Stored Programs

Database Applications in Health and Medicine

EMR: Electronic Medical Records

- Patient Demographics
- Doctors
- Insurance
- Medical history
- Allergies
- Procedures / Measurements (BP, Temp, O_2)
- Order Lab Tests Lab Results
- Dr. Notes

Next time you get a physical, notice how much time your doctor spends at a computer terminal!

EHR Functions

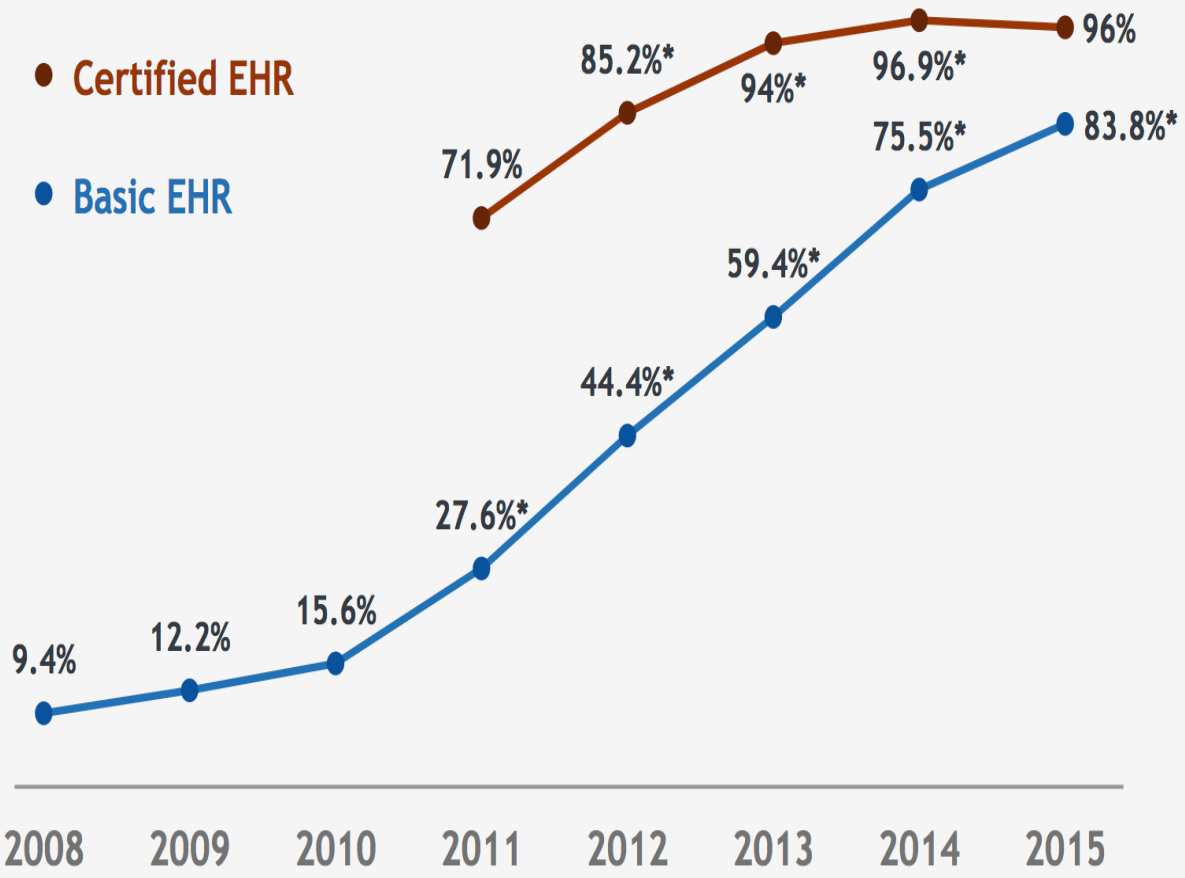
EHR Functions Required	Basic EHR without Clinician Notes	Basic EHR with Clinician Notes	Comprehensive EHR
Electronic Clinical Information			
Patient demographics	•	•	•
Physician notes		•	•
Nursing Assessments		•	•
Problem lists	•	•	•
Medication lists	•	•	•
Discharge Summaries	•	•	•
Advance directives			•
Computerized Provider Order Entry			
Lab reports			•
Radiology tests			•
Medications	•	•	•
Consultation requests			•
Nursing orders			•
Results Management			
View lab reports	•	•	•
View radiology reports	•	•	•
View radiology images			•
View diagnostic test results	•	•	•
View diagnostic test images			•
View consultant reports			•
Decision Support			
Clinical guidelines			•
Clinical reminders			•
Drug allergy results			•
Drug-drug interactions			•
Drug-lab interactions			•
Drug dosing support			•

Adoption of EHR

https://dashboard.healthit.gov

Basic EHR adoption increased while certified EHR adoption remained high

Figure 1: Percent of non-Federal acute care hospitals with adoption of at least a Basic EHR with notes system and possession of a certified EHR: 2008-2015

