Haitham A. El-Ghareeb

July 14, 2019

Faculty of Computers and Information Sciences
Mansoura University
Egypt
helghareeb@mans.edu.eg

Contacts

- https://youtube.com/helghareeb
- https://www.github.com/helghareeb
- http://eg.linkedin.com/in/helghareeb
- https://www.haitham.ws
- h.elghareeb@yahoo.com

Objectives

Objectives

After completing this chapter, you will be able to:

1. Discuss data modeling and why data models are important
2. Describe the basic data-modeling building blocks
3. Define what business rules are and how they influence database design
4. Understand how the major data models evolved
5. List emerging alternative data models and the needs they fulfill
6. Explain how data models can be classified by their level of abstraction

Overview

Overview

1. Data Modeling is the first step in DB Design
2. Serves as a bridge between real-world objects and the computer DB
3. Designers, Programmers, and end users see data in different ways
4. Consequently, different views of the same data can lead to database designs that do not reflect an organization's actual operations
5. DB Designers must obtain a precise description of the data nature and many uses
6. Data Modeling clarifies communication between DB System user stack

What we will Learn

1. Basic Data-Modeling Concepts
2. How current data-models developed from earlier models
3. Data Models in chronological order [1]
   - 3.1 Hierarchical Data Model
   - 3.2 Network Data Model
   - 3.3 Relational Data Model (ER Model - ERD)
   - 3.4 Object-Oriented Data Model
   - 3.5 Object/Relational Data Model
   - 3.6 NoSQL Data Model(s)
4. Data Abstraction

[1] (of a record of events) following the order in which they occurred. "the entries are in chronological order"

Data Modeling and Data Models

DB Design

1. focuses on how DB structure will be used to store and manage end-user data

Data Modeling

1. the first step in designing a database
2. refers to the process of creating a specific data model for a determined problem domain
3. iterative process
4. when done properly, the final data model effectively is a 'blueprint' with all instructions to build DB

Problem Domain

1. a clearly defined area within the real-world environment, with a well-defined scope and boundaries that will be systematically addressed

Data Model

1. relatively simple representation
2. usually graphical
3. of more complex real-world data structures
4. Represents
    - data structures
    - characteristics
    - relations
    - constraints
    - transformations
    - other constructs
5. with the purpose of supporting a specific problem domain

Implementation Data Model

Should include, at least:

1. Description of the data structure that will store the end-user data
2. Set of enforceable rules to guarantee the integrity of the data
3. Data manipulation methodology to support the real-world data transformations

Model

1. abstraction of a more complex real-world object or event
2. Model's main function is to help us understand the complexities of the real-world environment

Data Modeling and Data Models

Definitions

Data Modeling

1. The process of creating a specific data model for a determined problem domain

Data Model

1. A representation, usually graphic, of a complex 'real-world' data structure
2. Data models are used in the database design phase of the Database Life Cycle

The Importance of Data Models

Data Models Importance

1. Facilitate interaction among the designer, application programmer, and end user
2. Foster improved understanding of the organization
3. Data Models are a communication tool
4. Data constitutes the most basic information employed by a system
5. Applications are created to manage data and to help transform data into information
6. Data is viewed in different ways by different people

Manager vs. Clerk

1. Although both work for the same company
2. Manager is more likely to have an enterprise-wide view of company data than clerk
3. Even different managers view data differently

Company President

1. Take a universal view of the data
2. He must be able to tie the company's divisions to a common DB vision

Purchasing Manager vs. Inventory Manager

1. Have more restricted view of the data
2. Similar to Company's Inventory Manager
3. Each department manager works with a subset of the company's data
4. Inventory Manager: More concerned about inventory levels
5. Purchasing Manager: More concerned about the cost of items and about relationships with the suppliers of those items

Applications Programmers

1. Another view of data
2. Concerned with:
    - Data Location
    - Formatting
    - Specific Reporting Requirements
3. Translate company policies and procedures from a variety of sources into appropriate
    - Interfaces
    - Reports
    - Query Screens

Data Model Basic Building Blocks

Data Model Basic Building Blocks

Basic Building Blocks

Basic Building Blocks

1. Entities
2. Attributes
3. Relationships
4. Constraints

Data Model Basic Building Blocks

Entity

Entity

1. A person, place, thing, or event about which data will be collected and stored
2. Represents a particular type of object in the real world
3. Entity is distinguishable: each entity occurrence is unique and distinct
    - STUDENT entity would have many distinguishable students occurrences
4. May be: Physical objects or Abstractions

Data Model Basic Building Blocks

Attribute

Attribute

1. Characteristic of an entity
2. STUDENT entity would be described by attributes such as
    - first_name
    - last_name
    - phone_number
    - address
3. Attributes are the equivalent of fields in file systems

Data Model Basic Building Blocks

Relationship

Relationship

1. Describes an association among entities
2. Example: relationship exists between Doctors and Courses that can be:
   - A Doctor teach Many Courses
   - Many Courses are taught by One Doctor
3. Relationships are identified in both directions

Relationship Types

1. **One-to-Many** 1:M or 1 .. M
2. **Many-to-Many** M:N or M .. N
3. **One-to-One** 1:1 or 1 .. 1

One-to-Many

1. 1:M or 1 .. *
2. Order Contains Many Order Items
3. Each Order Item is Contained in One Order

Many-to-Many

1. M:N or * .. *
2. Student registers Many Courses
3. Each Course is registered by Many Students

One-to-One

1. 1:1 or 1 .. 1
2. Each Department is Managed by One Professor
3. Every Professor Manages only a single Department

Data Model Basic Building Blocks

Constraint

Constraint

1. Restriction placed on the data
2. Important
3. Help to ensure data integrity

Data Model Basic Building Blocks

Question

Question

How do we properly identify ...

