Introduction to Relational Databases
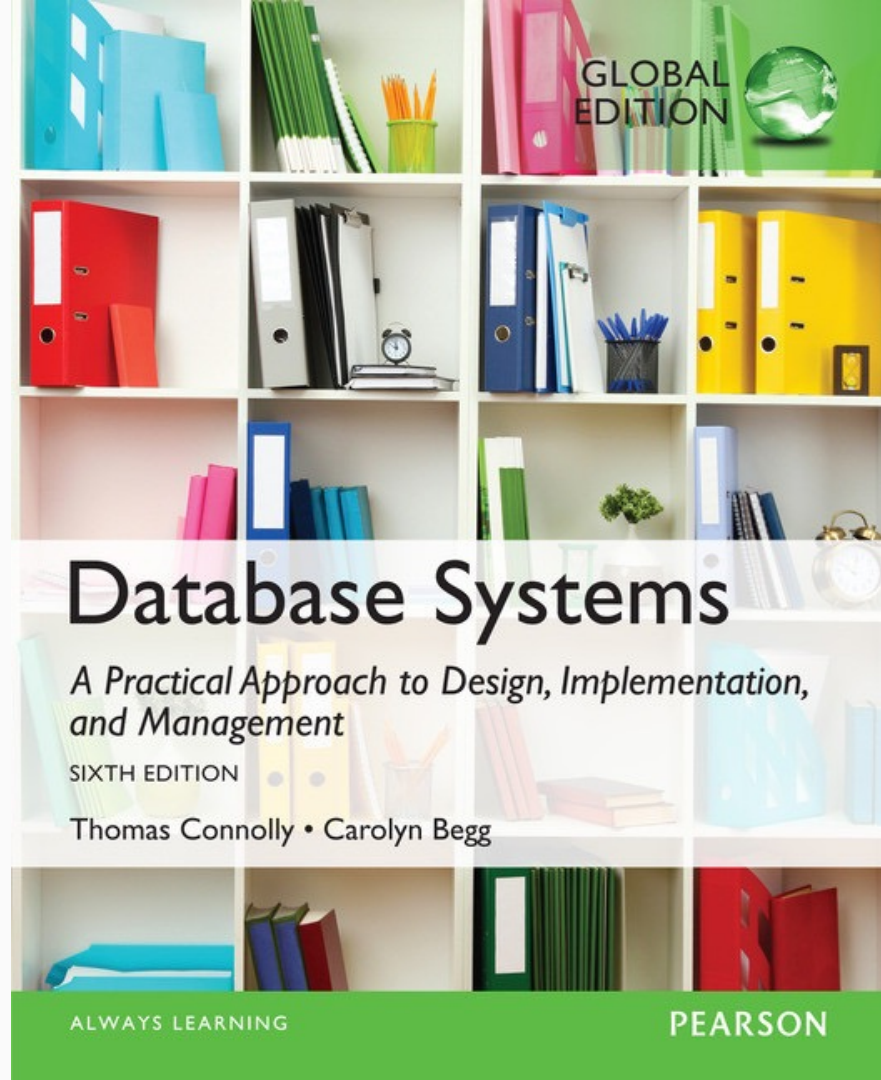
DBI - Databases and Interfaces
Dr Matthew Pike & Dr Yuan Yao

This Lecture

- Introduction to Databases
  - Basic features
  - Database Management Systems

- The Relational Model
  - Relational data structures
  - Candidate, Primary and Foreign Keys
  - Entity and Referential Integrity

The "DB Book"

- It is a very useful resource

- There will be numerous references to the book throughout the forthcoming lectures

GLOBAL EDITION

Database Systems
A Practical Approach to Design, Implementation, and Management

SIXTH EDITION

Thomas Connolly • Carolyn Begg

ALWAYS LEARNING

PEARSON

What is a Database?

What is a Database?

- "A collection of data arranged for ease and speed of search and retrieval."
  - American Heritage Science Dictionary
- "A structured set of data held in computer storage"
  - Oxford English Dictionary
- "One or more large structured sets of persistent data, usually associated with software to update and query the data"
  - Free Online Dictionary of Computing

Why Databases?

Why Study Databases?

- Core Topic
    - Fundamental Math
    - Practical Applications
- Project Work – GRP, FYP, etc
- Jobs
    - You *will* use databases in your future careers
    - You are expected to have a fundamental understanding

Databases are (virtually) everywhere!

