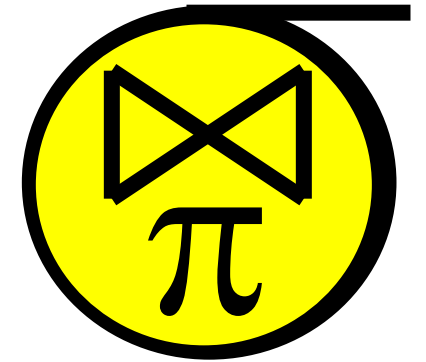




Information Storage and Management I

Dr. Alejandro Arbelaez



Relational Algebra
Relational Calculus

Another Example

country(name, code, capital)
city(name, country, population)
borders(country1, country2, length)

- Find the name of all countries that border Switzerland

Country

name	code	capital
Russia	1	Moscow
France	2	Paris
Spain	3	Madrid
Switzerland	4	Zurich
Italy	5	Rome
Germany	6	Berlin
...

borders

country1	country2
2	6
2	4
2	3
5	4
6	4
...	..



Another Example

country(name, code, capital)

city(name, country, population)

borders(country1, country2, length)

- Find the name of all countries that border Switzerland

Country

name	code	capital
Russia	1	Moscow
France	2	Paris
Spain	3	Madrid
Switzerland	4	Zurich
Italy	5	Rome
Germany	6	Berlin
...



borders

country1	country2
2	6
2	4
2	3
5	4
6	4
...	..



Another Example

country(name, code, capital)
city(name, country, population)
borders(country1, country2, length)

- Find the name of all countries that border Switzerland

Country

name	code	capital
Russia	1	Moscow
France	2	Paris
Spain	3	Madrid
Switzerland	4	Zurich
Italy	5	Rome
Germany	6	Berlin
...


$$\rho(s, \pi_{code}(\sigma_{name='Switzerland'}(country)))$$

s

code
4

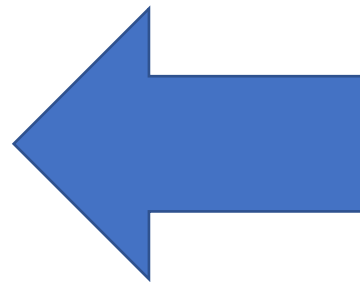
Another Example

country(name, code, capital)
city(name, country, population)
borders(country1, country2, length)

- Find the name of all countries that border Switzerland

temp1

country1	country2	code
2	6	4
2	4	4
2	3	4
5	4	4
6	4	4
...



$$\rho(s, \pi_{code}(\sigma_{name='Switzerland'}(country)))$$

$$\rho(b, borders)$$

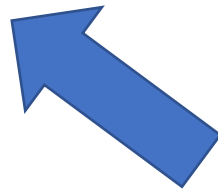
$$\rho(temp1, \sigma_{b.country2=s.code}(b \times s))$$

Another Example

country(name, code, capital)
city(name, country, population)
borders(country1, country2, length)

- Find the name of all countries that border Switzerland

country1	name	capital
2	France	Paris
5	Italy	Rome
6	Germany	Berlin
...


$$\rho(s, \pi_{code}(\sigma_{name='Switzerland'}(country)))$$
$$\rho(b, borders)$$
$$\rho(temp1, \sigma_{b.country2=s.code}(b \times s))$$
$$\pi_{name}(\sigma_{temp1.country1=country.code}(\pi_{country1}(temp1) \times country))$$

Another Example

country(name, code, capital)
city(name, country, population)
borders(country1, country2, length)

- Solution 1

$$\rho(s, \pi_{code}(\sigma_{name='Switzerland'}(country)))$$
$$\rho(b, borders)$$

$$\rho(temp1, \sigma_{b.country2=s.code}(b \times s))$$

$$\pi_{name}(\sigma_{temp1.country1=country.code}(\pi_{country1}(temp1) \times country))$$

- Solution 2

$$\rho(c', country)$$

$$\rho(c'', country)$$

$$\rho(b, border)$$

$$\rho(temp1, (\sigma_{c'.code=b.country1 \wedge c''.code=b.country1 \wedge c'.name='Switzerland'}(c' \times c'' \times b)))$$

$$\pi_{c''.name}(temp1)$$

RA Equivalences: Selections

- Two important equivalences involve selections:

- **Cascading of Selections:**

$$\sigma_{c1 \wedge \dots \wedge cn}(R) \equiv \sigma_{c1}(\dots \sigma_{cn}(R))$$

