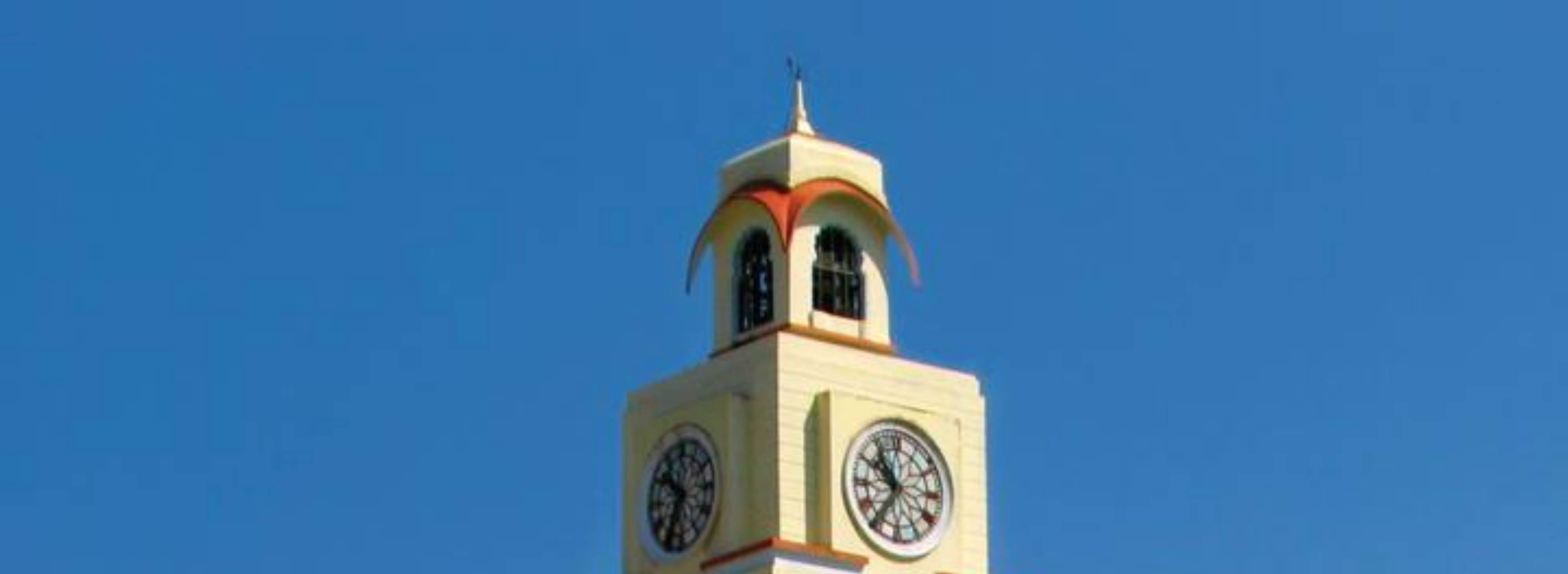
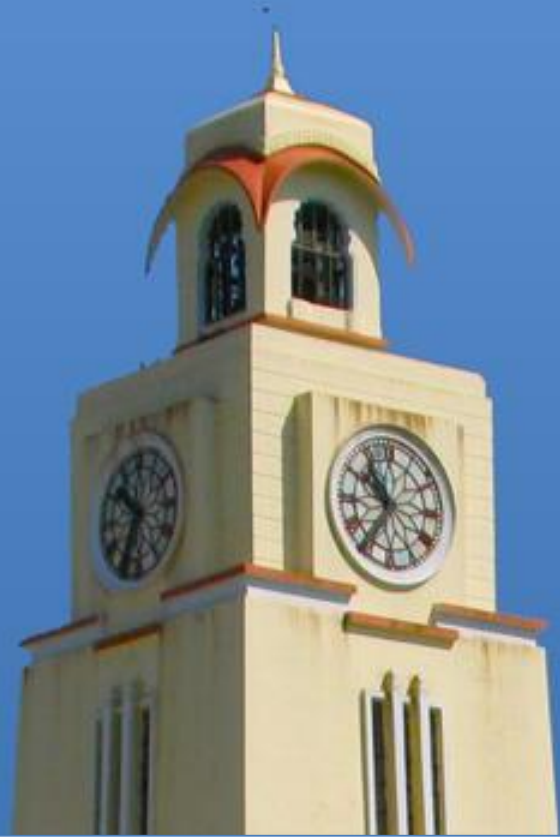


