Course Introduction

CSC343 - Introduction to Databases

Sina Meraji

Thanks to Ryan Johnson, John Mylopoulos, Arnold Rosenbloom and Renee Miller for material in these slides

COURSE ADMINISTRIVIA

Course administrivia

- Read the course syllabus online!
- Course Website:
  - http://www.cdf.toronto.edu/~csc343h/summer/index.shtml
- Contact:
  - website and discussion board (MYbb) are required reading
  - personal matters: email me (include "343" in the subject)
- Office hours:
  - Wed 5-6 –

Course administrivia(cont.)

- TAs
  - Fatemeh Nargesian(fnargesian at cs.toronto.edu)
  - Aakar Gupta (aakar at cs.toronto.edu)
  - Krish Perumal(krishperumal11 at gmail.com)
- Tutorials
  - Wed 8:00-9:00 pm at BA1240, BA2145, BA3116

Course prerequisites

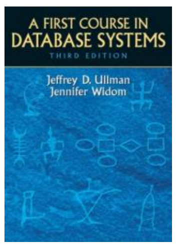
- A&S students
  - CSC165/240 or MAT137/157 and
  - CSC207
    Engineering students
  - Talk to me

Course Marking Scheme

Work	Weight	Comment
3 Assignments	30%	10% each
1 Midterm Test	15%	-
Final Exam	45%	You must get >=40% to pass the course

Recommended Resources

- A First Course in Database Systems, by Jeffrey D. Ullman and Jennifer Widom, 2008 (3rd Edition)



- Jennifer Widom's online mini-courses from Stanford (link on the website)

Assignment Policies

- You may work with a partner on assignments
  - Can change partners between assignments
  - You may not dissolve a partnership in an assignment without permission
- Assignments must be submitted via MarkUs
  - Your code must run on our lab computers ("cdf")
- Late policy
  - 10% per day
  - You can submit up to 2 days late
  - Submit on time!

To-do List

9

- Anyone new to the CDF labs:
  - You all have a CDF account; see details on the course website
  - Try logging in
- Read the course syllabus
- Bookmark the course website

Today

10

- What is a database?
- Relational Model

Today

11

- What is a database?
- Relational Model

What is a database *system*?

12

- Database: a large, integrated collection of data
- Models *relevant aspects of reality*
  - Entities (teams, players)
  - Relationships (Lionel Messi *plays for* Barcelona)
  - Constraints (*at least* one goalkeeper per team)
  - More recently, active components ("business logic")
- *Database Management System (DBMS):* a software system designed to store, manage, and facilitate access to databases

In the beginning…

- There was *The Mainframe*
  - Cost: millions
  - Watts: millions
  - Size: 2000 m^2
  - MIPS: 0.04
  - Memory: 2kB
  - Storage: 3.5MB (tape)

SAGE (1954)

Few organizations could afford two!

Early computing challenges

- Time sharing
  - ~100 terminals per mainframe
  - Users share hardware
  - Want to share *data*, too

SABRE (1960)

=> "The Database"

- Bare hardware
  - No OS
  - No device drivers
  - No file system

UNIVAC (1951)

=> File Management System

"The Database"

- Abstract concept dating back to the 1950's
  - *Centralized* repository for all the enterprise's data
  - *Real-time* updates from many sources
  - *Concurrent* access by many users
  - *Interactive* (ad-hoc) exploration and reporting

Example System: Semi Automatic Ground Environment (SAGE)

- **Goal**: Produce a single unified image of the airspace over an area
- Computer-aided tracking and interception of aircraft
- Dozens of SAGE installations (big one in North Bay)
- Hundreds of radar stations throughout North America
- Thousands of operators

Goal: all relevant information at your fingertips

File management systems (FMS)

- File management ca. 1935
  - **File**: box of punchcards
  - **Metadata**: label on the box
  - **Ad-hoc report**: no big deal
  - **Hardware change**: no big deal
- File management ca. 1955
  - **File**: several km of magnetic tape
  - **Metadata**: embedded in application logic
  - **Ad-hoc report**: hire a couple programmers
  - **Hardware change**: hire a dozen programmers…