1. Entities
2. Attributes
3. Relationships
4. Constraints

Answer

1. Clearly identify the Business Rules for the Problem Domain we are Modeling

Data Model Basic Building Blocks

Definitions

Entity

1. A person, place, thing, concept, or event for which data can be stored

Attribute

1. A characteristic of an entity or object
2. An attribute has a name and a data type

Relationship

1. An association between entities

One-to-Many (1:M or 1..*) Relationship

1. Associations among two or more entities that are used by data models
2. In a 1:M relationship, one entity instance is associated with many instances of the related entity

Many-to-Many (M:N or * .. *) Relationship

1. Association among two or more entities in which
   - one occurrence of an entity is associated with many occurrences of a related data entity, and
   - one occurrence of the related entity is associated with many occurrences of the first entity

One-to-One (1:1 or 1 .. 1) Relationship

1. Associations among two or more entities that are used by data models
2. In a 1:1 relationship, one entity instance is associated with only one instance of the related entity

Constraint

1. A restriction placed on data, usually expressed in the form of rules
2. Example: Student GPA must be between 0.00 and 4.00

Business Rules

Default

DB Designers need thorough information about

1. what types of data exist in an organization
2. how the data is used
3. what time frames it is used

However

1. Such data and information - by themselves
2. Do not yield the required understanding of the total business
3. From DB Perspective
    - Collection of data becomes meaningful only when it reflects properly defined business rules

Business Rules - Definition

- brief
- precise
- unambiguous
- description of
- policy
- procedure
- principle
- within a specific organization

Business Rules

1. Derived From: Detailed description of an organization's operations
2. Helps to: Create and enforce actions within that organization's environment
3. Must be: Rendered in writing and Updated to reflect any change in operations

Examples

1. A customer may generate many invoices
2. An invoice is generated by only one customer
3. A training session cannot be scheduled for fewer than 10 employees or for more than 30 employees

Business Rules

Discovering Business Rules

Main source of Business Rules

1. Company Managers
2. Policy Makers
3. Department Managers
4. Written Documentation
    - Company's Procedures
    - Standards
    - Operations Manuals
5. Direct interviews with End Users - Less Reliable

Business Rules Importance

1. Helps to standardize the company's view of data
2. Communication tool between users and designers
3. Allows the designer to understand the nature, role, and scope of the data
4. Allows the designer to understand business processes
5. Allows the designer to develop appropriate relationship participation rules and constraints and to create an accurate data model

Note

1. Not all business rules can be modeled
2. Example: No driver can drive more than 10 hours within any 24-hour period
3. Cannot be modeled in the DB model directly
4. However, such a Business Rule can be presented and enforced by application software

Business Rules

Definitions

Business Rule

1. A description of a policy, procedure, or principle within an organization.
2. For example, A Professor my teach up to four classes during a semester

Business Rules

From Business Rules to Data Model Components

Business Rules

1. Set the stage for the proper identification of
    - entities
    - attributes
    - relationships
    - constraints

In the real world

1. Names are used to identify objects
2. If business environment wants to keep track of the objects -¿
3. there will be specific business rules of the objects

General Rule

1. Noun in a business rule will translate into an entity in the model
2. Verb (active or passive) that associates the nouns will translate into a relationship among the entities

Example

1. Business Rule: A customer may generate many invoices
2. Two Nouns:
    - Customer
    - Invoices
3. One Verb:
    - Generate

Properly Identify Relationships

1. Relationships are Bidirectional
2. Ask two questions
   2.1 How many instances of B are related to one instance of A?
   2.2 How many instances of A are related to one instance of B?

Identify Relationships Example

1. How many classes can one student enroll?
    - Many Classes
2. How many students can enroll in one class?
    - Many Students
3. Relationship between student and class is Many-to-Many (M:N)

Business Rules

Naming Conventions

Entity Names

1. Must be
    - unique and distinguishable from other objects
2. Should be
    - Descriptive

Attribute Names

1. Should be
    - Descriptive
2. Good Practice
    - Prefix with the name or abbreviation of the entity

The Evolution of Data Models

Data Models Evolution

1. Several models
2. Each model attempt to resolve the previous model's critical shortcoming
3. Models represent schools of thought
4. Models represent
   - 4.1 What DB is
   - 4.2 What it should do
   - 4.3 Types of structures that it should employ
   - 4.4 Technology that would be used to implement these structures
5. Many of the 'new' DB concepts and structures resemble to the 'past'

Hierarchical Model

Hierarchical Data Model

1. Developed in the 1960s
2. Manage large amounts of data for complex manufacturing projects
3. Apollo Rocket (1969)
4. Basic Logical Structure: Upside-Down Tree
5. Contains Levels or Segments

Segment

1. Equivalent of a file system's record type
2. Higher layer is perceived as the parent of the segment directly beneath it, which is called: child
3. Depicts a set of one-to-many (1:M) relationships between Parent and its Children segments
4. Each parent can have many children, but each child has only one parent

Network Model

Network Model

1. Created to represent complex data relationships more effectively than hierarchical model
2. Improve data performance
3. Impose DB standard
4. User perceives the network DB as a collection of records in 1:M relationships
5. Allows record to have more than one parent
6. Generally, not used today
7. Presented important definitions

Schema

1. Conceptual organization of the entire DB as viewed by the DB administrator

Subschema

1. Portion of the DB 'seen' by application programs that actually produce the desired information from the data within DB

DML

1. Data Manipulation Language (DML)
2. Environment in which data can be managed and is used to work with the data in the DB

DDL

1. Data Definition Language (DDL)
2. enables DBA to define schema components

Challenges

1. Information needs grew
2. Absence of ad hoc queries
3. Structural change in DB produces havoc in all application programs

Network Model

Definitions

Hierarchical Model

1. An early database model whose basic concepts and characteristics formed the basis for subsequent database development
2. This model is based on an upside-down tree structure in which each record is called a segment
3. The top record is the root segment
4. Each segment has a 1:M relationship to the segment directly below it

Segment

1. In the hierarchical data model, the equivalent of a file system's record type

Network Model

1. An early data model that represented data as a collection of record types in 1:M relationships

Schema

1. A logical grouping of DB objects, such as
    - tables
    - indexes
    - views
    - queries
2. that are related to each other

Subschema

1. The portion of the DB that interacts with application programs

DML

1. Data Manipulation Language
2. The set of commands that allows an end user to manipulate the data in the DB
3. Such as
    - SELECT
    - INSERT
    - UPDATE
    - DELETE
    - COMMIT
    - ROLLBACK

DDL

1. Data Definition Language
2. The language that allows DBA to define the DB structure, schema, and subschema

Relational Model

Relational Model

1. Introduced in 1970
2. By E. F. Codd
3. IBM
4. Paper: A Relational Model of Data for Large Shared Data banks [2]
5. Mathematical Concept - Relation
6. Describes a precise set of data manipulation constructs based on advanced mathematical concepts

[2] E. F. Codd. 1970. A relational model of data for large shared data banks. Commun. ACM 13, 6 (June 1970), 377-387. DOI=10.1145/362384.362685 http://doi.acm.org/10.1145/362384.362685

