

CS/INFO 330 Data-Driven Web Applications

Mirek Riedewald
mirek@cs.cornell.edu

Course Goals

- Understand functionality of modern database systems
- Understand where database systems fit into an enterprise data management infrastructure
- Design and build a data-driven Web application
 - E.g., build your own eCommerce site
- Learn several important tools/technologies
 - SQL, Java EE, Web services, XML, XQuery
- CS330 versus CS432/433, CS530

CS/INFO 330

2

Instructor

- Mirek Riedewald
- <http://www.cs.cornell.edu/~mirek>
- Office hours: Fridays, 1:20-2:20, Upson Hall 4105C
- Always welcome to ask questions via email (mirek@cs.cornell.edu)
- Ask questions during and after the lecture

CS/INFO 330

3

Teaching Assistants

- Biswanath Panda
 - bpanda@cs.cornell.edu
- Rishit Ratnakar Shetty
 - rrs64@cornell.edu
- Office hours TBD

CS/INFO 330

4

Course Mechanics

- Homepage will have all relevant material
- Slides will be online before each lecture
- Re-grading policy
 - Request re-grade through CMS within 1 week
 - Grade might go up or down
- Academic integrity
- Exam schedule available
 - Notify by 8/31 about conflicts with religious holidays

CS/INFO 330

5

Course Outline

- Lectures
 - Material from the book and the Web
 - Hands-on lectures about the tools that we are using
- Homework
 - Six homework assignments
 - One large project that you will create throughout the semester (several milestones)
- Detailed course outline online

CS/INFO 330

6

Course Topics

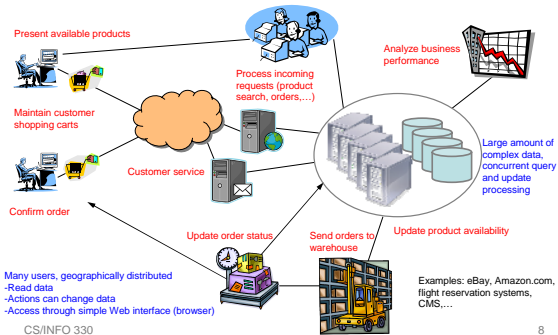
- Database design
- Relational model
- SQL
- Normalization
- XML, XPath, XQuery

- Three-tier architecture concepts, Web services

CS/INFO 330

7

Data-Driven Web Application



CS/INFO 330

8

Requirements

- Store data persistently and securely
- Support concurrent access
 - Data integrity, consistency
- Performance and scalability
- Availability
- Maintainability
- User-friendly interface

CS/INFO 330

9

The Three-Tier Architecture

Presentation tier

Client Program (Web Browser)

Middle tier

Application Server

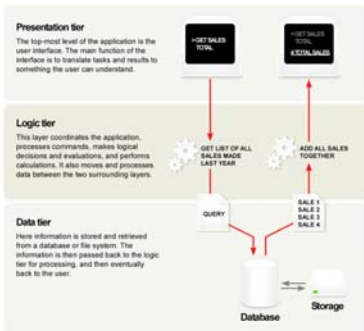
Data management tier

Database System

CS/INFO 330

10

From Wikipedia



CS/INFO 330

11

The Three Layers

- Presentation tier
 - Primary interface to the user
 - Adapt to different display devices (PC, PDA, cell phone, voice access)
- Middle tier
 - Business logic (implements complex actions, maintains state between different steps of a workflow)
 - Accesses different data management systems
- Data management tier
 - One or more standard database management systems

CS/INFO 330

12

Example 1: Airline Reservations

- Database System
 - Airline info, available seats, customer info...
- Application Server
 - Logic to make and cancel reservations, add new airlines, etc.
- Client Program
 - Log in different users, display forms and human-readable output

CS/INFO 330

13

Example 2: Course Enrollment

- Database System
 - Student info, course info, instructor info, course availability, pre-requisites...
- Application Server
 - Logic to add a course, drop a course, create a new course, etc.
- Client Program
 - Log in different users (students, staff, faculty), display forms and human-readable output

CS/INFO 330

14

Three-Tier Architecture: Advantages

- Heterogeneous systems
 - Tiers can be independently maintained, modified, and replaced
- Thin clients
 - Only presentation layer at clients (Web browsers)
- Integrated data access
 - Several database systems can be handled transparently at the middle tier
 - Central management of connections
- Scalability
 - Replication at middle tier permits scalability of business logic
- Software development
 - Code for business logic is centralized
 - Interaction between tiers through well-defined APIs: Can reuse standard components at each tier

CS/INFO 330

15

Technologies

Client Program
(Web Browser)

*HTML, JavaScript,
XSLT (Extensible Stylesheet
Language Transformations)*

Application Server

*XML, Java EE (JavaBeans, JSP),
ASP.NET, C#, CGI, PHP,
Cookies, XPath, Web services*

Database System (DBMS)

*SQL,
Transactions,
Stored Procedures*

CS/INFO 330

16

Why Store Data in a DBMS?

