CSE 180, Database Systems I
Shel Finkelstein, UC Santa Cruz
Lecture 1

Chapter 1: Introduction

Database System Concepts, 7th Edition.

Modified by Shel Finkelstein for CSE 180

Database System Applications

- DBMS contains information about a particular enterprise.
  - Collection of interrelated data
  - Set of programs to access the data
  - An environment that is both *convenient* and *efficient* to use
- Database systems are used to manage collections of data that are:
  - Highly valuable
  - Relatively large
  - Accessed by multiple users and applications, often at the same time.
- A modern database system is a complex software system whose task is to manage a large, complex collection of data.
- Databases touch all aspects of our lives.

Database Application Examples

- Enterprise Information
  - Sales: customers, products, purchases
  - Accounting: payments, receipts, assets
  - Human Resources: Information about employees, salaries, payroll taxes
- Manufacturing: management of production, inventory, orders, supply chain.
- Banking and finance
  - Customer information, accounts, loans, and banking transactions.
  - Credit card transactions
  - Finance: sales and purchases of financial instruments (e.g., stocks and bonds; storing real-time market data
- Universities: registration, grades

Database Application Examples (Cont.)

- Airlines: reservations, schedules
- Telecommunication: records of calls, texts, and data usage, generating monthly bills, maintaining balances on prepaid calling cards
- Web-based services
  - Online retailers: order tracking, customized recommendations
  - Online advertisements
- Document databases
- Navigation systems: For maintaining the locations of varies places of interest along with the exact routes of roads, train systems, buses, etc.

Purpose of Database Systems

In the early days, database applications were built directly on top of file systems, which leads to:

- Data redundancy and inconsistency: data is stored in multiple file formats resulting induplication of information in different files
- Difficulty in accessing data
  - Need to write a new program to carry out each new task.
- Data isolation
  - Multiple files and formats.
- Integrity problems
  - Integrity constraints (e.g., account balance > 0) become "buried" in program code rather than being stated explicitly.
  - Hard to add new constraints or change existing ones.

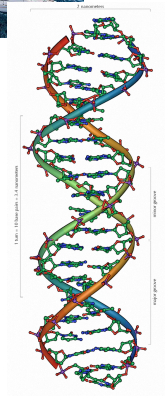
Purpose of Database Systems (Cont.)

- Atomicity of updates
  - Failures may leave database in an inconsistent state with partial updates carried out.
  - Example: Transfer of funds from one account to another should either complete or not happen at all.
- Concurrent access by multiple users
  - Concurrent access needed for performance.
  - Uncontrolled concurrent accesses can lead to inconsistencies.
    - Ex: Two people reading a balance (say 100) and updating it by withdrawing money (say 50 each) at the same time.
- Security problems
  - Hard to provide user access to some, but not all, data.

Database systems offer solutions to all of these problems..

Data…is Everywhere

MOST ENTERPRISES TODAY GENERATE MORE DATA THAN THEY CAN PROCESS



facebook

social

finance

health/
medical

learning

transportation

science

government

retail

entertainment

…AND THE AMOUNT OF DATA IS GROWING AT 50% PER YEAR

- according to IDC

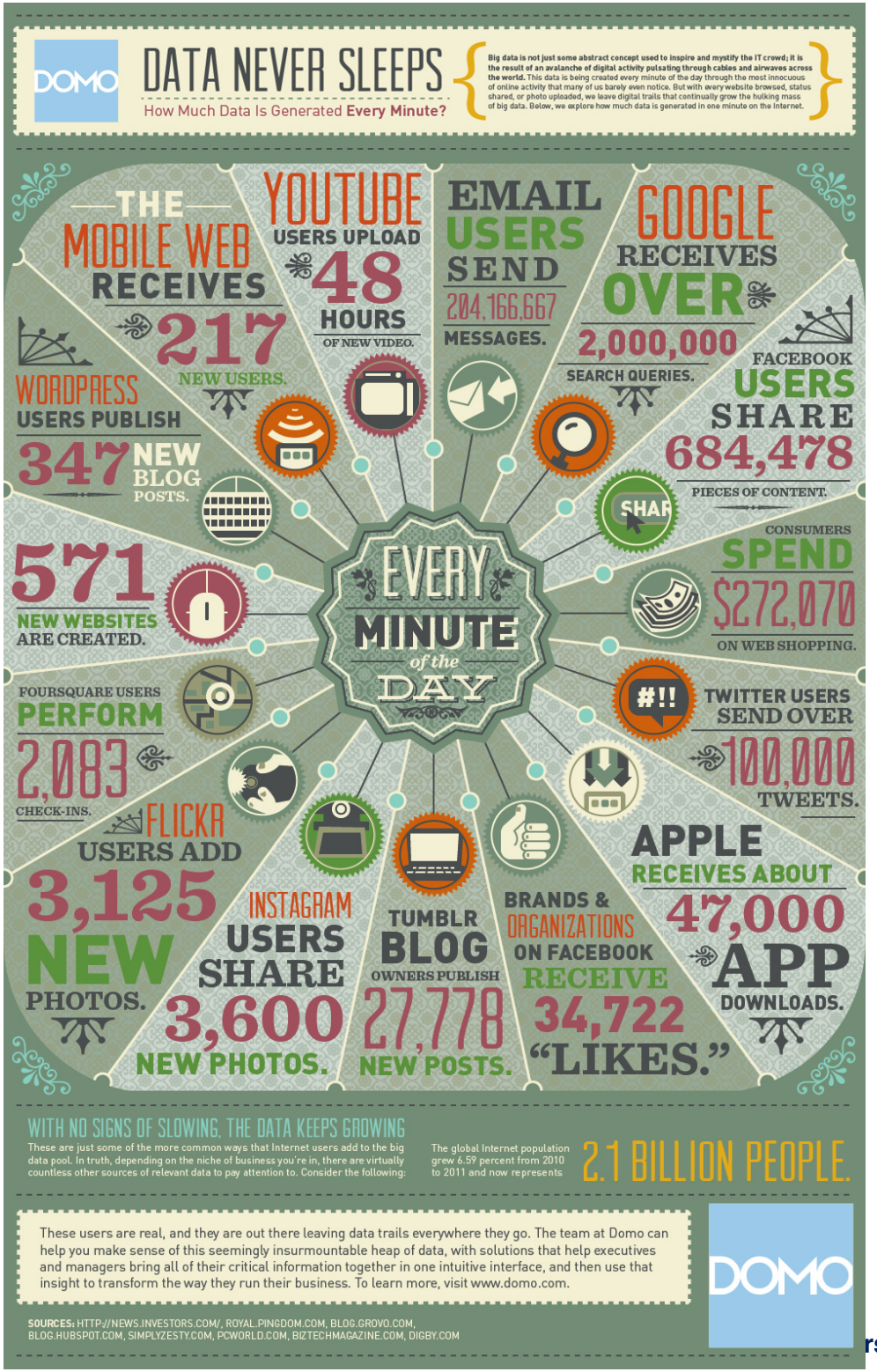
Data Never Sleeps 7.0

In every minute of every day, loads of data are being generated.

Just how much, you ask?

