why databases?

downsides of querying data by hand:

- error-prone
- tedious
- slow

Types of databases

relational (SQL)

free:
- Postgresql - reddit, hipmunk
- MySQL - Facebook, everybody
- Sqlite -
- Oracle -

Google App Engine's Datastore

Amazon Dynamo

NoSQL

mongo

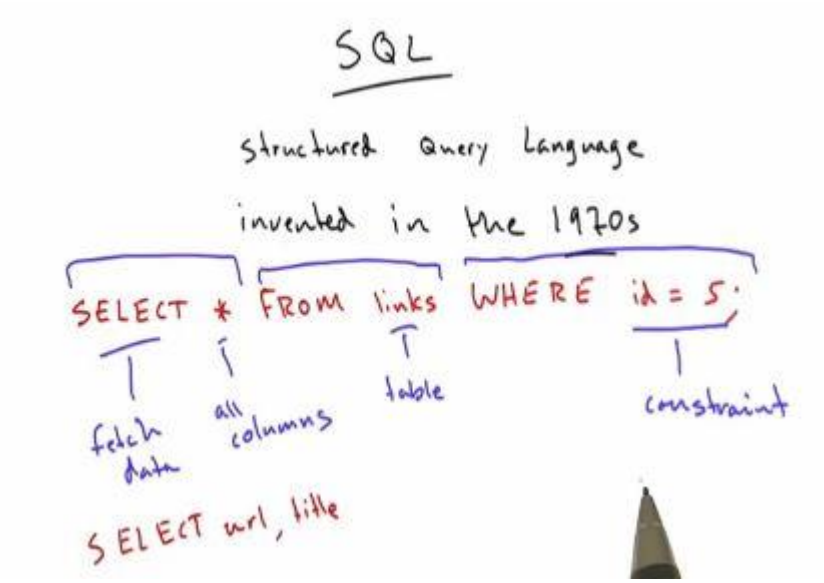
couch

```
SELECT * FROM links WHERE submitter_id = 5 OR
                          votes >= 23;
```

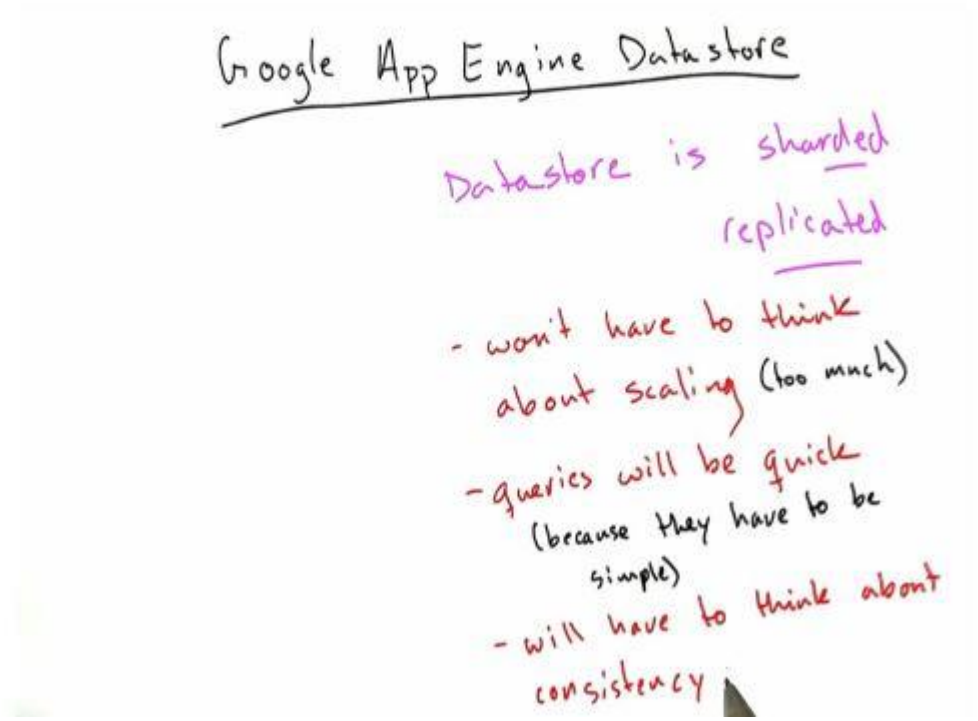
```
119 def query():
120     c = db.execute("select * from links where submitter_id = 62443 and votes > 1000")
121     link = Link(*c.fetchone())
122     return link.id
123
124 print query()
125
```

