere transactie leest

Schedule
- Verzameling transacties en ordening van statements in de transacties met volgende voorwaarde
  - Voor elke transactie T in de planning en voor alle statements s_1 en s_2 in die transactie geldt: Als s_1 voorafgaat aan s_2 in T, wordt s_1 voor s_2 gepland in uitvoering

- Volgorde van individuele statements binnen elke transactie
- Arbitraire volgorde van statements tussen transacties

Serial Schedule

- Als alle statements van dezelfde transactie zonder tussenkomst van andere transactie worden ingepland
- Voorkomen van parallelle uitvoering

Doel = Non-serial correct schedule

Serializable Schedule

- Non-serial schedule dat equivalent is aan een serial schedule
- Equivalente schedules:
  - *For each operation $read_x$ of T_i in S_1 the following holds: if a value x that is read by this operation, was last written by an operation $write_x$ of a transaction T_j in S_1, then that same operation $read_x$ of T_i in S_2 should read the value of x, as written by the same operation $write_x$ of T_j in S_2*
  - *For each value x that is affected by a write operation in these schedules, the last write operation $write_x$ in schedule S_1, as executed as part of transaction T_y should also be the last write operation on x in schedule S_2, again as part of transaction T_y.*

- Testen door opstellen van "precedence graph"
  - Indien een lus in graaf --> schedule is niet serializable

Optimistisch / Pessimistisch Schedule

- Optimistisch
  - Conflicten tussen gelijke uitvoeringen zijn zeldzaam
  - Transacties ingepland zonder vertraging
  - Transactie is geverifieerd op fouten als het wordt gecommit
  - Geen fouten --> Transactie commit, anders roll back

- Pessimistisch
  - Fouten in transacties komen vaker voor
  - Transacties inplannen met vertragingen om fouten te vermijden
  - Verminderen doorvoer van instructies
    Vb. Serial scheduler

Timestamping

- Gebruiken om volgorde van operaties te bekijken

Locks

- Gebruiken om gelijktijdigheid te bewaren
- Gebruiken om fouten tijdens uitvoering te detecteren

Locking Mechanisme

- In situaties waar meerdere transacties dezelfde data gebruiken, zorgen dat het in een manier kan zonder fouten
- Variabele invoeren op data-object
  - Waarde bepaald welke operaties mogelijk zijn op data-object

- Lock Manager
  - Verantwoordelijk voor locking en unlocking

Soorten Locks

- Exclusive Lock
  - Slechts één transactie toegang tot data-object
  - Kan schrijven, lezen, …
  - "Write-Lock" / "X-Log"

- Shared Lock
  - Meerdere transacties toegang tot data-object
  - Garantie dat object niet wordt bewerkt
  - "Read-Lock" / "S-Lock"

Compabiliteitsmatrix

Type of lock requested	Type of lock(s) currently held on object: unlocked	shared	exclusive
unlock	-	yes	yes
shared	yes	yes	no
exclusive	yes	no	no

Locking Protocol

- Geeft aan welke lock in welke situatie wordt toegekend
- Door locking manager
- Moet "Eerlijk" kunnen gebeuren
    - Starvation vermijden, zorgen dat alle transacties mogelijkheid hebben tot lock

Vb. Two-Phase Locking

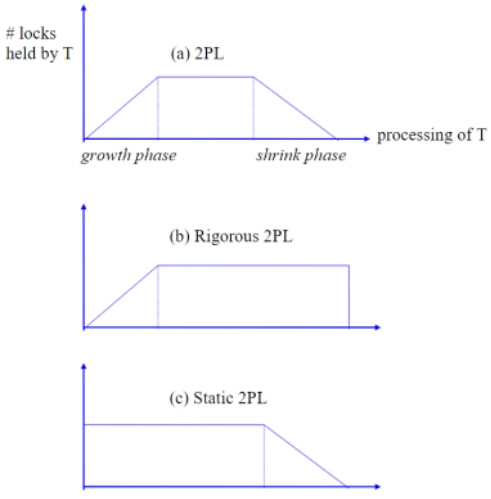
Two-Phase Locking Protocol

- Voor transactie kan lezen, shared/Exclusive lock nodig
- Lock manager bepaald via compabiliteitsmatrix of toegang wordt aanvaardt
- Opzetten/Vrijgeven lock in 2 fases
    - Growth Fase --> Locks kunnen opgezet maar niet vrijgegeven worden
    - Shrink Fase --> Locks worden vrijgegeven, geen locks worden opgezet