A lot happens in a minute:

- YouTube users upload 48 hours of new video
- Instagram users share 3,600 new photos
- Brands and organizations on Facebook receive 34,722 “likes”
- Over 100,000 tweets are sent

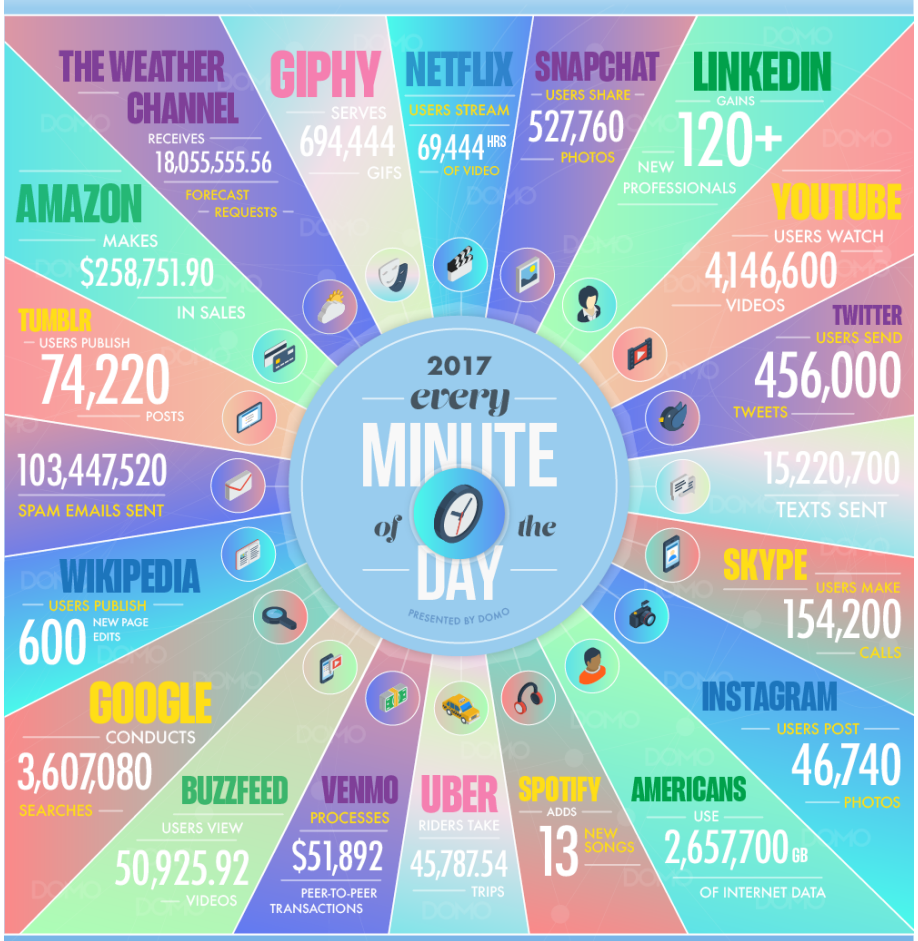


DATA NEVER SLEEPS 5.0

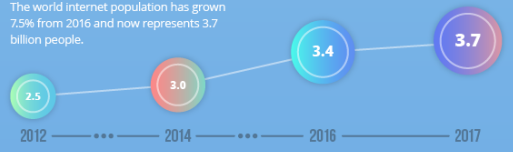
DOMO

How much data is generated *every minute?*

90% of all data today was created in the last two years—that's 2.5 quintillion bytes of data per day. In our 5th edition of Data Never Sleeps, we bring you the latest stats on just how much data is being created in the digital sphere—and the numbers are staggering.



The world internet population has grown 7.5% from 2016 and now represents 3.7 billion people.



GLOBAL INTERNET POPULATION GROWTH 2012–2017
(IN BILLIONS)

With each click, swipe, share, and like, businesses are using data to make decisions about the future. Domo gives everyone in your business real-time access to data from virtually any data source in a single platform for smarter decision-making at any moment.

Learn more at domo.com

SOURCES: EXPANDEDRAMBLINGS.COM, WEARESOCIAL.COM, WIKIPEDIA, FORBES, ADWEEK.COM, FORTUNE.COM, BLOOMBERG.COM, ONEREACH.COM, IBM, BUZZFEED, INTERNET LIVE STATS, INTERNET WORLD STATS, BBC

DOMO

Can You Answer These Questions?

Not a homework; will answer in class later this week, but for CSE 180, you should understand how to figure out answers to all of them.

0-If set S is {1,3,5,7} and set T is {2,3,5,7}, what are S UNION T and S INTERSECT T?

1-If set A is {1,2,3} and set B is {u,v,w,x,y}, how many ways can you pick pairs of items, with the first from A and the second from B?

2-If you have a set of employees (with names and salaries) where John makes 10K, George makes 20K, Ringo makes 30K and Paul makes 40K, what are the names of the employee(s) who make less than the average salary?

3-Can there ever be an employee who makes more than every employee? If so, give an example. If not, explain why not

4-Write the truth-table for p AND q, where p can be TRUE or FALSE and q can be TRUE or FALSE.

Data Models and Data Abstraction

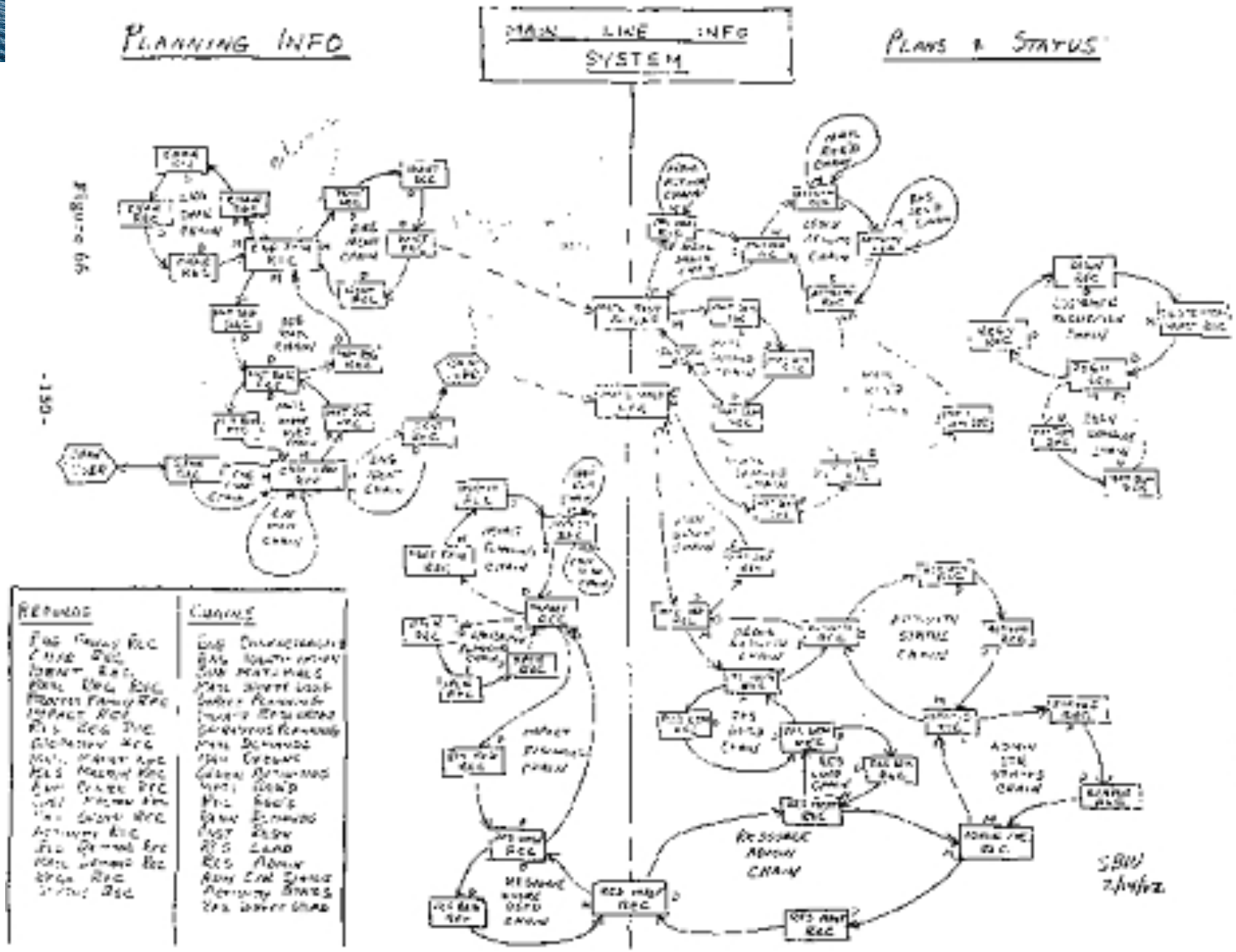
- A **database system** is a collection of interrelated data, and a set of capabilities that allow users to access and modify this data.
- A major purpose of a database system is to provide users with an abstract view of the data.
  - Data models
    - A collection of conceptual tools for describing data, data relationships, data semantics, and consistency constraints.
  - Data abstraction
    - Hide the complexity of data structures to represent data in the database from users through several levels of data abstraction.