Allows us to combine several selections into one selection

OR: Allows us to replace a selection with several smaller selections

- **Commutation of Selections:**

$$\sigma_{c1}(\sigma_{c2}(R)) \equiv \sigma_{c2}(\sigma_{c1}(R))$$

Allows us to test selection conditions in either order

RA Equivalences: Projections

- One important equivalence involves projections:
 - **Cascading of Projections:**

$$\pi_{a_1}(R) \equiv \pi_{a_1}(\dots(\pi_{a_n}(R)))$$

This says that successively eliminating columns from a relation is equivalent to simply eliminating all but the columns retained by the final projection!

RA Equivalences: Cross-Products and Joins

- Two important equivalences involve cross-products and joins:
 - **Commutative Operations:**

$$(R \times S) \equiv (S \times R)$$

$$(R \bowtie S) \equiv (S \bowtie R)$$

This allows us to choose which relation to be the inner and which to be the outer!

RA Equivalences: Cross-Products and Joins

- Two important equivalences involve cross-products and joins:
 - **Associative Operations:**

$$R \times (S \times T) \equiv (R \times S) \times T$$

$$R \bowtie (S \bowtie T) \equiv (R \bowtie S) \bowtie T$$

It follows: $R \bowtie (S \bowtie T) \equiv (T \bowtie R) \bowtie S$

This says that regardless of the order in which the relations are considered, the final result is the same!

This *order-independence* is fundamental to how a query optimizer generates alternative query evaluation plans

RA Equivalences: Selections, Projections, Cross Products and Joins

- **Selections with Projections:**

$$\pi_a(\sigma_c(R)) \equiv \sigma_c(\pi_a(R))$$

This says we can commute a selection with a projection if the selection involves only attributes retained by the projection!

- **Selections with Cross-Products:**

$$R \bowtie_c T \equiv \sigma_c(R \times S)$$

This says we can combine a selection with a cross-product to form a join (*as per the definition of a join*)!

RA Equivalences: Selections, Projections, Cross Products and Joins

- Selections with Cross-Products and with Joins:

$$\sigma_c(R \times S) \equiv \sigma_c(R) \times S$$

$$\sigma_c(R \bowtie S) \equiv \sigma_c(R) \bowtie S$$

Caveat: The attributes mentioned in c must appear only in R and *NOT* in S

This says we can commute a selection with a cross-product or a join if the selection condition involves only attributes of one of the arguments to the cross-product or join!

RA Equivalences: Selections, Projections, Cross Products and Joins

- Selections with Cross-Products and with Joins (Cont'd):

$$\begin{aligned}\sigma_c(R \times S) &\equiv \sigma_{c_1 \wedge c_2 \wedge c_3}(R \times S) \\ &\equiv \sigma_{c_1}(\sigma_{c_2}(\sigma_{c_3}(R \times S))) \\ &\equiv \sigma_{c_1}(\sigma_{c_2}(R) \times \sigma_{c_3}(S))\end{aligned}$$

This says we can push part of the selection condition c ahead of the cross-product!

This applies to joins as well!

RA Equivalences: Selections, Projections, Cross Products and Joins

- Projections with Cross-Products and with Joins:

