specifically, SQL is a domain specific programming language, a DSL.

But we don't program applications using SQL, it's used as a subroutine. we program using general languages like python or java or c++ or go.
And in these languages, how do we access and use a database?
What is the difference between application programming and database programming?
Can they interoperate naturally, or do we need to fixd some issues before it just works?

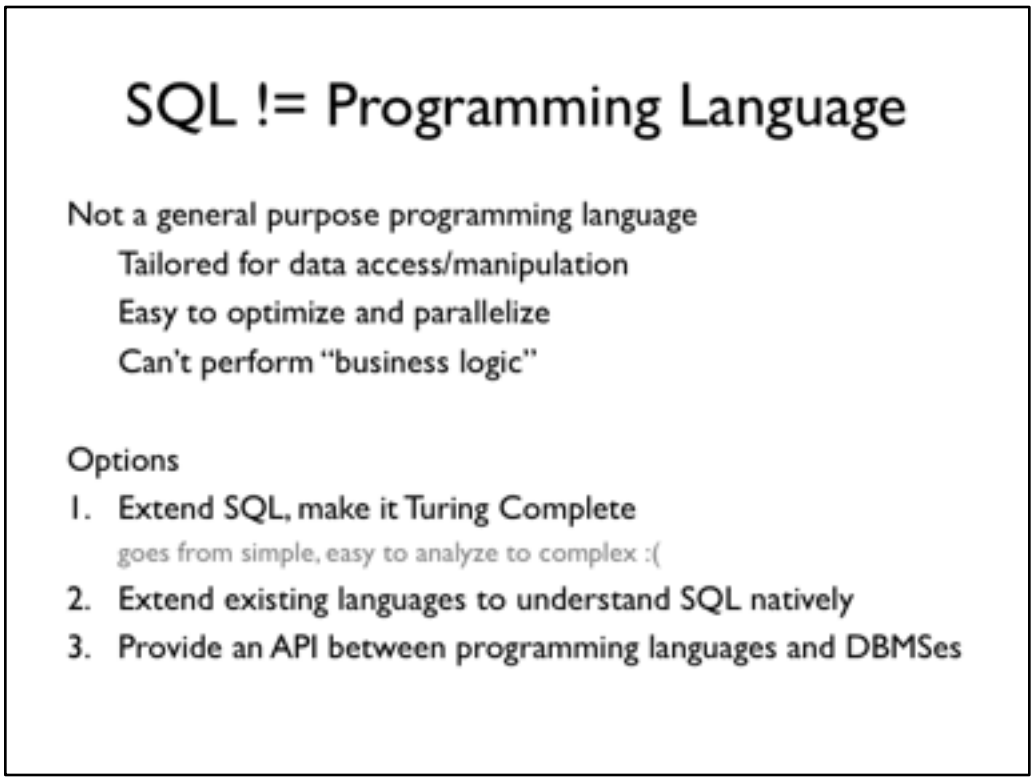
We already saw one example with user defined functions, in that context, we embed the programming language in the database, and we do it in a way that the functions can be called natively as part of normal SQL expressions. So what about vice versa?

[get suggestions]

Topics

Interfacing with applications
- Database APIs (DBAPIS)
- Cursors

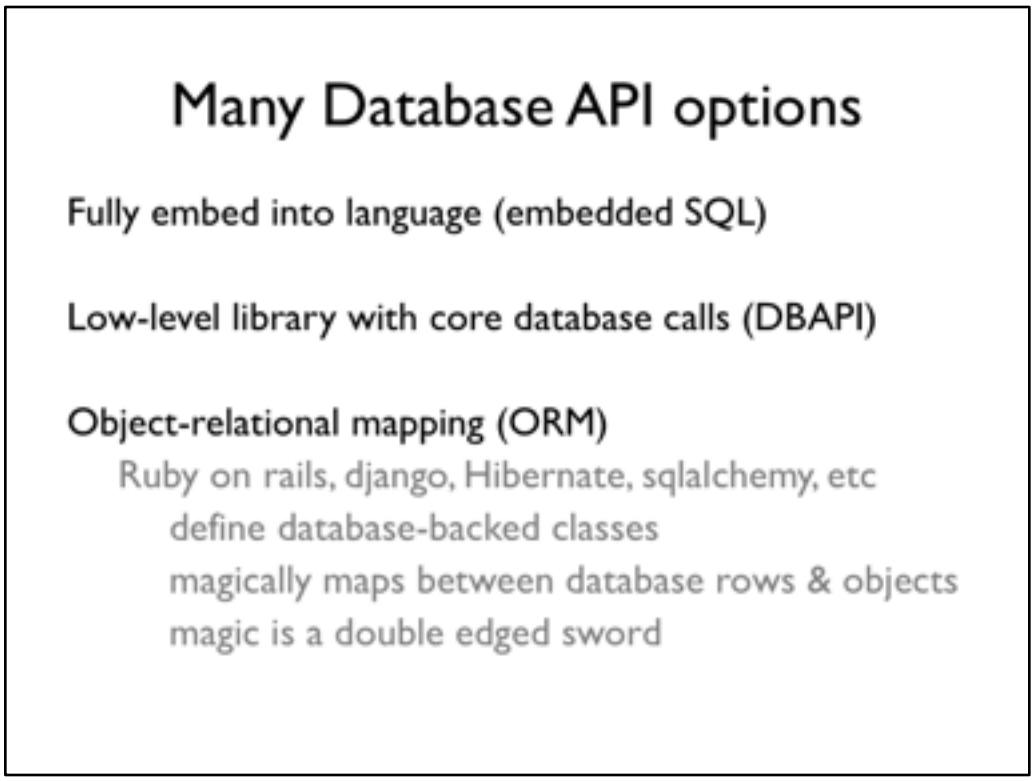
Some uncommon SQL-based analyses
- Graph analysis
- Math



for example, a web application is seemingly more than just a bunch of SQl queries. Need to figure out what text to write – maybe it's "hi mr wu" vs "Hi ms applebottom", and that's logic. Maybe we want to plot some of the data in a graph, that require a bunch of code once we get the data using a SQl query.

So there's a couple different solutions to this problem

- extend SQL to make it a real language – seems to give up a lot
- extend programming languages to understand SQL --
- create an API between programming languages and DBMSes



ORMs are intended to serve as an insulating layer away from the DBMS. Instead of writing sql queries, you define special classes (java classes) whose fields actually translate into database attributes, and you can create say a Users object, fill in the fields, and call a save method and it knows to create a new record in the database, or update an existing record. similarly, instead of writing SQL statements, you can make method calls such as filter on a Users object that represents the table, and it returns a list of User objects.

Lots of magic, it makes things convenient, and safe, but hard to debug if things don't work quite correctly

aversion to directly constructing and running SQL strings in the application, as we will see, because it's hard to get right and safe.

