

Principles of Databases

Relational Data Model

David Sinclair

Relational Model Overview

- The goal of a model is to reduce and abstract the real world into something computable.
- The *Relational Data Model* was devised by Ted Codd (IBM Research) in 1970.
- A relational database is represented as a collection of tables.
- Each table represents a *relation* and related data elements.
 - S(S_#, SName, Status, City)
 - P(P_#, PName, Colour, Weight, City)
 - SP(S_#, P_#, QTY)
- Each row in a table represents a fact that corresponds to a real-world entity or relationship.

S

S#	SName	Status	City
S1	Smith	20	Paris
S2	Jones	10	Paris
S3	Blake	30	Rome

P

P#	PName	Colour	Weight	City
P1	Nut	Red	12	Dublin
P2	Bolt	Green	17	Paris
P3	Screw	Blue	27	Rome
P4	Screw	Red	14	Dublin

SP

S#	P#	QTY
S1	P1	300
S1	P2	200
S1	P3	400
S2	P1	300
S2	P2	400
S3	P2	200

Relational Model Overview (2)

- The model describes a limited world, but can describe everything in that world.
- Other related parts of the modeled world can be included.
 $\text{Makes}(\text{P_}\#, \text{M_}\#, \text{Cost})$
 $\text{M}(\text{M_}\#, \text{MName}, \text{MAddr})$
- The entire information content of the database is represented by data values with no links, pointers or offsets between tables.
- All data values are **atomic**, exactly one value and never a set.
- A *domain* is a set of values of the same data type from which one or more attributes, in one or more tables, takes their values.

Relational Model Overview (3)

D_1	D_2	\dots	D_N	
\downarrow	\downarrow	\dots	\downarrow	
A_1	A_2	\dots	A_N	
				1
				\vdots
				n

- In the above table, attribute A_1 draws its values from the domain D_1 , A_2 from D_2 , etc.
- If two attributes draw values **from the same domain, then comparisons between tuples can be made on these attributes.**
- A relation R on domain D_1 to D_N consists of:
 - a set of attributes A_1 to A_N such that A_i corresponds to D_i ; and
 - a set of N -tuples or entries in the relation.
- N , the number of attributes, is the degree and the number of tuples n is the cardinality.

Relational Model Overview (4)

- A domain normally draws its values from a data type, analogous to a programming language data type.
- In addition, a domain may or may not include the additional value, NULL, as decided at domain definition time.
- If included in a domain, the value NULL does not correspond to 0, " " or infinity. It corresponds to:
 - Not known
 - Missing
 - Does not apply
- Thus a domain of 16 bit integers has a set of $2^{16} + 1 = 65537$ unique values, and hence it requires more than 16 bits to store.

Relational Model Overview (5)

- During a database's lifetime, the cardinality changes while the degree does not.
 - Cardinality = number of rows (tuples) in the table
 - Degree = number of columns (attributes) in the table.
- A domain may appear more than once in a given relation (table), e.g. `staff_id` and `student_id`.
- Domains may be simple or composite.
 - Simple: name, age, etc.
 - Composite: date ... number, street, city ... etc.
- A relation is a set, no duplicate tuples.
 - Tuples are unordered.
 - Attributes are unordered and are referenced by name and not position.
 - There is always at least one way to uniquely address tuples, i.e. the combination of all the attributes (*key*).

Relational Model Overview (6)

- Attributes names are unique within a table and may be reused in other tables.
- Table names are unique.
- A *superkey*, SK , is a subset of attributes (a combination of columns) that uniquely identify a tuple.
 - Given two tuples t_1 and t_2 , $t_1[SK] \neq t_2[SK]$.
 - Every relation has at least one *superkey*, the set of all its attributes.
- A *superkey* usually has redundant attributes. A *key* is a set of attributes with no redundancies that uniquely identifies a tuple.
 - It is a *minimal superkey*.