$$\pi_a(R \times S) \equiv \pi_{a1}(R) \times \pi_{a2}(S)$$

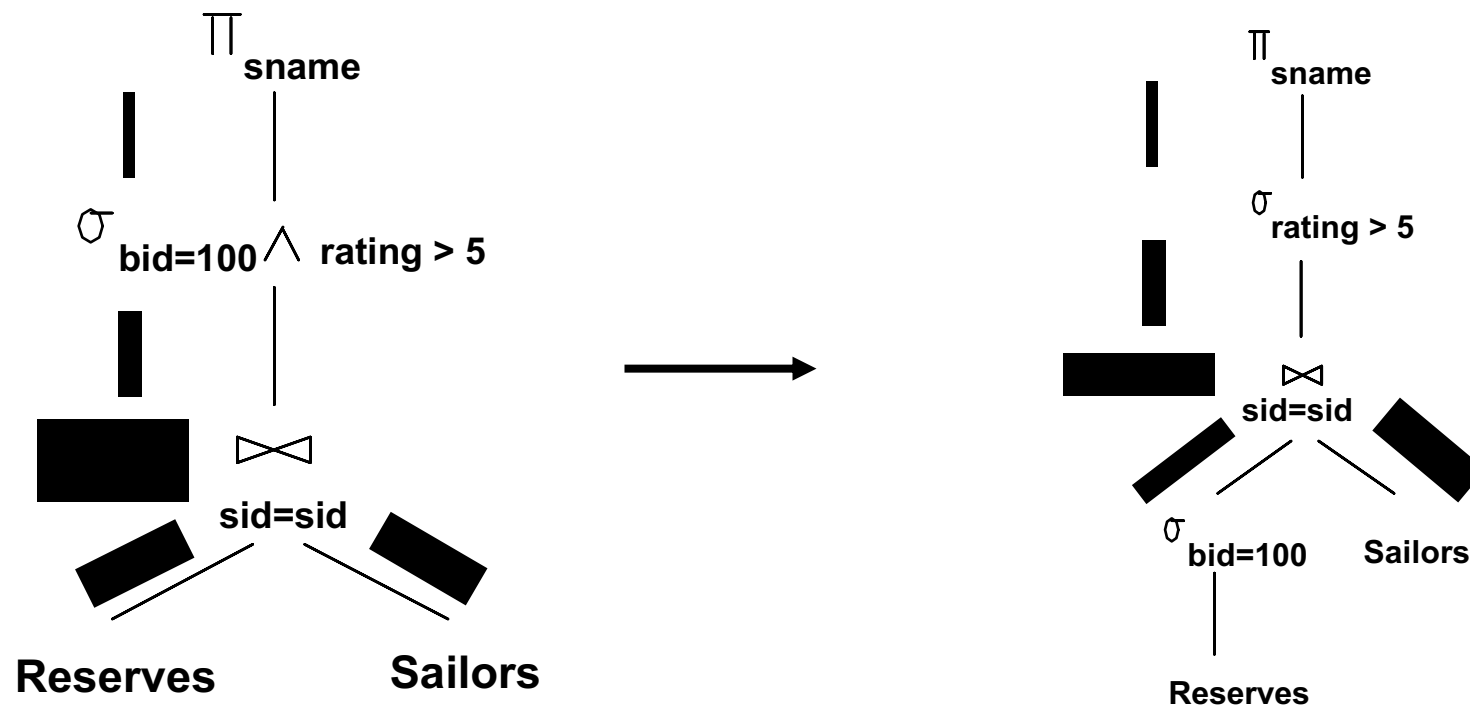
$$\pi_a(R \bowtie_c S) \equiv \pi_{a1}(R) \bowtie_c \pi_{a2}(S)$$

$$\pi_a(R \bowtie_c S) \equiv \pi_a(\pi_{a1}(R) \bowtie_c \pi_{a2}(S))$$

Intuitively, we need to retain only those attributes of R and S that are either mentioned in the join condition **c** or included in the set of attributes **a** retained by the projection

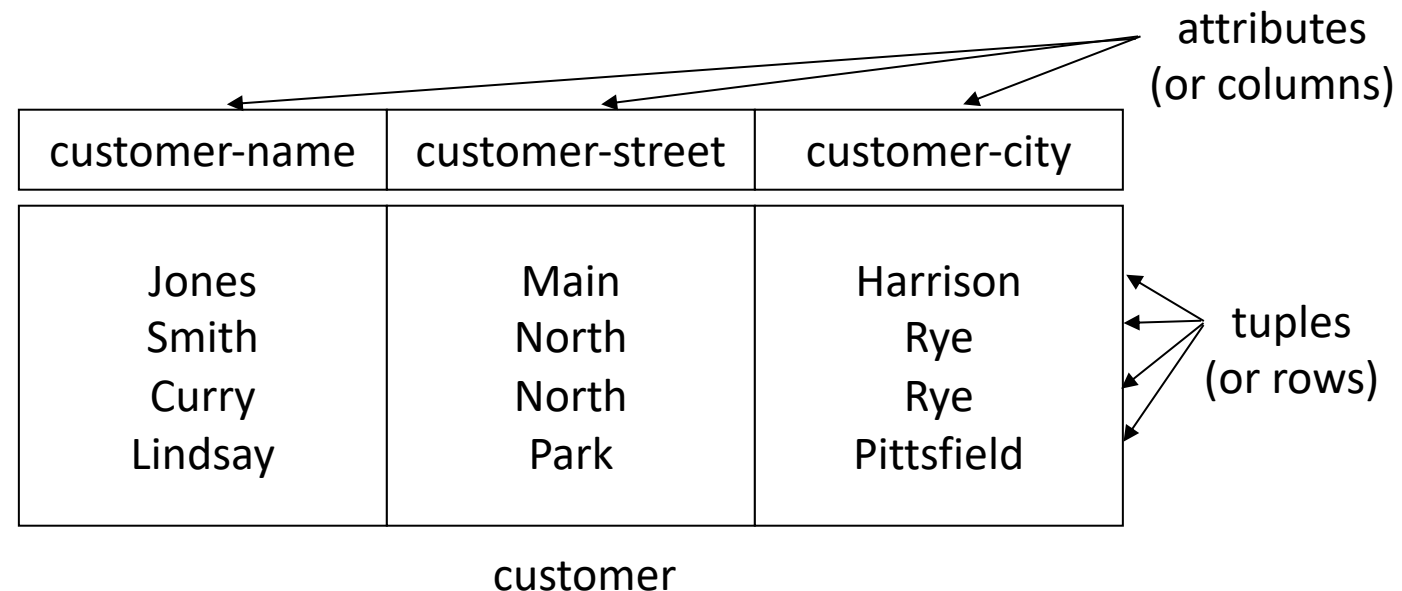
How to Estimate the Cost of Plans?