Relation

1. Abstract Mathematical Notation: Relation
2. Concrete: Table
3. Two Dimensional structure composed of intersecting rows and columns

Tuple

1. Each row in a table is called: Tuple

Attribute

1. Each column in a table represents an attribute

Challenges

1. 1970
2. Ingenious, but impractical
3. Computer Overhead
4. Computers at that time lacked the power to implement the relational model
5. Luckily, OS and PCs power grew exponentially

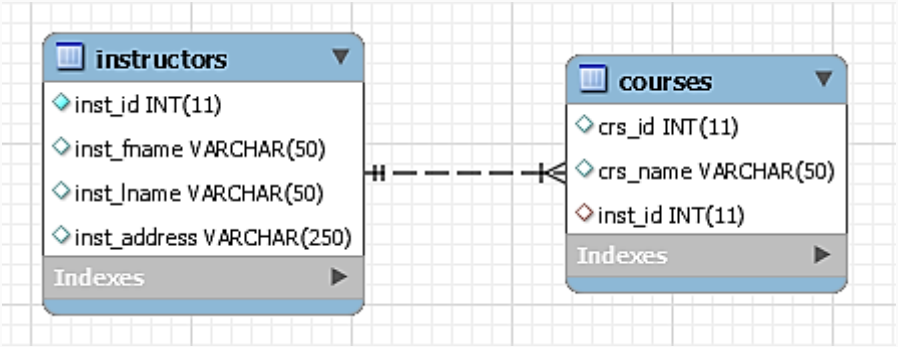
RDBMS

1. Relational Database Management System
2. Hide the complexities of the relational model from the user
3. Manages all of the physical details while the user sees the relational DB as a collection of tables in which data is stored

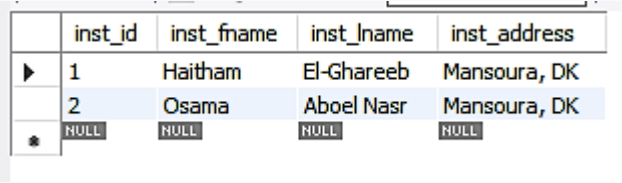
Relating Tables

1. Tables are related to each other through the sharing of a common attribute (a value in a column)
2. Example: COURSE and INSTRUCTOR - to match the course with its instructor

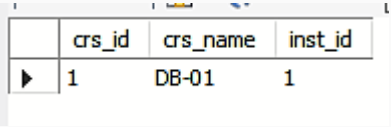
COURSE_INSTRUCTOR Relationship



COURSE_INSTRUCTOR Relationship Implementation



	inst_id	inst_fname	inst_lname	inst_address
▶	1	Haitham	El-Ghareeb	Mansoura, DK
	2	Osama	Aboel Nasr	Mansoura, DK
*	NULL	NULL	NULL	NULL



	crs_id	crs_name	inst_id
▶	1	DB-01	1

Controlled Redundancy

1. Relational model provides a minimum level of controlled redundancy
2. To eliminate most of the redundancies commonly found in file systems

Relational Diagram

1. Representation of
    1.1 relational DB entities
    1.2 attributes within those entities
    1.3 relationships between those entities

Relational Model

Table vs. File

Table vs. File

1. Relational DB Table resembles a file
2. Crucial Difference
   - 2.1 Table yields a complete data and structural independence
   - 2.2 Because table is purely logical structure
   - 2.3 How data is physically stored, no concern to the user or the designer

Relational Model

SQL

SQL

1. Powerful
2. Flexible
3. Query Language
4. Allows the user to specify what must be done without specifying how (Declarative Language)
5. RDBMS uses SQL to translate user queries into instructions

SQL-based Relational DB Application

Involves three parts:

1. end-user interface
2. collection of tables stored in the DB
3. SQL Engine

End-user Interface

1. allows end user to interact with the data
2. by automatically generating SQL code

Collection of Tables stored in DB

1. All data perceived to be stored in tables
2. Tables present the data to the end user in a way that is easy to understand
3. Each table is independent
4. Rows in different tables are related by common values in common attributes

SQL Engine

1. Hidden from end user
2. Executes all queries or data requests
3. Part of DBMS
4. Processes all user requests

Relational Model

Entity Relationship Model

Complex Design Requirements

1. Complex design activities require conceptual simplicity to yield successful results

ERM

1. Entity Relationship (ER)
2. Entity Relationship Model (ERM)
3. 1976 - Peter Chen
4. Graphical representation of entities and their relationships in database structure
5. Complemented relational data model concepts

ERD

1. Entity Relationship Diagram
2. ERM normally presented in ERD
3. use graphical representations to model DB components

ERM Components

1. Entity
2. Attributes
3. Relationships

Entity - 01

1. defined earlier
2. Anything about which data will be collected and stored
3. Represented by a Rectangle (entity box)
4. Entity name is written in capital letters and in singular form
5. Entity name is written in the center of the box
6. Usually, mapped to a relational table

Entity - 02

1. Each row is known as an entity instance / entity occurrence
2. Collection of like entities is known as: entity set
3. Collection of two instructors (entities) in the INSTRUCTOR entity set
4. ERD depicts entity sets

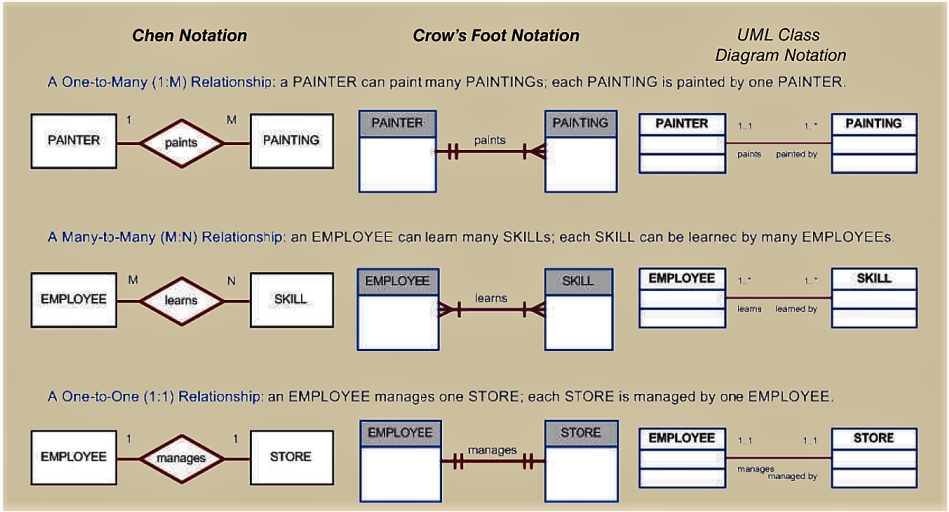
Attributes

1. Each entity consists of a set of attributes
2. Describes particular characteristics of the entity

Relationships

1. Describe associations among data
2. Most relationships describe associations between two entities
3. ERM uses the term **connectivity** to label the relationship types
4. Name is usually: passive or active verb