BITS Pilani

BITS Pilani
Hyderabad Campus

Dr. Manik Gupta
Associate Professor
Department of CSIS



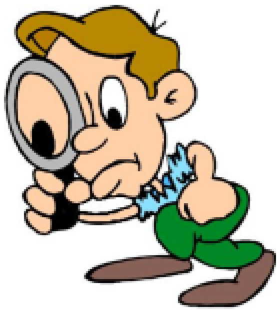
BITS Pilani
Hyderabad Campus

Distributed Data Systems (CS G544)
Lecture 5

Tuesday, 20th August 2024

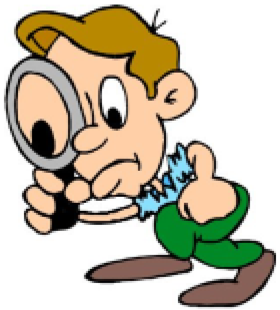
Lecture recap

- Architectures of distributed systems
  - Software and system architectures



Today's agenda

- Review of RDBMS concepts



Why relational model?

- Mathematical foundation of relational model makes it a good candidate for theoretical treatment
- Problems are easier to formulate
- RDBMS market is matured and sizable
- Most distributed database systems are relational

Features of relational model

- Data structures are simple
- Solid foundation to data consistency
    - Normalization aids database design to take care of various anomalies
    - Integrity rules help define and maintain database consistency
- Set orientation manipulation of relations leading to languages based on
    - Set theory (relational algebra)
    - Logic (relational calculus)

Relational Model

- A **database** is a structured collection of data related to some real-life phenomena that we are trying to model.
- A **relational database** is one where the database structure is in the form of tables.
- Formally, a **relation** R defined over n sets D_1, D_2….. D_n (not necessarily distinct) is a set of n-tuples (or simply tuples) $<d_1, d_2 \ldots .. d_n>$ such that $d_1 \in D_1$, $d_2 \in D_2 \ldots$ $d_n \in D_n$

Relation Schemes and Instances

Relational scheme

- A relation scheme is the definition; i.e., a set of attributes
- A relational database scheme is a set of relation schemes:
  - i.e., a set of sets of attributes

Relation instance (simply *relation*)

- An relation is an instance of a relation scheme

Can many instances be generated from one relation scheme?

Relation Schemes

Lets consider a database that models a manufacturing company

EMP

<u>ENO</u>	ENAME	TITLE	SAL	<u>PNO</u>	RESP	DUR

PROJ

<u>PNO</u>	PNAME	BUDGET

EMP(<u>ENO</u>, ENAME, TITLE, SAL, <u>PNO</u>, RESP, DUR)

PROJ (<u>PNO</u>, PNAME, BUDGET)

Underlined attributes are **relation keys.**

What are keys?

- **Minimum non-empty subset** of its attributes such that the values of the attributes comprising the key **uniquely identify each tuple of the relation**.

Example Relation Instances

EMP

ENO	ENAME	TITLE	SAL	PNO	RESP	DUR
E1	J. Doe	Elect. Eng.	40000	P1	Manager	12
E2	M. Smith	Analyst	34000	P1	Analyst	24
E2	M. Smith	Analyst	34000	P2	Analyst	6
E3	A. Lee	Mech. Eng.	27000	P3	Consultant	10
E3	A. Lee	Mech. Eng.	27000	P4	Engineer	48
E4	J. Miller	Programmer	24000	P2	Programmer	18
E5	B. Casey	Syst. Anal.	34000	P2	Manager	24
E6	L. Chu	Elect. Eng.	40000	P4	Manager	48
E7	R. Davis	Mech. Eng.	27000	P3	Engineer	36
E8	J. Jones	Syst. Anal.	34000	P3	Manager	40

PROJ

PNO	PNAME	BUDGET
P1	Instrumentation	150000
P2	Database Develop.	135000
P3	CAD/CAM	250000
P4	Maintenance	310000

Normalization

- Normalization is a step by step reversible process of replacing a given collection of relations by successive relations in which relations have a progressively **simpler and more regular structure.**
- The aim of normalization is to **eliminate various anomalies** (or undesirable aspects) of a relation in order to obtain “better” relations.
- Several problems can exist in a relation scheme.

