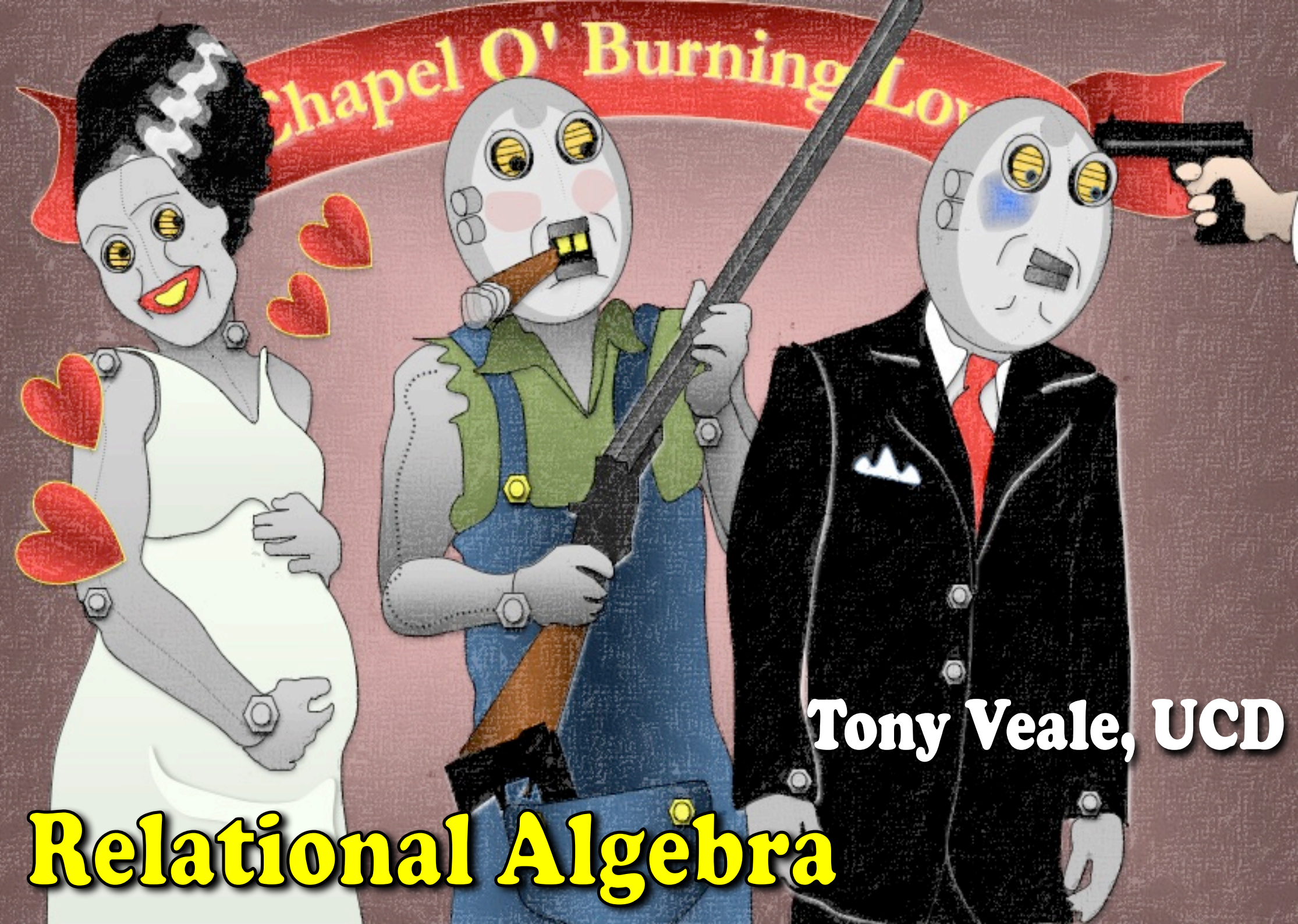

Chapel O' Burning Lo
Tony Veale, UCD
Relational Algebra

Relational Algebra offers a procedural basis for thinking about DB querying

Select operation

Project operation

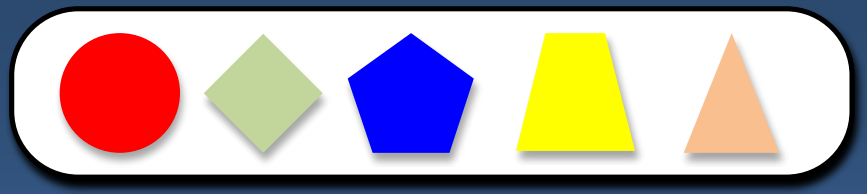
Union operation

Set difference

Cartesian product

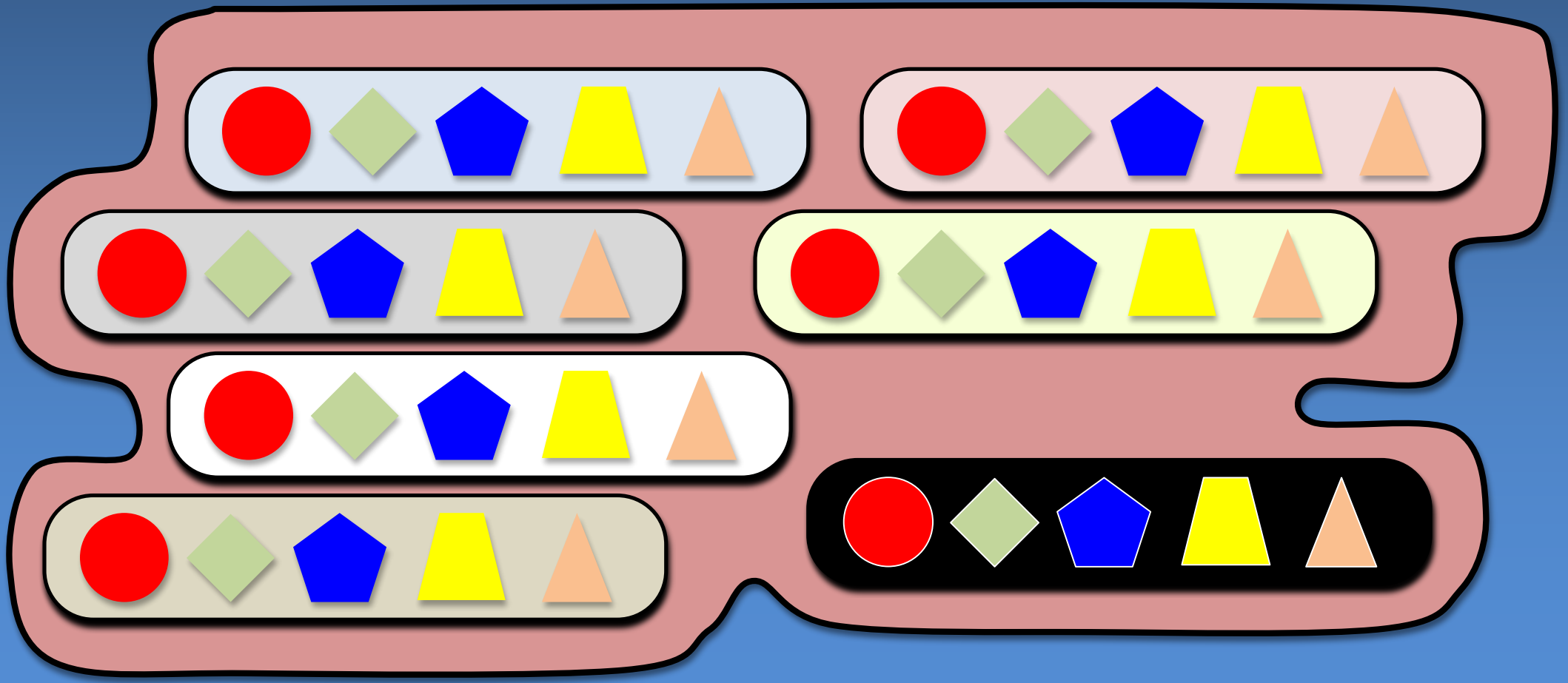
Rename as

A **tuple** is a structured grouping of associations:



(One tuple per entity)

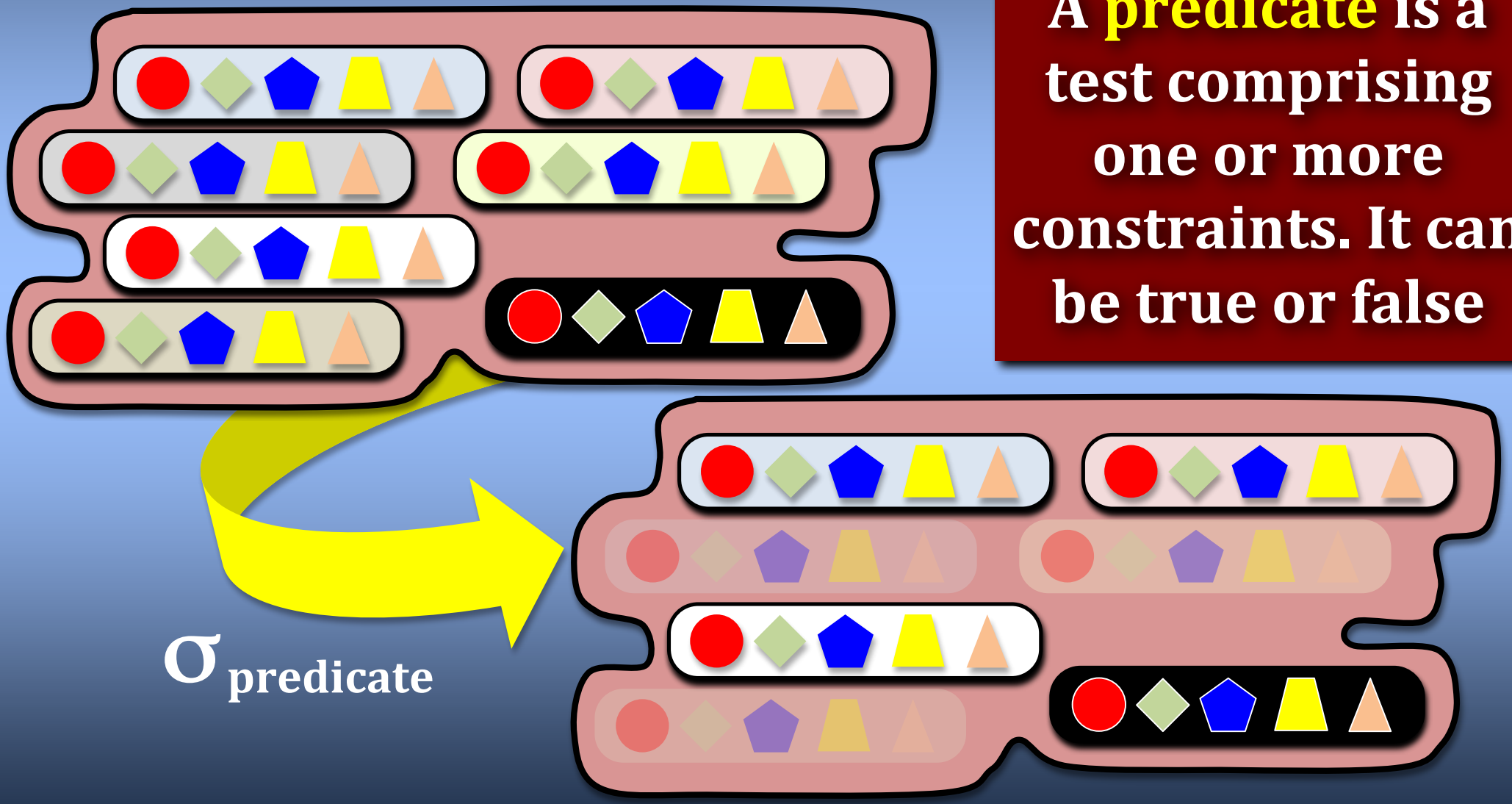
A **relation** is an exhaustive collection of tuples:

