Database Design

CMSC424 Lecture Notes

Nick Roussopoulos

Department of Computer Science
University of Maryland
Spring 2017

1

Database Management System (DBMS)

- DBMS contains information about a particular enterprise
  - Collection of interrelated data
  - Set of programs to access and manipulate the data
  - An environment that is both *convenient* and *efficient* to use
- Database Applications:
  - Banking: transactions
  - Airlines: reservations, schedules
  - Universities: registration, grades
  - Sales: customers, products, purchases
  - Online retailers: order tracking, customized recommendations
  - Manufacturing: production, inventory, orders, supply chain
  - Human resources: employee records, salaries, tax deductions
- Databases can be very large or huge (exabytes =10^18 Bytes)
- Databases touch all aspects of our lives

2

University Database Example

- Application program example
  - Add new students, instructors, and courses
  - Register students for courses, and generate class rosters
  - Assign grades to students, compute grade point averages (GPA) and generate transcripts

Early days: Used files to store databases

- Prior to 70's, database applications were built directly on top of file systems
- Data redundancy and inconsistency
  - Multiple file copies in a variety of data structures
  - duplication of information in different files
- Difficulty in accessing data
  - Need to write a new program to carry out a new task
  - Programmers kept their own files (hard for others to access)
- Integrity problems
  - Integrity constraints (e.g., account balance > 0) become "buried" in program code rather than being stated explicitly
  - Hard to add new constraints or change existing ones

Drawbacks of using files to store data (cont.)

- Atomicity of updates
  - Failures may leave database in an inconsistent state with partial updates carried out

    Example: Transfer of funds from one account to another should either complete or not happen at all

- Concurrent access by multiple users
  - Concurrent access needed for performance
  - Uncontrolled concurrent accesses can lead to inconsistencies

    Example: Two people reading a balance (say 100) and updating it by withdrawing money (say 50 each) at the same time

- Security problems
  - Hard to provide user access to some, but not all, data

Database systems offer solutions to all the above problems

 5

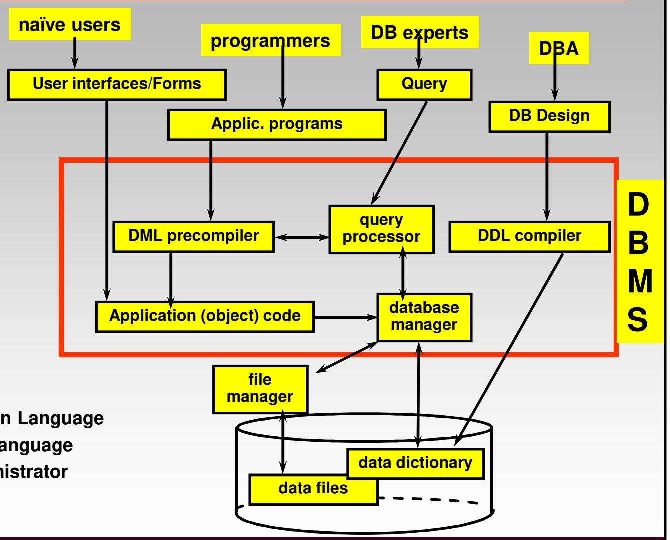
Reasons for Using a DBMS

- provides data independence
  - avoids hard wiring
- reduces data redundancy and inconsistency
- allows concurrent multi-user access
- provides crash recovery
- provides security of data
- protects the data from being corrupted
  - maintains integrity

6

Architecture of a DBMS

System Components
& Interfaces



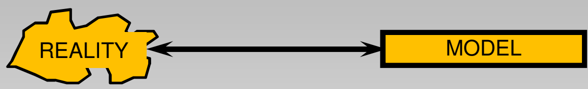
- DML: Data Manipulation Language
- DDL: Data Definition Language
- DBA: Data Base Administrator

Data Models

- A collection of tools for describing
  - Data
  - Data relationships
  - Data semantics
  - Data constraints
- Relational model
- Entity-Relationship data model (mainly for database design)
- Object-based data models (Object-oriented and Object-relational)
- Semistructured data model (XML)
- Other older models:
  - Network model
  - Hierarchical model

What are models for?