Repetition Anomaly

- The NAME,TITLE, SAL attribute values are repeated for **each project** that the employee is involved in.
  - Waste of space
  - Complicates updates

EMP

ENO	ENAME	TITLE	SAL	PNO	RESP	DUR
E1	J. Doe	Elect. Eng.	40000	P1	Manager	12
E2	M. Smith	Analyst	34000	P1	Analyst	24
E2	M. Smith	Analyst	34000	P2	Analyst	6
E3	A. Lee	Mech. Eng.	27000	P3	Consultant	10
E3	A. Lee	Mech. Eng.	27000	P4	Engineer	48
E4	J. Miller	Programmer	24000	P2	Programmer	18
E5	B. Casey	Syst. Anal.	34000	P2	Manager	24
E6	L. Chu	Elect. Eng.	40000	P4	Manager	48
E7	R. Davis	Mech. Eng.	27000	P3	Engineer	36
E8	J. Jones	Syst. Anal.	34000	P3	Manager	40

Update Anomaly

- If any attribute of project (say SAL of an employee) is updated, multiple tuples have to be updated to reflect the change.

EMP

ENO	ENAME	TITLE	SAL	PNO	RESP	DUR
E1	J. Doe	Elect. Eng.	40000	P1	Manager	12
E2	M. Smith	Analyst	34000	P1	Analyst	24
E2	M. Smith	Analyst	34000	P2	Analyst	6
E3	A. Lee	Mech. Eng.	27000	P3	Consultant	10
E3	A. Lee	Mech. Eng.	27000	P4	Engineer	48
E4	J. Miller	Programmer	24000	P2	Programmer	18
E5	B. Casey	Syst. Anal.	34000	P2	Manager	24
E6	L. Chu	Elect. Eng.	40000	P4	Manager	48
E7	R. Davis	Mech. Eng.	27000	P3	Engineer	36
E8	J. Jones	Syst. Anal.	34000	P3	Manager	40

Insertion Anomaly

- When a new employee joins the company, we cannot add personal information (name, title, salary) to the EMP relation unless an appointment to a project is made.

EMP

ENO	ENAME	TITLE	SAL	PNO	RESP	DUR
E1	J. Doe	Elect. Eng.	40000	P1	Manager	12
E2	M. Smith	Analyst	34000	P1	Analyst	24
E2	M. Smith	Analyst	34000	P2	Analyst	6
E3	A. Lee	Mech. Eng.	27000	P3	Consultant	10
E3	A. Lee	Mech. Eng.	27000	P4	Engineer	48
E4	J. Miller	Programmer	24000	P2	Programmer	18
E5	B. Casey	Syst. Anal.	34000	P2	Manager	24
E6	L. Chu	Elect. Eng.	40000	P4	Manager	48
E7	R. Davis	Mech. Eng.	27000	P3	Engineer	36
E8	J. Jones	Syst. Anal.	34000	P3	Manager	40

Deletion Anomaly

- If an employee works on only one project, and that project is terminated, it is not possible to delete the project information from the EMP relation. To do so would result in deleting the only tuple about the employee, thereby resulting in the loss of personal information we might want to retain.

EMP

ENO	ENAME	TITLE	SAL	PNO	RESP	DUR
E1	J. Doe	Elect. Eng.	40000	P1	Manager	12
E2	M. Smith	Analyst	34000	P1	Analyst	24
E2	M. Smith	Analyst	34000	P2	Analyst	6
E3	A. Lee	Mech. Eng.	27000	P3	Consultant	10
E3	A. Lee	Mech. Eng.	27000	P4	Engineer	48
E4	J. Miller	Programmer	24000	P2	Programmer	18
E5	B. Casey	Syst. Anal.	34000	P2	Manager	24
E6	L. Chu	Elect. Eng.	40000	P4	Manager	48
E7	R. Davis	Mech. Eng.	27000	P3	Engineer	36
E8	J. Jones	Syst. Anal.	34000	P3	Manager	40

What to do?

- Normalization transforms arbitrary relation schemes into ones without these problems.
  - **Top-down methodology** for producing a schema by subsequent **refinements and decompositions**.
- Take each relation individually and "improve" it in terms of the desired characteristics
  - The relation with one or more of the anomalies is split into two or more relations of a higher normal form.

Normalization Issues

- What criteria should the decomposed schemas follow in order to preserve the semantics of the original schema?
  - **Lossless decomposition**
    - No information loss and possible to join the decomposed relations to obtain the original relation
  - **Dependency preservation**
    - Constraints (i.e., dependencies) that hold on the original relation should be enforceable by means of the constraints (i.e., dependencies) defined on the decomposed relations.