RUN

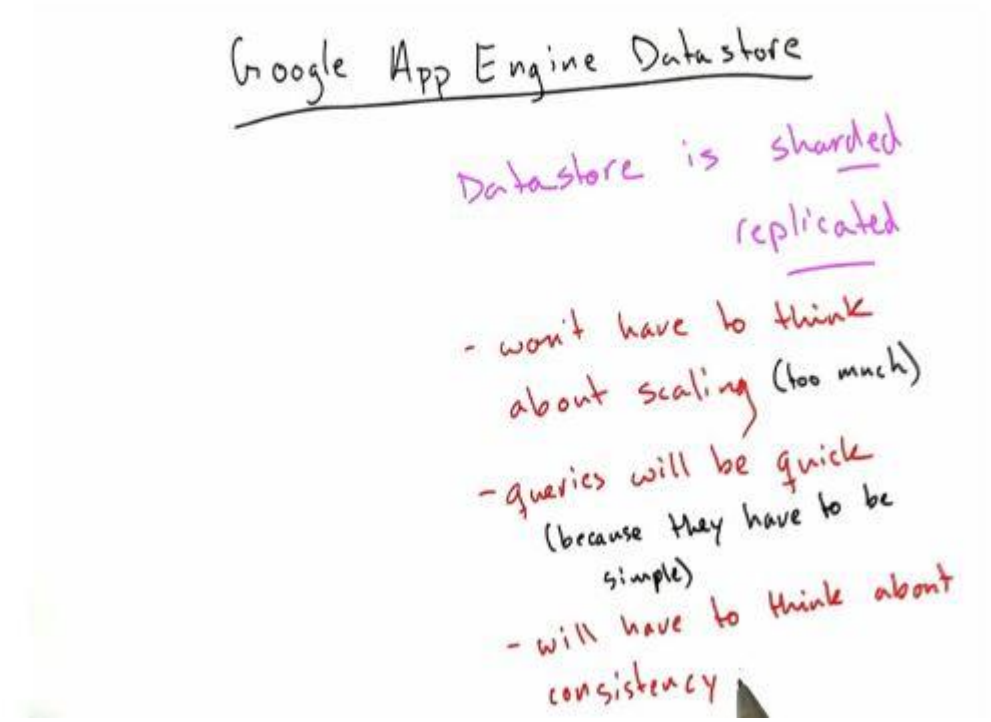
```
15
```



Automatic Sharding and Replication



Automatic Sharding and Replication

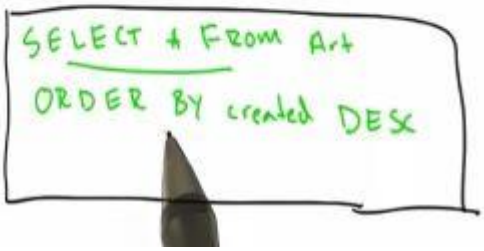



```
hipmunk=# explain analyze select name from hotels where id = 51492;
                                                 QUERY PLAN
-------------------------------------------------------------------------------------------------
 Seq Scan on hotels  (cost=0.00..52897.26 rows=1 width=23) (actual time=0.014..1
42.342 rows=1 loops=1)
   Filter: (id = 51492)
 Total runtime: 142.402 ms
(3 rows)
```

Quiz

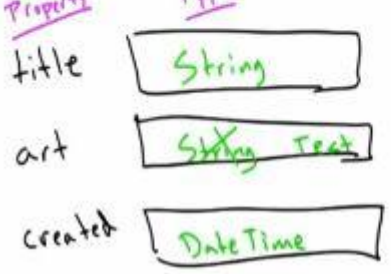
what is the query for fetching all of the Arts from the database sorted by creation time (most recent first).

created



Quiz

Integer Float String Date Time
Text DateTime
Email Link PostalAddress



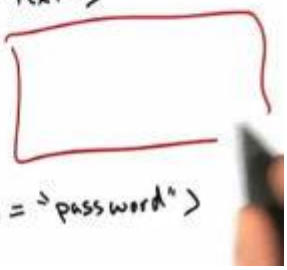
String - < 500 chars
indexed

Text - > 500 chars
not indexed

Quiz

which is the most appropriate form element for inputting ascii art?

- [ ] <input type="text">
- [x] <textarea>
- [ ] <pre>
- [ ] <input type="password">



Quiz

Do you understand everything there is to know about the App Engine Datastore?

- [ ] yes
- [x] no

Quiz

which is an appropriate technique for growing a database that won't fit on one machine)

- [ ] replicate the database
- [ ] get a bigger hard disk
- [x] Shard the database
- [ ] store less data

Quiz

which is an appropriate technique for increasing the read speed from a database?

