# Wide Column Stores

Felix Bruckner

Mannheim, 15.06.2012

# Agenda

- Einführung
  - Motivation
  - Grundlagen NoSQL
- Grundlagen Wide Column Stores
  - Anwendungsfälle
  - Datenmodell
  - Technik
  - Wide Column Stores & Cloud Computing
- API & Queries
- Anwendungsbeispiel

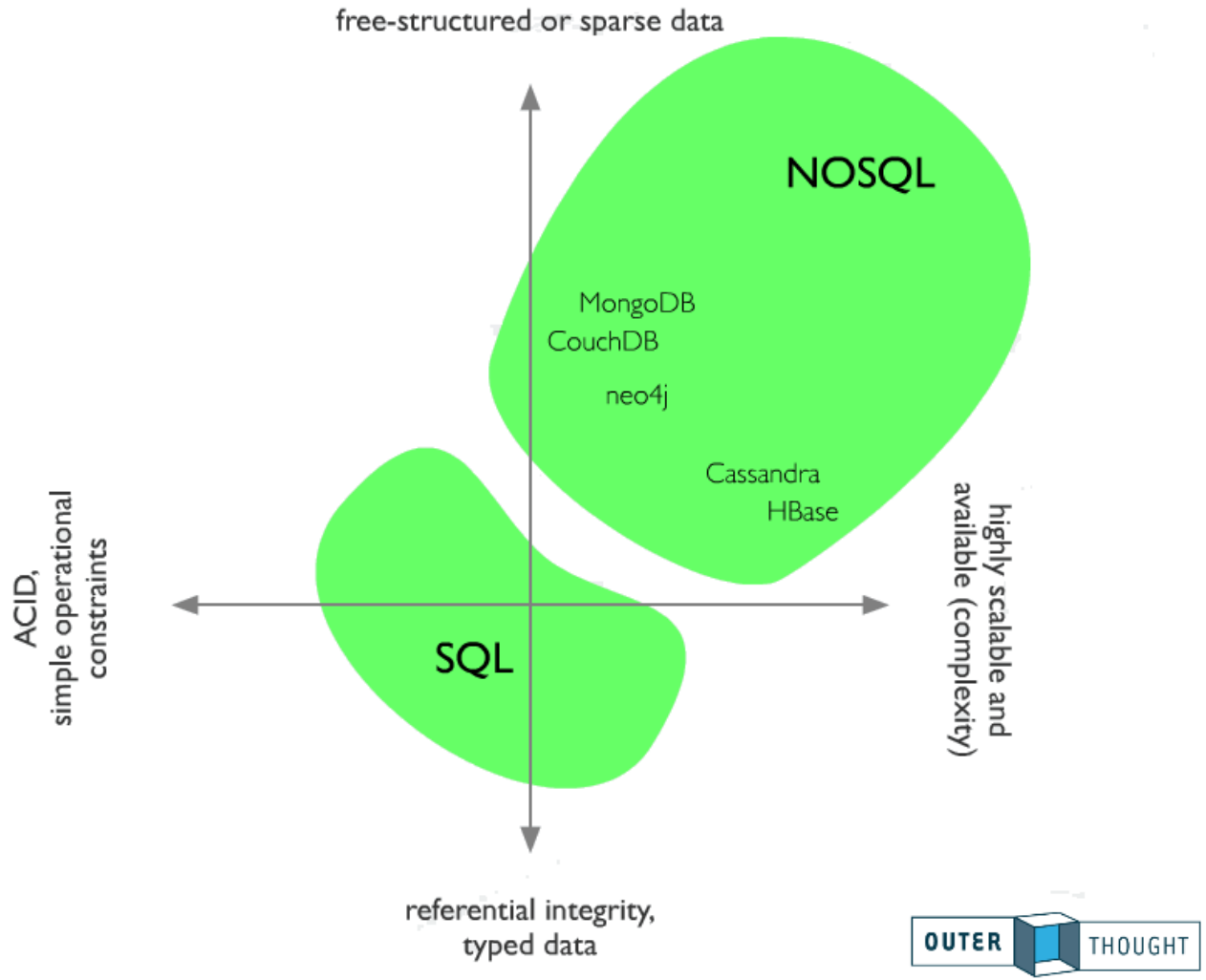
# Einleitung – Motivation

- Das Datenaufkommen in allen Bereichen steigt stetig
  - Web 2.0 (“User created content”) wird immer wichtiger
- Relationale Datenbanken skalieren schlecht
  - Vorallem in bei hoher Schreiblast (Disk I/O)

# Einleitung – Motivation

- Wie geht man mit den sich ändernden Rahmenbedingungen um?
  - Einsatz verteilter Datenbanken
  - Auflockerung der “relationalen Denkweise”
  - ACID-Konformität sekundär für einige Anwendungsgebiete:
    - Beispiele: Datei-Uploads von Benutzern, Messdaten oder Logdaten

# NoSQL vs SQL


free-structured or sparse data
NOSQL
MongoDB
CouchDB
neo4j
Cassandra
HBase
SQL
ACID,
simple operational
constraints
highly scalable and
available (complexity)
referential integrity,
typed data

# Definition

Wide Column Store

“a sparse, distributed multi-dimensional sorted map”