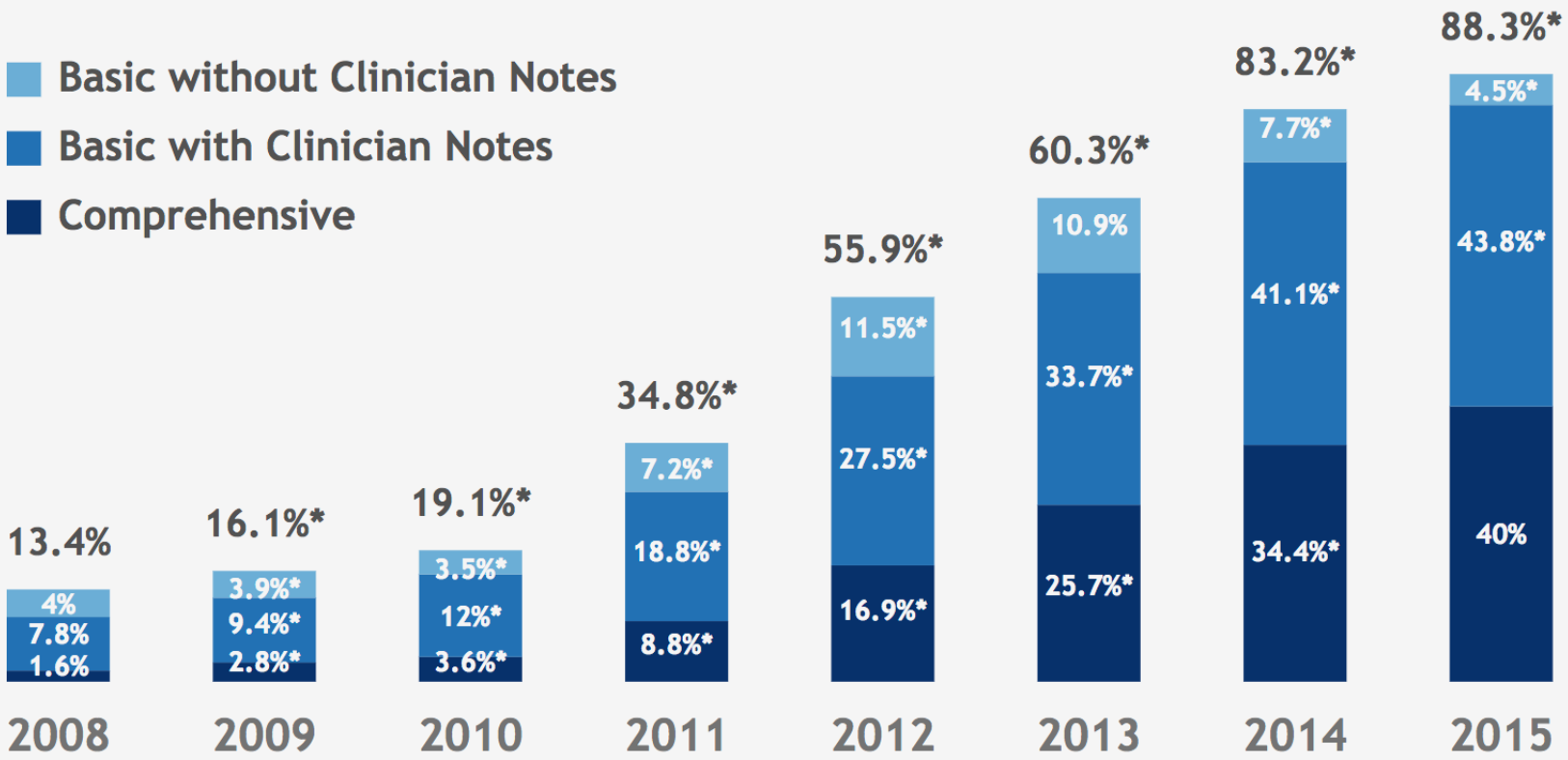


EHR Functions Required	Basic EHR without Clinician Notes	Basic EHR with Clinician Notes	Comprehensive EHR
Electronic Clinical Information			
Patient demographics	•	•	•
Physician notes		•	•
Nursing Assessments		•	•
Problem lists	•	•	•
Medication lists	•	•	•
Discharge Summaries	•	•	•
Advance directives			•
Computerized Provider Order Entry			
Lab reports			•
Radiology tests			•
Medications	•	•	•
Consultation requests			•
Nursing orders			•
Results Management			
View lab reports	•	•	•
View radiology reports	•	•	•
View radiology images			•
View diagnostic test results	•	•	•
View diagnostic test images			•
View consultant reports			•
Decision Support			
Clinical guidelines			•
Clinical reminders			•
Drug allergy results			•
Drug-drug interactions			•
Drug-lab interactions			•
Drug dosing support			•

Adoption of EHR

Trends in EHR adoption show increasing use of advanced functionality

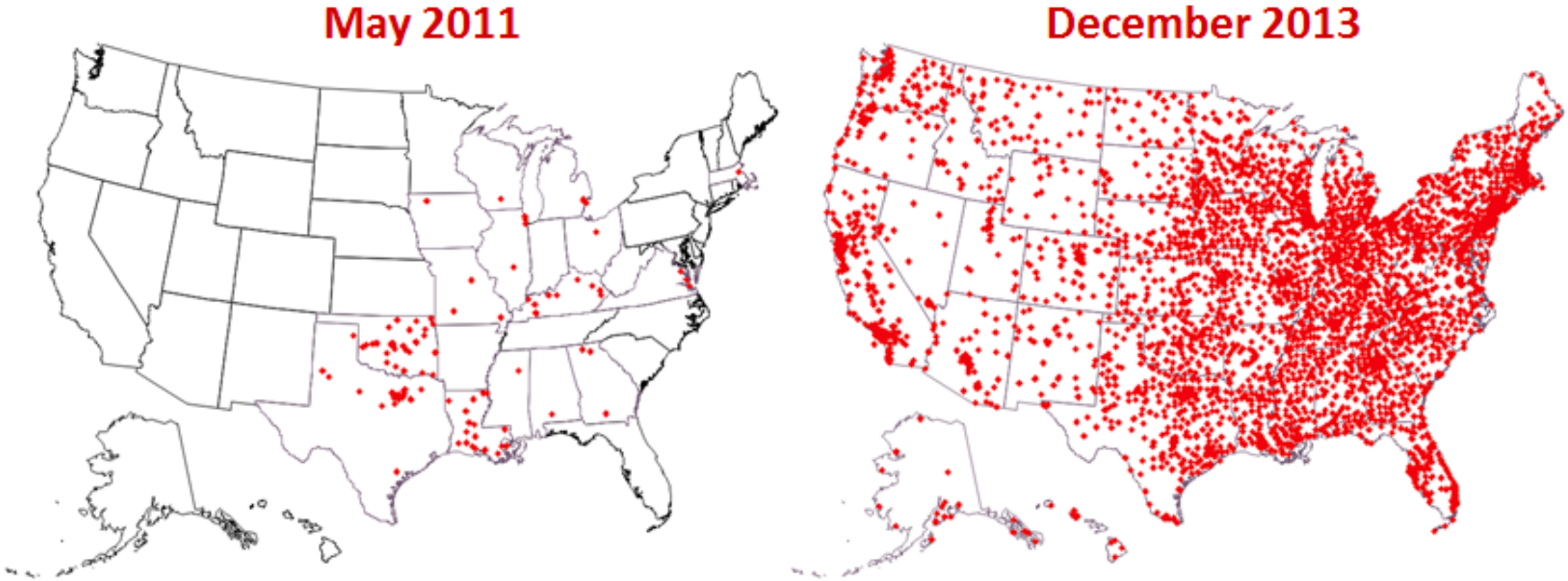
Figure 5: Percent of non-federal acute care hospitals with adoption of EHR systems by level of functionality: 2008 - 2015



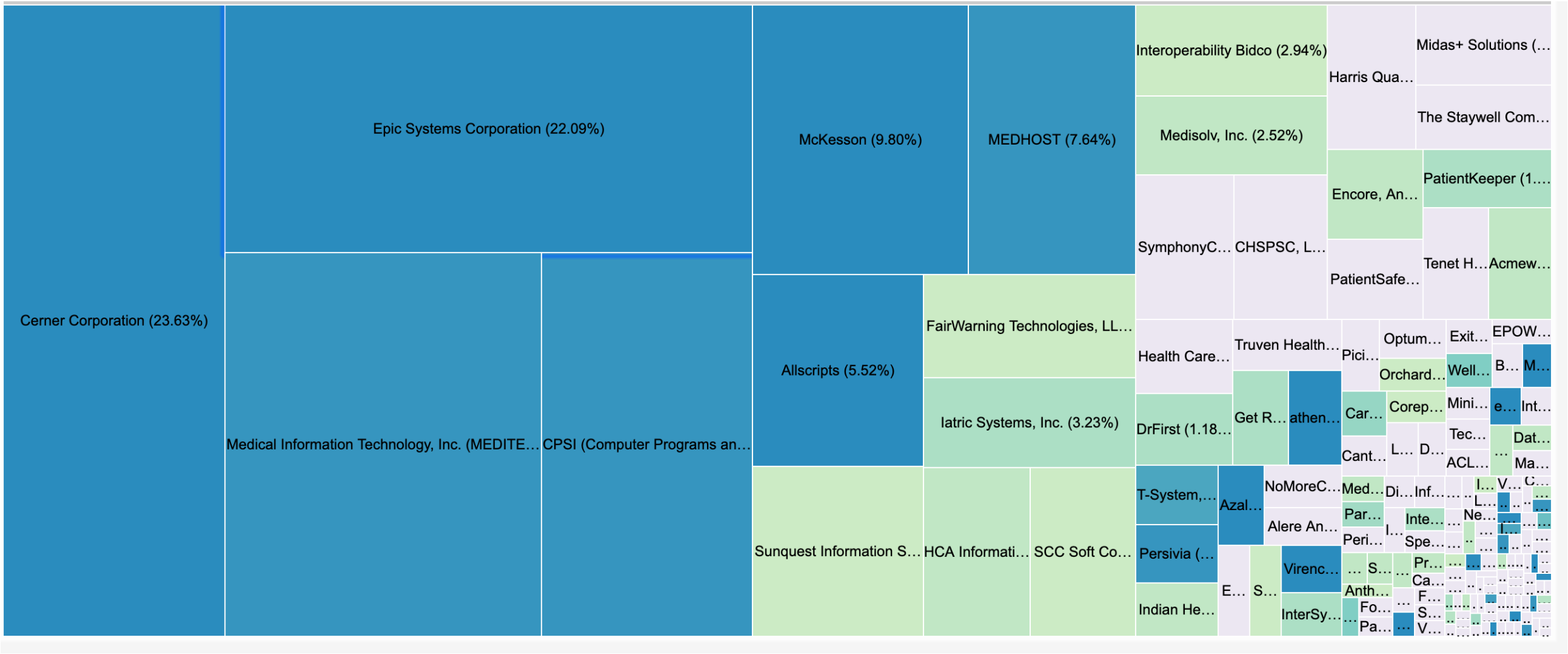
Source:

https://dashboard.healthit.gov

Hospitals receiving inceptive payments for EMR adoption



The EMR Information Technology Landscape



Source: https://dashboard.healthit.gov/quickstats/pages/2015-edition-market-readiness-hospitals-clinicians.php#hospitals

EMR / Database Opportunities

Preventative medicine / Coordinated Care

- Did a patient pick up their prescribed medicine? How responding?
- Do lab results indicate a potential problem requiring a follow-up visit?
- Is the patient being correctly guided through the healthcare system?