- [ ] get a faster machine
- [x] replicate the database
- [ ] store less data
- [ ] press the turbo button

```
# QUIZ - implement the function add_new_link() that both adds a link to the
# "links" list and updates the link_index dictionary.
def add_new_link(link):
    links.append(link)
    link_index[link.id] = link

l = Link(50, 1, 1, 1, "title", "url")
add_new_link(l)

print links[-1]
print link_by_id(50)
```

RUN

```
Link(id=50, submitter_id=1, submitted_time=1, votes=1, title='title', url='url')
Link(id=50, submitter_id=1, submitted_time=1, votes=1, title='title', url='url')
```

```
# QUIZ - implement the function build_link_index() that creates a python dictionary
# the maps a link's ID to the link itself
def build_link_index():
    index = {}
    for l in links:
        index[l.id] = l
    return index

print build_link_index()


def link_by_id(link_id):
    for l in links:
        if l.id == link_id:
            return l
```

RUN

```
{0: Link(id=0, submitter_id=60398, submitted_time=1334014208.0, votes=109, title='C
overtakes Java as the No. 1 programming language in the TIOBE index.',
```

```
# QUIZ - implement the function link_by_id() that takes a link's ID and returns
# the link itself
def link_by_id(link_id):
    for l in links:
        if l.id == link_id:
            return l

print link_by_id(24)
```

RUN

```
Link(id=24, submitter_id=48626, submitted_time=1333934004.0, votes=17, title='An R
programmer looks at Julia', url='http://www.r-bloggers.com/an-r-programmer-looks-at-
julia/')
```

```
def query():
    for l in links:
        if l.id == 15:
            return l.votes

print query()
```

RUN

```
1244
```

```
SELECT * FROM links
       WHERE votes > 10
       ORDER BY votes DESC
```

(ascending by default)

ASC - ascending

DESC - descending

Joins

Link

ID	votes	user_id	date	title	url
5	206	22	1/12/07	blah	example.com

user

ID	name	password	date
22	Spez	hunter2	5/20/05

```
SELECT link.* FROM link, user WHERE link.user_id = user.id AND
       user.name = 'spez'
```

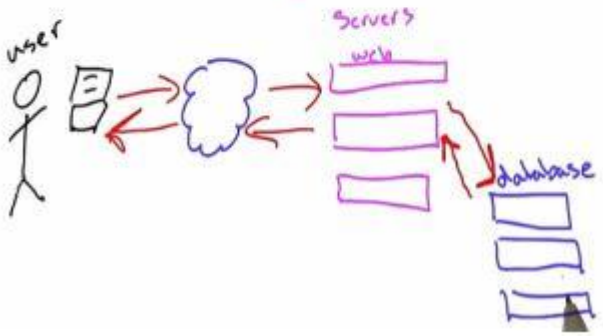
Databases

what is a database?