Huge need for portability, abstraction

Database Management System

- File management systems meet *The Database*
  - Protect users from each other (isolation, consistency)
  - Protect application from data changes (at logical level)
  - Protect data from hardware changes (at physical level)
- Split personality remains to this day
  - Theory/applications (declarative access to changing data)
  - Systems (make it run fast on ever-changing hardware)

This semester: the theory/application side

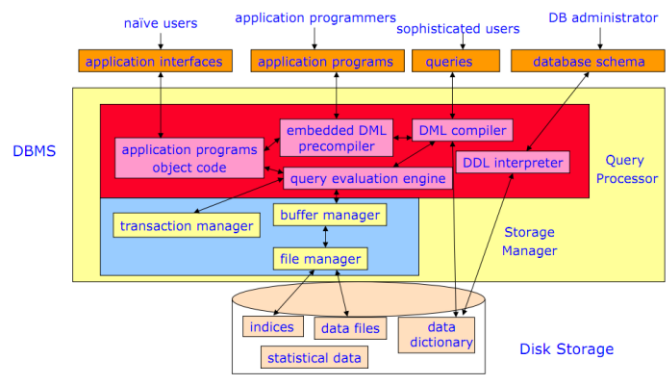
CSC343 Topics

- The Relational Data Model
- Relational Algebra
- Defining, Querying and Manipulating Databases
  - the Structured Query Language (SQL)
- Application Programming with SQL
- Database Design and Normalization
- Modeling and Querying semi-structured data
- Other Topics (SQL security, NoSQL, …)

Why study databases?

- Shift from computation to information
  - always true for corporate computing
  - Web made this point for personal computing
  - more and more true for scientific computing
- Need for DBMS has exploded
  - Corporate: retail swipe/clickstreams, “customer relationship mgmt”, “supply chain mgmt”, “data warehouses”, etc.
  - Scientific: digital libraries, Human Genome project, Sloan Digital Sky Survey, physical sensors, grid physics network
- A practical discipline spanning much of CS
  - OS, languages, theory, AI, multimedia, logic
  - Yet with a focus on real-world apps

DBMS High-level Architecture



Advantages of a DBMS

- Data independence
- Efficient data access
- Data integrity & security
- Data administration
- Concurrent access, crash recovery
- Reduced application development time
- So why not use them always?
  - Expensive/complicated to set up & maintain
  - Cost & complexity must be offset by need
  - General-purpose, not suited for special-purpose tasks (e.g. text search!)

What comes next?

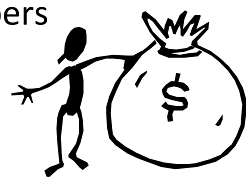
- If you are heading for industry:
  - Database professionals are in demand and well paid
- If you want to do research:
  - Many interesting problems ahead [The Claremont Report'08]
    - Revisiting Database Engines
    - Declarative Programming for Emerging Platforms
    - The Interplay of Structured and Unstructured Data
    - Cloud Data Services
    - Mobile Applications and Virtual Worlds
- Further studies in databases at DCS:
  - csc443: Database Systems Technology
  - csc2508: Advanced DBMSs (grad course, taken with permission)

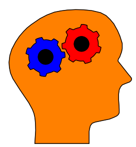
Summary (part 1)

- DBMS marries two very old concepts
  - The Database (idealistic vision)
  - File management system (imminently practical)
- Benefits
  - Maintain, query large datasets
  - Manipulate data and exploit semantics
  - Recover from system crashes
  - Juggle/balance concurrent access, automatic parallelization
  - Quick application development
  - Preserve data integrity and security
- Powerful abstractions provide data independence
  - Application safe from changes to data organization, hardware

Summary (cont.)

DB administrators, developers are the bedrock of the information economy





Data management R&D spans a broad, fundamental branch of the science of computation

This semester: become an effective DBMS user

Today

- What is a database?
- Relational Model

Data Models

- Data model: a notation for describing data, including
  - the structure of the data
  - constraints on the content of the data
  - operations on the data
- Many possible data models:
  - network data model
  - hierarchical data model
  - relational data model -- the most widely used
  - semistructured model (later in the term)

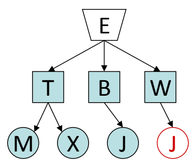
Comparing data models

Student job example

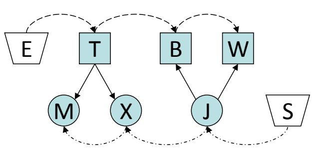
Mary (M) and Xiao (X) both work at Tim Hortons (T)

Jaspreet (J) works at both Bookstore (B) and Wind (W)

Hierarchical (tree)



Network (graph)



Relational (table)

E

B	…
T	…
W	…

S

J	…
M	…
X	…

R

M	T
X	T
J	B
J	W

Why the relational model?