Google Bigtable Paper (Kap 2: Data Model), ACM 2006

# Geschichte & Anwendungsfälle

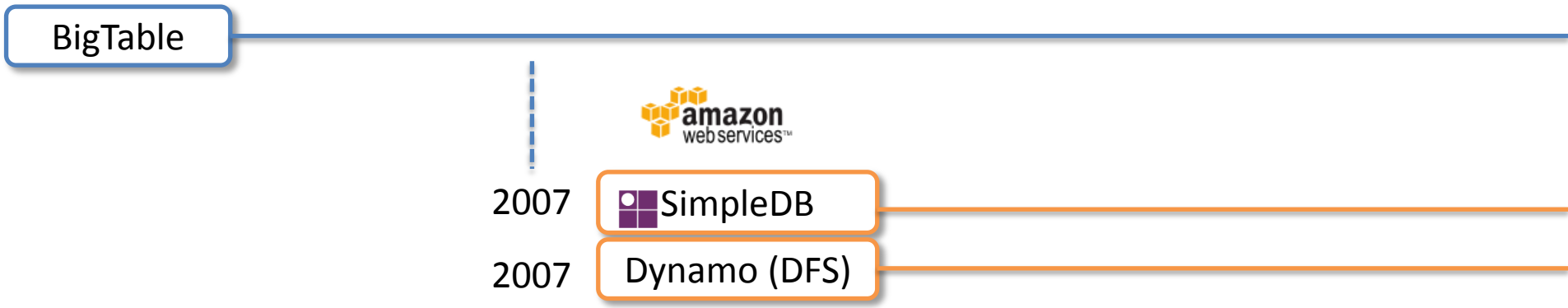
Google

2006 BigTable

BigTable:

- Entwickelt von Google
- Beschrieben 2006 in einem Paper
- Besteht aus zwei Teilen:
  - Verteiltes Dateisystem (GFS)
  - Verteilte Datenbank auf GFS
- Proprietär

# Geschichte & Anwendungsfälle



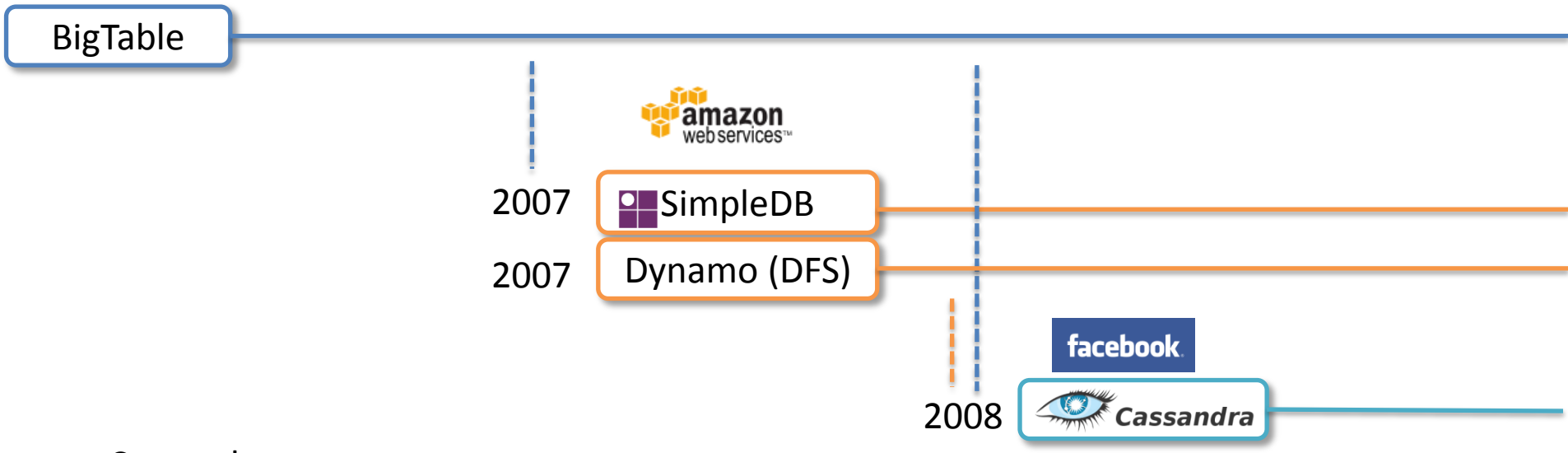

SimpleDB:

- Entwickelt von Amazon
- Verteilte Datenbank
- Proprietär

Dynamo:

- Verteiltes Dateisystem
- P2P-Prinzip für Ausfallsicherheit
- Hochskalierbar
- Backend für alle Amazon-Webseiten
- Seit 2012: DynamoDB verfügbar
  - DFS mit Datenbank on top
- Proprietär

