A Tale of Two Stores

DimeStore vs TakeYourNickelBack.com
DimeStor

The official unofficial Nickelback Fan-Store

NICKELBACK

FEED THE MACHINE TOUR
The Beginning of DimeStore
Cloud Computing

SaaS

PaaS
- Security Identity
- Integration Workflow
- Application Grid
- UI Services
- Database Grid

Admin. Services
- Packaging
- Configuration
- Deployment
- Scaling
- Lifecycle
- Management
- Utilization
- User Mgmt.
- IDE
- Integration

IaaS
- Virtual Machine
- Virtual Storage
- Virtual Network
1. Sign in to the AWS Management Console and open the Amazon S3 console at [https://console.aws.amazon.com/s3/](https://console.aws.amazon.com/s3/).
2. Create two buckets that match your domain name and subdomain (dimestore.com and www.dimestore.com).

**Use the Amazon S3 console to configure the bucket for website hosting**
1. In the S3 buckets list, choose the bucket with the same name as your domain. Then Choose Properties -> Static website hosting, Use this bucket to host a website & Index Document box, enter the name of your index page (index.html)
2. Redirect requests from www.dimestore.com to dimestore.com: Choose Properties -> Choose Static website hosting -> Choose Redirect requests. In the Target bucket or domain box, enter your domain (for example, example.com).
3. Upload Index and Website Content
5. Get Your Endpoints and Test Your Domain Endpoint
6. Add Alias Records for dimestore.com and www.dimestore.com

Amazon S3

Werner Vogel: “Let me emphasize the internal technology part before it gets misunderstood: Dynamo is not directly exposed externally as a web service; however, Dynamo and similar Amazon technologies are used to power parts of our Amazon Web Services, such as S3.”

Eventual Consistency

Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall, Werner Vogels: Dynamo: amazon's highly available key-value store. SOSP 2007: 205-220
After one too many arguments about how all the music that everyone else likes is terrible, and Sam finds that he is the only one with impeccable musical taste. Sam decides to start his own Anti-Nickelback website for making fun of these people.

Sam distrusts all large businesses (except hardware manufacturers for some reason) and decides to host the website on a web server running in his basement.
DimeStor
The official unofficial Nickelback Fan-Store
DimeStore is getting traction
DimeStores first real web-store

Amazon RDS database engines

[Images of various database engines such as Aurora, PostgreSQL, MySQL, MariaDB, Oracle, and SQL Server]
Typical scalable architecture on Amazon
Why do we need a database and transactions?
What do you do when your database fails?

Primary Backup

Application Server

Database Requests

Primary

Backup

Updates
What do you do when your database fails?

Primary Backup

Application Server  Application Server  Application Server

Database Requests

Backup

Updates
Amazon RDS - Traditional Replication

Steps 1, 3, and 5 are sequential and synchronous.
**Vertical Scaling**

Modifying DB Instance:
- **DB Engine Version**: MySQL 5.6.27 (default)
- **DB Instance Class**: db.2.large — 2 vCPU, 8 GB RAM
- **Multi-AZ Deployment**: db.m3.xlarge — 8 vCPU, 30.5 GB RAM
- **Storage Type**: m3.large — 2 vCPU, 16 GB RAM
- **Allocated Storage**: m3.large — 2 vCPU, 16 GB RAM

**Settings**
- **DB Instance Identifier**: db.m3.xlarge
- **New Master Password**: db.m3.xlarge

**Network & Security**
- **Security Group**: db.m3.xlarge

**Creates potential downtime**

---

**Horizontal Scaling**

Application Servers with MySQL Connectors

- **L4 or L7 Load Balancer**
- **Delete, Update, Insert**
- **Select**
- **RDS DB Instance**
- **RDS DB Instance Read Replica**
- **RDS DB Instance Read Replica**

**Consistent reads**
Amazon Aurora

The log stream generated by the writer and sent to the storage nodes is also sent to all read replicas. Each replica typically lags behind the writer by a short interval (20 ms or less).