- Matches how we think about data
- Real reason: *data independence*!
- Earlier models tied to physical data layout
  - Procedural access to data (low-level, explicit access)
  - Relationships stored in data (linked lists, trees, etc.)
  - Change in data layout => application rewrite
- Relational model
  - Declarative access to data (system optimizes for you)
  - Relationships specified by queries (schemas help, too)
  - Develop, maintain apps and data layout separately

Similar battle today with languages

What is the relational model?

- Logical representation of data
  - Two-dimensional tables (relations)
- Formal system for manipulating relations
  - Relational algebra (coming next)
- Result
  - High-level (logical, declarative) description of data
  - Mechanical rules for rewriting/optimizing low-level access
  - Formal methods to reason about soundness

Relational algebra is the key

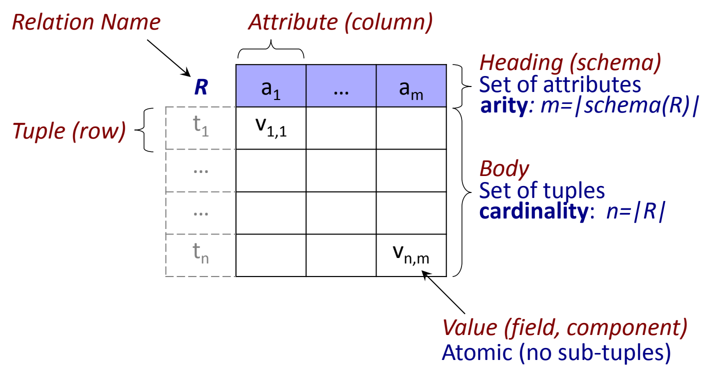
The Relational Model

- Proposed by Edgar F. Codd in 1970 (Turing Award, 1981) as a data model that strongly supports data independence
- Made available in commercial DBMSs in 1981 -- it is not easy to implement data independence efficiently and reliably!
- It is based on (a variant of) the mathematical notion of *relation*
- Relations are represented as tables

Mathematical Relations

- Given sets D_1, D_2, …, D_n, not necessarily distinct, the *Cartesian product* $D_1 \times D_2 \times \ldots \times D_n$ is the set of all (ordered) n-tuples $<d_1,d_2, \ldots,d_n>$ such that $d_1 \in D_1$, $d_2 \in D_2$, …, $d_n \in D_n$
- A *mathematical relation* on D_1, D_2, …, D_n is a subset of the Cartesian product $D_1 \times D_2 \times \ldots \times D_n$.
- D_1, D_2, …, D_n are *domains* of the relation, while n is the *degree* of the relation.
- The number of n-tuples in a given relation is the *cardinality* of that relation

Relations (tables) and tuples (rows)



An Example

- Games ⊆ String x String x Integer x Integer

Juve	Lazio	3	1
Lazio	Milan	2	0
Juve	Roma	1	2
Roma	Milan	0	1

- Note that String and Integer each play two roles, distinguished by means of position
- The structure of a mathematical relation is positional

Attributes

- We can make the structure of a relation non-positional by associating a unique name (attribute) with each domain that describes its role in the relation
- In the tabular representation, attributes are used as column headings

HomeTeam	VisitingTeam	HomeGoals	VisitorGoals
Juve	Lazio	3	1
Lazio	Milan	2	0
Juve	Roma	1	2
Roma	Milan	0	1

Notation

- t[A] (or t.A) denotes the value on attribute A for a tuple t
- In our example, if t is the first tuple in the table

  t[VisitingTeam] = Lazio
- The same notation is extended to sets of attributes, thus denoting tuples:

  t[VisitingTeam,VisitorGoals] is a tuple on two attributes, <Lazio,1>
- More generally, if X is a sequence of attributes A1,...An, t[X] is <t[A1],t[A2],...t[An]>

Value-based References

Students

RegNum	Surname	FirstName	BirthDate
6554	Rossi	Mario	5/12/1978
8765	Neri	Paolo	3/11/1976
9283	Verdi	Luisa	12/11/1979
3456	Rossi	Maria	1/2/1978

Exams

Student	Grade	Course
3456	30	04
3456	24	02
9283	28	01
6554	26	01

Courses

Code	Title	Tutor
01	Analisi	Neri
02	Chimica	Bruni
04	Chimica	Verdi

Value-based References (cont.)

