COSC 416
NoSQL Databases

Relational Model (Review)

Dr. Ramon Lawrence
University of British Columbia Okanagan
ramon.lawrence@ubc.ca

Relational Model History

The relational model was proposed by E. F. Codd in 1970.

One of the first relational database systems, System R, developed at IBM led to several important breakthroughs:

- the first version of SQL
- various commercial products such as Oracle and DB2
- extensive research on concurrency control, transaction management, and query processing and optimization

Commercial implementations (RDBMSs) appeared in the late 1970s and early 1980s. Currently, the relational model is the foundation of the majority of commercial database systems.

Relational Model Definitions

A ***relation*** is a table with columns and rows.

An ***attribute*** is a named column of a relation.

A ***tuple*** is a row of a relation.

A ***domain*** is a set of allowable values for one or more attributes.

The ***degree*** of a relation is the number of attributes it contains.

The ***cardinality*** of a relation is the number of tuples it contains.

A ***relational database*** is a collection of normalized relations with distinct relation names.

The ***intension*** of a relation is the structure of the relation including its domains.

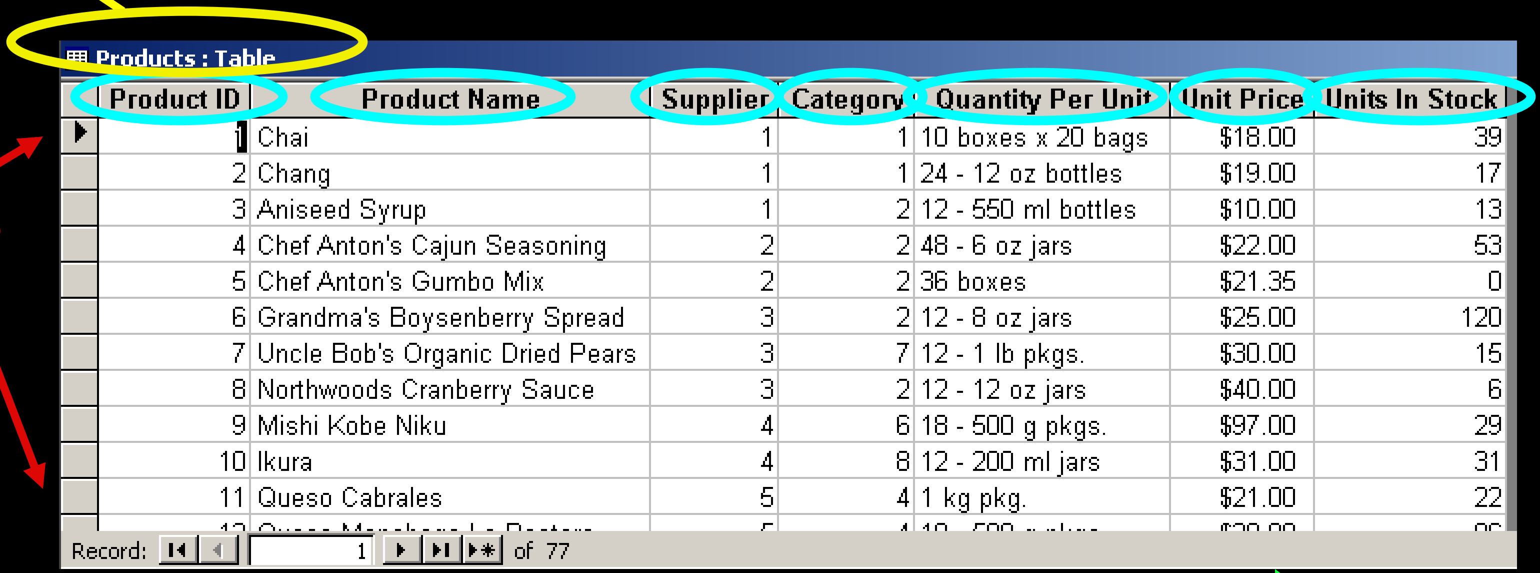
The ***extension*** of a relation is the set of tuples currently in the relation.

Relation Example

relation

attributes

tuples



	Product ID	Product Name	Supplier	Category	Quantity Per Unit	Unit Price	Units In Stock
▶	1	Chai	1	1	10 boxes x 20 bags	$18.00	39
	2	Chang	1	1	24 - 12 oz bottles	$19.00	17
	3	Aniseed Syrup	1	2	12 - 550 ml bottles	$10.00	13
	4	Chef Anton's Cajun Seasoning	2	2	48 - 6 oz jars	$22.00	53
	5	Chef Anton's Gumbo Mix	2	2	36 boxes	$21.35	0
	6	Grandma's Boysenberry Spread	3	2	12 - 8 oz jars	$25.00	120
	7	Uncle Bob's Organic Dried Pears	3	7	12 - 1 lb pkgs.	$30.00	15
	8	Northwoods Cranberry Sauce	3	2	12 - 12 oz jars	$40.00	6
	9	Mishi Kobe Niku	4	6	18 - 500 g pkgs.	$97.00	29
	10	Ikura	4	8	12 - 200 ml jars	$31.00	31
	11	Queso Cabrales	5	4	1 kg pkg.	$21.00	22
	12	Queso Manchego La Pastora	5	4	10 - 500 g pkgs.	$38.00	86

Record: |◀ ◀ 1 ▶ ▶| ▶* of 77

Degree = 7
Cardinality = 77

Domain of Unit Price is *currency*.

