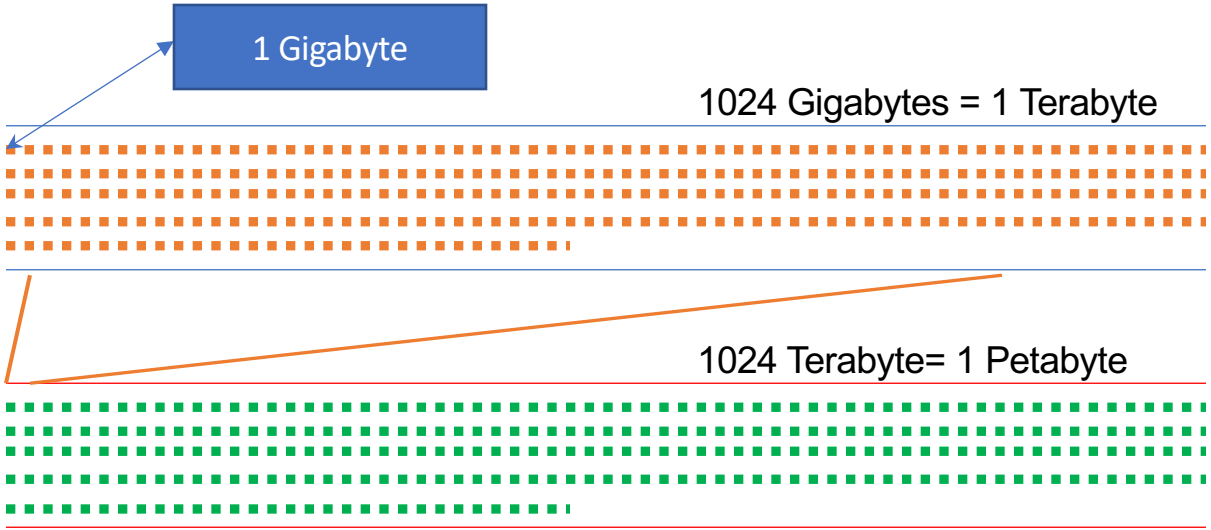
Today

- The Need for Databases
- Data Models
- Relational Databases
- Database Design
- Transaction Manager

How Big is Petabyte?

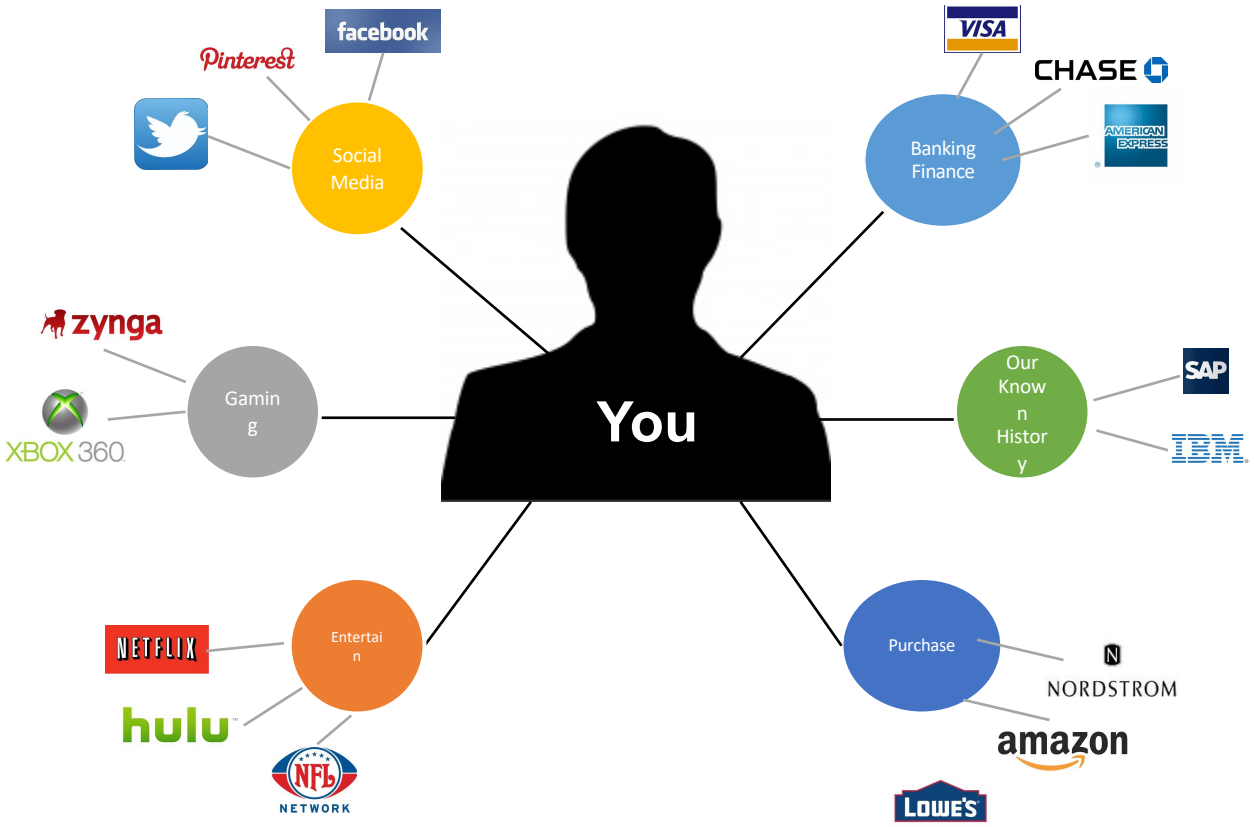
- First we must understand a Gigabyte
- 1 GB = 7 Minutes of HD-TV Video
- 2 GB = 20 Yards of Books on a shelf
- 4.7 GB = A standard DVD



Source – Mozy.com

The NSA
is thought to analyze
1.6% of all global
internet traffic –
around 30 petabytes
(30 million gigabytes)
every day.

Source


facebook
Pinterest
Social Media
VISA
CHASE
AMERICAN EXPRESS
Banking Finance
zynga
XBOX 360
Gamin g
You
Our Know n Histor y
SAP
IBM
NETFLIX
hulu
NFL NETWORK
Entertai n
Purchase
NORDSTROM
amazon
LOWE'S

Database Management System (DBMS)

- DBMS contains information about a particular enterprise
  - Collection of interrelated data
  - Set of programs to access the data
  - An environment that is both *convenient* and *efficient* to use
- Database Applications:
  - Banking: transactions
  - Airlines: reservations, schedules
  - Universities: registration, grades
  - Sales: customers, products, purchases
  - Online retailers: order tracking, customized recommendations
  - Manufacturing: production, inventory, orders, supply chain
  - Human resources: employee records, salaries, tax deductions
- Databases can be very large.
- Databases touch all aspects of our lives

University Database Example

- Application program examples
  - Add new students, instructors, and courses
  - Register students for courses, and generate class rosters
  - Assign grades to students, compute grade point averages (GPA) and generate transcripts
- In the early days, database applications were built directly on top of file systems

Drawbacks of using file systems to store data

- Data redundancy and inconsistency
  - Multiple file formats, duplication of information in different files
- Difficulty in accessing data
  - Need to write a new program to carry out each new task
- Data isolation
  - Multiple files and formats
- Integrity problems
  - Integrity constraints (e.g., account balance > 0) become “buried” in program code rather than being stated explicitly
  - Hard to add new constraints or change existing ones

Drawbacks of using file systems to store data (Cont.)