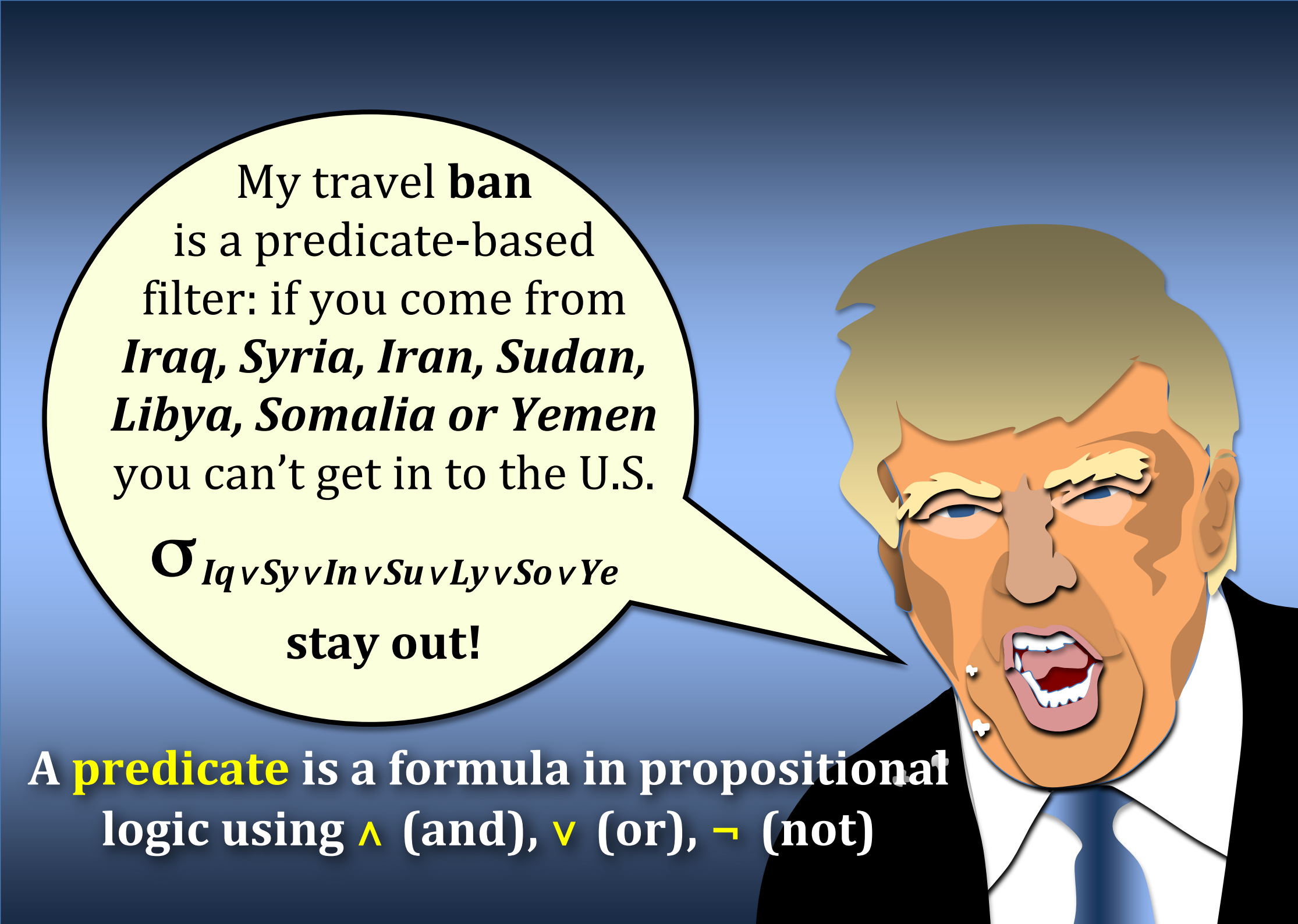


Select Operation $\sigma_{predicate}$

Select only tuples that satisfy a given test or predicate

A predicate is a test comprising one or more constraints. It can be true or false




My travel **ban**
is a predicate-based
filter: if you come from
Iraq, Syria, Iran, Sudan,
Libya, Somalia or Yemen
you can't get in to the U.S.
$\sigma_{Iq \vee Sy \vee In \vee Su \vee Ly \vee So \vee Ye}$
stay out!
A **predicate** is a formula in propositional
logic using ∧ (and), ∨ (or), ¬ (not)

$\sigma_{Fictive="yes"}(Entities)$ selects fictional entities

$\sigma_{Gender="male"}(Entities)$ selects male entities

$\sigma_{Gender="male" \wedge Fictive="yes"}(Entities)$ selects fictional male entities

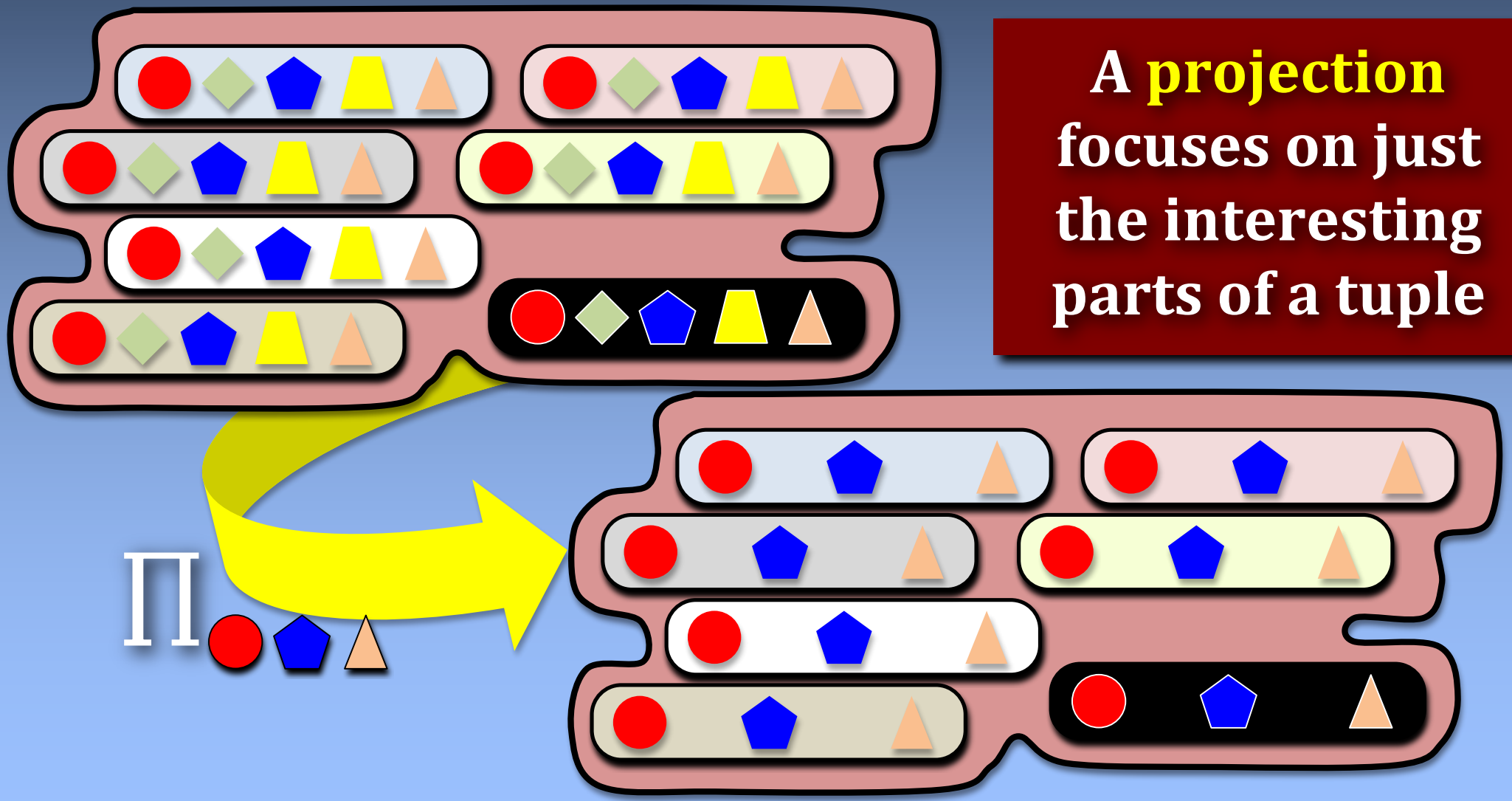
Entities (relation)

Name	Gender	Fictive	Opponent
Adam Smith	*male*	*no*	*Karl Marx*
Princess Leia	*female*	*yes*	*Darth Vader*
Hillary Clinton	*female*	*no*	*Donald Trump*
Al Capone	*male*	*no*	*Eliot Ness*

The output is a selection of matching tuples

Project Operation $\Pi_{\text{associations}}(r)$

*projects only **associations** from relation **r** into new tuple set*



A projection focuses on just the interesting parts of a tuple

$$\Pi_{\text{Name,Gender,Opponent}}(\text{Entities})$$

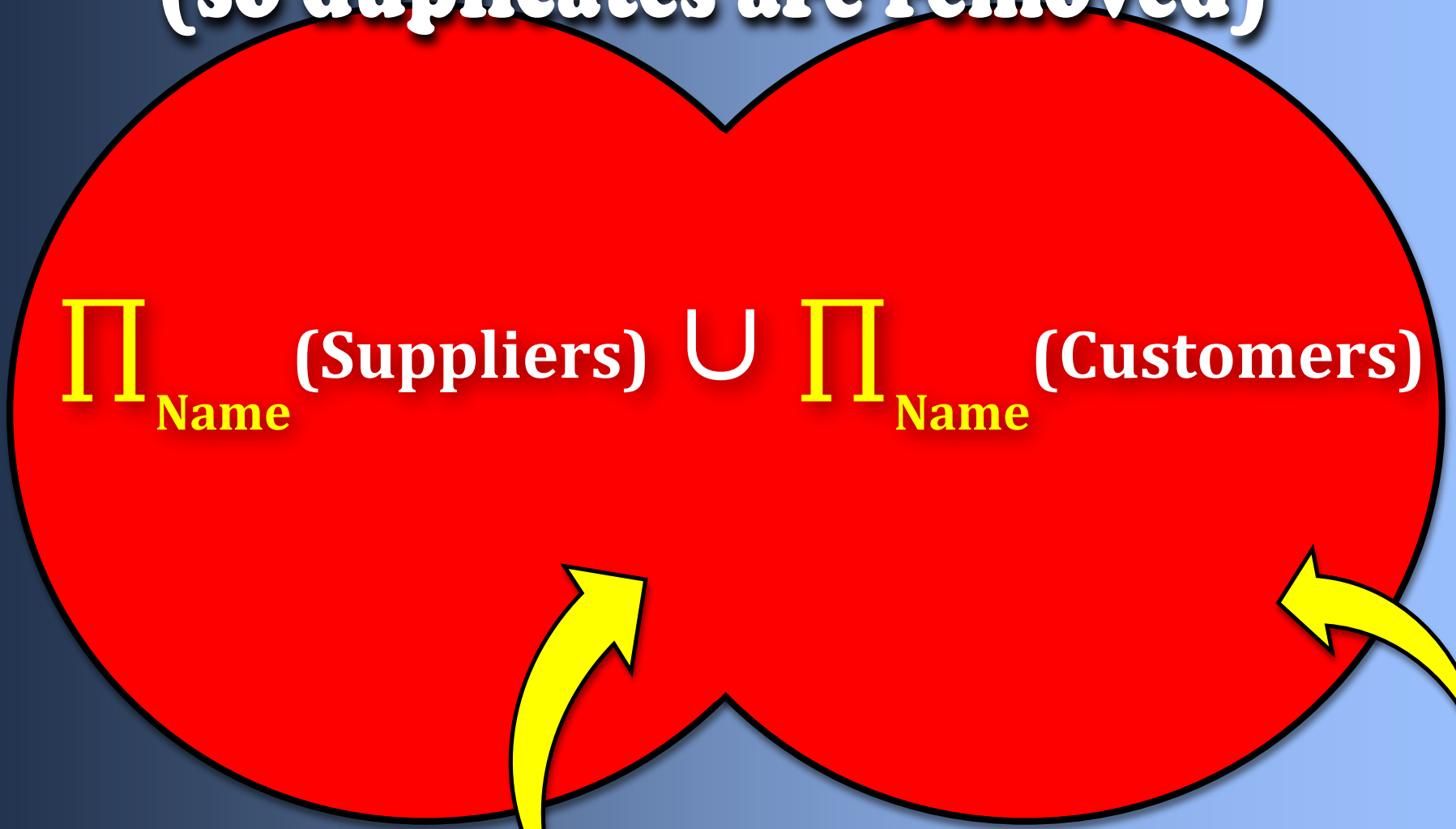
Entities (minus Fictive)

