UKDW
Universitas Kristen Duta Wacana
YOGYAKARTA

Obedience
Integrity
Excellence
Service

TI0173 – Sistem Basis Data

DATABASE BUILDING BLOCKS

Relational Database Model

What's wrong with this design?

last_name	first_name	phone	contribution	contribution2
Smith	Jane	4623598	Food preparation	Driving
Green	Rob	8965431	Transport	
Henry	James	9576342	Camping Gear	Cooking
Wang	Li	9612345	Cooking	

- Sebuah sekolah merencanakan untuk melakukan aktivitas di luar. Para guru meminta orang tua untuk turut membantu dan membuat desain database seperti pada gambar di atas.
- Masalah apa yang bisa terjadi dengan desain tersebut? Berikan solusi lain.

What's wrong with this design?

plantID	genus	species	common_name	use1	use2	use3
1	Dodonaea	viscosa	Akeake	shelter	hedging	soil stability
2	Cedrus	atlantica	Atlas cedar	shelter		
3	Alnus	glutinosa	Black alder	soil stability	shelter	firewood
4	Eucalyptus	nichollii	Black peppermint gum	shelter	coppicing	bird food
5	Juglans	nigra	Black walnut	timber		
6	Acacia	mearnsii	Black wattle	firewood	shelter	soil stability

- Terdapat sebuah database tentang informasi tanaman beserta manfaat dari masing-masing tanaman.
- Masalah apa yang bisa terjadi dengan desain tersebut? Berikan solusi lain.

Data Models

- Desain DB berfokus ke bagaimana struktur DB akan digunakan untuk menyimpan dan mengelola data end user.
- Pemodelan data adalah langkah pertama dalam mendesain DB.
- **Model data** itu sendiri adalah representasi yang sederhana, umumnya grafis, atau bisa disebut **abstraksi** dari struktur data nyata yang lebih kompleks.
- Fungsi model adalah membantu memahami kompleksitas dari lingkungan nyata.

Data Models

Dalam lingkungan DB, model data mewakili:

- struktur data
- karakteristik relasi
- konstrain
- transformasi
- dll.

dengan tujuan membantu memetakan suatu domain masalah tertentu.

Model Data yang Siap Diimplementasikan

Sebuah model data yang siap diimplementasikan musti memiliki komponen berikut:

1) Sebuah deskripsi dari struktur data yang akan menyimpan data end user.
2) Sebuah set dari aturan-aturan untuk menjamin integritas data.
3) Sebuah metodologi manipulasi data untuk mendukung transformasi data nyata.

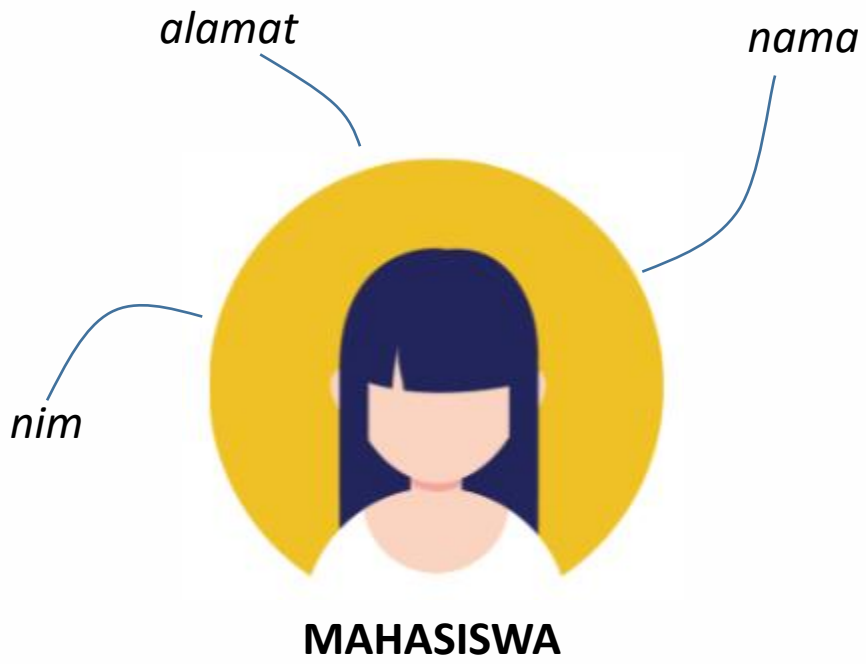
Pentingnya Model Data

- Model data dapat memfasilitasi interaksi antar desainer, programmer aplikasi, dan *end user*.
- Model data menghindarkan kesalahan dari cara melihat data yang berbeda-beda oleh setiap level user: manajer, pegawai harian, maupun programmer—mencegah fenomena "*blind people and the elephant*".
- Model data menghindarkan pengembangan DB yang tidak terarah, sehingga *budgeting* bisa diprediksikan lebih presisi.