Normal forms

- A relation is said to be in a normal form if it satisfies the conditions associated with that normal form.
- Codd initially defined the first, second, and third normal forms (1NF, 2NF, and 3NF, respectively).
- Boyce and Codd later defined a modified version of the third normal form, commonly known as the Boyce-Codd normal form (BCNF). This was followed by the definition of the fourth (4NF) and fifth normal forms (5NF).

Normal forms

- The normal forms are based on certain **dependency structures.**
  - BCNF and lower normal forms are based on **functional dependencies** (FDs)
  - 4NF is based on multivalued dependencies
  - 5NF is based on projection-join dependencies

Watch the video on normalization
https://www.youtube.com/watch?v=UrYLYV7WSHM

Functional Dependence

Let R be a relation defined over the set of attributes $A = \{A_1, A_2, \ldots, A_n\}$ and let $X \subset A$, $Y \subset A$. If for each value of X in R, there is only one associated Y value, we say that "X *functionally determines* Y" or that "Y is *functionally dependent* on X." Notationally, this is shown as $X \to Y$. The key of a relation functionally determines the non-key attributes of the same relation.

- Valid FD's
  - In relation EMP
    - (ENO, PNO) $\to$ (ENAME, TITLE, SAL, DUR, RESP)
  - In relation PROJ
    - PNO $\to$ (PNAME, BUDGET)
- If each employee is given unique employee numbers, then
  - ENO $\to$ (ENAME, TITLE, SAL)
  - (ENO, PNO) $\to$ (RESP, DUR)
- It may also happen that the salary for a given position is fixed, which gives rise to the FD
  - TITLE $\to$ SAL

Normalized Relations – Example

The following set of relation schemes are normalized with respect to the functional dependencies defined over the relations.

EMP

ENO	ENAME	TITLE
E1	J. Doe	Elect. Eng
E2	M. Smith	Syst. Anal.
E3	A. Lee	Mech. Eng.
E4	J. Miller	Programmer
E5	B. Casey	Syst. Anal.
E6	L. Chu	Elect. Eng.
E7	R. Davis	Mech. Eng.
E8	J. Jones	Syst. Anal.

ASG

ENO	PNO	RESP	DUR
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
E8	P3	Manager	40

PROJ

PNO	PNAME	BUDGET
P1	Instrumentation	150000
P2	Database Develop.	135000
P3	CAD/CAM	250000
P4	Maintenance	310000

PAY

TITLE	SAL
Elect. Eng.	40000
Syst. Anal.	34000
Mech. Eng.	27000
Programmer	24000

Relational Data languages

- Query languages for the relational model fall into two fundamental groups
  - Relational algebra languages
  - Relational calculus languages
- The **relational algebra** is **procedural** in that the user is expected to specify, using certain high-level operators, how the result is to be obtained.
- The **relational calculus**, on the other hand, is **non-procedural**; the user only specifies the relationships that should hold in the result.

Why Relational algebra is important?

- **Relational algebra** defined the basic set of operations for the formal relational model.
- These operations enable a user to specify basic retrieval requests as **relational algebra expressions.**
  - Provides a **formal foundation** for relational model operations
  - Basis for **implementing and optimizing queries** in the query processing and optimization modules that are integral parts of relational database management systems
  - Some of its concepts are incorporated into the SQL standard query language for RDBMSs, core operations and functions in the **internal modules of most relational systems are based on relational algebra operations.**

Why Relational Calculus is important?

- There is no order of operations to specify how to retrieve the query result—only **what information the result should contain.**
- Firm basis in **mathematical logic** and because the standard query language (SQL) for RDBMSs has some of its foundations in a variation of relational calculus known as the **tuple relational calculus**.

Relational Algebra

- Specify how to obtain the result using a set of operators that operate on relations

$$\langle \mathit{Operator} \rangle_{\langle \mathit{parameters} \rangle} \underset{\substack{\downarrow \\ \text{Relation (s)}}}{\langle \mathit{Operands} \rangle} \rightarrow \underset{\substack{\downarrow \\ \text{Relation}}}{\langle \mathit{Result} \rangle}$$

- Each operator takes one or two relations as operands and produces a result relation, which, in turn, may be an operand to another operator. These operations permit the querying and updating of a relational database.