- Atomicity of updates
  - Failures may leave database in an inconsistent state with partial updates carried out
  - Example: Transfer of funds from one account to another should either complete or not happen at all
- Concurrent access by multiple users
  - Concurrent access needed for performance
  - Uncontrolled concurrent accesses can lead to inconsistencies
    - Example: Two people reading a balance (say 100) and updating it by withdrawing money (say 50 each) at the same time
- Security problems
  - Hard to provide user access to some, but not all, data

Database systems offer solutions to all the above problems

Levels of Abstraction

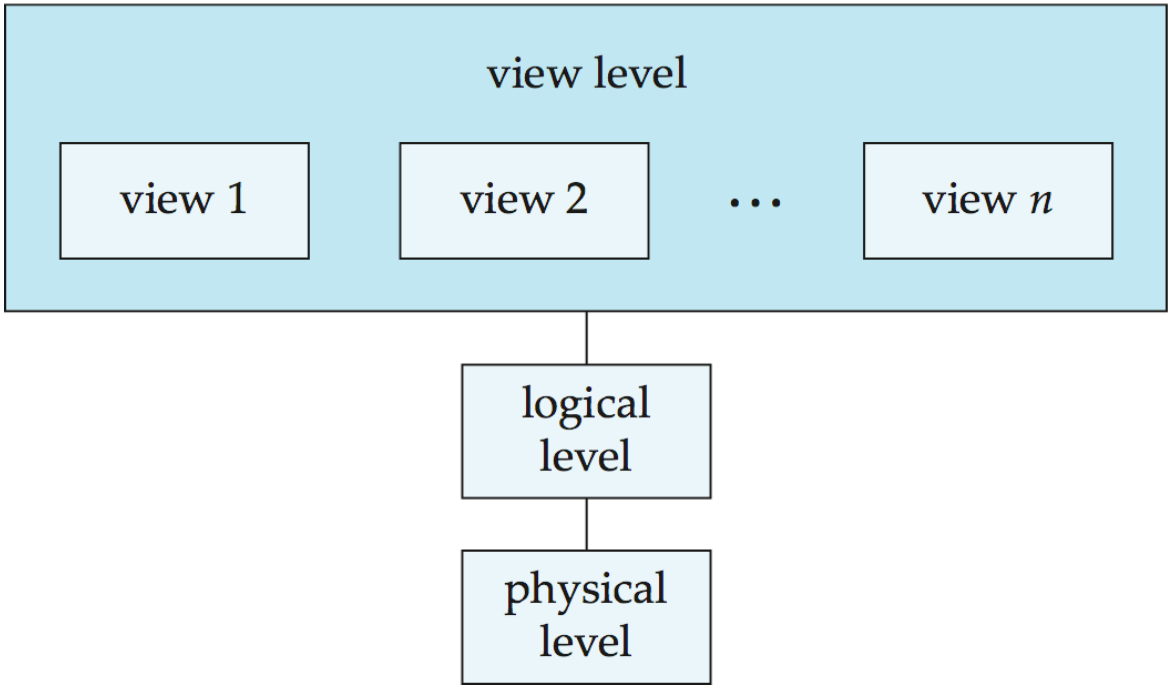
- **Physical level:** describes how a record (e.g., instructor) is stored.
- **Logical level:** describes data stored in database, and the relationships among the data.

```
type instructor = record
        ID : string;
        name : string;
        dept_name : string;
        salary : integer;
        end;
```

- **View level:** application programs hide details of data types. Views can also hide information (such as an employee's salary) for security purposes.

View of Data

An architecture for a database system



Instances and Schemas

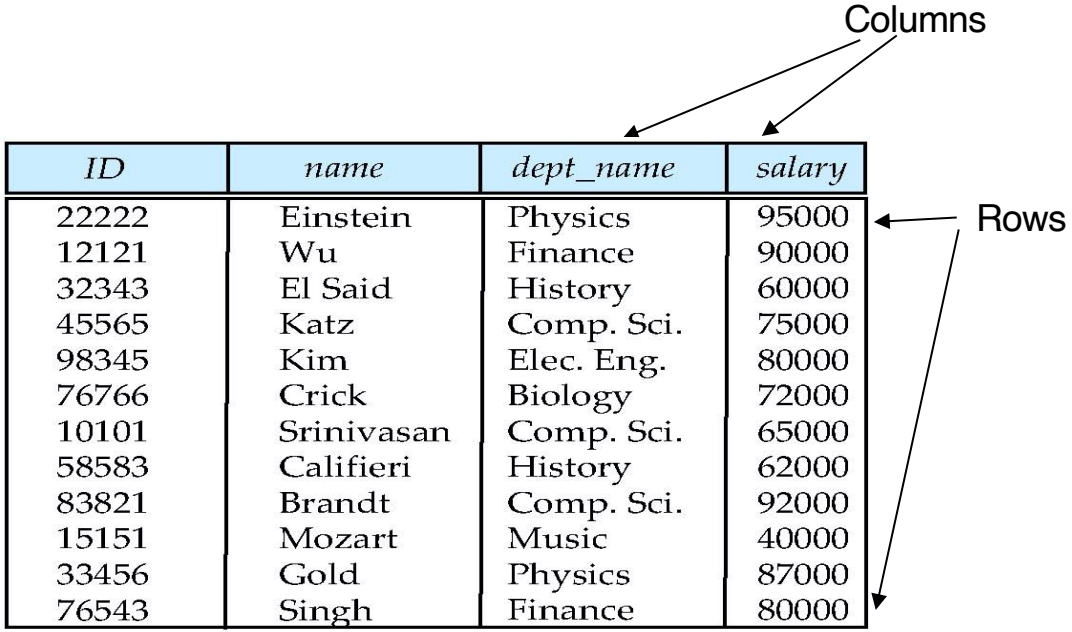
- Similar to types and variables in programming languages
- **Logical Schema** – the overall logical structure of the database
  - Example: The database consists of information about a set of customers and accounts in a bank and the relationship between them
    - Analogous to type information of a variable in a program
- **Physical schema**– the overall physical structure of the database
- **Instance** – the actual content of the database at a particular point in time
  - Analogous to the value of a variable
- **Physical Data Independence** – the ability to modify the physical schema without changing the logical schema
  - Applications depend on the logical schema
  - In general, the interfaces between the various levels and components should be well defined so that changes in some parts do not seriously influence others.