Name	Gender	Opponent
Adam Smith	*male*	*Karl Marx*
Princess Leia	*female*	*Darth Vader*
Hillary Clinton	*female*	*Donald Trump*
Al Capone	*male*	*Eliot Ness*

Output is all tuples minus unprojected associations

Set Union of two relations

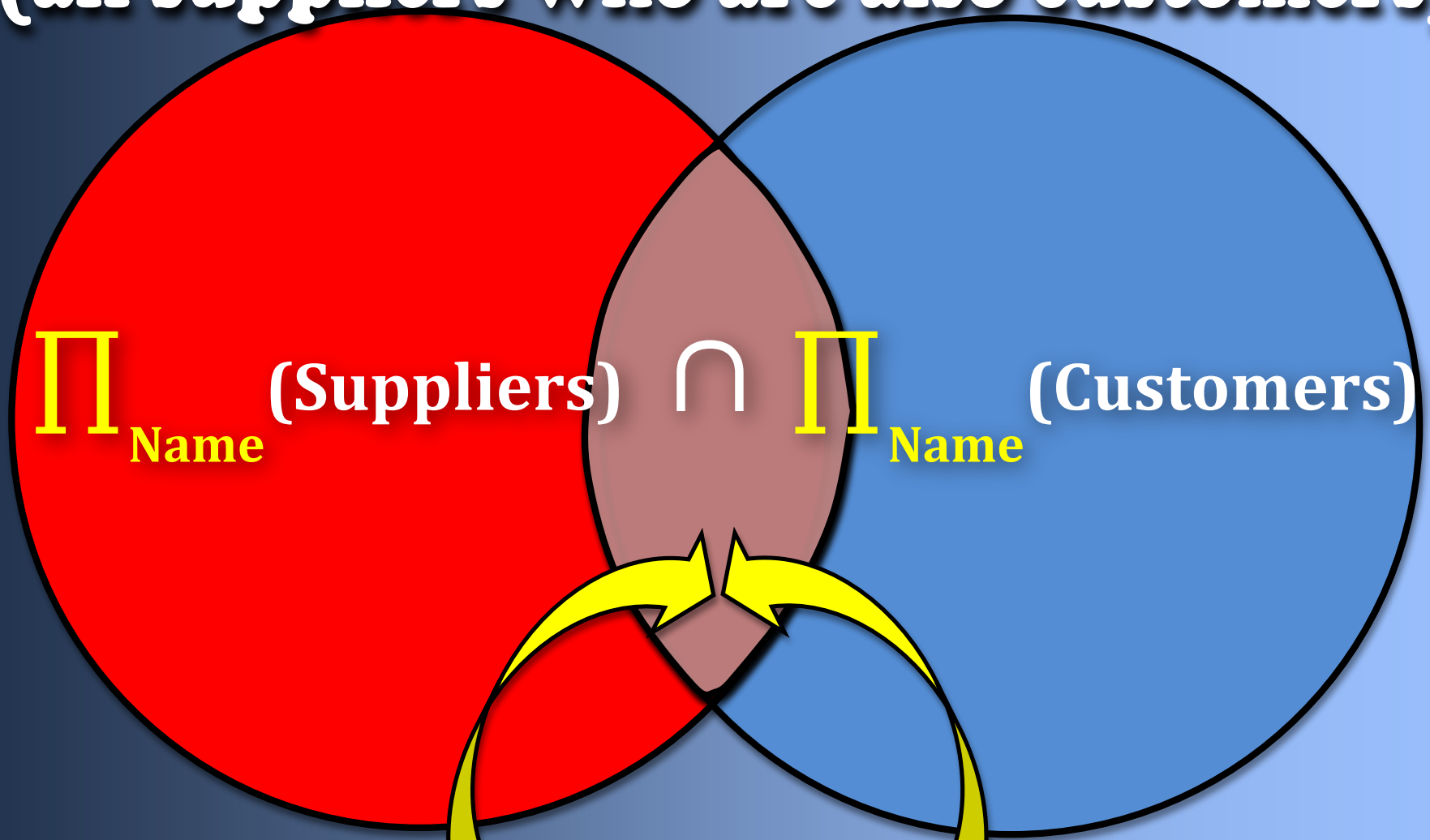
(so duplicates are removed)



The new relation contains every Customer and Supplier

SET Intersection of Two Relations

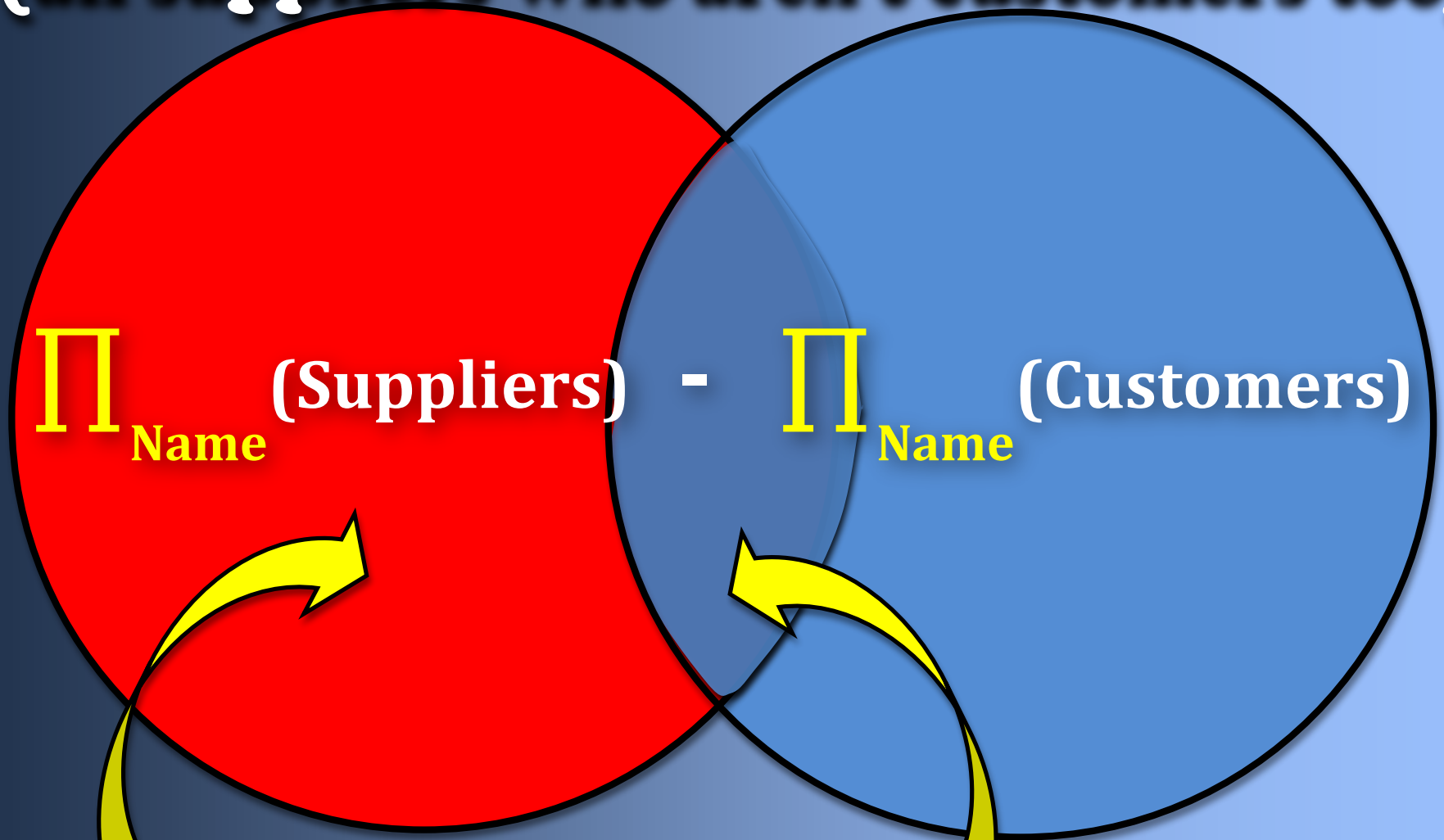
(all suppliers who are also customers)



New relation contains only the tuples in both relations

SET Difference of Two Relations

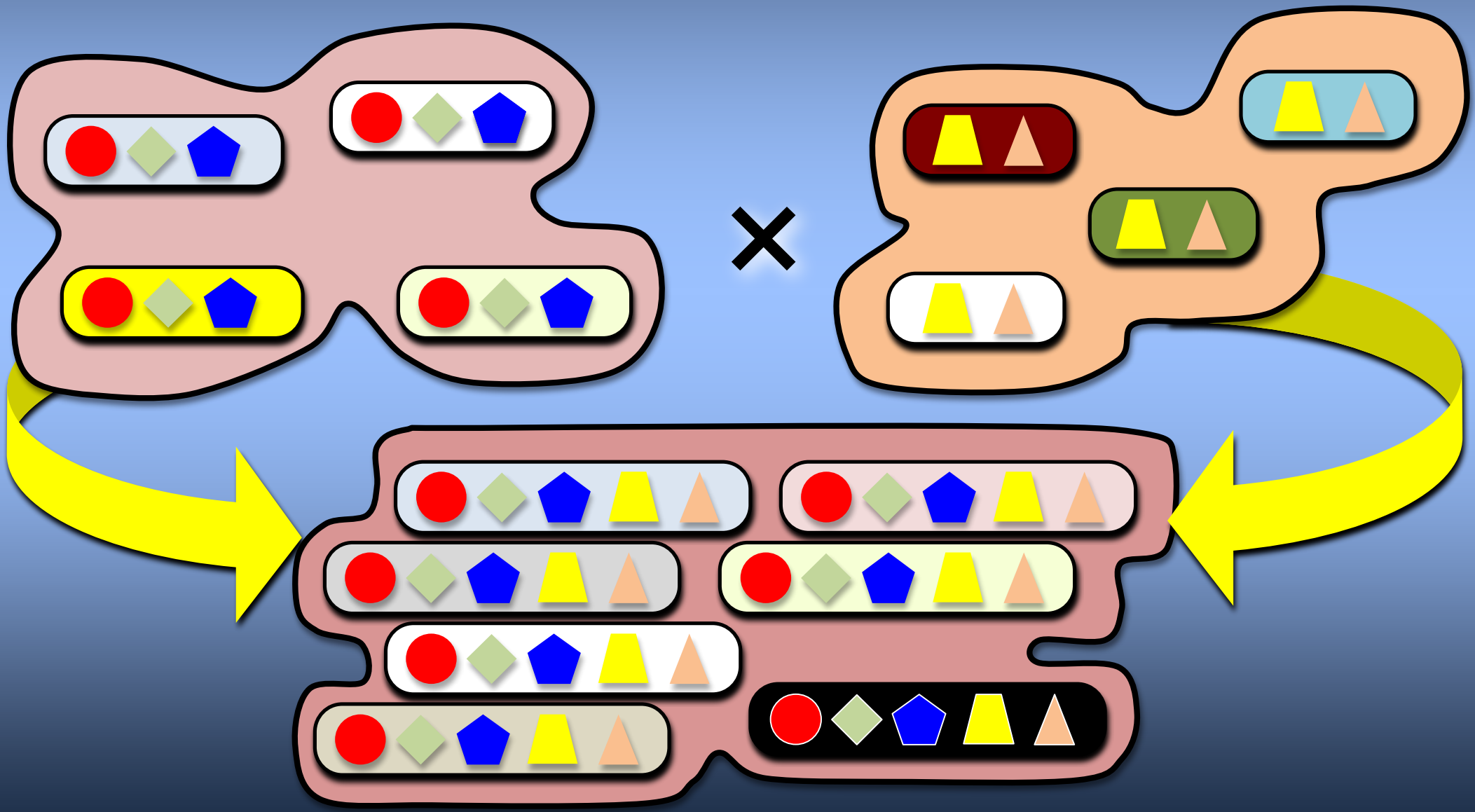
(all suppliers who aren't customers too)