ERM


Chen Notation
Crow's Foot Notation
UML Class Diagram Notation
A One-to-Many (1:M) Relationship: a PAINTER can paint many PAINTINGs; each PAINTING is painted by one PAINTER.
PAINTER
1
paints
M
PAINTING
PAINTER
paints
PAINTING
PAINTER
1..1
1..*
paints
painted by
PAINTING
A Many-to-Many (M:N) Relationship: an EMPLOYEE can learn many SKILLs; each SKILL can be learnec by many EMPLOYEEs.
EMPLOYEE
M
learns
N
SKILL
EMPLOYEE
learns
SKILL
EMPLOYEE
1..*
1..*
learns
learned by
SKILL
A One-to-One (1:1) Relationship: an EMPLOYEE manages one STORE; each STORE is managed by one EMPLOYEE.
EMPLOYEE
1
manages
1
STORE
EMPLOYEE
manages
STORE
EMPLOYEE
1..1
1..1
manages
managed by
STORE

Note

1. M:N relationships exist at a conceptual level
2. We should recognize them
3. They are removed at the Logical Design and Physical implementation (Later)
4. M:N relationships are not appropriate in a relational model

Relational Model

Definitions

Relational Model

1. Developed by E. F. Codd of IBM in 1970
2. The relational model is based on mathematical set theory
3. represents data as independent relations
4. Each relation (table) is conceptually represented as a two-dimensional structure of intersecting rows and columns
5. The relations are related to each other through the sharing of common entity characteristics (values in columns)

Table (Relation)

1. A logical construct perceived to be a two dimensional structure composed of intersecting rows (entities) and columns (attributes) that represents an entity set in the relational model

Tuple

1. In the relational model, a table row

RDBMS

1. Relational Database Management System
2. A collection of programs that manages a relational database
3. The RDBMS software translates a user's logical requests (queries) into commands that physically locate and retrieve the requested data

Relational Diagram

1. A graphical representation of a relational database's entities, the attributes within those entities, and the relationships among the entities

Relational Table

1. Stores a collection of related entities

ERM

1. Entity Relationship Model
2. Data model that describes relationships (1:1, 1:M, M:N) among entities at the conceptual level with the help of ER Diagrams

ERD

1. Entity Relationship Diagram
2. Diagram that depicts an entity relationship model's entities, attributes, and relations

Entity Instance

1. Entity Occurrence
2. A row in a relational table

Entity Set

1. Collection of like entities

Connectivity

1. The type of relationship between entities
2. Classification include: 1:1, 1:M, M:N

Chen Notation

1. Peter Chen
2. Original Notation
3. Connectivity are written next to each entity box
4. Relationships are represented by a diamond connected to the related entities through a relationship line
5. Relationship name is written inside the diamond

Crow's Foot Notation

1. Representation of the ERD that uses a three-pronged symbol to represent the 'many' sides of the relationship
2. Connectivity are represented by symbol
3. Relationship name is written above the line

Class Diagram Notation

1. Set of symbols used in the creation of Class Diagram
2. Part of Unified Modeling Language (UML)
3. Lines with symbols
4. Use names in both sides of the relationship

Object-Oriented Model

OODM

1. Object-Oriented Data Model
2. Closely represent the real world
3. Both data and its relationships are contained in a single structure known as **object**
4. Basis for the OODBMS (Object-Oriented Database Management System)
5. OODM is said to be a **Semantic Data Model**
6. Basic building block: Object

Object

1. Self contained
2. Contains
    - operational procedures - all operations that can be performed on it
        - changing its data values
        - finding a specific data value
        - printing data values
    - data
    - various types of relationships

Object-Oriented Model

Definitions

OODM

1. Object-Oriented Data Model
2. Data model whose basic modeling structure is an object

Object

1. An abstract representation of a real-world entity that has
    - a unique identity
    - embedded properties
    - ability to interact with other objects and itself

OODBMS

1. Object-Oriented Database Management System
2. Data management software
3. Used to manage data in an OODM

Semantic Data Model

1. The first of a series of data models that models both data and their relationships in a single structure known as an object

Class

1. A collection of similar objects with shared structure (attributes) and behavior (methods)
2. Class encapsulates an object's data representation and a method's implementation

Method

1. In OODM
2. A named set of instructions to perform an action
3. Methods represent real world actions

Class Hierarchy

1. The organization of classes in a hierarchical tree in which each parent class is a *superclass* and each child class is a *subclass*

Inheritance

1. In OODM
2. Ability of an object to inherit the data structure and methods of the classes above it in the class hierarchy

UML

1. Unified Modeling Language
2. A language based on object-oriented concepts that provides tools such as diagrams and symbols to graphically model a system

Class Diagram

1. Diagram used to represent data and their relationships in UML object notation

ERDM

1. Extended Relational Data Model
2. Model that includes the object-oriented model's best features in an inherently simpler relational DB structural environment

O/R DBMS

1. Object/Relational Data Base Management System
2. DBMS based on the ERDM
3. ERDM championed by many relational DB researchers
4. Constitutes the relational model's response to OODM
5. This model includes many of the object-oriented model's beset features within an inherently simpler relational DB structure

XML

1. Extensible Markup Language
2. Metalanguage used to represent and manipulate data elements
3. Unlike other markup languages, XML permits the manipulation of a document's data elements
4. XML facilitates the exchange of structured documents such as students and courses over the internet

Big Data and NoSQL

Big Data and NoSQL

Big Data

Definition

1. Movement to find new and better ways to manage large amounts of web and sensor generated data and derive business insight from it, while simultaneously providing high performance and scalability at a reasonable cost

Term

First used the term

1. John Mashey
2. Scientist, Silicon Graphics
3. 1990s

Characteristics

1. Douglas Laney
2. Data Analyst, Gartner Group
3. Three Vs
   3.1 Volume
   3.2 Velocity
   3.3 Variety

Volume

1. refers to the amounts of data being stored
2. Internet, social media, sensors
3. Millions of e-transactions were being stored daily
4. Even small amounts of data from sensors, with millions of sensors and continuous data = Big Data

