Relational Model

COSC 516 – Cloud Databases

Relational Model History

The relational model was proposed by E. F. Codd in 1970. Commercial implementations appeared in the late 1970s and early 1980s.

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

Currently, the relational model is the foundation of the majority of commercial database systems.

The Relational Model: Terminology

The ***relational model*** organizes data into tables called relations.

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.

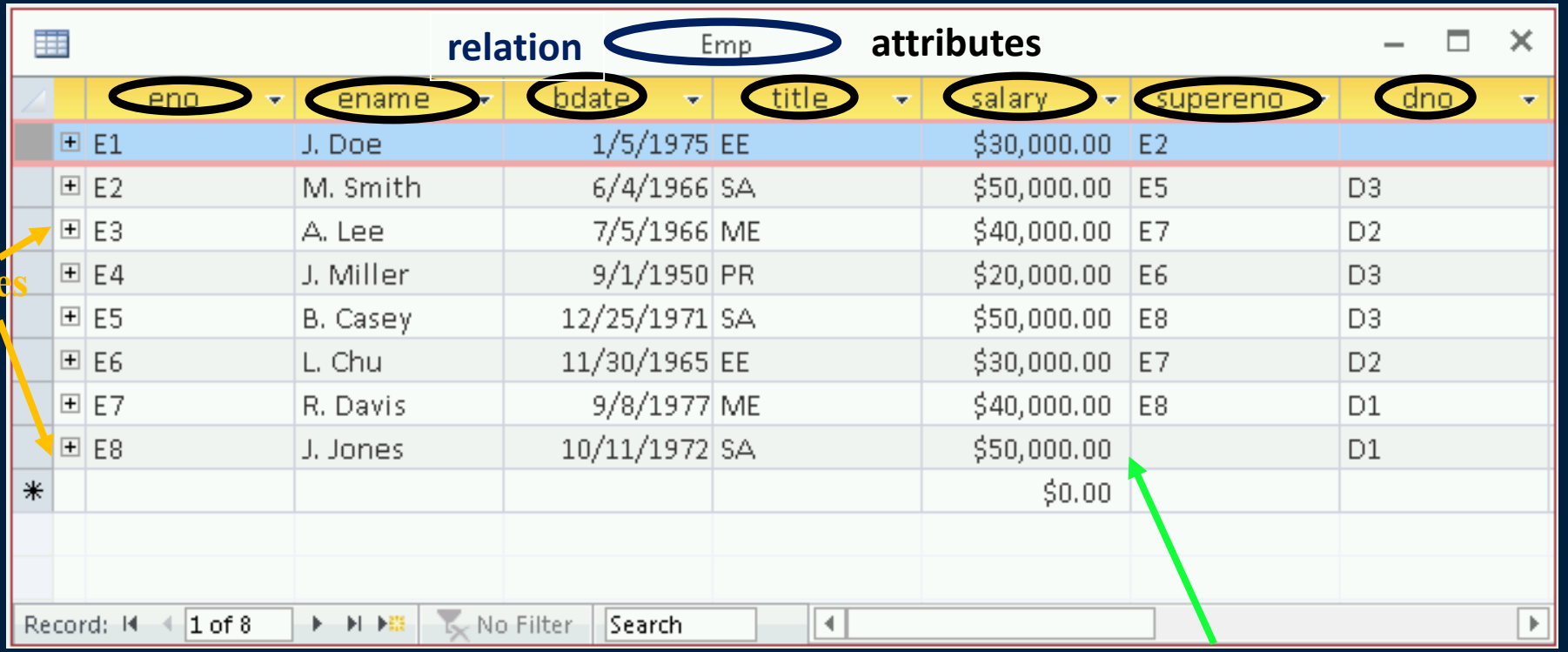
The ***degree*** of a relation is the number of attributes it contains.