Artificial Intelligence

- **Data Mining**:
  - Identifying adverse drug events (pharmacovigilance)
  - Discovering connections between diseases (e.g., diabetes and asthma)
  - Understanding the genetic and environmental factors that contribute to disease risk
  - How does a disease present itself in different patients? (phenotyping)
- **Natural Language Understanding**
- **Clinical Decision Support**

EMR / Database challenges

- Many different EMR systems. There is no universal standard making interoperability very difficult.
  - ✓ How do we obtain a “global” picture of a single patient whose data may be distributed across many healthcare systems?



- EMRs must manage highly complex and heterogeneous datasets: dozens of entities, hundreds of attributes.
  - ✓ Lab Results
  - ✓ Imaging data
  - ✓ EKGs and other time-series data
  - ✓ Doctor’s notes
- EMR integration, processing, and data management is a “big data” challenge

EMR Access Control – Who owns the data?

DOI:10.1145/3331142

Article development led by acmqueue
queue.acm.org

A discussion with David Evans, Richard McDonald, and Terry Coatta.

Access Controls and Healthcare Records: Who Owns the Data?

"Burdened by legacy and fragmented into silos so alien from one another they can scarcely communicate, healthcare recordkeeping has for decades frustrated any and all efforts to unify it."

"While many believe this is information that belongs to the patient, the current reality is that the patient doesn't actually even enjoy access to that data and has little to no control over how it's used."

Source: CACM Staff. 2019. Access controls and healthcare records: who owns the data?. Commun. ACM 62, 7 (June 2019), 41-46.

BBC Sign in | News | Sport | Weather | Shop | Reel | Travel | M

NEWS

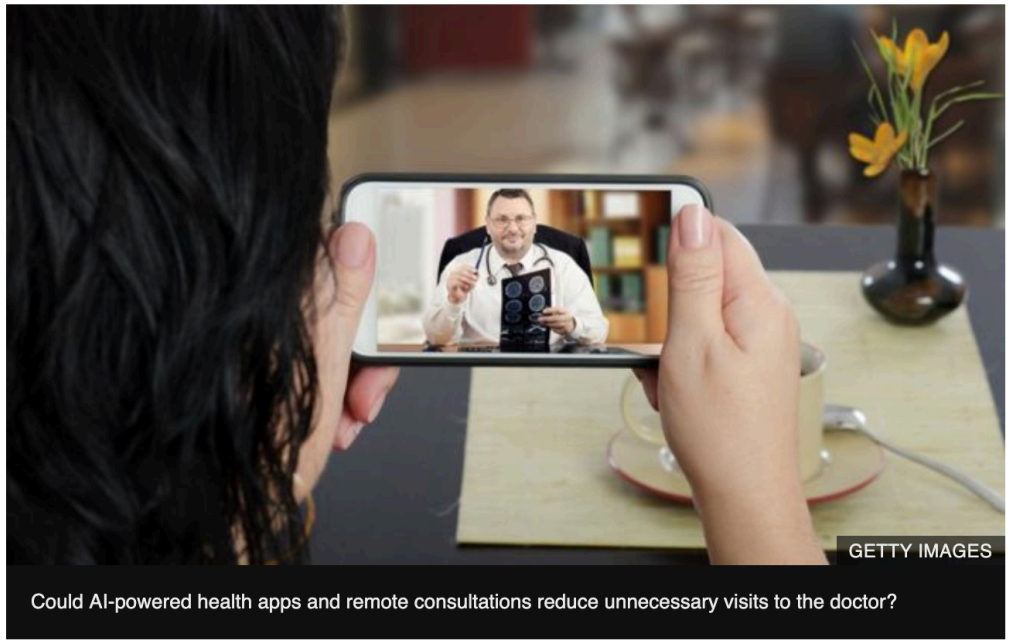
Home | Video | World | US & Canada | UK | Business | Tech | Science | Stories | Enter

Business | Market Data | Global Trade | Companies | Entrepreneurship | Technology of Bus

Are you happy to share your health data to benefit others?

By Matthew Wall
Technology of Business editor

28 June 2019 Share



Could AI-powered health apps and remote consultations reduce unnecessary visits to the doctor?

From automated eye scans to analysing the cries of new-born babies, faster drug development to personalised medicine, artificial intelligence (AI) promises huge advances in the field of healthcare. But major challenges remain.

At the recent AI for Good Summit in Geneva, Switzerland, we were told how AI could speed up the development of new drugs, lead to personalised medicine informed by our genomes, and help diagnose diseases in countries suffering from underdeveloped health services and a chronic shortage of doctors.

But there are two main obstacles preventing access to this utopian destination.

One is that the AI being applied to the world's health problems isn't quite good enough yet. **The other related issue is the lack of good quality digital data - less than 20% of the world's medical data is available in a form that AI machine learning algorithms can ingest and learn from**, the WHO estimates.

Source: https://www.bbc.com/news/business-48784205

Advanced EMR: A doctor's assistant … or a distraction?

- Compute proper dosing for prescription medicine
- Alert the doctor of any dangerous drug interactions
- Make recommendations to improve patient health
- Help with diagnosis

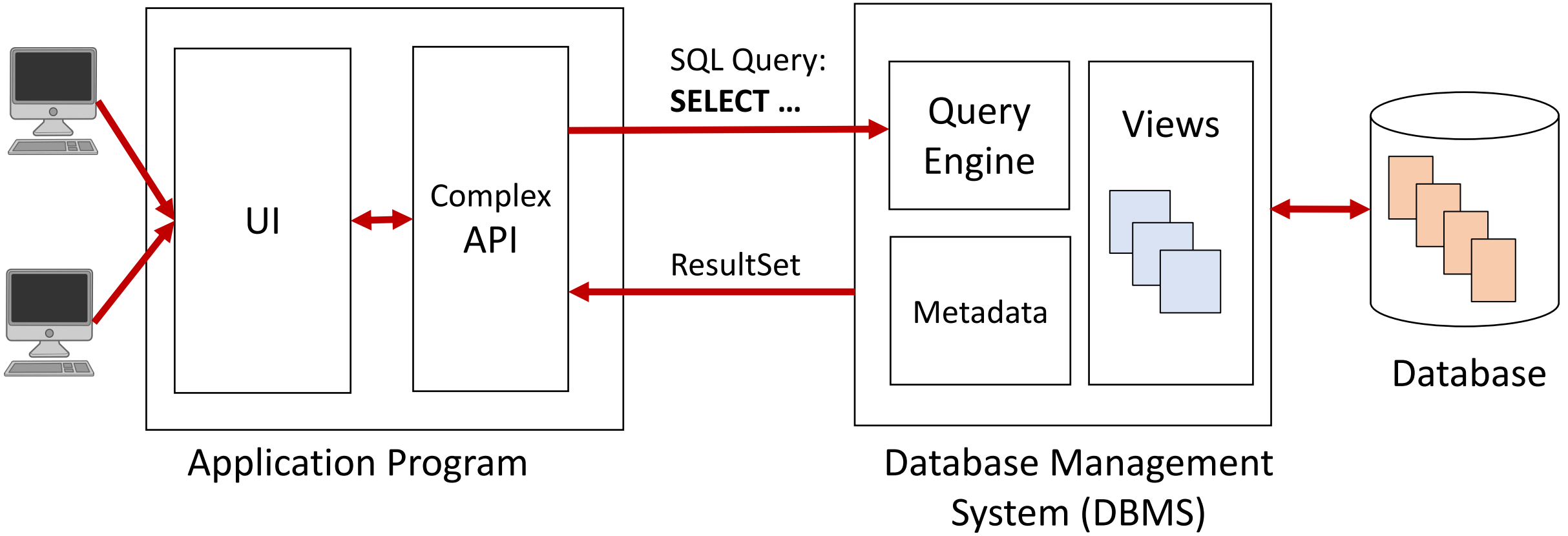
BUT BEWARE!!!!

83% of patients whose physicians were *less engaged with their computers* during the encounter felt the care they received was '**excellent**'.

48% of the patients of physicians with *heavy computer use* during clinical encounters rated the care they received as '**excellent**' on patient experience surveys.