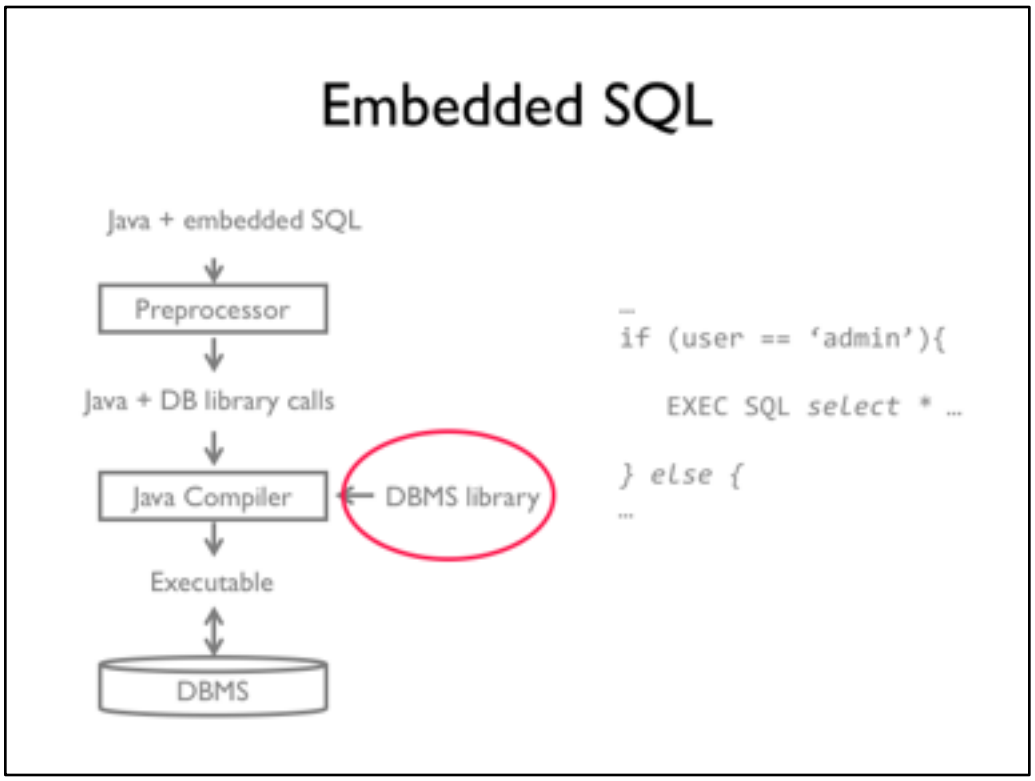
Embedded SQL

Extend host language (python) with SQL syntax

e.g., EXEC SQL *sql-query*

goes through a preprocessor

Compiled into program that interacts with DBMS directly



Why do you think embedded SQL approaches have not taken off?

* Hard enough to develop programming languages, writing and testing compliers, but also to maintaining compatibility with SQL? while SQL evolves?
* Need to re-implement a preprocessor for each language under the sun? hacking a compiler is painful.
* who's going to pay for this? it's a lot of hard work, and maintenance is forever
*

Ultimately you don't want to have a customized runtime system (e.g., JVM) as well, so embedded sql solutions typically boil down into a translation into java + a bunch of database library calls along with a library that gets linked into the executable.

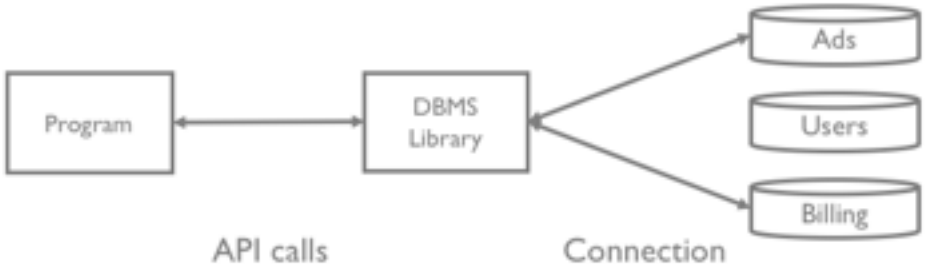
But scripting languages don't generate executables anyways, and updating the library requires re-deploying new executables which is hard.

It seems like the common denominator is the DBMS library.

Embedded SQL: Simply not good software engineering

What does a library need to do?

Single interface to possibly multiple DBMS engines
Connect to a database
Manage transactions (later)
Map objects between host language and DBMS
Manage query results

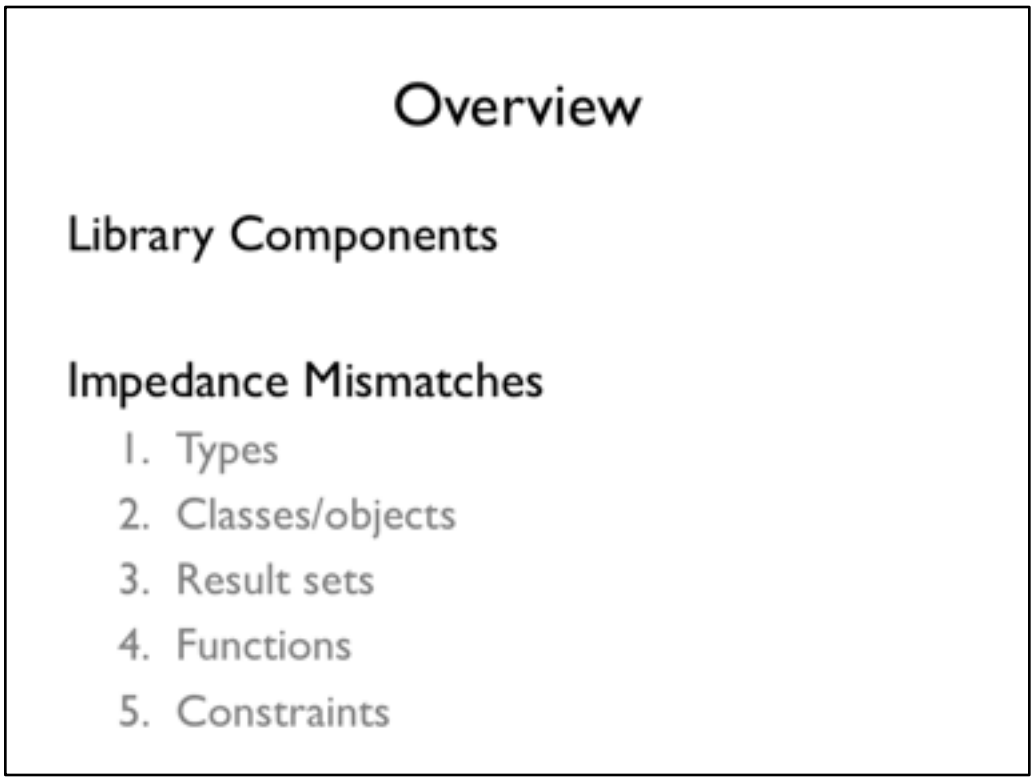


1) distinguish Databases, so need a naming scheme that describes where the database is, how to log in, etc etc JUST LIKE A WEB APP

