

Lecture 1: Part I

Emerging Database Technology,
Research and Applications

Database Management: A Historical Perspective

- **1965-1980 Hierarchical/Network Models**
 - VAX DBMS
- **1980-1990 Relational Model**
 - IBM DB2, ORACLE, SYBASE, INFORMIX
- **1990s Object Oriented DBMS's**
 - ObjectStore, Versant
- **late 90's**
 - **Object-Relational Databases**
 - Informix Universal Server, Oracle Universal Server
 - **Data Warehousing**
 - Teradata + the usual suspects

Emerging Applications

– *Each domain imposes different requirements on the DBMS in terms of storage, access methods, knowledge discovery and data analysis.*

- **WWW**
 - *less than 10% of the WWW is mapped by current search engines, also how do you represent semi-structured data?*
- **Engineering & Scientific Applications**
 - *complex interrelationships need to be mapped*
- **Telecommunication**
 - *streaming data needs to be handled*
- **Medicine**
 - *large heterogeneous (genomic, MRI, ultrasound) datasets, fast sequence & association matching & access required*
- **Multimedia**
 - *Different forms of data need to be represented and accessed interactively (real-time response)*

Key Challenges

- Complex Objects & Relationships
- Multiple data types
 - Numbers, Text, Images, Audio, Video
- Multiple dimensions of Information
 - Structural, Low-level, High level (Meta-data)
- Real time response for high volume data
- User Expectations on the rise
 - Better Interfaces/Visualizations
 - Going beyond raw data to meaningful information
 - **Knowledge Discovery and Data Mining**
 - Extraction, Cleaning, Mining, Result Interpretation

Standards or a lack thereof

- Document/Text
 - HTML, XML, PMML etc.....
- Engineering
 - Products Data Exchange, STEP ISO
- Objects
 - COM, CORBA, etc.....
- Security
 - ?? mostly in-house & specialized
- GUI/Scripting languages
 - CGI, Perl, Javascript
- Object/Relational Database Management
 - SQL3?

Emerging Technologies tied to Data Management

- **Communicating Databases**
 - ubiquitous mobile applications and data
- **Intelligence in Data Management**
 - Data Mining, Machine Learning
- **Multimedia**
 - Display, Visualization and Animation
- **Component Design Methodologies**
 - for better control of the design process
- **Computer Supported Collaborative Work**
 - working at home and in wide-spread teams
 - e.g. Boeing 777 design
- **Interoperability, Heterogeneity, Adaptable**

Current Trends to Consider

- Internet access to large public datasets
- Prominent interoperable standards
 - JDBC, DCOM, CORBA
- Functionality Trends
 - Metadata, parallel processing
- Adaptive Database Technologies
 - dynamic specs (refresh/push-pull technologies)
 - mobile databases that adapt to user needs

Current Research Thrusts

- New data models for multimedia types
- Component based data engineering
- Parallel database processing
- Knowledge Discovery and Data Mining
- Mediators, Secure Databases
- Content based Retrieval
- Coping with WWW Information Overload

Challenges for Database Professionals

- Learning the new applications
 - jargon
 - complexities, typical scenarios, rules, constraints
- Apply DB techniques to help in application
 - views, transactions
 - application-specific modelling and access functions
- Apply other techniques to DB area
 - AI, IR, ML, SE, UI

Future

- More variety of Applications and Data
- A great need for domain experts who can also understand DB modeling & concepts
- More demands on performance
 - scaling to larger databases
 - staying within decent response times
- More variety of users
 - K-12 children, housewives, naïve users

TOWARD

- Multilingual
- Multiplatform
- Multiprocessor
- Intelligent
- Interoperable
- Adaptive
 - *Database and Knowledge Base Systems*

Lecture 1: Part II

Relational Data Model

Relational Model

- **Relational Data Structures**
- Relational Integrity Rules
- Relational Algebra

Relational Data Structures Basic Definitions

- domain - set of atomic values
- atomic value - nondecomposable
 - called simple domains as opposed to composite domains
- composite domains - Cartesian product of simple domains

Example Date = {Month Day Year}
 Month = {1, 2, ..., 12}
 Year = {1900, 1901, 2000}

e.g. 10 9 99 October 9, 1999

Relations

Relation schema R or $R(A_1, A_2, \dots, A_n)$

- fixed set of attributes A_1, A_2, \dots, A_n
- each attribute A_j corresponds to exactly one of the underlying domains D_j ($j = 1, 2, \dots, n$), not necessarily distinct

$$D_j = \text{domain}(A_j)$$

EMP(EMPNO, NAME, DNO, JOB, MGR, SAL, COMMISSION)
note: EMPNO and MGR take values from the same domain, the set of valid employee numbers

N , the number of attributes in the relation scheme is called the **degree** of the relation.