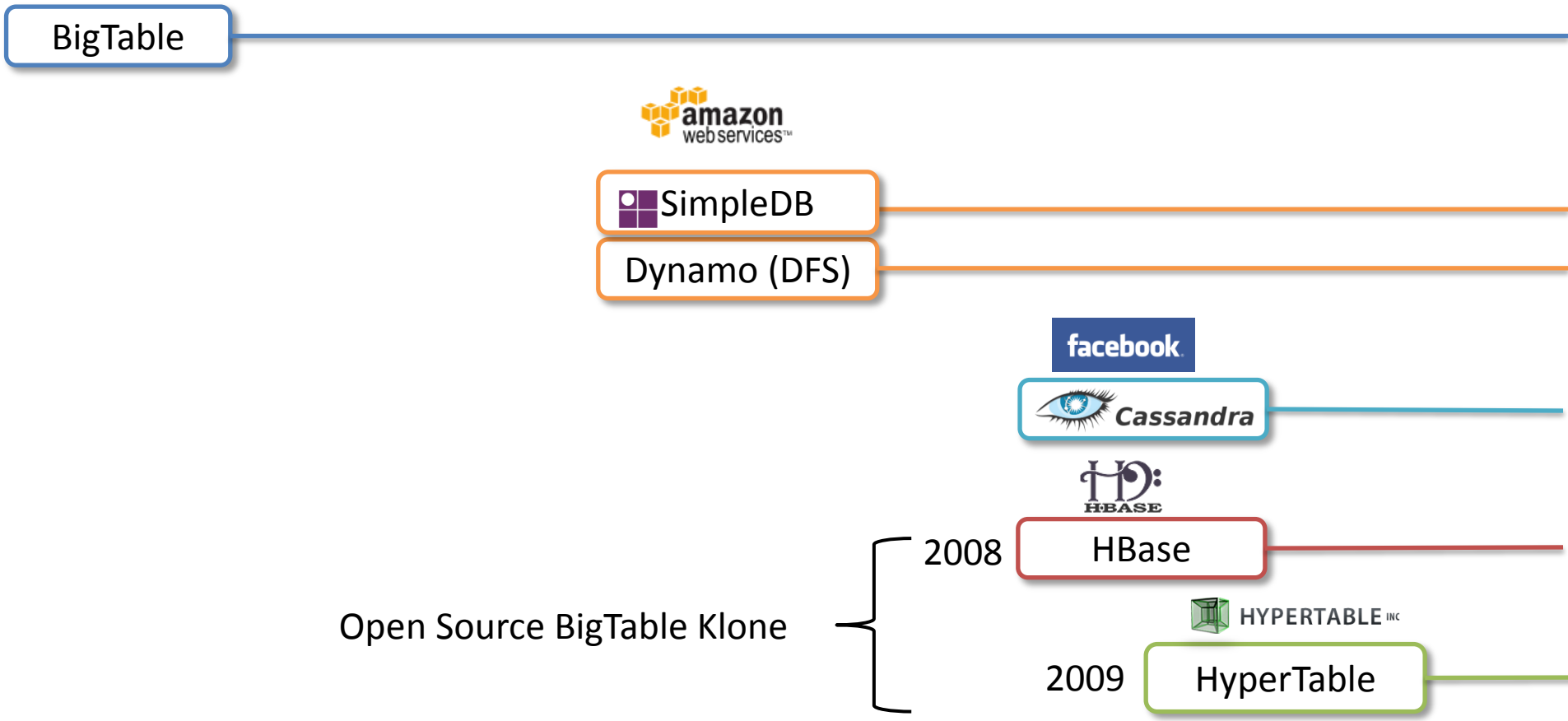
# Geschichte & Anwendungsfälle




Cassandra:

- Entwickelt von Facebook
- Open Source seit 2008
- Vereint BigTable und Amazon Dynamo
- Vorallem für hohe Schreiblast geeignet

# Geschichte & Anwendungsfälle




[1] http://en.wikipedia.org/wiki/BigTable

# Anwendungsfälle

- Speicherung sehr großer Datenmengen (> 1 Petabyte)
- Auswertung von Webstatistiken (Werbung)
- Echtzeit-Verarbeitung großer Datenmengen
- Hohe Schreiblast (> 1 Mio / Sekunde)
- Komplexe Abfragen über große Datenmengen
- Social Media Analysen

# Anforderungen

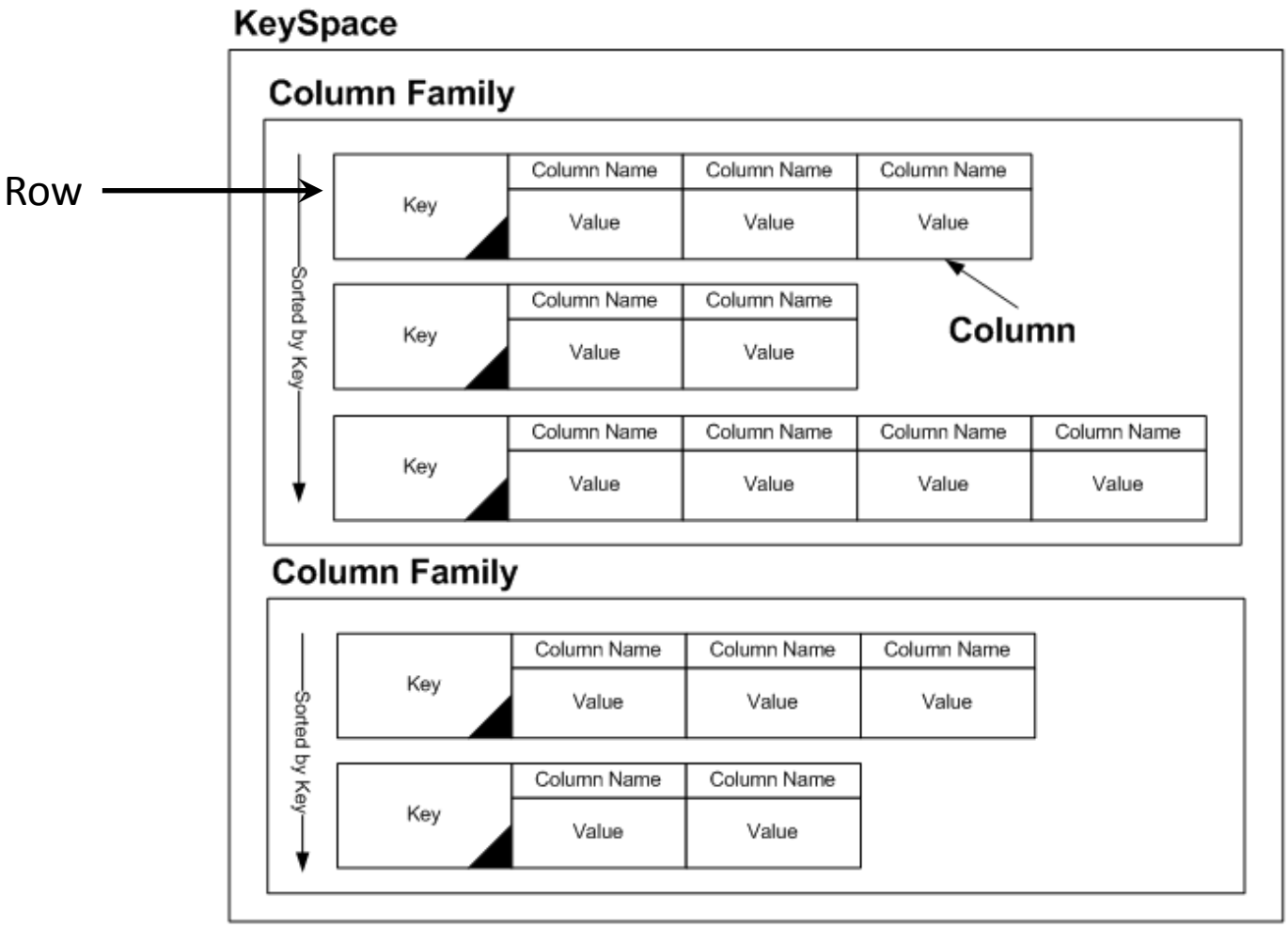
*“[...]even if disks are failing, network routes are flapping, or data centers are being destroyed by tornados. ”*