Relational Algebra Operators

Fundamental
- Selection
- Projection
- Union
- Set difference
- Cartesian product

Additional
- Intersection
- θ-join
- Natural join
- Semijoin
- Division

Selection

Produces a horizontal subset of the operand relation

General form

$$\sigma_F(R)=\{t \mid t \in R \text{ and } F(t) \text{ is true}\}$$

where

- *R* is a relation, *t* is a tuple variable
- *F* is a formula consisting of
  - operands that are constants or attributes
  - arithmetic comparison operators

    $<, >, =, \neq, \leq, \geq$
  - logical operators

    $\wedge, \vee, \neg$

Selection Example

EMP

ENO	ENAME	TITLE
E1	J. Doe	Elect. Eng.
E2	M. Smith	Syst. Anal.
E3	A. Lee	Mech. Eng.
E4	J. Miller	Programmer
E5	B. Casey	Syst. Anal.
E6	L. Chu	Elect. Eng.
E7	R. Davis	Mech. Eng.
E8	J. Jones	Syst. Anal.

$\sigma_{\text{TITLE='Elect. Eng.'}}(\text{EMP})$

ENO	ENAME	TITLE
E1	J. Doe	Elect. Eng
E6	L. Chu	Elect. Eng.

Projection

Produces a vertical slice of a relation

General form

$$\Pi_{A_1,\ldots,A_n}(R)=\{t[A_1,\ldots,A_n] \mid t\in R\}$$

where

- *R* is a relation, *t* is a tuple variable
- $\{A_1,\ldots,A_n\}$ is a subset of the attributes of *R* over which the projection will be performed

Note: projection can generate duplicate tuples. Commercial systems (and SQL) allow this and provide

- Projection with duplicate elimination
- Projection without duplicate elimination

Projection Example

PROJ

PNO	PNAME	BUDGET
P1	Instrumentation	150000
P2	Database Develop.	135000
P3	CAD/CAM	250000
P4	Maintenance	310000

$\Pi_{PNO,BUDGET}(PROJ)$

PNO	BUDGET
P1	150000
P2	135000
P3	250000
P4	310000

Union

Similar to set union

General form

$$R \cup S = \{t \mid t \in R \text{ or } t \in S\}$$

where *R*, *S* are relations, *t* is a tuple variable

- Result contains tuples that are in *R* or in *S*, but not both (duplicates removed)
- *R*, *S* should be union-compatible

Set Difference

General Form

$$R - S = \{t \mid t \in R \text{ and } t \notin S\}$$

where R and S are relations, t is a tuple variable

- Result contains all tuples that are in R, but not in S.
- $R - S \neq S - R$ (Asymmetric)
- R, S union-compatible

Cartesian (Cross) Product

Given relations

- R of degree k_1, cardinality n_1
- S of degree k_2, cardinality n_2

Cartesian (cross) product:

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

The result of $R \times S$ is a relation of degree $(k_1 + k_2)$ and consists of all $(n_1 * n_2)$-tuples where each tuple is a concatenation of one tuple of R with one tuple of S.

What happens when two relations have attributes with the same name?

Cartesian Product Example

EMP

ENO	ENAME	TITLE
E1	J. Doe	Elect. Eng
E2	M. Smith	Syst. Anal.
E3	A. Lee	Mech. Eng.
E4	J. Miller	Programmer
E5	B. Casey	Syst. Anal.
E6	L. Chu	Elect. Eng.
E7	R. Davis	Mech. Eng.
E8	J. Jones	Syst. Anal.

PAY

TITLE	SALARY
Elect. Eng.	55000
Syst. Anal.	70000
Mech. Eng.	45000
Programmer	60000

EMP × PAY

ENO	ENAME	EMP.TITLE	PAY.TITLE	SALARY
E1	J. Doe	Elect. Eng.	Elect. Eng.	55000
E1	J. Doe	Elect. Eng.	Syst. Anal.	70000
E1	J. Doe	Elect. Eng.	Mech. Eng.	45000
E1	J. Doe	Elect. Eng.	Programmer	60000
E2	M. Smith	Syst. Anal.	Elect. Eng.	55000
E2	M. Smith	Syst. Anal.	Syst. Anal.	70000
E2	M. Smith	Syst. Anal.	Mech. Eng.	45000
E2	M. Smith	Syst. Anal.	Programmer	60000
E3	A. Lee	Mech. Eng.	Elect. Eng.	55000
E3	A. Lee	Mech. Eng.	Syst. Anal.	70000
E3	A. Lee	Mech. Eng.	Mech. Eng.	45000
E3	A. Lee	Mech. Eng.	Programmer	60000
E8	J. Jones	Syst. Anal.	Elect. Eng.	55000
E8	J. Jones	Syst. Anal.	Syst. Anal.	70000
E8	J. Jones	Syst. Anal.	Mech. Eng.	45000
E8	J. Jones	Syst. Anal.	Programmer	60000

