Example: build a better Workday

- Professors offer classes, students register, get grades
- What are some questions we (students or faculty) could ask of this database system?
  - Find my GPA.
  - …Show current schedule.
  - Plan out a 4-yr plan for my major
  - Transcript download.
  - Let us see how many other ppl have a class on a same schedule.
  - # of seats left in a class/ wait list.
  - Mon-Fri calendar of classes.

Example: build a better Workday

- Why are security, concurrency, and atomicity important here?

multiple things happening at once.

Grouping changes together so that they all happen or none of them happen.

Solution 1

- Advantages?
  - No down time.
  - Reasonably secure
- Disadvantages?
  - No backups
  - Get lost / damaged
  - Need physical space

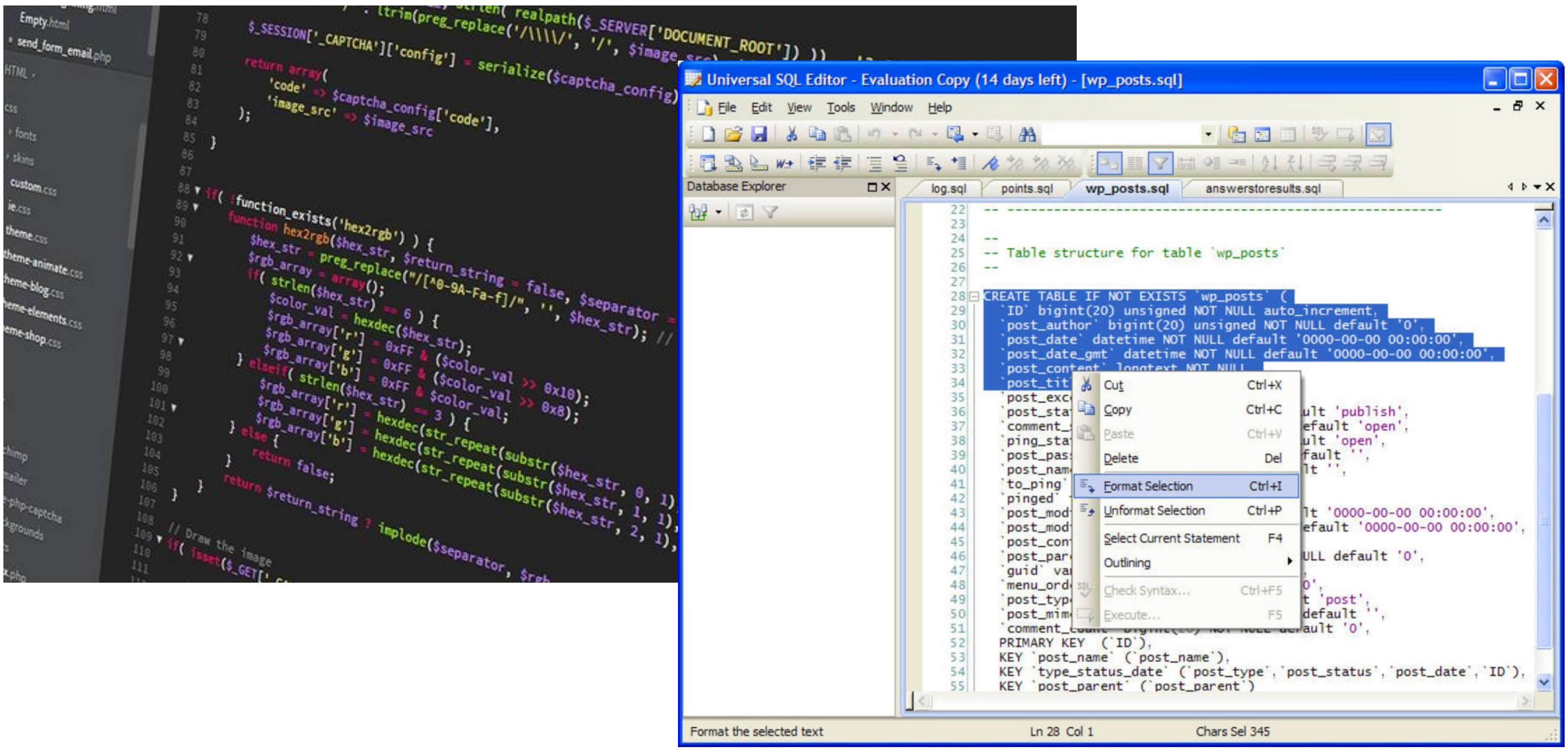
Solution 2

Adv
Backups - easier
Remotely accessible
Easy to send to ppl

Disadv
Hackable/delete

- Text files and Python/Java programs

easier to answer questions



Solution 3

Let's use CSV:
(comma-separated values)

~~last name~~ first last R# Class Grade

```
Hermione,Granger,R123,Potions,A
Draco,Malfoy,R111,Potions,B
Harry,Potter,R234,Potions,A