Data Models

- A collection of tools for describing
  - Data
  - Data relationships
  - Data semantics
  - Data constraints
- Relational model
- Entity-Relationship data model (mainly for database design)
- Object-based data models (Object-oriented and Object-relational)

Relational Model

- All the data is stored in various tables.
- Example of tabular data in the relational model



ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

(a) The *instructor* table

A Sample Relational Database

ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

(a) The *instructor* table

dept_name	*building*	*budget*
Comp. Sci.	Taylor	100000
Biology	Watson	90000
Elec. Eng.	Taylor	85000
Music	Packard	80000
Finance	Painter	120000
History	Painter	50000
Physics	Watson	70000

(b) The *department* table

Data Definition Language (DDL)

- Specification notation for defining the database schema

  Example: **create table** *instructor* (
  ID **char**(5),
  name **varchar**(20)**,**
  dept_name **varchar**(20),
  salary **numeric**(8,2))

- DDL compiler generates a set of table templates stored in a ***data dictionary***
- Data dictionary contains metadata (i.e., data about data)
  - Database schema
  - Integrity constraints
    - Primary key (ID uniquely identifies instructors)
  - Authorization
    - Who can access what

Data Manipulation Language (DML)

- Language for accessing and manipulating the data organized by the appropriate data model
  - DML also known as query language
- Two classes of languages
  - **Pure** – used for proving properties about computational power and for optimization
    - Relational Algebra
    - Tuple relational calculus
    - Domain relational calculus
  - **Commercial** – used in commercial systems
    - SQL is the most widely used commercial language

SQL

- The most widely used commercial language
- SQL is NOT a Turing machine equivalent language
- SQL is NOT a Turing machine equivalent language
- To be able to compute complex functions SQL is usually embedded in some higher-level language
- Application programs generally access databases through one of
  - Language extensions to allow embedded SQL
  - Application program interface (e.g., ODBC/JDBC) which allow SQL queries to be sent to a database

Database Design

The process of designing the general structure of the database:

- Logical Design – Deciding on the database schema. Database design requires that we find a "good" collection of relation schemas.
  - Business decision – What attributes should we record in the database?
  - Computer Science decision – What relation schemas should we have and how should the attributes be distributed among the various relation schemas?
- Physical Design – Deciding on the physical layout of the database

Database Design (Cont.)

- Is there any problem with this relation?

ID	*name*	*salary*	*dept_name*	*building*	*budget*
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Watson	90000
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
83821	Brandt	92000	Comp. Sci	Taylor	100000
15151	Mozart	40000	Music	Packard	80000
33456	Gold	87000	Physics	Watson	70000
76543	Singh	80000	Finance	Painter	120000

Design Approaches

- Need to come up with a methodology to ensure that each of the relations in the database is “good”
- Two ways of doing so:
  - Entity Relationship Model
    - Models an enterprise as a collection of *entities* and *relationships*
    - Represented diagrammatically by an *entity-relationship diagram:*
  - Normalization Theory
    - Formalize what designs are bad, and test for them

Object-Relational Data Models

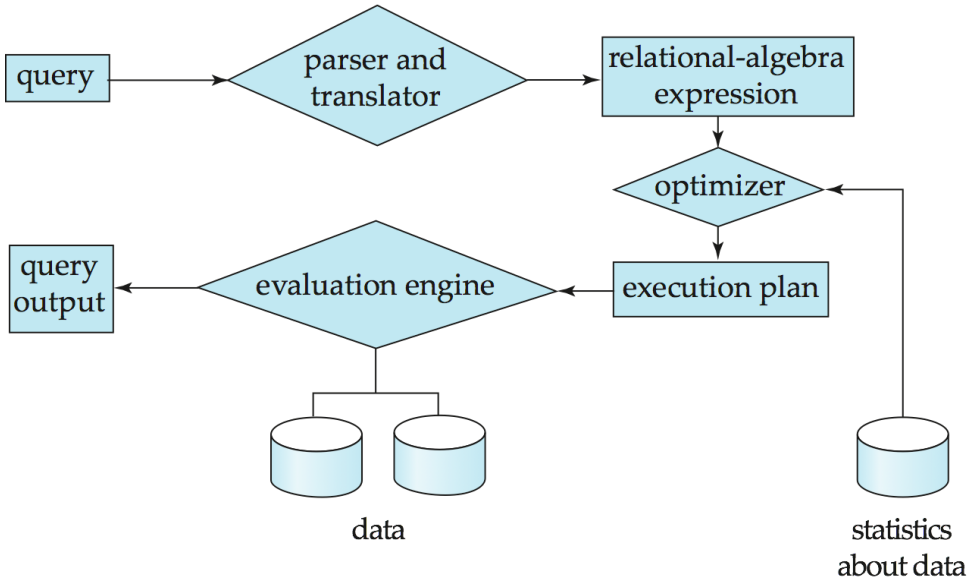
- Relational model: flat, “atomic” values
- Object Relational Data Models
  - Extend the relational data model by including object orientation and constructs to deal with added data types.
  - Allow attributes of tuples to have complex types, including non-atomic values such as nested relations.
  - Preserve relational foundations, in particular the declarative access to data, while extending modeling power.
  - Provide upward compatibility with existing relational languages.

Database Engine

- Storage manager
- Query processing
- Transaction manager

Query Processing

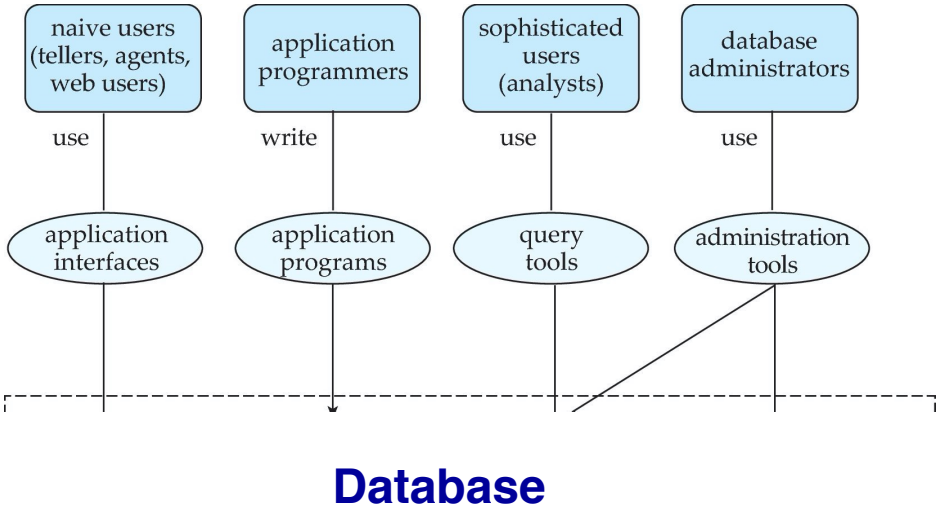
1. Parsing and translation
2. Optimization
3. Evaluation