Data Model Building Blocks

- **Entitas**—adalah semua (orang, tempat, hal, atau peristiwa) yang terkait dengan sistem yang perlu dicatat dan disimpan. Contoh: MAHASISWA, MATAKULIAH, INVOICE, BARANG
- **Atribut**—karakteristik dari entitas.
  Contoh: Nama mahasiswa, bobot SKS dari mata kuliah
- **Relationship**—menjelaskan asosiasi antar entitas dengan sifat bidirectional.
  Contoh: Satu dosen wali mengasuh beberapa mahasiswa, tetapi satu mahasiswa hanya diasuh oleh satu dosen wali saja selama kuliah.
  - One-to-many (1:M atau 1..*)
  - Many-to-many (M:N atau *..*)
  - One-to-one (1:1 atau 1..1)

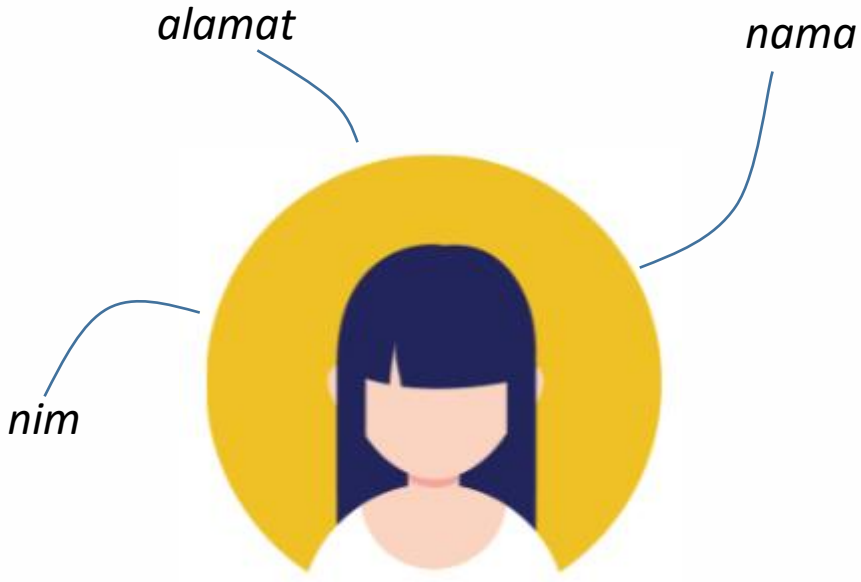
Data Model Building Blocks



Data Model Building Blocks

MATAKULIAH

Apa relasinya?



MAHASISWA

Data Model Building Blocks

- **Konstrain**—adalah batasan yang diterapkan terhadap data. Konstrain penting karena membantu menjamin integritas data.
  - IPK harus diisi antara 0,00 hingga 4,00 → lebih rinci ke tabel
  - Tiap mata kuliah harus ada dosen pengampunya → mempengaruhi lebih dari satu tabel

Lalu bagaimana kita mengidentifikasi manakah entitas, atribut, relationship dan konstrain?

Langkah pertama adalah mengidentifikasi dengan jelas **aturan bisnis** untuk domain masalah yang akan dimodelkan.

Aturan Bisnis

- Mengumpulkan sembarang data bukanlah pekerjaan yang sulit. Tapi tidak semua data benar-benar dipakai dalam bisnis.
- Pengumpulan data menjadi berarti hanya ketika data tersebut merefleksikan aturan bisnis yang telah didefinisi sebelumnya.
- **Aturan bisnis** adalah deskripsi yang jelas, presisi, dan non-ambigu terhadap suatu kebijakan, prosedur, atau prinsip yang ada dalam suatu organisasi.
- Aturan bisnis harus ditulis dan diperbarui untuk mewakili setiap perubahan dalam lingkup operasional organisasi.

Aturan Bisnis

- Aturan bisnis berasal dari manager perusahaan, pembuat kebijakan, dokumentasi tertulis tentang prosedur, standar, manual dari perusahaan tersebut.
- Aturan bisnis juga bisa didapatkan dengan observasi langsung dan interview (meski harus hati-hati karena persepsi setiap orang bisa berbeda!). *Verify end-user perceptions!*
- Beberapa aturan bisnis tidak bisa dimodelkan secara langsung pada database. Contoh: “tidak boleh ada pilot yang terbang selama lebih dari 10 jam dalam 1 hari”.