Ronald,Weasley,R345,Potions,C
```

What's the issue here?

```
*Hermione,Granger,R123,Potions,A
 Draco,Malfoy,R111,Potions,B
 Harry,Potter,R234,Potions,A
 Ronald,Weasley,R345,Potions,C
 Harry,Potter,R234,Herbology,B
*Hermione,Granger,R123,Herbology,A
```

Data duplication (BAD)

File 1:

```
Hermione,Granger,R123
Draco,Malfoy,R111
Harry,Potter,R234
Ronald,Weasley,R345
Ronald, Weasley, R999
```

Normalization

File 2:

```
R123,Potions,A
R111,Potions,B
R234,Potions,A
R345,Potions,C
R234,Herbology,B
R123,Herbology,A
R999 ________, __
```

Problems

- Inconvenient – need to know Python/Java to get at data!
- Redundancy/inconsistency
- Integrity problems : Ex: more students registered for a class than # of seats.
- Atomicity problems
- Concurrent access problems
- Security problems

Why are there problems?

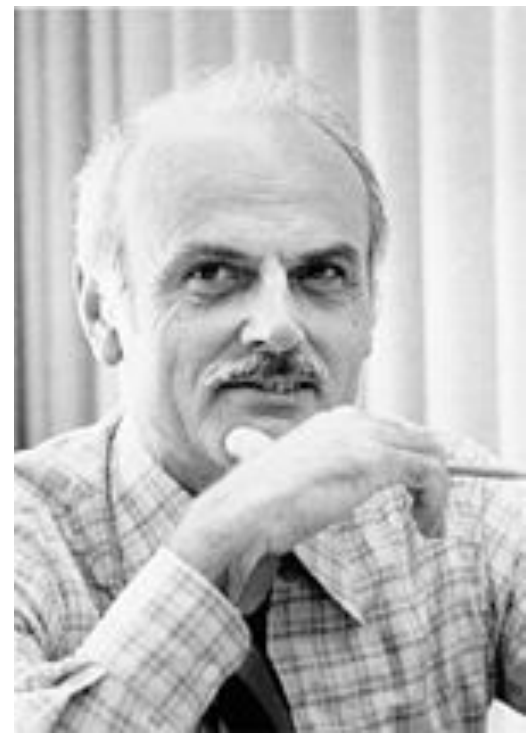
- Two main reasons:
  - The description of how the files are laid out is buried within the Python/Java code itself (if it's documented at all)
  - There is no support for **transactions** (supporting concurrency, atomicity, integrity, and recovery)
- **DBMSs handle exactly these two problems.**

Database management system.