- Library catalogues
- Medical records
- Bank accounts
- Stock market data
- Personnel systems
- Product catalogues
- Telephone directories

- Train timetables
- Airline bookings
- Credit card details
- Student records
- Customer histories
- Stock market prices
- Pretty much **every** website you ever use!
- Many, many more applications

The problem

We want to store our data in a format that allows us to easily query, update, insert and delete without affecting the integrity of the data.
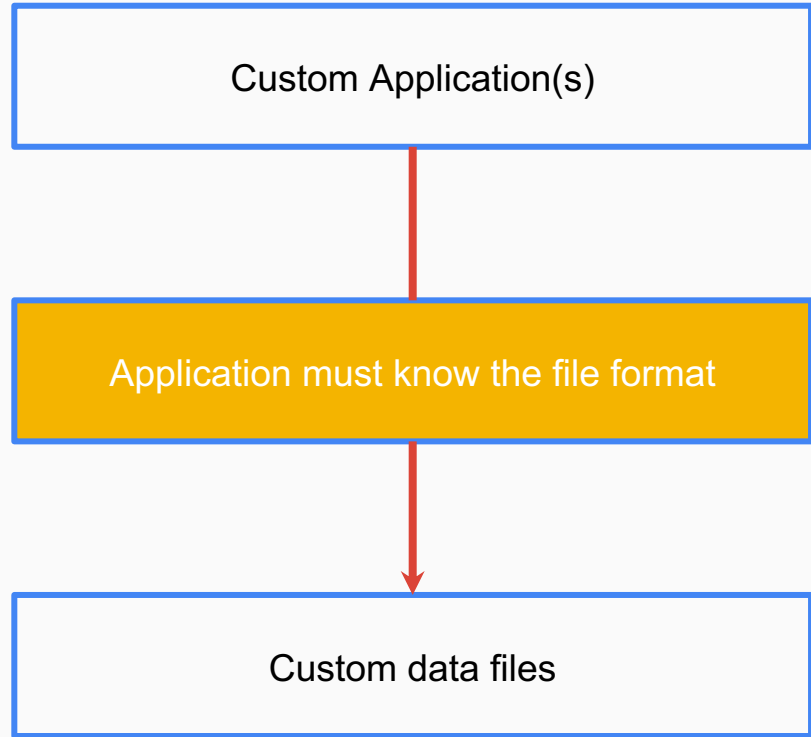
For Example - Student Records

- **Insert** - New Students
- **Update** - Students details change
- **Delete** - Student leaves
- **Query** - How many students study CS?

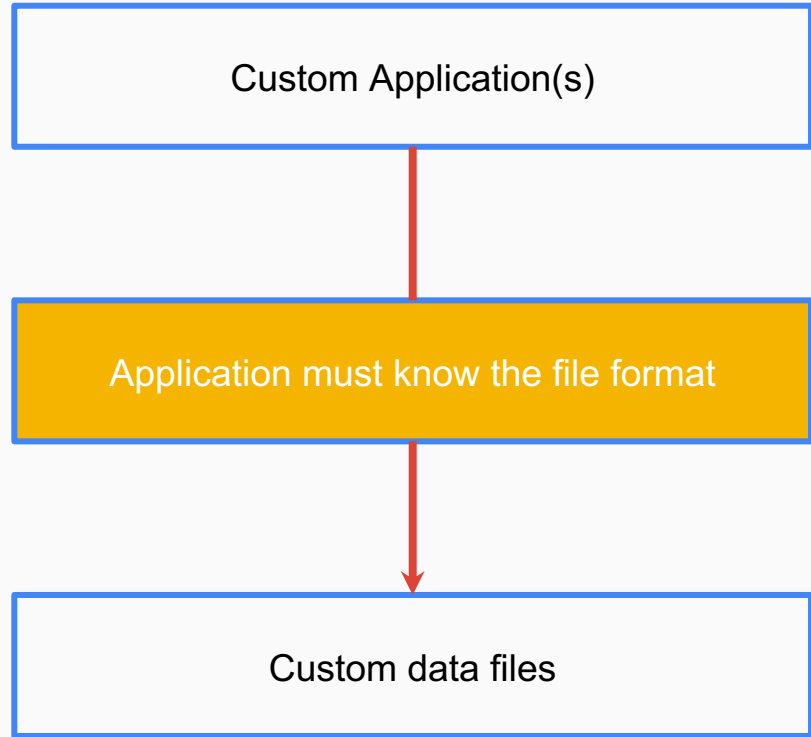https://thethesiswhisperer.files.wordpress.com/2010/09/pile-of-paper.jpg