Varianten

- Rigorous 2PL
    - Locks behouden tot commit transactie

- Static 2PL
    - Transactie zet lock op aan start van transactie



Deadlock

- 2 of meerdere transacties wachten op elkaars lock
- Preventief voorkomen
    - Static 2PL: Transactie start locks bij start uitvoering

- Detectie
    - Wait For Graph
        - Lus in graaf is deadlock

Isolatie Niveaus

- 2PL zorgt voor sterke isolatie van transacties
  - Lange-termijn Lock
    - Lock behouden tot transactie afgerond is
  - Korte-termijn Lock
    - Enkel op bepaalde operaties
    - Gaat in tegen 2PL maar verbetert doorloop en efficiëntie!

- Read Uncommitted
  - Laagste niveau, Lange-termijn locks niet van toepassing
  - Aanname dat geen fouten voorvallen
  - Gebruikt bij enkel read-only operaties

- Read Committed
  - Lange-termijn locks voor schrijven, korte-termijn locks voor lezen
  - Voorkomt 'lost-update' probleem

- Repeatable Read
  - Lange-termijn locks voor zowel schrijven als lezen
  - Probleem van phantom reads nog steeds aanwezig

- Serializable
  - Lost phantom reads op!
  - Leunt niet meer aan bij implementatie 2PL

Isolation level	Lost update	Uncommitted dependency	Inconsistent analysis	Nonrepeatable read	Phantom read
Read uncommitted	Yes	Yes	Yes	Yes	Yes
Read committed	No	No	Yes	Yes	Yes
Repeatable read	No	No	No	No	Yes
Serializable	No	No	No	No	No

Lock Granularity

- Afweging tussen locking overhead en doorloop transacties

- Myltiple Granularity Licking MGL
  - Protocol dat conflicten tussen locks voorkomt
  - Voor een lock te plaatsen, een intentie-lock plaatsen
  - Toevoegen van nieuwe locks-methoden
    - Intention Shared Lock
    - Intention Exclusive Lock
    - Shared and intention exclusive lock

Type of lock(s) currently held on object

Type of lock requested	*unlocked*	*is-lock*	*ix-lock*	*s-lock*	*six-lock*	*x-lock*
unlocked	-	yes	yes	yes	yes	yes
is-lock	yes	yes	yes	yes	yes	no
ix-lock	yes	yes	yes	no	no	no
s-lock	yes	yes	no	yes	no	no
six-lock	yes	yes	no	no	no	no
x-lock	yes	no	no	no	no	no

The ACID Properties of Transaction

ACID Eigenschappen

- Atomicity
  - Meerdere database operaties in één atomaire stap uitvoeren

- Consistency
    - Behouden van consistente staat van database overheen uitvoeringen

- Isolation
    - Uitkomst van gelijktijdige uitvoering is gelijk aan afzonderlijke uitvoeringen na elkaar

- Durability
    - Effecten van uitgevoerde transactie steeds persistent op de database