„[...]selbst wenn Festplatten versagen, Netzwerkverbindungen verrückt spielen oder Rechenzentren von Tornados zerstört werden.“

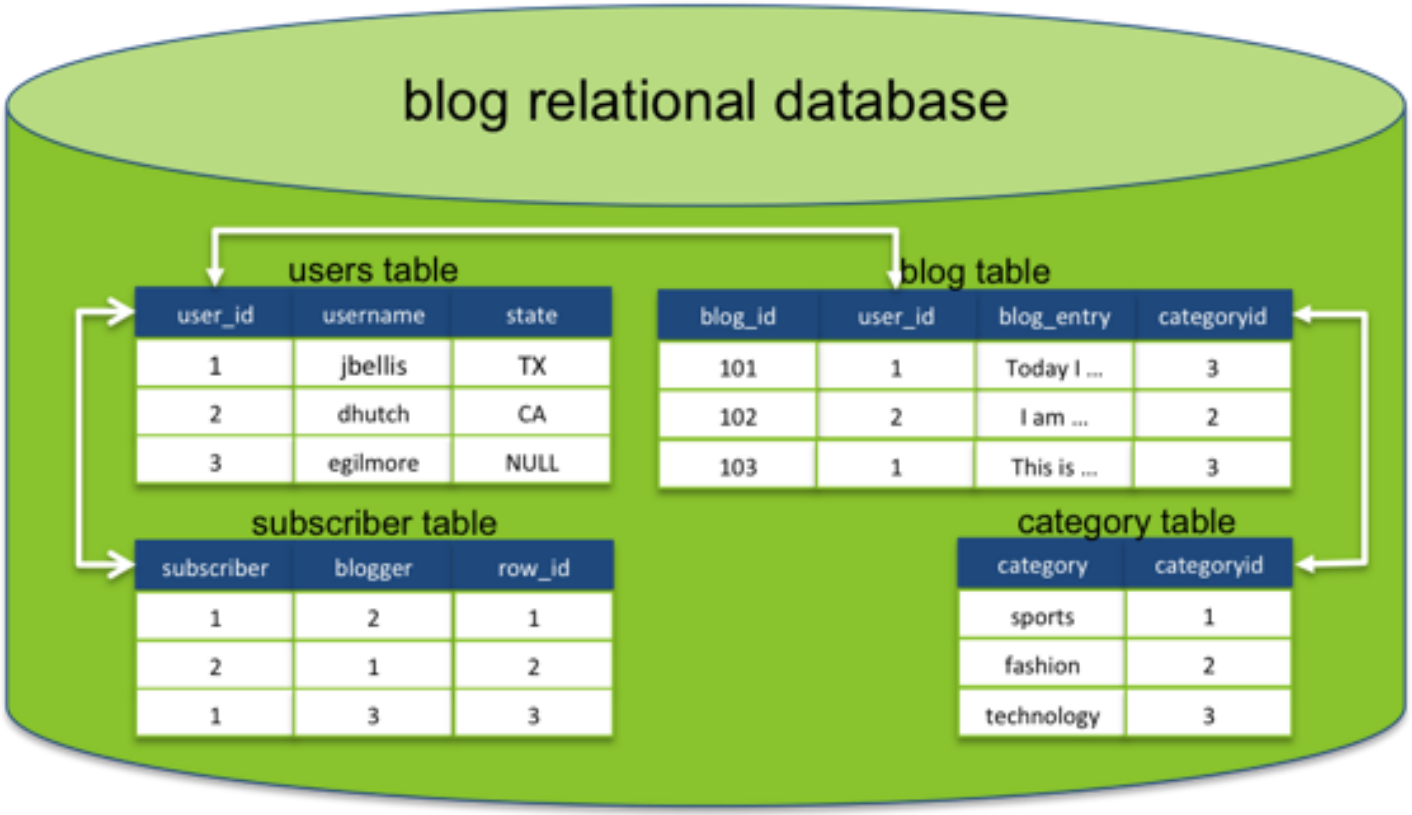
– WERNER VOGELS, AMAZON.COM: *Amazon's Dynamo*