Relational Keys

Keys are used to uniquely identify a tuple in a relation.

- ⇨ Note that keys apply to the relational schema not to the relational instance. That is, looking at the current instance cannot tell you for sure if the set of attributes is a key.

A ***superkey*** is a set of attributes that uniquely identifies a tuple in a relation.

A ***key*** is a *minimal* set of attributes that uniquely identifies a tuple in a relation.

A ***candidate key*** is one of the possible keys of a relation.

A ***primary key*** is the candidate key designated as the distinguishing key of a relation.

A ***foreign key*** is a set of attributes in one relation referring to the primary key of another relation.

- ⇨ Foreign keys allow referential integrity to be enforced.

★ *Relational Integrity*

Integrity rules are used to insure the data is accurate.

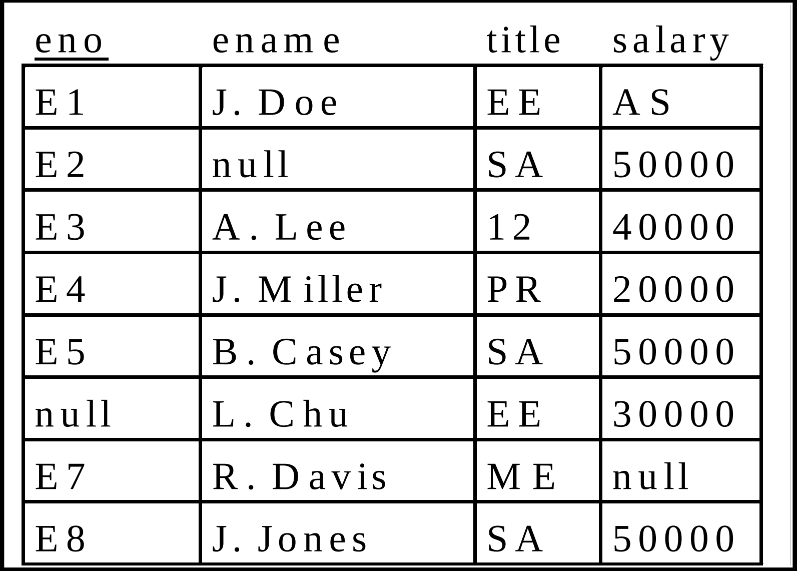
Constraints are rules or restrictions that apply to the database and limit the data values it may store.

Types of constraints:

- ***Domain constraint*** - Every value for an attribute must be an element of the attribute's domain or be `null`.
  - ⇨ `null` represents a value that is currently unknown or not applicable.
  - ⇨ `null` is not the same as zero or an empty string.
- ***Entity integrity constraint*** - In a base relation, no attribute of a primary key can be null.
- ***Referential integrity constraint*** - If a foreign key exists in a relation, then the foreign key value must match a primary key value of a tuple in the referenced relation or be null.

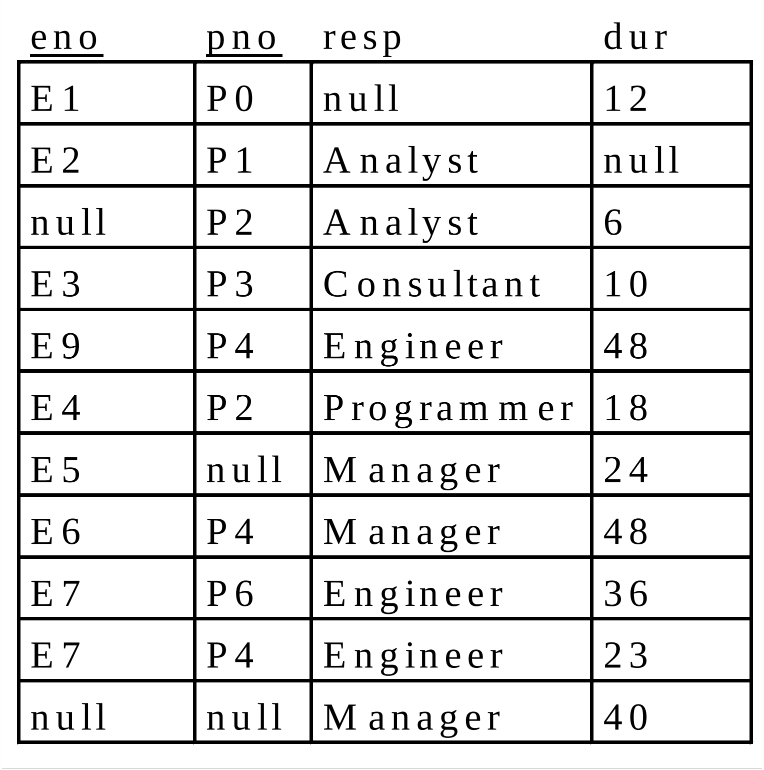
Integrity Questions

Emp Relation



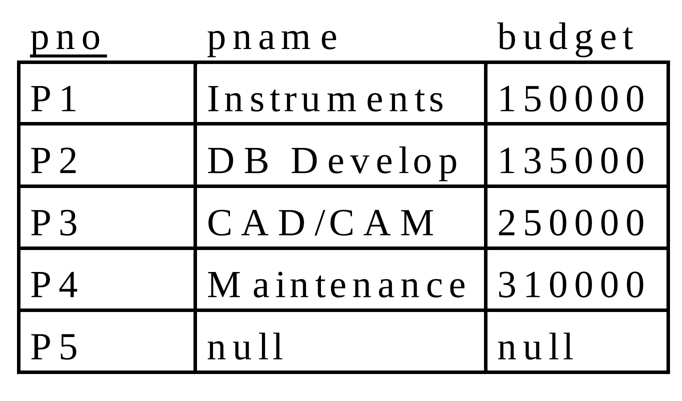
eno	ename	title	salary
E1	J. Doe	EE	AS
E2	null	SA	50000
E3	A. Lee	12	40000
E4	J. Miller	PR	20000
E5	B. Casey	SA	50000
null	L. Chu	EE	30000
E7	R. Davis	ME	null
E8	J. Jones	SA	50000