RDBMS

Relational database systems

- Edgar F. Codd was a researcher at IBM who conceived a new way of organizing data based on the mathematical concept of a ***relation***. (1970)
- Relation: a set of ordered tuples (oh, no, CS172 stuff…)



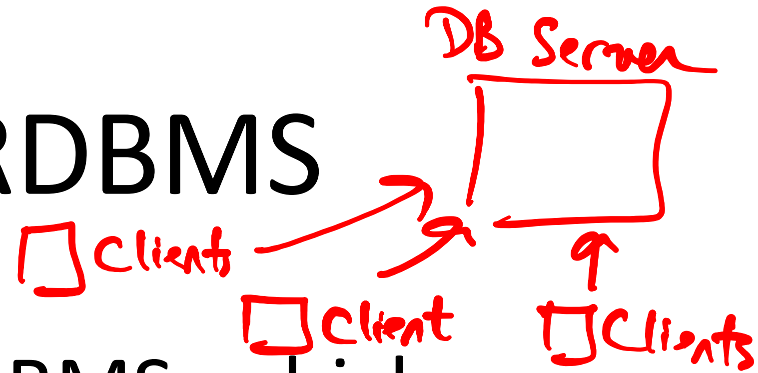
Highlights of RDBMS

- (R)DBMS = relational database management system.
- Data is stored in ***relations***, which resemble tables:

First	Last	Course	Grade
Hermione	Granger	Potions	A
Draco	Malfoy	Potions	B
Harry	Potter	Potions	A
Ronald	Weasley	Potions	C

- Underlying data structures are more complicated.

Highlights of RDBMS



- Users issue ***queries*** to the DBMS, which are handled by the ***query processor***.
  - Behind the scenes: *query optimizer* handles all the details of figuring out the most efficient way to answer the query, which might involve combining multiple tables, sorting the data, selecting only a subset of it, ...
- The ***transaction manager*** handles all the details of atomicity and concurrency.

Data Models

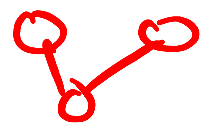
- A way of describing data.
  - Better: a description of how to conceptually structure the data, what operations are possible on the data, and any constraints on the data.
- Structure: how we view the data abstractly
- Operations: what is possible to do with the data?
- Constraints: how can we control what data is legal and what is not?

Relational model

First	Last	Course	Grade
Hermione	Granger	Potions	A
Draco	Malfoy	Potions	B
Harry	Potter	Potions	A
Ronald	Weasley	Potions	C

- Structure: ***relation*** (table)
- Operations: ***relational algebra*** (select certain rows, certain columns, where properties are true/false, also combine multiple tables together to answer more complicated questions), ***(SQL)***
- Constraints: can enforce restrictions like Grade must be in the set {A, B, C, D, F}

Other models

- ***Semi-structured:*** data that is still "structured" but not in relational format.
  - XML, JSON Javascript object notation
- Object databases, or object-relational
- Graph databases 
- NoSQL, NewSQL (document databases, key-value) ← HOT

Semi-structured model

- Structure: Trees or graphs
  - e.g., XML
- Operations: Follow paths in the implied tree from one element to another.
  - e.g., XQuery
- Constraints: can constrain data types, possible values, etc.
  - e.g., DTDs (document type definition), XML Schema

Object-relational

- Similar to relational, but
  - Values in a table can have their own structure, rather than being simple strings or ints.
  - Relations can have associated methods.

NoSQL, NewSQL

- Lots of different types, but main idea is there is no separate schema definition.
- Main reasons for using: conceptually simpler, easy to replicate across clusters of machines, can be faster than relational.
- Drawbacks: harder to write queries, lack of "joins," possible lack of consistency.

redis cassandra mongoDB Couchbase

Relational model is most common

- Simple: built around a single concept for modeling data: the ***relation*** or table.
  - A relational database is a collection of relations.
  - Each relation is a table with rows and columns.
  - An RDBMS can manage many databases at once.
- Supports high-level programming language (SQL)

  Structured query language

  - Limited but useful set of operations.
- Has elegant mathematical theory behind it.