- Skalierbarkeit
- Hochverfügbarkeit
- Ausfallsicherheit

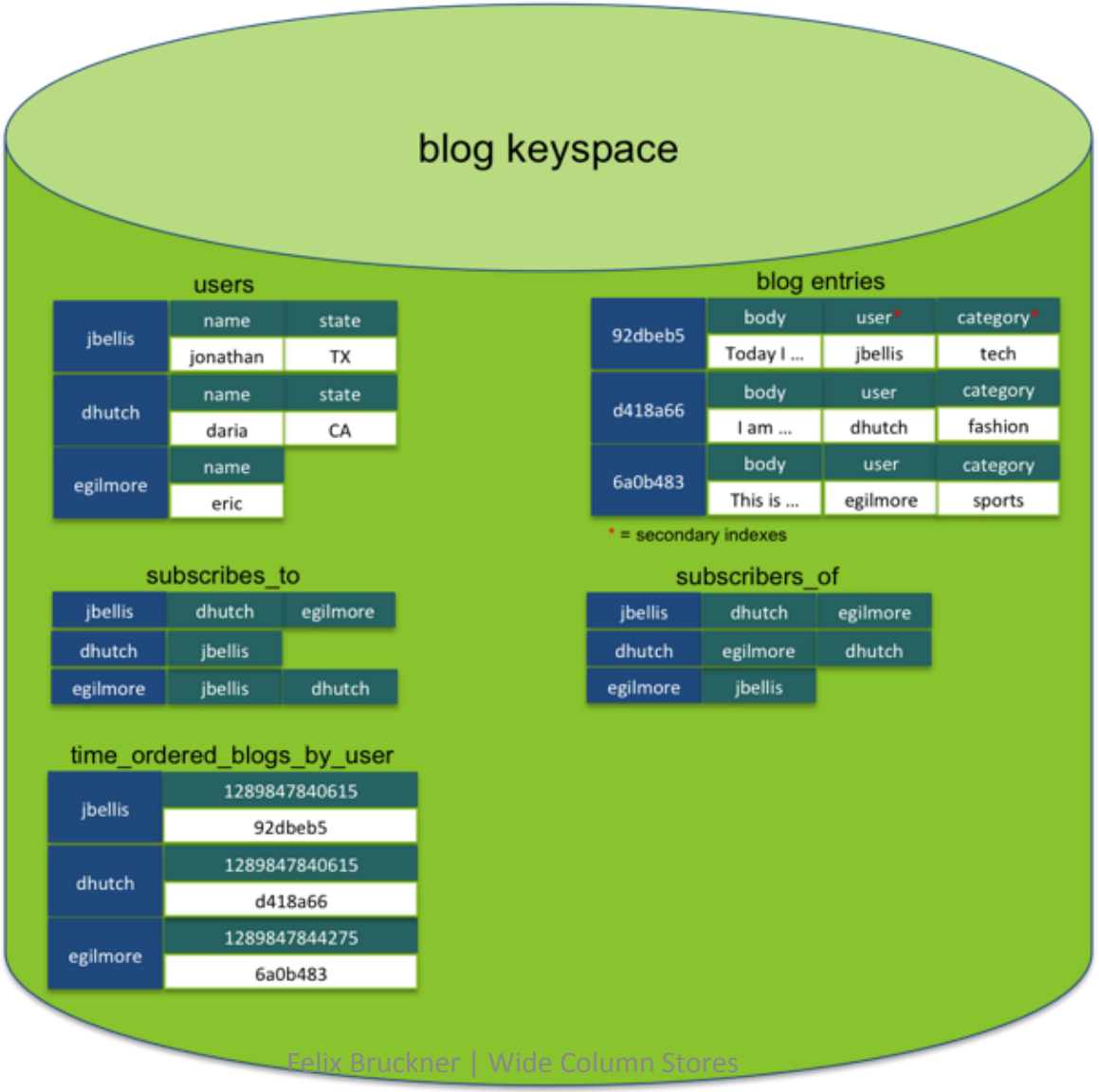
# Datenmodell



# Datemodell - Beispiel



# Datemodell - Beispiel

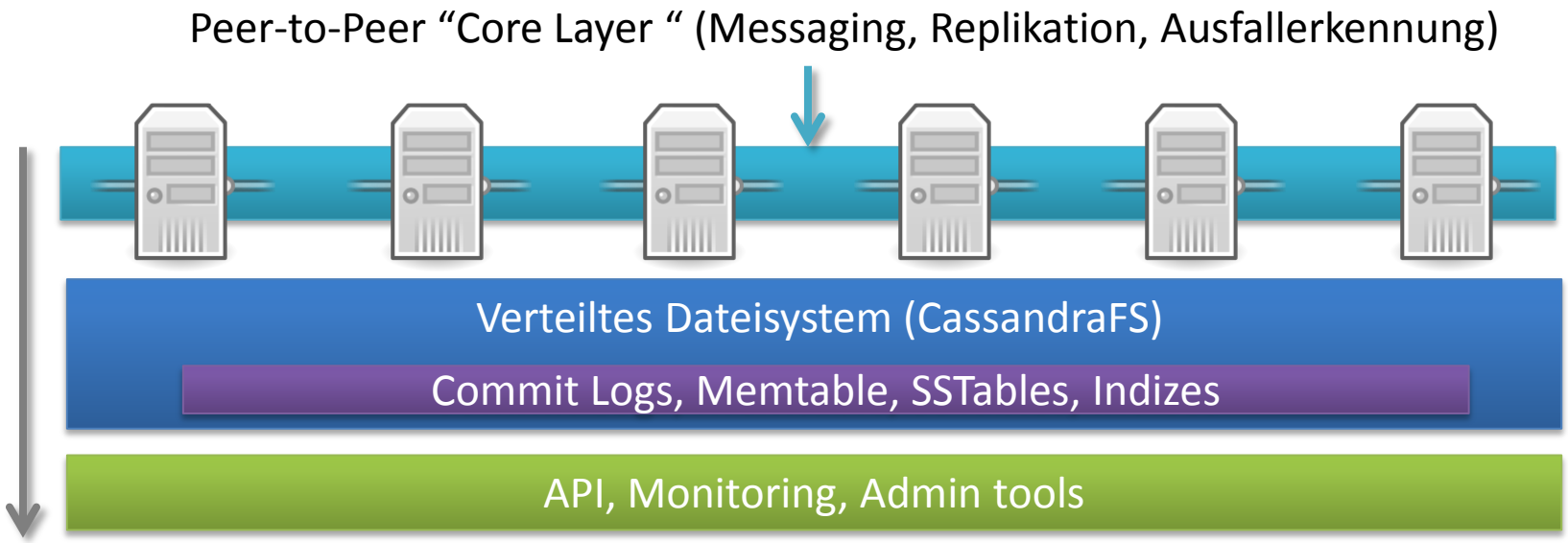


# Datenmodell

- Zeile (Row) mit Schlüssel
- Beliebige Anzahl Spalten pro Zeile
  - Spalte besteht aus: Key, Value, Timestamp
- Zeilen werden in “Column Families” zusammengefasst
- Column Families werden in Keyspaces zusammengefasst
- Denomarlisierung relationaler Daten
- Zeilen wachsen stetig in der Breite
  - Tendenz: Eher viele Spalten, dafür weniger Zeilen