Data Models

- A collection of tools for describing
  - Data
  - Data relationships
  - Data semantics
  - Data constraints
- Older models:
  - Network model
  - Hierarchical model
- Relational model
- Entity-Relationship data model (mainly for database design)
- Object-based data models (Object-oriented and Object-relational)
- Semi-structured data models (XML, JSON)

Network Data Model

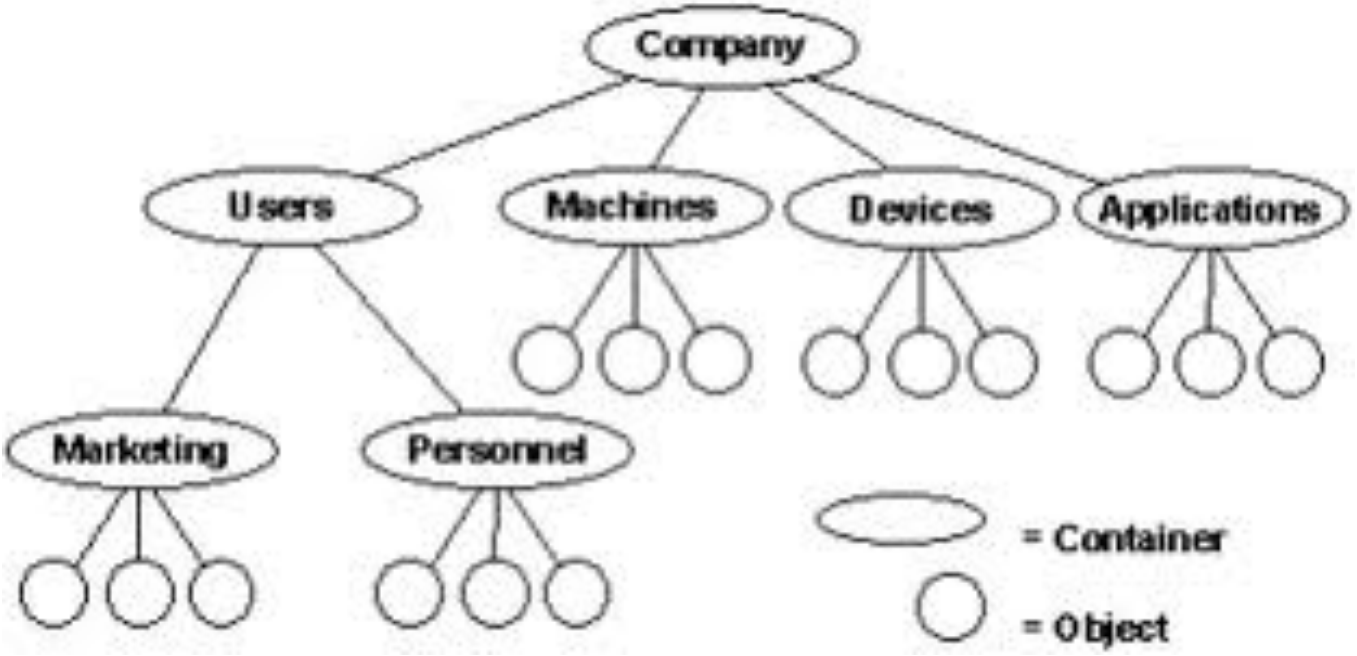
- 1960s
  - First general purpose DBMS was built.
  - Integrated data store (IDS)
    - by Charles Bachman of General Electric.
  - Network Data Model
    - The computer navigates through a space of data records connected by pointers. A graph-based data structure.
    - A user needs to formulate the process of navigating through records and pointers to compute an answer for a query.
  - 1973 Turing award lecture.
    - “The Programmer as Navigator”



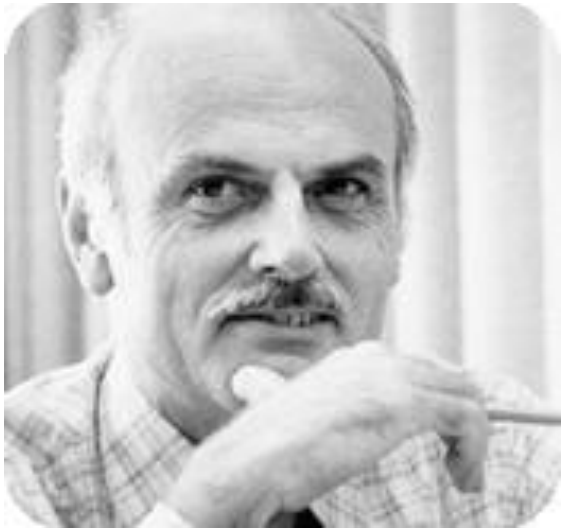
Source: Fifty years of databases http://wp.sigmod.org/?p=688

Hierarchical Data Model

- Also in the 1960s:
  - Hierarchical Data Model proposed IBM (IMS product)
  - A tree-based data structure.



Relational Data Model



- 1970s
  - The beginning of *relational* database management systems.
  - Edgar (Ted) F. Codd at the IBM San Jose Research Laboratory (now IBM Almaden Research Center) published a seminal paper:

  "*A relational model for data for large shared data banks*" Communications of the ACM, 1970.