Relational Model Overview (7)

- A relational DBMS (RDBMS) is a database where the data is represented as a collection of time-varying normalised relations of assorted degrees and cardinalities.
 - Tables
 - Attributes
 - Tuples
 - Domains
 - Degrees and Cardinalities
 - Keys

Relational Model Integrity

- *Integrity* is the property of a database being consistent with some predefined set of rules.
- Correct with respect to:
 - Domain values (independent)
 - Dependencies across tables

S			
S#	SName	Status	City
S1	Smith	20	Paris
S2	Jones	10	Paris
S3	Blake	30	Rome

P				
P#	PName	Colour	Weight	City
P1	Nut	Red	12	Dublin
P2	Bolt	Green	17	Paris
P3	Screw	Blue	27	Rome
P4	Screw	Red	14	Dublin

SP		
S#	P#	QTY
S1	P1	300
S1	P2	200
S1	P3	400
S2	P1	300
S2	P2	400
S3	P2	200

Relational Model Integrity (2)

- Replace S3 in the SP table for P2 shipments with the value S4.
- Suppose there is an application rule that no two suppliers can come from the same city.

S			
S#	SName	Status	City
S1	Smith	20	Paris
S2	Jones	10	Paris
S3	Blake	30	Rome

P				
P#	PName	Colour	Weight	City
P1	Nut	Red	12	Dublin
P2	Bolt	Green	17	Paris
P3	Screw	Blue	27	Rome
P4	Screw	Red	14	Dublin

SP		
S#	P#	QTY
S1	P1	300
S1	P2	200
S1	P3	400
S2	P1	300
S2	P2	400
S4	P2	200


- The relational model has built-in integrity checks for the first kind of problem, but not the second.

Relational Model Integrity (3)

- Some definitions:
 - Candidate Key* - (Combination of) attribute(s) that uniquely identify a tuple in a table.
 - Primary Key* - One of the candidate keys.
 - Alternate Key* - The candidate keys (if any) not chosen as the primary key.
 - Foreign Key* - (Combination of) attribute(s) in one relation whose value(s) are required to be equal to the primary key of another relation.
 - A foreign key is not necessarily part of the primary key.

	S				
PK for S FK for SP	S#	SName	Status	City	
	S1	Smith	20	Paris	
	S2	Jones	10	Paris	
	S3	Blake	30	Rome	
	P				
PK for P FK for SP	P#	PName	Colour	Weight	City
	P1	Nut	Red	12	Dublin
	P2	Bolt	Green	17	Paris
	P3	Screw	Blue	27	Rome
	P4	Screw	Red	14	Dublin
		SP			
	S#	P#	QTY		
	S1	P1	300		
	S1	P2	200		
	S1	P3	400		PK for SP
	S2	P1	300		
	S2	P2	400		
	S3	P2	200		

Relational Model Integrity (4)

- Entity Integrity
 - No attribute forming part of a primary key of a base table is allowed to have NULL values. 
- Referential Integrity
 - If a relation $R2$ includes a foreign key FK matching the primary key PK of some base table $R1$, then every value $R2.FK$ must:
 - be equal to the value $R1.PK$, or
 - be wholly NULL, i.e. each attribute $R2.FK$ must be NULL.
 - A base relation refers to a real-world entity or relationship, not a view.
- These rules refer to the state of the database, not to transactions.
- Other semantic rules, like “No two suppliers from the same city”, are not in the relational model.
- Most RDBMS support stopping updates that would violate the above initial two rules.

Relational Algebra

- The *Relational Algebra* is a set of operators that operate on relations, producing other relations.
- The Relational Algebra has 8 operators:
 1. *Selection*
 2. *Projection*
 3. *Product*
 4. *Union*
 5. *Intersection*
 6. *Difference*
 7. *Join*
 8. *Divide*
- Additional functions that are often used in queries and summarise data, such as COUNT, MAXIMUM, AVERAGE, are called *aggregate functions* and, though not part of the original relational algebra, they are generally included in modern relational algebras.