All of Left relation minus overlap with Right

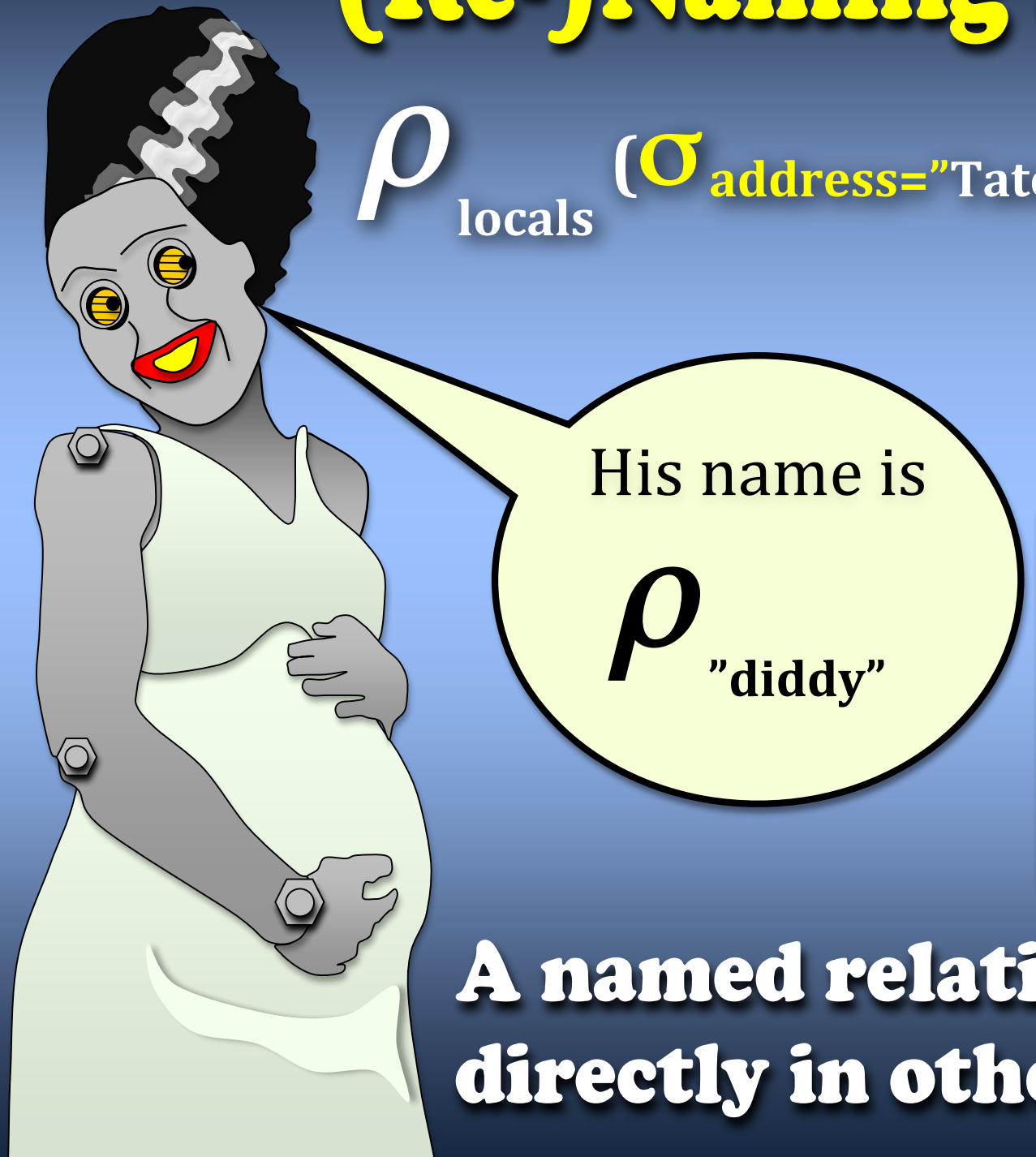
Cartesian Product of Two Relations

E.g., $\sigma_{\text{order\_id=”ORD3753”}}$(Customers x Orders)



(Re-)Naming a Relation

$$\rho_{locals}(\sigma_{address=\text{"Tatooine"}}(Customers))$$

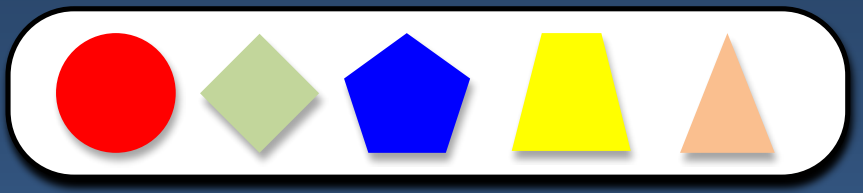


Here we name a selected subset of the Customers relation as Locals

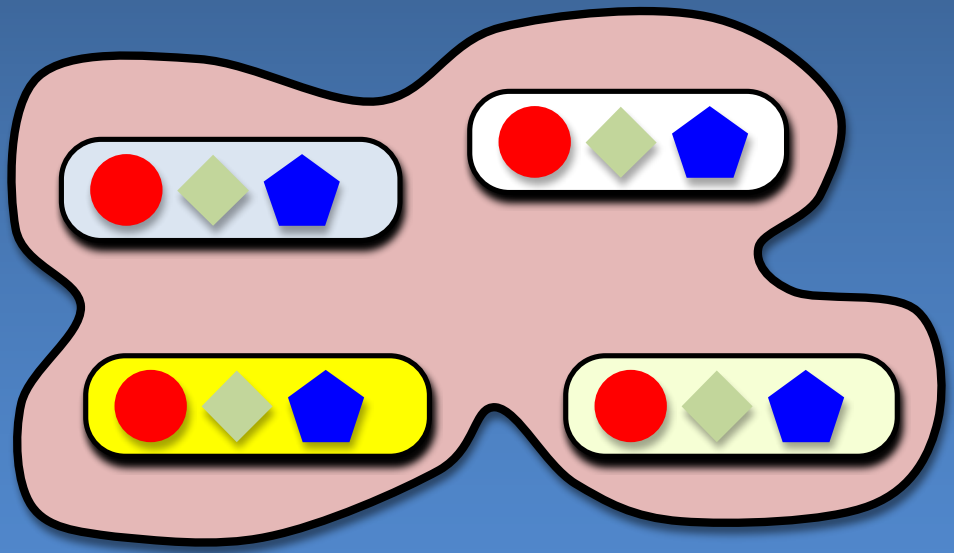
A named relation can be used directly in other formulations

A DB relates specific **entities** to specific **relations**

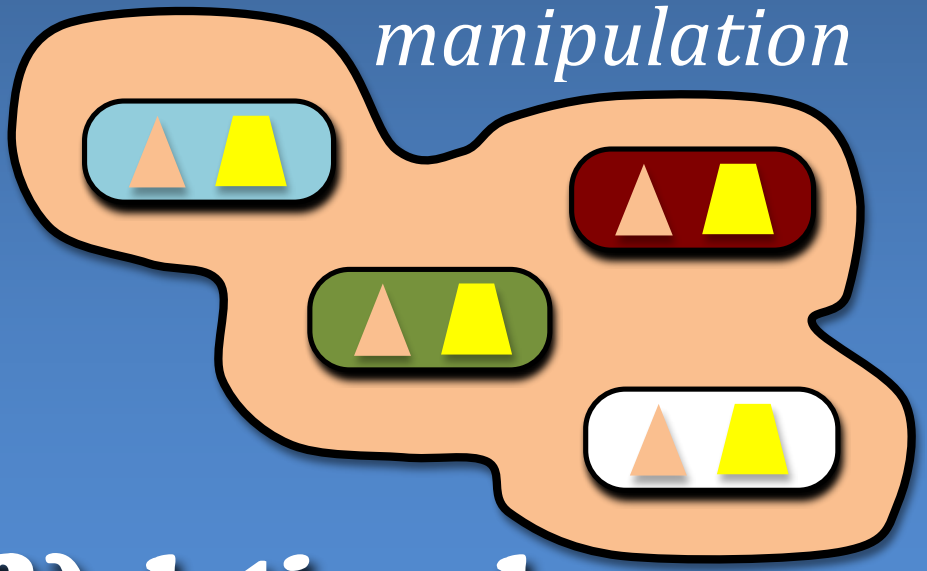
whether in tuples ...



Or in sets of tuples ...



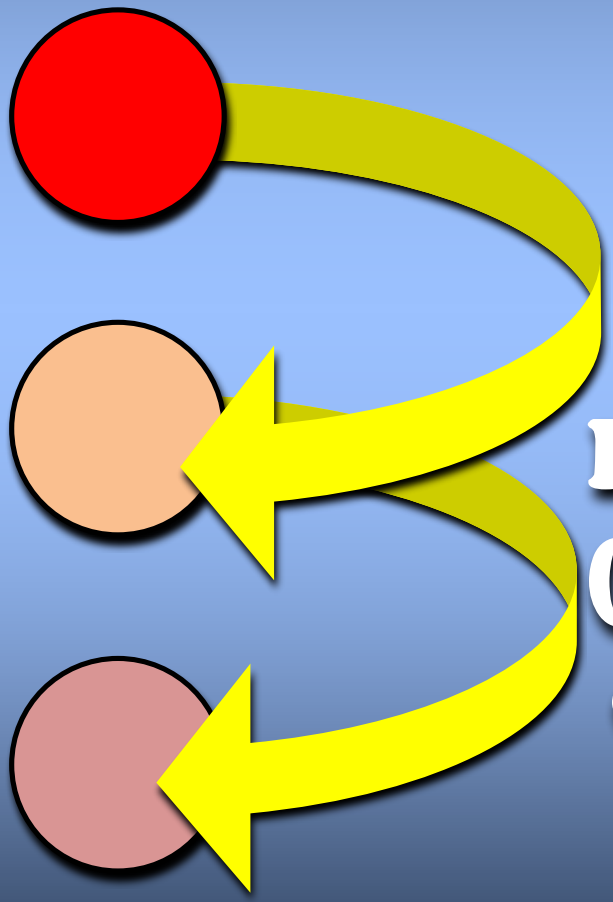
... which are subject to set-theoretic manipulation



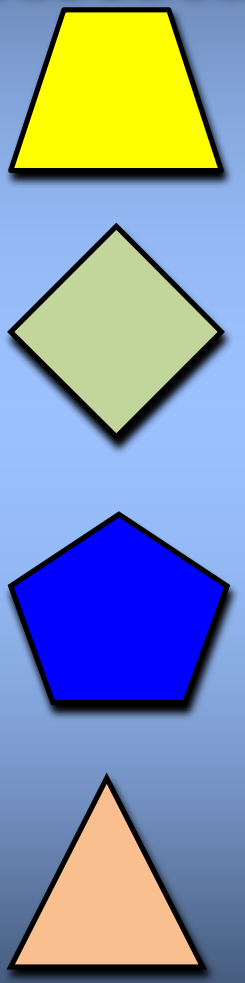
We model the (E)entity-(R)elational structure of a DB with ER-graphs

Entities (things in the world)

Attributes (properties of entities)



Relations (between entities)



Simple Attributes (indivisible properties)

For example,

- Student IDs
- Telephone Numbers
- ZIP Codes
- First Names
- Surnames
- Order Numbers

Composite Attributes
(Multiple Parts combined)

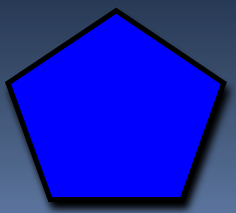
For example,

Customer Name (First and Last Names)

Address (Street, Number, City, Country)

Order (Item, Quantity, Price, Delivery Options)

Derived Attributes (Functions of other Attributes)



For example,

MAX(Salary) (Salary of best-paid entity)

MIN(Age) (Age of youngest entity)

AVG(Hours) (Mean hours worked per entity)

Single-Value Attributes
(Just one value per entity but may be composite)

For example,

Salary (Just one salary per entity)

Tax Number (Just one tax number each)

Date-Of-Birth (Everybody is born ONCE)

Multi-Value Attributes
(Multiple values per entity)

For example,

Email address (Possibly many per entity)

Phone Number (Home, Work, Mobile, etc.)