WorksOn Relation



eno	pno	resp	dur
E1	P0	null	12
E2	P1	Analyst	null
null	P2	Analyst	6
E3	P3	Consultant	10
E9	P4	Engineer	48
E4	P2	Programmer	18
E5	null	Manager	24
E6	P4	Manager	48
E7	P6	Engineer	36
E7	P4	Engineer	23
null	null	Manager	40

Proj Relation



pno	pname	budget
P1	Instruments	150000
P2	DB Develop	135000
P3	CAD/CAM	250000
P4	Maintenance	310000
P5	null	null

Question:
1) Find all violations of integrity constraints in these three relations.

Relational Algebra

A ***query language*** is used to update and retrieve data that is stored in a data model.

Relational algebra is a set of relational operations for retrieving data.

- Just like algebra with numbers, relational algebra consists of operands (which are relations) and a set of operators.

Every relational operator takes as input one or more relations and produces a relation as output.

- Closure property - input is relations, output is relations
- Unary operations - operate on one relation
- Binary operations - have two relations as input

A sequence of relational algebra operators is called a ***relational algebra expression***.

Selection Operation

The ***selection operation*** is a unary operation that takes in a relation as input and returns a new relation as output that contains a subset of the tuples of the input relation.

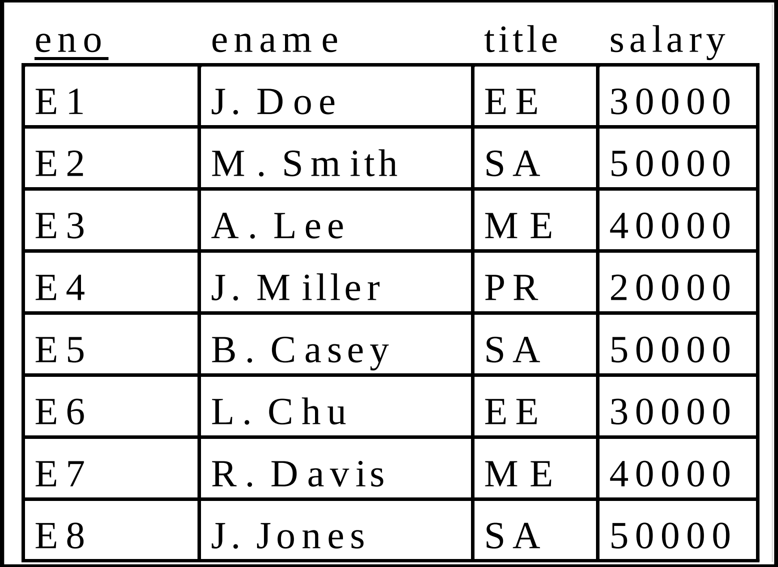
- That is, the output relation has the same number of columns as the input relation, but may have less rows.

To determine which tuples are in the output, the selection operation has a specified condition, called a ***predicate***, that tuples must satisfy to be in the output.

- The predicate is similar to a condition in an `if` statement.

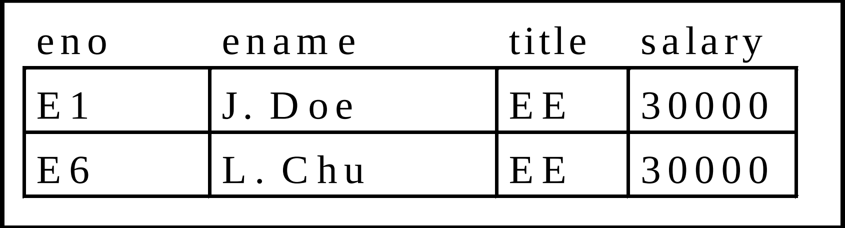
Selection Example

Emp Relation



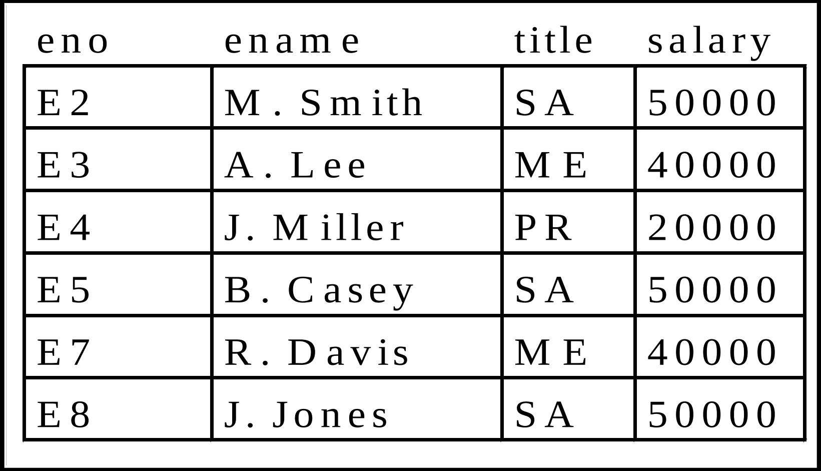
eno	ename	title	salary
E1	J. Doe	EE	30000
E2	M. Smith	SA	50000
E3	A. Lee	ME	40000
E4	J. Miller	PR	20000
E5	B. Casey	SA	50000
E6	L. Chu	EE	30000
E7	R. Davis	ME	40000
E8	J. Jones	SA	50000