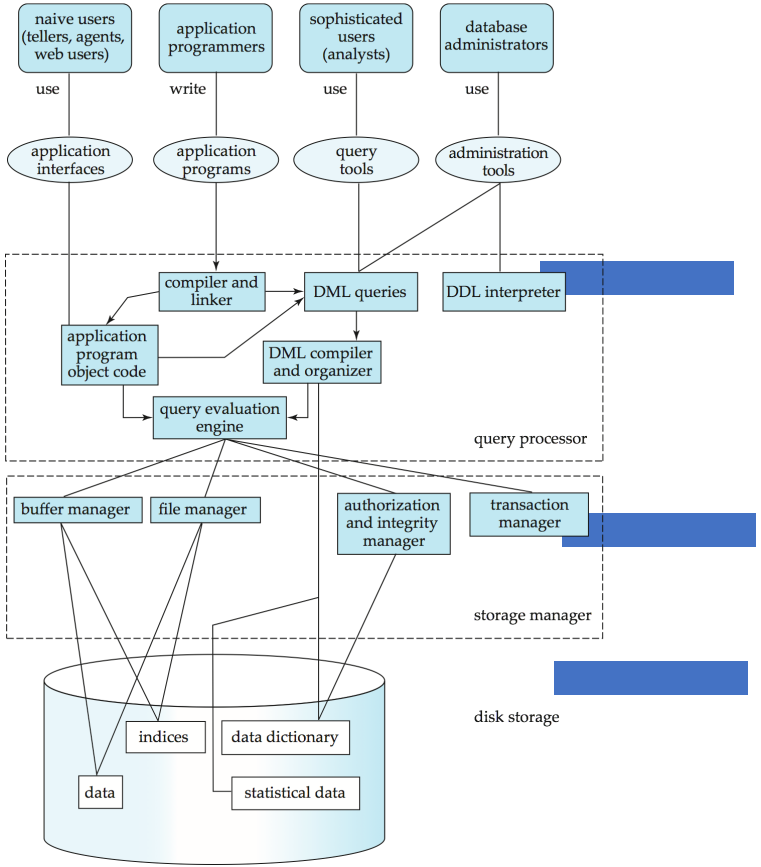
Transaction Management

- What if the system fails?
- What if more than one user is concurrently updating the same data?
- A **transaction** is a collection of operations that performs a single logical function in a database application
- **Transaction-management component** ensures that the database remains in a consistent (correct) state despite system failures (e.g., power failures and operating system crashes) and transaction failures.
- **Concurrency-control manager** controls the interaction among the concurrent transactions, to ensure the consistency of the database.

Database Users and Administrators



Database System Internals



Database Architecture

The architecture of a database systems is greatly influenced by

the underlying computer system on which the database is running:

- Centralized
- Client-server
- Parallel (multi-processor)
- Distributed

History of Database Systems

- 1950s and early 1960s:
  - Data processing using magnetic tapes for storage
    - Tapes provided only sequential access
  - Punched cards for input
- Late 1960s and 1970s:
  - Hard disks allowed direct access to data
  - Network and hierarchical data models in widespread use
  - Ted Codd defines the relational data model
    - Would win the ACM Turing Award for this work
    - IBM Research begins System R prototype
    - UC Berkeley begins Ingres prototype
  - High-performance (for the era) transaction processing

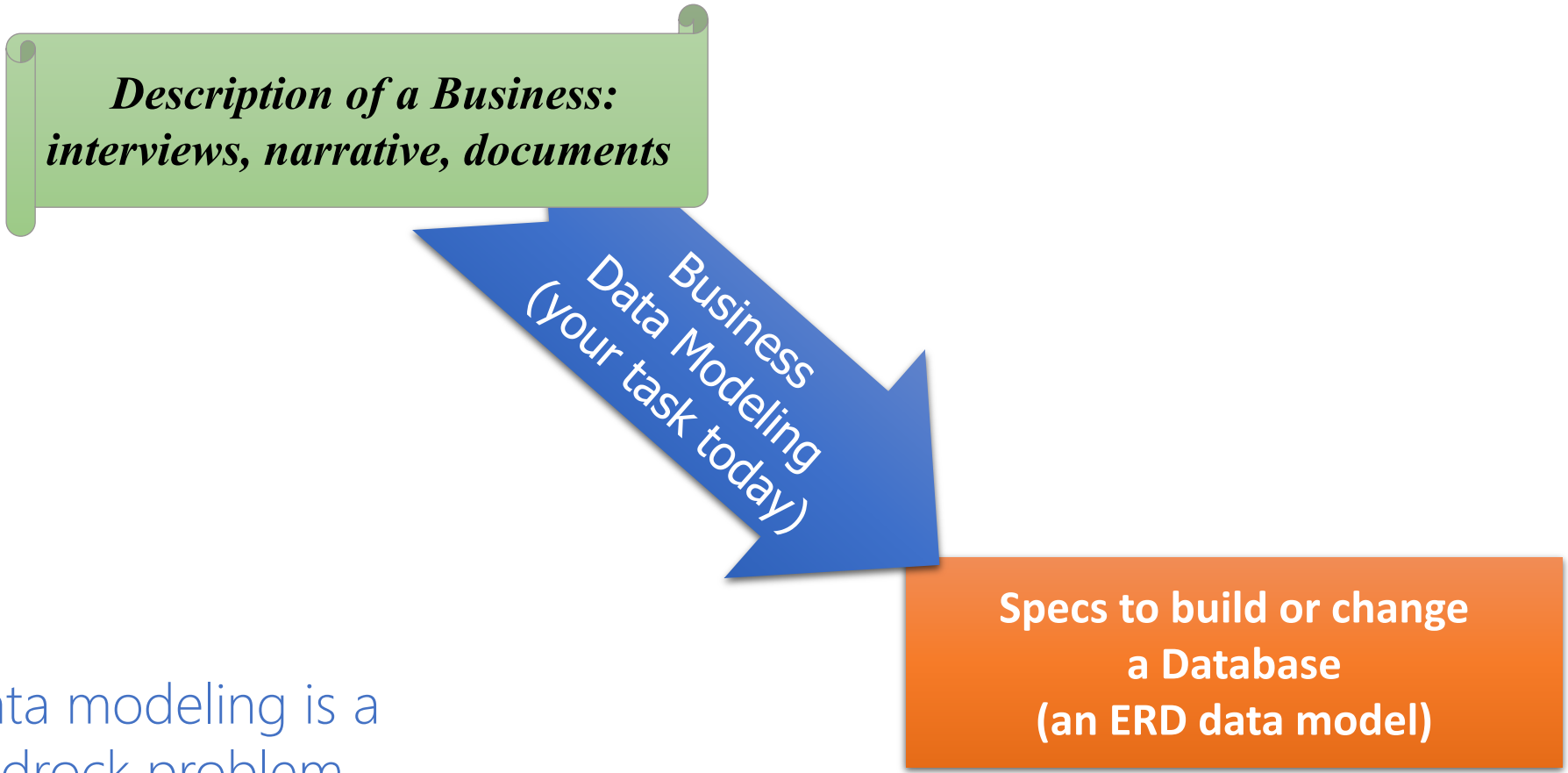
History (cont.)

- 1980s:
  - Research relational prototypes evolve into commercial systems
    - SQL becomes industrial standard
  - Parallel and distributed database systems
  - Object-oriented database systems
- 1990s:
  - Large decision support and data-mining applications
  - Large multi-terabyte data warehouses
  - Emergence of Web commerce
- Early 2000s:
  - XML and XQuery standards
  - Automated database administration
- Later 2000s:
  - Giant data storage systems
    - Google BigTable, Yahoo PNuts, Amazon, ..