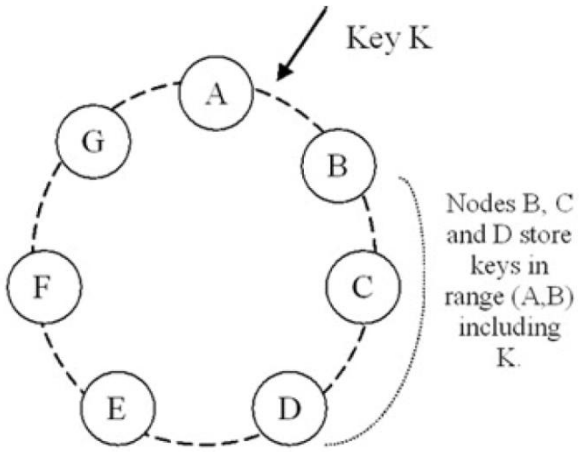
# Aufbau (Cassandra)

- Beispiel: Cassandra



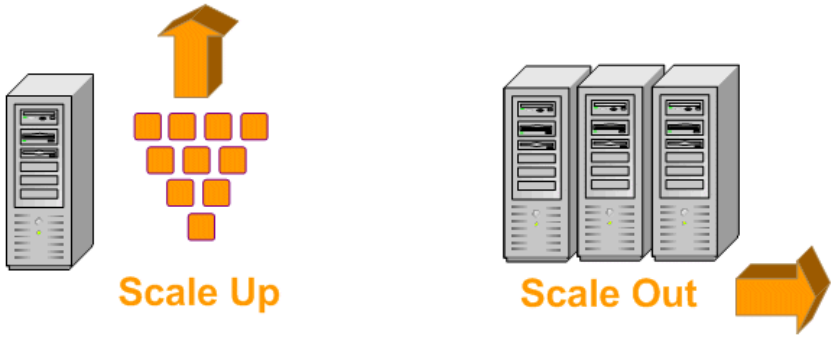
# Aufbau (Cassandra)

- Knoten sind in einem logischen Ring angeordnet
- P2P-Protokoll (Gossip)
  - Kommunikation unter den Knoten
  - Ausfallbenachrichtigungen
- Redundanz
  - Redundanz durch Replikation
  - Mehr Festplattenspeicher für Daten
  - Kein RAID nötig
- Ausfallsicherheit
  - alle Knoten gleichberechtigt (keine Masterserver)



# Skalierbarkeit

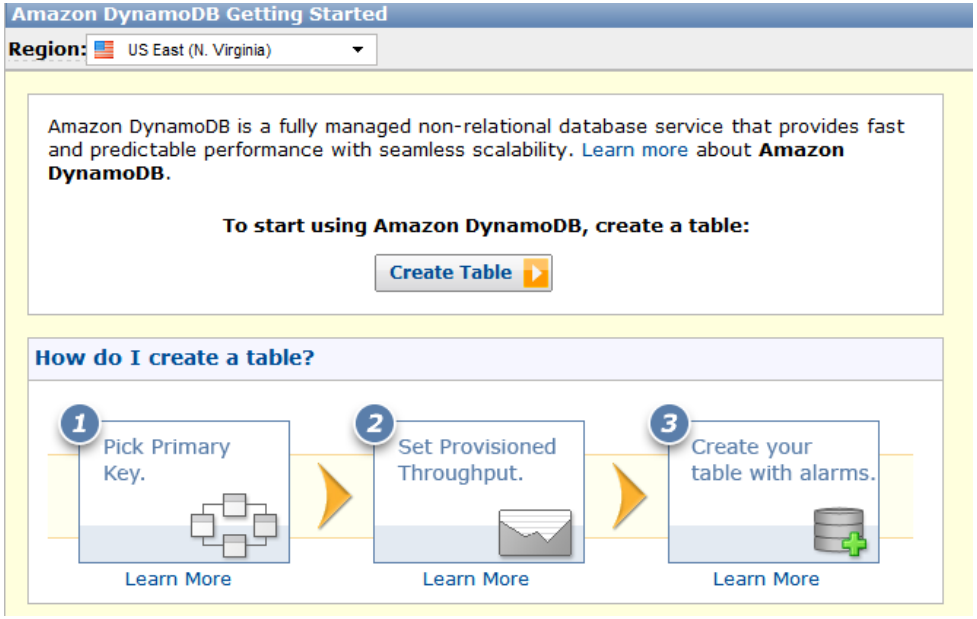
- Verteilte Datenbanken skalieren horizontal
- Standardhardware anstelle teurer Datenbank-Rechner




- Hinzufügen von Knoten ist denkbar einfach
  - Integration in logischen Ring mithilfe von Token

# Wide Column Stores & Cloud

- Vorteil: Wide Column Stores sind skalierbar „on demand“
- Bsp. DynamoDB von Amazon WS



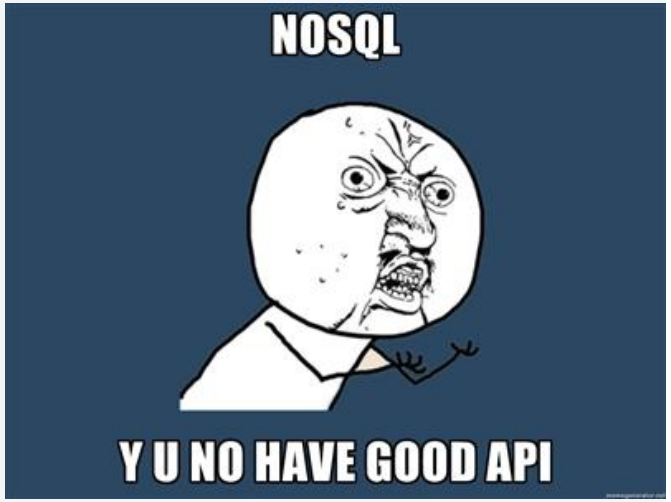

So, how do I query the database?
It's not a database. It's a key-value store!
Ok, it's not a database. How do I query it?
You write a distributed map reduce function in Erlang!
Did you just tell me to go fuck myself?
I believe I did, Bob.