Relation Terminology

- **Relation** == 2D table
  - **Attribute** == column name
  - **Tuple** == row (not the header row)
- **Database** == collection of relations

First	Last	Course	Grade
Hermione	Granger	Potions	A
Draco	Malfoy	Potions	B
Harry	Potter	Potions	A
Ronald	Weasley	Potions	C

4 tuples
4 attribs

Relation Terminology

- A relation includes two parts:
  - The relation **schema** defines the column headings of the table (attribute names)
  - The relation **instance** defines the data rows (tuples, rows, or records) of the table.

First	Last	Course	Grade
Hermione	Granger	Potions	A
Draco	Malfoy	Potions	B
Harry	Potter	Potions	A
Ronald	Weasley	Potions	C

Schema

- A schema is written as the name of the relation followed by a parenthesized list of attributes.
  - `Grades(First, Last, Course, Grade)`
- A **relational database schema** is the set of schemas for all the relations in a DB.

First	Last	Course	Grade
Hermione	Granger	Potions	A
Draco	Malfoy	Potions	B
Harry	Potter	Potions	A
Ronald	Weasley	Potions	C

Tuples

- A tuple is a row of a relation.
- Notation:

  `(Draco, Malfoy, Potions, B)`

First	Last	Course	Grade
Hermione	Granger	Potions	A
Draco	Malfoy	Potions	B
Harry	Potter	Potions	A
Ronald	Weasley	Potions	C

Domains

- A relational DB requires that every component of a row (tuple) have a specific elementary data type, or **domain**.
  - string, int, float, date, time (no complicated objects!)

schema

```
Grades(First:string, Last:string,
Course:string, Grade:char)
```

Equivalent representations of a relation

grades

First	Last	Course	Grade
Hermione	Granger	Potions	A
Draco	Malfoy	Potions	B
Harry	Potter	Potions	A
Ronald	Weasley	Potions	C

```
Grades(First, Last, Course, Grade)
```

- Relation is a ***set*** of tuples, not a list. No order
- Attributes in a schema are a ***set*** as well.
  - However, the schema specifies a "standard" order for the attributes.
- How many equivalent representations are there for a relation with *m* attributes and *n* tuples? $= m! \cdot n!$