The *Selection* Operator

- The *selection* operator is used to select a subset of tuples from a single relation that satisfy a selection criterion.
- $$\sigma_{\langle \text{selection_condition} \rangle}(\text{relation})$$
- The argument of the selection operator is a single relation or an algebraic expression that evaluates to a single relation.
 - The selection condition is a boolean expression.
 - The selection condition compares an attribute to a constant or another attribute (if drawn from the same domain) using $=, <, \leq, >, \geq, \neq$ as comparitors and boolean connectives (\wedge, \vee, \neg) .
 - σ is unary and commutative.

S

S#	SName	Status	City
S1	Smith	20	Paris
S2	Jones	10	Paris
S3	Blake	30	Rome

$\sigma_{\langle S.S\# = S1 \rangle}(S)$

S#	SName	Status	City
S1	Smith	20	Paris

The *Selection* Operator (2)

- A series of nested selections is equivalent to a selection with a co-joined selection condition.

S

S#	SName	Status	City
S1	Smith	20	Paris
S2	Jones	10	Paris
S3	Blake	30	Rome

$\sigma_{\langle S.Status=10 \wedge S.City=Paris \rangle}(S)$

S#	SName	Status	City
S2	Jones	10	Paris

- An equivalent query is $\sigma_{\langle S.Status=10 \rangle}(\sigma_{\langle S.City=Paris \rangle}(S))$.

The *Projection* Operator

- The *projection* operator is used to select a subset of attributes (columns) from a single relation.

$\pi_{\langle \text{attribute list} \rangle}(\text{relation})$

- As with the selection operator the argument of the operator is a single relation or an algebraic expression that evaluates to a single relation.
- If the attribute list includes only non-key attributes then duplicates are could be possible and then the result would not be a relation. Hence, **projection removes duplicates**.

S

S#	SName	Status	City
S1	Smith	20	Paris
S2	Jones	10	Paris
S3	Blake	30	Rome

$\pi_{\langle SName, Status \rangle}(S)$

SName	Status
Smith	20
Jones	10
Blake	30

$\pi_{\langle City \rangle}(S)$

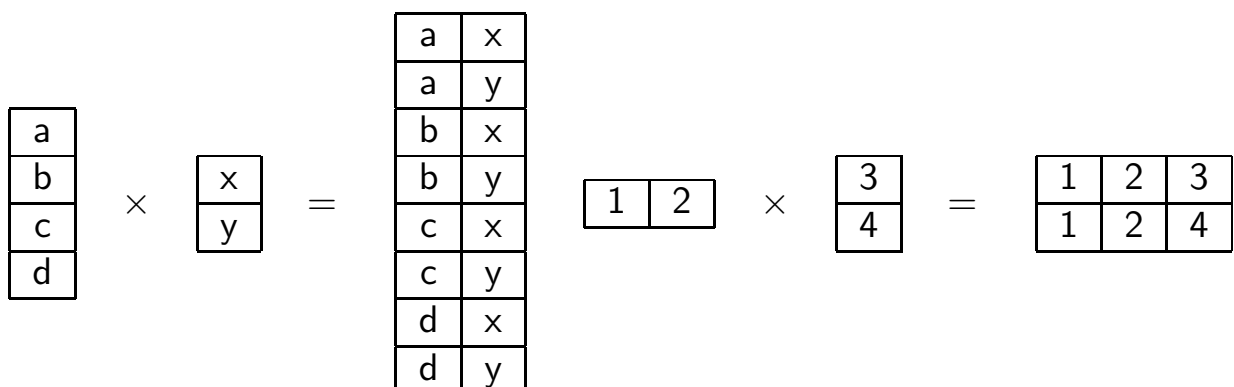
City
Paris
Rome

The *Rename* Operator

- The *Rename* operator is just a combination of the *Projection* operator and the assignment operator that allows attribute to be renamed.
- If $R(A_1, A_2, \dots, A_n)$, then $\rho_{(B_1, B_2, \dots, B_n)} R$ renames the attributes of R as B_1, B_2, \dots, B_n .
- $S = \rho_{(B_1, B_2, \dots, B_n)} R$ is equivalent to $S(B_1, B_2, \dots, B_n) = \pi_{\langle A_1, A_2, \dots, A_n \rangle} R$