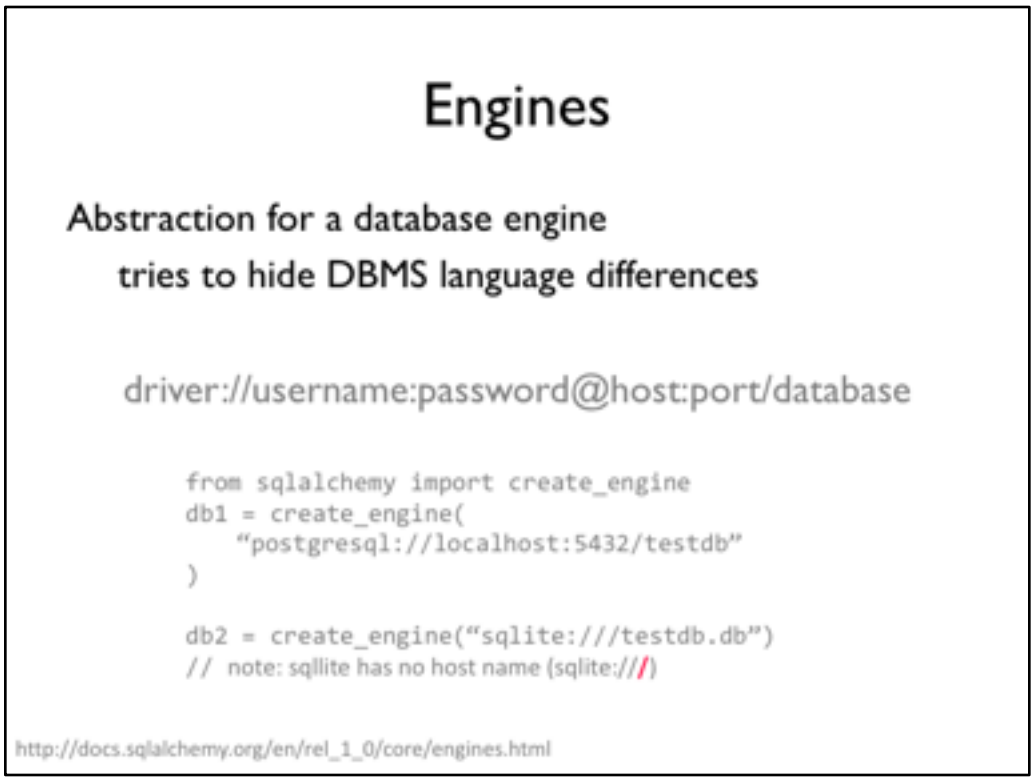
2) connect to db so it can send and receive messages, this is the same way your web browser needs to create a tcp connection with a webserver before it can issue requests to download HTML and javascript files to show you your favorite webpage

3) types may not be exactly the same, may need to do special processing so a program object can be translated into say a JSON object in the database system

basically a middleman, turns out to be a good idea if it helps manage complexity between two complex systems

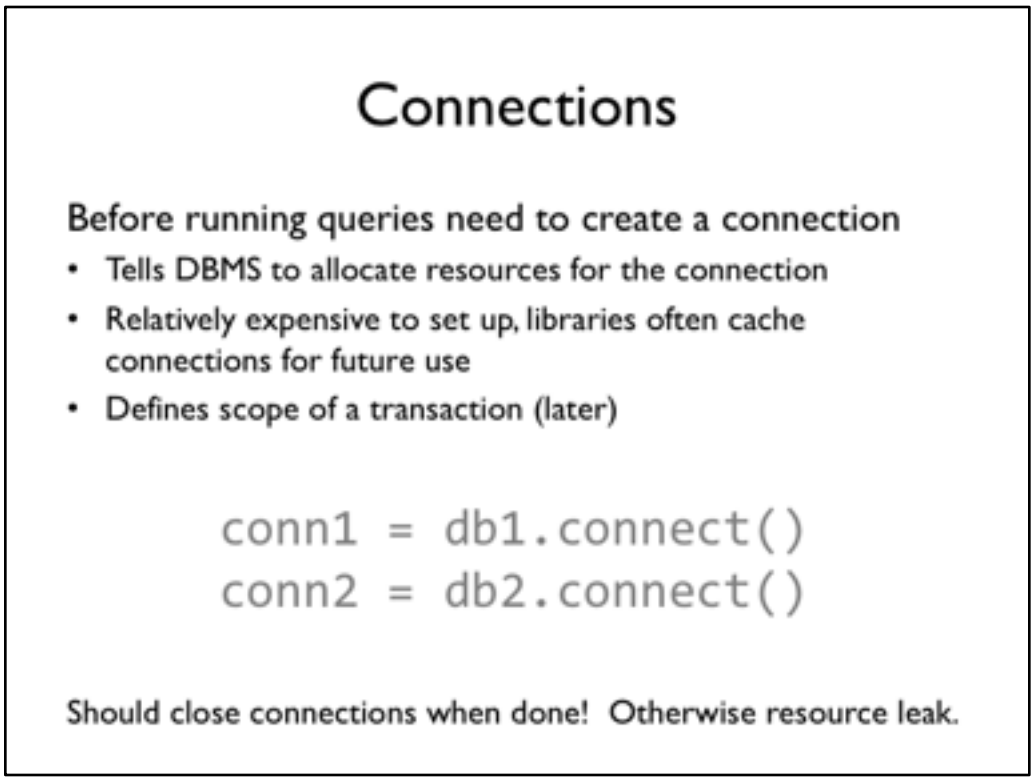


throughout the lecture we will be talking about different impedance mismatches – programming difficulties when dealing with SQL from an object oriented language
In practice, relational data bases had their own naming systems, their own data type systems, and their own conventions for returning data as a result of a query.
Whatever programming language was used alongside a relational data base also had its own version of all of these facilities
Hence, to bind an application to the data base required a conversion from "programming language speak" to "data base speak" and back. This was like "gluing an apple onto a pancake"



Engines are important because:

1) abstract representation of a database engine (postgresq, sqlite, db2, etc)
2) know what the location and credentials to access database engine are
3) may manage a connection pool (later)



Continuing from the previous slide, db1 and db2 are engines we have defined, so we run connect() to actually create the connection.
Can take upwards of 0.5 seconds for a new connection – 1.5 billion clock cycles, or read 3.2 gigs of data on a typcial DDR3 SDRAM
Also, connection can't be closed unless by the client, so can take up lots of database server resources.
Some databases limit for license reasons.

transactions are on a connection basis, so you connect to a database, start a transaction, every query on the connection is part of the xact, then you close.
you could imagine other wonkey ways where you name a transaction, and anyone could add a query to that named transaction, etc etc, but it gets messy.

close connection otherwise if your website popular, will run out of resources

For eaxmple, psql is basically a program that creates a connection to the DBMS and executes your queries. That's pretty much it!

Query Execution

```
conn1.execute("update table test set a = 1")
conn1.execute("update table test set s = 'wu'")
```

the signature is pretty simple, execute, takes a query string, and a possible set of parameters

for an update query like this, we don't expect any results back, so it's probably ok, BUT

Query Execution

```
foo = conn1.execute("select * from big_table")
```

Challenges

- What is the return type of `execute()`?
- Type impedance
- How to pass data between DBMS and host language?
- Can we only pass data between DBMS and host language?

To decide what issues could pop up think about what happens or should happen when we run this function.