Based on MySQL. Changed InnoDB version.
Everyone thinks they have impeccable musical taste and emails wanting to buy anti-nickelback merch.

David Grohl @FooGDave 14h
If you play a Nickelback song backwards you’ll hear messages from the devil. Even worse, if you play it forwards you’ll hear Nickelback.
TakeYourNickelBack.com

• Has a forum for people complaining about Nickelback
• Has a store for great Anti-Nickelback merch
• Allows users to design their own merch and the top voted designs will be sold on the website.
• Has a blog where I talk about how much better my music tastes are than my friends. Allows users to like and comment.
• Maybe more in the future.

• Maybe I could build on top of an existing embedded database key value store like RocksDB or SQLite
Choice of DBMS

Embedded DBMS
- If your application server fails, you lose data
- Do not easily scale to application workload
- Usually runs in the same process
SQLite

- An embedded SQL database that supports transactions. A row-store optimized for transactions.

- Disadvantages: Single user and not built for large applications.
SQLite

• An embedded SQL database that supports transactions. A row-store optimized for transactions.

• Disadvantages: ?
RocksDB

• Log Structure Merge Tree (previous lecture by Sam)
• You get a key-value database (like a hash table) that stores data on disk, supports transactions and is optimized for write-heavy workloads, and can support range scans.
• Is used as a storage engine for other systems (MySQL)

• Disadvantages: ?
RocksDB

• Log Structure Merge Tree (previous lecture by Sam)
• You get a key-value database (like a hash table) that stores data on disk, supports transactions and is optimized for write-heavy workloads, and can support range scans.
• Is used as a storage engine for other systems (MySQL)

• Disadvantages:
  1. Lacks a full query language, and many common database features. If you need foreign keys, columns, indexes etc you have to build it yourself.
  2. Single User
  3. Doesn’t scale beyond a single process (what do we do when our website grows to more users?)
Choice of DBMS

**Embedded DBMS**
- If your application server fails, you lose data
- Do not easily scale to application workload
- Usually runs in the same process

**Independent DBMS**
- Scale Application and DB independently
- Designed for many concurrent users
What are my options for data management?

• A full featured transaction processing SQL system like Postgres, MySQL (MariaDB), Oracle or Microsoft SQL Server (but not open source). Optimized for many concurrent users and offers failover features.

• Used by the largest companies for their most important operational data. If tuned well, you can trust them to handle your data properly.

• Since we are running these ourselves, we need to hire a database administrator to ensure things are running smoothly and safely.
What does it mean to host your own database on-premise (or on EC2)

• Increasing reliability requires buying new machines
• Scale-out/Scale-up requires buying new machines
• They have to be configured correctly
  • Independent drives
  • Backups
  • ...
• Disaster recovery needs to be automated (how to recover from power-outage)
• Load-balancing etc. needs to be implemented
• Hire administrators
•..

[Image of a person with an 'X' over their face]
What about NoSQL?

- Stores JSON Documents in key-value pairs
- JSON only API
- Distributed across multiple nodes
- Has indexes for fast access
- Tend to store data in a denormalized way (instead of customer, orders table, each customer document stores all their orders)
- Started with no multi-document transactions, no joins. Have since added this functionality
- Have to give up SQL functionality, query optimization and other common DB features.
- Is Web Scale
MongoDB for building a BitCoin exchange?

How Do You Transfer BitCoins with MongoDB?
“Flexcoin was a Bitcoin exchange that shut down on March 3rd, 2014, when someone allegedly hacked in and made off with 896 BTC in the hot wallet. Because the half-million dollar heist from the hot wallet was too large for the company to bear, it folded.”