- A model represents a perception of reality



- The modeling process is to fix a perception of reality
- During the modeling process we
  - select aspects
  - abstract to form a simple comprehension of the phenomena in the real world
- A Database is a Model of Reality
- A Database System is software to support the definition & use of the database

9

What are models for?

- useful when we want to
  - Examine
  - Evaluate
  - Manage a subset of the real world
- the cost of using the models is considerably lower than the cost of doing experiments in the real world

Examples:

- airplane simulator
- nuclear plant simulator
- economic model
- data model
- a map

10

The Relational Model

wrote	author	ISDN	publish-city	year
	Brown	56789	New York	1988
	Jeff	Tim	Washington	2001
	Jim	John	Clarcksville	2012

- Data Entities (objects) are represented by ***relations*** (alias ***tables***)
  - each data entity is a single ***tuple*** (alias ***row***) in a relation
  - each property of the entity is an ***attribute*** (alias ***column***) in the tuple
- ***Relationships among data entities are also represented by relations***
  - each relationship is a tuple in a relation
  - each property of the relationship is an attribute (column)
- Attributes take values from a specific set called the ***domain*** of the column
  - the domain defines the ***type***)
- ➔ ***relation schema***: the name of a relation and the names of its attributes (along with their underlying domains)
- ➔ ***database schema:*** the set of all relation schemas

The Relational Model (cont)

- intension and extension of a relation
  - intension ============>>
  - extension ============>>

attributes (or columns)

author	ssn	name	city	phone
	123	Sue	New York	301-444-4444
	124	Tim	Washington	202-666-7666
	125	John	Clarcksville	301-888-9999

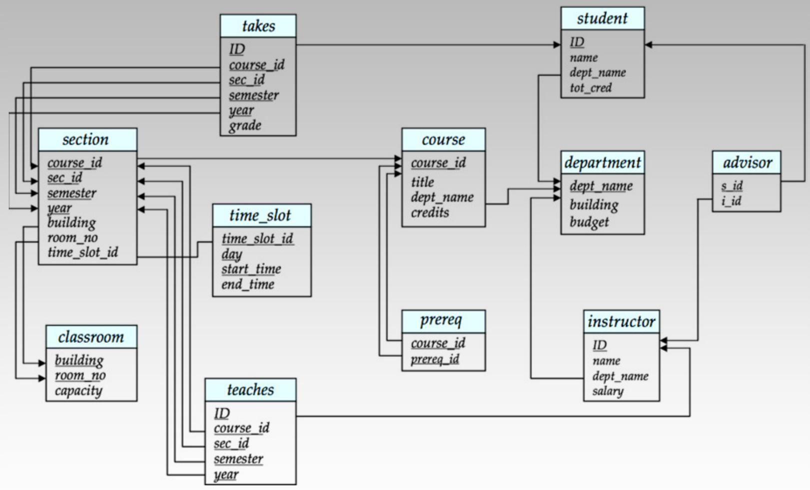
tuples (or rows)

- generalities
  - a relation is a set of tuples
    - order is not important and not guaranteed
    - tuples are unique (set)- however, in real life, RDBMSs allow pseudo sets
  - order of attributes is no longer important as we use the attribute names
    - author.name specifies the 2nd column
    - if t in author, then t.name specifies the value of the second column
  - ➔ the number of attributes in a relation determines the ***arity*** or ***degree*** of it
    - author 4th degree, book 3rd degree (***ternary***), library 2nd degree (***binary***)
  - ➔ the number of tuples in a relation is called the ***cardinality*** of it.
  - ➔ cardinality is a property of a relation- NOT the relation scheme