Ted Codd's Relational Model

- Advocates the a radically different data model, called the *relational* data model.
  - *All* data must be stored in flat, table-like relations.
    - No pointers, no hierarchies!
  - Two database query languages:
    - Relational algebra and relational calculus

Employees

EmpNo	FirstName	LastName	Department
100	Sally	Baker	10-L
101	Jack	Douglas	10-L
102	Sarah	Schultz	20-B
103	David	Drachmeier	20-B

Equipment

SerialNum	Type	UserEmpNo
3009734-4	Computer	100
3-23-283742	Monitor	100
2-22-723423	Monitor	101
232342	Printer	100

Early Relational Projects and Products

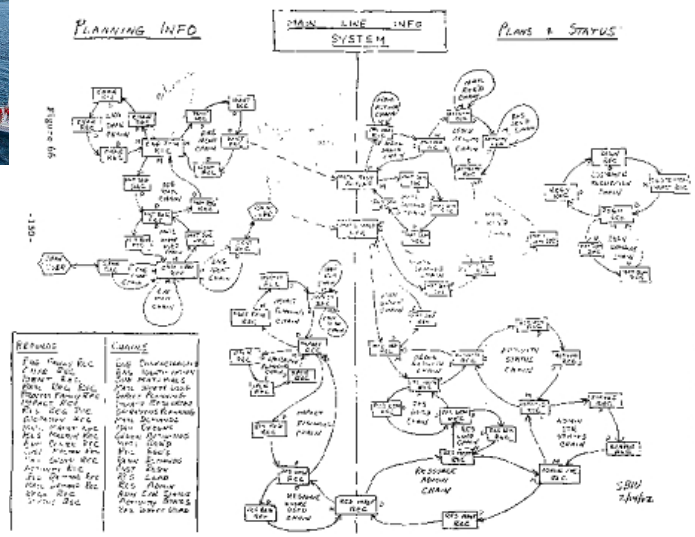
- System R project started at IBM Research in 1974.
  - System R became today's DB2.
  - 1981 Turing Award to Edgar F. Codd, "Relational Database: A Practical Foundation for Productivity".
  - 1998 Turing Award to Jim Gray for Transaction Processing.
- Michael Stonebraker and Eugene Wong at UC Berkeley started INGRES project based on Codd's papers.
  - Relational Technology Inc. became company, INGRES.
  - Later POSTGRES project led to company, Illustra Information Technologies and became open-source PostgreSQL.
  - 2014 Turing Award to Mike Stonebraker.
- Larry Ellison founded Relational Storage Inc., which became Oracle. First Oracle Relational Database Management System (RDBMS) was released in 1979 (and was called v2).

RDBMS Today

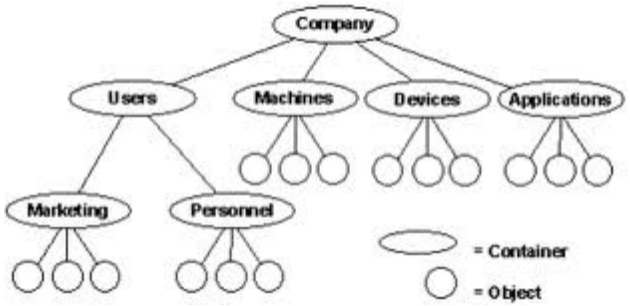
- Lots of relational database management systems
  - http://en.wikipedia.org/wiki/List_of_relational_database_management_systems
- Examples of open-source RDBMS
  - MySQL, PostgreSQL
- Examples of proprietary RDBMS
  - Oracle, IBM DB2, Microsoft SQL Server, SAP HANA
- Cloud relational databases
  - Amazon Aurora, Microsoft Azure SQL Database
  - Amazon RedShift, Microsoft Azure SQL Data Warehouse, Google BigQuery

Data Management without Databases

- 1990s
  - Emergence of Web commerce
  - Semi-structured data (XML)
- 2000s
  - Big data storage systems
    - Google BigTable, Yahoo PNuts, Amazon
    - “NoSQL” systems.
  - Big data analysis: beyond SQL
    - Map Reduce, Hadoop, and much more
  - Simpler semi-structured data web (JSON)
- 2010s
  - SQL reloaded (NOSQL vs NoSQL)
    - SQL front ends to Big Data systems
  - Massively parallel database systems
  - Multi-core main-memory databases
  - … and much more!



1960s: Network data model and hierarchical data model



1970s: Relational data model

EmpNo	First Name	Last Name	Dept. Num	Serial Num	Type	User EmpNo
100	Sally	Baker	10-L	3009734-4	Computer	100
101	Jack	Douglas	10-L	3-23-283742	Monitor	100
102	Sarah	Schultz	20-B	2-22-723423	Monitor	100
103	David	Drachmeier	20-B	232342	Printer	100


```
{
    "photos": {
        "page": 1,
        "pages": 94276,
        "perpage": 15,
        "total": "1414129",
        "photo": [{
            "id": "3891667770",
            "owner": "35468133120@N01",
            "secret": "4479baebf9",
            "server": "2451",
            "farm": 3,
            "title": "Mexican train dominoes with Brian and Michelle",
            "ispublic": 1,
            "isfriend": 0,
            "isfamily": 0
        },
        {
            "id": "3891661852",
            "owner": "10648301@N07",
            "secret": "79de582257",
            "server": "2500",
            "farm": 3,
            "title": "Peaches",
            "ispublic": 1,
            "isfriend": 0,
            "isfamily": 0
        },
```


2010s: JSON


```
<Books>
    <Book ISBN="0553212419">
        <title>Sherlock Holmes: Complete Novels...
        <author>Sir Arthur Conan Doyle</author>
    </Book>
    <Book ISBN="0743273567">
        <title>The Great Gatsby</title>
        <author>F. Scott Fitzgerald</author>
    </Book>
    <Book ISBN="0684826976">
        <title>Undaunted Courage</title>
        <author>Stephen E. Ambrose</author>
    </Book>
    <Book ISBN="0743203178">
        <title>Nothing Like It In the World</title>
        <author>Stephen E. Ambrose</author>
    </Book>
</Books>
```


1990s: XML (Semi-structured data model)

A Few Examples of Relations

- New York Stock Exchange Quotes
- Presidential info Page down or search for Washington
- Current NFL Stadium info
- 2018-2019 National Basketball Association Standings

Data Also Resides Outside Databases

- Before the Web
  - Business data typically resided in databases On-Premises
  - Typically sensitive enterprise information, such as bank accounts, employee data, sale transactions
- Today
  - Data resides in both databases and in less structured formats in files
  - Consumer data is a huge business: Advertising, Influencing
    - Webpage clicks, location data, analysis of social media and mail/text, sensor data, …
  - Data entry and data access is via the Web
    - More and more data is stored in the Cloud, rather than on Premises