Process Modeling and ERD

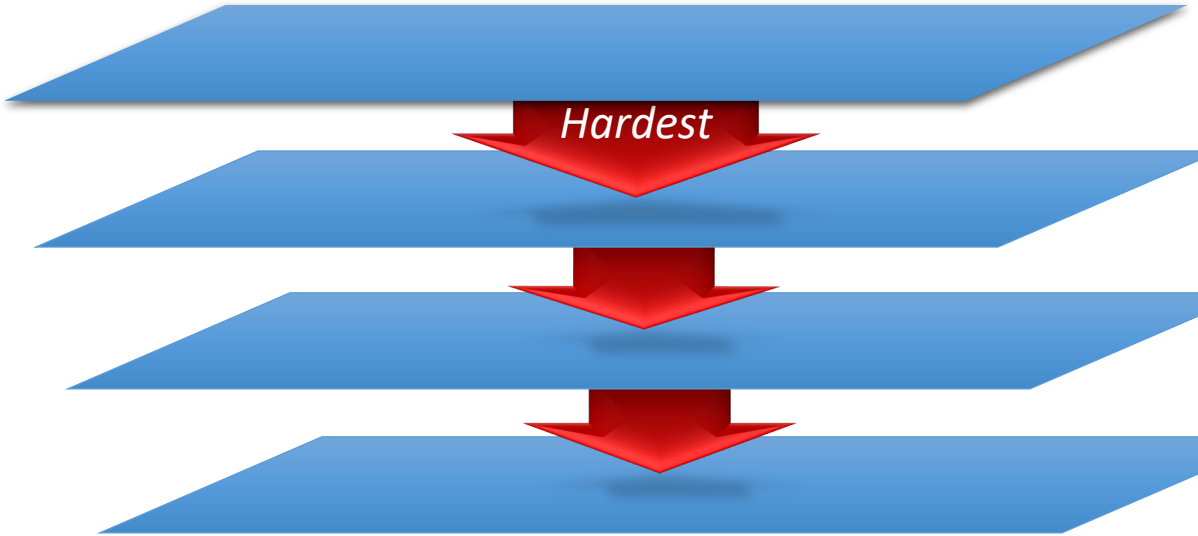
Logical Database Design:
Data Modeling

Data Modeling



Data modeling is a
bedrock problem

Using a divide-and-conquer approach to building databases



Business information

Logical ERD (our focus)

Physical: table specs

Implemented data structures (SW & HW)

Note: we differ from Hoffer et al. – this is just another way to express similar ideas

Process Modeling Next Week

Monday Today Next Week

Business Process Management

BP Modeling
1. "Pure" BPM

2. ERD

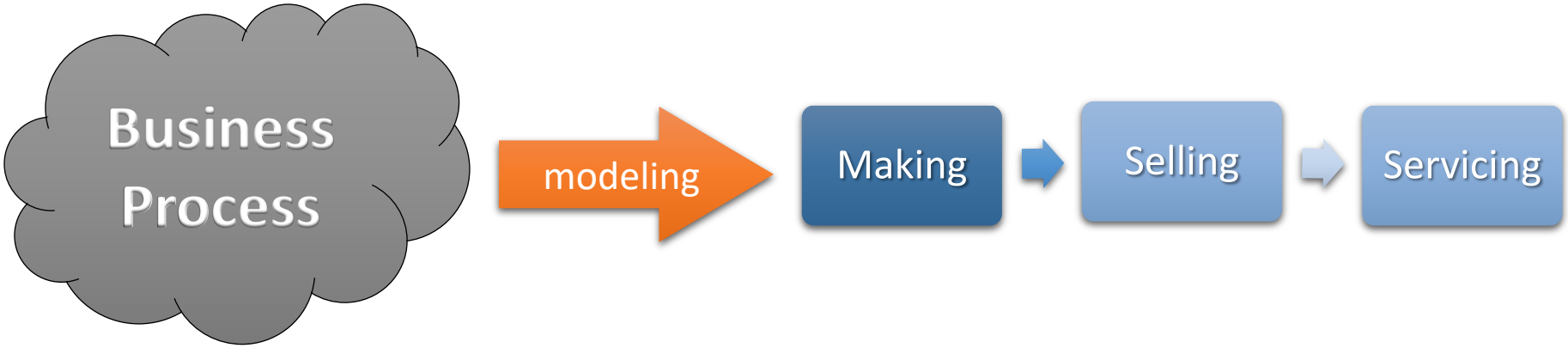
Workshop

Data Modeling - Graded

What is a Business Process?

a sequence of steps
that achieve a business goal.

What is a Business Process Model?



BP models are text and/or graphics that describe business processes: flowcharts, list of steps, recipes, storyboards, programs. Many types and techniques.

What do you use BP models for?

- Management tool
  - BP assessment and improvement: identifying inefficiencies, metrics, costing, training…
- Engineering tool
  - sketching, blueprinting, programming

Process modeling is much like summarizing a movie for a friend

- What makes a summary *good*?
- Is there such a thing as a *wrong* summary?
- Can a summary be factually correct *and* bad?
- Is the audience important?
- Is the language (idiom, syntax, grammar…) important?
- What if there is a syntax/grammar error in the summary?

“Pure” BP Modeling

Describe the process of
buying groceries
by writing down
half a dozen
key steps

Before jumping into modeling the process of buying groceries...

Step back and identify the 3 Ps:

- **P**urpose:
  - Why are we modeling? As-is, to be?
- **P**rocess:
  - What is this process, exactly? Beginning/end?
- **P**erspective:
  - Who does it?

“Pure” BP Modeling

Describe the process of buying groceries by writing down half a dozen key steps

Buying Groceries - Narrative

Customer perspective

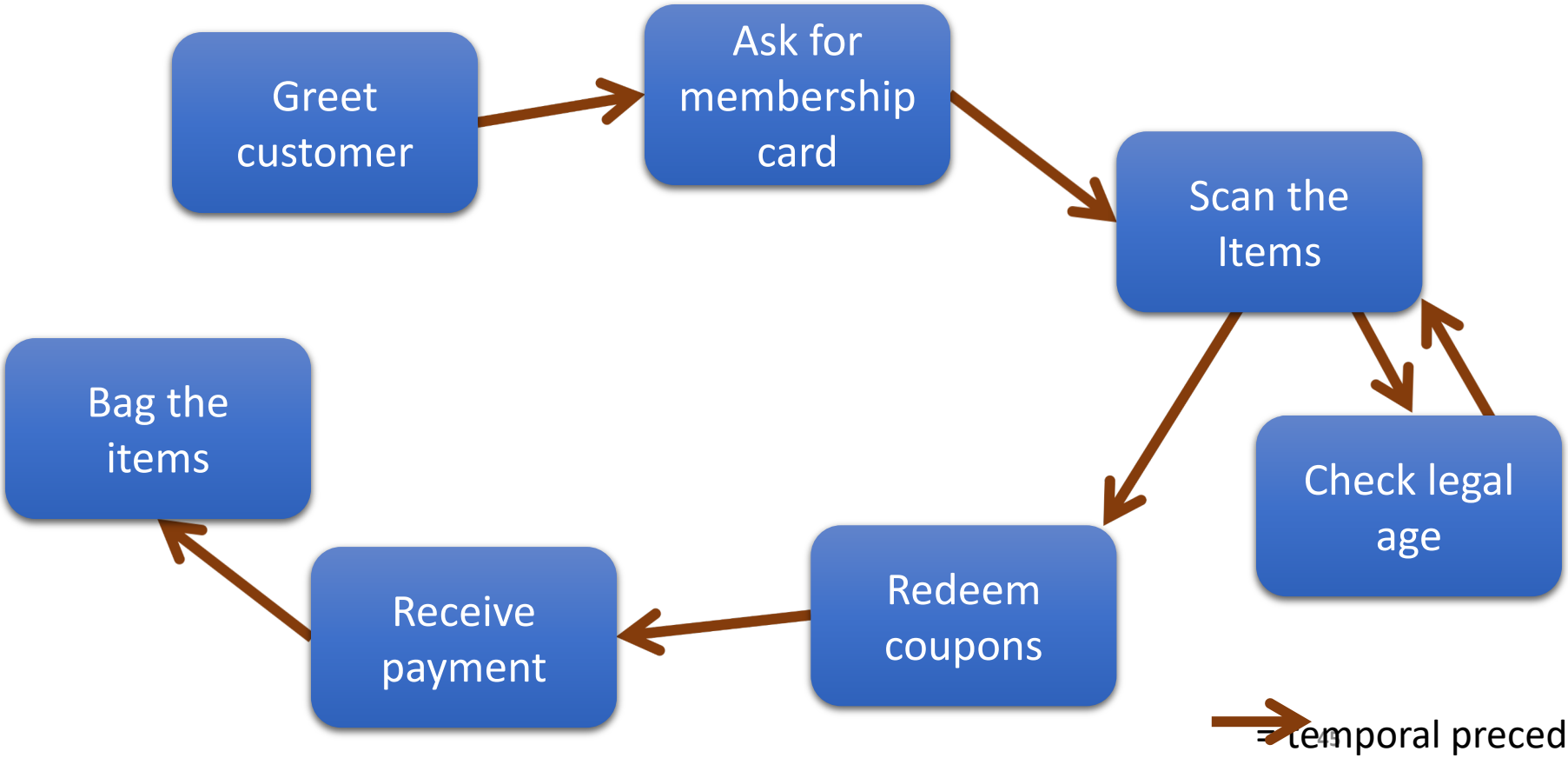
The customer enters the store, browses the items, picks what she/he wants to buy, lines up at the cash register, optionally presents a loyalty card to the cashier, redeems coupons, pays, picks up the bagged items and exits the store.