Keys of Relations

- Keys are defined to distinguish among rows of the table
  - note that keys are properties of the relation scheme- NOT the relation. By looking at the relation we cannot tell what is a key but may be able to tell what isn't.
- Let K $\subseteq$ R. *K* is a superkey of *R* if values for *K* are sufficient to identify a unique tuple of each possible relation *r(R)*
  - Example: {*ID*} and {ID,name} are both superkeys of *instructor.*
- Superkey *K* is a candidate key if *K* is minimal
  Example: {*ID*} is a candidate key for *Instructor.*
- One of the candidate keys is selected to be the primary key.
  - which one?
- Foreign key constraint: Values in one relation must appear in the primary key values of another relation
  - Referencing relation
  - Referenced relation

Schema Diagram for University Database



Relational Query Languages

- Procedural vs.non-procedural, or declarative
- "Pure" languages:
  - Relational algebra
  - Tuple relational calculus
  - Domain relational calculus
- Embedded in host languages or environments

15

Relational Algebra

- set of operators that map one or more relations into another relation
- closed algebraic system
  - the best feature- operations on operations
  - form relational algebraic expressions
- two types of operation: database specific and set theoretic operators
  - database specific
    - [horizontal] ***selection*** (σ) (or ***restriction***)
    - projection (π)
    - join
    - outer join
    - semijoin
    - division
  - set operators
    - union
    - difference
    - intersection
    - cartesian (cross) product

16

Employee-Department Database

Emp	ename	sal	dept
	Gary	30K	toy
	Shirley	35K	candy
	Christos	37K	shoe
	Robin	22K	toy
	Uma	30K	shoe
	Tim	12K	

Dept	dept	floor	mgr
	candy	1	Irene
	toy	2	Jim
	men	2	John
	shoe	1	George

Relational Operators

- [horizontal] selection (σ): picks a subset of the rows

  $\sigma_{dept = \text{'toy'}}$ (Emp)

Result	ename	sal	dept
	Gary	30K	toy
	Robin	22K	toy

- [vertical] projection (π): picks a subset of columns

  π_{floor} (Dept)

Result	floor
	1
	2

Relational Algebra (more operators)

- Cartesian product: r X s

A	B
α	1
β	2

r

C	D	E
α	10	a
β	10	a
β	20	b
γ	10	b

s

A	B	C	D	E
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

Joins

Emp	ename	sal	dept
	Gary	30K	toy
	Shirley	35K	candy
	Christos	37K	shoe
	Robin	22K	toy
	Uma	30K	shoe
	Tim	12K	

Dept	dept	floor	mgr
	candy	1	Irene
	toy	2	Jim
	men	2	John
	shoe	1	George

- **Equijoin (based on equality)**

  Emp ⋈ Dept

  Emp.dept=Dept.dept

Result	ename	sal	dept	dept	floor	mgr
	Gary	30K	toy	toy	2	Jim
	Shirley	35K	candy	candy	1	Irene
	Christos	37K	shoe	shoe	1	George
	Robin	22K	toy	toy	2	Jim
	Uma	30K	shoe	shoe	1	George

- **Natural join (also based on equality)**

  Emp ⋈ Dept

Result	ename	sal	dept	floor	mgr
	Gary	30K	toy	2	Jim
	Shirley	35K	candy	1	Irene
	Christos	37K	shoe	1	George
	Robin	22K	toy	2	Jim
	Uma	30K	shoe	1	George

- **θ join (θ is an arbitrary boolean expression)**

  Emp $\bowtie_{\theta}$ Dept

  When θ is true $r \bowtie_{true} s = r \times s$

Joins (cont)

Emp	ename	sal	dept
	Gary	30K	toy
	Shirley	35K	candy
	Christos	37K	shoe
	Robin	22K	toy
	Uma	30K	shoe
	Tim	12K	

Dept	dept	floor	mgr
	candy	1	Irene
	toy	2	Jim
	men	2	John
	shoe	1	George

- **Semijoin** Emp ⋉ Dept = $\pi_{Emp.*}$ (Emp ⋈ Dept)

Result	ename	sal	dept
	Gary	30K	toy
	Shirley	35K	candy
	Christos	37K	shoe
	Robin	22K	toy
	Uma	30K	shoe

- **Antijoin** Emp ▷ Dept = $\pi_{Emp.*}$ Emp - (Emp ⋉ Dept)