The *Product* Operator

- A standard set theoretic operation is the *Cartesian Product* or *Cross Product* that combines the tuples from two relations in all possible combinations.
- Written as $R \times S$
- If R has degree n and cardinality m , and S has degree j and cardinality k , then $R \times S$ has degree $(n + j)$ and cardinality $(m * k)$.



The *Union* Operator

- Another set theoretic operation that operates on union compatible relations (same degree and domain matching).
- Denoted $R_1 \cup R_2$ and results in a relation that includes all the tuples in either R_1 or R_2 or both.
- Duplicate tuples are eliminated.
- Union is commutative and associative.
 - $R_1 \cup R_2 = R_2 \cup R_1$
 - $(R_1 \cup R_2) \cup R_3 = R_1 \cup (R_2 \cup R_3)$
- Example:

FN	LN			FName	LName			FN	LN
Susan	Yao	∪		John	Smith	=		Susan	Yao
Amy	Ford			Susan	Yao			Amy	Ford
Tony	Walsh			Francis	Gray			Tony	Walsh
								John	Smith
								Francis	Gray

The *Intersection* Operator

- Another set theoretic operation that operates on union compatible relations (same degree and domain matching).
- Denoted $R_1 \cap R_2$ and results in a relation that includes all the tuples in both R_1 and R_2 .
- Duplicate tuples are eliminated.
- Intersection is commutative and associative.
 - $R_1 \cap R_2 = R_2 \cap R_1$
 - $(R_1 \cap R_2) \cap R_3 = R_1 \cap (R_2 \cap R_3)$
- Example:

FN	LN			FName	LName			FN	LN
Susan	Yao	∩		John	Smith	=		Susan	Yao
Amy	Ford			Susan	Yao				
Tony	Walsh			Francis	Gray				

The *Difference* Operator

- Another set theoretic operation that operates on union compatible relations (same degree and domain matching).
- Denoted $R_1 - R_2$ and results in a relation that includes all the tuples in R_1 but not in R_2 .
- Duplicate tuples are not an issue.
- Difference is not commutative.
 - $R_1 - R_2 \neq R_2 - R_1$

- Example:

FN	LN	-	FName	LName	=	FN	LN
Susan	Yao		John	Smith		Amy	Ford
Amy	Ford		Susan	Yao		Tony	Walsh
Tony	Walsh		Francis	Gray			

The *Join* Operator

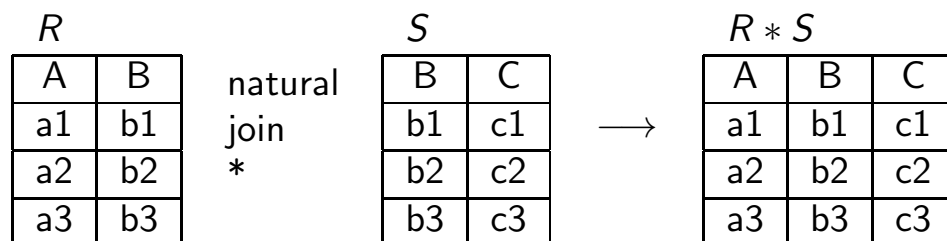
- The *Join* operator is a binary operator that is used to combine tuples from two relations where the tuples are related by some join expression.