Velocity

1. Not only the speed with which data grows
2. also the need to process this data quickly in order to generate information and insight
3. Business Response Time

Variety

1. Data collected comes in multiple different data formats
2. Great portion of data comes in formats not suitable to be handled by the typical operational DB based on Relational Model

Third Platform and e-Learning Ecosystem

1. EMC Knowledge Sharing 2015 [3]
2. Download Paper [4]
3. EMC Proven Professional Spotlight [5]

[3] https://education.emc.com/content/emc/en-us/home/certification-overview/knowledge-sharing-archive.html#KSA2015

[4] https://education.emc.com/content/dam/dell-emc/documents/en-us/2015KS_El-Ghareeb-Third_Platform_and_e-Learning_Ecosystem.pdf

[5] https://www.dell.com/community/Education-Services/EMC-Proven-Professional-Spotlight-Haitham-A-El-Ghareeb/td-p/7037425

Big Data and NoSQL

Relational Model Problems with Big Data

Relational Model Problems with Big Data - 01

1. Relational approach does not always match the needs of organizations with Big Data challenges

Relational Model Problems with Big Data - 02

2. It is not always possible to fit unstructured, social media and sensor-generated data into the conventional relational structure of rows and columns

Relational Model Problems with Big Data - 03

3. Adding millions of rows of multi-format (structured and non-structured) data on a daily basis will inevitably lead to the need for more storage, processing power, and sophisticated data analysis tools that may not be available in the relational environment

Relational Model Problems with Big Data - 04

4. Data Analysis based on OLAP tools has proven to be very successful in relational environments with highly structured data. However, mining for usable data in the vast amounts of unstructured data collected from web sources require different approach

Big Data and NoSQL

Big Data Technologies

Big Data Technologies

1. Hadoop
2. MapReduce
3. NoSQL

Big Data and NoSQL

Hadoop

Hadoop

1. A Java-based open-source, high-speed, fault-tolerant distributed storage and computational framework
2. Uses low-cost hardware to create clusters of thousands of computer nodes to store and process data

HDFS

1. Hadoop Distributed File System
2. Highly distributed, fault-tolerant file storage system designed to manage large amounts of data at high speeds
3. Three node types:
   3.1 Name Node
   3.2 Data Node
   3.3 Client Node

Name Node

1. One of three types of nodes used in HDFS
2. Stores all the metadata about the file system

Data Node

1. One of three types of nodes used in HDFS
2. Stores fixed-size data blocks (that could be replicated to other data nodes)

Client Node

1. One of three types of nodes used in HDFS
2. Acts as the interface between the user application and HDFS

Big Data and NoSQL

MapReduce

MapReduce

1. Open Source API (Application Programming Interface)
2. Provides fast data analytic services
3. Distributes the processing of the data among thousands of nodes in parallel
4. Works with structured and unstructured data

MapReduce Functions

1. Map
    - Takes a job and divides it into smaller units of work
2. Reduce
    - Collects all the output results generated from the nodes and integrates them into a single result set

Big Data and NoSQL

NoSQL

NoSQL

1. Large Scale Distributed DB System
2. Stores structured and unstructured data in efficient ways

Relational DB vs. NoSQL

1. Relational DBs remain the preferred and dominant DBs to support most day-to-day transactions and structured data needs
2. Each DBMS technology has its areas of application
3. Best approach is to use the best tool for the job
4. Object / Relational DBs serve 98% of **operational** market need

NoSQL Characteristics

1. Not based on the relational model and SQL
2. Support highly distributed DB architectures
3. Provide high scalability, high availability, and fault tolerance
4. Support very large amounts of sparse data (data with large number of attributes but where the actual number of data instances is low)
5. Geared toward performance rather than transaction consistency

Most Common Approaches

1. Key-Value Stores
2. Document DB
3. Column DB
4. Graph DB

Big Data and NoSQL

Definitions

Big Data

1. A movement to find new and better ways to manage large amounts of web-generated data and derive business insight from it, while simultaneously providing high performance and scalability at a reasonable cost

3 Vs

1. Three basic characteristics of Big Data DB: Volume, Velocity, and Variety

Hadoop

1. A Java-based open-source, high-speed, fault-tolerant distributed storage and computational framework
2. Uses low-cost hardware to create clusters of thousands of computer nodes to store and process data

HDFS

1. Hadoop Distributed File System
2. Highly distributed, fault-tolerant file storage system designed to manage large amounts of data at high speeds
3. Three node types:
   3.1 Name Node
   3.2 Data Node
   3.3 Client Node

Name Node

1. One of three types of nodes used in HDFS
2. Stores all the metadata about the file system

Data Node

1. One of three types of nodes used in HDFS
2. Stores fixed-size data blocks (that could be replicated to other data nodes)

Client Node

1. One of three types of nodes used in HDFS
2. Acts as the interface between the user application and HDFS

MapReduce

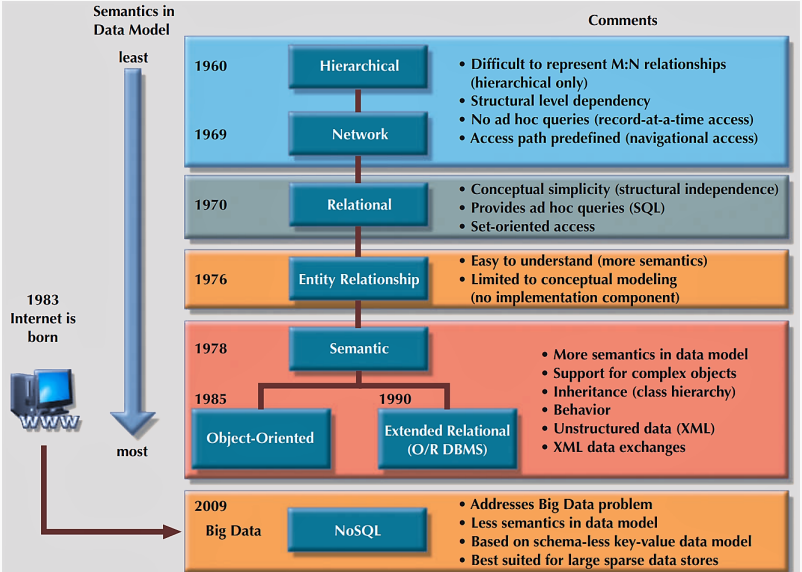
1. Open Source API (Application Programming Interface)
2. Provides fast data analytic services
3. One of the main Big Data technologies
4. Allows organizations to process massive data stores