Students

RegNum	Surname	FirstName	BirthDate
6554	Rossi	Mario	5/12/1978
8765	Neri	Paolo	3/11/1976
9283	Verdi	Luisa	12/11/1979
3456	Rossi	Maria	1/2/1978

Exams

Student	Grade	Course
	30	
	24	
	28	
	26	

Courses

Code	Title	Tutor
01	Analisi	Neri
02	Chimica	Bruni
04	Chimica	Verdi

Advantages of Value-Based References

- Value-based references lead to independence from physical data structures, such as pointers
  - Pointers are implemented differently on different hardware, inhibit portability of a database

Definitions

- Relation schema: Relation name R with a set of attributes A1,..., An:

  R(A1,..., An)
- Database schema: A set of relation schemas with different names

  D = {R1(X1), ..., Rn(Xn)}
- Relation (instance) on a relation schema

  R(X): Set r of tuples on X
- Database (instance) on a schema

  D= {R1(X1), ..., Rn(Xn)}: Set of relations r = {r1,..., rn} (where ri is a relation on Ri)

Example Data

Da Mario		
	Receipt No: 1357	
	Date: 5/5/92	
3	covers	3.00
2	hors d'oeuvre	5.00
3	first course	9.00
2	steak	12.00
	Total:	29.00

Da Mario		
	Receipt No: 2334	
	Date: 4/7/92	
2	covers	2.00
2	hors d'oeuvre	2.50
2	first course	6.00
2	bream	15.00
2	coffee	2.00
	Total:	27.50

Da Mario		
	Receipt No: 3007	
	Date: 4/8/92	
2	covers	3.00
2	hors d'oeuvre	6.00
3	first course	8.00
1	bream	7.50
1	salad	3.00
2	coffee	2.00
	Total:	29.50

Data Representation

Details

Number	Quantity	Description	Cost
1357	3	Covers	3.00
1357	2	Hors d'oeuvre	5.00
1357	3	First course	9.00
1357	2	Steak	12.00
2334	2	Covers	2.00
2334	2	Hors d'oeuvre	2.50
2334	2	First course	6.00
2334	2	Bream	15.00
2334	2	Coffee	2.00
3007	2	Covers	3.00
3007	2	Hors d'oeuvre	6.00
3007	3	First course	8.00
3007	1	Bream	7.50
3007	1	Salad	3.00
3007	2	Coffee	2.00

Receipts

Number	Date	Total
1357	5/5/92	29.00
2334	4/7/92	27.50
3007	4/8/92	29.50

Questions

- Have we represented all details of receipts?
- Well, it depends on what we are interested in:
  - does the order of lines matter?
  - could we have duplicate lines in a receipt?
    - If so, there is a problem … Why?
- If needed, an alternative representation is possible …

More Detailed Representation

Details

Number	Line	Quantity	Description	Cost
1357	1	3	Covers	3.00
1357	2	2	Hors d'oeuvre	5.00
1357	3	3	First course	9.00
1357	4	2	Steak	12.00
2334	1	2	Covers	2.00
2334	2	2	Hors d'oeuvre	2.50
2334	3	2	First course	6.00
2334	4	2	Bream	15.00
2334	5	2	Coffee	2.00
3007	1	2	Covers	3.00
3007	2	2	Hors d'oeuvre	6.00
3007	3	3	First course	8.00
3007	4	1	Bream	7.50
3007	5	1	Salad	3.00
3007	6	2	Coffee	2.00

Receipts

Number	Date	Total
1357	5/5/92	29.00
2334	4/7/92	27.50
3007	4/8/92	29.50

Incomplete Information: Motivation

(County towns have government offices, other towns do not.)

- Florence is a county town; so it has a government office, but we do not know its address
- Tivoli is not a county town; so it has no government office
- Prato has recently become a county town; has the government office been established? We don‘t know!