Now that correctness is ensured, how can the DBMS estimate the costs of various plans?



Relational Calculus

- Also known as *predicate calculus*, or *first order logic*
- The current values (*relation instance*) of a relation are specified by a table
- An element t of r is a *tuple*, represented by a *row* in a table



customer-name	customer-street	customer-city
Jones	Main	Harrison
Smith	North	Rye
Curry	North	Rye
Lindsay	Park	Pittsfield

customer

Relational Calculus

- Comes in two flavors: Tuple relational calculus (TRC) and Domain relational calculus (DRC).
- Expressions in the calculus are called *formulas*. An answer tuple is essentially an assignment of constants to variables that make the formula evaluate to *true*.
- **Relational Calculus** is essentially first-order logic over the schema relations
 - A query has the form “**find all tuples (x_1, \dots, x_n) that satisfy a given condition**”

Tuple Relational Calculus

- **Query** has the form: $\{ t \mid p(t) \}$

Answer includes all tuples t that make the *formula* $p(t)$ be *true*.

Formula is recursively defined, starting with simple *atomic formulas* (getting tuples from relations or making comparisons of values), and building bigger and better formulas using the *logical connectives*.

Find all sailors with a rating above 7

- $\{S \mid S \in \text{Sailors} \wedge S.\text{rating} > 7\}$

All rows S such that S is a Sailor and S.rating > 7

Sid	Sname	Rating	Age
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

Find all sailors with a rating above 7

- $\{S \mid S \in \text{Sailors} \wedge S.\text{rating} > 7\}$
- Query is evaluated on an instance of **Sailors**
- Tuple variable S is instantiated to each tuple of this instance in turn, and the condition “ **$S.\text{rating} > 7$** ” is applied to each such tuple.
- Answer contains all **instances of S** (which are tuples of Sailors) satisfying the condition.

TRC Formulas

- **Atomic formula:**

- $R \in \text{Rel}$, or $R.a \text{ op } S.b$, or $R.a \text{ op constant}$
- op is one of $<, >, =, \leq, \geq, \neq$

- **Quantifiers:**

- $\exists X(p(X))$, there exists X such that $p(X)$ is true (Existential)
- $\forall X(p(X))$, for all x , $p(X)$ is true (Universal)

- **Formula:**

- an atomic formula, or
- $\neg p, p \wedge q, p \vee q$, where p and q are formulas, or
- $\exists X(p(X))$, where variable X is **free** in $p(X)$, or
- $\forall X(p(X))$, where variable X is **free** in $p(X)$

Free and Bound Variables

- A variable v is **bound** in a predicate p when p is of the form $\forall v...$ or $\exists v...$
- A variable occurs **free** in p if it occurs in a position where it is not bound by an enclosing \forall or \exists
- Examples:
 - x is free in $x > 2$
 - x is bound in $\exists x. x > y$

Free and Bound Variables

- The use of **quantifiers** $\forall X$ and $\exists X$ in a formula is said to bind X .
 - A variable that is **not bound** is free.
- Let us revisit the definition of a **query**: $\{t \mid p(t)\}$

There is an important restriction: the variable **t** that appears to the left of `|' must be the **only** free variable in the formula $p(\dots)$.