- Benefits
 - Transactions (concurrent data access, recovery from system crashes)
 - High-level abstractions for data access, manipulation, and administration
 - Data integrity and security
 - Performance and scalability

CS/INFO 330

17

What Is a Transaction?

The execution of a program that performs a function by accessing a database.

- Examples of business transactions:
 - Reserve an airline seat. Buy an airline ticket
 - Withdraw money from an ATM, wire transfer
 - Order an item from an Internet retailer
 - Download a video clip and pay for it
 - Place a bid at an on-line auction

CS/INFO 330

18

Challenges

- Other users might make conflicting changes
 - Reserve tickets for same flight
 - Auto-payment while using ATM
 - Buy same product and only few were in stock
 - Bid on same item

Writing distributed applications is hard...

CS/INFO 330

19

Transaction Properties

- Solution: Database transactions
 - Provide powerful model for dealing with concurrent actions
- Atomic sequence of actions
- Must leave the system in a consistent state (if system was consistent when the transaction started)
- The ACID Properties:
 - Atomicity
 - Consistency
 - Isolation
 - Durability

CS/INFO 330

20

Example: Online Store

Your purchase transaction:

- **Atomicity**: Either the complete purchase happens, or nothing
- **Consistency**: The inventory and internal accounts are updated correctly
- **Isolation**: It does not matter whether other customers are also currently making a purchase
- **Durability**: Once you have received the order confirmation number, your order information is permanent, even if the site crashes

CS/INFO 330

21

Transactions (Contd.)

A transaction will *commit* after completing all its actions, or it could *abort* (or be aborted by the DBMS) after executing some actions.

CS/INFO 330

22

Example Transaction: ATM

You withdraw money from the ATM machine

- Atomicity
- Consistency
- Isolation
- Durability

Commit versus Abort?

What are reasons for commit or abort?

CS/INFO 330

23

Transactions: Examples

Give examples of transactions in the following applications. Which of the ACID properties are needed?

- EBay (www.ebay.com)
- Barnes and Noble (www.bn.com)
- Your cell phone

CS/INFO 330

24

Transaction Processing Challenges

- Reliability - system should rarely fail
- Availability - system must be up all the time
- Response time - within a few seconds
- Throughput - thousands of transactions/second
- Scalability - start small, ramp up to Internet-scale
- Security – for confidentiality and high finance
- Configurability - for above requirements + low cost
- Atomicity - no partial results
- Durability - a transaction is a legal contract
- Distribution - of users and data

CS/INFO 330

25

Reliability and Availability

- Reliability - system should rarely fail
- Availability - system must be up all the time: How many "nines"?
 - 1 hour of downtime can cost eBay/Amazon/Google \$1M

Downtime

- 1 hour/day 95.8%
- 1 hour/week 99.41%
- 1 hour/month 99.86%
- 1 hour/year 99.9886%
- 1 minute/day 99.9988%
- 1 hour/20years 99.99942%
- 1 second/week 99.99983%

Availability



Traditional phone service

CS/INFO 330

26

Performance

- Response time—within seconds
- Throughput—thousands of transactions/second
- Scalability—start small, ramp up to Internet-scale
- Caution: There is an inherent **tradeoff** between consistency, availability, and tolerance to network partitions (Eric Brewer, UC Berkeley)
 - Maintaining consistent state across 100s of machines requires expensive agreement (communication)
 - Failures reduce availability, unless consistency is weakened
 - 1000 machines => there is always a failure somewhere
 - Have to sacrifice some ACID properties (esp. C, I) for some system components to achieve extreme scalability; or find tailored solution based on specific workload

CS/INFO 330

27

What Makes TP Important?

- At the core of eCommerce
- Used by most medium-to-large businesses for their production systems
 - Business cannot operate without it
- Huge slice of computer system market
 - Over \$50B/year
 - Probably the single largest application of computers

CS/INFO 330

28

TP System Infrastructure

- User's viewpoint
 - Enter request from browser or other display device
 - System performs application-specific work, including database accesses
 - Receive reply (usually, but not always)
- TP system ensures that each transaction
 - is an independent unit of work,
 - executes exactly once, and
 - produces permanent results
- TP system makes it easy to program transactions
- TP system has tools to make it easy to manage

CS/INFO 330

29

System Characteristics

- Typically < 100 transaction types per application
- Transaction size has high variance; typically
 - 0 - 30 disk accesses
 - 10K - 1M instructions executed
 - 2 - 20 messages
- Large-scale example: Airline reservations
 - 150,000 active display devices
 - Plus indirect access via Internet travel agents
 - Thousands of disk drives
 - 3000 transactions per second, peak

CS/INFO 330

30

Concurrency Control for Isolation

(Start: A=\$0; B=\$100)

Consider two transactions:

- T1: START, A=A+100, B=B-100, COMMIT
- T2: START, A=1.06*A, B=1.06*B, COMMIT
- The first transaction is transferring \$100 from B's account to A's account. The second transaction is crediting both accounts with a 6% interest payment.
- Database systems try to do as many operations **concurrently** as possible, to increase performance.