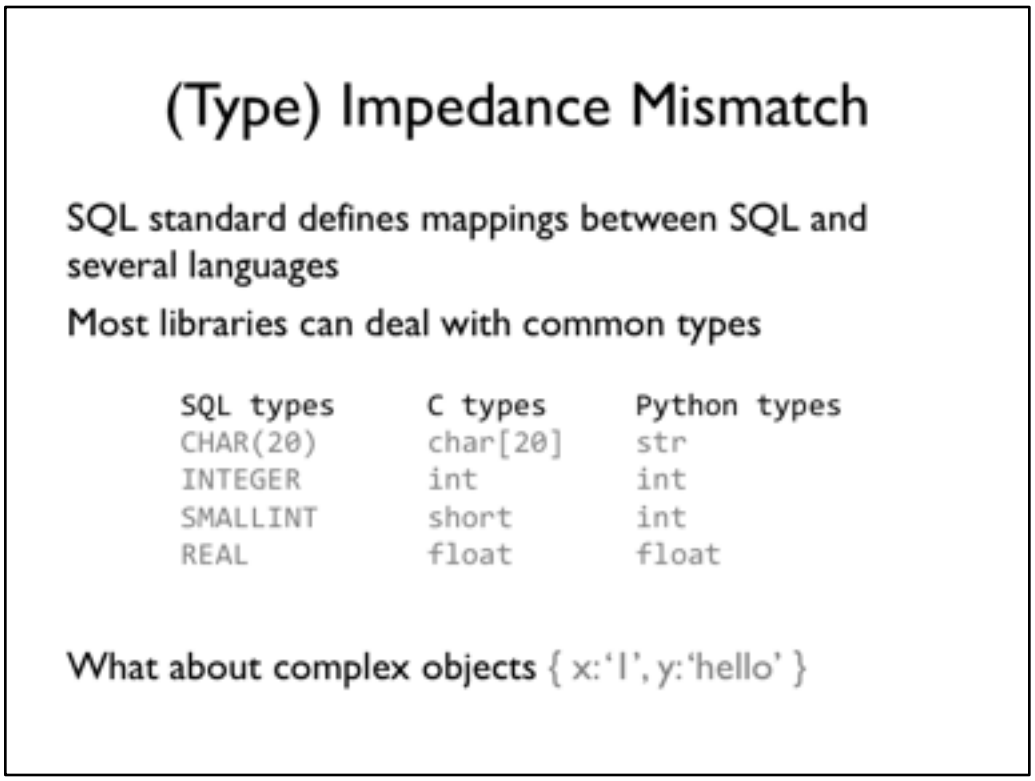
Send text over, run it, return results by encoding it and sending over the connection, and decoding in the library. Stop. Many questions now

What if it's a big result?

What if a value is a complex type like datetime? Which class should it be in Python? Or Java?

Is the return type a list? A set? What type of implementation? A hash set? Linked list?

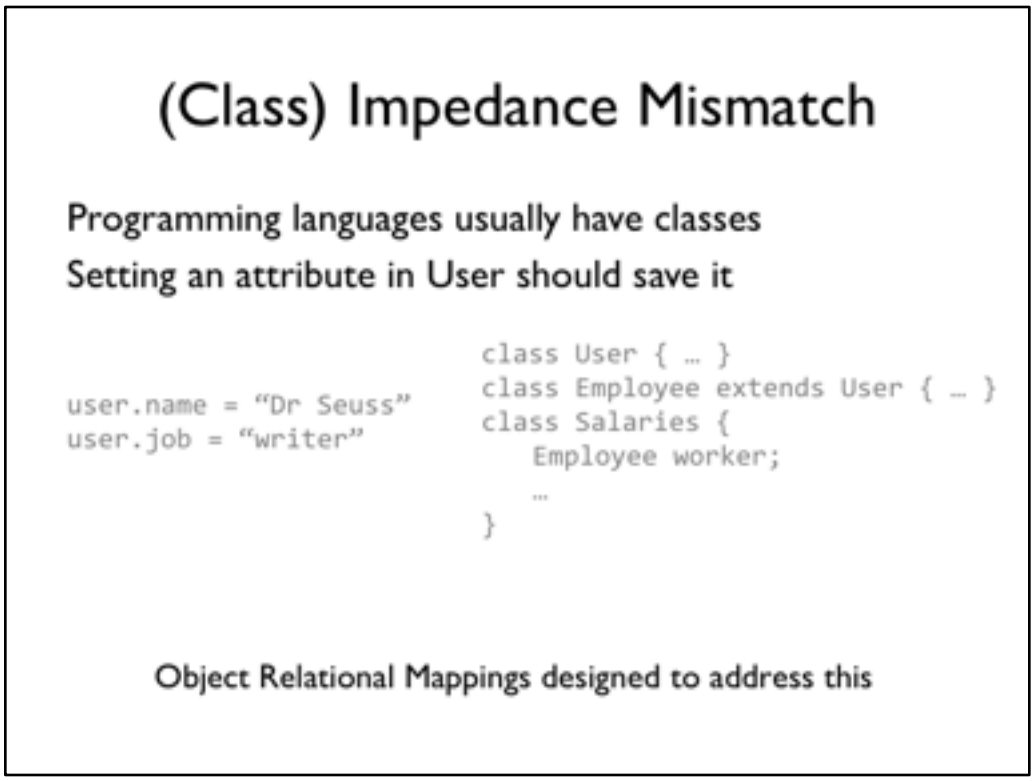
What about errors?



most libarries let you define your own type mappings
what about JSON objects?

- JSON is a data format that is encoded as a human readable string
- almost all modern languages support reading and writing JSON
- traditionally, would need to store as a string – meaning can't query on X or Y
- modern databases support JSON natively
- doing it efficiently is still a hard problem

W. R. Cook and A. H. Ibrahim. Integrating programming languages and databases: What is the problem. In In ODBMS.ORG, Expert Article, 2005.



How could you do this? You could define your user object. And when you want to load its data from the database, write a SQL query, get the results, go through the attrs and load them into the user object's fields – and do this every single time. Would like a better way that doesn't result in such redundant code.
Manually go between in-memory variables in the program and on disk data in the DBMS
if I have a user object, and say change the name, it would be great for a library to automatically save it, rather than me writing lots of SQL strings.

There was a whole period where object database systems were the rage in academia – embedded SQL, BUT augmented the classes – C++ classes so they knew how to persist in a database, but you use them like normal classes
That didn't go so well

- same problem as embedded sql
- real companies use a smattering of programming languages
- they are not willing to give up performance

class inheritance, as we saw it may be possible, but requires a bit of book keeping. Can library worry about it?
Class instances (Employee) as fields
Maybe we would like worker to be a weak entity, or a total participation relationship, maybe the library can worry about ensuring these!

If this is all stored safely in a DBMS, how to map?

In additino, there are more basic mismatches
- class interfaces for accessing data vs views
- access: public/private in OO vs all is visible in db
- integrity: exceptions in PL vs constraints
- free form collections of objects/structs vs global, consistent named relations.
- set oriented operation generall not how OO languages operate

Query Execution

How to pass values into a query?

```
Users(id int serial, name text)

name = "eugene"

conn1.execute("""
   INSERT INTO users(name)
   VALUES(<what to put here??>)""")
```

I have a variable name and I want to add it to my Users table.
What do I do?