Example Queries

Relation \rightarrow loan (loan_number, bank-name, amount)

Find the loan_number, branch_name, and amount for **loans** of over \$1200

$\{t \mid t \in \text{loan} \wedge t.\text{amount} > 1200\}$ Relational Calculus

$\sigma_{\text{amount} > 1200}(\text{loan})$ Relational Algebra

Example Queries

Find the loan_number, branch_name, and amount for loans of over \$1200

$$\{t \mid t \in \text{loan} \wedge t.\text{amount} > 1200\}$$

Find the loan number for each loan of an amount greater than \$1200

$$\{t \mid \exists s \in \text{loan} (t.\text{loan_number} = s.\text{loan_number} \wedge s.\text{amount} > 1200)\} \rightarrow \text{RC}$$

$$\pi_{\text{loan_number}}(\sigma_{\text{amount} > 1200}(\text{loan})) \rightarrow \text{RA}$$

Notice that a relation on schema $[\text{loan_number}]$ is implicitly defined by the RC query

Find sailors rated > 7 who've reserved boat #103

- $\{S \mid (S \in \text{Sailors}) \wedge (S.\text{rating} > 7) \wedge (\exists R \in \text{Reserves} (R.\text{sid} = S.\text{sid} \wedge R.\text{bid} = 103))\}$
- Note the use of \exists to find a tuple in Reserves that 'joins with' the Sailors tuple under consideration.
- R is **bound**, S is not

Find the ID, name, dept_name, salary for
instructors whose salary is greater than 80,000

Relation \rightarrow instructor(*ID, name, dept_name, salary*)

$\{t \mid t \in \text{instructor} \wedge t.\text{salary} > 80000\}$

Notice that a relation on schema (*ID, name, dept_name, salary*) is implicitly
defined by the query

Find the ID, name, dept_name, salary for
instructors whose salary is greater than 80,000

Relation \rightarrow instructor(*ID*, *name*, *dept_name*, *salary*)

$\{t \mid t \in \text{instructor} \wedge t.\text{salary} > 80000\}$

Notice that a relation on schema (*ID*, *name*, *dept_name*, *salary*)
is implicitly defined by the query

As in the previous query, but output only the *ID* attribute value

$\{t \mid \exists s \in \text{instructor} (t.ID = s.ID \wedge s.salary > 80000)\}$

Notice that a relation on schema (*ID*) is implicitly defined
by the query

Find the names of all instructors whose department is in the Watson building

Relations:

instructor(ID, name, dept_name, salary)

department(dept_name, building)

$$\{t \mid \exists s \in instructor (t.name = s.name \wedge \exists u \in department (u.dept_name = s.dept_name \wedge u.building = \text{"Watson"}))\}$$

Find the set of all courses taught in the Fall 2009 semester, or in the Spring 2010 semester, or both

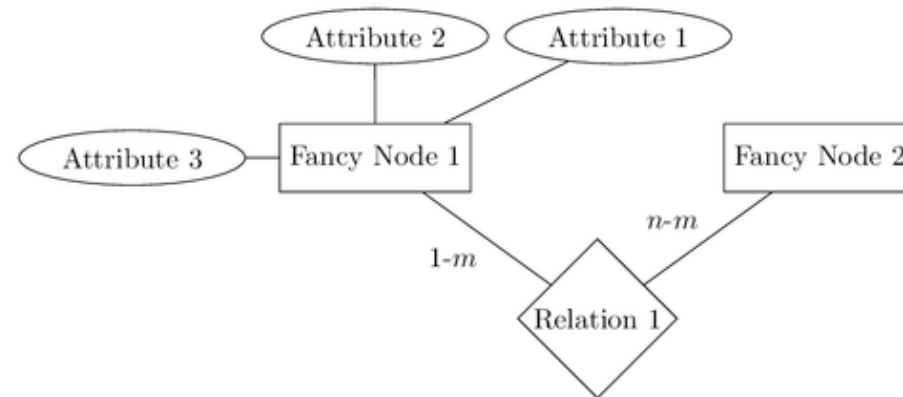
$$\{t \mid \exists s \in \text{section} (t.\text{course_id} = s.\text{course_id} \wedge s.\text{semester} = \text{"Fall"} \wedge s.\text{year} = 2009)$$


Fall and
2009

$$\vee \exists u \in \text{section} (t.\text{course_id} = u.\text{course_id} \wedge u.\text{semester} = \text{"Spring"} \wedge u.\text{year} = 2010))\}$$