A program that stores and retrieves data

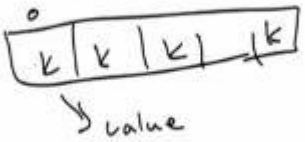
large amounts

structured



Indexes for sorting

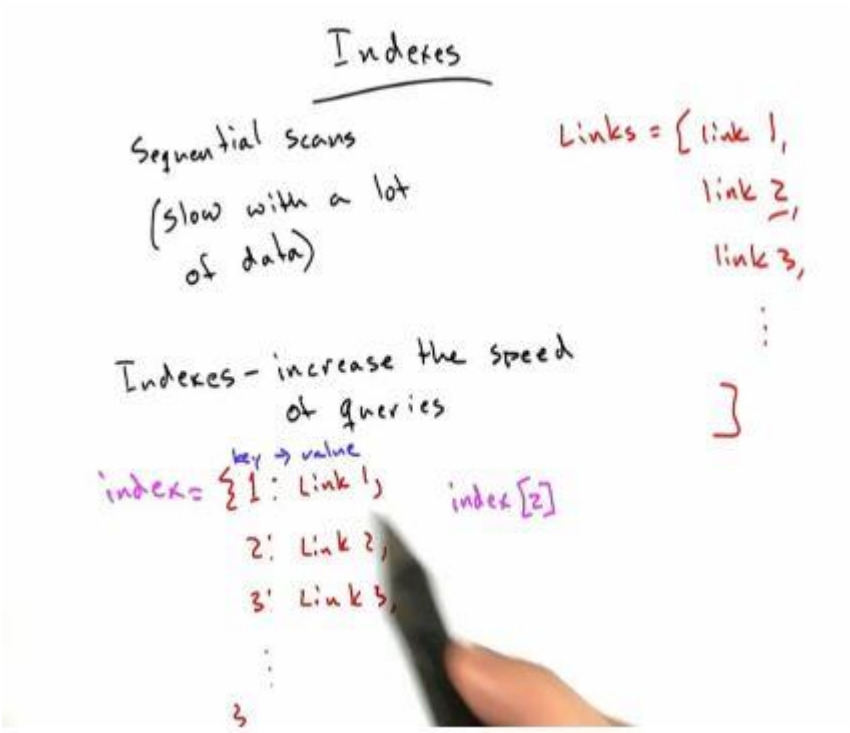
hashtable - not sorted, constant time



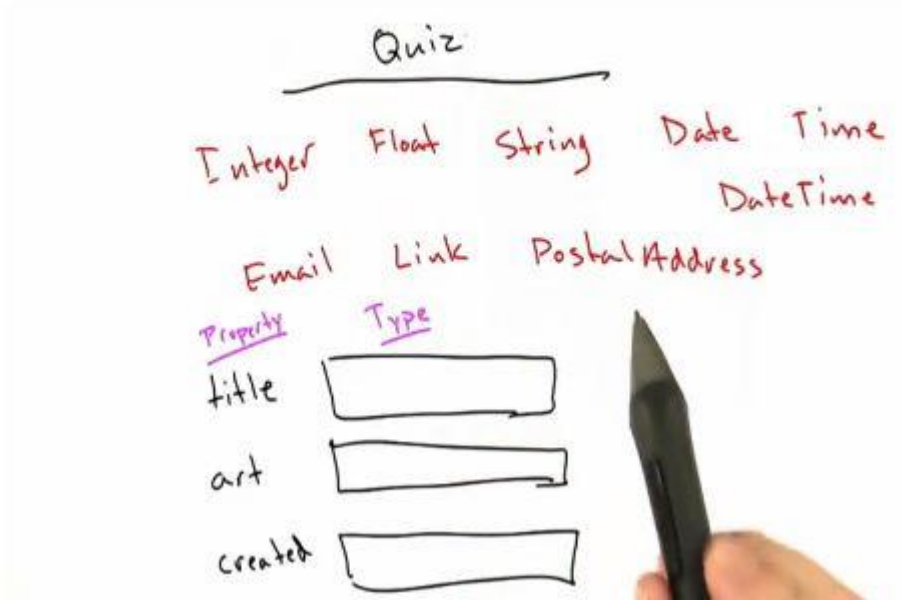
tree - sorted - lookups are slower

log n

```
hipmunk=# create index hotel_id on hotels(id);
CREATE INDEX
hipmunk=# explain analyze select name from hotels where id = 51492;
                                                    QUERY PLAN
------------------------------------------------------------------------------------------------------------
 Index Scan using hotel_id on hotels  (cost=0.00..8.30 rows=1 width=23) (actual
time=0.123..0.125 rows=1 loops=1)
   Index Cond: (id = 51492)
 Total runtime: 0.163 ms
(3 rows)
```



Datastore Types



Google App Engine Datastore

SQL → GQL
- all queries begin with SELECT *
- no joins

run arbitrary queries → all queries must be indexed

Google App Engine Datastore

tables → entities
- columns are not fixed
- all have an ID
- parents/ancestors

Reddit → Link

Reddit Hotness Algo

1. ↑ Zombie Dogs!
2. ↓ Ron Paul for president
3. ↑ Look at this cat
4. ⋮

SELECT * FROM links
ORDER BY score DESC

hot_idx (score)

Link

ID	ups	down	date	score	float
	10	1	—	25	

+1

amt = hot(ups, downs, date)

+ amt
+ 1
+ 10

Quiz

Replication lag is an example of the loss of which property?

- [ ] Atomicity
- [x] Consistency
- [ ] Isolation
- [ ] Durability

ACID

Atomicity - all parts of a transaction succeed or fail together

Consistency - the database will always be consistent

Isolation - no transaction can interfere with another's

Durability - once the transaction is comitted, it won't be lost

up down
Link
score ✓
user
karma ✓

Which statements are true?

- [x] indexes increase the speed of database reads.
- [ ] indexes ~~increase~~ decrease the speed of database inserts.

which database is the Best?

- [ ] ~~MySQL~~
- [ ] PostgreSQL
- [ ] Google App Engine Datastore
- [ ] Dynamo

what can a database refer to?

- [x] A program that stores and retrieves data
- [x] the machine running that program
- [x] a group of machines working together to store/retrieve data