Query Execution

How to pass values into a query?

```
Users(id int serial, name text)

name = "eugene"

conn1.execute ("""
   INSERT INTO users(name)
   VALUES('{name}')""".format(name=name))
```

Why is this a *really* bad idea?

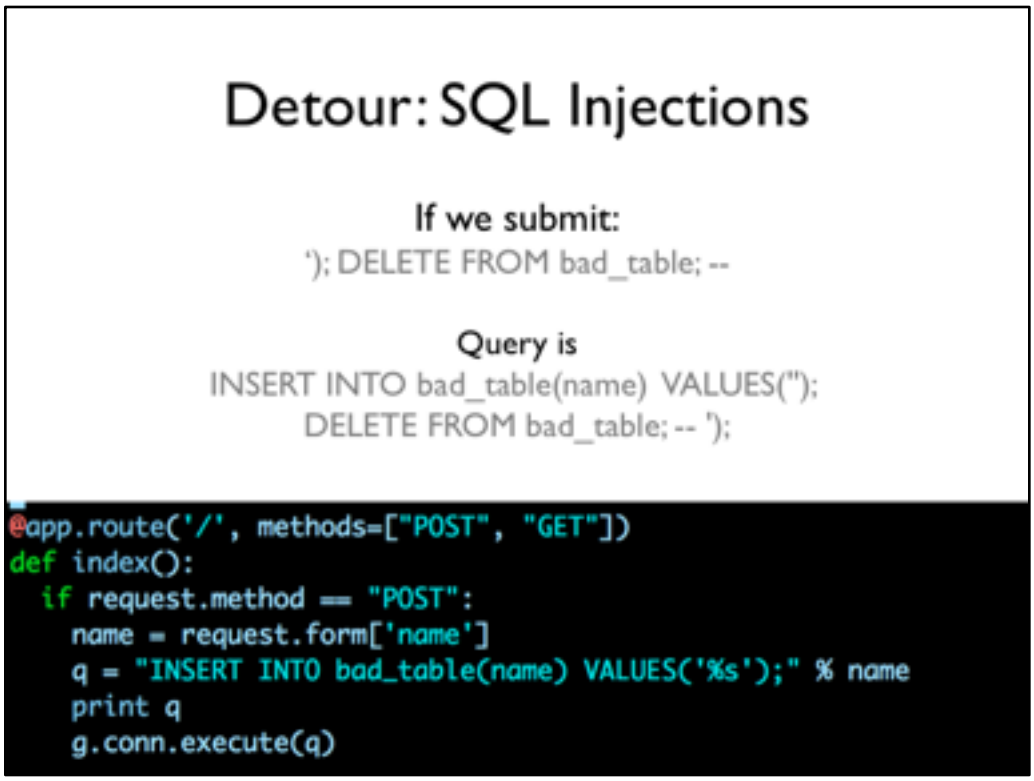
Explain what "{name}".format(name=name) is

Detour: SQL Injections

http://w4111db1.cloudapp.net:8888

code on github:
syllabus/src/injection/


```
@app.route('/', methods=["POST", "GET"])
def index():
  if request.method == "POST":
    name = request.form['name']
    q = "INSERT INTO bad_table(name) VALUES('%s');" % name
    print q
    g.conn.execute(q)
```



huge numbero f possible attacks, including attacks where you don't even see the query results – based on timing!
see wikipedia

Problem here is that the variable q will be modified by external (unsafe) inputs and executed as code (SQL statements).
What should be data is interpreted as code!!!! BAD BAD BAD

Detour: SQL Injections

Safe implementation

Pass form values as arguments to the `execute()` function
Library sanitizes inputs automatically (and correctly!)

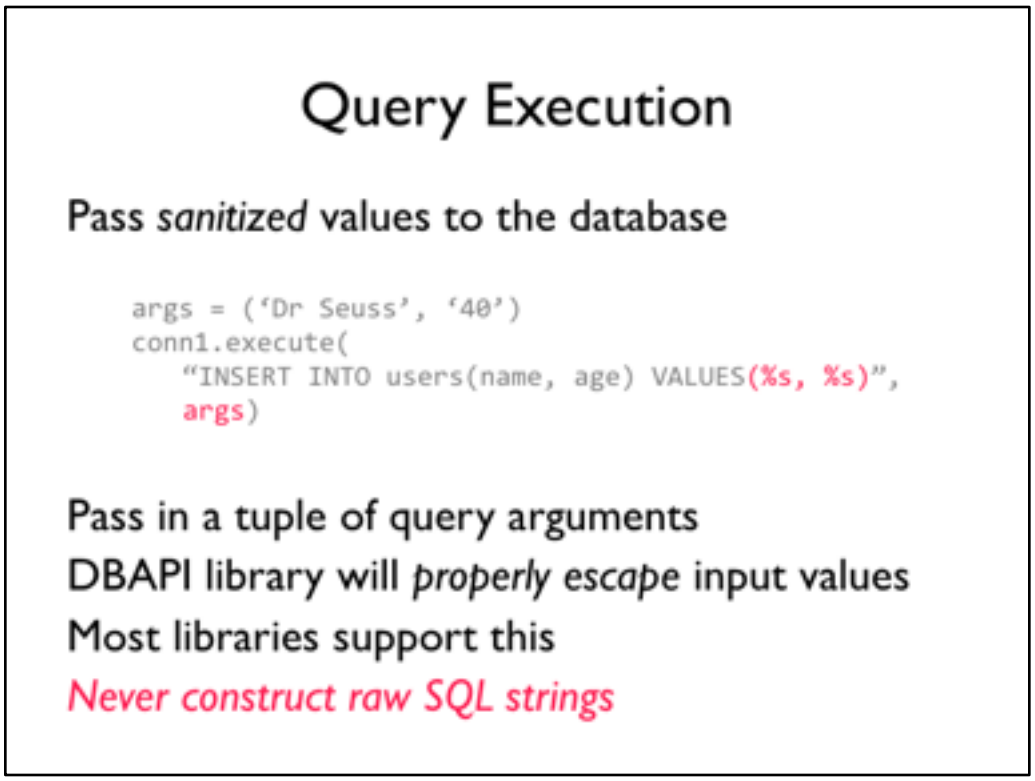
```
@app.route('/safe/', methods=["POST", "GET"])
def safe_index():
  if request.method == "POST":
    name = request.form['name']
    q = "INSERT INTO bad_table(name) VALUES(%s);"
    print q
    g.conn.execute(q, (name,))
```

Safe way . notice that q doesn't change

Detour: SQL Injections

Project: You'll need to protect against simple SQL injections

Why is the "–" important at the end of this? Because if the subsequent queries fail, the xact may be rolled back – don't want that!!



There are NUMEROUS other ways you can collect data about a database through a form interface in order to gain access or learn about information that you should not have access to.
Wikipedia has a very entertaining article about sql injection.

sO the lesson is to always pass sanitized values to the database, and the way to do it is to use the DBAPI library to construct the query strings.