R			S			$R \bowtie S$			
A	B		B	C		A	B	B	C
a1	b1	join \bowtie	b1	c1	\longrightarrow	a1	b1	b1	c1
a2	b2		b2	c2		a2	b2	b2	c2
a3	b3		b3	c3		a3	b3	b3	c3

- The *Join* operation is written:
 - $R \bowtie_{\langle \text{join condition} \rangle} S$
 - If $R(A_1, A_2, \dots, A_n)$ and $S(B_1, B_2, \dots, B_m)$ then $R \bowtie_{\langle \text{join condition} \rangle} S$ is a relation with $n + m$ attributes $(A_1, A_2, \dots, A_n, B_1, B_2, \dots, B_m)$ in that order.
- *Join* is **commutative and associative**.
- All tuples in R and S that satisfy the *join condition* are included in the resulting relationship.

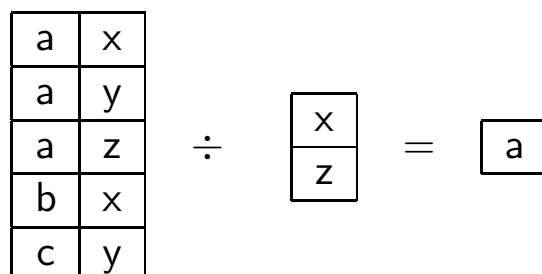
The *Join* Operator (2)

- There are a number of different types of *Joins*
 - A *theta join*, θ -join is a *Join*, where the *join condition* contains $>, \leq, >, \geq, =, \neq$ as comparitors, possibly augmented by binary logical connectives such as \wedge, \vee, \neg .
 - An *equijoin* is a *Join* where the *join condition* only contains $=$ as a comparitor, possibly augmented by binary logical connectives such as \wedge, \vee, \neg . It is a special case of a θ -join.
 - A *Natural Join*, denoted $R * S$, is a join on attributes with the same name, same values and with the duplicate attributes removed.



The *Divide* Operator

- The *Divide* (or *Division*) operator is another binary operation on relations. It is denoted by $R \div S$ and applies only when the set of attributes of S is a subset of the set of attributes of R .
- $R(\mathcal{Z}) \div S(\mathcal{X})$, where $\mathcal{X} \subseteq \mathcal{Z}$, yields a relation $T(\mathcal{Y})$ where $\mathcal{Y} = \mathcal{Z} - \mathcal{X}$.
- For a tuple to appear in $R(\mathcal{Z}) \div S(\mathcal{X})$, the value in that tuple must appear in R in combination with every tuple in S .
 - $R \div S$ gives attribute values that when paired with S can be found in R .
- The degree and cardinality of the divisor S should be small.
 - Otherwise an empty relation could result.



The Relational Algebra Again

- The set of operators {Select, Project, Product, Union, Difference} or $\{\sigma, \pi, \times, \cup, -\}$ are *primitive* operators in that the other three operators can be defined in terms of these *primitive* operators.
 - $R \cap S = (R \cup S) - ((R - S) \cup (S - R))$
 - $R \bowtie_{\langle condition \rangle} S = \sigma_{\langle condition \rangle}(R \times S)$
 - $R \div S = \pi_Y(R) - \pi_Y(S \times \pi_Y(R) - R)$

Query 1

Consider the following relations:

S

S#	SName	Status	City
S1	Smith	20	Paris
S2	Jones	10	Paris
S3	Blake	30	Rome

P

P#	PName	Colour	Weight	City
P1	Nut	Red	12	Dublin
P2	Bolt	Green	17	Paris
P3	Screw	Blue	27	Rome
P4	Screw	Red	14	Dublin

SP

S#	P#	QTY
S1	P1	300
S1	P2	200
S1	P3	400
S2	P1	300
S2	P2	400
S3	P2	200

Retrieve the names and status of suppliers who supply 300 units of any part.

Query 1 (2)

Query 2

Retrieve the colours of the parts supplied by either S1 or S2.

Query 2 (2)

Query 3

Retrieve the name and city of suppliers who supply any kind of part that is either green or made in Paris.

Query 3 (2)