$\sigma_{title\ =\ 'EE'}\ (\text{Emp})$



eno	ename	title	salary
E1	J. Doe	EE	30000
E6	L. Chu	EE	30000

$\sigma_{salary\ >\ 35000\ OR\ title\ =\ 'PR'}\ (\text{Emp})$



eno	ename	title	salary
E2	M. Smith	SA	50000
E3	A. Lee	ME	40000
E4	J. Miller	PR	20000
E5	B. Casey	SA	50000
E7	R. Davis	ME	40000
E8	J. Jones	SA	50000

Projection Operation

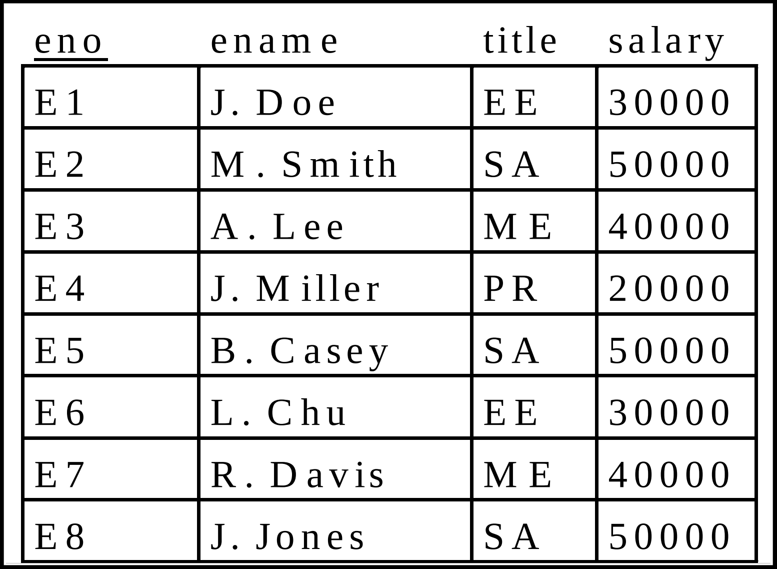
The ***projection operation*** is a unary operation that takes in a relation as input and returns a new relation as output that contains a subset of the attributes of the input relation and all non-duplicate tuples.

- The output relation has the same number of tuples as the input relation unless removing the attributes caused duplicates to be present.
- Question: When are we guaranteed to never have duplicates when performing a projection operation?

Besides the relation, the projection operation takes as input the names of the attributes that are to be in the output relation.

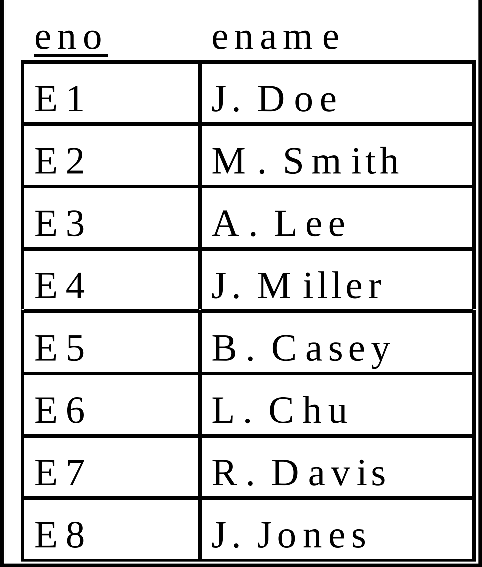
Projection Example

Emp Relation



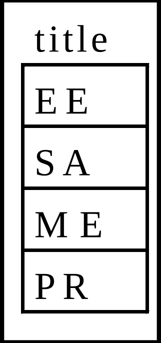
eno	ename	title	salary
E1	J. Doe	EE	30000
E2	M. Smith	SA	50000
E3	A. Lee	ME	40000
E4	J. Miller	PR	20000
E5	B. Casey	SA	50000
E6	L. Chu	EE	30000
E7	R. Davis	ME	40000
E8	J. Jones	SA	50000

$\Pi_{\mathit{eno,ename}}\,(\mathrm{Emp})$



eno	ename
E1	J. Doe
E2	M. Smith
E3	A. Lee
E4	J. Miller
E5	B. Casey
E6	L. Chu
E7	R. Davis
E8	J. Jones

$\Pi_{\mathit{title}}\,(\mathrm{Emp})$



title
EE
SA
ME
PR

Cartesian Product

The ***Cartesian product*** of two relations *R* (of degree k_1) and S (of degree k_2) is:

$$\mathbf{R \times S = \{t \mid t\ [A_1,\ldots,A_{k_1}] \in R \text{ and } t\ [A_{k_1+1},\ldots,A_{k_1+k_2}] \in S\}}$$