Spring
and 2010

Entity Relationship Models



Design Phases

- **Initial phase** → Analyze and characterize your data needs
- **Second phase** → Choosing a data model to translate the requirements into a conceptual schema of the database
- **Final phase** → Moving from an abstract data model to the implementation of the database

Design Approaches

Entity Relationship Model

- Models an enterprise as a collection of **entities** and **relationships**
- **Entity**: a “thing” or “object” in the enterprise that is distinguishable from other objects
 - Described by a set of **attributes**
- **Relationship**: an association among several entities

Represented diagrammatically by an entity-relationship diagram

A bit of History...

The Entity-Relationship Model--Toward a Unified View of Data (ACM Transactions on Database Systems, 1976).

This paper is one of the most cited papers in the computer field, and has been considered one of the most influential papers in computer science.



Peter Chen

Google Scholar

The Entity-Relationship Model--Toward a Unified View of Data

Articles

About 6,860 results (0.12 sec)

Any time

Since 2019

Since 2018

Since 2015

Custom range...

The entity-relationship model—toward a unified view of data

PPS Chen - ACM Transactions on Database Systems (TODS), 1976 - dl.acm.org

A data model, called the entity-relationship model, is proposed. This model incorporates some of the important semantic information about the real world. A special diagrammatic technique is introduced as a tool for database design. An example of database design and ...

☆ 11522 Cited by 11522 Related articles All 74 versions

Design Alternatives

- In designing a database schema, we must ensure that we avoid two major pitfalls:
 - **Redundancy**: a bad design may result in repeat information.
 - Redundant representation of information may lead to data inconsistency among the various copies of information
 - **Incompleteness**: a bad design may make certain aspects of the enterprise difficult or impossible to model.
- **Avoiding bad designs is not enough**. There may be a large number of good designs from which we must choose.

Database Design

- **Goal:** specification of database schema
- **Methodology:**
 - Use ER model to get a high-level graphical view of essential components of enterprise and how they are related
 - Convert E-R diagram to DDL
- ER Model: enterprise viewed as set of
 - Entities
 - Relationships among entities

ER model -- Database Modeling

- The ER data model was developed to **facilitate database design** by allowing specification of an **enterprise schema** that represents the overall **logical structure of a database**.
- The ER data model employs three basic concepts:
 - entity sets,
 - relationship sets,
 - attributes.
- The ER model also has an associated diagrammatic representation, the ER diagram, which can express the **overall logical structure** of a database graphically.

Entities

- Entity: an object that is involved in the enterprise
 - Ex: John, CSE305
- Entity Type: set of similar objects
 - Ex: students, courses
- Attribute: describes one aspect of an entity type
 - Ex: name, maximum enrollment

Entities are the people, places, things, or events that are of interest for a system that we are planning to build.

Entity Type

- Entity type described by set of attributes
 - Student: Id, Name, Address, Hobbies
- Domain: possible values of an attribute
 - Value can be a set (in contrast to relational model)
 - (111111, John, 123 Main St, (stamps, coins))
- Key: minimum set of attributes that uniquely identifies an entity (candidate key)
- Entity Schema: entity type name, attributes (and associated domain), key constraints

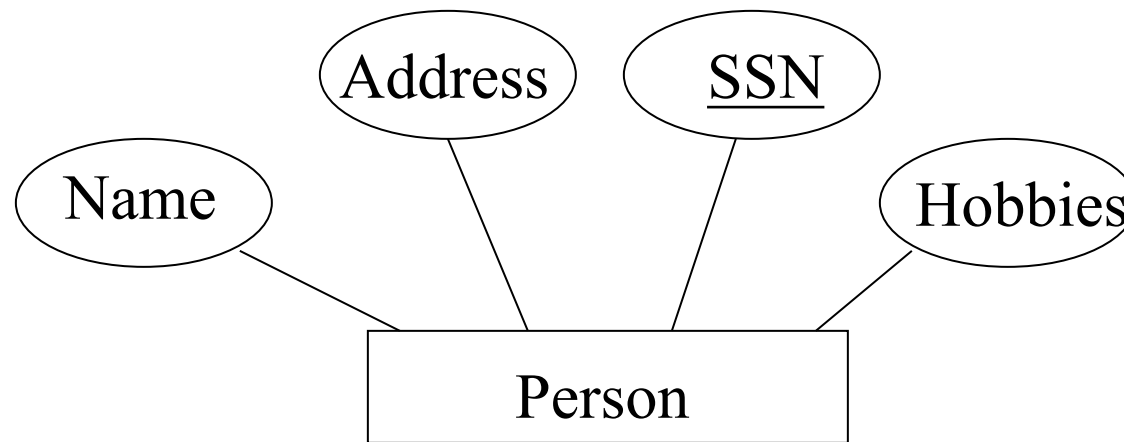
Attributes

- Attributes are the characteristics that describe entities
- A Student entity may be described by attributes including:
 - student number
 - name
 - Address
 - date of birth
 - degree major