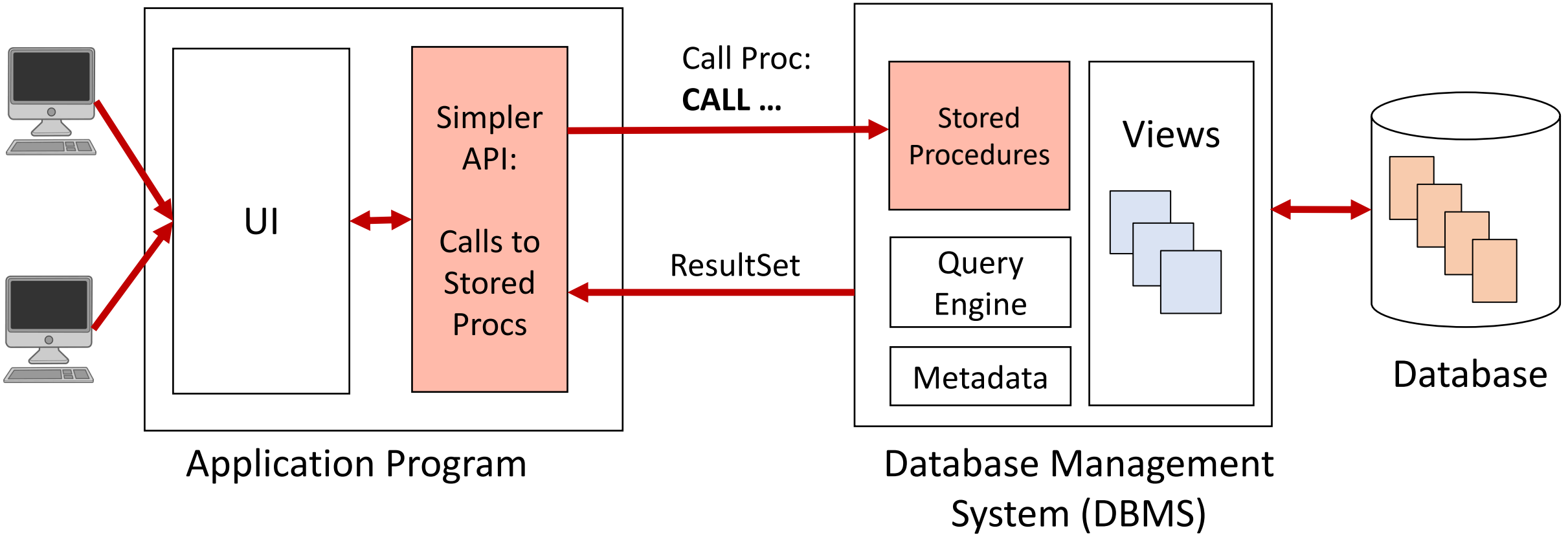
Source: Ratanawongsa N. JAMA Intern Med. 2016;176(**1**):125-128

Database Programming: Model 1 (Complex API)

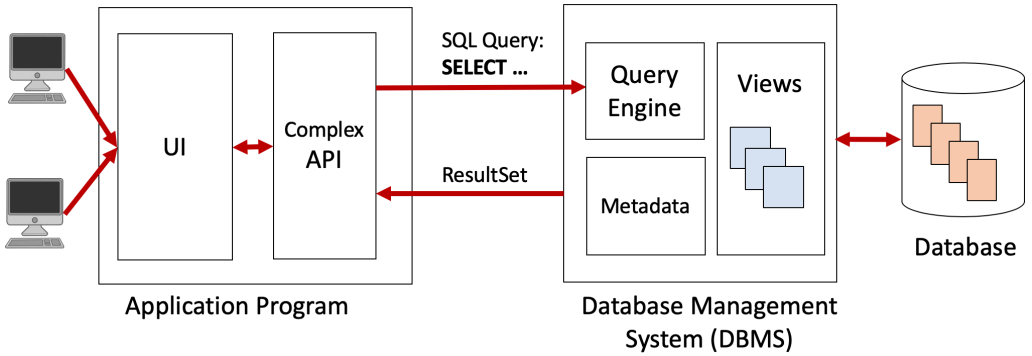


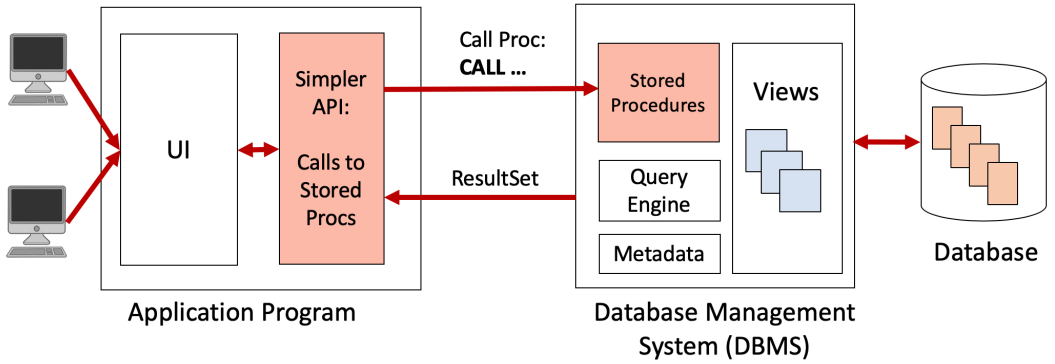
Note: This diagram depicts two-tier client-server.

Programming Model 2: Simple API with Stored Procs.



Tradeoffs between Programming Models





- Application developer implements API and must understand the underlying schema
- The application requires API updates if the underlying database changes. Otherwise the application is broken.
- Compute intensive work is off-loaded from the DBMS to either the client or possibly an intermediate application server.

- Application developer need only make calls to stored procedures, simplifying the API.
- The application requires API doesn't change. However the implementation of the stored procedures might. Still, there is a cleaner separation of concerns.
- DBMS incurs overhead associated with executing program code.
- Since there is less standardization when it comes to stored programs, vendor lock-in is now an issue.

UnitedHealth Group and Optum

UnitedHealth Group Incorporated operates as a diversified health care company in the United States. It operates through four segments: UnitedHealthcare, OptumHealth, OptumInsight, and OptumRx.

<u>2018 Statistics:</u>

- **Revenue**: 226.2 billion (US$)
- **Net Income**: 12.0 billion (US$)
- **Employees**: 300,000
- **Subscribers**: 115 million

Optum Analytics: Aggregator of 90+ million patient records

.
.
.

Staging
(raw data)