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

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

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

Relation Example



eno	ename	bdate	title	salary	supereno	dno
E1	J. Doe	1/5/1975	EE	$30,000.00	E2	
E2	M. Smith	6/4/1966	SA	$50,000.00	E5	D3
E3	A. Lee	7/5/1966	ME	$40,000.00	E7	D2
E4	J. Miller	9/1/1950	PR	$20,000.00	E6	D3
E5	B. Casey	12/25/1971	SA	$50,000.00	E8	D3
E6	L. Chu	11/30/1965	EE	$30,000.00	E7	D2
E7	R. Davis	9/8/1977	ME	$40,000.00	E8	D1
E8	J. Jones	10/11/1972	SA	$50,000.00		D1
*				$0.00		

Degree =7

Cardinality = 8

Domain of salary is *currency*

Relation Practice Questions

WorksOn

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

Record: 1 of 9 No Filter Search

1) What is the name of the relation?
2) What is the cardinality of the relation?
3) What is the degree of the relation?
4) What is the domain of `resp`? What is the domain of `hours`?
5) What is larger the size of the intension or extension?
6) Is a relation's cardinality always bigger than its degree?

Relation Properties

1) No two relations have the same name.
2) Each attribute of a relation has a distinct name.
3) Each tuple is distinct. There are no duplicate tuples.
4) The order of attributes is not important.
5) The order of tuples has no significance.

Relational Keys

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

- Note that keys apply to the schema not to the data. That is, looking at the current data 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 (***candidate***) ***key*** is a *minimal* set of attributes that uniquely identifies a tuple in a relation.

- There may be more than 1 candidate key for a relation with different # of attributes.

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 a relation.

- Foreign keys enforce referential integrity. Note: A FK may refer to its own relation.

Keys and Superkeys: Test Your Understanding

1) True or false: A key is always a superkey.

2) True or false: It is possible to have more than one key for a table and the keys may have different numbers of attributes.

3) True or false: It is possible to always determine if a field is a key by looking at the data in the table.

Example Relational Data Questions

Keys are underlined

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

Proj Relation

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

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
E7	P5	Engineer	23
E8	P3	Manager	40

Questions:

1) Is *ename* a key for *emp*?
2) Is *eno* a key for *WorksOn*?
3) List all the superkeys for *WorksOn*.