CS/INFO 330

31

Example (Contd.)

(Start: A=\$0; B=\$100)

- Consider a possible interleaving (schedule):

T1: A=A+\$100, B=B-\$100 COMMIT

T2: A=1.06*A, B=1.06*B COMMIT

End result: A=\$106; B=\$0

- Another possible interleaving:

T1: A=A+100, B=B-100 COMMIT

T2: A=1.06*A, B=1.06*B COMMIT

End result: A=\$106; B=\$6

The second interleaving is incorrect. Concurrency control of a database system ensures the second schedule cannot happen.

CS/INFO 330

32

Ensuring Atomicity

- DBMS ensures atomicity (all-or-nothing property) even if the system **crashes in the middle of a transaction**
- Idea: Keep a log (history) of all actions carried out by the DBMS while executing
 - Before modifying database content, write corresponding log entry to a safe location
 - After a crash, undo effects of partially executed transactions using the log

CS/INFO 330

33

Recovery

- DBMS records all elementary events in a log on stable storage
- Log contains everything that changes data
 - Inserts, updates, deletes
- Reasons for logging
 - Need to UNDO transactions
 - Recover from a system crash

CS/INFO 330

34

Recovery: Example

(Simplified process)

- Insert customer data into the database
- Check order availability
- Insert order data into the database
- Write recovery data (the log) to stable storage
- Return order confirmation number to the customer

CS/INFO 330

35

Why Store Data in a DBMS?

- Benefits
 - Transactions (concurrent data access, recovery from system crashes)
 - High-level abstractions for data access, manipulation, and administration
 - Data integrity and security
 - Performance and scalability

CS/INFO 330

36

Data Model

= Collection of concepts for describing data

- Examples:
 - ER model (used for conceptual modeling)
 - Relational model, object-oriented model, object-relational model (implemented in current DBMS)

CS/INFO 330

37

Relational Data Model

A relational database is a set of relations. Turing Award (“Nobel Prize” in CS) for Codd in 1980 for his work on the relational model

- Example relation:

Customers(cid: integer, name: string, byear: integer, state: string)

cid	name	byear	state
1	Jones	1960	NY
2	Smith	1974	CA
3	Smith	1950	NY

CS/INFO 330

38

Relational Model Terminology

- Relation instance and schema
- Field (column)
- Record or tuple (row)
- Cardinality

cid	name	byear	state
1	Jones	1960	NY
2	Smith	1974	CA
3	Smith	1950	NY

CS/INFO 330

39

Customer Relation

In an enterprise, you are more likely to encounter a schema similar to the following:

Customers(cid, identifier, nameType, salutation, firstName, middleNames, lastName, culturalGreetingStyle, gender, customerType, degrees, ethnicity, companyName, departmentName, jobTitle, primaryPhone, primaryFax, email, website, building, floor, mailstop, addressType, streetNumber, streetName, streetDirection, POBox, city, state, zipCode, region, country, assembledAddressBlock, currency, maritalStatus, bYear, profession)

CS/INFO 330

40

Product Relation

- Relation schema:
Products(pid: integer, pname: string, price: float, category: string)
- Relation instance:

pid	pname	price	category
1	Intel PIII-700	300.00	hardware
2	MS Office Pro	500.00	software
3	IBM DB2	5000.00	software
4	Thinkpad 600E	5000.00	hardware

CS/INFO 330

41

Purchase Relation

- Relation schema:
Purchase(
tid: integer,
tdate: date,
cid: integer,
pid: integer)
- Relation instance:

tid	tdate	cid	pid
1	1/ 1/ 2000	1	1
1	1/ 1/ 2000	1	2
2	1/ 1/ 2000	1	4
3	2/ 1/ 2000	2	3
3	2/ 1/ 2000	2	4

CS/INFO 330

42

Object-Oriented Data Model

- Richer data model
 - Goal: Bridge impedance mismatch between programming languages and database system
 - Example components of the data model: Relationships between objects directly as pointers
- Can store abstract data types directly in DBMS
 - Pictures
 - Geographic coordinates
 - Movies
 - CAD objects

CS/INFO 330

43

Object-Oriented DBMS

- Advantages:
 - Engineering applications (CAD and CAM and CASE computer aided software engineering), multimedia applications.
- Disadvantages:
 - Querying is much harder

The simpler the data model, the easier it is to achieve high performance.

