CIS4301 Notes

Ryan Roden-Corrent

Wed Jan 22 10:49:19 EST 2014

1 Functions of a Database

- **Create** - Add new records
- **Query** - Search for records meeting a description
- **Store** - Hold data in records
- **Durability** - data remains available and uncorrupted (backups)
  Example: Oracle provides prompt maintenance if a server you bought from them goes down
- **Access** - Manageable permissions. Control who can see/edit data

2 History of Databases

1870s

Charles Babbage and Ada Lovelace create the difference engine, and later, the analytic engine (mechanical computers)

1876

Invention of Dewey Decimal system: a general cataloging system for libraries that is still in use today.

1962

Bachman: integrated data store

- **network data model**: a tree with no loops
- **graph data model**: loops allowed

models are specific to the data being represented

1960s

Disks (random access) replace **tapes** (serial access). Now one can access data without reading the preceding data.
At this time, people interested in databases are mostly programmers.
IBM Builds **IMS** (Information Management System) to track Apollo.
CODASYL: Commitee standardizzed network data model

1970s

Edward Codd writes 'A relational model of data for large shared banks'. It conflicts with the design of the current most popular databases. Eventually, IBM agrees to dedicate a research team to Codd's idea; this team produces SQUARE

1977

Another IBM team develops **Query By Example (QBE)**.
A group of CIA employees branch off to found **Oracle** and create the first commercial implementation of **SEQUEL**.
SEQUEL stands for **Structured English Query Language** but is abbreviated to **SQL** to avoid a copyright dispute.

1990s

Research into object oriented database implementations

2000s

Web Boom: Many services suddenly need to store massive amounts of data.

3 NOSQL

1970 No SQL - SQL doesn't exist

1980 SQL invented, standard for databases

1990 No, SQL - If you want to do database stuff, use SQL

2000 Not Only SQL - There are other things out there

2010 No, really, use SQL

4 Is a filesystem a database?

Create can create and remove records (touch, rm, mkdir, rmdir)

Query can search records (find, windows search bar)

Store files can sotre content that can be read back

Durability debateable

Access issues arise if two sources try to edit the same file

Which of these services are actually part of the filesystem, and which are really provided by the operating system?

5 Applications Of Databases

Bank Store Account information

Social Media Site User info, posts, need to find users/posts by query

Advertising Store and utilize user data for targeted advertising

Music Library Query music by artist, album, or other property

Employee Records Name, salary, ect.

5.1 Early Database Adopters

Early adopters of database systems were banks, airlines, telecom companies, and companies interested in tracking employee data.

5.2 Large Database Users

1. Google/Youtube
2. NSA
3. Amazon
4. Yahoo
5. Yahoo
6. Facebook
7. Sprint
8. AT&T
9. Apple
10. National Energy Research Scientific Computing (NERSC)
11. Lexis Nexis

All of these companies keep track of **user-generated content**

6 An overview of DMBS

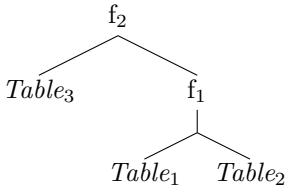
DDL: Data Definition Language
CREATE, ALTER, DROP, TRUNCATE, COMMENT, RENAME
DML: Data Manipulation Language
SELECT, INSERT, UPDATE, DELETE

6.1 Flow of a query

$Query \rightarrow QueryParser \rightarrow QueryPlan \rightarrow QueryPreprocessor \rightarrow QueryOptimizer \rightarrow QueryPlan$
SQL is declaritive - say what you want, and an optimal search method is chosen for you

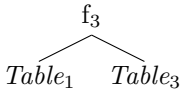
6.2 Query Structure

$f_2(f_1(Table_1, Table_2)), Table_3)$



The query optimizer may collapse this

$f_3(Table_1, Table_3)$



6.3 Parts of a Database

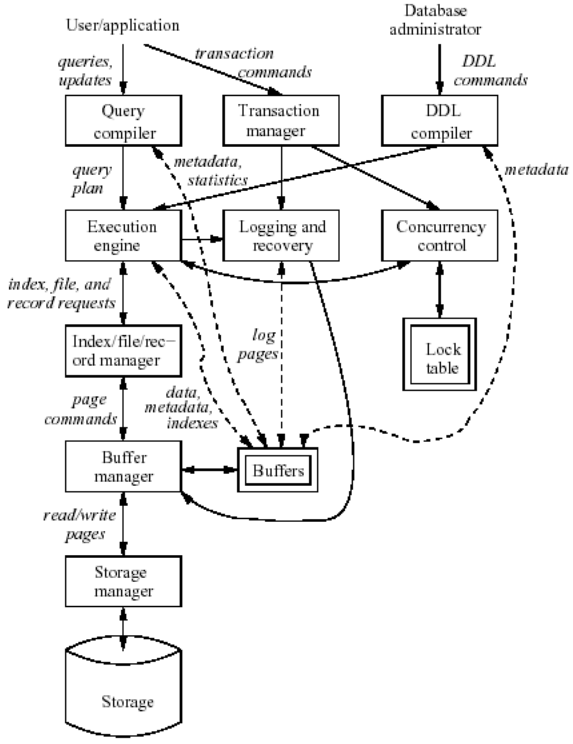


Figure 1: Database Flow Diagram

Query Compiler Form query

Index/File/Record Manager Can speed up queries using index

Buffer Manager Decide whether to cache in RAM (much faster than disk!)