Relational Integrity

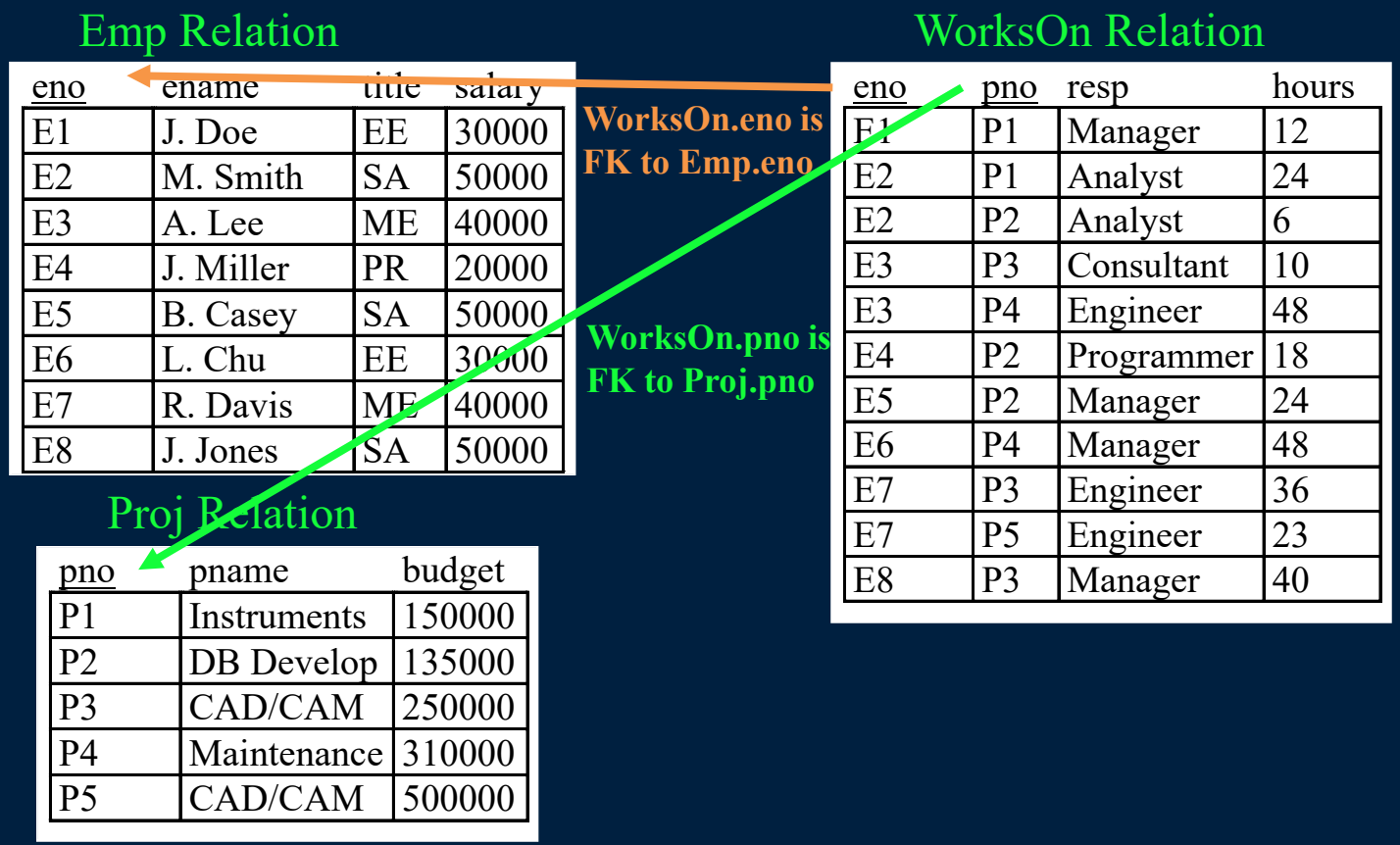
Integrity rules are used to insure the data is accurate.

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*** - 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.

Foreign Keys Example



Foreign Keys Example (2)

Proj Relation

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

Proj.dno is FK to Dept.dno

Department Relation

dno	dname
D1	Management
D2	Consulting
D3	Accounting
D4	Development

Constraints: Test Your Understanding

Question #1: A primary key has three fields. Only one field is `null`. Is the entity integrity constraint violated?

A) Yes

B) No

Question #2: A foreign key has a `null` value in the table that contains the foreign key fields. Is the referential integrity constraint violated?

A) Yes

B) No

Integrity Questions

Emp Relation

eno	ename	title	salary
E1	J. Doe	EE	AS
E2	null	SA	50000
E3	A. Lee	null	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

Proj Relation

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

WorksOn Relation

eno	pno	resp	hours
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

Question: How many rows have violations of integrity constraints? Note: salary, budget, hours are number fields.

A) 8 B) 9 C) 10 D) 11 E) 12

Relational Algebra Query Language

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.

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

Relational algebra is the foundation of all relational database systems. SQL gets translated into relational algebra.

Relational Algebra Operators

Relational Operators:

- selection σ - return subset of rows
- projection π - return subset of columns
- Cartesian product $\times$ - all combinations of two relations
- join $\bowtie$ - combines σ and $\times$
- duplicate elimination δ - eliminates duplicates

Set operators:

- Union $\cup$ - tuple in output if in either or both
- Difference - - tuple in output if in 1st but not 2nd
- Intersection $\cap$ - tuple in output if in both
- Union compatibility means relations must have the same number of columns with compatible domains.

Selection and Projection

The ***selection operation*** returns an output relation that has a subset of the tuples of the input by using a ***predicate***.