Les 11 - API 1

donderdag 23 mei 2024 14:38

Liam Luys – r0934449 - 2e BA Informatica
Gegevensbanken

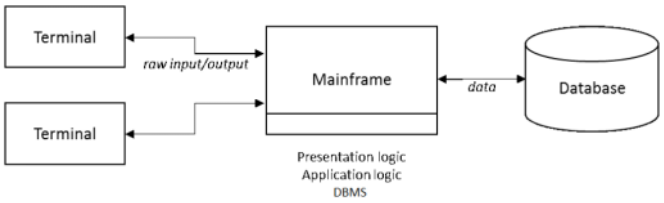
KU LEUVEN

Accessing Databases and Database APIs

Database System Architectures

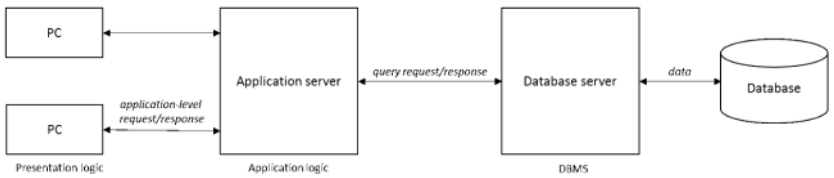
Gecentraliseerde Architectuur

- Alle verantwoordelijkheden van DBMS centraliseren naar 1 entiteit (mainframe)
- Vandaag de dag moeilijk en duur om te bekomen



Tiered Architectuur

- Combineren van rekenkracht centrale computer en flexibiliteit van PCs
- Varianten
    - Two-Tier Architectuur / Client-Server Architectuur
    - Three-Tier Architectuur: aparte laag voor applicatie-logica van DBMS



Fat Client Architectuur

- Presentatielogica en applicatie logica door de klant geregeld
- DBMS volledig op database server

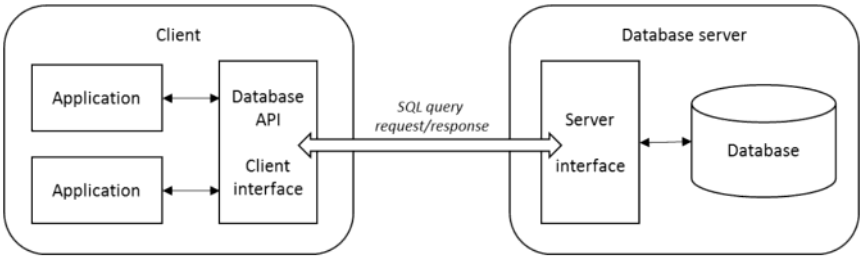
Thin Client Architectuur

- Enkel Presentatielogica door de klant geregeld
- Applicaties en database op de server

Database APIs

Application Programming Interface API

- Voorzien door DBMS
- Maakt mogelijk dat klant-applicaties gebruik maken van DBMS-services en functies
- Interface waarmee klant query en DBMS-functies kan aanspreken



Soorten API

- Proprietary
  - DBMS specifieke API
  - Klant moet op de hoogte zijn van de DBMS en eigenschappen

- Universal API
  - Gemakkelijk om te werken met meerdere DMBSs, overheen meerdere

Embedded/Call-level API

- Embedded
  - Embed van SQL statements in host programmeertaal
  - SQL wordt deel van de broncode van het programma (SQL Pre-Compiler)
  - Voordelen
    - Pre-compiler doet check op fouten en syntax
    - Early binding --> Efficiënte queries

  - Nadelen
    - Moeilijk om te beheren
    - Niet populair

- Call-Level
  - Oproepen van bepaalde functies/procedures of methodes uit de API
  - Geen directe SQL in de code
  - Late Binding

Early / Late Binding

- Vertaling van SQL naar lager-niveau representatie in DBMS
- Early Binding
  - Voor de uitvoering SQL binding (Static SQL)
  - Mogelijk bij gebruik van pre-compiler en embedded API
  - Slechts één keer binding door pre-compiler

- Late Binding
  - Tijdens uitvoering SQL binding (Dynamische SQL)
  - Geen check op syntax errors
  - Moeilijker om te testen, minder efficiënt