Today

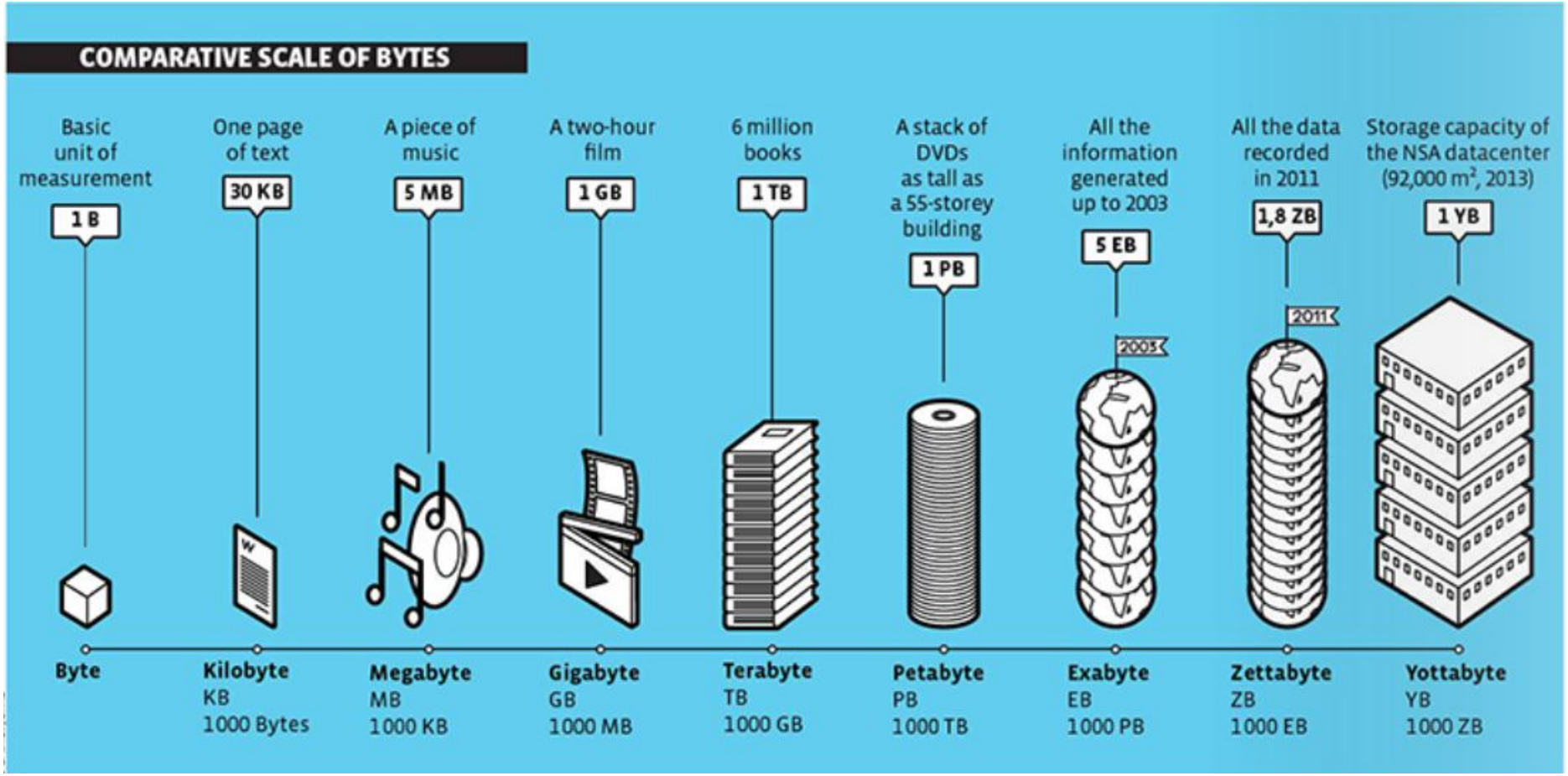
- NOSQL ("Not Only SQL") systems
  - Map/Reduce (Hadoop, Spark)
  - Column store (HBase, Cassandra)
  - Graph databases (Neo4J, Virtuoso)
  - Document databases (MongoDB)

- Simplicity of design, easy scale-out
- Compromise consistency in favor of availability and partition tolerance
- Lack of full ACID support, use of low-level query languages, lack of standardized interfaces (changing)

From Bytes to Yottabytes

Multiples of bytes V · T · E				
SI decimal prefixes		Binary usage	IEC binary prefixes	
Name (Symbol)	Value		Name (Symbol)	Value
kilobyte (kB)	10^3	2^{10}	kibibyte (KiB)	2^{10}
megabyte (MB)	10^6	2^{20}	mebibyte (MiB)	2^{20}
gigabyte (GB)	10^9	2^{30}	gibibyte (GiB)	2^{30}
terabyte (TB)	10^{12}	2^{40}	tebibyte (TiB)	2^{40}
petabyte (PB)	10^{15}	2^{50}	pebibyte (PiB)	2^{50}
exabyte (EB)	10^{18}	2^{60}	exbibyte (EiB)	2^{60}
zettabyte (ZB)	10^{21}	2^{70}	zebibyte (ZiB)	2^{70}
yottabyte (YB)	10^{24}	2^{80}	yobibyte (YiB)	2^{80}
See also: Multiples of bits · Orders of magnitude of data				