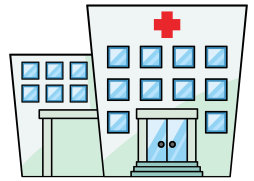
Common
Data
Repository

magic

Extract

Transform

Load



Healthcare
Providers

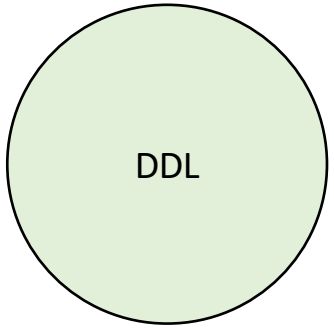
Pharmaceutical
Companies

What questions would you ask if you had the medical records for 90+ million patients?

```
,txn.orig_service_date
,txn.modifier_one
,txn.modifier_two
,txn.modifier_three
,txn.modifier_four
,txn.detail_type
,txn.amount
,txn.performing_prov_id
,case when txn.pat_enc_csn_id = '-1' then null else txn.pat_enc_csn_id end as pat_enc_csn_id
,txn.billing_provider_id
,txn.procedure_quantity
,txn.post_date
,case when txn.localcode = '-1' then null else txn.localcode end as localcode
,localname
,standardcode
,standardname
,case
   when '&grpid' in ('H406239','H704847') and o.pos_type_c <> '-1'
   then o.pos_type_c
 end                                as pos
           from  (select x.*
            ,row_number() over (partition by tx_id,localcode,standardcode order by case when detail_type = '1' then 1 else 0 end desc, orig_service_date desc )

                        from (select coalesce(icd_hx_9.code, icd_curr_9.code)           as mapped_icd9,
                                     coalesce(icd_hx_10.code, icd_curr_10.code)     as mapped_icd10,
                  a.*
                from (select /*+ &parallel_hint */ *
                                              from &src_schm..profbilling_txn partition (Pnumber) bill
                      unpivot  ((localcode) for localname in (dx_one_id,dx_two_id,dx_three_id,dx_four_id,dx_five_id,dx_six_id))
                        where (detail_type is null or detail_type != '10')     and   nvl(tx_id, '-1') != '-1') a
             left outer join &src_schm..ZH_V_EDG_HX_ICD10 icd_hx_10 on (icd_hx_10.dx_id = a.localcode and orig_service_date  between  icd_hx_10.eff_start_date
nd_date)
                                       left outer join &src_schm..zh_edg_curr_icd10 icd_curr_10 on (icd_curr_10.dx_id = a.localcode )
             left outer join &src_schm..ZH_V_EDG_HX_ICD9 icd_hx_9 on (icd_hx_9.dx_id = a.localcode and orig_service_date  between  icd_hx_9.eff_start_date and
te)
                                          left outer join &src_schm..zh_edg_curr_icd9 icd_curr_9 on (icd_curr_9.dx_id = a.localcode )
             )
          unpivot include nulls ((standardcode) for standardname in (mapped_icd9 as 'ICD9',mapped_icd10 as 'ICD10')) x
                           ) txn
 left outer join (select tx_id
                        ,pat_id
                        ,pos_id
                        ,row_number() over (partition by tx_id order by hx_datetime desc nulls last) as rownumber
                   from &src_schm..profbilling_inv partition (Pnumber)
                  where nvl(tx_id, '-1') != '-1'
                   and  nvl(pat_id, '-1') != '-1'
                   and inv_status_c not in('7','4')) inv
         on (txn.tx_id = inv.tx_id and inv.rownumber = 1)
 left outer join &src_schm..zh_orgres o on inv.pos_id = o.pos_id
 txn.rownumber = 1
no rows
```

The Language Hierarchy

Data Definition Language

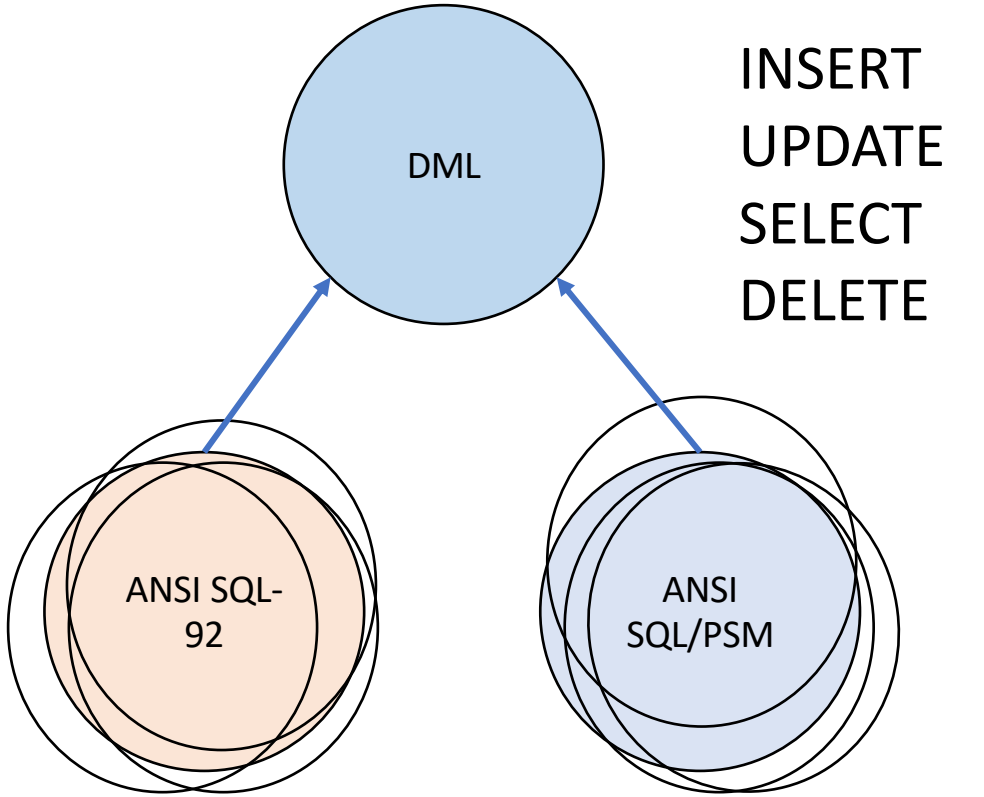


CREATE
DROP
ALTER

- Operations that create or modify the database Schema

- Metadata-focused
- Allows the DBA or user to describe and name entities, attributes, and relationships required for the application plus any associated integrity and security constraints.

Data Manipulation Language



INSERT
UPDATE
SELECT
DELETE

Declarative (4GL)

Procedural (3GL)

Procedural Languages for DB are numerous

Source	Common name	Full name
ANSI/ISO Standard	SQL/PSM	SQL/Persistent Stored Modules
Interbase / Firebird	PSQL	Procedural SQL
IBM DB2	SQL PL	SQL Procedural Language (implements SQL/PSM)
IBM Informix	SPL	Stored Procedural Language
IBM Netezza	NZPLSQL[18]	(based on Postgres PL/pgSQL)
Microsoft / Sybase	T-SQL	Transact-SQL
Mimer SQL	SQL/PSM	SQL/Persistent Stored Module (implements SQL/PSM)
MySQL	SQL/PSM	SQL/Persistent Stored Module (implements SQL/PSM)
MonetDB	SQL/PSM	SQL/Persistent Stored Module (implements SQL/PSM)
NuoDB	SSP	Starkey Stored Procedures
Oracle	PL/SQL	Procedural Language/SQL (based on Ada)
PostgreSQL	PL/pgSQL	Procedural Language/PostgreSQL Structured Query Language (implements SQL/PSM)
Sybase	Watcom-SQL	SQL Anywhere Watcom-SQL Dialect
Teradata	SPL	Stored Procedural Language
SAP	SAP HANA	SQL Script

Source:
https://en.wikipedia.org/wiki/SQL

Four types of stored programs

Type	Purpose
Stored Procedure	Can be called from an application that has access to the database. Carries out queries, transformations, etc.
Stored Function	Can be called from a SQL statement, just like other MySQL functions.
Trigger	Executed in response to an INSERT, UPDATE, or DELETE statement on a specified table.
Event	Executed at a schedule time.