The ***projection operation*** returns an output relation that contains a subset of the attributes of the input.

Note: Duplicate tuples are eliminated.

Selection Example

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

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

Input 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

Projection Example

$\Pi_{eno,ename}(\text{Emp})$

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

Cartesian Product

The ***Cartesian (or cross) product*** of two relations R (of degree k_1) and S (of degree k_2) combines the tuples of R and S in all possible ways.

The result of $R \times S$ 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 $\times$ S is $|R| * |S|$.

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

Join

Theta (θ) join combines cross product and selection: $R \bowtie_F S = \sigma_F(R \times S)$.

An ***equijoin*** only contains the equality operator (=) in the join predicate.

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

A ***natural join*** $R \bowtie S$ is the equijoin of R and S over a set of attributes common to both R and S that removes duplicate join attributes.

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

Review Question

Given these tables and the query:

$$\Pi_{eno,\ ename}\ (\sigma_{title='EE'}(Emp \bowtie_{emp.dno=dept.dno} Dept))$$

How many rows in the result?

A) 0

B) 1

C) 2

D) 8

Dept Relation

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

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

Union

Union takes two relations *R* and *S* as input and produces an output relation that includes all tuples that are either in *R*, or in *S*, or in both *R* and *S*. Duplicate tuples are eliminated. Syntax: $R \cup S$

R and *S* must be ***union-compatible***:

- 1) Both relations have same number of attributes.
- 2) Each attribute pair, R_i and S_i, have compatible data types for all attribute indexes *i*.
- Note that attributes do not need to have the same name.
- Result has attribute names of first relation.

Union Example

Emp

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

WorksOn

eno	pno	resp	hours
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	P5	Engineer	23

$\Pi_{eno}(\text{Emp}) \cup \Pi_{eno}(\text{WorksOn})$

eno
E1
E2
E3
E4
E5
E6
E7
E8

Set Difference

Set difference takes two relations *R* and *S* as input and produces an output relation that contains all the tuples of *R* that are not in *S*.

Syntax: ***R – S***

Note:

- R – S ≠ S – R
- R and S must be union compatible.

Set Difference 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

WorksOn Relation

eno	pno	resp	hours
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	P5	Engineer	23

$\Pi_{eno}(\text{Emp}) - \Pi_{eno}(\text{WorksOn})$

eno
E4
E8

Question: What is the meaning of this query?

Question: What is $\Pi_{eno}(\text{WorksOn}) - \Pi_{eno}(\text{Emp})$?

Intersection

Intersection takes two relations R and S as input and produces an output relation which contains all tuples that are in both R and S.

- R and S must be union-compatible.

Syntax: $\boldsymbol{R \cap S}$

Note that $R \cap S = R - (R - S) = S - (S - R)$.

Intersection Example

Emp

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

WorksOn

eno	pno	resp	hours
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	P5	Engineer	23

$\Pi_{eno}(\text{Emp}) \cap \Pi_{eno}(\text{WorksOn})$

eno
E1
E2
E3
E5
E6
E7

Set Operations: Union-compatible Question

Question: Two tables have the same number of fields in the same order with the same types, but the names of some fields are different.
True or false: The two tables are union-compatible.

A) true

B) false

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*** - $R \mathbin{⟕} S$ - 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*** - $R \mathbin{⟖} S$ - 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*** - $R \mathbin{⟗} S$ - 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	hours
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 $\overset{\subset}{\bowtie}_{WorksOn.pno = Proj.pno}$ Proj

eno	pno	resp	hours	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

Outer Join Question

Question: Given this table and the query:

$$\text{WorksOn} \overset{\text{WorksOn.pno = Proj.pno}}{\supset\!\bowtie} \text{Proj}$$

How many rows are returned?

A) 10

B) 9

C) 8

D) 7

WorksOn Relation

eno	pno	resp	hours
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	P4	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

Semi-Join and Anti-Join

A ***semi-join*** between tables returns rows from the first table where one or more matches are found in the second table.