Comparative Size of Bytes – credit to Testyotta



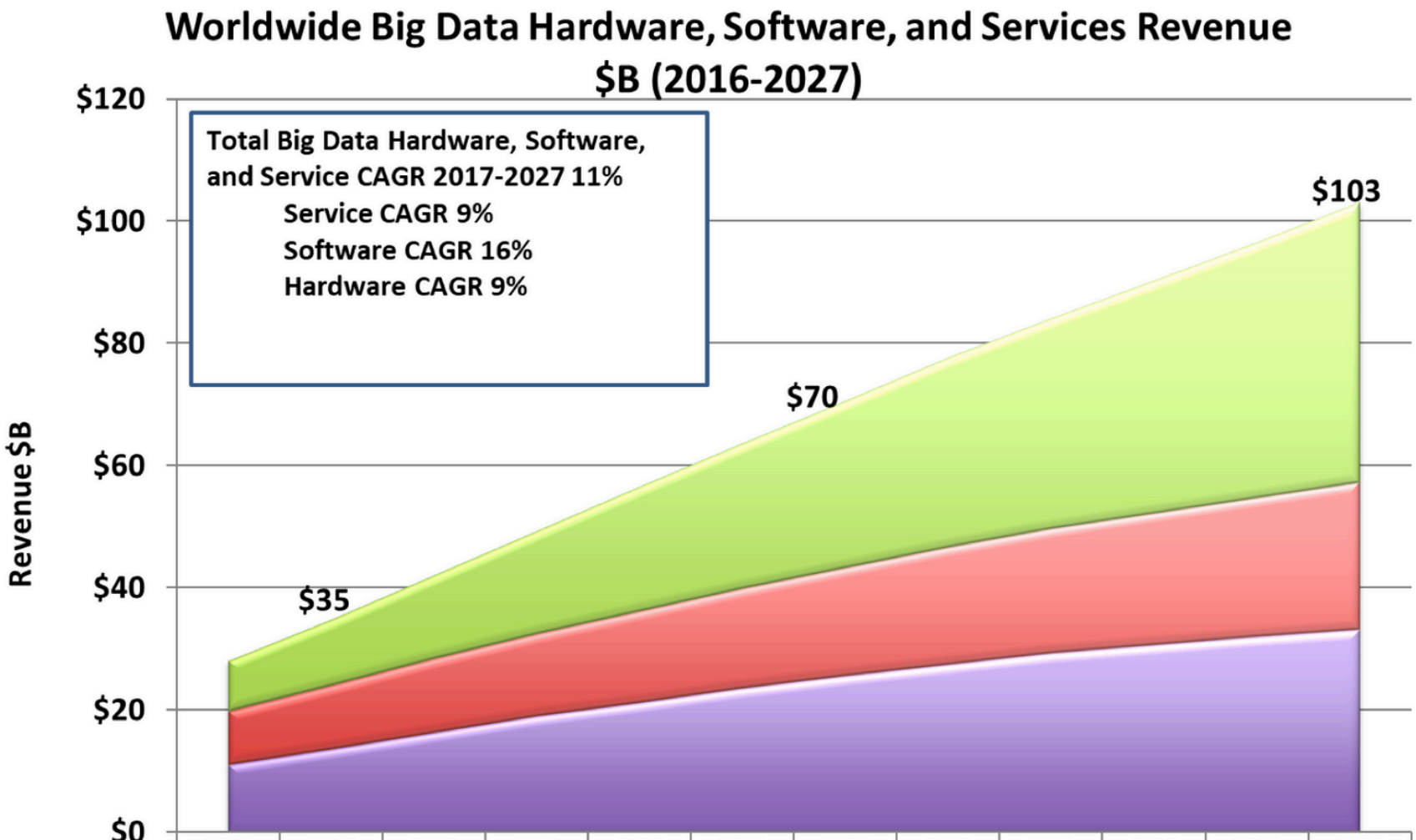
DB-Engines Popularity Rating for RDBMS

Rank			DBMS	Database Model	Score		
Aug 2019	Jul 2019	Aug 2018			Aug 2019	Jul 2019	Aug 2018
1.	1.	1.	Oracle	Relational, Multi-model	1339.48	+18.22	+27.45
2.	2.	2.	MySQL	Relational, Multi-model	1253.68	+24.16	+46.87
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	1093.18	+2.35	+20.53
4.	4.	4.	PostgreSQL	Relational, Multi-model	481.33	-1.94	+63.83
5.	5.	5.	IBM Db2	Relational, Multi-model	172.95	-1.19	-8.89
6.	6.	6.	Microsoft Access	Relational	135.33	-1.98	+6.24
7.	7.	7.	SQLite	Relational	122.72	-1.91	+8.99
8.	8.	↑ 9.	MariaDB	Relational, Multi-model	84.95	+0.52	+16.66
9.	9.	↑ 11.	Hive	Relational	81.80	+0.93	+23.86
10.	10.	↓ 8.	Teradata	Relational, Multi-model	76.64	-1.18	-0.77
11.	11.	↑ 12.	FileMaker	Relational	58.02	+0.12	+1.96
12.	12.	↓ 10.	SAP Adaptive Server	Relational	55.86	-0.79	-4.57
13.	13.	13.	SAP HANA	Relational, Multi-model	55.43	-0.11	+3.50
14.	14.	14.	Microsoft Azure SQL Database	Relational, Multi-model	27.99	-0.67	+1.89
15.	15.	15.	Informix	Relational, Multi-model	25.68	-0.18	+0.29
16.	16.	↑ 20.	Google BigQuery	Relational	24.47	+0.55	+10.06
17.	17.	17.	Vertica	Relational, Multi-model	23.80	+0.96	+3.76
18.	↑ 19.	↑ 19.	Amazon Redshift	Relational	22.61	+1.68	+7.43
19.	↑ 20.	↓ 18.	Netezza	Relational	20.84	+0.23	+4.50
20.	↓ 18.	↓ 16.	Firebird	Relational	20.51	-0.88	+0.22
21.	↑ 22.	↑ 24.	dBASE	Relational	16.88	+0.24	+6.75
22.	↓ 21.	22.	Spark SQL	Relational	16.34	-0.41	+3.54
23.	23.	↓ 21.	Impala	Relational, Multi-model	15.07	+0.21	+1.57
24.	24.	↓ 23.	Greenplum	Relational, Multi-model	12.77	+0.28	+2.39
25.	25.	25.	Oracle Essbase	Relational	12.41	+0.71	+4.34
26.	26.	26.	Microsoft Azure SQL Data Warehouse	Relational	9.94	+0.64	+3.59
27.	↑ 30.	↑ 30.	Amazon Aurora	Relational, Multi-model	7.57	+1.14	+2.58

DB-Engines Popularity Rating for DB-Engines

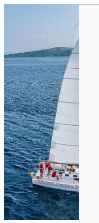
Rank			DBMS	Database Model	Score		
Aug 2019	Jul 2019	Aug 2018			Aug 2019	Jul 2019	Aug 2018
1.	1.	1.	Oracle	Relational, Multi-model	1339.48	+18.22	+27.45
2.	2.	2.	MySQL	Relational, Multi-model	1253.68	+24.16	+46.87
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	1093.18	+2.35	+20.53
4.	4.	4.	PostgreSQL	Relational, Multi-model	481.33	-1.94	+63.83
5.	5.	5.	MongoDB	Document	404.57	-5.36	+53.59
6.	6.	6.	IBM Db2	Relational, Multi-model	172.95	-1.19	-8.89
7.	7.	↑ 8.	Elasticsearch	Search engine, Multi-model	149.08	+0.27	+10.97
8.	8.	↓ 7.	Redis	Key-value, Multi-model	144.08	-0.18	+5.51
9.	9.	9.	Microsoft Access	Relational	135.33	-1.98	+6.24
10.	10.	10.	Cassandra	Wide column	125.21	-1.80	+5.63
11.	11.	11.	SQLite	Relational	122.72	-1.91	+8.99
12.	12.	↑ 13.	Splunk	Search engine	85.88	+0.39	+15.39
13.	13.	↑ 14.	MariaDB	Relational, Multi-model	84.95	+0.52	+16.66
14.	14.	↑ 18.	Hive	Relational	81.80	+0.93	+23.86
15.	15.	↓ 12.	Teradata	Relational, Multi-model	76.64	-1.18	-0.77
16.	16.	↓ 15.	Solr	Search engine	59.12	-0.52	-2.78
17.	17.	↑ 19.	FileMaker	Relational	58.02	+0.12	+1.96
18.	↑ 20.	↑ 21.	Amazon DynamoDB	Multi-model	56.57	+0.15	+4.91
19.	↓ 18.	↓ 17.	HBase	Wide column	56.54	-1.00	-2.27
20.	↓ 19.	↓ 16.	SAP Adaptive Server	Relational	55.86	-0.79	-4.57
21.	21.	↓ 20.	SAP HANA	Relational, Multi-model	55.43	-0.11	+3.50
22.	22.	22.	Neo4j	Graph	48.39	-0.59	+7.47
23.	23.	23.	Couchbase	Document, Multi-model	33.83	+0.12	+0.88
24.	24.	↑ 29.	Microsoft Azure Cosmos DB	Multi-model	29.94	+0.85	+10.41
25.	25.	25.	Microsoft Azure SQL Database	Relational, Multi-model	27.99	-0.67	+1.89
26.	26.	↓ 24.	Memcached	Key-value	27.01	-0.06	-5.90
27.	27.	↓ 26.	Informix	Relational, Multi-model	25.68	-0.18	+0.29
28.	28.	↑ 33.	Google BigQuery	Relational	24.47	+0.55	+10.06
29.	29.	↓ 28.	Vertica	Relational, Multi-model	23.80	+0.96	+3.76
30.	↑ 31.	↑ 32.	Amazon Redshift	Relational	22.61	+1.68	+7.43