The Early Days

- Applications store their data in files
- Each file has its own format
  - defined by the application itself
- Program has to know format
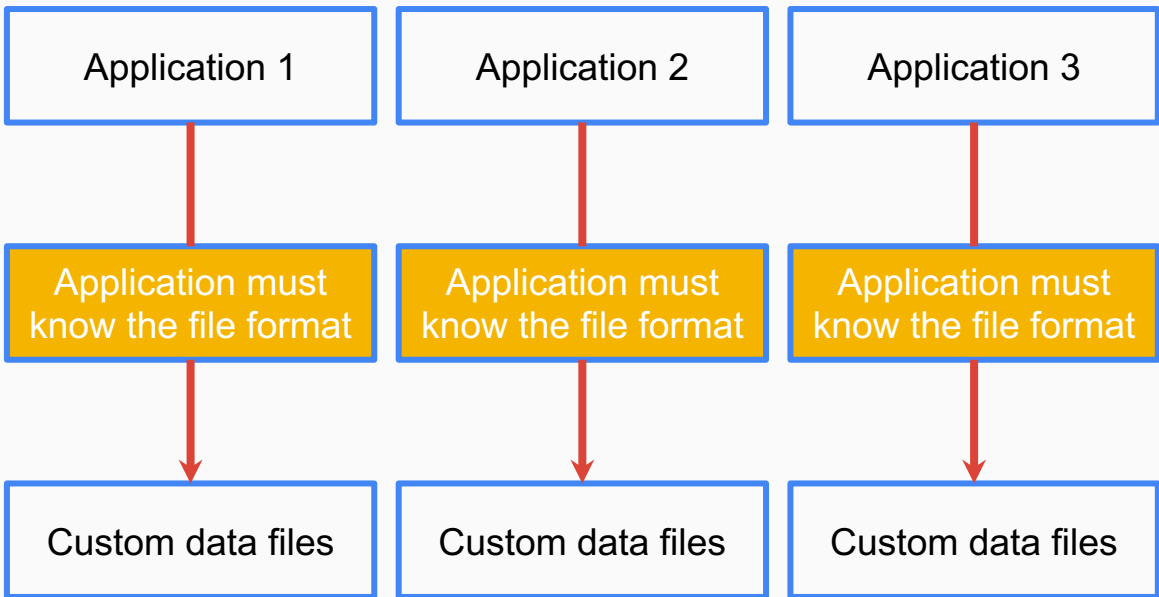- Any other program using the file has to know format

Custom Application(s)

Application must know the file format

Custom data files

Any problems?

Custom Application(s)

Application must know the file format

Custom data files

The Early Days: Multiple Applications

- No standards
- Incompatible file formats
- Data duplication
- Data dependence
- Fixed queries
- Concurrency
- Security
- No theoretical foundations
- …

Application 1	Application 2	Application 3
Application must know the file format	Application must know the file format	Application must know the file format
Custom data files	Custom data files	Custom data files

A better approach: Database Management System (DBMS)

- A database management system (DBMS) is system software for creating and managing databases.
- Data Definition Language(DDL)
- Data Manipulation Language (DML)
- Data Control Language(DCL)
- Data Query Language(DQL)
- Structured Query Language (SQL): data definition, manipulation and query.

Application 1	Application 2	Application 3

Application must know the DBMS Interface

Database Management System (DBMS)