City	GovtAddress
Roma	Via IV novembre
Florence	?
Tivoli	??
Prato	???

Null Value

- A null value is a special value (not a value of any domain) which denotes the absence of a value
- Types of Null Values:
  - unknown value: there is a domain value, but it is not known (Florence)
  - non-existent value: the attribute is not applicable for the tuple (Tivoli)
  - no-information value: we don't know if a value exists or not (Prato). (This is the disjunction - logical or - of the other two)
- DBMSs do not distinguish between these types: they implicitly adopt the no-information value

A Meaningless Database …

Exams

RegNum	Name	Course	Grade	Honours
6554	Rossi	B01	K	
8765	Neri	B03	C	
3456	Bruni	B04	B	honours
3456	Verdi	B03	A	honours

Courses

Code	Title
B01	Physics
B02	Calculus
B03	Chemistry

Honours are awarded only if grade is A. Can you spot some others?

Integrity Constraints

- An integrity constraint is a property that must be satisfied by all meaningful database instances
- A database is legal if it satisfies all integrity constraints
- Types of constraints:
  - Intra-relational constraints
    - domain constraints
    - tuple constraints
    - keys
  - Inter-relational constraints
    - foreign keys

Rationale for Integrity Constraints

- Describe the application in greater detail
- Contribute to data quality
- An important part of the database design process (we will discuss later normal forms)
- Used by the system in choosing a strategy for query processing

Tuple and Domain Constraints

- A *tuple constraint* expresses conditions on the values of each tuple, independently of other tuples
  - `NOT((Honours = 'honours') OR (Grade ='A'))`
  - `Net = Gross - Deductions`
- A *domain constraint* is a tuple constraint that involves a single attribute
  - `(Grade ≤ 'A') AND (Grade ≥ 'F')`

Unique Identification for Tuples

RegNum	Surname	FirstName	BirthDate	DegreeProg
284328	Smith	Luigi	29/04/59	Computing
296328	Smith	John	29/04/59	Computing
587614	Smith	Lucy	01/05/61	Engineering
934856	Black	Lucy	01/05/61	Fine Art
965536	Black	Lucy	05/03/58	Fine Art

- Registration number identifies students
  - no pair of tuples with the same value for RegNum
- Personal data could identify students as well
  - E.g. no pair of tuples with the same values for all of Surname, FirstName, BirthDate

Keys

- A *key* is a set of attributes that uniquely identifies tuples in a relation
- More formally:
  - A set of attributes K is a *superkey* for a relation r if r can not contain two distinct tuples t_1 and t_2 such that $t_1[K] = t_2[K]$;
  - K is a *key* for r if K is a minimal superkey; that is, there exists no other superkey K' such that $K' \subset K$

An Example

RegNum	Surname	FirstName	BirthDate	DegreeProg
284328	Smith	Luigi	29/04/59	Computing
296328	Smith	John	29/04/59	Computing
587614	Smith	Lucy	01/05/61	Engineering
934856	Black	Lucy	01/05/61	Fine Art
965536	Black	Lucy	05/03/58	Fine Art

- RegNum is a key
  - RegNum is a superkey and it contains a sole attribute, so it is minimal
- Surname, Firstname, BirthDate is a key
  - the three attributes form a superkey and there is no proper subset that is also a superkey

Beware!

RegNum	Surname	FirstName	BirthDate	DegreeProg
296328	Smith	John	29/04/59	Computing
587614	Smith	Lucy	01/05/61	Engineering
934856	Black	Lucy	01/05/61	Fine Art
965536	Black	Lucy	05/03/58	Engineering

- There is no pair of tuples with the same values on both Surname and DegreeProg; i.e., in each programme students have different surnames.
- Can we conclude that Surname and DegreeProg form a key for this relation? No! There could be students with the same surname in the same programme

Existence of Keys (Proof Sketch)

- Relations are sets; therefore each relation is composed of distinct tuples
- It follows that the whole set of attributes for a relation defines a superkey
- Therefore each relation has a key, which is the set of all its attributes (or a subset thereof)
- The existence of keys guarantees that each piece of data in the database can be accessed

Keys are a major feature of the Relational Model and allow to say that it is "value-based"

Keys and Null Values

- If there are nulls, keys do not work well:
  - They do not guarantee unique identification
  - They do not help in establishing correspondences between data in different relations

RegNum	Surname	FirstName	BirthDate	DegreeProg
NULL	Smith	John	NULL	Computing
587614	Smith	Lucy	01/05/61	Engineering
934856	Black	Lucy	NULL	NULL
NULL	Black	Lucy	05/03/58	Engineering

How do we access the first tuple?