Kegunaan Aturan Bisnis

- Standardisasi pandangan terhadap data yang ada pada perusahaan
- Sebagai alat komunikasi antara desainer dan user
- Membantu desainer untuk memahami sifat, peran, dan lingkup data
- Membantu desainer untuk memahami proses bisnis
- Membantu desainer untuk mengembangkan relasi dan konstrain yang tepat sehingga data model lebih akurat

Contoh Aturan Bisnis

- Seorang pelanggan bisa melakukan lebih dari satu transaksi pada suatu toko, dan dari situ setiap transaksi mendapatkan satu invoice.
- Sebuah invoice hanya bisa dibuat/dihasilkan oleh satu pelanggan.
- Yang bisa mengambil matakuliah skripsi adalah mahasiswa yang telah mengambil SKS minimal 100.
- Mobil yang sedang disewa, tidak bisa disewa oleh orang lain.

Dari aturan bisnis di atas bisa diketahui mana entitas, atribut, relationship, dan konstrain.

Aturan Bisnis » Komponen Model Data

- [Trik] Ubah saja kata benda/*noun* yang disebut-sebut dalam aturan bisnis menjadi entitas, dan kata kerja/*verb* yang berasosiasi ke *noun* menjadi relationship antar entitas.
- Contoh: “Seorang mahasiswa bisa mengambil lebih dari satu mata kuliah” terdiri dari dua noun (**mahasiswa** dan **mata kuliah**) serta satu verb (**mengambil**) yang berasosiasi terhadap noun.
- Dari aturan di atas, kita bisa deduksikan bahwa:
  - Mahasiswa dan mata kuliah adalah *objects of interest* untuk lingkup ini dan perlu direpresentasikan dengan entitas yang tepat.
  - Ada hubungan “mengambil” antar mahasiswa dan mata kuliah.

Aturan Bisnis » Komponen Model Data

- Jangan lupa, *relationship* adalah *bidirectional*; yaitu dalam kasus ini: “Seorang mahasiswa bisa mengambil lebih dari satu mata kuliah” dapat dikomplemenkan “Sebuah mata kuliah bisa diambil oleh lebih dari satu mahasiswa”.
- Ini yang disebut relationship *many-to-many* (M:N)
- Bacanya tidak boleh seperti ini: “Banyak matakuliah diambil oleh banyak mahasiswa?”
- Bagaimana dengan contoh *one-to-many* (1:N)? Ingat lagi hubungan antara dosen wali dengan mahasiswa dan sebutkan pernyataan kebalikannya.

Aturan Bisnis » Komponen Model Data

- Aturan umum berikutnya, untuk bisa mengidentifikasi tipe relationship, tanyakan dua pertanyaan ini:
  1) Berapa instans B yang terhubung ke satu instans A?
  2) Berapa instans A yang terhubung ke satu instans B?
- Contoh:
  1) Berapa buku yang bisa dipinjam oleh seorang mahasiswa?
  2) Berapa banyak mahasiswa yang bisa meminjam satu buku tersebut?
- Dari contoh di atas relationship-nya adalah *many-to-many* (M:N).

Konvensi Penamaan

- Konvensi penamaan yang baik akan meningkatkan kemampuan model data untuk memfasilitasi komunikasi antar desainer, programmer aplikasi, dan end user.
- Nama entitas harus deskriptif terhadap obyek yang diwakilinya, serta menggunakan terminologi yang dikenal user.
- Bisa gunakan underscore atau camelCase sebagai pengganti spasi.
- Misalnya untuk tabel **`mahasiswa`**, memiliki atribut:

Underscore style	camelCase
mhs_nim	nimMahasiswa
mhs_nama	namaMahasiswa
mhs_tgllahir	tglLahirMahasiswa

EVOLUTION OF MAJOR DATA MODELS

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)

Introduction to Relational Model

- RDM diciptakan oleh E.F. Codd tahun 1970 berdasarkan logika predikat dan teori set matematika.
- Model data relasional memudahkan desainer untuk lebih fokus pada **representasi logika** dari data, daripada memikirkan bagaimana detail penyimpanan fisiknya.
- Bayangkan seperti motor transmisi otomatis.
- Dengan konsep tabel-tabel dan hubungannya, model data relasional jauh lebih mudah dipahami daripada model-model lainnya; seperti hirarkis atau network.