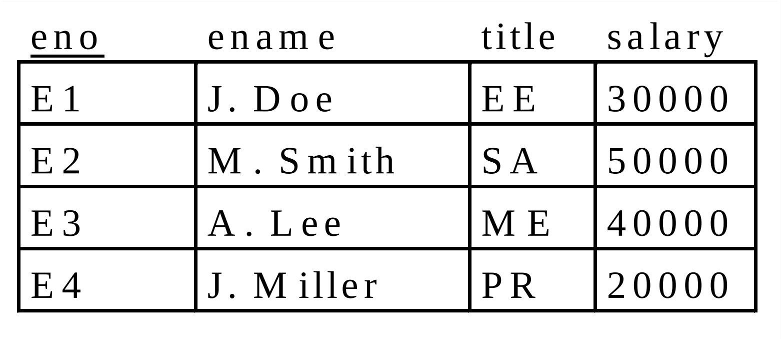
The result of *R* × *S* is a relation of degree ($k_1 + k_2$) and consists of all ($k_1 + k_2$)-tuples where each tuple is a concatenation of one tuple of *R* with one tuple of *S*.

The cardinality of R × S is |*R*| * |*S*|.

The Cartesian product is also known as ***cross product***.

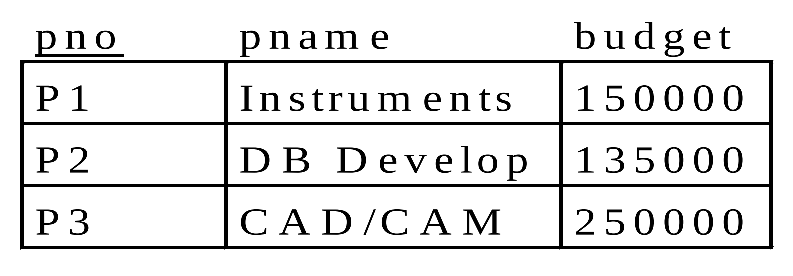
Cartesian Product Example

Emp Relation



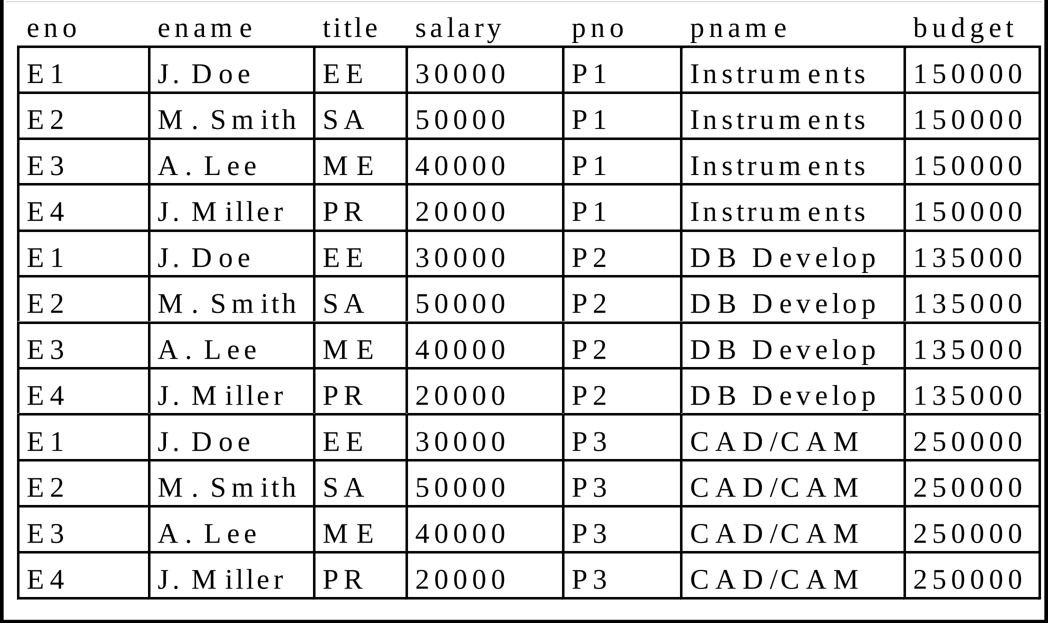
eno	ename	title	salary
E1	J. Doe	EE	30000
E2	M. Smith	SA	50000
E3	A. Lee	ME	40000
E4	J. Miller	PR	20000

Proj Relation



pno	pname	budget
P1	Instruments	150000
P2	DB Develop	135000
P3	CAD/CAM	250000

Emp × *Proj*



eno	ename	title	salary	pno	pname	budget
E1	J. Doe	EE	30000	P1	Instruments	150000
E2	M. Smith	SA	50000	P1	Instruments	150000
E3	A. Lee	ME	40000	P1	Instruments	150000
E4	J. Miller	PR	20000	P1	Instruments	150000
E1	J. Doe	EE	30000	P2	DB Develop	135000
E2	M. Smith	SA	50000	P2	DB Develop	135000
E3	A. Lee	ME	40000	P2	DB Develop	135000
E4	J. Miller	PR	20000	P2	DB Develop	135000
E1	J. Doe	EE	30000	P3	CAD/CAM	250000
E2	M. Smith	SA	50000	P3	CAD/CAM	250000
E3	A. Lee	ME	40000	P3	CAD/CAM	250000
E4	J. Miller	PR	20000	P3	CAD/CAM	250000

θ -Join

Theta (θ) join is a derivative of the Cartesian product. Instead of taking all combinations of tuples from *R* and *S*, we only take a subset of those tuples that match a given condition *F*:

$$R \bowtie_F S = \{t \mid t\,[A_1,\ldots,A_{k_1}] \in R \text{ and } t\,[A_{k_1+1},\ldots,A_{k_1+k_2}] \in S \text{ and } F(t) \text{ is true}\}$$

where

- *R*, *S* are relations, *t* is a tuple variable
- *F*(*t*) is a formula defined as that of selection.