Buying Groceries - Key Steps

1. Greet the Customer,
2. Ask for membership card,
3. Scan the items,
4. Check legal age,
5. Redeem coupons,
6. Receive payment,
7. Bag the items.

Cashier perspective

A “Pure” BP Model has only processes

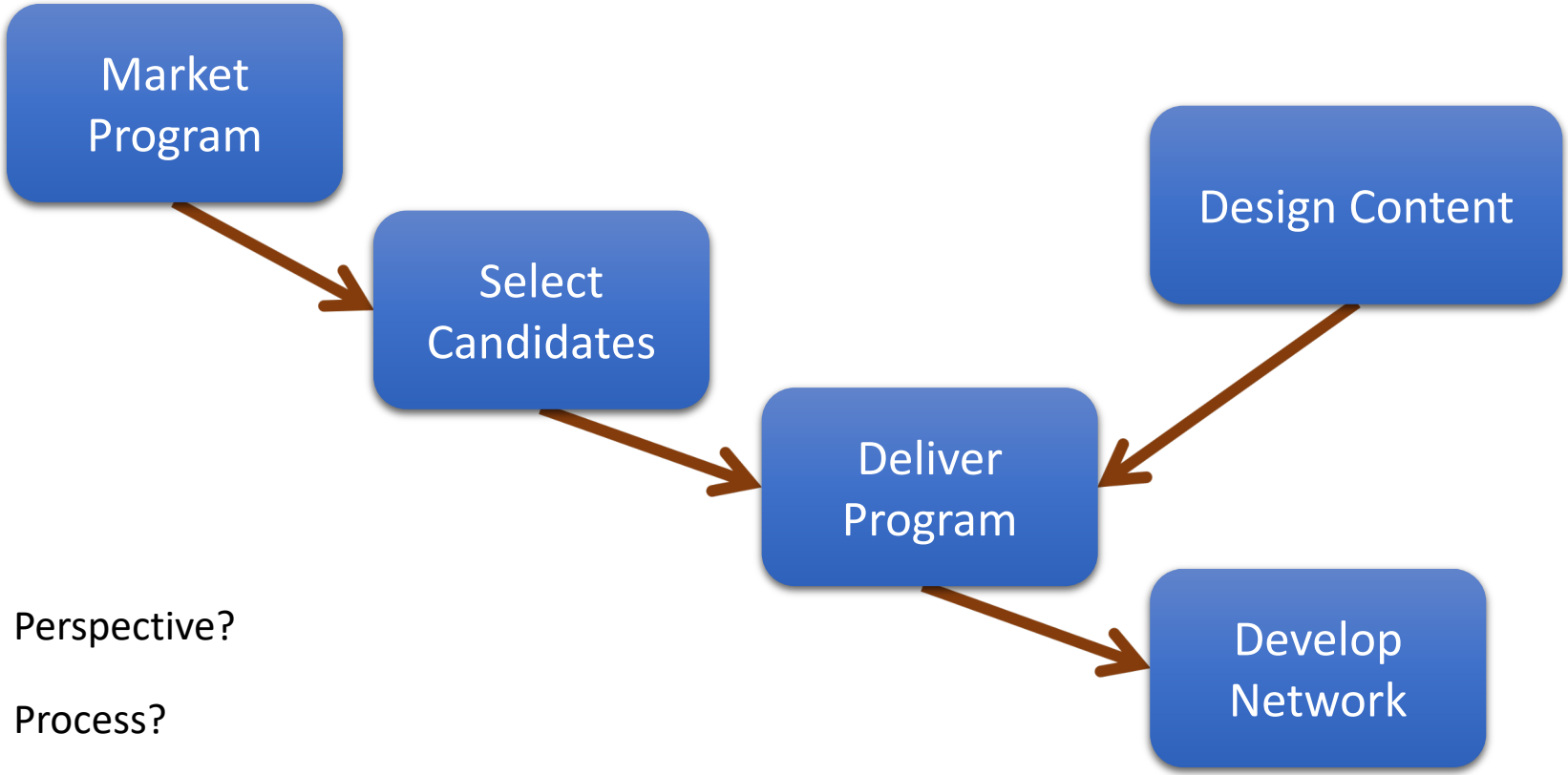


What if…

- I asked you to write down 4 steps? Possible?
- I asked you to write down 40 steps? Possible?

- What are the tradeoffs?

Another “Pure” BP Model: MBA Program



- Perspective?
- Process?
- Purpose?

Business Processes: Takeaways

- Incredibly useful. Frame for thinking.
- Appreciate that BPM is cognitively challenging for some individuals
- Keep in mind the 3 Ps. The answers to many question you might have about your models lie there
- Many “right” solutions and some wrong ones
- Good process models seem obviously clear

Takeaways (cont.)

- Pure BP models have no technology!

- Details can become overwhelming: *analysis paralysis*
  - Do not obsess on the “right” model
  - Begin by focusing on a **typical** & **successful** process, add detail later
  - Use zoom-in/zoom out techniques
  - Keep it simple, but do not sweep under the rug needed complexity.

Data modeling techniques are languages

ER Diagramming (Chen 1976)

- 1 - Entities
- 2 - Attributes
- 3 - Relationships

1 – Entities are conceptual types

Examples: Customer, Order, Stock, Employee…

- Entities are not instances.
  Entities *have* instances.

- Each instance must be uniquely identifiable.

ERD	RDBMS
Entity	Table
Instance	Record
Identifier	Primary key

Attributes are properties of entities

- Example: product name, description, price…

- An attribute *is not* a value, an attribute *has* a value for each instance of an entity.