	Embedded APIs	Call-level APIs
Early binding **(“static” SQL)**	**Possible as a pre-compiler is used** •Performance benefit, especially when the same query must be executed many times •Pre-compiler detects errors before the actual execution of the code •SQL queries must be known upfront	**Only possible through stored procedures**
Late binding **(“dynamic” SQL)**	**Not used with embedded APIs**	**Necessary as no pre-compiler is used** •Flexibility benefit: SQL statements can be dynamically generated and used during execution •Errors are only detected during program execution •Possibility to use prepared SQL statements to perform binding once during execution

Universal Databases API

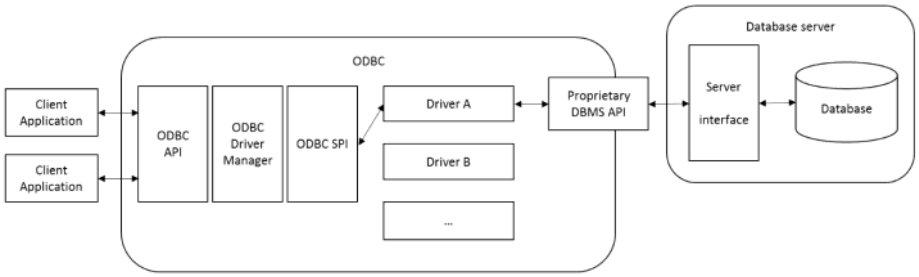
- Verschillende standaarden overheen jaren
- Verschillen op
  - Embedd/Call Level
  - Programmeertaal
  - Functionaliteiten

- Voorbeelden
  - ODBC
  - OLE DB
  - ADO.NET
  - JDBC

Open DataBase Connectivity ODBC

- Door Microsoft
- Gemeenschappelijke, uniforme interface voor verschillende DBMS
- 4 Basiscomponenten
  - ODBC API
  - ODBC Driver Manager
  - Database Driver
  - Service Provider Interface SPI

- Nadelen
  - Draait enkel op Microsoft platformen
  - Gebaseerd op C programmeertaal, niet gebruiksvriendelijk
  - Performancedaling door extra laag in structuur



Object Linking and Embedding for Databases OLE DB

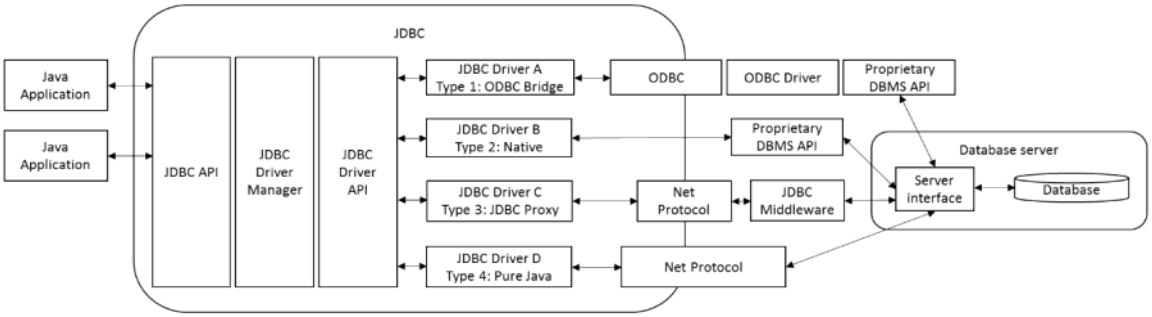
- Uitbreiding ODBC voor uniforme toegang van databronnen
- Mogelijkheden voor object databases, spreadsheets, andere bronnen…
- Poging van Microsoft naar "universal data access"