Department (Work for several bosses)

Super-Keys

A **Super-Key** is a set of 1 or more attributes that uniquely identifies every entity in an entity set.

A **Minimal Super-Key** is one that does NOT contain a smaller Super-Key as a subset.
This is a **Candidate Key** → **Primary Key**

A relationship is an association between specific entities. E.g, *married_to* is a relationship between two people.

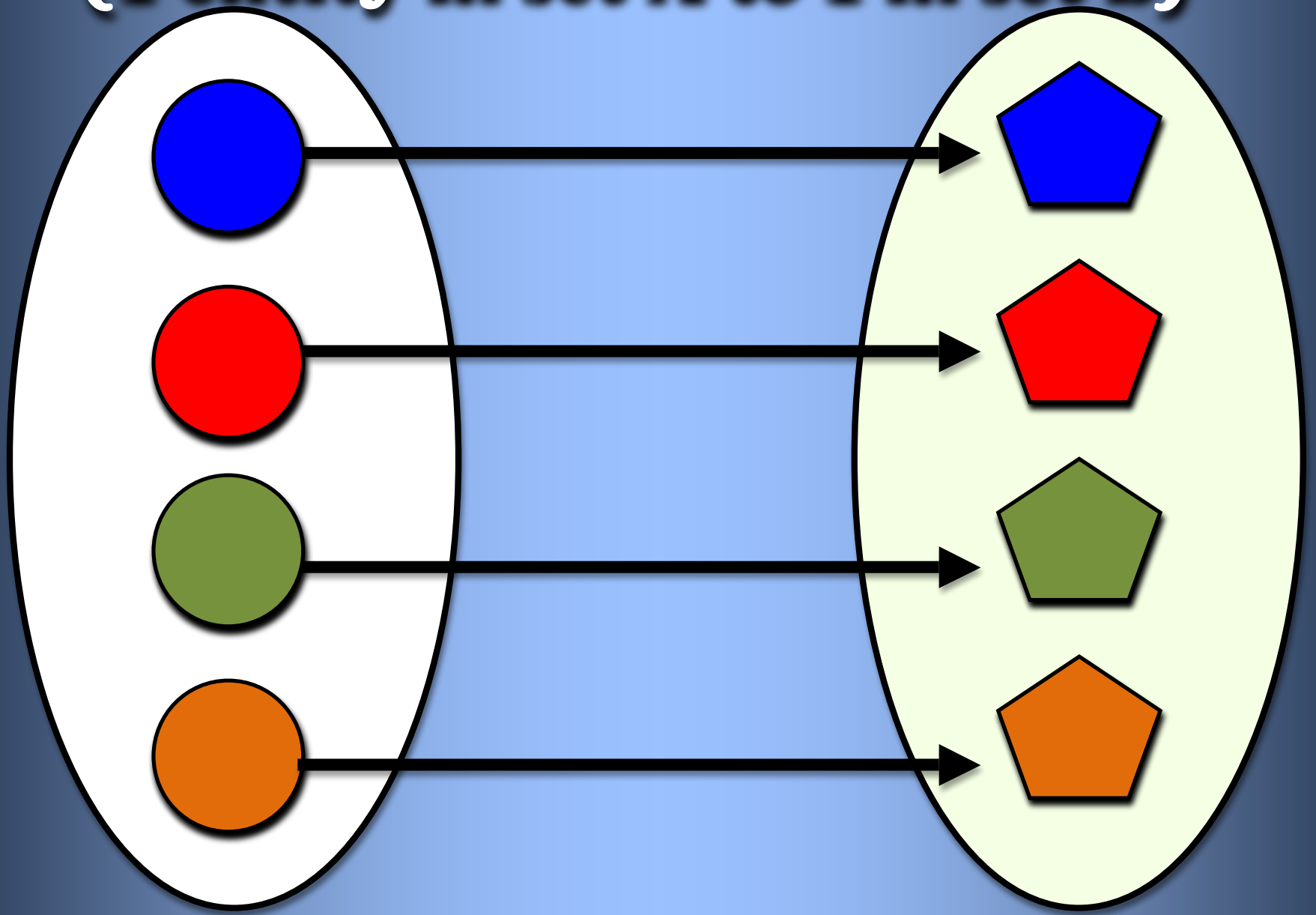
A relationship set is a set of relationships of the same type. E.g. *Marriage* can be defined as the set of all married_to instances.

Monogamous marriage is a ***binary*** relationship between two entities (husband and wife).

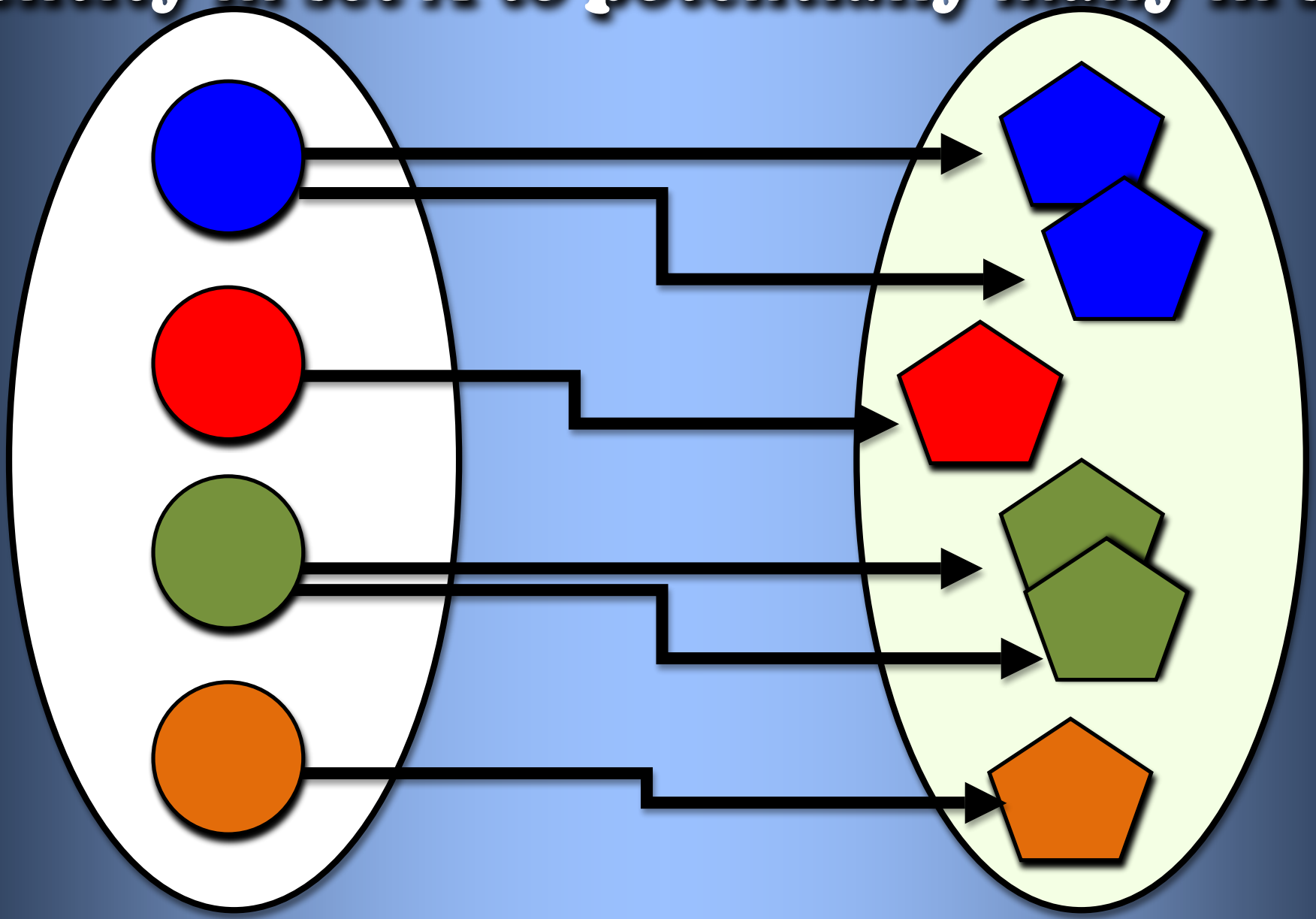
A ***ternery*** relationship is a relationship between **three** entities

An **n-ary** relationship is a relationship between **n** different entities

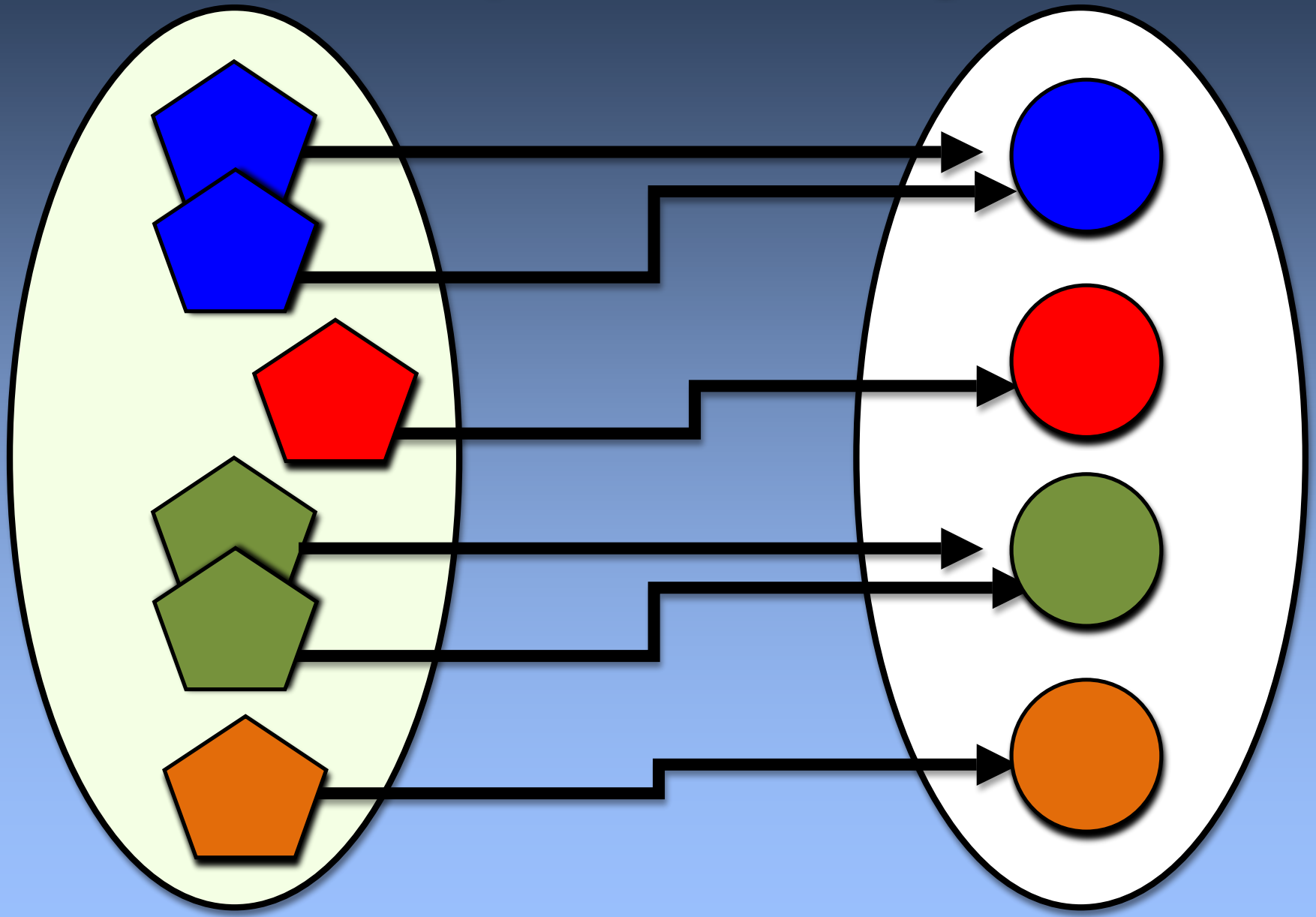
1-to-1 Relationships
(1 entity in set A to 1 in set B)