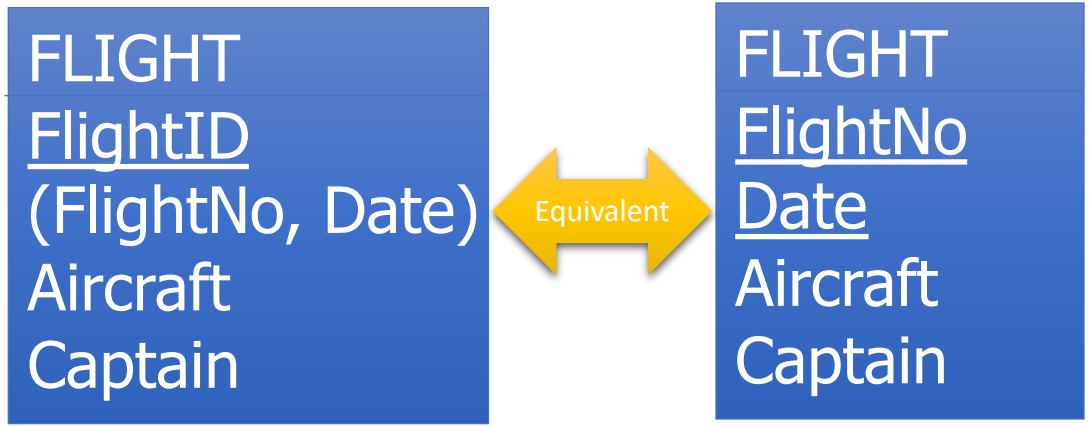
ERD	RDBMS
Attribute	Column / field
Value	Cell content / value

Identifiers are special attributes

- Example: CustomerID, ProductNo, …

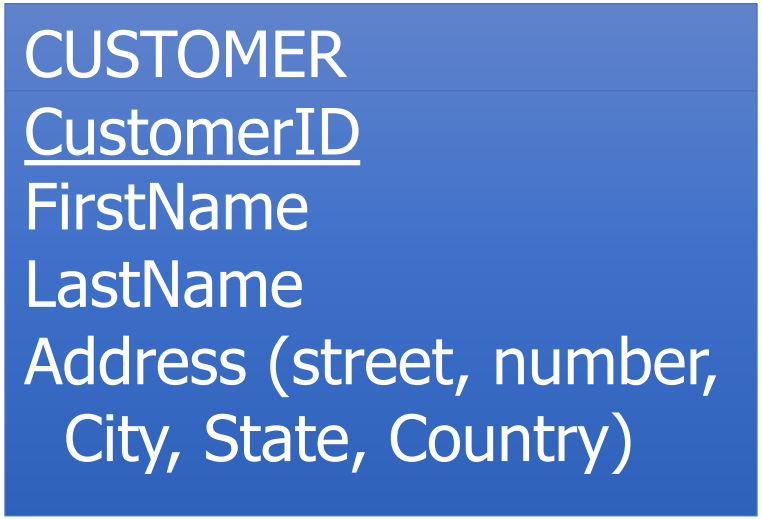
- Identifiers must be unique
- Composite identifiers

ERD	RDBMS
Identifier	Primary key



Types of Attributes

- Composite attributes
- Multivalued attributes



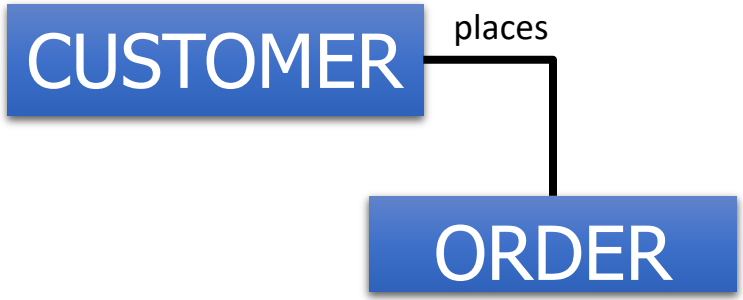
3 – Relationships express associations between entities

examples: “has,” “is_composed_of,” “works_in”

- A relationship is not a process
  - ‘customer places order’ only means that there is a list of customers and a list of orders, and that these two lists are connected, so that we know who placed what.

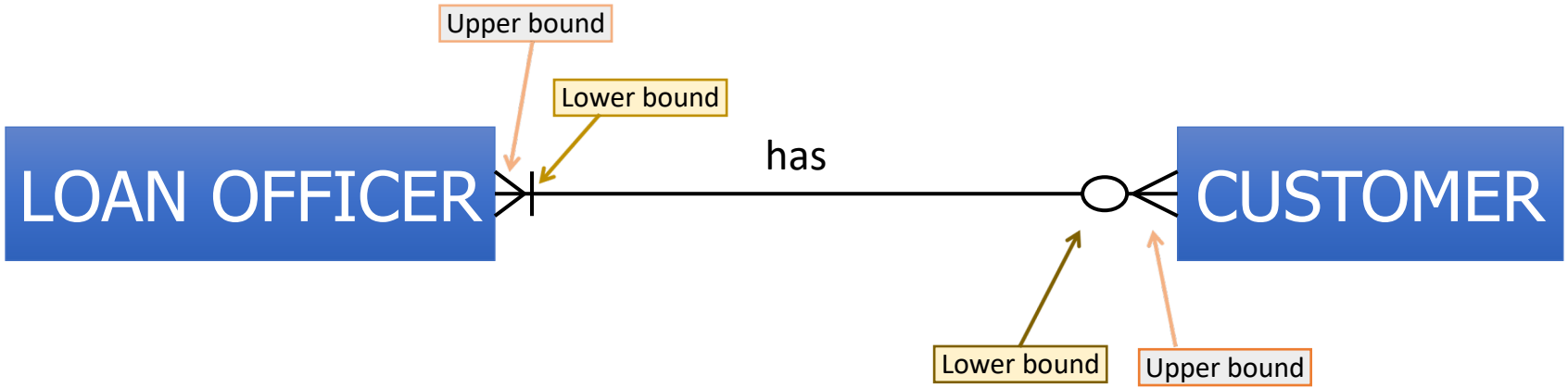
Common mistake

ERD	RDBMS
Relation	Foreign key



Relationships have a Cardinality

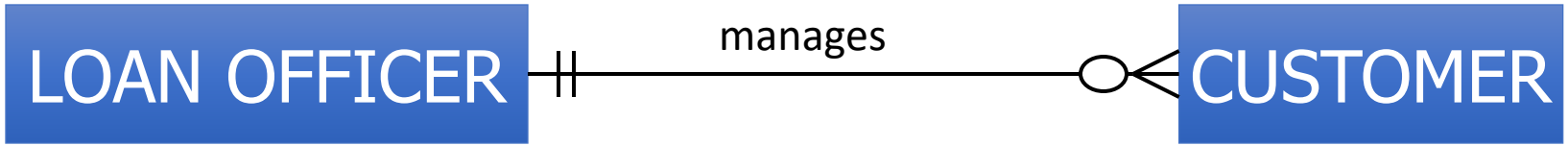
It is the number of instances in an entity that are associated to one instance in a related entity



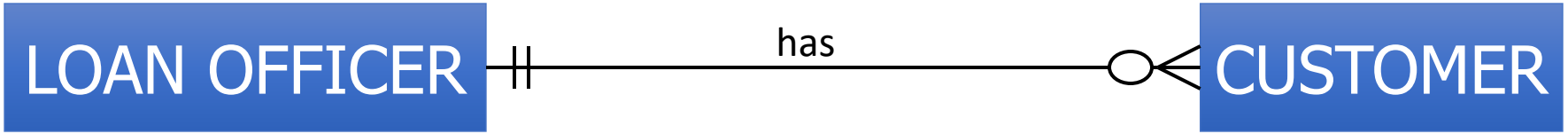
Read it this way: "Each loan officer has either no (zero) customers or many (n) customers. Each customer is assigned to at least one - and possibly many - loan officers."