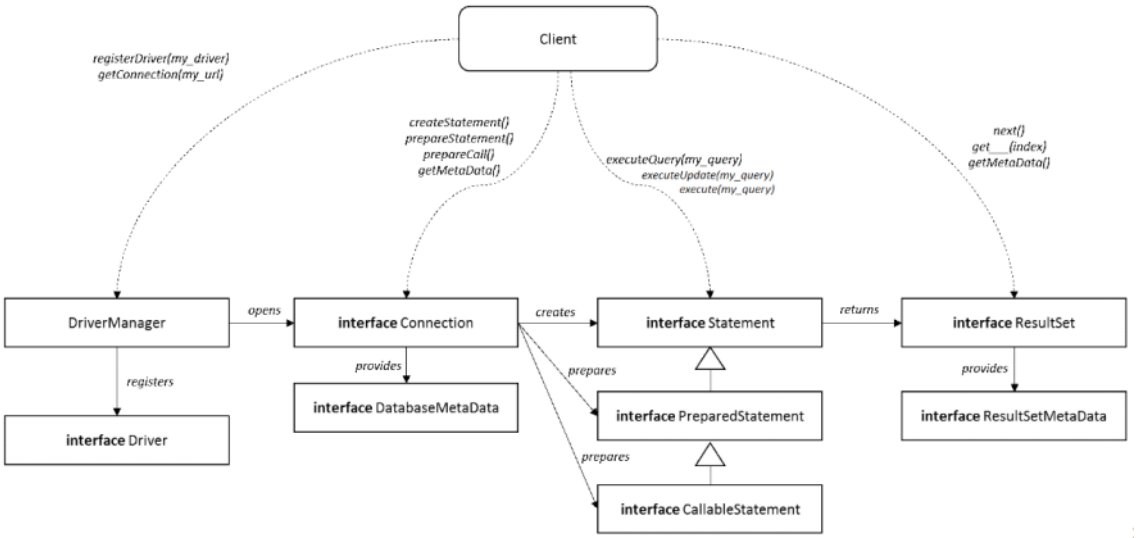
- Combinatie mogelijk met ADO

ActiveX Data Objects ADO
- Biedt betere, programmeervriendelijke model bovenop OLE DB

ADO.NET
- Combinatie OLE DB en ADO
- Data Providers

Java DataBase Connectivity JDBC
- Geinspireerd door ODBC
- Volledig in Java geprogrammeerd (object-georienteerd)
- Biedt object interfaces om drivers, connecties, SQL uit te drukken





Zie uitleg slides en handboek

- Nadeel JDBC
  - Gebrek aan compile-time controle en validatie (geen syntactische/semantische controle)

Language-Integrated Querying
- In programmeertaal language-native expressies gebruiken voor queries
- Worden vertaald naar SQL om DBMS aan te spreken

- Voorbeeld: jOOQ, QueryDSL, LINQ, …

Nagekeken en vervolledigd op 25/05/2024

Les 12 - API 2

maandag 13 mei 2024 10:05

Liam Luys – r0934449 - 2e BA Informatica
Gegevensbanken

KU LEUVEN

Accessing Database API

Object Persistence and Object-Relational Mappers

API Technologie
- Zoals JDBC, ADO.NET
- Stelt database elementen voor in object-georiënteerde manier

Object Persistence
- Domeinelementen voorstellen als plain objects
- Gebruik maken van programmeertaal

Object Persistence API
- Ook beschrijven van volledige bedrijfsdomein in de host language
- Mogelijk maken van efficiënte queries

Object Relational Mapping ORM
- Converteren van relationele database naar object-georiënteerde programmeertaal

Persistentie met Enterprice JavaBeans (EJB)

JavaBean
- Javaklasse binnen API, object-georiënteerd software componenten
- Zelf configureerbaar (plug and play)
- Enterprice Edition Platform
    - Koppeling tussen bedrijfslogica een klantapplicaties

Types Enterprice Beans
- Session Bean: niet presistent, logische uitvoering zonder op te slaan
- Message-Driven Bean: asynchrone communicatie
- Entity Beans: persistente bean, representatie van bepaalde entiteiten

Zorgen voor persistente bean
- Bean-Managed Persistence BMP
    - Persistentie codering overlaten aan programmeur

- Container-Managed Persistence CMP
    - Zorgt zelf (EJB) voor persistentie van objecten

Persistentie met Java Persistence API (JPA)