Are the third and fourth tuple the same?

Primary Keys

- The presence of nulls in keys has to be limited
- Each relation must have a primary key on which nulls are not allowed
- Notation: the attributes of the primary key are underlined
- References between relations are realized through primary keys

<u>RegNum</u>	Surname	FirstName	BirthDate	DegreeProg
643976	Smith	John	NULL	Computing
587614	Smith	Lucy	01/05/61	Engineering
934856	Black	Lucy	NULL	NULL
735591	Black	Lucy	05/03/58	Engineering

References Between Relations

Students

<u>RegNum</u>	Surname	FirstName	BirthDate
6554	Rossi	Mario	5/12/1978
8765	Neri	Paolo	3/11/1976
9283	Verdi	Luisa	12/11/1979
3456	Rossi	Maria	1/2/1978

Exams

<u>Student</u>	Grade	<u>Course</u>
3456	30	04
3456	24	02
9283	28	01
6554	26	01

Courses

<u>Code</u>	Title	Tutor
01	Analisi	Neri
02	Chimica	Bruni
04	Chimica	Verdi

Do we Always Have Primary Keys?

- In most cases YES
- In other cases NO
  - need to introduce new attributes by identifying codes
- Goal: Unambiguously identify things
  - social insurance number
  - student number
  - area code
  - …

…Consider…

- Suppose we want a database that maintains information on course offerings at the University of Toronto and use the following relation schema

  Course(name,dept,year,sem)

- What would it mean if we used each of the following attribute sets as a primary key:

  name?

  name,sem,dept?

  sem,year?

  name,dept?

  dept,year?

Referential Constraints (Foreign Keys)

- Data in different relations are referenced through (primary) key values
- Referential integrity constraints are imposed in order to guarantee that the values refer to existing tuples in the referenced relation
- For example, if a student with id “3456” took an exam for course with id “04”, there better be a student with such an id and a course with such an id in the referenced relations
- Also called inclusion dependencies

Example of Referential Constraints

Offences

Code	Date	Officer	Dept	Registration
143256	25/10/1992	567	75	5694 FR
987554	26/10/1992	456	75	5694 FR
987557	26/10/1992	456	75	6544 XY
630876	15/10/1992	456	47	6544 XY
539856	12/10/1992	567	47	6544 XY

Officers

RegNum	Surname	FirstName
567	Brun	Jean
456	Larue	Henri
638	Larue	Jacques

Cars

Registration	Dept	Owner	…
6544 XY	75	Cordon Edouard	…
7122 HT	75	Cordon Edouard	…
5694 FR	75	Latour Hortense	…
6544 XY	47	Mimault Bernard	…

Referential Constraints

- A referential constraint requires that the values on a set X of attributes of a relation R1 must appear as values for the primary key of another relation R2
- In such a situation, we say that X is a foreign key of relation R1
- In the previous example, we have referential constraints between the attribute Officer of the relation Offences and the relation Officers; also between the attributes Registration and Department of relations Offences and Cars.

Violation of Referential Constraints

Offences

Code	Date	Officer	Dept	Registration
987554	26/10/1992	456	75	5694 FR
630876	15/10/1992	456	47	6544 XY

Officers

RegNum	Surname	FirstName
567	Brun	Jean
638	Larue	Jacques

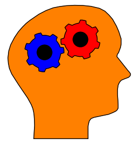
Cars

Registration	Dept	Owner	…
7122 HT	75	Cordon Edouard	…
5694 FR	93	Latour Hortense	…
6544 XY	47	Mimault Bernard	…

Referential Constraints: Comments

- Referential constraints play an important role in making the relational model value-based
- It is possible to have features that support the management of referential constraints (“actions” activated by violations)

Summary



- The relational model
  - Relations, tuples, attributes
  - Value-based References
  - Incomplete information: The NULL value
  - Integrity Constraints
    - domain constraint
    - tuple constraint
    - unique tuple identification constraint (primary key)
    - referential constraints (foreign Key)
- Next: Relational Algebra