Stored programs

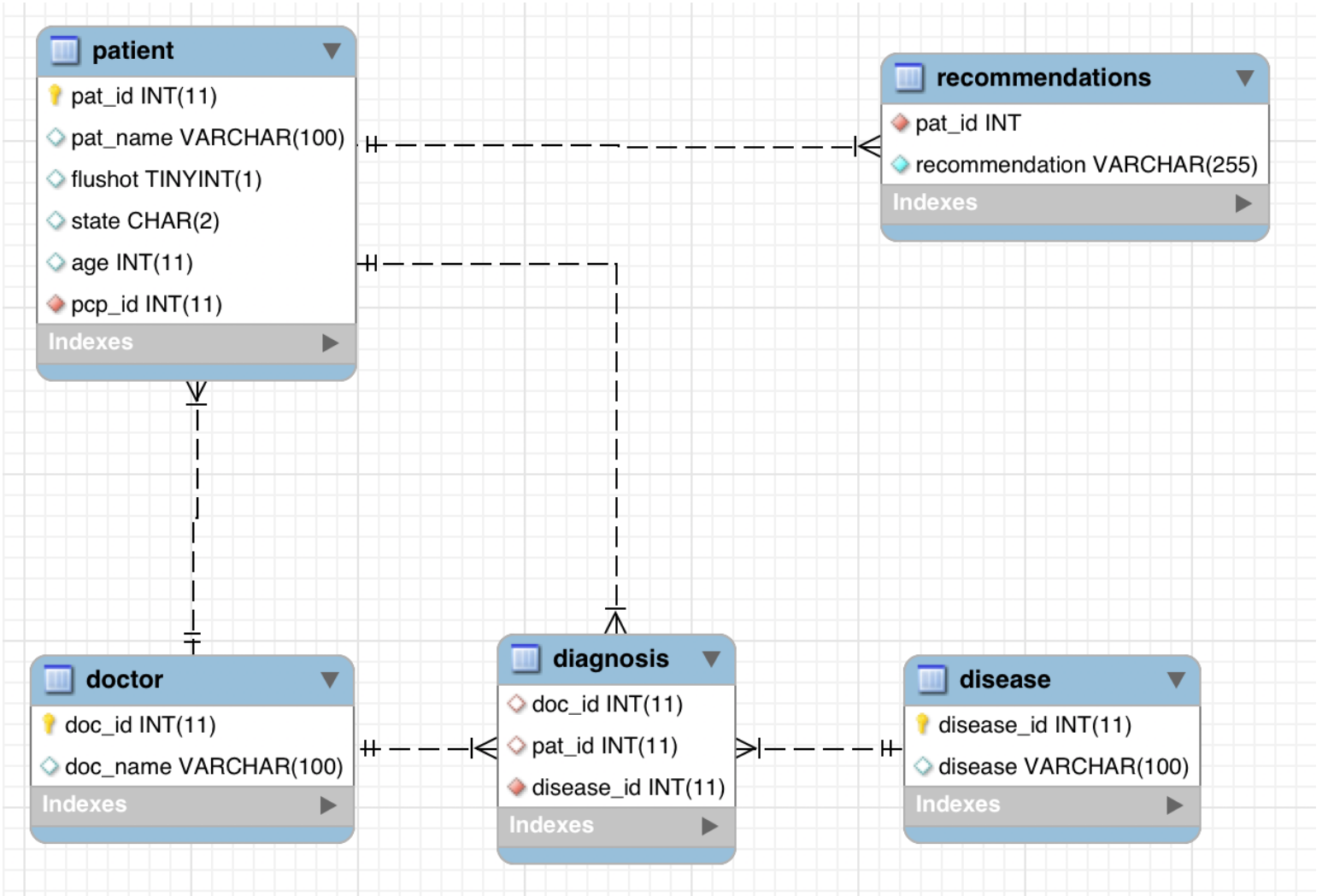
Advantages

- Allow us to embed additional processing methods directly into the database, simplifying client application logic
- SQL is extendable using user-defined functions
- You can trigger actions in response to INSERT, UPDATE, or DELETE actions
- Enable processing operations on a pre-defined schedule

Disadvantages

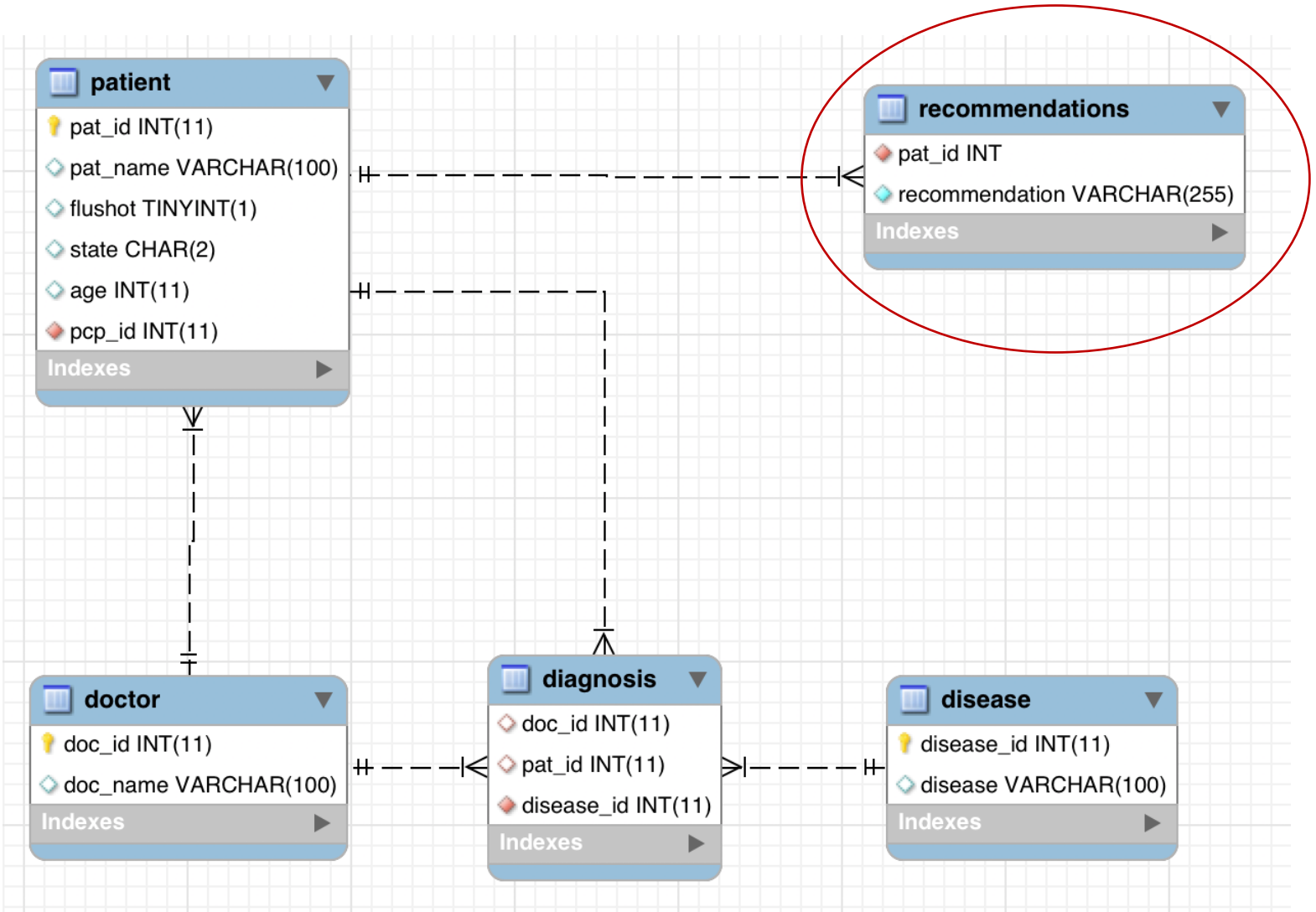
- Vendor lock-in: The stored programming syntax is vendor-specific
- Requires database-specific development expertise
- Stored programs place additional processing load on the DBMS
- Not scalable – the more code you add to your DBMS the slower it gets.

A toy EMR



- Patients receives diagnoses from a doctor on a given date
- Every Patient has a PCP Doctor
- A Diagnosis is by a Doctor, for a Patient about a Disease

An advanced EMR: Coordinated Care



- The rule-based recommendation engine scans a patients health record and catalogs suggestions or issues that the doctor should convey to the patient during the course of the visit.
- Improve care quality.
- Preventative medicine to avoid future healthcare costs.
- An “intelligent” assistant – many opportunities for AI research!

A “Hello World” stored procedure

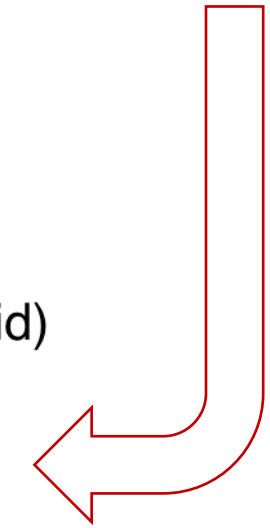
```
DELIMITER //

CREATE PROCEDURE helloWorld()
BEGIN

 SELECT "Hello World!" as Message;

END //

DELIMITER ;


CALL helloWorld(); -- Call the procedure
```

Input parameters

```
DELIMITER //

CREATE PROCEDURE medicalHistory
(
  patientName VARCHAR(50)
)
BEGIN
    select
        pat_name 'patient name',
        pcp.doc_name 'pcp',
        doc.doc_name 'diagnosing doctor',
        disease 'diagnosis',
        diagnosis_date 'date'
    from patient pat
        left join diagnosis diag on (pat.pat_id = diag.pat_id)
        left join disease dis on (diag.disease_id = dis.disease_id)
        left join doctor doc on (diag.doc_id = doc.doc_id)
        join doctor pcp on (pat.pcp_id = pcp.doc_id)
    where pat_name = patientName
    order by date;
END //

DELIMITER ;
```

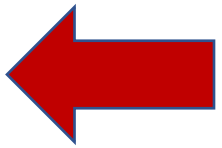
Input parameter

used in query



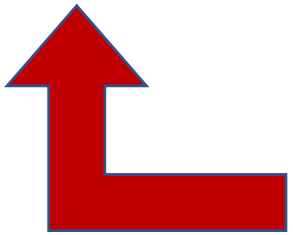
Variables

```
DECLARE numPatients INT;
DECLARE numPatientsHadFluShot INT;
DECLARE pctFluShot FLOAT;

SELECT COUNT(pat_id), SUM(fluShot)
INTO numPatients, numPatientsHadFluShot
FROM patient JOIN doctor ON (patient.pcp_id = doctor.doc_id)
WHERE doc_name = docName;

SET pctFluShot = numPatientsHadFluShot * 100.0 / numPatients;
```