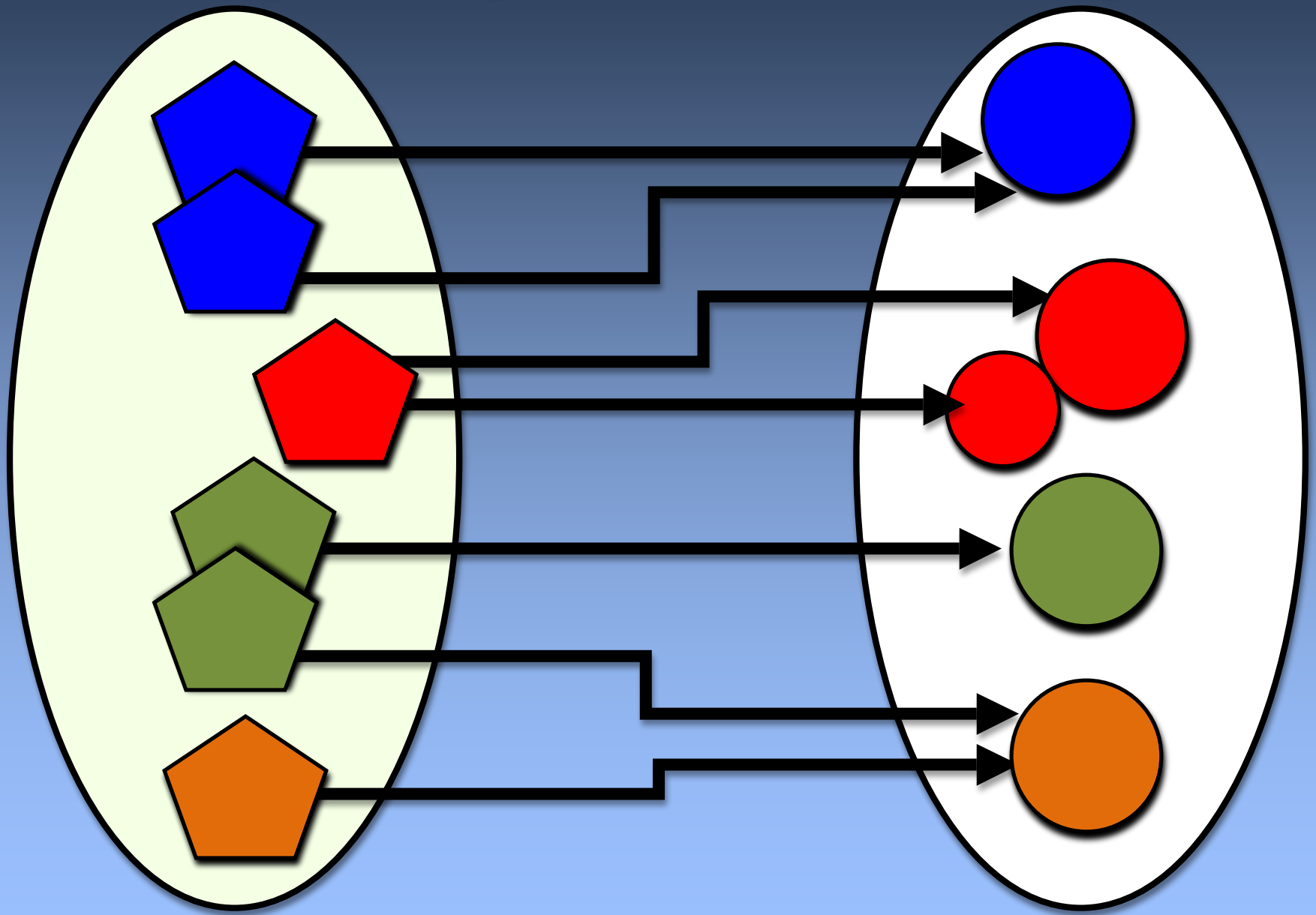
1-to-Many Relationships
(1 entity in set A to potentially many in set B)



Many-to-1 Relationships
(Potentially many in set A to just 1 in set B)



Many-to-Many Relationships
(Potentially many in set A to many in set B)



Mentor

Intern

Department

We depict **entities** in our E-R graphs as ***boxes***

Gender

Name

Date-of-Birth

… and depict **attributes** in our E-R graphs as ***ovals***

Attributes may be composite ...