Annotation	Description
@Entity	Declares a persistent POJO class
@Table	Allows explicitly specifying the name of the relational table to map the persistent POJO class to
@Column	Allows explicitly specifying the name of the relational table column
@ID	Maps a persistent POJO class field to a primary key of a relational table
@Transient	Allows to define POJO class fields that are transient and should not be made persistent

Java Persistence API JPA

- Eenvoudige setup voor persistente objecten
- Eigen query language: JPQL
- Vooral gericht op relationele DBMS

Persistentie met Java Database Objects (JDO)

Java Database Objects

- Geen invloed van technologie DBMS (relationeel, …)
- Eigen query language: JDOQL

Java API Summary

Technology	Embedded or call-level	Early or late binding	Objects in host programming language represent	Data sources	Other
ODBC	Call-level	Late binding, though prepared SQL statements possible as well as calling stored procedures	A result set with rows of fields	Mainly relational databases, though other structured tabular sources possible as well	Microsoft-based technology, not object-oriented, mostly outdated
JDBC	Call-level	Late binding, though prepared SQL statements possible as well as calling stored procedures	A result set with rows of fields	Mainly relational databases, though other structured tabular sources possible as well	Java-based technology, portable, still in wide use
SQLJ	Embedded	Early binding	A result set with rows of fields	Relational databases supporting SQL	Java-based technology, uses a pre-compiler, mostly outdated

Language-integrated query technologies	Uses an underlying call-level API	Uses an underlying late-binding API	A result set with rows of fields, sometimes converted to a plain collection of objects representing entities	Relational databases supporting SQL or other data sources	Examples: jOOQ and LINQ, works together with another API to convert expressions to SQL
OLE DB and ADO	Call-level	Late binding, though prepared SQL statements possible as well as calling stored procedures	A result set with rows of fields	Mainly relational databases, though other structured tabular sources possible as well	Microsoft-based technology, backwards compatible with ODBC, mostly outdated
ADO.NET	Call-level	Late binding, though prepared SQL statements possible as well as calling stored procedures	A result set with rows of fields provided by a DataReader, or a DataSet: a collection of tables, rows, and fields, retrieved and stored by DataAdapters	Various data sources	Microsoft-based technology, backwards compatible with ODBC and OLE DB
Enterprise JavaBeans (EJB 2.0)	Uses an underlying call-level API	Uses an underlying late-binding API	Plain Java entity Beans as the main representation	Mainly relational databases, though other structured tabular sources possible as well	Java-based technology, works together with another API to convert expressions to SQL
Java Persistence API (JDA in EJB 3.0)	Uses an underlying call-level API	Uses an underlying late-binding API	Plain Java objects as the main representation	Mainly relational databases, though other structured tabular sources possible as well	Java-based technology, works together with another API to convert expressions to SQL
Java Database Objects (JDO)	Uses an underlying call-level API	Uses an underlying late-binding API	Plain Java objects as the main representation	Various data sources	Java-based technology
ORM APIs (ActiveRecord, Entity Framework, SQL Alchemy)	Uses an underlying call-level API	Uses an underlying late-binding API	Plain objects defined in the programming language as the main representation	Relational databases	Various implementations available for each programming language

Database Access in the World Wide Web

Standaard API

- HTTP verzoeken naar webserver
- Antwoord met content voor de klant
    - HTML-opmaak
    - XML, JSON, YAML, …

Common Gateway Interface CGI

- Voorstel technologie om dynamische pagina's te construeren
- Basisvorm interactie klant en server via HTML forms
- Elk klantverzoek start nieuw proces --> Nadelig voor server en resources
- Voorloper PHP

Vandaag de dag

- Populaire technologie als HTML, CSS, JavaScript
- Toevoegingen als AJAX (Asynchronous JavaScript en XML) verhogen efficiëntie
- REST API (HTTP protocol)

Examen

Examen

- Multiple choise
- 30 vragen, giscorrectie

Succes!

Nagekeken en vervolledigd op 25/05/2024