

CS180

Database Management Systems

- Instructor: Wang-Chiew Tan
 - Email: wctan@cs.ucsc.edu
 - Office Hours: Tue 4.30-5.30pm, Thu 4.30-5.30pm (at BE359A)
- TA: Deepa Tuteja
 - Email: deepa@soe.ucsc.edu
 - Office Hours: To be announced
- Class Web Page:
 - <http://www.cse.ucsc.edu/~cmps180/winter03/>

Winter 2003

1

Course Information

- Course Textbook
 - *Database Management Systems*. Third Edition. McGraw-Hill Higher Education.
By R. Ramakrishnan and J. Gehrke.
- Other textbooks
 - *Database Systems Concept*. Fourth Edition. McGraw-Hill Higher Education.
By A. Silberschatz and H.F. Korth and S. Sudarshan.
 - *Database Systems: The Complete Book*. Prentice Hall.
H. Garcia-Molina and J. Ullman and J. Widom
 - *Foundations of Databases*. Addison Wesley
S. Abiteboul and R. Hull and V. Vianu
 - *Data on the Web*. Morgan Kaufmann
S. Abiteboul and P. Buneman and D. Suciu

Winter 2003

2

Course Information

- Course Syllabus
 - A number of foundational topics (see web page) will be covered
 - Roughly follows the chapters 1-5, 8, 12–15, 19, 24, and 27 of textbook
 - Subject to change
- Lecture notes will be provided
- Homeworks & Project
 - There will be roughly 6 homework assignments and a project
 - No solutions to homeworks or project will be provided

Winter 2003

3

Course Information

- Newsgroup: `ucsc.class.cmps180`
 - Discussions in this newsgroup about lecture notes, homework, and lab assignments are greatly encouraged
 - You get the fastest (and usually the best) answers from your peers
- Grading
 - Homeworks 15%
 - Project 20%
 - Midterm 30%
 - Final 35%
- Cheating
 - Don't even think about it!
 - No form of academic dishonesty will be tolerated in this class.
 - See http://www.ucsc.edu/academics/academic_integrity/

Winter 2003

4

What is a Database Management System?

- A database is a collection of data
- The collection of data models the real-world. Examples:
 - It may model entities: student John, course cs180, or
 - relationships: John is taking cs180
- A database management system (DBMS) is a software designed to assist in maintaining and utilizing large collections of data.
- Why is it different from storing them in files and writing applications to access these files?

Winter 2003

5

Outline

- Overview of DB systems
- History
- File system vs DBMS
- Structure of a DBMS

Winter 2003

6

Files VS. DBMS

Consider the following scenario:

You have 500GB of company data and these data are to be accessed by different employees concurrently. How can you efficiently manage the data?

Suppose you store the data in operating system files.

1. Do you have enough memory to hold all the data?
 - Coding is required whether the answer is yes or no
2. How can you allow users to pose queries?
 - Special code for different queries.
 - Complex programs for efficient access.
 - Need to design the query language first!

Winter 2003

7

Files VS. DBMS

- How can you protect data from inconsistent changes made by users accessing the data concurrently?
 - Need some complex coding
- How can you ensure the data is restored to a correct state if the system crashes?
- How can you enforce different security policies?

Winter 2003

8

Advantages of a DBMS

- Data Independence
 - Ideally, application programs should not be aware of how data is stored and represented
- Efficient Data Access
 - Efficiency of data access with a DBMS is often hard to beat!
- Concurrent Access and Crash Recovery
 - Perhaps one of the most important uses of a DBMS
- Data Integrity and Security
 - State and enforce integrity constraints and access controls easily
- Data Administration
 - Fine tuning
- Reduced Application Development Time

Winter 2003

9

History

- Early 1960s (Network Data Model)
 - Integrated Data Store by Charles Bachman at GE
 - Influential through the 1960s
 - Formed the basis of network data model (directed graph)
- Late 1960s (Hierarchical Data Model)
 - Tree model
 - Information Management System (IMS) by IBM
 - Used even today
- 1970 (Relational Data Model)
 - Edgar Codd (at IBM Almaden Research, San Jose) proposed a new data model
 - Idea is followed by the development of several DBMS (e.g., System R) and development of theoretical results.