Degree 1 unary degree 3 ternary
degree 2 binary degree n n -ary

Relation (or relation instance) r of relation schema $R(A_1, A_2, \dots, A_n)$ or $r(R)$

Alternative Definitions (1):

- | | |
|--|--|
| <p>***** 1a *****</p> <ul style="list-style-type: none"> • A set of n-tuples (or tuples),
$r = \{t_1, t_2, \dots, t_m\}$ • each tuple
$t = \langle v_1, v_2, \dots, v_n \rangle$ • v_j an element of
$D_j = \text{domain}(A_j)$ | <p>***** 1b *****</p> <ul style="list-style-type: none"> • $r(R) \subseteq D_1 \times D_2 \times \dots \times D_n$ |
|--|--|

Relation (or relation instance) r of relation schema $R(A_1, A_2, \dots, A_n)$ or $r(R)$

Alternative definition 2a:

- Set of n-tuples (or tuples, where each tuple consists of attribute-value pairs (A_j, v_j) ($j = 1, 2, \dots, n$), one such pair for each attribute A_j in the relation schema.
- For any given attribute-value pair (A_j, v_j) , v_j is a value from the unique domain D_j that is associated with that attribute.

Relation (or relation instance) r of relation schema $R(A_1, A_2, \dots, A_n)$ or $r(R)$

Alternative definition 2b:

- Finite set of mappings $r = \{t_1, t_2, \dots, t_m\}$
- each tuple t is a mapping from
 $R = R(A_1, A_2, \dots, A_n)$
to
 $D = \text{domain}(A_1) \cup \text{domain}(A_2) \cup \dots \cup \text{domain}(A_n)$
- I.e. $t(A_j) \in \text{domain}(A_j)$

Attribute Names vs.. Domain Names

Attribute names are usually chosen to be the same as domain names. However two attributes may use the same domain.

EMP(EMPNO, NAME, DNO, JOB, MGR, SAL, COMMISSION)

1234	...	4567	...
2222	...	4567	...
4567	...	9999	...
9999

•Here both attributes EMPNO and MGR take values from the *same domain*, namely legal employee numbers.

•Thus attributes are given unique **role names**, in this case EMPNO and MGR.

Keys

- A relation is a **set** of tuples.
- All elements of a set are distinct.
- Hence all tuples must be distinct.
- There may also be a subset of the attributes with the property that values **must** be distinct. Such a set is called a **superkey**.
 - SK a set of attributes Guaranteed by the
 - t_1 and t_2 tuples "real world".
 - t_1 [SK] \neq t_2 [SK]

Example

EMP(EMPNO, NAME, DNO, JOB, MGR, SAL, COMMISSION)

Some superkeys:

EMPNO, NAME, DNO, JOB, MGR, SAL, COMMISSION
EMPNO, NAME, DNO, JOB
EMPNO, NAME
EMPNO, NAME, SAL, COMMISSION
EMPNO
NAME ???

Candidate Key - a *minimal* superkey

- A set of attributes, CK is a superkey,
- but no proper subset is a superkey.

Example
EMPNO
NAME ???

Primary Key - one *arbitrarily chosen* candidate key.

Example
EMPNO

Example

Relation **Candidate Key**

EMP EMPNO
 NAME ??

DEPT DNO
 DNAME
 LOC ??

Primary Keys: EMPNO, DNO

Note: The *only* way to identify a *unique* tuple is by its *primary key*.

Properties of Relations

- Tuples are *unordered*.
- There are *no* tuple *duplicates*.
- Attributes are *unordered* (Definition 2).
- All attribute values are *atomic*.

Tuples are *unordered*

- Mathematical definition of a set.
 - *Concrete* realization, i. e. printed table does have an order.
 - *Abstract* construct does not.
 - Much easier to understand effects of operations if no ordering is specified.

There are *no* tuple *duplicates*.

- Mathematical definition of a set.
 - Primary key must always exist.
 - Might have to be all attributes.

Attributes are *unordered* (Definition 2).

- Since relation schema is a *set*.
- *Concrete* representation is misleading.

All attribute values are *atomic*.

- Relations may not contain *repeating groups*.
- Such a relation is said to be **normalized** or in **1st normal form**.

Before:	<u>S#</u>	<u>(PQ)*</u>	After:	<u>S#</u>	<u>P#</u>	<u>QTY</u>
		<u>P#</u>	<u>QTY</u>			
	S1	P1 300		S1	P1 300	
		P2 200		S1	P2 200	
		P3 400		S1	P3 400	
	S2	P1 300		S2	P1 300	
		P2 400		S2	P2 400	
	S3	P2 200		S3	P2 200	
	3 "records"			6 "records"		

Relational Database *Schema S*

- Set of Relation Schemas $S = \{R_1, R_2, \dots, R_p\}$
- Set of **integrity constraints, IC**
 - Integrity constraints will be discussed next.

Relational Database (*Instance*) DB of Schema S

- Set of relation instances, DB
- $DB = \{r_1, r_2, \dots, r_p\}$
- Each r_k is an instance of relation R_k
- The r_k 's satisfy the integrity constraints, IC

Database Example - *Schema*

EMP(EMPNO, NAME, DNO, JOB, MGR, SAL, COMMISSION)
EMPNO from domain EmployeeNumbers
...
Integrity constraints to be added
DEPT(DNO, DNAME, LOC)
DNO from domain DepartmentNumber
Integrity constraints to be added

Database Example - *Database (instance)*

EMP(EMPNO, NAME, DNO, JOB, MGR, SAL, COMMISSION)
1234 Smith D2 J1 4567 30K 10
2222 Jones D2 J2 4567 25K
4567 Blue D1 J4 9999 40K
9999 Green D3 J5 100K

DEPT(DNO, DNAME, LOC)
D1 xxx yyy
D2 aaa yyy
D3 bbb zzz