DB data file(s)
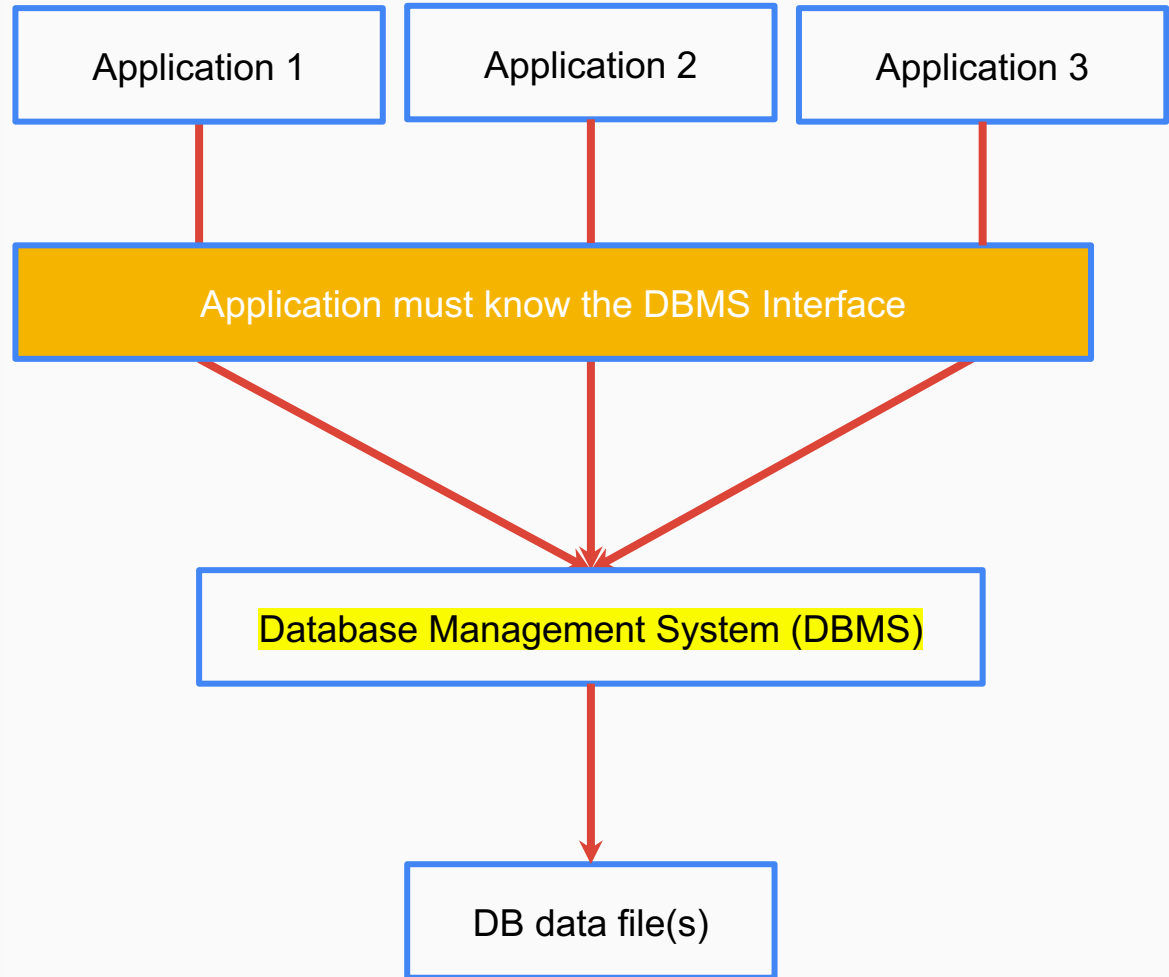
DBMS Functions / Must Haves

- Allow users to store, retrieve and update data
- Ensure either that all the updates corresponding to a given action are made or that none of them is made
- Ensure that DB is updated correctly when multiple users are updating it concurrently
- Recover the DB in the event it is damaged in any way
- Ensure that only authorised users can access the DB
- Be capable of integrating with other software

Examples of Modern DBMS

- DBMS is the software that implements the database and its functional operations
- Examples:
  - SQLite
  - Oracle
  - DB2
  - MySQL
  - PostgreSQL
  - Microsoft SQL Server
  - MS Access

ANSI/SPARC Model

ANSI/ SPARC Architecture

- Proposed a framework for DBMS in 1975

  - American National Standards Institute

  - Standards Planning Requirements Committee

- Three tier/level architecture

  - External level - for database users

  - Conceptual level - for database designers

  - Internal level - for systems designers
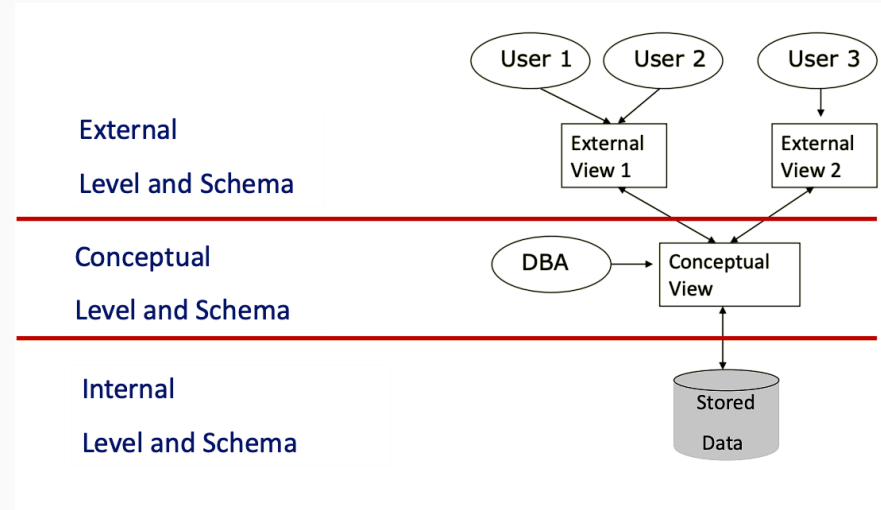
Layers

- Internal Level

  - Defines how data is stored in the database (storage space allocation, data structures, indexing, data compression, encryption etc.)