(results) Impedance Mismatch

SQL relations and results are sets of records

What is the type of `table`?

```
table = execute("SELECT * FROM big_table")
```

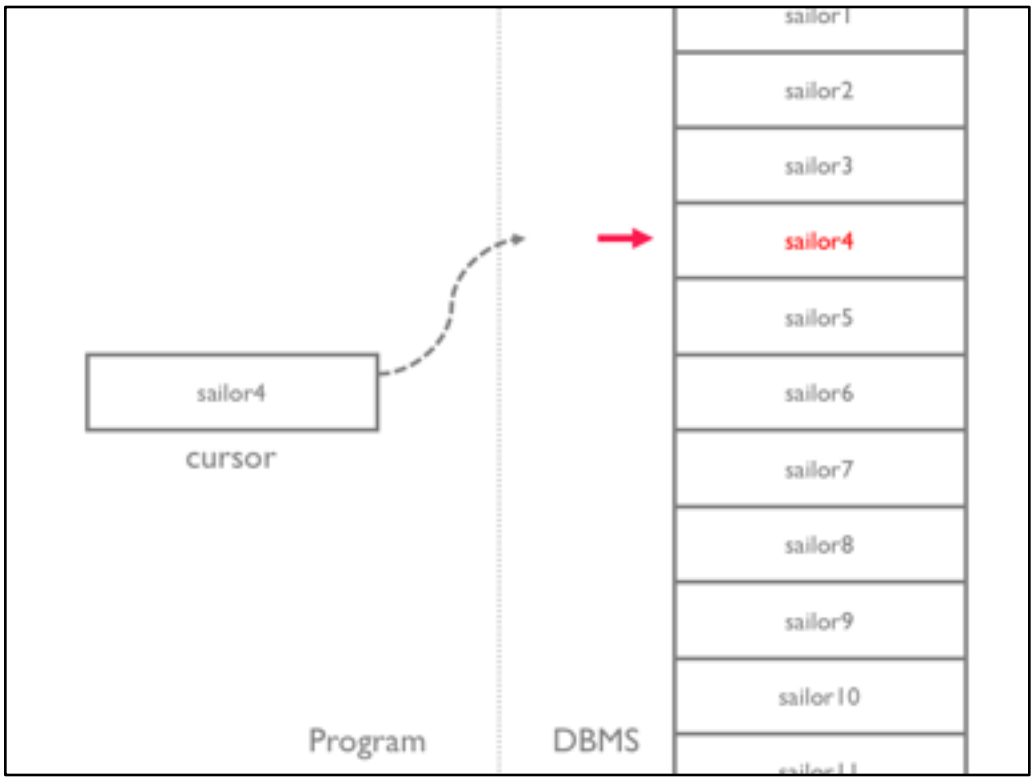
Cursor over the Result Set

- similar to an iterator interface
- Note: relations are unordered!
  - Cursors have no ordering guarantees
  - Use ORDER BY to ensure an ordering

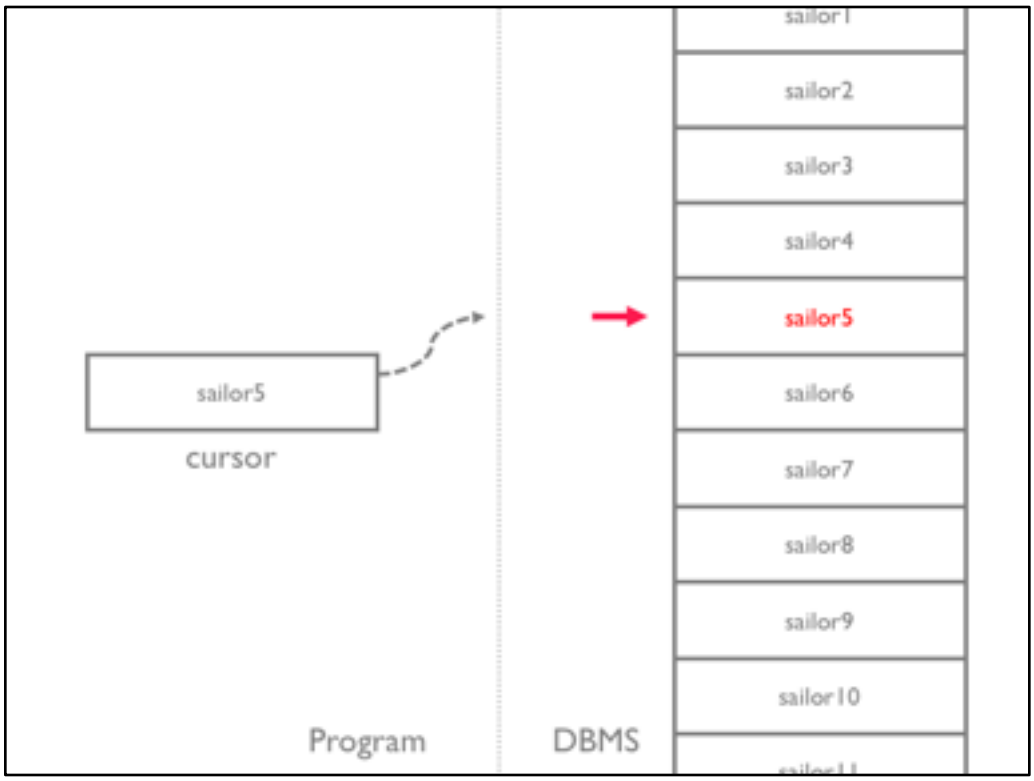
We discussed this earlier,
what if you just ran select * on a huge table? what should happen? should it return a list? a table object? what is the type?

Should it pass all of the results to the program all at once? Could that cause the progrma to run out of memory?
Returning something the size of the whole database is a bad idea, so instead we return cursor

Cursor represents a pointer to a record in the result set



database says ok, you want a cursor, so I'll keep these results or something that can recompute these results around, as well as a pointer to the current record you're looking at.
The cursor on the program side has some way to reference the red arrow, and acess that data


sailor1
sailor2
sailor3
sailor4
sailor5
sailor6
sailor7
sailor8
sailor9
sailor10
sailor5
cursor
Program
DBMS

(results) Impedance Mismatch

Cursor similar to an iterator (next() calls)

```
cursor = execute("SELECT * FROM bigtable")
```

Cursor attributes/methods (logical)

```
rowcount
keys()
previous()
next()
get(idx)
```

(results) Impedance Mismatch

Cursor similar to an iterator (next() calls)

```
cursor = execute("SELECT * FROM bigtable")
cursor.rowcount()  # 1000000
cursor.fetchone()  # (0, 'foo', …)
for row in cursor: # iterate over the rest
   print row
```

Actual Cursor methods vary depending on implementation

In sqlalchemy, the library only lets you move forward in the iterator, and its methods are fetchone, to read the next row and return it to you, or fetchall() to get everything Since it's an iterator, can loop through the cursor just like any other iterator

(functions) Impedance Mismatch

What about functions?