Winter 2003

10

History

- Benefits of RDBMS are widely recognized and RDBMS became a commercial standard
- 1980s
 - SQL was standardized as an RDBMS language
 - Current standard: SQL:1999
 - Widely used for its transaction capability
 - James Gray: transaction management
- Interestingly, Bachman, Codd, and Gray each received a Turing award (1973, 1981, and 1999 resp.) for their work in the database area

Winter 2003

11

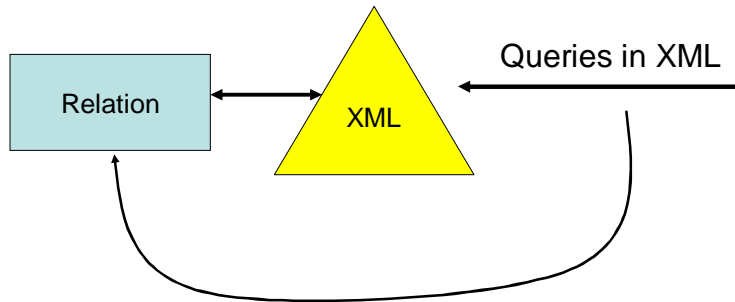
History

- 1980s
 - Object-oriented databases (Versant, ObjectStore etc.)
 - Idea never really caught on
 - Deductive databases (mostly research prototypes. Influenced commercial DB systems)
- Late 1980s, the 1990s
 - More complex queries
 - Data warehouses, data mining
 - Store and manage new data types. E.g., images, video
 - Early 1990s: Semistructured data research
- Today
 - XML is the widely used format for data exchange (tree model !)
 - New requirements on RDBMS
 - Publish XML data with a relational backend, store XML data into relational tables, and query the system regardless of how the underlying data is stored!

Winter 2003

12

Relational databases and XML



Winter 2003

13

The Relational Model

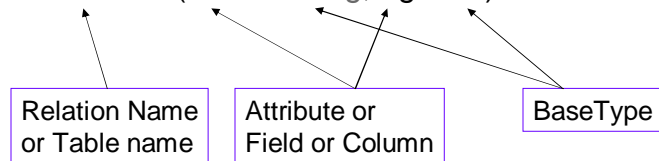
- Central concept is a relation
 - A relation instance is a set of records of the same type
 - A record is a “ n -ary tuple” where each component is a field name and value pair
<name: “John”, age: 19>
 - Example of a Students relation where each of its record consists of Name and Age fields:
Students =
{ <name: “John”, age: 19>,
 <name: “Ann”, age: 20> }

Winter 2003

14

Schema

- The **schema** of a relation (relation schema) consists of
 - The relational name
 - A set of attribute and type pairs
- The schema of the above relation is
`Students(name: string, age: int)`



- Typically written “sloppily” as `Students(name, age)`

Winter 2003

15

Relation, Instance, Extent

- A relation or instance or extent of the relation schema
- They are typically shown as a table
- An instance of the relation schema

`Students(name:string, age:int, GPA:real)`
is shown below

Students

Name	Age	GPA
John	19	3.5
Ann	20	3.2
Joe	23	3.3

a row or tuple or record

Winter 2003

16

Integrity Constraints

- An integrity constraint specifies a condition that an instance of a relation schema must satisfy
- Example:
 - Suppose name is the key for Student(name, age, GPA)
 - Every student in the relation must have a unique name value
 - Alternatively, it is impossible to find two students with the same name in the relation

Winter 2003

17

Main DBMS today

- IBM's DB2
- Oracle
- Microsoft SQL Server
- Sybase
- etc.

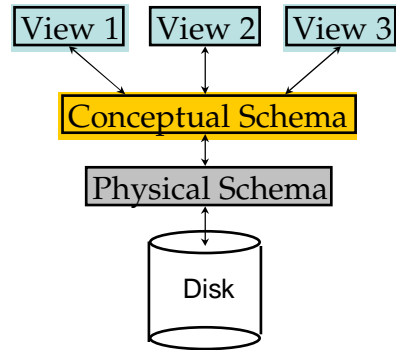
- FACT 1: The evolution of DBMS systems go hand in hand with research (IBM and Microsoft both have serious research laboratories).
- So what?
- FACT 2: RDBMS is the main DBMS used by many companies nowadays and is a billion \$\$\$ industry

Winter 2003

18

Levels of Abstraction in a DBMS

- Three levels of abstraction
- View or External Schema
 - Employee(name, addr, salary)
 - Defined through a Data Definition Language (DDL), e.g., SQL
- Conceptual or Logical Schema
 - Describes the underlying data in terms of the data model, e.g., relations
 - Conceptual database design
- Physical schema
 - Describes how data described at the logical level are stored on disk or tapes, e.g., what file organization and what indexes



Winter 2003

19

Example - a university database

- Conceptual Schema
 - Students(sid: string, name: string, login: string, age: integer, gpa: real)
 - Courses(cid: string, cname: string, credits: integer)
 - Enrolled(sid: string, cid: string, grade: string)
 - Faculties(fid: string, courseid: string, fname: string, sal: real)
- View or External Schema
 - Faculty_info(fid: string, fname: string, courseid: string)
 - Defined through the SQL query

```
SELECT  fid, fname, courseid
FROM    Faculties
```
 - The rest are the same

Winter 2003

20

Example

- Physical Schema
 - Relations stored as unordered files.
 - Index on first column of Students and Courses
 - Need an understanding on how data is typically accessed