Intersection

Typical set intersection

$$R \cap S = \{t \mid t \in R \text{ and } t \in S\}$$

$$= R - (R - S)$$

R, *S* union-compatible

θ–Join

General form

$R \bowtie_{F(R.A_i,S.B_j)} S=\{t[A_1,\ldots,A_n,B_1,\ldots,B_m] \mid$

$t[A_1,\ldots,A_n] \in R$ and $t[B_1,\ldots,B_m] \in S$

and $F(R.A_i, S.B_j)$ is true$\}$

where

- R, S are relations, t is a tuple variable
- $F(R.A_i, S.B_j)$ is a formula specifying join predicate

The join of two relations is equivalent to performing a selection, using the join predicate as the selection formula, over the Cartesian product of the two operand relations.

- $R \bowtie_F S = \sigma_F(R \times S)$

Join Example

EMP

ENO	ENAME	TITLE
E1	J. Doe	Elect. Eng
E2	M. Smith	Syst. Anal.
E3	A. Lee	Mech. Eng.
E4	J. Miller	Programmer
E5	B. Casey	Syst. Anal.
E6	L. Chu	Elect. Eng.
E7	R. Davis	Mech. Eng.
E8	J. Jones	Syst. Anal.

ASG

ENO	PNO	RESP	DUR
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
E8	P3	Manager	40

$EMP \bowtie_{EMP.ENO=ASG.ENO} ASG$

ENO	ENAME	TITLE	PNO	RESP	DUR
E1	J. Doe	Elect. Eng.	P1	Manager	12
E2	M. Smith	Syst. Anal.	P1	Analyst	12
E2	M. Smith	Syst. Anal.	P2	Analyst	12
E3	A. Lee	Mech. Eng.	P3	Consultant	12
E3	A. Lee	Mech. Eng.	P4	Engineer	12
E4	J. Miller	Programmer	P2	Programmer	12
E5	J. Miller	Syst. Anal.	P2	Manager	12
E6	L. Chu	Elect. Eng.	P4	Manager	12
E7	R. Davis	Mech. Eng.	P3	Engineer	12
E8	J. Jones	Syst. Anal.	P3	Manager	12

Types of Join

Equi join

- The formula *F* only contains equality
- $R \bowtie_{R.A=S.B} S$

Natural join

- Equi-join of two relations *R* and *S* over an attribute (or attributes) common to both *R* and *S* and projecting out one copy of those attributes

Natural Join Example

EMP

ENO	ENAME	TITLE
E1	J. Doe	Elect. Eng
E2	M. Smith	Syst. Anal.
E3	A. Lee	Mech. Eng.
E4	J. Miller	Programmer
E5	B. Casey	Syst. Anal.
E6	L. Chu	Elect. Eng.
E7	R. Davis	Mech. Eng.
E8	J. Jones	Syst. Anal.

PAY

TITLE	SALARY
Elect. Eng.	55000
Syst. Anal.	70000
Mech. Eng.	45000
Programmer	60000

EMP $\bowtie$ PAY

ENO	ENAME	TITLE	SALARY
E1	J. Doe	Elect. Eng.	55000
E2	M. Smith	Analyst	70000
E3	A. Lee	Mech. Eng.	45000
E4	J. Miller	Programmer	60000
E5	B. Casey	Syst. Anal.	70000
E6	L. Chu	Elect. Eng.	55000
E7	R. Davis	Mech. Eng.	45000
E8	J. Jones	Syst. Anal.	70000

Join is over the common attribute TITLE

Semijoin

- The semijoin of relation R, defined over the set of attributes A, by relation S, defined over the set of attributes B, is the subset of the tuples of R that participate in the join of R with S.

$$R \ltimes_F S = \Pi_A(R \bowtie_F S)$$

where

R, S are relations

A is a set of attributes

What are the advantages of semijoins?