Declaration of variables

Assign through select



Assign through expression

IF…ELSEIF….ELSE

An example of a procedure that might be used as part of a care coordination application.

```
CREATE PROCEDURE fluShotPct
(
   docName VARCHAR(50)
)
BEGIN
   DECLARE numPatients INT;
   DECLARE numPatientsHadFluShot INT;
   DECLARE pctFluShot FLOAT;

   SELECT COUNT(pat_id), SUM(fluShot)
   INTO numPatients, numPatientsHadFluShot
   FROM patient JOIN doctor ON (patient.pcp_id = doctor.doc_id)
   WHERE doc_name = docName;

   SET pctFluShot = numPatientsHadFluShot * 100.0 / numPatients;


   IF pctFluShot < 50 THEN
      SELECT CONCAT('ALERT!  Patients of Dr. ', docName,' need flu shots!, (',pctFluShot,')') as Alert;
   ELSEIF pctFluShot >= 50 THEN
      SELECT CONCAT('Flu shot targets for Dr. ', docName,' have been met (',pctFluShot,')') as Message;
   ELSE
      SELECT CONCAT(docName,' has no primary care patients') as Warning;
   END IF;

END //
```

- Calculate % of patients having had their flu shots
- Issue an alert if percentage < 50%

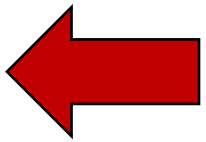
Cursors – Taking an action on every row

```
CREATE PROCEDURE procedureName()
BEGIN

   DECLARE row_not_found  TINYINT DEFAULT FALSE;

   DECLARE table_cursor CURSOR FOR
      SELECT '<table select query>';

   DECLARE CONTINUE HANDLER FOR NOT FOUND
      SET row_not_found = TRUE;

   OPEN table_cursor;

   FETCH table_cursor INTO x,y,z; -- read first row
   WHILE row_not_found = FALSE DO
      -- DO ROW-LEVEL LOGIC HERE
      FETCH table_cursor INTO x,y,z; -- read next row
   END WHILE;

   CLOSE table_cursor;

END //
```

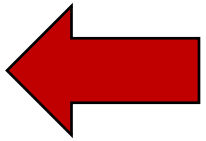
A basic template for a procedure that uses cursors

Identify patients that haven't had a flu shot

```
CREATE PROCEDURE recommendFluShots()
BEGIN

   DECLARE pat_id_var INT;
   DECLARE flushot_var TINYINT;
   DECLARE row_not_found  TINYINT DEFAULT FALSE;
   DECLARE update_count    INT DEFAULT 0;
   DECLARE row_count INT DEFAULT 0;

   DECLARE patient_cursor CURSOR FOR
      SELECT pat_id, flushot from patient;

   DECLARE CONTINUE HANDLER FOR NOT FOUND
      SET row_not_found = TRUE;

   OPEN patient_cursor;

   FETCH patient_cursor INTO pat_id_var, flushot_var; -- read first row
   WHILE row_not_found = FALSE DO

      IF flushot_var = FALSE THEN
         INSERT INTO recommendations
         VALUES (pat_id_var, 'Ask patient if he/she has had a flushot');
         SET update_count = update_count + 1;
      END IF;

      SET row_count = row_count + 1;
      FETCH patient_cursor INTO pat_id_var, flushot_var; -- read next row
   END WHILE;

   CLOSE patient_cursor;

   select CONCAT(row_count, ' patients scanned. ', update_count,' recommendations added') as status;

END //
```



Define the cursor



Did patient get a flushot?

Functions – Extending the SQL language

```
DELIMITER //

CREATE FUNCTION body_mass_index
(
    weight_lbs FLOAT,
    height_inches FLOAT
)
RETURNS FLOAT
DETERMINISTIC
BEGIN
    DECLARE weight_kg FLOAT;
    DECLARE height_meters FLOAT;
    DECLARE bmi FLOAT;

    SET weight_kg = weight_lbs * 0.4536; -- weight in kilos
    SET height_meters = height_inches * 2.54 / 100.0; -- height in meters
    SET bmi = weight_kg / (height_meters * height_meters); -- bmi = weight / height^2 [kg / m^2]
    RETURN bmi;
END //

DELIMITER ;
```

BMI for every patient

```
select
    pat_name,
    weight,
    height,
    round(body_mass_index(weight, height),2) bmi
from patient;
```

pat_name	weight	height	bmi
Smith	240	72	32.55
Jones	160	71.5	22.00
Johnson	130	65	21.63
Phillips	116	61	21.92
Lee	240	73	31.66
Marcus	150	70	21.52
Samuels	130	68	19.77
van Dyke	165	71	23.01

Recommend diet and exercise

```
    .
    .
    .
FETCH patient_cursor INTO pat_id_var, weight_var, height_var; -- read first row
WHILE row_not_found = FALSE DO

    IF body_mass_index(weight_var, height_var) >= 25.0 THEN
        INSERT INTO recommendations
        VALUES (pat_id_var, 'Encourage patient to diet and exercise');
        SET update_count = update_count + 1;
    END IF;

    SET row_count = row_count + 1;
    FETCH patient_cursor INTO pat_id_var, weight_var, height_var; -- read next row
END WHILE;


        .
        .
```

Recommend diet and exercise

pat_name	bmi	flushot	message
Lee	31.66	0	Ask patient if he/she has had a flushot
Lee	31.66	0	Encourage patient to diet and exercise
Marcus	21.52	0	Ask patient if he/she has had a flushot
Samuels	19.77	0	Ask patient if he/she has had a flushot
Smith	32.55	1	Encourage patient to diet and exercise
van Dyke	23.01	0	Ask patient if he/she has had a flushot

MySQL Triggers

- A **trigger** is a named block of code that is configured to automatically run or *fire* in response to an INSERT, UPDATE, or DELETE command.
- Introduced in MySQL 5.0
- Often used to enhance data integrity

Trigger Syntax

```
CREATE TRIGGER trigger_name
   {BEFORE|AFTER} {INSERT|UPDATE|DELETE} ON table_name
   FOR EACH ROW
sql_block
```

- Triggers can fire either before or after the event

- In MySQL triggers are defined for a particular table.
  Each modified row separately fires the trigger.

- **OLD**.*column* = original column value of the changing row
  NEW.*column* = new column value of the changing row

Data Consistency using Triggers

```
-- Check consistency between sex and is_pregnant fields
-- This is a typical usecase of triggers - inforcing special kinds of constraints.

DROP TRIGGER IF EXISTS patient_pregnancy_check;

DELIMITER //

CREATE TRIGGER patient_pregnancy_check
    BEFORE UPDATE ON patient
    FOR EACH ROW
BEGIN
    IF (NEW.is_pregnant = TRUE AND NEW.sex = 'M') THEN
            SIGNAL SQLSTATE 'HY000'
                SET MESSAGE_TEXT = 'Male patient cannot be pregnant!';
    END IF;
END; //

DELIMITER ;
```

MySQL Error Messages

The message displayed contains three types of information:

- A numeric error code (`1146`). This number is MySQL-specific and is not portable to other database systems.
- A five-character SQLSTATE value (`'42S02'`). The values are taken from ANSI SQL and ODBC and are more standardized. Not all MySQL error numbers have corresponding SQLSTATE values. In these cases, `'HY000'` (general error) is used.
- A message string that provides a textual description of the error.

From: https://dev.mysql.com/doc/refman/5.7/en/error-messages-server.html

Commonly used MySQL error codes

Error code	SQLSTATE code	Description
1329	02000	Occurs when a program attempts to fetch data from a row that doesn't exist.
1062	23000	Occurs when a program attempts to store duplicate values in a column that has a unique constraint.
1048	23000	Occurs when a program attempts to insert a NULL value into a column that doesn't accept NULL values.
1216	23000	Occurs when a program attempts to add or update a child row but can't because of a foreign key constraint.
1217	23000	Occurs when a program attempts to delete or update a parent row but can't because of a foreign key constraint.