Wikibon Big Data Compound Annual Growth Rate Prediction



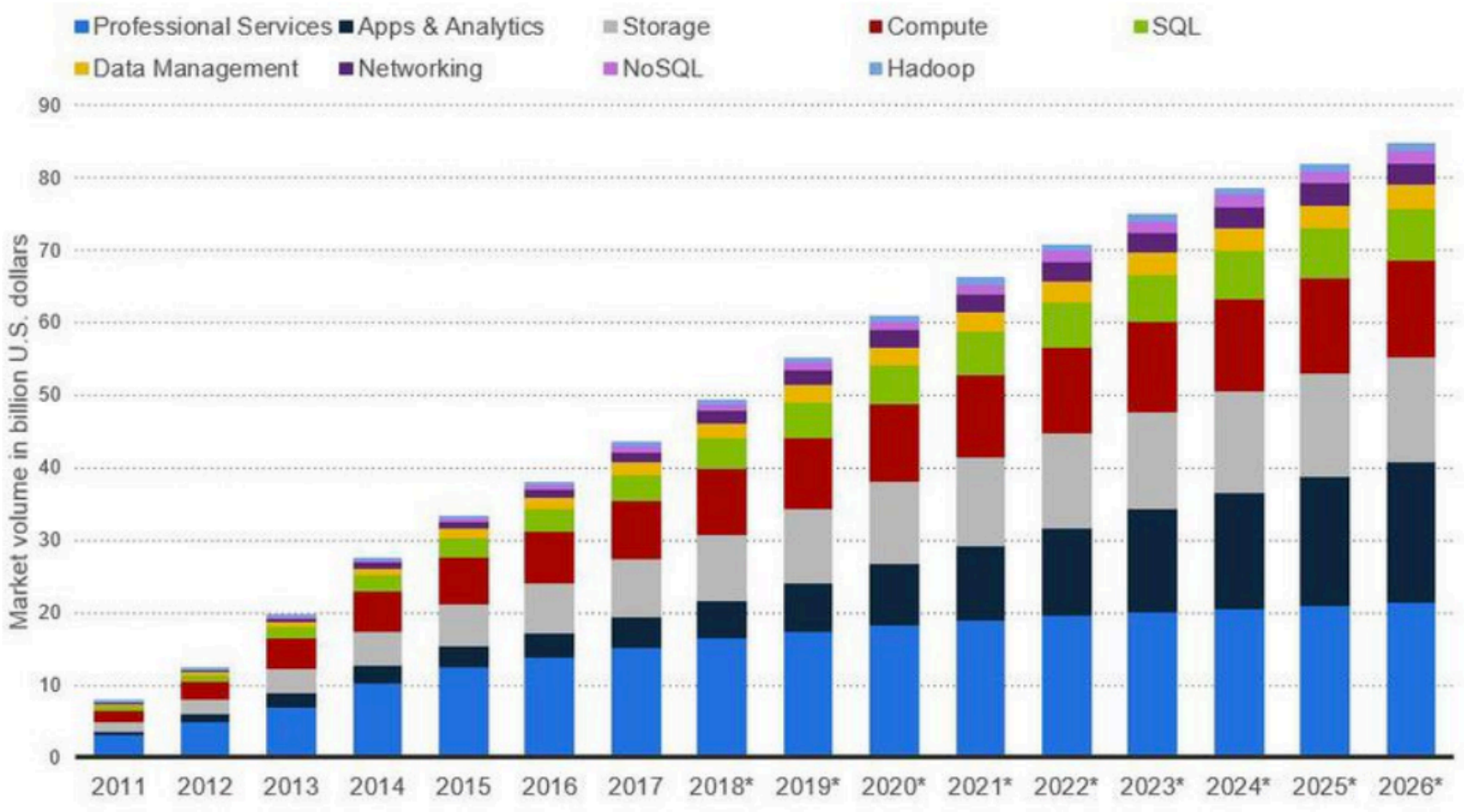
	2016	2017	2018	2019	2020	2021	2022	2023	2024	2025	2026	2027
Software	$8	$11	$14	$17	$20	$24	$27	$31	$34	$38	$42	$46
Hardware	$9	$10	$12	$14	$15	$16	$18	$19	$20	$22	$23	$24
Service	$11	$14	$16	$19	$21	$24	$26	$27	$29	$31	$32	$33
Total Big Data	$28	$35	$42	$49	$56	$64	$70	$77	$84	$90	$96	$103

Figure 1: Worldwide Big Data Hardware, Software, and Services Revenue $B 2016-2027



Big Data Market Worldwide Segment Revenue Forecast 2011-2026

Big Data Market Forecast Worldwide from 2011 to 2026, by segment (in billion U.S. dollars)



statista

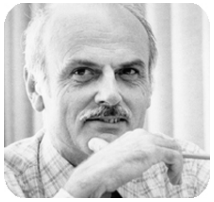
SOURCE: WIKIBON AND REPORTED BY STATISTA.

University Database Example

- Textbook uses a university database to illustrate concepts
  - Course also sometimes uses this example
- Data consists of information about:
  - Students
  - Instructors
  - Courses
  - Etc.
- Application program examples
  - Add new students, instructors, and courses
  - Register students for courses, and generate class rosters
  - Assign grades to students, compute grade point averages (GPA) and generate transcripts

Relational Model

- All the data is stored in various **tables**.
- Example of tabular data in the relational model



Ted Codd
Turing Award 1981

Columns

ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

Rows

(a) The *instructor* table

ID, *name*, *dept_name* and *salary* are the **attributes** of the *instructor* table.

A Sample Relational Database

ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

(a) The *instructor* table

dept_name	*building*	*budget*
Comp. Sci.	Taylor	100000
Biology	Watson	90000
Elec. Eng.	Taylor	85000
Music	Packard	80000
Finance	Painter	120000
History	Painter	50000
Physics	Watson	70000

(b) The *department* table

Levels of Abstraction

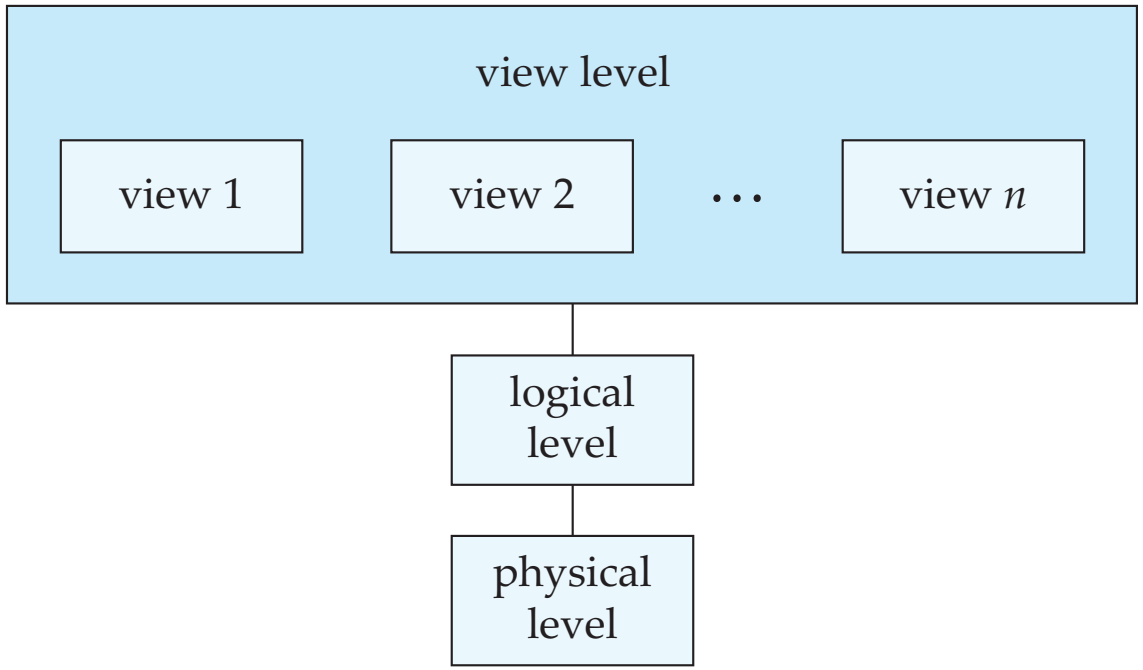
- **Physical level:** Describes how data records for a table (e.g., the *instructor* table) is stored.
- **Logical level:** Describes the data types of the tables stored in the database, and the relationships among the data.

```
type instructor = record
        ID : string;
        name : string;
        dept_name : string;
        salary : integer;
    end;
```

- **View level:** Describes data types of **views** which are defined using queries on one or more tables.
  - Views are like tables, but they are (usually) not stored in the database.
  - Views can also hide information (such as an employee's salary) for security purposes.