NoSQL

1. New generation of DBMS that is not based on the traditional relational DB Model

Data Models Summary

Data Models Evolution - Historically



Data Models Evolution with Examples

GENERATION	TIME	DATA MODEL	EXAMPLES	COMMENTS
First	1960s–1970s	File system	VMS/VSAM	Used mainly on IBM mainframe systems Managed records, not relationships
Second	1970s	Hierarchical and network	IMS, ADABAS, IDS-II	Early database systems Navigational access
Third	Mid-1970s	Relational	DB2 Oracle MS SQL Server MySQL	Conceptual simplicity Entity relationship (ER) modeling and support for relational data modeling
Fourth	Mid-1980s	Object-oriented Object/relational (O/R)	Versant Objectivity/DB DB2 UDB Oracle 12c	Object/relational supports object data types Star Schema support for data warehousing Web databases become common
Fifth	Mid-1990s	XML Hybrid DBMS	dbXML Tamino DB2 UDB Oracle 12c MS SQL Server	Unstructured data support O/R model supports XML documents Hybrid DBMS adds object front end to relational databases Support large databases (terabyte size)
Emerging Models: NoSQL	Early 2000s to present	Key-value store Column store	SimpleDB (Amazon) BigTable (Google) Cassandra (Apache) MongoDB Riak	Distributed, highly scalable High performance, fault tolerant Very large storage (petabytes) Suited for sparse data Proprietary application programming interface (API)

Advantages and Disadvantages of Various DB Models

DATA MODEL	DATA INDEPENDENCE	STRUCTURAL INDEPENDENCE	ADVANTAGES	DISADVANTAGES
Hierarchical	Yes	No	1. It promotes data sharing. 2. Parent/child relationship promotes conceptual simplicity. 3. Database security is provided and enforced by DBMS. 4. Parent/child relationship promotes data integrity. 5. It is efficient with 1:M relationships.	1. Complex implementation requires knowledge of physical data storage characteristics. 2. Navigational system yields complex application development, management, and use; requires knowledge of hierarchical path. 3. Changes in structure require changes in all application programs. 4. There are implementation limitations (no multiparent or M:N relationships). 5. There is no data definition or data manipulation language in the DBMS. 6. There is a lack of standards.
Network	Yes	No	1. Conceptual simplicity is at least equal to that of the hierarchical model. 2. It handles more relationship types, such as M:N and multiparent. 3. Data access is more flexible than in hierarchical and file system models. 4. Data owner/member relationship promotes data integrity. 5. There is conformance to standards. 6. It includes data definition language (DDL) and data manipulation language (DML) in DBMS.	1. System complexity limits efficiency—still a navigational system. 2. Navigational system yields complex implementation, application development, and management. 3. Structural changes require changes in all application programs.
Relational	Yes	Yes	1. Structural independence is promoted by the use of independent tables. Changes in a table's structure do not affect data access or application programs. 2. Tabular view substantially improves conceptual simplicity, thereby promoting easier database design, implementation, management, and use. 3. Ad hoc query capability is based on SQL. 4. Powerful RDBMS isolates the end user from physical-level details and improves implementation and management simplicity.	1. The RDBMS requires substantial hardware and system software overhead. 2. Conceptual simplicity gives relatively untrained people the tools to use a good system poorly, and if unchecked, it may produce the same data anomalies found in file systems. 3. It may promote islands of information problems as individuals and departments can easily develop their own applications.
Entity relationship	Yes	Yes	1. Visual modeling yields exceptional conceptual simplicity. 2. Visual representation makes it an effective communication tool. 3. It is integrated with the dominant relational model.	1. There is limited constraint representation. 2. There is limited relationship representation. 3. There is no data manipulation language. 4. Loss of information content occurs when attributes are removed from entities to avoid crowded displays. (This limitation has been addressed in subsequent graphical versions.)
Object-oriented	Yes	Yes	1. Semantic content is added. 2. Visual representation includes semantic content. 3. Inheritance promotes data integrity.	1. Slow development of standards caused vendors to supply their own enhancements, thus eliminating a widely accepted standard. 2. It is a complex navigational system. 3. There is a steep learning curve. 4. High system overhead slows transactions.
NoSQL	Yes	Yes	1. High scalability, availability, and fault tolerance are provided. 2. It uses low-cost commodity hardware. 3. It supports Big Data. 4. Key-value model improves storage efficiency.	1. Complex programming is required. 2. There is no relationship support—only by application code. 3. There is no transaction integrity support. 4. In terms of data consistency, it provides an eventually consistent model.

Degrees of Data Abstraction

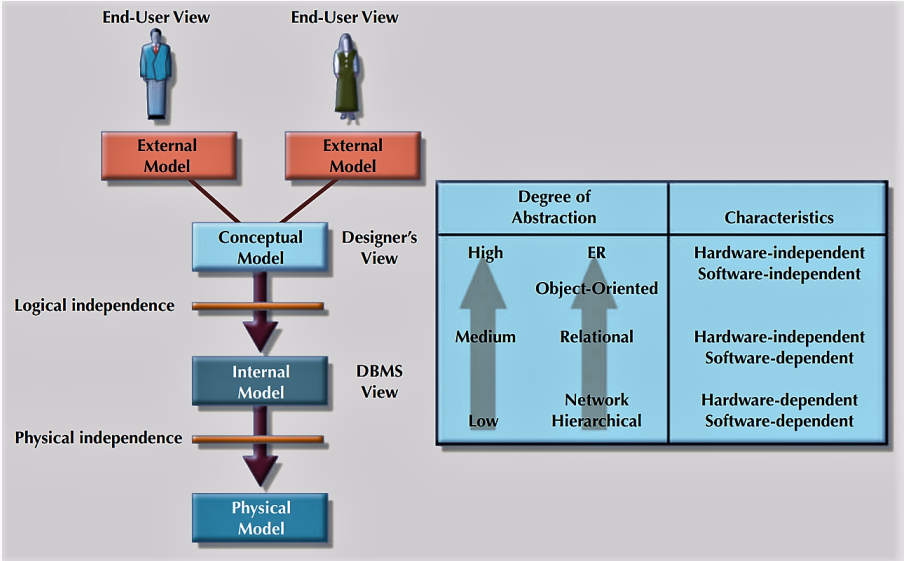
Degrees of Data Abstraction

ANSI/SPARC

ANSI/SPARC Architecture

1. American National Standards Institute (ANSI)
2. Standards Planning and Requirements Committee (SPARC)
3. Defined a framework for data modeling based on degrees of data abstraction
4. Three levels of data abstraction
    4.1 external
    4.2 conceptual
    4.3 internal