  - Used by System designers.

- Conceptual (Middle) Level

  - Defines what data is stored in the database and the relationships between data (e.g., table definitions, constraints on the data, security and integrity information)

  - Deals with the organisation of the entire database content

  - Database Administrator.

Layers

- External

  - Defines the user's view of the database (the part of the database that is relevant to each user)

    - Based on the user's interests,

      - E.g., YY/MM/DD, DD/MM/YYYY

    - Used by users and applications programmers

- Mappings translate between layers
- Provide data independence

Logical data independence
Conceptual changes should not affect external views

Physical data independence
Changes to internal structure should not affect conceptual view

External Schema

Conceptual Schema

Internal Schema

User 1   User 2   User 3

External View 1   External View 2

Mappings

DBA   Conceptual View

Mapping

Stored Data
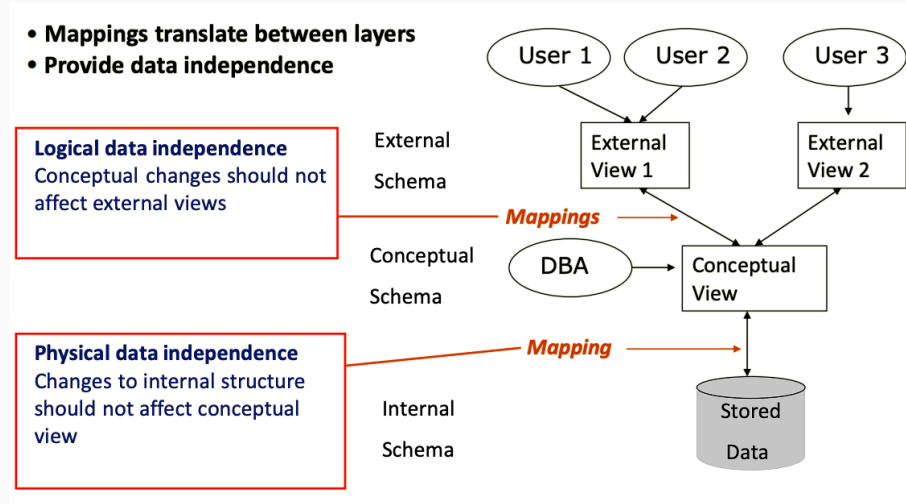
Relational Model

The Relational Model

- The relational model is an approach to managing data

- Originally Introduced by E.F. Codd in his paper
  - "A Relational Model of Data for Large Shared Databanks", 1970

- The foundation for most (but not all) modern database systems

- The model uses a structure and language that is consistent with
  - First-order predicate logic

- Provides a declarative method for specifying data and queries

- Chapter 4 of the DB book.

Relational Data Structure

- Data is stored in **relations** which are tables with columns and rows.
- An **attribute** is a named column of a relation
- A mathematical relation is a set of **tuples** (rows), e.g., (John, 23)