First Name

Surname

Name

Gender

Intern

Date-of-Birth

Connect Entities to Attributes

Double-ring multi-value attributes

First Name

Surname

Gender

Name

Intern

Phone Number

Date-of-Birth

Derived attributes get ***dashed*** ovals

First Name

Surname

Name

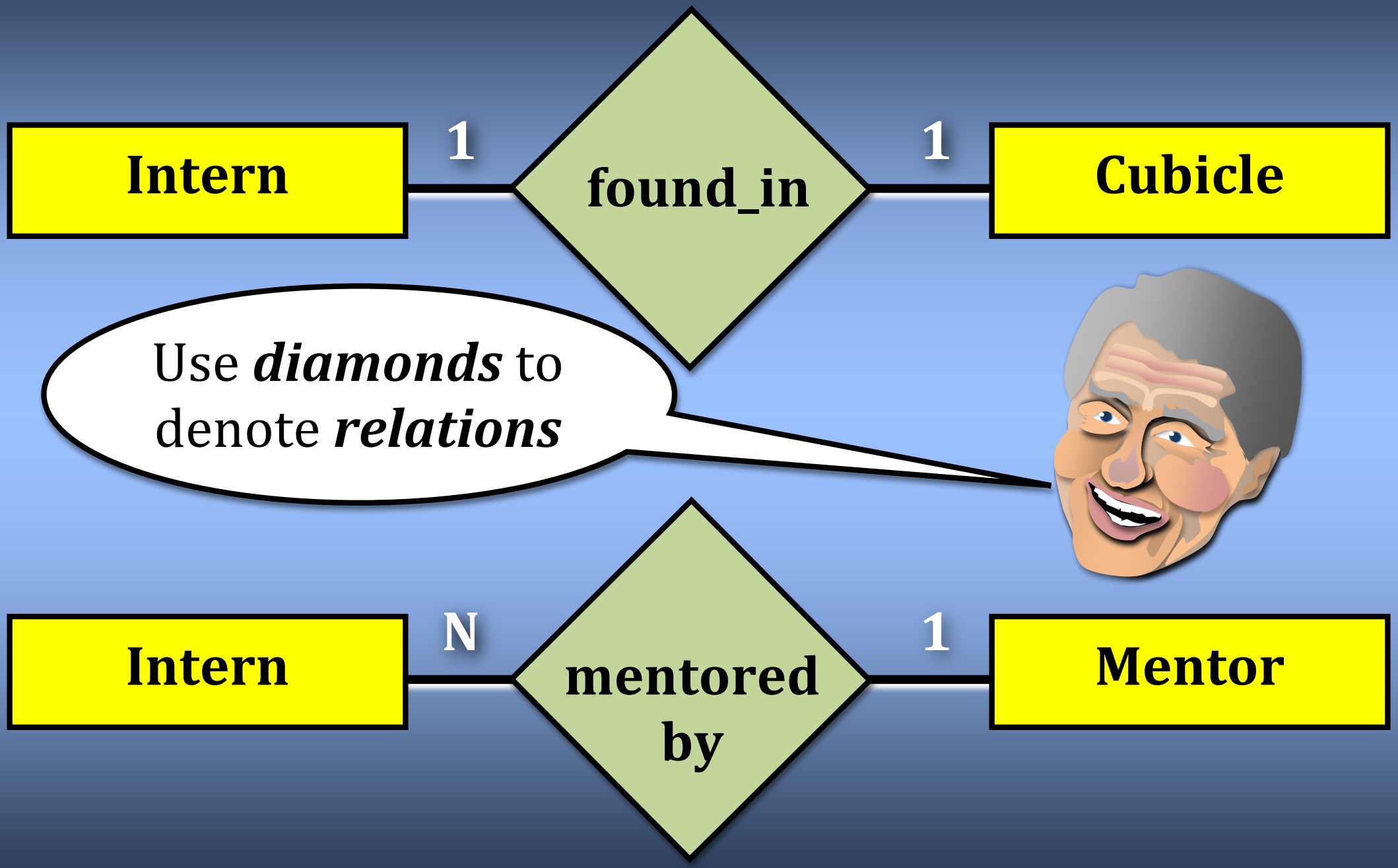
Age

Intern

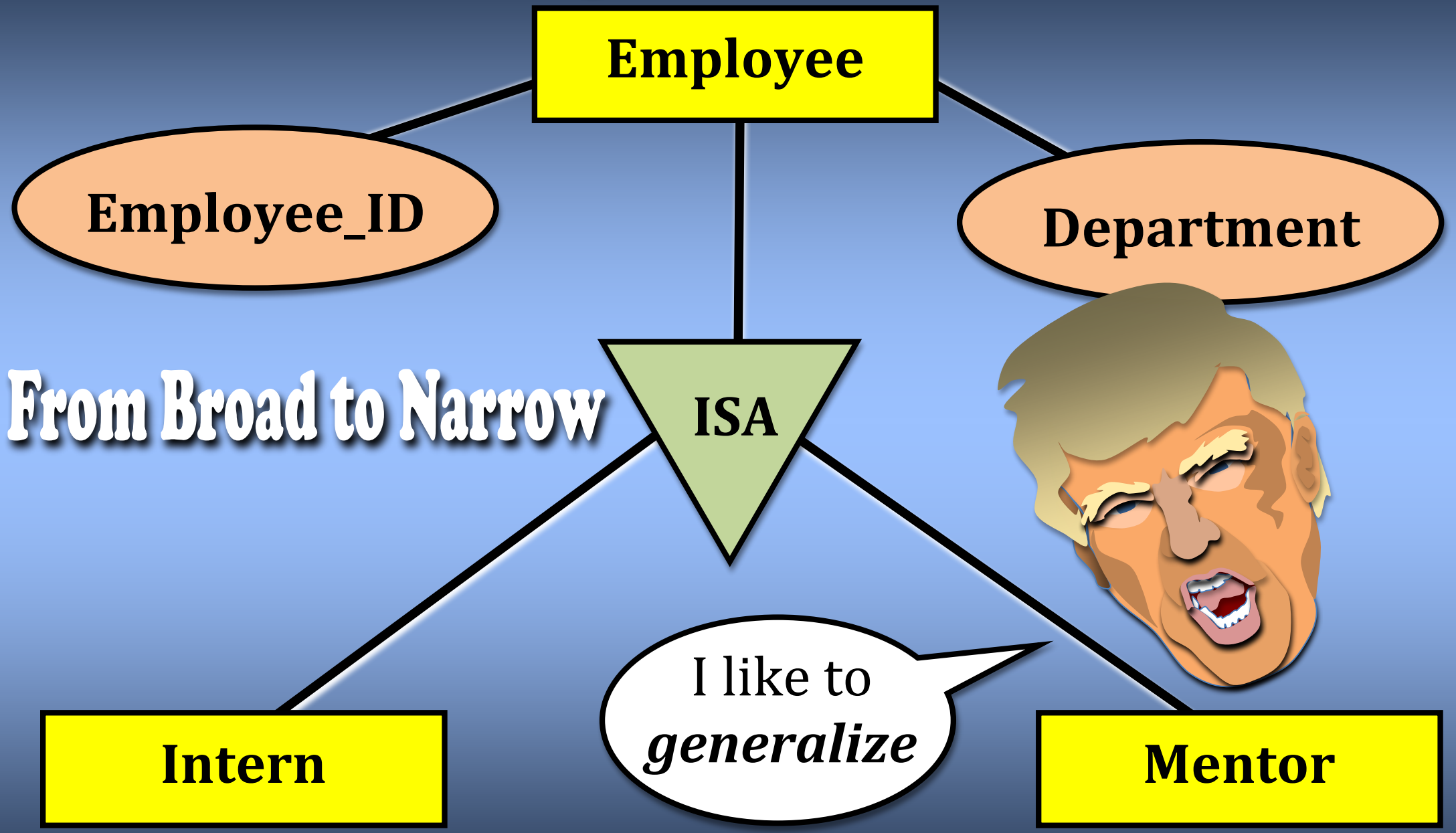
Phone Number

Gender

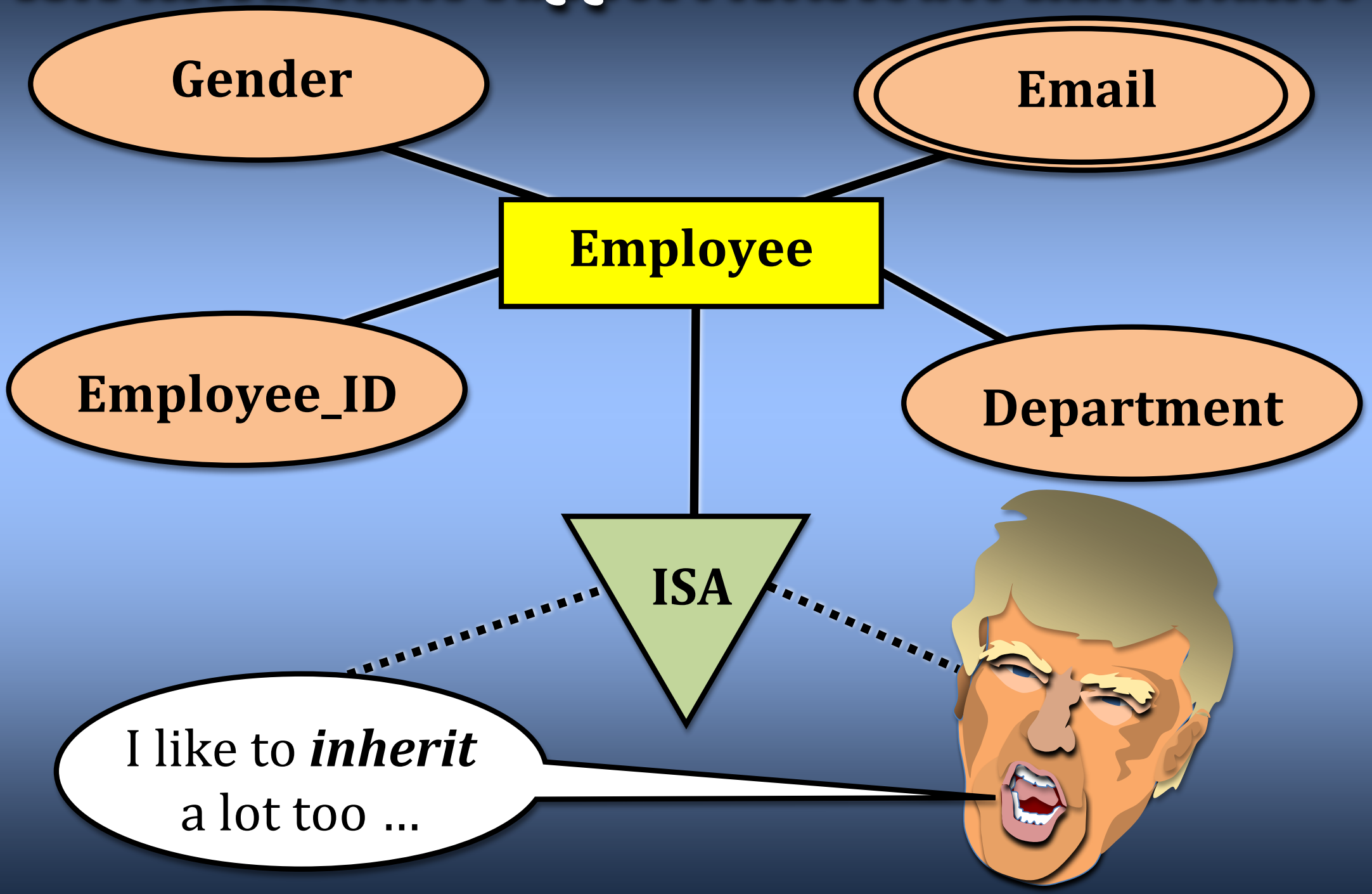
Date-of-Birth


Intern
1
found_in
1
Cubicle
Use *diamonds* to denote *relations*
Intern
N
mentored by
1
Mentor

ISA Hierarchies generalize Entities



ISA Hierarchies support Attribute Inheritance



Map Entities onto Tables, Attributes onto Columns and Assign a Primary Key

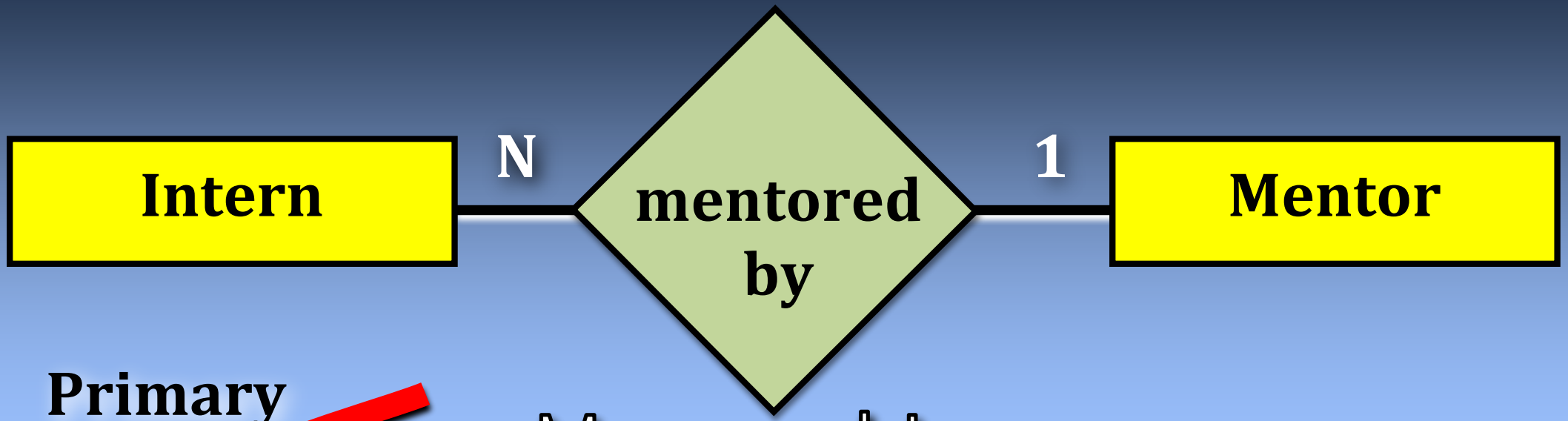
Mentors

Staff_ID	Name	Date Of Birth	Gender	Email
THX1138	Darth Vader	28.01.0024	Male	*Darth.V@AOL.com*
REX2910	Sheev Palpatine	01.09.0000	Male	*Uggo@hotmail.com*
OBK1204	Obiwan Kenobi	07.10.0012	Male	*OB1@gmail.com*
PAD4908	Padmé Amidala	14.12.0020	Female	*Padm@outlook.com*

Interns

Student_ID	Name	Date Of Birth	Gender	Email
8721138	Anakin Skywalker	28.01.0024	Male	*Darth.V@AOL.com*
3872614	Luke Skywalker	21.05.0042	Male	*Luke@hotmail.com*
3861638	Rey Skywalker	18.03.0066	Female	*Rey.S@gmail.com*

Map Entities onto Tables, Attributes onto Columns and Assign a Primary Key

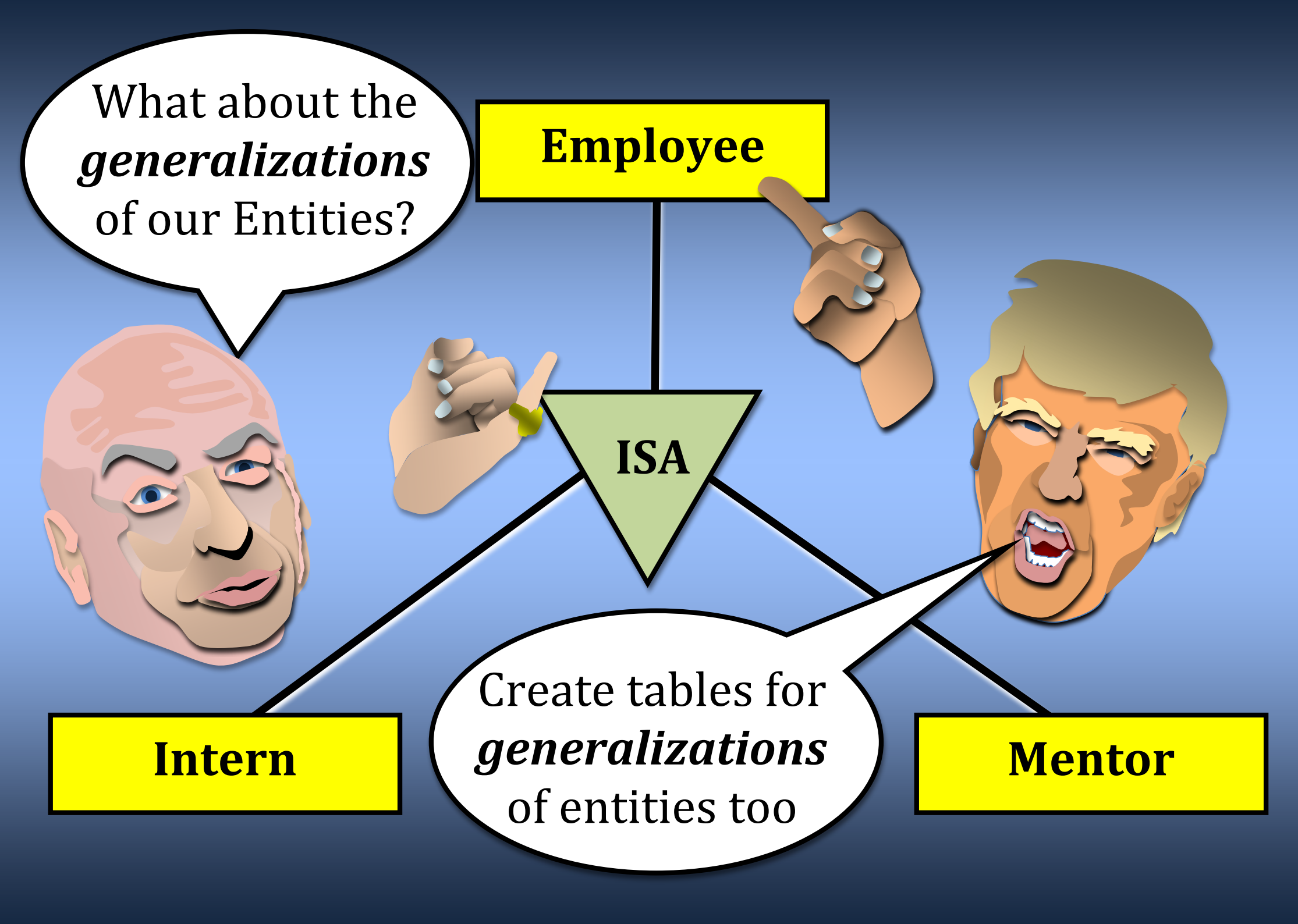


Primary Key

Mentorships

Staff_ID	Student_ID	Start Date	End Date	Department
OBK1204	8721138	19.07.0036	01.04.0042	*Jedi Affairs*
OBK1204	3872614	11.08.0061	21.04.0061	*Rebel Alliance*
REX2910	3861638	01.04.0042	11.06.0065	*Old Republic*
LKS2614	8721138	24.06.0099		*New Republic*

Map Relations onto Tables to connect Entity Tables


What about the *generalizations* of our Entities?
Employee
ISA
Intern
Create tables for *generalizations* of entities too
Mentor