Note that $R \bowtie_F S = \sigma_F(R \times S)$.

θ-Join Example

WorksOn Relation

eno	pno	resp	dur
E1	P1	Manager	12
E2	P1	Analyst	24
E2	P2	Analyst	6
E3	P4	Engineer	48
E5	P2	Manager	24
E6	P4	Manager	48
E7	P3	Engineer	36
E7	P4	Engineer	23

Proj Relation

pno	pname	budget
P1	Instruments	150000
P2	DB Develop	135000
P3	CAD/CAM	250000
P4	Maintenance	310000
P5	CAD/CAM	500000

WorksOn $\bowtie_{dur*10000 > budget}$ Proj

eno	pno	resp	dur	P.pno	pname	budget
E2	P1	Analyst	24	P1	Instruments	150000
E2	P1	Analyst	24	P2	DB Develop	135000
E3	P4	Engineer	48	P1	Instruments	150000
E3	P4	Engineer	48	P2	DB Develop	135000
E3	P4	Engineer	48	P3	CAD/CAM	250000
E3	P4	Engineer	48	P4	Maintenance	310000
E5	P2	Manager	24	P1	Instruments	150000
E5	P2	Manager	24	P2	DB Develop	135000
E6	P4	Manager	48	P1	Instruments	150000
E6	P4	Manager	48	P2	DB Develop	135000
E6	P4	Manager	48	P3	CAD/CAM	250000
E6	P4	Manager	48	P4	Maintenance	310000
E7	P3	Engineer	36	P1	Instruments	150000
E7	P3	Engineer	36	P2	DB Develop	135000
E7	P3	Engineer	36	P3	CAD/CAM	250000
E7	P4	Engineer	23	P1	Instruments	150000
E7	P4	Engineer	23	P2	DB Develop	135000

Types of Joins

The θ-Join is a general join in that it allows any expression in the condition *F*. However, there are more specialized joins that are frequently used.

A ***equijoin*** only contains the equality operator (=) in formula *F*.

- e.g. $\text{WorksOn} \bowtie_{\textit{WorksOn.pno = Proj.pno}} \text{Proj}$

A ***natural join*** over two relations *R* and *S* denoted by R ⋈ S is the equijoin of *R* and *S* over a set of attributes common to both *R* and *S*.

- It removes the “extra copies” of the join attributes.
- The attributes must have the same name in both relations.

Equijoin Example

WorksOn Relation

eno	pno	resp	dur
E1	P1	Manager	12
E2	P1	Analyst	24
E2	P2	Analyst	6
E3	P4	Engineer	48
E5	P2	Manager	24
E6	P4	Manager	48
E7	P3	Engineer	36
E7	P4	Engineer	23

Proj Relation

pno	pname	budget
P1	Instruments	150000
P2	DB Develop	135000
P3	CAD/CAM	250000
P4	Maintenance	310000
P5	CAD/CAM	500000

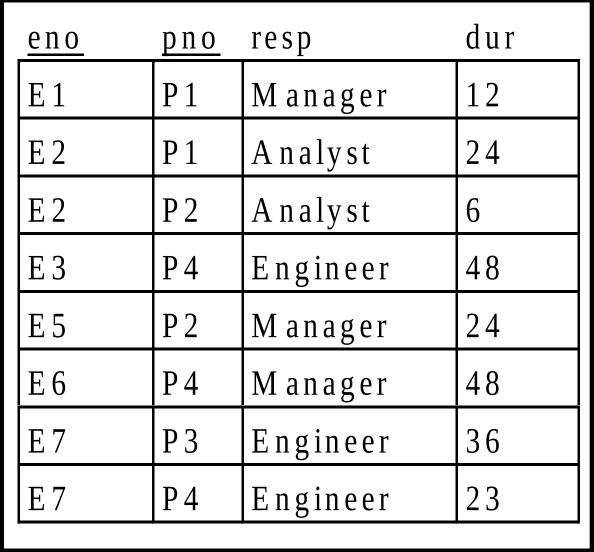
$\text{WorksOn} \bowtie_{\mathit{WorksOn.pno\ =\ Proj.pno}} \text{Proj}$

eno	pno	resp	dur	P.pno	pname	budget
E1	P1	Manager	12	P1	Instruments	150000
E2	P1	Analyst	24	P1	Instruments	150000
E2	P2	Analyst	6	P2	DB Develop	135000
E3	P4	Engineer	48	P4	Maintenance	310000
E5	P2	Manager	24	P2	DB Develop	135000
E6	P4	Manager	48	P4	Maintenance	310000
E7	P3	Engineer	36	P3	CAD/CAM	250000
E7	P4	Engineer	23	P4	Maintenance	310000

What is the meaning of this join?

