
DBMS
data
database
distributed
parallel
shared
system
architecture
processing
network
DBMSs
Distributed
DDBMS
systems
processors
performance
access
centralized
memory
multiple
sites
site
processor
nothing
applications
computers
single
disk
computer
technology
Shared
may
fragments
typically
major
require
physically
within
users
machines
Database
control
scalability
approach
improve
distinction
logical
Parallel
also
make
provide
local
number
user
dis
application
management
transparent
bottleneck
interconnection
Figure
nodes
definition
development
sometimes
computing
one
split
order
scalable
architectures
disks
islands
resources
large
remote
defined
controlled
based
decentralized
directly
Date
com
operational
referred
often
Management
replicated
discuss
many
reliability
eliminates
communication
distribution
fact
across
logically
stored
incompatible
way
global
associated
partitioned
used
least
consists
organizational
administered
mode
capable
databases
collection
structure
provides
must
System
makes
per
like

Database System Concept (CSE 3103)

Lecture 02-Day 01

Nazmus Sakib, Assistant Professor, Dept. of CSE, AUST

Attribute Types

- The set of allowed values for each attribute is called the **domain** of the attribute
- Attribute values are (normally) required to be **atomic**; that is, indivisible
- The special value ***null*** is a member of every domain. Indicated that the value is “unknown”
- The null value causes complications in the definition of many operations

Relation Schema and Instance

- $A_1, A_2, \ldots, A_n$ are *attributes*

- $R = (A_1, A_2, \ldots, A_n)$ is a *relation schema*

  Example:

  instructor = (*ID, name, dept_name, salary*)

- Formally, given sets $D_1, D_2, \ldots. D_n$ a **relation *r*** is a subset of

  $D_1 \times D_2 \times \ldots \times D_n$

  Thus, a relation is a set of n-tuples $(a_1, a_2, \ldots, a_n)$ where each $a_i \in D_i$

  - The current values (**relation instance**) of a relation are specified by a table
  - An element t of r is a *tuple*, represented by a *row* in a table

Relations are Unordered

- Order of tuples is irrelevant (tuples may be stored in an arbitrary order)
- Example: *instructor* relation with unordered tuples

ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

Keys

- 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: Value in one relation must appear in another
  - **Referencing** relation
  - **Referenced** relation
  - Example – *dept_name* in i*nstructor* is a foreign key from *instructor* referencing *department*

Relational Query Languages

- Procedural vs .non-procedural, or declarative
- “Pure” languages:
  - Relational algebra
  - Tuple relational calculus
  - Domain relational calculus
- The above 3 pure languages are equivalent in computing power
- We will concentrate in this chapter on relational algebra
  - Not turning-machine equivalent
  - consists of 6 basic operations