Data Abstraction Levels



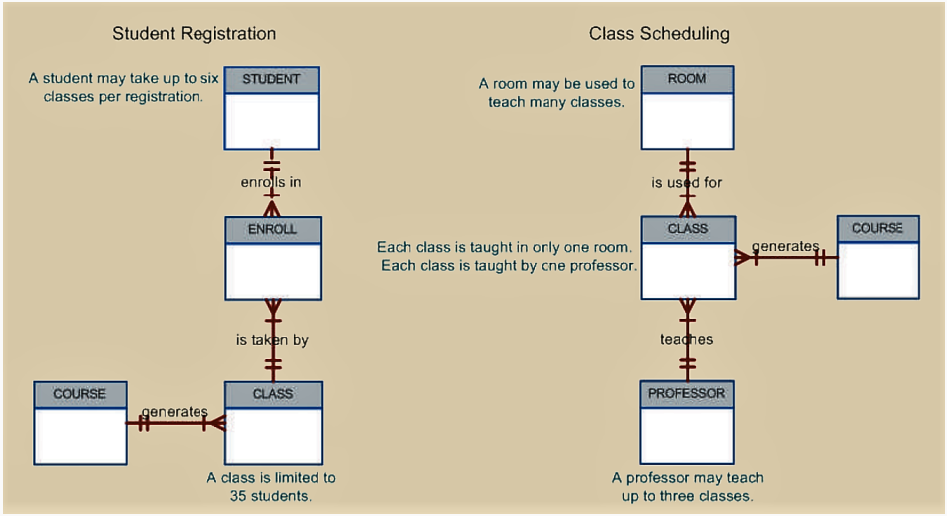
Degrees of Data Abstraction

External Model

External Model

1. end users' view of the data environment
2. end user = people who use the application programs to manipulate the data and generate information
3. Companies, usually are divided into business units: sales, marketing, etc.
4. Each business unit is subject to specific constraints and requirements
5. Each business unit uses a subset of the overall data
6. ERD is used to represent external view
7. Specific representation of external view is known as: external schema

External Schema Example



External Schema Advantages

1. It is easy to identify specific data required to support each business unit's operations
2. It makes the designer's job easy by providing feedback about the model's adequacy. Specifically, the model can be checked to ensure that it supports all processes as defined by their external models, as well as all operational requirements and constraints
3. It helps to ensure security constraints in the database design. Damaging an entire database is more difficult when each business unit works with only a subset of data
4. It makes application program development much simpler

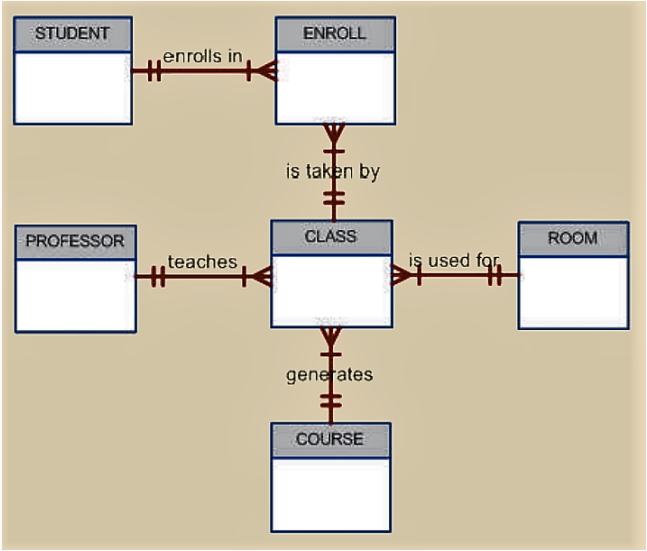
Degrees of Data Abstraction

Conceptual Model

Conceptual Model

1. Represents a global view of the entire DB by the entire organization
2. Integrates all external views (entities, relationships, constraints, and processes) into a single global view of the data
3. aka Conceptual Schema
4. It is the basis for the identification and high-level description of the main data objects (avoiding any DB model-specific details)
5. ERD is used to graphically represent conceptual schema

Example


STUDENT
enrolls in
ENROLL
is taken by
PROFESSOR
teaches
CLASS
is used for
ROOM
generates
COURSE

Advantages

1. Provides a bird's eye (macro level) view of the data environment that is relatively easy to understand
2. Independent of both software and hardware
   - Software Independence: model does not depend on the DBMS software used to implement the model
   - Hardware Independence: model does not depend on the hardware used in the implementation of the model
3. Changes in either software or hardware will have no effect on DB design at the conceptual level
4. **Logical Design** refers to the task of creating a conceptual data model that could be implemented in any DBMS

Degrees of Data Abstraction

Internal Model

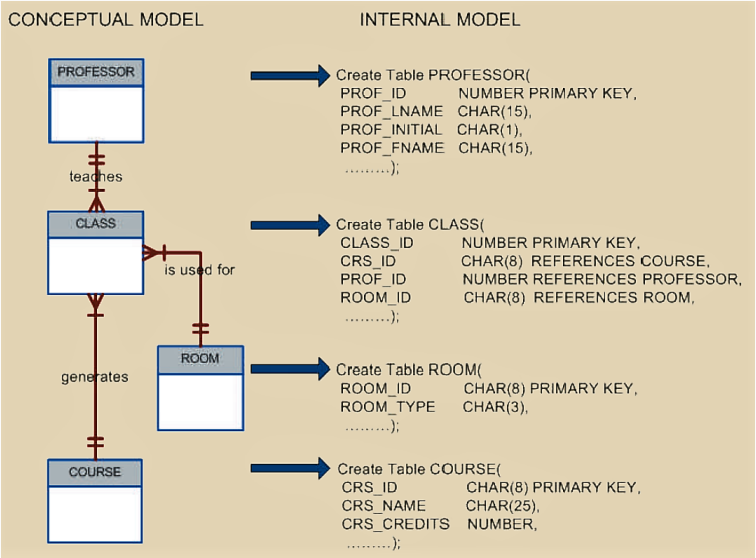
Internal Model

1. Once a specific DBMS has been selected, the internal model maps the conceptual model to the DBMS
2. is the representation of DB as 'seen' by the DBMS
3. Requires the designer to match the conceptual model's characteristics and constraints to those of the selected implementation model
4. **Internal Schema** depicts a specific representation of an internal model, using the DB constructs supported by the chosen DB