# API

- Datenzugriff & Manipulation
  - API (Java, C++, …)
  - Binärprotokoll/RPC (Thrift, Protocol Buffers)
  - High-Level API Wrapper (Astyanax / Hector)
- Beschränkter Funktionsumfang
  - Keine Joins
  - Keine Aggregationen
- Wenn Abfragesprache vorhanden:
  - Nicht standarisert
  - Stellt nur ein Subset aus SQL dar
  - Ändert sich bei teilweise bei neuen Releases

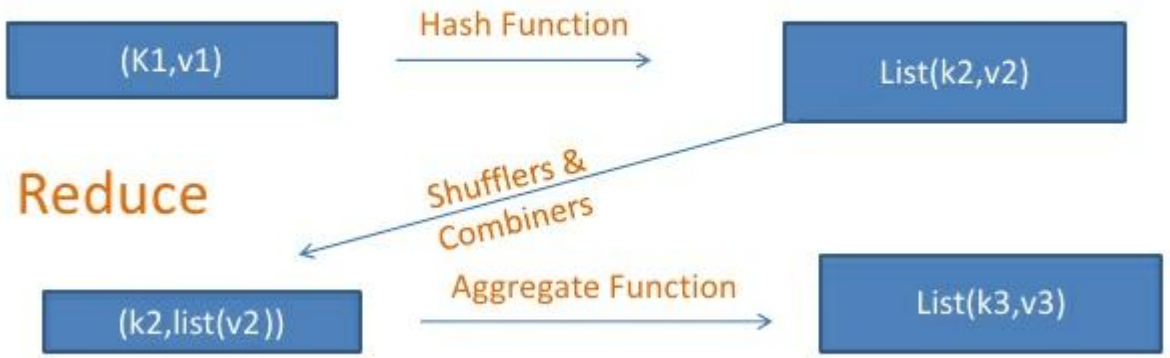
# Queries – Cassandra Insert (Java)

```
TTransport tr = new TFramedTransport(new TSocket("localhost", 9160));
TProtocol proto = new TBinaryProtocol(tr);

Cassandra.Client client = new Cassandra.Client(proto);
tr.open();
String keyspace = "Keyspace1";
client.set_keyspace(keyspace);
//record id
String key_user_id = "1";
String columnFamily = "Standard1";
// insert data
long timestamp = System.currentTimeMillis();
Random r = new Random(timestamp);
Column nameColumn = new Column(ByteBuffer.wrap("name".getBytes()));
nameColumn.setValue(Long.toHexString(r.nextLong()).getBytes());
nameColumn.setTimestamp(timestamp);

Column ageColumn = new Column(ByteBuffer.wrap("age".getBytes()));
ageColumn.setValue(Long.toHexString(r.nextLong()).getBytes());
ageColumn.setTimestamp(timestamp);

ColumnParent columnParent = new ColumnParent(columnFamily);
client.insert(ByteBuffer.wrap(key_user_id.getBytes()), columnParent,nameColumn,ConsistencyLevel.ALL) ;
client.insert(ByteBuffer.wrap(key_user_id.getBytes()), columnParent,ageColumn,ConsistencyLevel.ALL);

//Gets column by key
SlicePredicate predicate = new SlicePredicate();
predicate.setSlice_range(new SliceRange(ByteBuffer.wrap(new byte[0]), ByteBuffer.wrap(new byte[0]), false, 100));
List<ColumnOrSuperColumn> columnsByKey = client.get_slice(ByteBuffer.wrap(key_user_id.getBytes()), columnParent, predicate, ConsistencyLevel.ALL);
System.out.println(columnsByKey);


//Get all keys
KeyRange keyRange = new KeyRange(100);
keyRange.setStart_key(new byte[0]);
keyRange.setEnd_key(new byte[0]);
List<KeySlice> keySlices = client.get_range_slices(columnParent, predicate, keyRange, ConsistencyLevel.ONE);
System.out.println(keySlices.size());
System.out.println(keySlices);
for (KeySlice ks : keySlices) {
        System.out.println(new String(ks.getKey()));
}
tr.close();
```



# Einschub: Map/Reduce

- Parallele Abarbeitung eines Problems (Anfrage)
  - Problem muss zerlegbar sein (“Splits” von ~64MB)
  - Mapper: Filtert & Transformiert Splits
  - Reducer: Aggregiert Mapper-Ausgabe
    - Map
    - Reduce



# Einschub: Hive

- Map/Reduce:
  - Queries bzw. Jobs werden programmiert
  - APIs für Java, C++, Erlang, u.v.m
- Hive:
  - abstrahiert Map/Reduce Jobs mithilfe einer Querysprache (HiveQL):