Winter 2003

21

Data Independence

- One benefit of using a DBMS is that application programs are insulated from changes in the way data is structured or stored
- A user should be able to ask a query without worrying about the actual logical structure of data
 - Logical data independence: protects the user from changes in the logical structure of data
 - Faculty_public(fid: string, fname: string, office: integer)
 - Faculty_private(fid: string, sal: real)

faculty_info view can be redefined in terms of Faculty_public

Winter 2003

22

Data Independence

- A user should be able to ask a query without knowing how data is actually stored
 - Physical data independence: protects the user from changes in the physical structure of data
 - The user query need not be changed when an additional index on fid is built on the Faculties relation. However, the user might notice that the query now runs faster

Winter 2003

23

Relational Algebra & Relational Calculus

SQL:

```
SELECT  fid, fname, courseid
FROM    Faculties
```

Relational Algebra (RA):

$\Pi_{fid, fname, courseid}$ (Faculties)

Tuple Relational Calculus (TRC):

$\{ y \mid x \in \text{Faculties},$
 $y.fid = x.fid,$
 $y.fname = x.fname,$
 $y.courseid = x.courseid \}$

Domain relational calculus (DRC) :

$\{ \langle f, n, c \rangle \mid \langle f, n, c, s \rangle \in \text{Faculties} \}$

Winter 2003

- They have equivalent expressive power (core of SQL, RA, TRC, and DRC).
- SQL is an end-user language.
 - More english-like, good for end users to write a query without having to deal with symbols.
 - SQL is usually translated into a RA expression by a DBMS
- RA is a procedural language.
 - Good for charting out query execution plans
- TRC or DRC is a declarative language.
 - Good for thinking about the query without worrying about how it will be executed
 - Has influenced SQL and QBE

24

A typical scenario

- User queries are received by the DBMS through web forms, application front ends, or an SQL interface
- SQL query is parsed and sent to the query optimizer
- Query optimizer generates one or more query evaluation plans and picks the best one to execute
- A query evaluation plan shows the procedure that the DBMS should take in order to arrive at the answer to the query (typically using an annotated tree of relational operators)
- Through the files and access methods layer, the evaluator accesses the relevant data in order to carry out the execution plan

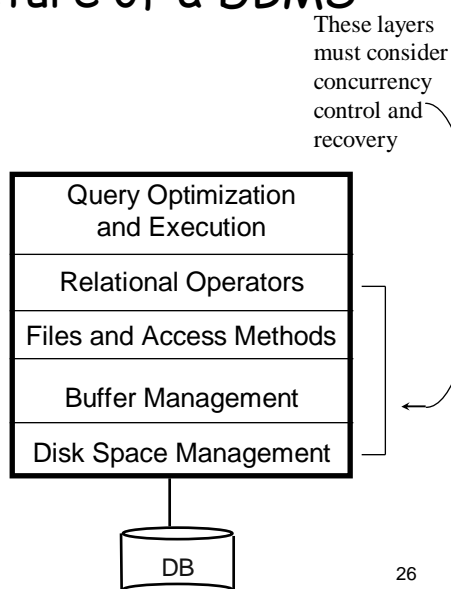
Winter 2003

25

Simplified Structure of a DBMS

A typical scenario (cont'd):

- The read requests from the files and access methods layer are handled by the buffer manager. They bring in the relevant pages from disk to main memory
- A lower level disk space manager deals with the actual physical management of space on disk



Winter 2003

26

Concurrency Control and Recovery

- The picture is not complete without a transaction manager.
- A transaction is an atomic sequence of database actions (read/write) on data items.
- Many transactions (from possibly different users) can be running at the same time but to each user, only her program is running
- The transaction manager decides the schedule of transaction processing
 - someone deposits some money \$X into account A and another person debits some money \$Y from the same account at the same time
- Key concept for recoverability is that of a log : keeping track of all actions carried out by the db so that a consistent state can be recovered by analysing the log in case a system failure occurs.

Winter 2003

27

Connections to other areas of CS

- Concepts used in a DBMS span many areas of CS
 - Programming languages and software engineering
 - SQL, ER modeling
 - Algorithms and data structures
 - Join algorithms, B+ trees
 - Logic, discrete math, and theory of computation
 - Expressive power of SQL?
 - “Systems” issues: concurrency, operating systems, file organization and networks.
 - How are relations actually stored on disk for the best IO efficiency?

Winter 2003

28

What you will learn in CS180

- Query languages
 - SQL, Relational Algebra, Tuple relational calculus, Domain relational calculus, QBE
- Relational database design
 - ER, Relational model, Functional dependencies, Normalization, schema refinement
- Overview of Storage and Indexing techniques
- Overview of Query Evaluation techniques
- Deductive databases
- XML, DTDs, XPath, XML Query language

Winter 2003

29

Recommended Reading

- Philip Greenspun's introduction to database management systems
 - <http://philip.greenspun.com/sql/introduction.html>
- Chapter 1 of textbook

Winter 2003

30