Person Entity (Example)

We illustrate attributes using ovals.

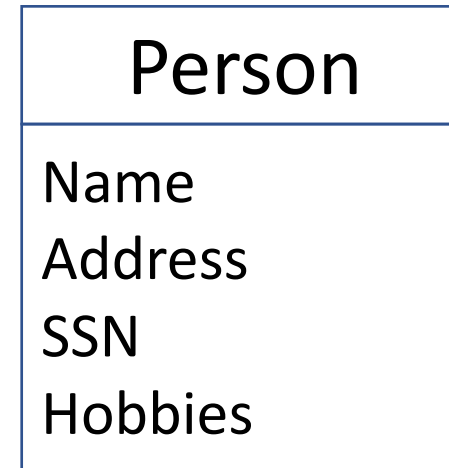
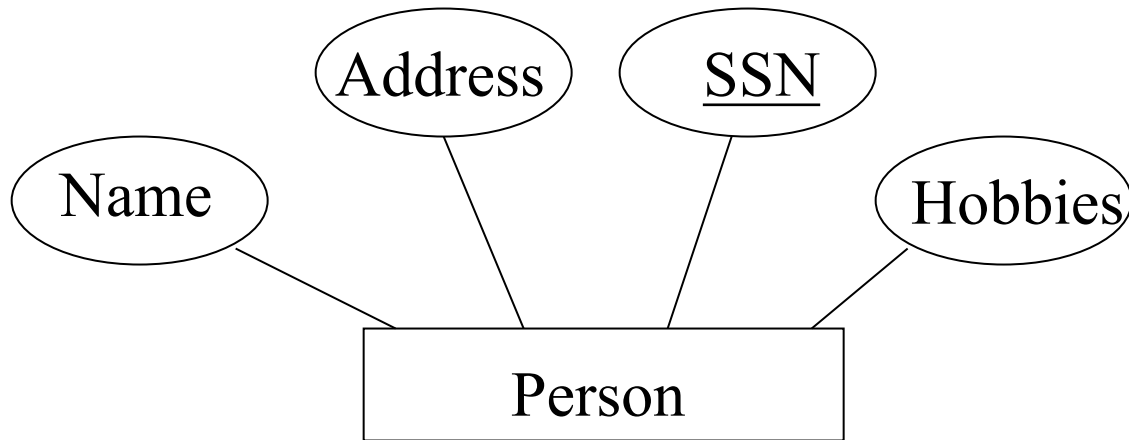
- Graphical Representation in ER diagram:



We use a rectangular symbol to represent an entity set in the Entity Relationship Diagram.

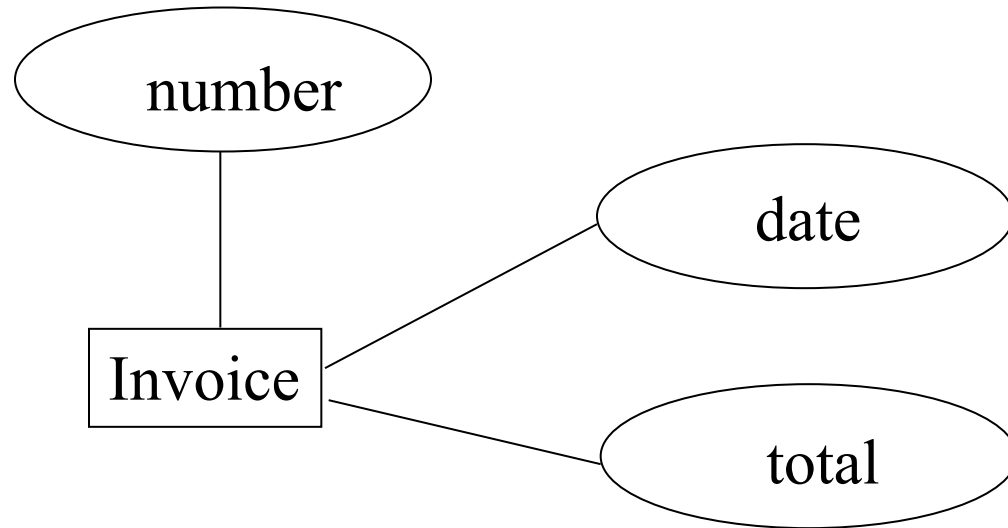
Person Entity (Example)