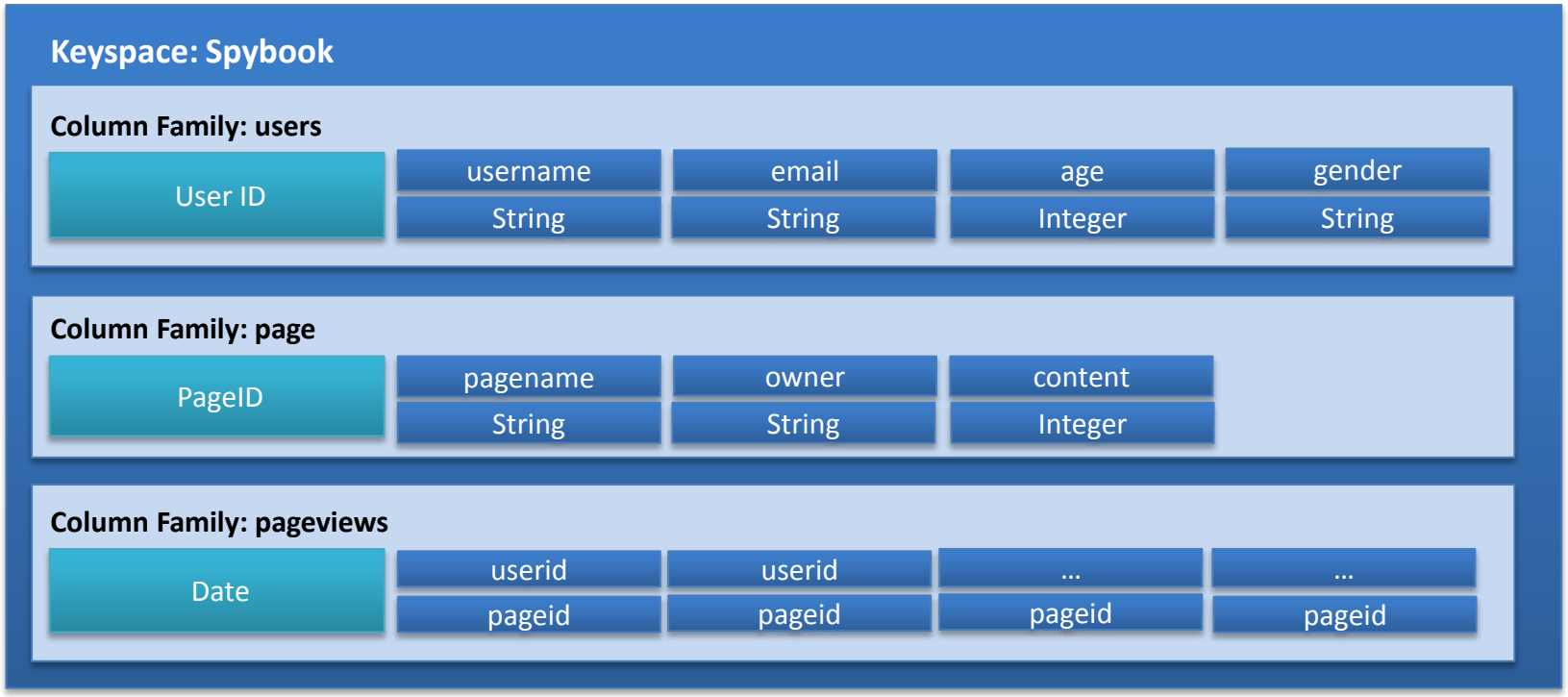
```
SELECT    pv.*, u.gender, u.age, f.friends
FROM      page_view pv
JOIN      user u ON (pv.userid = u.id)
JOIN      friend_list f ON (u.id = f.uid)
WHERE     pv.date = '2008-03-03';
```


# Anwendungsbeispiel

- Cassandra-Datenbank
  - Kleines Subset aus Facebook nachbauen (spybook)
    - Benutzer
    - Pages
    - Pageviews
- Map/Reduce Job:
  - Demographische Analyse über Pageviews
  - Abfrage wird mit Hive gestellt

# Anwendungsbeispiel

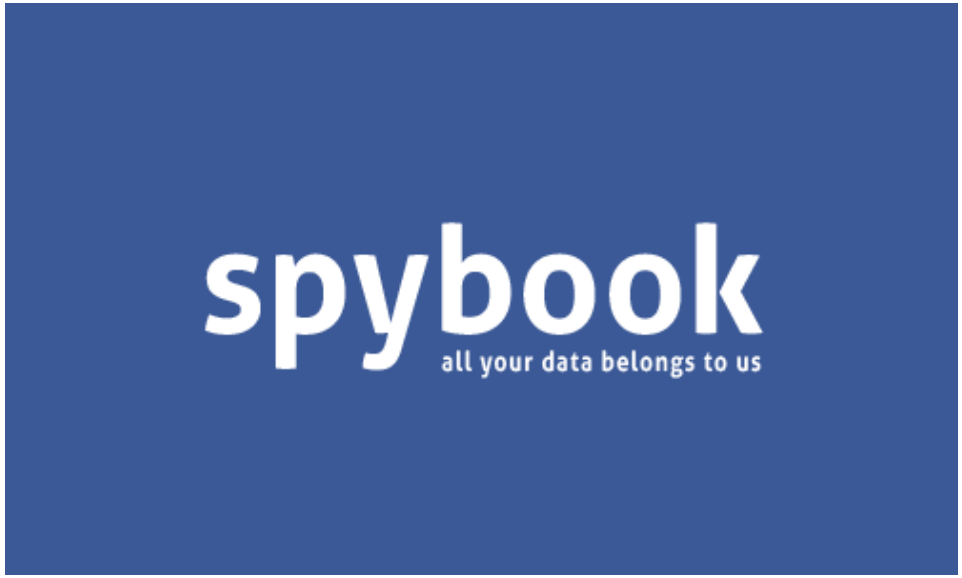
- Datenmodell:



# Anwendungsbeispiel

- Vorgehen:
  - Daten mit Java-Tool generiert
    - 3000 Benutzer
    - 100.000 Pageviews auf 10 Seiten
  - Daten werden in Cassandra geschrieben
  - Mapping von Cassandra nach Hive
  - Abfrage ausführen (Map/Reduce)

# Live Demo



# Anwendungsbeispiel: Infrastruktur

| Amazon Web Service | Open-Source Substitut |
|---|---|
| Amazon EC2 | -- (lokal: Virtual Box) |
| Amazon DynamoDB | Cassandra |
| Amazon S3 / Dynamo | Hadoop / CassandraFS |
| Amazon ElasticMapReduce | Hadoop mit Hive |

Vielen Dank für die Aufmerksamkeit.

# Fragen?

# Quellen & Links

- http://www.slideshare.net/royans/facebooks-petabyte-scale-data-warehouse-using-hive-and-hadoop
- http://www.slideshare.net/hurrycane/nosql-in-the-context-of-social-web-4348152
- http://en.wikipedia.org/wiki/CAP_theorem
- http://de.wikipedia.org/wiki/MapReduce

- Papers:
  - BigTable: http://static.googleusercontent.com/external_content/untrusted_dlcp/research.google.com/de//archive/bigtable-osdi06.pdf
  - Dynamo: http://w.lifeisagraph.com/p2p/dynamo.pdf