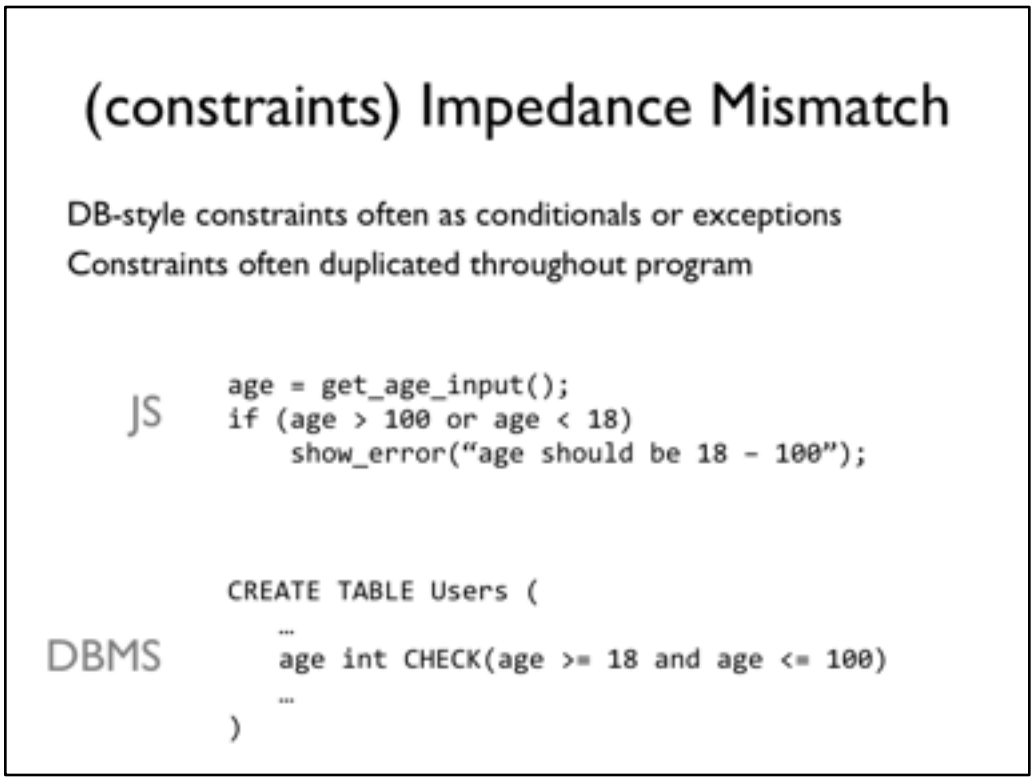
```
def add_one(val):
   return val + 1

conn1.execute("SELECT add_one(1)")
```

Would need to embed a language runtime into DBMS

- Many DBMSes support runtimes e.g., python
- Can register User Defined Functions (UDFs)

We've talked about using SQL functionality in the programming language but what about vice versa?
Can we run a programming language in the DBMS?

we've seen how user defined functions allow us to run arbitrary functions inside a database
however, it's not seamless



DB have constraints, but in OO it's often as IF statements or exception handling. Even so, with database constraints, often end up duplicating this in the application code. Why?
Better user experience, requires know things about the data closer to the user in the application.
some ORM systems provide ways to specify these constraints in the class definitions, and install them in the database as well as check for then in the application, to reduce logic redundancy

In particular, it's considered a good practice (from an end-user productivity point of view) to design user interfaces such that the UI prevents illegal transactions (those which cause a database constraint to be violated) from being entered; to do so requires much of the logic present in the relational schemata to be duplicated in the code.

(constraints) Impedance Mismatch

Some ORMs try to have one place to define constraints

```
class Person(models.Model):
    first_name = models.CharField(max_length=30)
    last_name  = models.CharField(max_length=30, null=True)
```

```
CREATE TABLE myapp_person (
    "id" serial NOT NULL PRIMARY KEY,
    "first_name" varchar(30) NOT NULL,
    "last_name" varchar(30)
);
```

so knowing this in the class definition, it could potentially create user interface widgets that can check themselves on the client

Some Useful Names

DBMS vendors provide libraries for most libraries

Two heavyweights in enterprise world

- **ODBC** Open DataBase Connectivity
  - Microsoft defined for Windows libraries

- **JDBC** Java DataBase Connectivity
  - Sun developed as set of Java interfaces
  - java.sql.*
  - javax.sql.* (recommended)

Modern Database APIs

DryadLinq, SparkSQL

DBMS executor in same language (dotNET, Spark) as app code

- what happens to language impedance?
- what happens to exception handling?
- what happens to host language functions?

```
val lines = spark.textFile("logfile.log")
val errors = lines.filter(_ startswith "Error")
val msgs = errors.map(_.split("\t")(2))

msgs.filter(_ contains "foo").count()
```

Part of the issue with all of this impedence mismatch is because the programming language between the systems, the data representation, the program scalability, the runtimes were are significantly different

But that is changing with modern implementations. DryadLinQ was a data processing system from Microsoft written in C# that was used in C#. So basically, you write C# code to construct what are essentially relational algebra-like operator plans that the same runtime knows how to execute.
Same thing with SparkSQL, it's a SQL implementation on top of the Spark system, both are written in Scala and run on the JVM. What happens to these impedence mismatches?

it is SO CONVENIENT!!! You may give up some raw performance, but with modern jitting and llvm compiler technology, even that is disappearing.

Some Tricky Queries

Lets write some tricky queries

- social graph analysis
  - how many friends?
  - clustering coefficient
- statistics
  - median

Social Network

```
-- A directed friend graph. Store each link once
CREATE TABLE Friends(
    fromID integer,
    toID integer,
    since date,
    PRIMARY KEY (fromID, toID),
    FOREIGN KEY (fromID) REFERENCES Users,
    FOREIGN KEY (toID) REFERENCES Users,
    CHECK (fromID < toID));

-- Return edges in both directions
CREATE VIEW BothFriends AS
    SELECT * FROM Friends
    UNION
    SELECT F.toID, F.fromID, F.since
    FROM Friends F;
```

Why have the CHECK constraint – undirected graph, ensures only one copy of each edge

as setup, we'll want to have a directed version of this database, how?