- Graphical Representation in ER diagram:



Alternative Notation

Invoice Entity (Example)



An Invoice entity may be described by attributes including:

- invoice number
- invoice date
- invoice total

Representation in Relational Model

- Entity type corresponds to a relation
- Relation's attributes = entity type's attributes

Attributes are the characteristics that describe entities

Representation in Relational Model

- Entity type corresponds to a relation
- Relation's attributes = entity type's attributes
 - Problem: entity type can have set valued attributes.



shutterstock.com • 344347193

Two phone numbers

Attributes are the characteristics that describe entities

Representation in Relational Model

- Entity type corresponds to a relation
- Relation's attributes = entity type's attributes
 - Problem: entity type can have set valued attributes.
 - Solution: Use several rows to represent a single entity
 - (111111, John, 123 Main St, 123456)
 - (111111, John, 123 Main St, 789134)

Representation in Relational Model

- Entity type corresponds to a relation
- Relation's attributes = entity type's attributes
 - Problem: entity type can have set valued attributes.
 - Solution: Use several rows to represent a single entity
 - (111111, John, 123 Main St, 123456)
 - (111111, John, 123 Main St, 789134)
 - Problems with solution:
 - Redundancy
 - Key of entity type not key of relation
 - => resulting relation must be further transformed

Attributes are the characteristics that describe entities

Attribute Classification

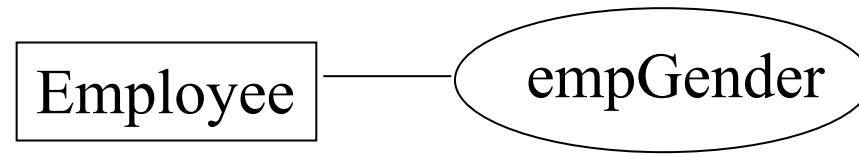
- Atomic Attributes
- Single-valued Attributes
- Multi-valued Attributes
- Key Attributes
- Partial Keys
- Surrogate Key
- Non-key Attributes
- Derived Attributes

Atomic Attributes

- A simple, or atomic, attribute is one that cannot be decomposed into meaningful components
- **Example:** an attribute for product price, prodPrice, cannot be decomposed, because you cannot subdivide prodPrice into a finer set of meaningful attributes. The value of the attribute prodPrice could be \$21.03.
- Of course, one could decompose prodPrice into two attributes: the dollar component (21), and the cents component (03)
- but our assumption here is that such a decomposition is not meaningful to the intended application or system that will make use of it.

Single-valued (or atomic) Attributes

An attribute is single-valued if, for any instance of the pertinent entity set, there is only one value at a given time for the attribute.



Multi-valued Attributes

- Consider an attribute to track each employee's university degrees - empDegree.
- Since an employee could have none, one, or several degrees, we say empDegree is multi-valued.
- sample data for three employees.

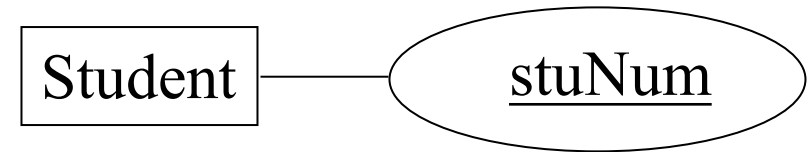
empNum	empPhone	empDegree
123	233-9876	
333	233-1231	BA, BSc, PhD
679	233-1231	BSc, MSc

One of these employees has no degrees, another has 3 degrees, and the last one has 2 degrees.

Key Attributes

- A key is an attribute, or combination of attributes, where the attribute value identifies an individual entity
- Suppose an educational institution assigns students individual student numbers in such a way that each student's number is different from that assigned to any other student
- Student numbers are *unique*, each student has a different number.

In an ER keys are shown underlined



For every entity set, we want to specify which attribute is a key attribute. In this case, Student is shown to have a key named stuNum.