“The attacker successfully exploited a flaw in the code which allows transfers between flexcoin users. By sending thousands of simultaneous requests, the attacker was able to "move" coins from one user account to another until the sending account was overdrawn, before balances were updated. “

mybalance = database.read("account-number")
newbalance = mybalance - amount
database.write("account-number", newbalance)
dispense_cash(amount) // or send bitcoins to customer

See more at: http://hackingdistributed.com/2014/04/06/another-one-bites-the-dust-flexcoin/#sthash.rWoYKi78.dpuf
MongoDB Started To Support Secondary Indexes

No Recovery for Indexes???
Not Only SQL
The rapid growth of Nickelback fans causes a backlash, and my store is growing!

• I bought a ton of new machines and put them in my basement!
• My ISP is starting to wonder what I’m doing and my hardware costs are through the roof.
DimeStor
The official unofficial Nickelback Fan-Store
Sales are dropping and we need to find new revenue.
Data Science to the rescue
Data Science to the rescue
Data Science to the rescue
OLTP Schema

OrderStatus
Order
Line Items
Reviews
Products

Customer

OLAP Schema

FactTable
OrderStatus

Product

Customer

ETL

AWS Glue
Star Schema

**Shop**
- Shop_ID
- Business_Type
- City
- City_Population
- State
- ...

**Time**
- Date_ID
- Month
- Quarter
- Year
- ...

**Customer**
- Customer_ID
- Name
- Segment
- Group_Name
- ...

**Product**
- Product_ID
- Type
- Brand
- Description
- ...

**Fact Table**
- Shop_ID
- Customer_ID
- Date_ID
- Product_ID
- Amount
- Volume
- Profit
- ...

---

**Star Schema**
## Star vs. Snowflake Schema

<table>
<thead>
<tr>
<th></th>
<th>Snowflake</th>
<th>Star</th>
</tr>
</thead>
<tbody>
<tr>
<td>Space</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Query Performance</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ease of Use</td>
<td></td>
<td></td>
</tr>
<tr>
<td>When to use</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Normalization/De-Normalization</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Star vs. Snowflake Schema

<table>
<thead>
<tr>
<th></th>
<th>Snowflake</th>
<th>Star</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Space</strong></td>
<td>Smaller</td>
<td>Bigger (Redundancy)</td>
</tr>
<tr>
<td><strong>Query Performance</strong></td>
<td>More Joins → slower</td>
<td>Fewer Joins → faster</td>
</tr>
<tr>
<td><strong>Ease of Use</strong></td>
<td>Complex Queries</td>
<td>Pretty Simply Queries</td>
</tr>
<tr>
<td><strong>When to use</strong></td>
<td>When dimension table is relatively big in size, snowflaking is better as it reduces space.</td>
<td>When dimension table contains less number of rows, we can go for Star schema.</td>
</tr>
<tr>
<td><strong>Normalization/De-Normalization</strong></td>
<td>Dimension Tables are in Normalized form but Fact Table is still in De-Normalized form</td>
<td>Both Dimension and Fact Tables are in De-Normalized form</td>
</tr>
</tbody>
</table>
What are the most common use cases for a data warehouse?

Data Exploration

Dashboards

(AWS QuickSight)
Amazon Redshift

The leader node accepts connections from client programs, parses requests, generates & compiles query plans for execution on the compute nodes, performs final aggregation of results when required.

Anurag Gupta, Deepak Agarwal, Derek Tan, Jakub Kulesza, Rahul Pathak, Stefano Stefani, Vidhya Srinivasan: Amazon Redshift and the Case for Simpler Data Warehouses. SIGMOD Conference 2015: 1917-1923
The user can specify whether data is distributed in a round robin fashion, hashed according to a distribution key, or duplicated on all slices.
Amazon Redshift Spectrum
Life of a query

Query
SELECT COUNT(*)
FROM S3.EXT_TABLE
GROUP BY...