View of Data

An architecture for a database system



Schemas and Instances

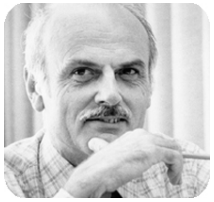
- Similar to types and variables in programming languages
- **Logical Schema** – the overall logical structure of the database
  - Example: The database consists of information about a set of customers and accounts in a bank, and the relationship between them
    - Analogous to type information of a variable in a program
- **Physical schema**– the overall physical structure of the database
- **Instance** – the actual contents of the database at a particular point in time
  - Analogous to the value of a variable

Schemas and Instances (Cont.)

- **Relation Schema:** The logical structure of a relation, which has a name, a data type and (possibly) other "rules of the road" for rows in the relation.
  - Relation is the term use in the Relational Model to refer to a table.
- **Relational Instance** for a Relation Schema**:** A set of rows that follow the "rules of the road" type for that Relation Schema.
- **Database Schema:** A set of Relation Schemas that have different names.
- **Database Instance** for a Database Schema**:** A set of relation instances, with one instance for each of the relations in that Database Schema.

Relational Model

- All the data is stored in various **tables**.
- Example of tabular data in the relational model



Ted Codd
Turing Award 1981

Columns

ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

Rows

(a) The *instructor* table

ID, *name*, *dept_name* and *salary* are the **attributes** of the *instructor* table.

Physical Data Independence

- **Physical Data Independence** – the ability to modify the physical schema without changing the logical schema
  - Applications depend on the logical schema (plus views).
  - The interfaces between the various levels and components should be well defined, so that changes in the physical representation do not affect correctness of the application.
- However, physical representation can have a big impact on query performance. Examples:
  - Row/column storage
    - Data might be stored by rows (records), with all the attribute values that are in a row stored together.
    - Data might be stored by columns instead, with all the values of an attribute (such as dept_name) stored together for every rows.
  - The dept_name attribute in instructors might be indexed (like the index of a book) to make it easy to find all the instructors who are in a particular department.
    - But the same queries can be executed on instructors, whether or not that index exists!

Logical Data Independence

- **Logical Data Independence** – Protects views from changes in the logical schema of the database.
  - Okay to change which tables are in the database, as long as views can be defined correctly to support retrieval and modification operations.
  - Users may not even know whether they're accessing a table or accessing a view!
  - Updates of views are tricky; we'll say more about this later in the quarter.
- Example:
  - We could split the *instructors* table into two tables:
    instructors_sciences and *instructors_humanities*
    and queries on the *instructors* table could instead address an *instructors* view, defined as the union of those two tables.

Data Definition Language (DDL)

- DDL is a specification notation for defining the database schema.

  Example: **create table** *instructor* (
  ID **char**(5),
  name **varchar**(20),
  dept_name **varchar**(20),
  salary **numeric**(8,2))

- DDL compiler generates a set of table templates stored in a ***data dictionary*** (catalog) stored in the database.
- Data dictionary contains metadata (i.e., data about data).
  - Database schema
  - Integrity constraints
    - Primary key (ID uniquely identifies instructors)
  - Authorization
    - Who can access and modify what
  - … and much more

Data Manipulation Language (DML)

- Language for accessing and updating the data organized by the appropriate data model
  - DML also known as query language
- Two classes of languages
  - Pure – used for proving properties about computational power and for optimization
    - Relational Algebra
    - Relational Calculus
      - Tuple relational calculus
      - Domain relational calculus
  - Commercial – used in Proprietary and Open Source Relational Database Management Systems (RDBMS)
    - SQL is the most widely used commercial language

Data Manipulation Language (Cont.)

- Two types of Data Manipulation Languages have existed.
  - **Procedural DML** -- user specifies what data is needed, and how to get that data.
  - **Declarative DML** -- user specifies what data is needed, without specifying how to get that data.
- Declarative DMLs are usually easier to learn and use than Procedural DMLs.
- Declarative DMLs are also referred to as **Non-Procedural** DMLs
- The portion of a DML that involves information retrieval is called a **query** language.

SQL Query Language

- SQL query language is largely Declarative.
  - A query takes as its input one or more tables, and always returns a single table.
- Example: To find the names of all the instructors in the Comp. Sci. dept:

```
SELECT name
FROM instructor
WHERE dept_name = 'Comp. Sci.';
```

Procedural vs. Declarative Languages

- **Procedural program**
  - The program is specified as a sequence of operations to obtain the desired the outcome. I.e., *how* the outcome is to be obtained.
  - E.g., Java, C, …
- **Declarative program**
  - The program specifies *what* is the expected outcome, and not *how* the outcome is to be obtained.
  - E.g., Scheme, Ocaml, …

An Example: Travel from Baskin Engineering, UC Santa Cruz to Soda Hall, UC Berkeley

Declarative (non-procedural):

- Go from Baskin Engineering, at McLaughlin Dr. and Heller Dr. in Santa Cruz, CA, to Soda Hall, at Hearst Ave. and Oxford St. in Berkeley, CA

An Example: Travel from Baskin Engineering, UC Santa Cruz to Soda Hall, UC Berkeley

Procedural (first part)

- Turn left onto Heller Dr,
- Turn left onto Empire Grade
- Continue onto High St
- High St turns right and becomes Storey St,
- Turn left onto King St
- Turn left onto CA-1 S/Mission St
- Continue to follow CA-1 S
- Take the CA-17 exit on the left toward San Jose/Oakland
- Continue onto CA-17 N
- Keep left to continue on I-880 N

An Example: Travel from Baskin Engineering, UC Santa Cruz to Soda Hall, UC Berkeley

Procedural (continued):

- Keep right at the fork, follow signs for CA-24/I-980/Walnut Cr. Continue onto I-980 E
- Keep left to continue on CA-24 E
- Take the exit toward 51st Street
- Turn right onto 52nd St
- Take the 1st left onto Shattuck Ave
- Turn right onto University Ave
- Turn left onto Oxford St
- Turn right onto Hearst Ave
- Turn left; destination will be on the right

Database Access from Application Program

- SQL is NOT a complete programming language; there's lots of functionality that SQL does not have.
  - Loops, input from users, output to displays, network communication
- To perform general computations, **Application Programs** can embed SQL in some higher-level language (such as C/C++, Java, Python).
  - Language extensions that allow embedded SQL.
  - Application program interfaces built using libraries, such as ODBC/JDBC, which allow SQL queries to be sent to a database.
  - Also: Functions/Procedures can be stored and executed inside the database.

Database Design

The process of designing the general structure of the database:

- Logical Design – Deciding on the database schema. Database design requires that we find a "good" collection of relation schemas.
  - Business decision – What attributes should we record in the database?
  - Computer Science decision – What relation schemas should we have and how should the attributes be distributed among the various relation schemas?
- Physical Design – Deciding on the physical layout of the database

Database Administrator

A person who has central control over the DBMS is called a **database administrator (DBA),** whose functions include:

- Schema definition
- Storage structure and access-method definition
- Schema and physical-organization modification
- Granting of authorization for data access
- Routine maintenance
- Periodically backing up the database
- Ensuring that enough free disk space is available for normal operations, and upgrading disk space as required
- Monitoring jobs running on the database and ensuring that performance is not degraded by very expensive tasks submitted by some users

Some of these tasks may now be handled automatically (or semi-automatically)

DBMS Structure (CSE 181)