“One To Many” Relationships

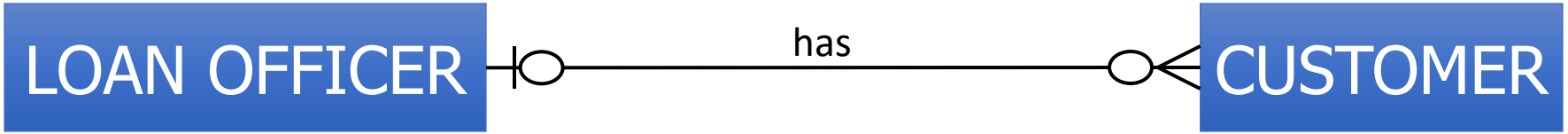
Each loan officer manages many customers. All customers have one and only one loan officer.



Mandatory Vs. Optional constraints



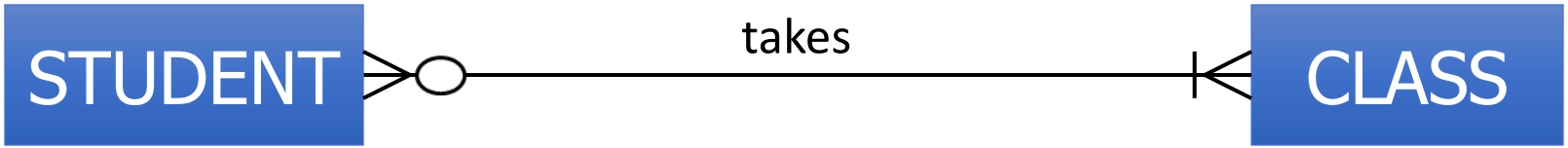
*Read it this way: "Each loan officer has either no customers or many customers; each customer is assigned to at least one and at most one loan officer (loan officer is **mandatory**)*



*Read it this way: each customer is assigned to either no loan officer or at most one loan officer" (loan officer is **optional**)*

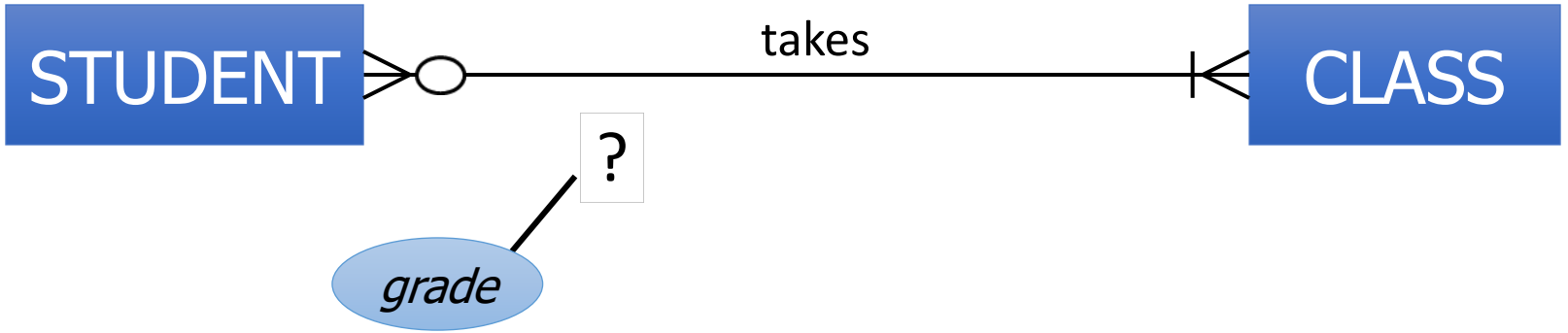
“Many To Many” relationships

Each student takes at least one class. Each class may be attended by several students.



Attaching information to *many to many* relationships

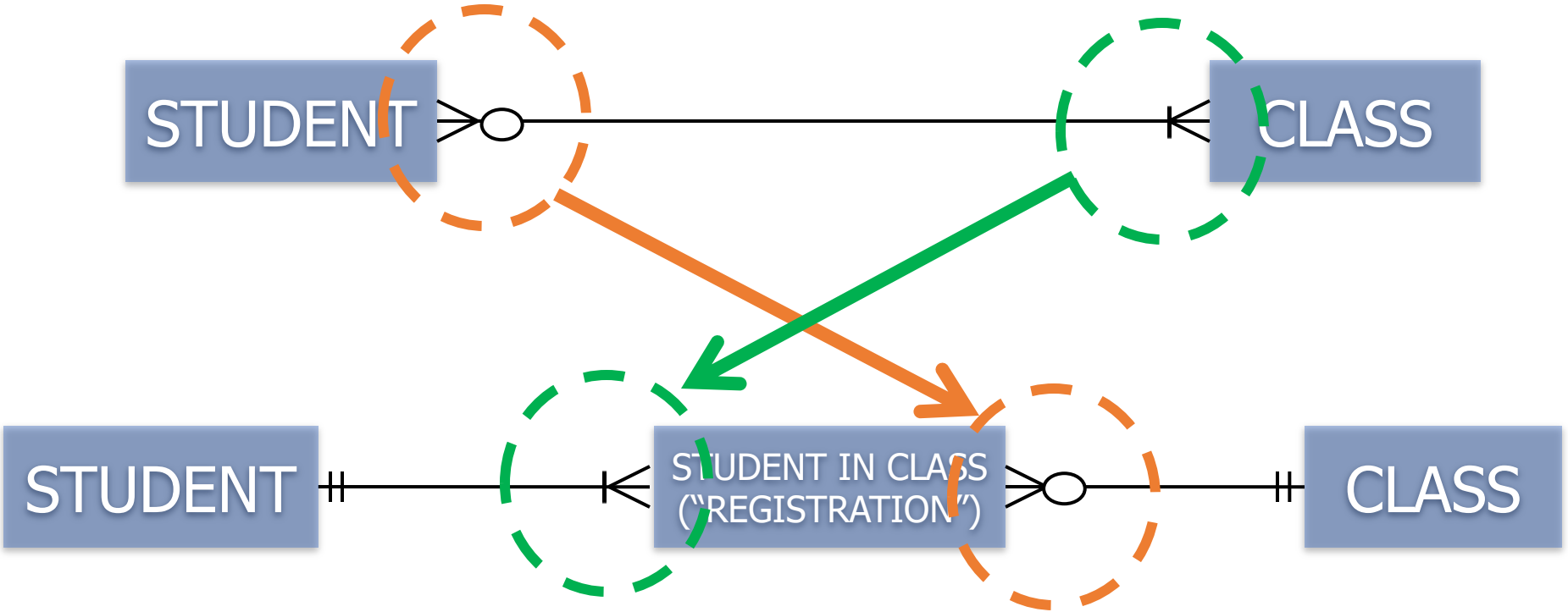
A student attend one or more classes. Each class may be attended by several students. *Students get grades for each class.*



Attaching information to *many to many* relationships

Students gets grades for each class.

Creating an Association Entity



Foreign Keys

A foreign key is an attribute in one entity and an identifier (primary key) in another.

Foreign keys implement the relationships between tables