CS/INFO 330

44

Object-Relational DBMS

- Mixture between object-oriented and relational data model
 - Combines ease of querying with ability to store abstract data types
 - Adds richer data types to relational DBMS
- All major relational vendors have added object-relational functionality to some degree

CS/INFO 330

45

XML

Computer Table

<i>Id</i>	<i>Speed</i>	<i>RAM</i>	<i>HD</i>
101	800Mhz	256MB	40GB
102	933Mhz	512MB	40GB

XML is more flexible than the relational model

- No schema necessary
- Irregular structure easily supported

```
<Table>
<Computer Id='101'>
  <Speed>800Mhz</Speed>
  <RAM>256MB</RAM>
  <HD>40GB</HD>
</Computer>
<Computer Id='102'>
  <Speed>933Mhz</Speed>
  <RAM>512MB</RAM>
  <HD>40GB</HD>
</Computer>
</Table>
```

Query Languages

We need a high-level language to describe and manipulate the data.

- Requirements:
 - Precise semantics
 - Easy integration into applications written in C++/Java/Visual Basic/etc.
 - Easy to learn
 - DBMS needs to be able to efficiently evaluate queries written in the language

Relational Query Languages

- The relational model supports simple, powerful querying of data
 - Precise semantics for relational queries
 - Efficient execution of queries by the DBMS
 - Independent of physical storage

SQL: Structured Query Language

- Developed by IBM (System R) in the 1970s
- ANSI standard since 1986:
 - SQL-86
 - SQL-89 (minor revision)
 - SQL-92 (major revision, current standard)
 - SQL-99 (major extensions)
- More about SQL in later lectures

CS/INFO 330

49

Example Query

```
SELECT  
  cid,  
  name,  
  byear,  
  state
```

```
FROM Customers
```

```
WHERE cid = 3
```

cid	name	byear	state
1	Jones	1960	NY
2	Smith	1974	CA
3	Smith	1990	NY

cid	name	byear	state
3	Smith	1990	NY

CS/INFO 330

50

Why Store Data in a DBMS?

- Benefits
 - Transactions (concurrent data access, recovery from system crashes)
 - High-level abstractions for data access, manipulation, and administration
 - **Data integrity and security**
 - Performance and scalability

CS/INFO 330

51

Integrity Constraints

- Integrity Constraints (ICs): Condition that must be true for any instance of the database
 - Specified when schema is defined
 - Checked when relations are modified
- Legal instance of a relation: One that satisfies all specified ICs
 - DBMS should only allow legal instances
- Example: Domain constraints (data types)

CS/INFO 330

52

Primary Key Constraints

- A set of fields is a **superkey** for a relation if no two distinct tuples can have the same values in all key fields.
- A set of fields is a **key** if the set is a superkey, and none of its subsets is a superkey.
- Example:
 - {cid, name} is a superkey for Customers
 - {cid} is a key for Customers
- Where do primary key constraints come from?

CS/INFO 330

53

Primary Key Constraints (Contd.)

- Can there be more than one key for a relation?
- What is the maximum number of superkeys for a relation with k fields?

CS/INFO 330

54

Where do ICs Come From?

- Based upon the semantics of the real-world enterprise that is being described in the database relations
- We can check a database instance to see if an IC is violated, but we can **never** infer ICs from an instance.
 - An IC is a statement about **all possible** instances

cid	name	byear	state
1	Jones	1960	NY
2	Smith	1974	CA
3	Smith	1980	NY

State cannot be a key by itself

- Key and foreign key ICs are very common; a DBMS supports more general ICs.

CS/INFO 330

55

Security

- **Secrecy**: Users should not be able to see things they are not supposed to.
 - Student cannot see other students' grades
- **Integrity**: Users should not be able to modify things they are not supposed to.
 - Only instructors can assign grades
- **Availability**: Users should be able to see and modify things they are allowed to.

CS/INFO 330

56

Why Store Data in a DBMS?

- **Benefits**
 - Transactions (concurrent data access, recovery from system crashes)
 - High-level abstractions for data access, manipulation, and administration
 - Data integrity and security
 - **Performance and scalability**

CS/INFO 330

57

DBMS and Performance

- Efficient implementation of all database operations
- Indexes
 - Auxiliary structures that allow fast access to the portion of data that a query is about
- Smart buffer management
- Query optimization
 - Finds the best way to execute a query
- Automatic high-performance concurrent query execution, query parallelization

CS/INFO 330

58

Summary Of DBMS Benefits

- Transactions
 - ACID properties, concurrency control, recovery
- High-level abstractions for data access
 - Data models
- Data integrity and security
 - Key constraints, foreign key constraints, access control
- Performance and scalability
 - Parallel DBMS, distributed DBMS, performance tuning
- Note: We will also talk about DBMS limitations.

CS/INFO 330

59