Emp	ename	sal	dept
	Tim	12K	

© Nick Roussopoulos

21

Outerjoins

Emp	ename	sal	dept
	Gary	30K	toy
	Shirley	35K	candy
	Christos	37K	shoe
	Robin	22K	toy
	Uma	30K	shoe
	Tim	12K	

Dept	dept	floor	mgr
	candy	1	Irene
	toy	2	Jim
	men	2	John
	shoe	1	George

- **Left outerjoin** Emp ⟕ Dept

Result	ename	sal	dept	dept	floor	mgr
	Gary	30K	toy	toy	2	Jim
	Shirley	35K	candy	candy	1	Irene
	Christos	37K	shoe	shoe	1	George
	Robin	22K	toy	toy	2	Jim
	Uma	30K	shoe	shoe	1	George
	Tim	12K		NULL	NULL	NULL

- **Right outerjoin** Emp ⟖ Dept

Result	ename	sal	dept	dept	floor	mgr
	Gary	30K	toy	toy	2	Jim
	Shirley	35K	candy	candy	1	Irene
	Christos	37K	shoe	shoe	1	George
	Robin	22K	toy	toy	2	Jim
	Uma	30K	shoe	shoe	1	George
	NULL	NULL	NULL	men	2	John

- **Full outerjoin** Emp ⟗ Dept

Result	ename	sal	dept	dept	floor	mgr
	Gary	30K	toy	toy	2	Jim
	Shirley	35K	candy	candy	1	Irene
	Christos	37K	shoe	shoe	1	George
	Robin	22K	toy	toy	2	Jim
	Uma	30K	shoe	shoe	1	George
	Tim	12K		NULL	NULL	NULL
	NULL	NULL	NULL	men	2	John

22

© Nick Roussopoulos

Do we need all these joins?

- More examples on Relational Algebra can be found in the book and
- https://en.wikipedia.org/wiki/Relational_algebra

Two More Relational Operators

- Rename: $\rho_{new\_name}(E)$ - where E is a relational expression
- Generalized projection $\pi_{F1,F2,..,}(E)$

  - where F1, F2,... Are functions on columns

Comparing Tuples in the same relation

- **Example: Find the smallest salary employee**
  - we need to compare every tuple with every other tuple in Emp
  - bring them together by taking the Cartesian product Emp X Emp to compare the two sal columns
  - how can we distinguish the two sal columns?
    Rename one Emp with ρ_{Emp2} (Emp)
  - Join the two Emp $\bowtie$ Emp2
  - and compare Emp.sal > Emp2.sal
  - this will give all the employees that have at least another employee with smaller salary
  - subtract from Emp the above to find Tim

Emp	ename	sal	dept
	Gary	30K	toy
	Shirley	35K	candy
	Christos	37K	shoe
	Robin	22K	toy
	Uma	30K	shoe
	Tim	12K	

Emp2	ename	sal	dept
	Gary	30K	toy
	Shirley	35K	candy
	Christos	37K	shoe
	Robin	22K	toy
	Uma	30K	shoe
	Tim	12K	

$$\pi_{Emp.ename}(Emp) - \pi_{Emp.ename}(Emp \bowtie_{Emp.sal > Emp2.sal} \rho_{Emp2}(Emp))$$

Relational Algebra - Set Operators

- **union: r U s - both relations must be union-compatible i.e. same degree and the same domains**

r

A	B
α	1
α	2
β	1

s

A	B
α	2
β	3

A	B
α	1
α	2
β	1
β	3

- **difference: r - s - both relations must be union compatible**

r

A	B
α	1
α	2
β	1

s

A	B
α	2
β	3

A	B
α	1
β	1

- **intersection: r ∩ s - both relations must be union compatible**

r

A	B
α	1
α	2
β	1

s

A	B
α	2
β	3

A	B
α	2

Relational Calculi

- based on predicate calculus
- non-procedural (descriptive) as opposed to procedural (prescriptive)