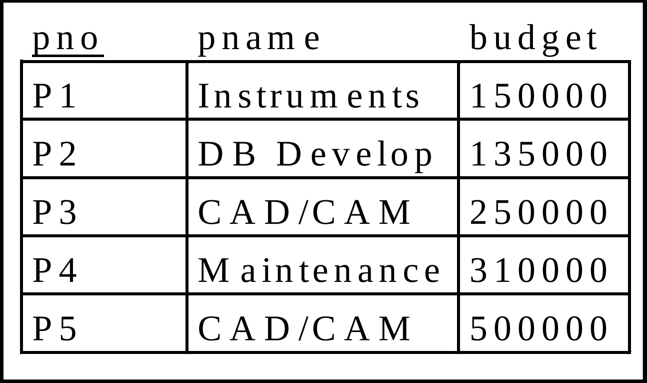
Natural join Example

WorksOn Relation



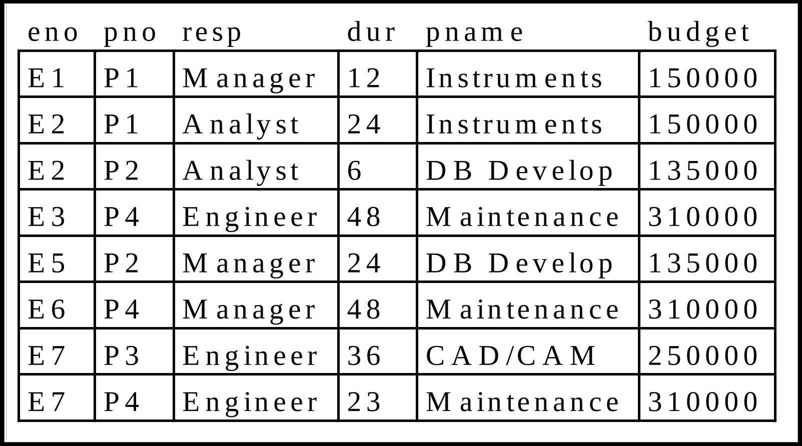
eno	pno	resp	dur
E1	P1	Manager	12
E2	P1	Analyst	24
E2	P2	Analyst	6
E3	P4	Engineer	48
E5	P2	Manager	24
E6	P4	Manager	48
E7	P3	Engineer	36
E7	P4	Engineer	23

Proj Relation



pno	pname	budget
P1	Instruments	150000
P2	DB Develop	135000
P3	CAD/CAM	250000
P4	Maintenance	310000
P5	CAD/CAM	500000

WorksOn ⋈ Proj



eno	pno	resp	dur	pname	budget
E1	P1	Manager	12	Instruments	150000
E2	P1	Analyst	24	Instruments	150000
E2	P2	Analyst	6	DB Develop	135000
E3	P4	Engineer	48	Maintenance	310000
E5	P2	Manager	24	DB Develop	135000
E6	P4	Manager	48	Maintenance	310000
E7	P3	Engineer	36	CAD/CAM	250000
E7	P4	Engineer	23	Maintenance	310000

Natural join is performed by comparing *pno* in both relations.

Outer Joins

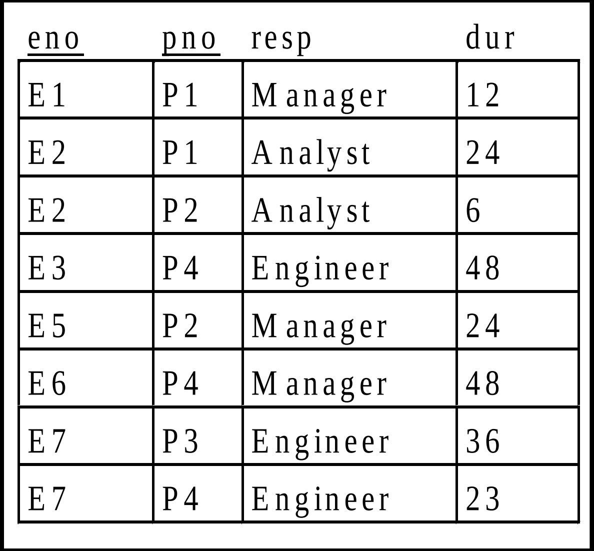
Outer joins are used in cases where performing a join "loses" some tuples of the relations. These are called *dangling tuples*.

There are three types of outer joins:

- 1) ***Left outer join*** - *R* ⟕ *S* - The output contains all tuples of *R* that match with tuples of *S*. If there is a tuple in *R* that matches with no tuple in *S*, the tuple is included in the final result and is padded with nulls for the attributes of *S*.
- 2) ***Right outer join*** - *R* ⟖ *S* - The output contains all tuples of *S* that match with tuples of *R*. If there is a tuple in *S* that matches with no tuple in *R*, the tuple is included in the final result and is padded with nulls for the attributes of *R*.
- 3) ***Full outer join*** - *R* ⟗ *S* - All tuples of *R* and *S* are included in the result whether or not they have a matching tuple in the other relation.