Relation

John	23
Mary	20
Mark	18
Jane	21
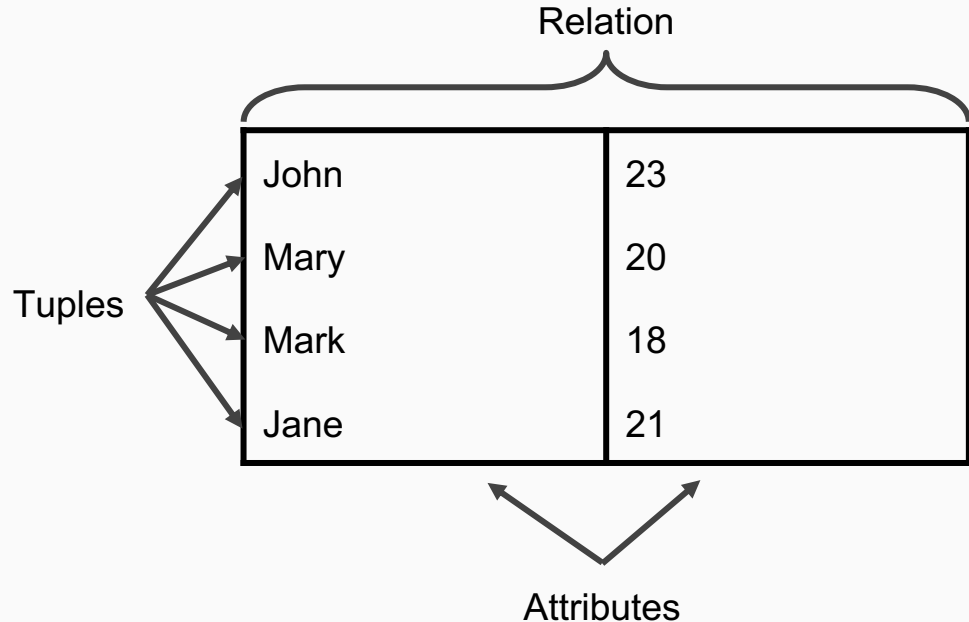
Tuples

Attributes

Relational Data Structure

- Each relation has a *schema*
  - sometimes called
    - scheme
    - heading
- The *schema* defines the relation's attributes (columns).
- Each column has a **domain** - a set from which all possible values can come.

Name	Age
John	23
Mary	20
Mark	18
Jane	21

Relation

Schema
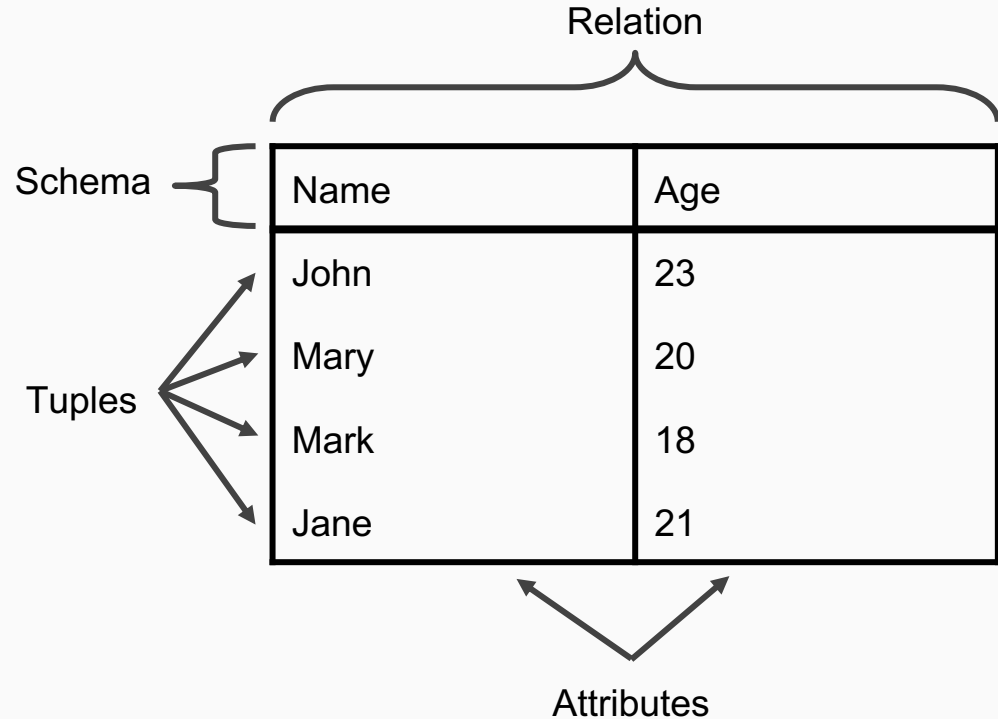
Tuples

Attributes

Named and Unnamed Tuples

- Tuples specify values for each attribute in a relation

- When writing tuples down, they can be named as sets of pairs, e.g.
  - { (Name, John), (Age, 23) }

- Or unnamed, for convenience, e.g.
  - (John, 23) (equivalent to the above)

- There is no real difference between named and unnamed tuples, but be careful with the ordering of unnamed tuples.

Relational Data Structure

- More formally:
  - A relational **schema** is a set of **attributes**
  - A **tuple** assigns a value to each **attribute** in the **schema**
  - A **relation** is a set of **tuples** with the same **schema**

Name	Age
John	23
Mary	20
Mark	18
Jane	21

{ { (Name, John), (Age, 23) },
{ (Name, Mary), (Age, 20) },
{ (Name, Mark), (Age, 18) },
{ (Name, Jane), (Age, 21) } }

Terminology

- **Degree of a relation**: how long each tuple is, or ==how many columns the== table has
  - In the previous example (name, age), the degree of the relation is 2

- **Cardinality of a relation**: ==how many different tuples there are==, or how many rows a table has
  - The previous example had a cardinality of - 4