- Semi-joins are used in `EXISTS` and `IN` constructs in SQL.

An ***anti-join*** between two tables returns rows from the first table where ***no*** matches are found in the second table.

- Anti-joins are used with `NOT EXISTS`, `NOT IN`, and `FOR ALL`.
- Anti-join is the complement of semi-join: $R \triangleright S = R - R \ltimes S$

Semi-Join Example

WorksOn Relation

eno	pno	resp	hours
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

Proj $\ltimes_{Proj.pno = WorksOn.pno}$ WorksOn

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

Anti-Join Example

WorksOn Relation

eno	pno	resp	hours
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

Proj $\triangleright_{Proj.pno = WorksOn.pno}$ WorksOn

pno	pname	budget
P5	CAD/CAM	500000

Combining Operations

Relational algebra operations can be combined in one expression by nesting them:

$$\Pi_{eno,pno,hours}(\sigma_{ename='J.\ Doe'}(Emp) \bowtie \sigma_{hours>16}(WorksOn))$$

- Return the eno, pno, and hours for employee 'J. Doe' when he has worked on a project for more than 16 months.

Can also use a temporary relation variables for intermediate results.

- Use the assignment operator $\leftarrow$ for indicating that the result of an operation is assigned to a temporary relation.

$$\begin{aligned}
\text{empdoe} &\leftarrow \sigma_{ename='J.\ Doe'}(\text{Emp}) \\
\text{wohours} &\leftarrow \sigma_{hours>16}(\text{WorksOn}) \\
\text{empwo} &\leftarrow \text{empdoe} \bowtie \text{wohours} \\
\text{result} &\leftarrow \Pi_{eno,pno,hours}(\text{empwo})
\end{aligned}$$

Operator Precedence

Just like mathematical operators, the relational operators have precedence.

The precedence of operators from highest to lowest is:

- unary operators - σ, Π, ρ
- Cartesian product and joins - X, $\bowtie$, division
- intersection
- union and set difference

Parentheses can be used to changed the order of operations.

It is a good idea to ***always*** use parentheses around the argument for both unary and binary operators.

Practice Questions

Relational database schema:

branch (bname, address, city, assets)
customer (cname, street, city)
deposit (accnum, cname, bname, balance)
borrow (accnum, cname, bname, amount)

1) List the names of all branches of the bank.
2) List the names of all deposit customers together with their account numbers.
3) Find all cities where at least one customer lives.
4) Find all cities with at least one branch.
5) Find all cities with at least one branch or customer.
6) Find all cities that have a branch but no customers who live in that city.

Conclusion

The ***relational model*** represents data as relations which are sets of tuples. Each relational schema is a set of attributes with domains.

- A relation is a table with columns and rows. An attribute is a named column. A tuple is a row. Degree is the # of attributes, and cardinality is the # of tuples.

Keys are used to uniquely identify tuples in relations.

- Superkey is any set of attributes that identifies a tuple in a relational. A candidate key is a minimal set of attributes that identifies a tuple.

The relational model has ***constraints*** to guarantee data integrity including: domain, entity integrity and referential integrity constraints.

Relational algebra is a set of operations for answering queries:

- Selection (σ), projection (π), Cartesian product ($\times$), join ($\bowtie$), duplicate elimination (δ), union ($\cup$), intersection ($\cap$), set difference (-)
- These operators are used in other data models besides relational model.

Objectives

Define: relation, attribute, tuple, domain, degree, cardinality, intension, extension, null

List the properties of relations.

Define: superkey, key, candidate key, primary key, foreign key

Define: integrity, constraints, domain constraint, entity integrity constraint, referential integrity constraint

Given a relation be able to:

- identify its cardinality, degree, domains, keys, and superkeys
- determine if constraints are being violated

Define: relational algebra, query language

Define and perform all relational algebra operators.


UBC
THE UNIVERSITY OF BRITISH COLUMBIA