Employees

Staff_ID	Name	Date Of Birth	Gender	Email
THX1138	Darth Vader	28.01.0024	Male	*Darth.V@AOL.com*
REX2910	Sheev Palpatine	01.09.0000	Male	*Uggo@hotmail.com*
OBK1204	Obiwan Kenobi	07.10.0012	Male	*OB1@gmail.com*
PAD4908	Padmé Amidala	14.12.0020	Female	*Padm@outlook.com*
LKS2614	Luke Skywalker	21.05.0042	Male	*Luke@hotmail.com*
RYS1638	Rey Skywalker	18.03.0066	Female	*Rey.S@gmail.com*

When we create the EMPLOYEES Table, we define the inheritable attributes at this level. There is no need to define these attributes as columns in the MENTORS & INTERNS tables (but now, we must give Staff_IDs to Interns)

In Specialization tables, put specialized attributes

Mentors

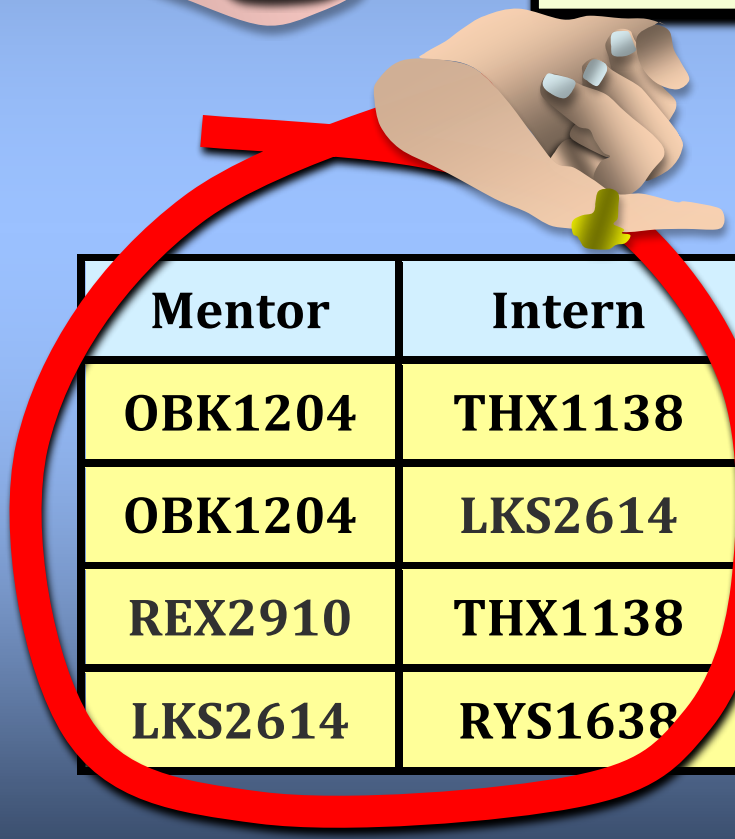
Staff_ID	Skill Level	Skill Set
THX1138	4	speed racing, sabre duelling, choke holds
REX2910	10	manipulation, politics, leadership
OBK1204	9	sabre duelling, voice control
PAD4908	4	politics, leadership, youth activism

Interns

Staff_ID	Student_ID	Education
THX1138	8721138	High School, Bsc, Msc
LKS2614	3872614	High School, MA(Rebellion)
RYS1638	3861638	High School

Inherited attributes are accessible via Table Joins

Now that Mentors and Interns are both specializations of Employee and both kinds of people have STAFF_IDs, we can simplify our table for the relation Mentorships between Interns and Mentors.



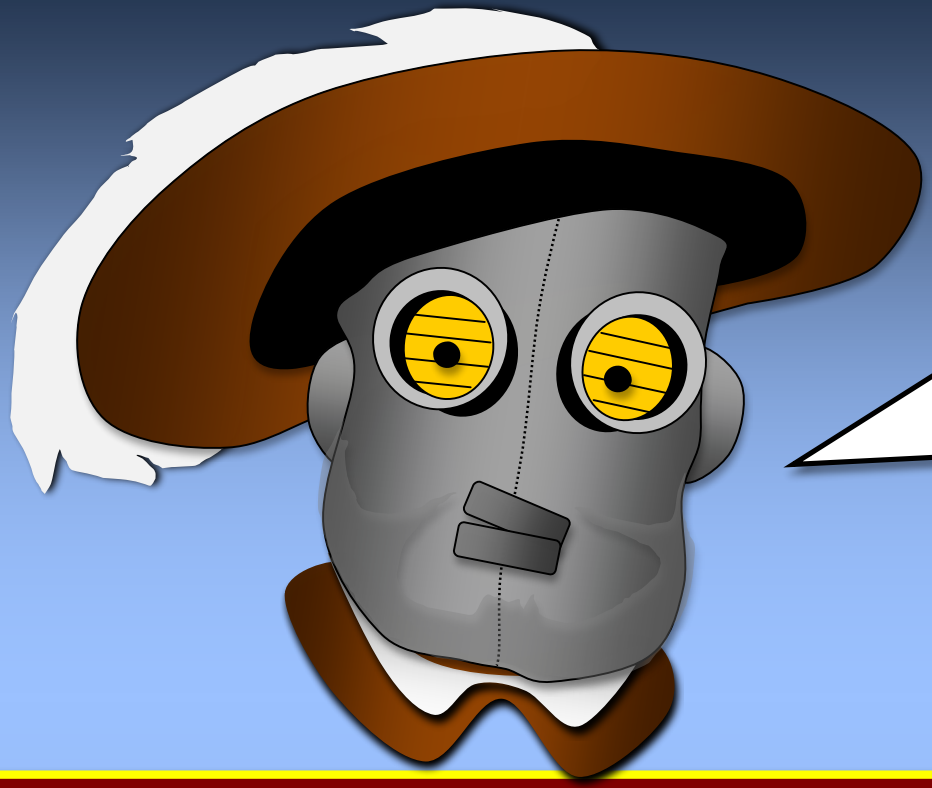
Mentorships

Mentor	Intern	Start Date	End Date	Department
OBK1204	**THX1138**	19.07.0036	01.04.0042	*Jedi Affairs*
OBK1204	**LKS2614**	11.08.0061	21.04.0061	*Rebel Alliance*
REX2910	**THX1138**	01.04.0042	11.06.0065	*Old Republic*
LKS2614	**RYS1638**	24.06.0099		*New Republic*

The Primary Key for this M-to-N relation is Two-fold

A determinant is any attribute *(simple or composite)* on which an other attribute is fully functionally dependent.

A relation is in Boyce Codd Normal Form (or BCNF) if, and only if, every determinant is a candidate key.



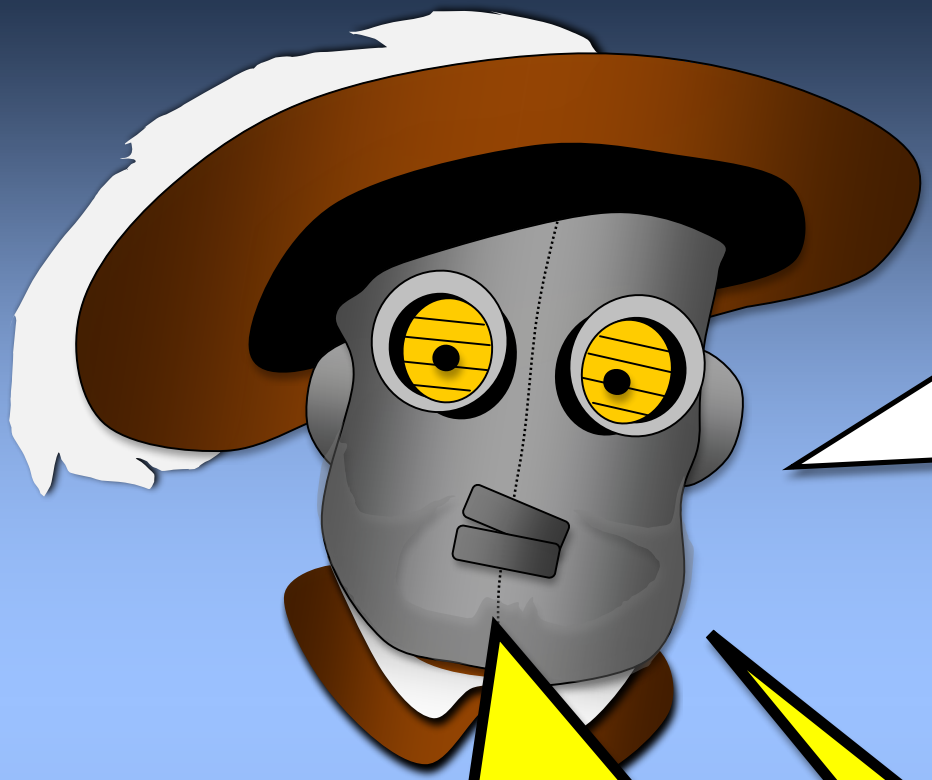
Let us consider the nature of **determination** in the abstract before exploring a concrete example.

Let R be a ternary relation between four things, A, B, C and D. So let R(a, b, c, d) be any instance of the relation R.

Let us suppose that: a + c *determines* b + d and a + d *determines* b for every instance of R

Let us suppose that:
a + c *determines* b + d
and
a + d *determines* b
for every instance of R

This pattern of determination suggests that a,c is a candidate key for *R* but a,d is not (because c is not determined by a,d)



We have at least two determinants for **R**: a+c (→ b+d) and a+d (→ b).

They ***overlap***!

A relation or table is in <u>BCNF</u> *(Boyce-Codd Normal Form)* when no determinants overlap *(no competing systems of determination)*.

A relation or table can be in 3rd Normal Form *(3NF)* but not be in BCNF *(Boyce-Codd Normal Form)*. So further normalization is needed.

I run an evil clinic where patients have four stages of appointments (0 … 3). Each stage is scheduled at different times, and with doctors with appropriate training.

Patient_Num	Patient_Name	Appointment_id	Time	Doctor
1	Mordo	0	09:00	Strange
2	Scott	0	09:00	Evil
3	Dormammu	1	10:00	Strange
4	Austin	0	13:00	Evil
5	Christine	1	14:00	Strange

What are the determinants here? First, *Patient_Num → Patient_Name*. Second, *Patient_Num + Appointment_id → Time + Doctor*. Third, *Time → Appointment_id*.

So what should be the *primary key* of my clinic's appointment book? Should it be Patient_Num, Appointment_id or should it be Patient_Num, Time ?

Patient_Num	Patient_Name	Appointment_id	Time	Doctor
1	Mordo	0	09:00	Strange
2	Scott	0	09:00	Evil
3	Dormammu	1	10:00	Strange
4	Austin	0	13:00	Evil
5	Christine	1	14:00	Strange

Let's choose Patient_Num, Appointment_id for primary key. Now let's put the database into 1NF, then 2NF, then 3NF and then BCNF. The above table is already in 1NF (we have a key that is unique for every row).

2NF: Factor out the partial key dependencies, e.g.:

Patient_Num → Patient_Name

Patient_Num	Appointment_id	Time	Doctor
1	0	09:00	Strange
2	0	09:00	Evil
3	1	10:00	Strange
4	0	13:00	Evil
5	1	14:00	Strange

Patient_Num	Patient_Name
1	Mordo
2	Scott
3	Dormammu
4	Austin
5	Christine

Remember, Patient_NUM is not the whole primary key, so the dependency of Patient_Name on this attribute is a partial key dependency.

3NF: Factor out transitive dependencies. Here there are: **None**

Let's look at the refactored main table. Is it in BCNF? What are its determinants?
Patient_Num + Appointment_id
→ Time + Doctor ? That's the primary key

Patient_Num	Appointment_id	Time	Doctor
1	0	09:00	Strange
2	0	09:00	Evil
3	1	10:00	Strange
4	0	13:00	Evil
5	1	14:00	Strange

What about Time → Appointment_id ? It seems that the appointment type is dependent upon the time, but time is not a candidate key! So this table is *not* in BCNF.

Patients

Patient_Num	Patient_Name
1	Mordo
2	Scott
3	Dormammu
4	Austin
5	Christine

Patient Schedule

Patient_Num	Time	Doctor
1	09:00	Strange
2	09:00	Evil
3	10:00	Strange
4	13:00	Evil
5	14:00	Strange

Time	Appointment_id
09:00	0
09:00	0
10:00	1
13:00	0
14:00	1

Time Allocation

BCNF: Factor out non-candidate-key dependencies: All Gone!