Example Relation

ID	Name	Salary	Department
M139	John Smith	18000	Marketing
M140	Mary Jones	22000	Marketing
A368	Jane Brown	22000	Accounts
P222	Mark Brown	24000	Personnel
A367	David Jones	20000	Accounts

Example Relation

ID	Name	Salary	Department
M139	John Smith	18000	Marketing
M140	Mary Jones	22000	Marketing
A368	Jane Brown	22000	Accounts
P222	Mark Brown	24000	Personnel
A367	David Jones	20000	Accounts

Schema is { ID, Name, Salary, Department }

Example Relation

Attributes are: ID, Name, Salary & Department

The degree of the relation is 4

Schema is { ID, Name, Salary, Department }

ID	Name	Salary	Department
M139	John Smith	18000	Marketing
M140	Mary Jones	22000	Marketing
A368	Jane Brown	22000	Accounts
P222	Mark Brown	24000	Personnel
A367	David Jones	20000	Accounts

Example Relation

Attributes are: ID, Name, Salary & Department

The degree of the relation is 4

ID	Name	Salary	Department
M139	John Smith	18000	Marketing
M140	Mary Jones	22000	Marketing
A368	Jane Brown	22000	Accounts
P222	Mark Brown	24000	Personnel
A367	David Jones	20000	Accounts

Schema is { ID, Name, Salary, Department }

Tuples, e.g.
{ (ID, A368),
(Name, Jane Brown),
(Salary, 22,000),
(Department, Accounts) }

The cardinality of the relation is 5

Properties of Relations

- Each cell contains ==exactly one single value.==

- Each attribute has ==a distinct name.==

- The values of an attribute ==are all from the same domain.==

- Each tuple is distinct, i.e., ==no duplicate tuples.==

- The order of attributes has no significance.

- The order of tuples has no significance either.

Name	Age
John	23
Mary	20
Mark	18
Jane	21

Candidate Keys

A candidate key is a single field or the least combination of fields that uniquely identifies each record in the table

A **set of attributes** in a relation is a candidate **key** if, and only if:

- Every tuple has a unique value for that set of attributes: **uniqueness**
- No proper subset of the set has the uniqueness property: **minimality** **(Superkey)**

ID	First	Last
S139	Alan	Carr
S140	Jo	Brand
S141	Alan	Davies
S142	Jimmy	Carr

Assessing potential Candidate Keys		
Attribute(s)	Unique	Minimal
{ID}	Yes	Yes
{First, Last}	No	-
{ID, First}, {ID, Last}, {ID, First, Last}	Yes	No
{First}, {Last}	No	-

Choosing Candidate Keys

- You can't necessarily infer the candidate keys based solely on the data in your table
  - More often than not, an instance of a relation will only hold a small subset of all the possible values

- You must use knowledge of the real-world to help

Choosing Candidate Keys

What are the candidate keys of the following relation?

OfficeID	Name	Country	Postcode	Phone
O1001	Headquarters	UK	W1 1AA	0044 20 1545 3241
O1002	R&D Labs	UK	W1 1AA	0044 20 1545 4984
O1003	US West	USA	94130	001 415 665981
O1004	US East	USA	10201	001 212 448731
O1005	Telemarketing	UK	NE5 2GE	0044 1909 559862
O1006	Telemarketing	USA	84754	001 385 994763

Choosing Candidate Keys

The candidate keys are : {OfficeID}, {Phone}, {Name, Postcode/Zip}, {Name, Country}

Note: Keys like {Name, Postcode/Zip, Phone} satisfy uniqueness, but not minimality

OfficeID	Name	Country	Postcode	Phone
O1001	Headquarters	UK	W1 1AA	0044 20 1545 3241
O1002	R&D Labs	UK	W1 1AA	0044 20 1545 4984
O1003	US West	USA	94130	001 415 665981
O1004	US East	USA	10201	001 212 448731
O1005	Telemarketing	UK	NE5 2GE	0044 1909 559862
O1006	Telemarketing	USA	84754	001 385 994763

Primary Keys

- One candidate key is usually chosen to identify tuples in a relation

- This is called the Primary Key or just Key

- Often a special ID is used as the Primary Key

ID is Unique and Minimal = Good Candidate for Primary Key

ID	First	Last
S139	Alan	Carr
S140	Jo	Brand
S141	Alan	Davies
S142	Jimmy	Carr

NULLs and Primary Keys

- Missing information can be represented using **NULLs**

- A **NULL** indicates a missing or unknown value

- **NULL** is not the same as 0 or blank space character

- Entity integrity:
  - Can Primary Keys contain **NULL** values?