Tabel dan Karakteristiknya

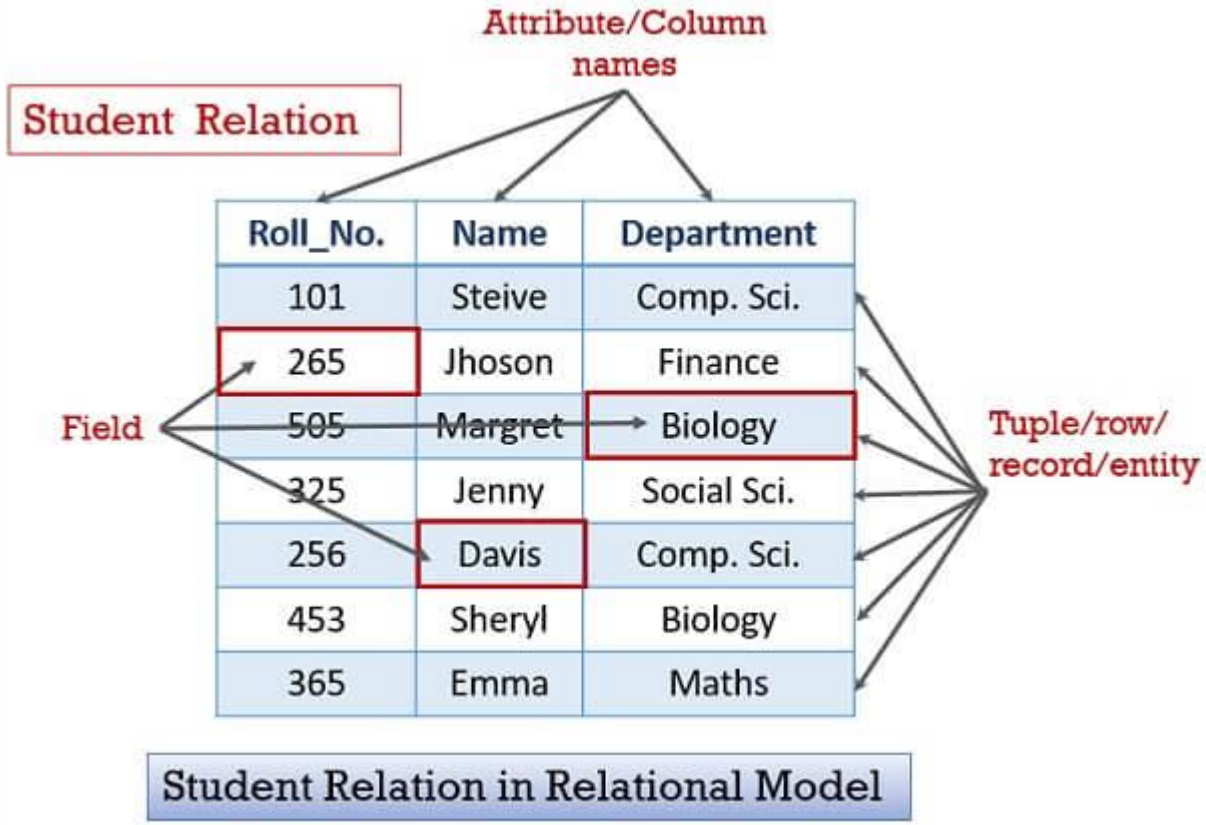
- Pandangan logika dari DB relasional dibantu dengan pembuatan hubungan antar data berdasarkan konstruksi logika yang disebut **relasi**.
- Oleh karena "relasi" adalah konstruksi matematis, dan sering terkacaukan dengan relasi antar tabel, maka *end user* lebih mudah melihat dan menyebutnya sebagai **tabel**.
- Tabel memiliki struktur dua dimensi yang terdiri dari **rows** dan **columns**.
- Sebuah tabel berisi sekumpulan kejadian dari entitas yang saling berelasi.
- Contoh: tabel `MAHASISWA` berisi sekumpulan instans dari entitas, masing-masing mewakili seorang mahasiswa.

Tabel dan Karakteristiknya

1) Tabel dipersepsikan sebagai struktur dua dimensi yang terdiri dari *rows* dan *columns.*
2) Tiap *row* **(tuple)** mewakili satu kejadian entitas di antara set entitas.
3) Tiap *column* mewakili satu **atribut**, dan tiap *column* memiliki nama yang berbeda.
4) Tiap irisan *row* dan *column* mewakili data bernilai tunggal.
5) Seluruh nilai dalam satu *column* harus memiliki format data yang sama
6) Tiap *column* memiliki batas nilai tertentu yang dikenal sebagai **attribute domain.**
7) Urutan dari *row* dan *column* tidak penting bagi DBMS (*)
8) Tiap tabel harus memiliki sebuah atribut atau sekumpulan atribut yang membedakan tiap *row* secara unik.

Roll_No.	Name	Department
101	Steive	Comp. Sci.
265	Jhoson	Finance
505	Margret	Biology
325	Jenny	Social Sci.
256	Davis	Comp. Sci.
453	Sheryl	Biology
365	Emma	Maths

Student Relation in Relational Model



Student Relation in Relational Model

Taken from https://binaryterms.com/relational-data-model.html

Catatan:

Beberapa vendor DBMS menggunakan penamaan yang bermacam-macam.

Aktivitas Kelas!