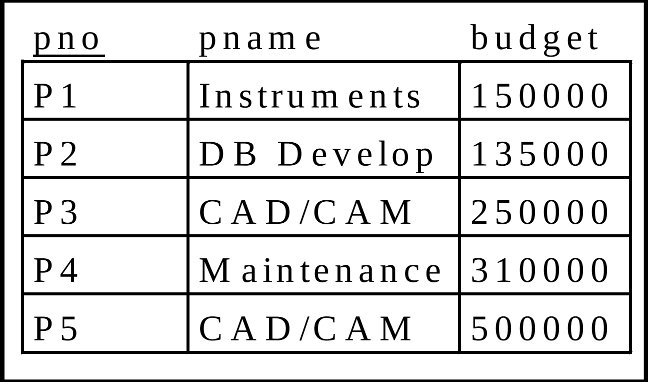
Right Outer Join Example

WorksOn Relation



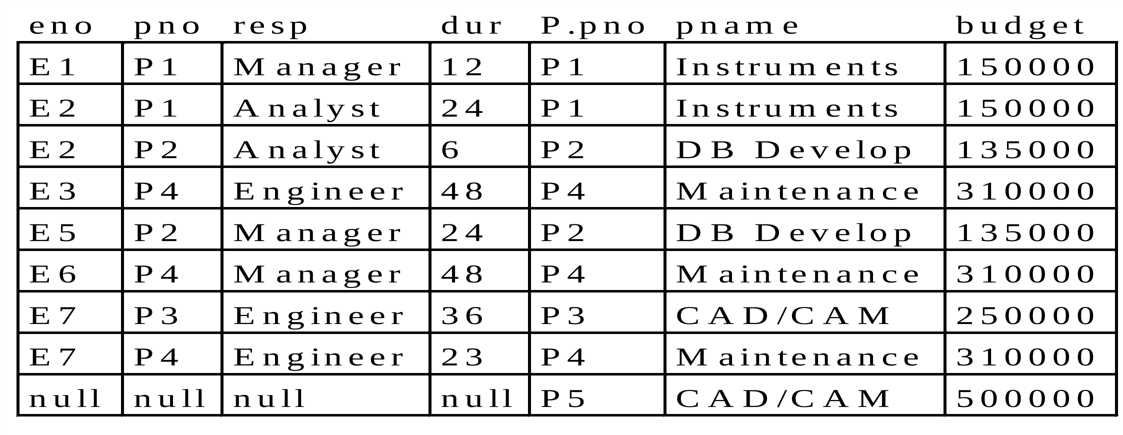
eno	pno	resp	dur
E1	P1	Manager	12
E2	P1	Analyst	24
E2	P2	Analyst	6
E3	P4	Engineer	48
E5	P2	Manager	24
E6	P4	Manager	48
E7	P3	Engineer	36
E7	P4	Engineer	23

Proj Relation



pno	pname	budget
P1	Instruments	150000
P2	DB Develop	135000
P3	CAD/CAM	250000
P4	Maintenance	310000
P5	CAD/CAM	500000

WorksOn ⟖$_{WorksOn.pno = Proj.pno}$ Proj



eno	pno	resp	dur	P.pno	pname	budget
E1	P1	Manager	12	P1	Instruments	150000
E2	P1	Analyst	24	P1	Instruments	150000
E2	P2	Analyst	6	P2	DB Develop	135000
E3	P4	Engineer	48	P4	Maintenance	310000
E5	P2	Manager	24	P2	DB Develop	135000
E6	P4	Manager	48	P4	Maintenance	310000
E7	P3	Engineer	36	P3	CAD/CAM	250000
E7	P4	Engineer	23	P4	Maintenance	310000
null	null	null	null	P5	CAD/CAM	500000

SQL Query Summary

The general form of the `SELECT` statement is:

SELECT <attribute list>
FROM <table list>
[WHERE (*condition*)]
[GROUP BY <grouping attributes>]
[HAVING <group condition>]
[ORDER BY <attribute list>]

- Clauses in square brackets ([,]) are optional.
- There are often numerous ways to express the same query in SQL.

Example Relation Instances

Emp Relation

eno	ename	bdate	title	salary	supereno	dno
E1	J. Doe	01-05-75	EE	30000	E2	null
E2	M. Smith	06-04-66	SA	50000	E5	D3
E3	A. Lee	07-05-66	ME	40000	E7	D2
E4	J. Miller	09-01-50	PR	20000	E6	D3
E5	B. Casey	12-25-71	SA	50000	E8	D3
E6	L. Chu	11-30-65	EE	30000	E7	D2
E7	R. Davis	09-08-77	ME	40000	E8	D1
E8	J. Jones	10-11-72	SA	50000	null	D1

WorksOn Relation

eno	pno	resp	hours
E1	P1	Manager	12
E2	P1	Analyst	24
E2	P2	Analyst	6
E3	P3	Consultant	10
E3	P4	Engineer	48
E4	P2	Programmer	18
E5	P2	Manager	24
E6	P4	Manager	48
E7	P3	Engineer	36

Proj Relation

pno	pname	budget	dno
P1	Instruments	150000	D1
P2	DB Develop	135000	D2
P3	Budget	250000	D3
P4	Maintenance	310000	D2
P5	CAD/CAM	500000	D2

Dept Relation

dno	dname	mgreno
D1	Management	E8
D2	Consulting	E7
D3	Accounting	E5
D4	Development	null

SQL Practice Questions

1) Return the project names that have a budget > 250000.

2) List all employee names where the employee's name contains an 'S' and the responsibility ends in 'ER'.

3) Give a list of all employees who work on a project for the 'Management' department ordered by project number (asc).

4) For each employee, return the total number of hours they have worked.

5) List the employees with title 'EE’ that make more than all employees with title 'PR'.

Conclusion

The ***relational model***:

- represents data as relations which are sets of tuples.
- has several forms of ***constraints*** to guarantee data integrity.
- uses ***keys*** to uniquely identify tuples in relations.
- can be queried using ***relational algebra*** or ***SQL***.

Objectives

- Define: relation, attribute, tuple, domain, degree, cardinality, relational DB, intension, extension
- Define: superkey, key, candidate key, primary key, foreign key
- Define: integrity, constraints, domain constraint, entity integrity constraint, referential integrity constraint

★ Be able to write an English query in SQL.