UNION ALL preserves duplicates
UNION removes dups

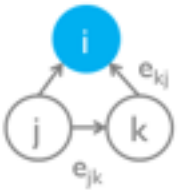
How many friends of friends do I have?

```
SELECT    count(distinct F3.toID)
FROM      BothFriends F1,
          BothFriends F2,
          BothFriends F3
WHERE     F1.toID = F2.fromID AND
          F2.toID = F3.fromID AND
          F1.fromID = <myid>;
```

friends of friends for each user?

```
SELECT    F1.fromID,  count(distinct F3.toID)
FROM      BothFriends F1,
          BothFriends F2,
          BothFriends F3
WHERE     F1.toID = F2.fromID AND
          F2.toID = F3.fromID
GROUP BY  F1.fromID;
```

Clustering Coefficient



$$C_i = 2|\{e_{jk}\}| / k_i(k_i-1)$$

friends that are actually friends

max possible edges between friends

K_i # neighbors of node i

e_{jk} edge between nodes j and k (j < k)

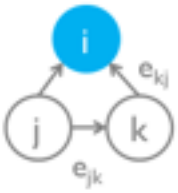
Cliqui-ness: % of your friends that are friends with each other

Clustering coefficient of graph = avg cliqui-ness of all nodes

if my friends are friends as well, weform a clique (everyone is one hop away).

so let's create tables for each of these

Clustering Coefficient

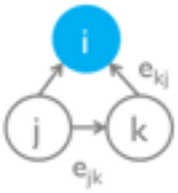


$$C_i = 2|\{e_{jk}\}| / k_i(k_i-1)$$

```
CREATE VIEW NEIGHBOR_COUNT AS
SELECT      fromID AS nodeID, count(*) AS friend_cnt
FROM        BothFriends
GROUP BY    nodeID;
```

neighbor count: how many friends does each user have, that'll tell us ehe max number
triangles:

Clustering Coefficient

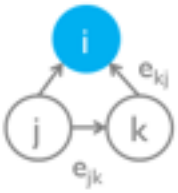


$$C_i = 2|\{e_{jk}\}| / k_i(k_i-1)$$

```
CREATE VIEW TRIANGLES AS
SELECT F1.toID as root, F1.fromID AS f1, F2.fromID AS f1
FROM   BothFriends F1, BothFriends F2, Friends F3
WHERE  F1.toID = F2.toID        /* j,k both point to i */   AND
       F1.fromID = F3.fromID    /* j two outgoing edges  */ AND
       F3.toID = F2.fromID      /* j and k are friends */ ;
```

ok we want triangles like in the picture above, and we can do that using the both friends table as a link
so a triangle has three links

Clustering Coefficient



$$C_i = 2|\{e_{jk}\}| / k_i(k_i-1)$$

```
CREATE VIEW NEIGHBOR_EDGE_COUNT AS
SELECT     root, COUNT(*) as cnt
FROM       TRIANGLES
GROUP BY   root;

CREATE VIEW CC_PER_NODE AS
SELECT NE.root,
       2.0*NE.cnt / (N.friend_cnt*(N.friend_cnt-1)) AS CC
FROM   NEIGHBOR_EDGE_COUNT NE,
       NEIGHBOR_COUNT N
WHERE  NE.root = N.nodeID;

SELECT AVG(cc) FROM CC_PER_NODE;
```

Median

Given n values in sorted order, value at idx n/2

if n is even, can take lower of middle 2

Robust statics compared to avg

- if want avg to equal 0, what fraction of values need to be corrupted?
- if want median to be 0, what fraction?

Breakdown point of a statistic

crucial if there are outliers

helps with over-fitting

Median

Given n values in sorted order, value at idx n/2

```
SELECT     T.c
FROM       T
ORDER BY   T.c
LIMIT      1
OFFSET     (SELECT COUNT(*)/2
            FROM T AS T2)
```

N/2 to the left
N/2 to the right

Median

Given n values in sorted order, value at idx n/2

```
SELECT   c AS median
FROM     T
WHERE
   (SELECT COUNT(*) FROM T AS T1
    WHERE T1.c < T.c)
   =
   (SELECT COUNT(*) FROM T AS T2
    WHERE T2.c > T.c);
```

N/2 to the left
N/2 to the right
off by one. what iff even number of elements

Faster Median

```
SELECT  x.c as median
FROM    T x, T y
GROUP BY x.c
HAVING
   SUM(CASE WHEN y.c <= x.c THEN 1 ELSE 0 END)
    >= (COUNT(*)+1)/2
   AND
   SUM(CASE WHEN y.c >= x.c THEN 1 ELSE 0 END)
    >= (COUNT(*)/2)+1 ;
```

Window Functions

How to run queries over ordered data

- O(n logn)
- Works with even # of items

```
CREATE VIEW twocounters AS
(SELECT   x,
          ROW_NUMBER() OVER (ORDER BY x ASC) AS RowAsc,
          ROW_NUMBER() OVER (ORDER BY x DESC) AS RowDesc
 FROM numbers );


SELECT AVG(x)
FROM twocounters
WHERE RowAsc IN (RowDesc, RowDesc - 1, RowDesc + 1);
```

Summary

DBAPIs
- Impedance mismatch
- Cursors
- SQL injection

Some hard queries
- More in the HW

Windows are optional material
SQL Injection: only what's in slides

What to Understand

Impedance mismatch

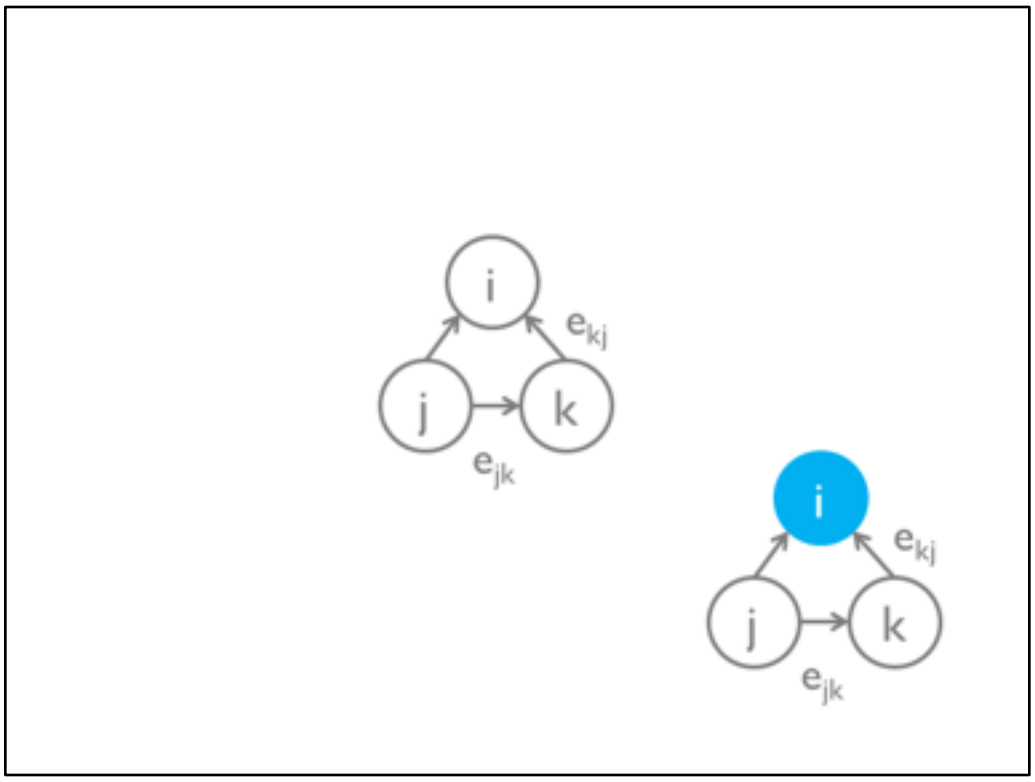
examples, and possible solutions

SQL injection and how to protect

The different uses of a DBAPI

Why Embedded SQL is no good

What good are cursors?


i
e_{kj}
j
k
e_{jk}
i
e_{kj}
j
k
e_{jk}