Foreign Keys

- Foreign Keys are used to <mark>link data in two relations.</mark>

- A set of attributes in the first (referencing) relation is a Foreign Key if its value:

    - Matches a Primary/Candidate Key value in a second (referenced) relation

  or:

    - Is NULL

- This is called **Referential Integrity**

Foreign Key Example

Department

DID	DName
13	Marketing
14	Accounts
15	Personnel

Employee

EID	EName	DID
15	John Smith	13
16	Mary Brown	14
17	Mark Jones	13
18	Jane Smith	NULL

{*DID*} is a Candidate Key for **Department** – Each entry has a unique value for DID

{*DID*} is a Foreign Key in **Employee** – each employee's *DID* value is either NULL, or matches an entry in the **Department** relation. This links each **Employee** to at most one **Department**

Referential Integrity

- When relations are updated, referential integrity might be violated
- This usually occurs when a referenced tuple is updated or deleted
- There are a number of options when this occurs:

- RESTRICT
  - Stop the user from doing it
- CASCADE
  - Let the changes flow on
- SET NULL
  - Make referencing values null

Referential Integrity Example

- What happens if
  - Marketing's *DID* is changed to 16 in **Department**?
  - The entry for Accounts is deleted from **Department**
- Using RESTRICT, CASCADE and SET NULL

Department

DID	DName
13	Marketing
14	Accounts
15	Personnel

Employee

EID	EName	DID
15	John Smith	13
16	Mary Brown	14
17	Mark Jones	13
18	Jane Smith	NULL

RESTRICT

- What happens if :
  - Marketing's *DID* is changed to 16 in **Department**?
  - The entry for *Accounts* is deleted from **Department**

Department

DID	DName
13	Marketing
14	Accounts
15	Personnel

Employee

EID	EName	DID
15	John Smith	13
16	Mary Brown	14
17	Mark Jones	13
18	Jane Smith	NULL

RESTRICT

- **RESTRICT** stops any action that violates integrity
  - You cannot update or delete *Marketing* or *Accounts*
  - You *can* change Personnel as it is not referenced

Department

DID	DName
13	Marketing
14	Accounts
15	Personnel

Employee

EID	EName	DID
15	John Smith	13
16	Mary Brown	14
17	Mark Jones	13
18	Jane Smith	NULL

CASCADE

- What happens if
  - Marketing's *DID* is changed to 16 in **Department**?
  - The entry for *Accounts* is deleted from **Department**

Department

DID	DName
13	Marketing
14	Accounts
15	Personnel

Employee

EID	EName	DID
15	John Smith	13
16	Mary Brown	14
17	Mark Jones	13
18	Jane Smith	NULL

CASCADE

- CASCADE allows the changes made to flow through
  - If Marketing's *DID* is changed to 16 in **Department**, then the *DIDs* for John Smith and Mark Jones also change
  - If *Accounts* is deleted then so is Mary Brown

Department

DID	DName
~~13~~ 16	Marketing
~~14~~	~~Accounts~~
15	Personnel

Employee

EID	EName	DID
15	John Smith	~~13~~ 16
~~16~~	~~Mary Brown~~	~~14~~
17	Mark Jones	~~13~~ 16
18	Jane Smith	NULL

SET NULL

- What happens if
  - Marketing's *DID* is changed to 16 in **Department**?
  - The entry for *Accounts* is deleted from **Department**

Department

DID	DName
13	Marketing
14	Accounts
15	Personnel

Employee

EID	EName	DID
15	John Smith	13
16	Mary Brown	14
17	Mark Jones	13
18	Jane Smith	NULL

SET NULL

- SET NULL allows the changes to happen but
  - If *Marketing's DID* is changed to 16 in **Department**, then the *DIDs* for John Smith and Mark Jones is set to **NULL**
  - If *Accounts* is deleted then Mary Brown's *DID* is set to **NULL**

Department

DID	DName
~~13~~ 16	Marketing
~~14~~	~~Accounts~~
15	Personnel

Employee

EID	EName	DID
15	John Smith	~~13~~ NULL
16	Mary Brown	~~14~~ NULL
17	Mark Jones	~~13~~ NULL
18	Jane Smith	NULL

Additional Reading

- Databases Book
  - Chapter 1 - Introduction to Databases
  - Chapter 4 - The Relational Model
    - 4.1 → 4.3

- Codd, Edgar Frank. "A relational model of data for large shared data banks." *Communications of the ACM* 26.1 (1983): 64-69.