Amazon S3
Exabyte-scale object storage

Data Catalog
Apache Hive Metastore
Query is optimized and compiled at the leader node. Determine what gets run locally and what goes to Amazon Redshift Spectrum.
Life of a query

Query plan is sent to all compute nodes

Amazon S3
Exabyte-scale object storage

Data Catalog
Apache Hive Metastore
Life of a query

Compute nodes obtain partition info from Data Catalog; dynamically prune partitions

Amazon S3
Exabyte-scale object storage

Data Catalog
Apache Hive Metastore
Life of a query

5 Each compute node issues multiple requests to the Amazon Redshift Spectrum layer.

Amazon S3
Exabyte-scale object storage

Data Catalog
Apache Hive Metastore
Life of a query

Amazon Redshift Spectrum nodes scan your S3 data

Amazon S3
Exabyte-scale object storage

Data Catalog
Apache Hive Metastore
Life of a query

Amazon Redshift

Spectrum projects, filters, joins and aggregates

Amazon S3

Exabyte-scale object storage

Data Catalog

Apache Hive Metastore
Life of a query

Final aggregations and joins with local Amazon Redshift tables done in-cluster

Amazon S3
Exabyte-scale object storage

Data Catalog
Apache Hive Metastore
Life of a query

Result is sent back to client

Amazon Redshift

JDBC/ODBC

Amazon S3
Exabyte-scale object storage

Data Catalog
Apache Hive Metastore

1
2
3
4
N
Comparison of DW

AWS Lambda

Online order is placed → Amazon DynamoDB: Order data is stored in an operational database → Lambda is triggered → AWS Lambda: Lambda runs data transformation code → Amazon Redshift: Lambda loads results into data warehouse → Analytics generated from data

https://aws.amazon.com/blogs/architecture/ten-things-serverless-architects-should-know/
• As we get more cash hungry, we decide to do analytics on our companies data. We purchase data from Cambridge Analytica to find more potential customers (shh)

• Now we need an analytical system that can store tens of terabytes of data and query it quickly (for our new Marketing team)
Data Warehousing systems

• Not as many options for Open source systems
• Mostly have columnar storage and are optimized for analytical queries on TBs of data
• Run SQL but are not optimized for small transactions
• Usually ingesting new data from operational database at a granularity of minutes to days
Extract Transform Load

• Need a way to get our data from operational system to the analytical systems.
• Extract data from the operational system (where we are making sales)
• Transform it as necessary for our analytical system
• Load it into the analytical system
• Could use Spark or Hadoop to write these jobs (and many others)
Unfortunately it only got worse
Last attempt – Pay for positive reviews

**Make Money**
by working on HITs

HITs - Human Intelligence Tasks - are individual tasks that you work on. Find HITs now.

As a Mechanical Turk Worker you:
- Can work from home
- Choose your own work hours
- Get paid for doing good work

![Find HITs Now](image)

**Get Results**
from Mechanical Turk Workers

Ask workers to complete HITs - Human Intelligence Tasks - and get results using Mechanical Turk. Get started.

As a Mechanical Turk Requester you:
- Have access to a global, on-demand, 24 x 7 workforce
- Get thousands of HITs completed in minutes
- Pay only when you're satisfied with the results

![Get Started](image)
Microtasking – Virtualized Humans

• Current leader: Amazon Mechanical Turk

• Requestors place Human Intelligence Tasks (HITs)
  • Minimum price: $0.01
  • Other parameters: #of replicas (assignments), expiration, User Interface,…
  • API-based: “createHit()”, “getAssignments()”, “approveAssignments()”, “forceExpire()”
  • Requestors approve jobs and payment

• Workers (a.k.a. “turkers”) choose jobs, do them, get paid
Aftermath

Nickelback ranked one of the worst bands ever

According to rollingstone.com

Readers' Poll: The Ten Worst Bands of the Nineties
1. Creed. It’s no surprise that Creed won this poll.
2. Nickelback. It’s hard not to feel a little bad for Nickelback
3. Limp Bizkit. If you think that Limp Bizkit fans are a bunch of bullies, you aren’t alone. ...
4. Hanson. ...
5. Nirvana. ... 
6. Hootie and the Blowfish. ...
7. Bush. ...
8. Spin Doctors. ...