Semijoin Example

$EMP \ltimes_{EMP.TITLE=PAY.TITLE} PAY$

ENO	ENAME	TITLE
E1	J. Doe	Elect. Eng.
E2	M. Smith	Analyst
E3	A. Lee	Mech. Eng.
E4	J. Miller	Programmer
E5	B. Casey	Syst. Anal.
E6	L. Chu	Elect. Eng.
E7	R. Davis	Mech. Eng.
E8	J. Jones	Syst. Anal.

Relational Algebra Operations

OPERATION	PURPOSE	NOTATION
SELECT	Selects all tuples that satisfy the selection condition from a relation R.	$\sigma_{\text{<selection condition>}}(R)$
PROJECT	Produces a new relation with only some of the attributes of R, and removes duplicate tuples.	$\pi_{\text{<attribute list>}}(R)$
THETA JOIN	Produces all combinations of tuples from R_1 and R_2 that satisfy the join condition.	$R_1 \bowtie_{\text{<join condition>}} R_2$
EQUIJOIN	Produces all the combinations of tuples from R_1 and R_2 that satisfy a join condition with only equality comparisons.	$R_1 \bowtie_{\text{<join condition>}} R_2$, OR $R_1 \bowtie_{(\text{<join attributes 1>}),(\text{<join attributes 2>})} R_2$
NATURAL JOIN	Same as EQUIJOIN except that the join attributes of R_2 are not included in the resulting relation; if the join attributes have the same names, they do not have to be specified at all.	$R_1 *_{\text{<join condition>}} R_2$, OR $R_1 *_{(\text{<join attributes 1>}),(\text{<join attributes 2>})} R_2$ OR $R_1 * R_2$
UNION	Produces a relation that includes all the tuples in R_1 or R_2 or both R_1 and R_2; R_1 and R_2 must be union compatible.	$R_1 \cup R_2$
INTERSECTION	Produces a relation that includes all the tuples in both R_1 and R_2; R_1 and R_2 must be union compatible.	$R_1 \cap R_2$
DIFFERENCE	Produces a relation that includes all the tuples in R_1 that are not in R_2; R_1 and R_2 must be union compatible.	$R_1 - R_2$
CARTESIAN PRODUCT	Produces a relation that has the attributes of R_1 and R_2 and includes as tuples all possible combinations of tuples from R_1 and R_2.	$R_1 \times R_2$

Tuple Relational Calculus

- When we write a relational-algebra expression, we provide a **sequence of procedures** that generates the answer to our query.
- The tuple relational calculus, by contrast, is a **nonprocedural query language**. It describes the desired information without giving a specific procedure for obtaining that information.

Tuple Relational Calculus

Query are specified as $\{t|F\{t\}\}$ where

- t is a tuple variable
- F is a well-formed formula

Atomic formula are of two forms:

- Tuple-variable membership expressions
  - $R.t$ or $R(t)$: tuple t belongs to relation R
- Conditions
  - $s[A]\ \theta\ t[B]$; s and t are tuple variables, A and B are attributes of s and t, respectively, $\theta \in \{<, >, =, \neq, \leq, \geq\}$; e.g., s[SAL] > t[SAL]
  - $s[A]\ \theta\ c$; s, A, and θ as defined above, c is a constant; e.g., s[ENAME] = 'Smith'

Domain Relational Calculus

- A second form of relational calculus, called domain relational calculus, uses **domain variables** that take on values from an attributes domain, rather than values for an entire tuple.
- Domain relational language uses domain variables that range over the values of a domain and specifies a component of a tuple
- Query are of the form $x_1, x_2, \ldots, x_n | F(x_1, x_2, \ldots, x_n)$ where
  – $x_1, x_2, \ldots, x_n$ are domain variables
  – F is a well-formed formula

Examples

Find the ID, name, dept name, salary for instructors whose salary is greater than $80,000.

- $\{t \mid t \in instructor \land t[salary] > 80000\}$

Find all instructor ID for instructors whose salary is greater than $80,000:

- $\{t \mid \exists\ s \in instructor\ (t[ID] = s[ID] \land s[salary] > 80000)\}$

Lecture summary

Topics covered

- Review of RDMS concepts
  - Normalization
  - Relational Data Languages
    - Relational Algebra
    - Relational Calculus

Essential Readings

- Chapter 2 Tamer Ozsu

Thanks…

Next Lecture

- Introduction to Distributed databases
- Distributed DBMS Architecture

Questions??