Local Databases

- Registers
- Cache/Buffer
- RAM (Main Memory)
- Hard Disk
- localStorage/sessionStorage (in browser)

(Extra credit: http://www.eecs.berkeley.edu/~rcs/research/interactive_latency.html)

Relational Databases

- MySQL
- Postgres
- SQL Server
- Oracle DB

(Extra credit: http://sql.learncodethehardway.org/)

NoSQL Databases

- Key/Value Stores
    - Memcache
        - Best used: when you want a really, really fast in-memory store. This has limited storage capability though because you will lose information when the user leaves the session.
        - For example: session storage and single use counters
    - Redis
        - Best used: For rapidly changing data with a foreseeable database size (should fit mostly in memory).
        - For example: Stock prices. Analytics. Real-time data collection. Real-time communication. And wherever you used memcached before.
- Document Stores
    - MongoDB (Mongoose)
        - Best used: If you need dynamic queries. If you prefer to define indexes, not map/reduce functions. If you need good performance on a big DB. If you wanted CouchDB, but your data changes too much, filling up disks.
        - For example: For most things that you would do with MySQL or PostgreSQL, but having predefined columns really holds you back.
    - CouchDB
        - Best used: For accumulating data that rarely changes in structure, on which pre-defined queries are to be run. Places where versioning is important.
        - For example: CRM, CMS systems. Master-master replication is an especially interesting feature, allowing easy multi-site deployments.
    - Cassandra
        - Best used: When you write more than you read (logging). If every component of the system must be in Java. ("No one gets fired for choosing Apache's stuff.")
        - For example: Banking, financial industry (though not necessarily for financial transactions, but these industries are much bigger than that.) Writes are faster than reads, so one natural niche is data analysis.
    - Riak
        - Best used: If you want something Dynamo-like data storage, but no way you're gonna deal with the bloat and complexity. If you need very good single-site scalability, availability and fault-tolerance, but you're ready to pay for multi-site replication.
        - For example: Point-of-sales data collection. Factory control systems. Places where even seconds of downtime hurt. Could be used as a well-update-able web server.
- Graph Databases
    - Neo4j (V1.5M02)
        - Best used: For graph-style, rich or complex, interconnected data. Neo4j is quite different from the others in this sense.

        - For example: For searching routes in social relations, public transport links, road maps, or network topologies.
- Big Data Stores
    - HBase (V0.92.0)
        - Best used: Hadoop is probably still the best way to run Map/Reduce jobs on huge datasets. Best if you use the Hadoop/HDFS stack already.
        - For example: Search engines. Analysing log data. Any place where scanning huge, two-dimensional join-less tables are a requirement.
    - ElasticSearch (0.20.1)
        - Best used: When you have objects with (flexible) fields, and you need "advanced search" functionality.
        - For example: A dating service that handles age difference, geographic location, tastes and dislikes, etc. Or a leaderboard system that depends on many variables.
    - Hypertable (0.9.6.5)
        - Best used: If you need a better HBase.
        - For example: Same as HBase, since it's basically a replacement: Search engines. Analysing log data. Any place where scanning huge, two-dimensional join-less tables are a requirement.

(Extra credit: http://research.google.com/archive/bigtable.html; Search for Cloudera and Hortonworks)

Advanced Databases

- **Percolator** (Caffiene): Handling individual updates, a NoSQL db that can handle transactions
- **Hummingbird** – major search index update, including complex search and voice search
- **Pregel**: scalable graph computing, for calculating data based on social graphs, Facebook
- **Dremel**: an interactive database with an SQL-like language for structured data.
- **Spanner**: planet-scale structured storage system, next generation of BigTable stack

(Extra credit: https://developers.google.com/bigquery/what-is-bigquery)

Other Concepts

- CAP Theorem - Consistency, Availability, Partition Tolerance
- Relational DB vs. NoSQL DB
- Normalized vs. Denormalized
- Horizontal Partitioning (Sharding) vs. Vertical Partitioning
- Inner Join vs. Left Join vs. Right Join vs. Outer Join
- Object Relational Mappers (ORMs and ODMs)