Tuple Calculus

Query: {t | P(t)}

P: is a predicate associated with some relation R

t: tuple variable ranging over the relation R

t[A]: value of attribute A in t

- students in CMSC 424
  - $\{t \mid t \in$ enroll $(t[course\#] = CMSC424)\}$
- students in CMSC-424 conforming with the CMSC-351 prerequisite
  $\{t \mid t \in$ enroll , $s \in$ enroll $(t[course\#]=CMSC424) \wedge (s[course\#]=CMSC351) \wedge (t[ss\#]=s[ss\#])\}$

- Quantifiers, Bound and Free Variables
  - $\exists s$ (or $\forall s$) quantify the variable s following them-
    s is bound by the quantifiers
  - queries have free variables that take independent values: t above is free

Atoms & Formulas

- Atoms
  - R(t) where t is a tuple variable
  - $t[l]\ \theta\ s[j]$ where t,s are tuple variables and $\theta \in \{<, >, =, \neq, \leq, \geq\}$

- Formulas are:
  - an atom is a formula
  - if P and Q are formulas, then so are (P), $\neg$ P, P $\wedge$ Q, P $\vee$ Q
  - if P(t) is a formula and t is free, then $\exists$ t P(t) and $\forall$ t P(t) are also formulas

- Equivalences
  - $\neg(P \wedge Q) \equiv \neg P \vee \neg Q$
  - $\neg(P \vee Q) \equiv \neg P \wedge \neg Q$
  - $\forall t \in r(P(t)) \equiv \neg\exists t \in r(\neg P(t))$
  - $P \Rightarrow Q \equiv \neg(P) \vee Q$

Tuple Calculus (cont)

- Problems with negation

  {t | t ∉ Enroll} - Unsafe

  Restriction: put negation as a second operand of a conjunction. The first operand of the conjunction defines the scope and the second filters this scope.

  $\{t \mid \underbrace{t \in \text{Enroll}}_{\text{positive scope}} \wedge \underbrace{t[\text{course\#}] \neq \text{CMSC-420}}_{\text{negative clause}}\}$ - Safe

Domain Calculus

- dom(X) - set of values of a domain; more than one attribute may receive values from dom(X)
- Atoms
  - <x1,x2,…,xN> ∈ R - set of domain variables receiving values from the attributes of R
  - x θ y where x,y are domain variables and θ ∈{<, >, =, ≠, ≤, ≥ }
  - x θ c where c is a constant
- Formulas are:
  - an atom is a formula
  - if P and Q are formulas, then so are (P), ¬ P, P ∧ Q, P ∨ Q
  - if P(x) is a formula and x is a domain variable, then ∃x P(x) and ∀x P(x) are also formulas
- Queries:

  {<x1,x2,…,xN> | Ψ(x1,x2,…,xN)}

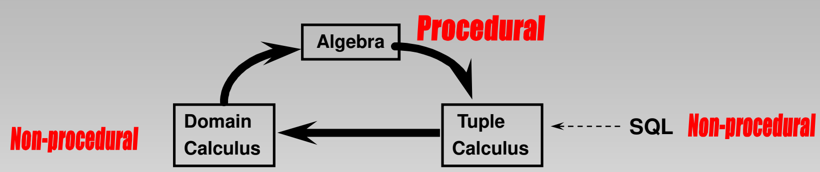
  - Examples:

    {<ss#,course#,semester> | Enroll(ss#,course#,semester)}

    {<x,y,z> | Enroll(x,y,z) ∧ y=CMSC-424}

- Domain calculus expressions are <u>safe</u>

Reductions of Relational Algebra & Calculi



- the reduction proves equivalence in expressing power
- these languages are used as benchmarks for other languages
- a relational language L is relationally complete iff statements in one of the above can be mapped to statements in L and vice versa

Relational Algebra (one last operator)

- division R / S: given R(A,B) and S(B) t is in R/S if for all s in S there exists an r in R such that r.B=s.B and t.A=r.A

Construction:

$$R/S = \pi_A(R) - \pi_A((\pi_A(R) \times S) - R)$$

<==all possible pairs including the desired

<== not desired tuples