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

<table>
<thead>
<tr>
<th>Example</th>
<th>Date = (Month Day Year)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Month = {1, 2, …, 12}</td>
</tr>
<tr>
<td></td>
<td>Year = {1900, 1901, 2000}</td>
</tr>
<tr>
<td>e.g. 10 9 99</td>
<td>October 9, 1999</td>
</tr>
</tbody>
</table>

Relations

**Relation schema** \( R \) or \( R(A_1, A_2, \ldots, A_n) \)

- fixed set of attributes \( A_1, A_2, \ldots, A_n \)
- each attribute \( A_j \) corresponds to exactly one of the underlying domains \( D_j \) (\( j = 1, 2, \ldots, n \)), 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 2: binary
Degree 3: ternary
Degree n: n-ary
**Relation** (or **relation instance**) \( r \) of relation schema \( R(A_1, A_2, \ldots, A_n) \) or \( r(R) \)

**Alternative Definitions (1):**

1a

- A set of \( n \)-tuples (or tuples),
  \( r = \{t_1, t_2, \ldots, t_m\} \)
- each tuple \( t = \langle v_1, v_2, \ldots, v_n \rangle \)
- \( v_j \) an element of \( D_j = \text{domain}(A_j) \)

1b

- \( r(R) \subseteq D_1 \times D_2 \times \cdots \times D_n \)

**Relation** (or **relation instance**) \( r \) of relation schema \( R(A_1, A_2, \ldots, A_n) \) or \( r(R) \)

**Alternative definition 2a:**

- Set of \( n \)-tuples (or tuples), where each tuple consists of attribute-value pairs \( (A_j, v_j) \) (\( j = 1, 2, \ldots, n \) ), 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 \( R(A_1, A_2, \ldots, A_n) \) or \( r(R) \)

**Alternative definition 2b:**

- Finite set of mappings \( r = \{t_1, t_2, \ldots, t_m\} \)
- each tuple \( t \) is a mapping from \( R = R(A_1, A_2, \ldots, A_n) \) to \( D = \text{domain}(A_1) \cup \text{domain}(A_2) \cup \cdots \cup \text{domain}(A_n) \)
- 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
• \( t_1 \) and \( t_2 \) tuples
• \( t_1 \ [SK] \neq t_2 \ [SK] \)

Guaranteed by the “real world”.

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

---

<table>
<thead>
<tr>
<th>Relation</th>
<th>Candidate Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMP</td>
<td>EMPNO</td>
</tr>
<tr>
<td></td>
<td>NAME ??</td>
</tr>
<tr>
<td>DEPT</td>
<td>DNO</td>
</tr>
<tr>
<td></td>
<td>DNAME</td>
</tr>
<tr>
<td></td>
<td>LOC ??</td>
</tr>
</tbody>
</table>

**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*.

<table>
<thead>
<tr>
<th>Before:</th>
<th>After:</th>
</tr>
</thead>
<tbody>
<tr>
<td>S# (P#)</td>
<td>S# P# QTY</td>
</tr>
<tr>
<td>P# QTY</td>
<td>P# QTY</td>
</tr>
<tr>
<td>S1 P1 300</td>
<td>S1 P1 300</td>
</tr>
<tr>
<td>P2 200</td>
<td>S1 P2 200</td>
</tr>
<tr>
<td>P3 400</td>
<td>S1 P3 400</td>
</tr>
<tr>
<td>S2 P1 300</td>
<td>S2 P1 300</td>
</tr>
<tr>
<td>P2 400</td>
<td>S2 P2 400</td>
</tr>
<tr>
<td>S3 P2 200</td>
<td>S3 P2 200</td>
</tr>
</tbody>
</table>

3 "records" → 6 "records".

---

Relational Database *Schema S*

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

Relational Database (*Instance*)

**DB of Schema S**

- *Set of relation instances*, $DB$
- $DB = \{r_1, r_2, \ldots, r_p\}$
- 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)

<table>
<thead>
<tr>
<th>EMPNO</th>
<th>NAME</th>
<th>DNO</th>
<th>JOB</th>
<th>MGR</th>
<th>SAL</th>
<th>COMMISSION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1234</td>
<td>Smith</td>
<td>D2</td>
<td>J1</td>
<td></td>
<td>4567</td>
<td>30K</td>
</tr>
<tr>
<td>2222</td>
<td>Jones</td>
<td>D2</td>
<td>J2</td>
<td></td>
<td>4567</td>
<td>25K</td>
</tr>
<tr>
<td>4567</td>
<td>Blue</td>
<td>D1</td>
<td>J4</td>
<td></td>
<td>9999</td>
<td>40K</td>
</tr>
<tr>
<td>9999</td>
<td>Green</td>
<td>D3</td>
<td>J5</td>
<td></td>
<td>9999</td>
<td>100K</td>
</tr>
</tbody>
</table>

DEPT(DNO, DNAME, LOC)

<table>
<thead>
<tr>
<th>DNO</th>
<th>DNAME</th>
<th>LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>xxx</td>
<td>yyy</td>
</tr>
<tr>
<td>D2</td>
<td>aaa</td>
<td>yyy</td>
</tr>
<tr>
<td>D3</td>
<td>bbb</td>
<td>zzz</td>
</tr>
</tbody>
</table>