F, L, C, G (4) (3) (2) (1)
1 2 3 4

$4! = m!$ $n!$

Degree and cardinality

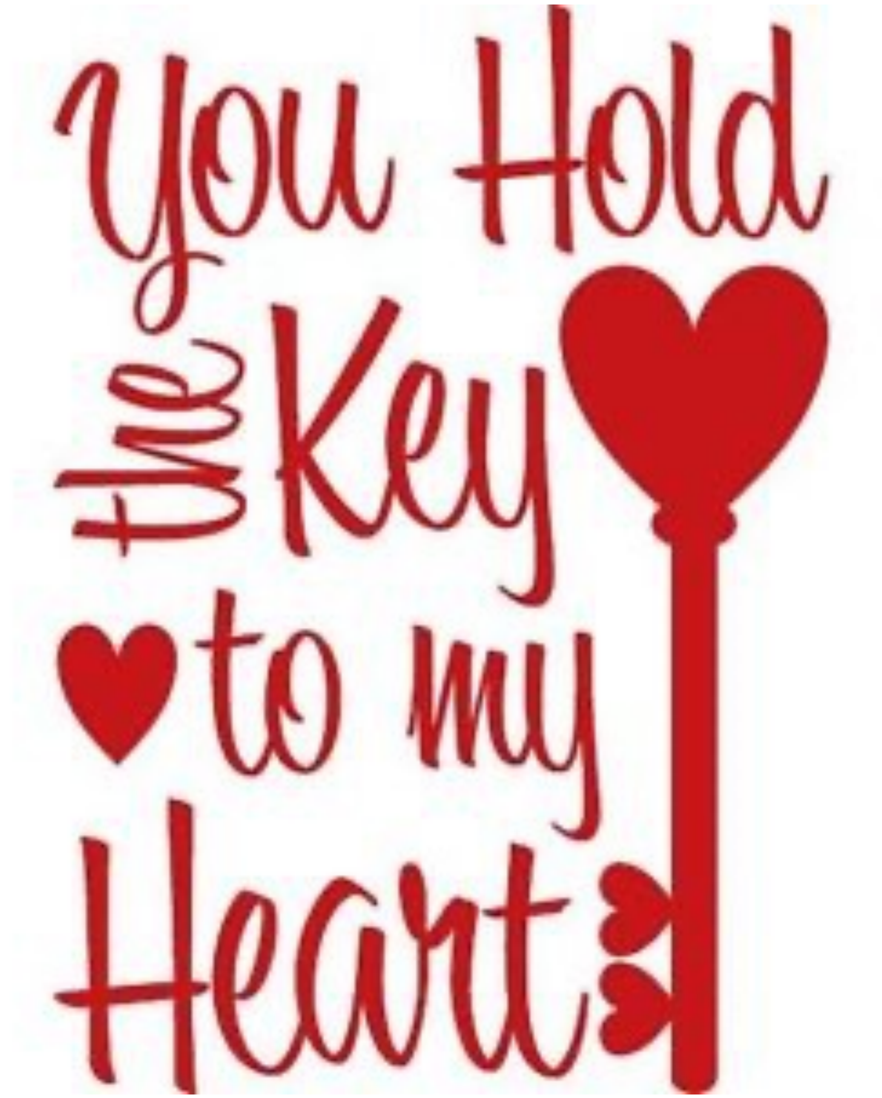
First	Last	Course	Grade
Hermione	Granger	Potions	A
Draco	Malfoy	Potions	B
Harry	Potter	Potions	A
Ronald	Weasley	Potions	C

- **Degree/arity** of a relation is the number of attributes in a relation.

  binary & Relation = 2 cols
  ternary Relation = 3 cols

- **Cardinality** is the number of tuples in a relation.

Keys to a good relation(ship)



Keys to a good relation(ship)

Relation

Keys of a relation

- Keys are a kind of **integrity constraint**.
- A set of attributes *K* forms a key for a relation *R* if
  - no pair of tuples in an instance of *R* may have the same values for *all* attributes of *K*.

First	Last	Course	Grade
Hermione	Granger	Potions	A
Draco	Malfoy	Potions	B
Harry	Potter	Potions	A
Ronald	Weasley	Potions	C

`Grades(First, Last, Course, Grade)`

Artificial Keys

Keys of a relation

- Keys help associate tuples in different relations.

`Students(SID, First, Last)`

SID	First	Last
123	Hermione	Granger
111	Draco	Malfoy
234	Harry	Potter
345	Ronald	Weasley

`Grades(SID, CRN, Grade)`

SID	CRN	Grade
123	777	A
111	777	B
234	777	A
345	777	C

`Courses(CRN, Name, Sem., Year)`

CRN	Name	Semester	Year
777	Potions	Fall	1997
888	Potions	Spring	1997
999	Transfiguration	Fall	1996
789	Transfiguration	Spring	1996

Example

- Let's expand these relations to handle the kinds of things you'd like to see in BannerWeb.
- Keep track of students, professors, courses, who teaches what, enrollments, pre-requisites, grades, departments & their chairs.
  - Only one chair per department.
  - Student cannot enroll in multiple copies of the same course in one semester.
  - Other constraints that are logical.

- Keep track of students, professors, courses, who teaches what, enrollments, pre-requisites, grades, departments & their chairs.
  - Only one chair per department.
  - Student cannot enroll in multiple copies of the same course in one semester.
  - Other constraints that are logical.

So far: `Students(SID, First, Last)`

`Courses(CRN, Name, Sem., Year)`

`Grades(SID, CRN, Grade)`