Example

1. Because we focus on relational model, a relational DB was chosen to implement the internal model
2. Internal schema should map the conceptual model to the relational model constructs
3. Entities are mapped to tables
4. Internal schema is expressed using SQL

Example


CONCEPTUAL MODEL
INTERNAL MODEL
PROFESSOR
teaches
CLASS
is used for
ROOM
generates
COURSE
Create Table PROFESSOR(
PROF_ID NUMBER PRIMARY KEY,
PROF_LNAME CHAR(15),
PROF_INITIAL CHAR(1),
PROF_FNAME CHAR(15),
.....);
Create Table CLASS(
CLASS_ID NUMBER PRIMARY KEY,
CRS_ID CHAR(8) REFERENCES COURSE,
PROF_ID NUMBER REFERENCES PROFESSOR,
ROOM_ID CHAR(8) REFERENCES ROOM,
.....);
Create Table ROOM(
ROOM_ID CHAR(8) PRIMARY KEY,
ROOM_TYPE CHAR(3),
.....);
Create Table COURSE(
CRS_ID CHAR(8) PRIMARY KEY,
CRS_NAME CHAR(25),
CRS_CREDITS NUMBER,
.....);

Characteristics

1. Software Dependent
2. Change in the DBMS requires the internal model be changed
3. **Logical Independence** when changing the internal model without affecting conceptual model
4. Hardware Independent: unaffected by the type of computer on which the software is installed
5. Change in storage devices or even a change in operating systems will not affect the internal model

Degrees of Data Abstraction

Physical Model

Physical Model

1. Operates at the lowest level of abstraction
2. Describing the way data is saved on storage media such as magnetic, solid state, or optical media
3. Requires the definition of both the physical storage devices and the physical access methods required to reach the data within those storage devices, making it both software and hardware dependent
4. **Physical Independence** Changing the physical model without affecting the internal model
5. Example: change in storage devices or methods and even a change in Operating System will not affect the internal model

Degrees of Data Abstraction

Data Abstraction Levels Summary

Data Abstraction Levels Summary

MODEL	DEGREE OF ABSTRACTION	FOCUS	INDEPENDENT OF
External	High	End-user views	Hardware and software
Conceptual		Global view of data (database model independent)	Hardware and software
Internal		Specific database model	Hardware
Physical	Low	Storage and access methods	Neither hardware nor software

Degrees of Data Abstraction

Definitions

ANSI

1. American National Standards Institute
2. The group that accepted the DBTG recommendations and augmented database standards in 1975 through its SPARC committee

External Model

1. The application programmer's view of the data environment
2. Given its business focus, an external model works with a data subset of the global database schema

External Schema

1. The specific representation of an external view; the end user's view of the data environment

Conceptual Model

1. The output of the conceptual design process
2. The conceptual model provides a global view of an entire database and describes the main data objects, avoiding details

Conceptual Schema

1. A representation of the conceptual model, usually expressed graphically

Software Independence

1. A property of any model or application that does not depend on the software used to implement it

Hardware Independence

1. A condition in which a model does not depend on the hardware used in the model's implementation
2. Therefore, changes in the hardware will have no effect on the database design at the conceptual level

Logical Design

1. A stage in the design phase that matches the conceptual design to the requirements of the selected DBMS and is therefore software dependent
2. Logical design is used to translate the conceptual design into the internal model for a selected database management system, such as DB2, SQL Server, Oracle, IMS, Informix, Access, or Ingress

Internal Model

1. In database modeling, a level of data abstraction that adapts the conceptual model to a specific DBMS model for implementation
2. The internal model is the representation of a database as "seen" by the DBMS
3. In other words, the internal model requires a designer to match the conceptual model's characteristics and constraints to those of the selected implementation model

Internal Schema

1. A representation of an internal model using the database constructs supported by the chosen database

Logical Independence

1. A condition in which the internal model can be changed without affecting the conceptual model
2. The internal model is hardware independent because it is unaffected by the computer on which the software is installed
3. Therefore, a change in storage devices or operating systems will not affect the internal model

Physical Model

1. A model in which physical characteristics such as location, path, and format are described for the data
2. The physical model is both hardware and software dependent

Physical Independence

1. A condition in which the physical model can be changed without affecting the internal mode

Summary

Summary

1. A data model is an abstraction of a complex real-world data environment
2. Database designers use data models to communicate with programmers and end users
3. The basic data-modeling components are entities, attributes, relationships, and constraints
4. Business rules are used to identify and define the basic modeling components within a specific real-world environment

Summary

5. The hierarchical and network data models were early models that are no longer used, but some of the concepts are found in current data models

Summary

6. The relational model is the current database implementation standard
7. In the relational model, the end user perceives the data as being stored in tables
8. Tables are related to each other by means of common values in common attributes
9. The entity relationship (ER) model is a popular graphical tool for data modeling that complements the relational model
10. The ER model allows database designers to visually present different views of the data—as seen by database designers, programmers, and end users—and to integrate the data into a common framework

Summary

11. The object-oriented data model (OODM) uses objects as the basic modeling structure
12. Like the relational model's entity, an object is described by its factual content
13. Unlike an entity, however, the object also includes information about relationships between the facts, as well as relationships with other objects, thus giving its data more meaning

Summary

14. The relational model has adopted many object-oriented (OO) extensions to become the extended relational data model (ERDM)
15. Object/relational database management systems (O/R DBMS) were developed to implement the ERDM
16. At this point, the OODM is largely used in specialized engineering and scientific applications, while the ERDM is primarily geared to business applications

Summary

17. Big Data technologies such as Hadoop, MapReduce, and NoSQL provide distributed, fault-tolerant, and cost-efficient support for Big Data analytics
18. NoSQL databases are a new generation of databases that do not use the relational model and are geared to support the very specific needs of Big Data organizations
19. NoSQL databases offer distributed data stores that provide high scalability, availability, and fault tolerance by sacrificing data consistency and shifting the burden of maintaining relationships and data integrity to the program code

Summary

20. Data-modeling requirements are a function of different data views (global versus local) and the level of data abstraction
21. The American National Standards Institute Standards Planning and Requirements Committee (ANSI/SPARC) describes three levels of data abstraction: external, conceptual, and internal
22. The fourth and lowest level of data abstraction, called the physical level, is concerned exclusively with physical storage methods

https://www.haitham.ws