Storage Manager Provide access to data on disk in *pages*

Transaction Manager Process transactions (see next section)

7 Transactions

Transactions are operations that must be executed reliably and consistently. Order is important, and transactions should not interfere with eachother. In **multi-threaded** applications, race-conditions and deadlocks can occur while trying to execute transactions. Deadlock resolution must occur. Transactions are atomic - they need to happen all at once or not at all. The transaction manager must initiate **rollback** (undo) in case of an error. Transactions are run in isolation from other database operations.

7.1 ACID

Atomicity transaction happens just once or not at all

Consistency predictable ordering, same result every time

Isolation no interference between transactions

Durability no data missing after transaction

7.2 Transaction Example

```
cg == 1
T_1
BEGIN
1/13 | cg | +5
1/14 | cg | -1
END;
cg == 5

T_2
BEGIN
1/13 | cg | -5
1/13 | cg | +1
END;

End result should be the same regardless of order of execution
T_1; T_2 == 1
T_1; T_2 == 1
```

8 Relational Data Model

1. Physical Model
2. Operations on data
3. Constraints on data

8.1 Physical Model Example

```
public class Animal
{
    public int Eyes;
    public String birthDate;
    public double Weight;

    public void beCute();
}

Animal a = new Animal();
[header][integer][birthDate...][double][beCute]
```

Constraints:

- birthDate can't be null
- Eyes can't be negative

8.2 Relational Model

Separate data from operations.

Relations no more than 2

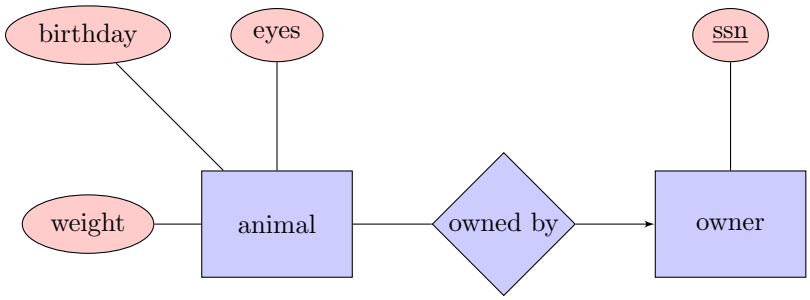
Attributes order matters, **atomic** (int,double,String)

Tuples

Domain

Keys uniquely define attribute instance

8.3 Relational Model Example



The underline indicates that ssn is the unique key for an owner. As the set (birthday, weight, eyes) uniquely identifies an animal, they do not need to be underlined.

```
Animal(eyes:integer, birthDate:String, weight:double)
(0001, 2, "November 2", 12.0) //cannot have 2 of these, must be unique
(0002, 3, "November 2", 12.0)
```

```
//don't have to store entire animal, could just store certain columns
var a[] = [2,3]
```

8.4 Thinking with Relational Models

Don't think about how data will be stored, just how it will be used.
Entity Relational Model:

Entity/Sets

Relationships

Attributes

8.5 Constraints

many-many

one-one

one-many/many-one : a pet only has one owner, an owner may have many pets

Figure 2: one-one relation

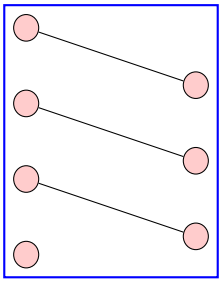
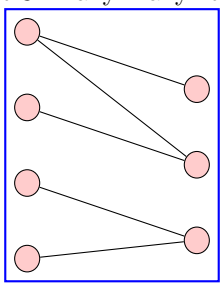


Figure 3: many-many relation



9 Data Models

9.1 Important Aspects

Efficient Access

Efficient Modification

Programmer Productivity

9.2 JAVA Data Access example

```
class Animal {}

Animal kizzy = new Animal();
kizzy.getEyeCount();

Vector<Animals> cats = new Vector<>();
cats.get(3).talk();
```

```
// Options for serialization
//serialize as binary
byte[]catBytes = cats.serialize();

try {
  http.push(catBytes);
}

//serialize as XML
<cats>
  <cat name='kizzy'>
    <name>kizzy</name>
  </cat>
</cat>

//serialize as JSON
cats = {
  cat = [ { name: 'kizzy', eyes: 2}, { name: 'garfield' } ]
}

//later
Vector<Cats> v = cats.unserialize(bytes);

//search for a specific element
for (Cats c : v) {
  if (c.name = 'kizzy')
    //found it
}
```

Efficient Access Get from disk and serialize

Efficient Modification Get from disk, deserialize, modify, reserialize

Programmer Productivity

9.3 SQL Data Access example

```
CREATE DATABASE cats;
CREATE TABLE cats (
  name VARCHAR,
  eyes integer
);
INSERT cats VALUES ('kizzy', '2');

SELECT name
FROM cats
WHERE name = 'kizzy'
```