Notice that SQLSTATE codes are more general than MySQL codes.

Built-in named conditions

Named condition	Description
NOT FOUND	Occurs when a program attempts to use a FETCH statement or a SELECT statement to retrieve data and no data is found.
SQLEXCEPTION	Occurs when any error condition other than the NOT FOUND condition occurs.
SQLWARNING	Occurs when any error condition other than the NOT FOUND condition occurs or when any warning messages occur.

SIGNAL and error

The syntax of the SIGNAL statement

```
SIGNAL SQLSTATE [VALUE] sqlstate_value
[SET MESSAGE_TEXT = message[, MYSQL_ERRNO = mysql_error_number]]
```

SIGNAL SQLSTATE 'HY000'
SET MESSAGE_TEXT = 'Male patient cannot be pregnant!';

HY000 indicates a “General Error”
HY000 is used for MySQL codes that don’t otherwise have an associated SQLSTATE

Events

```
set global event_scheduler = on; -- REQUIRES ROOT !

-- we create an event that invokes the recommendation procedures
-- every 5 seconds:

delimiter //

create event recommendation_engine
on schedule every 5 second
do begin
  call recommendFluShots;
  call recommendDietAndExercise;
end //

delimiter ;
```

Stored programs - Revisited

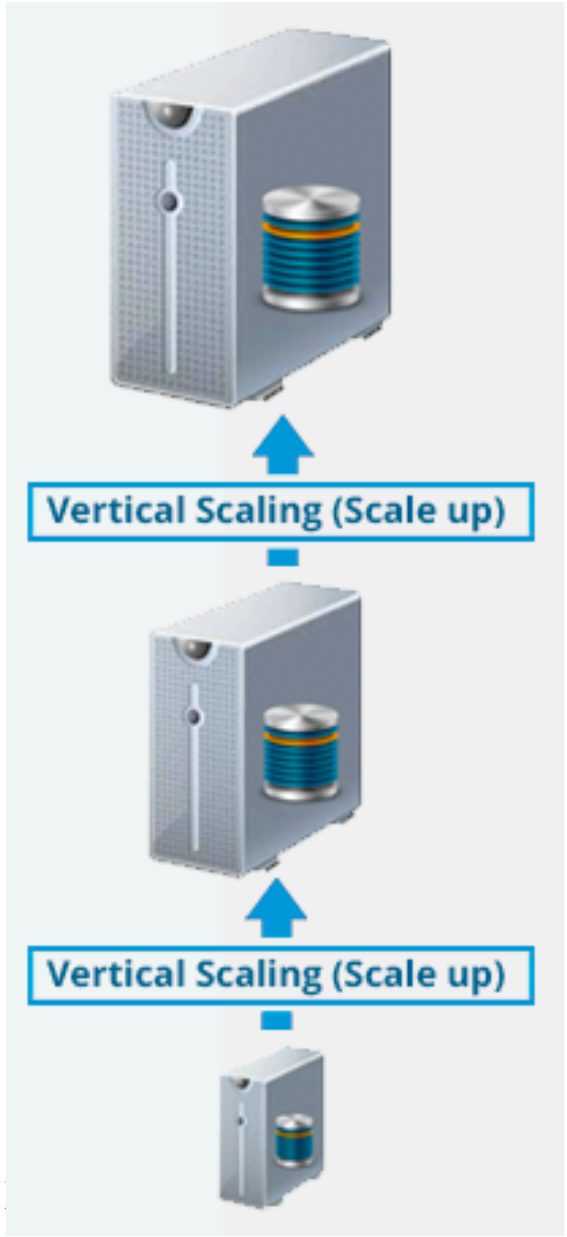
Advantages

- Allow us to embed additional processing methods directly into the database, simplifying client application logic
- SQL is extendable using user-defined functions
- You can trigger actions in response to INSERT, UPDATE, or DELETE actions
- Enable processing operations on a pre-defined schedule

Disadvantages

- Vendor lock-in: The stored programming syntax is vendor-specific
- Requires database-specific development expertise
- Stored programs place additional processing load on the DBMS
- Not scalable – the more code you add to your DBMS the slower it gets.

Scaling Up: Add CPU, Memory, Storage



ADVANTAGES

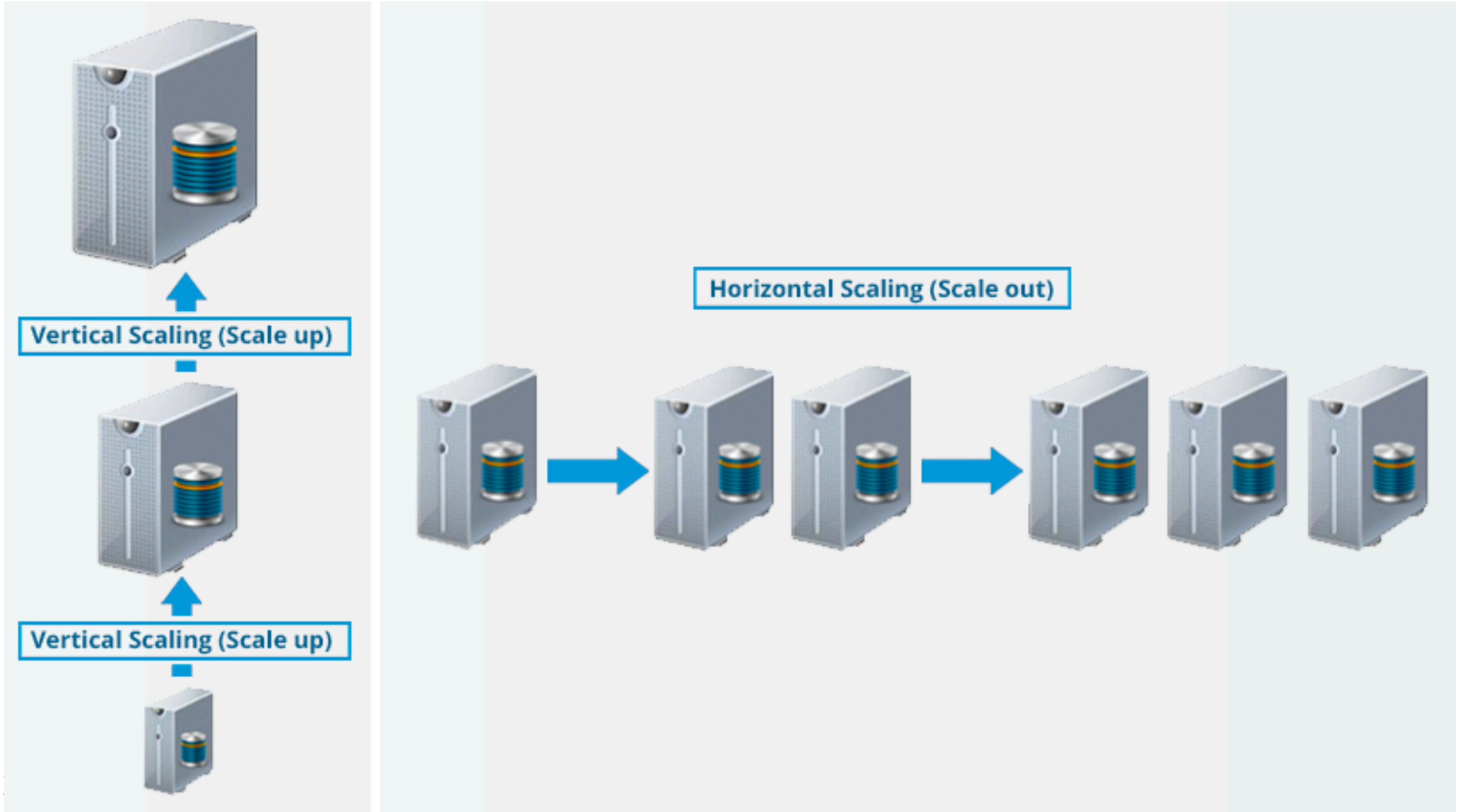
- Leveraging a platform you already know
- Software migration is easier
- Administration is easier
- Relatively easy to manage: add more RAM, CPU or buy a new server if necessary
- Cooling overhead is lower
- Networking hardware overhead is lower

DISADVANTAGES

- Exponential server cost
- Vendor lock-in
- Single point of failure
- Limits on available hardware expansion

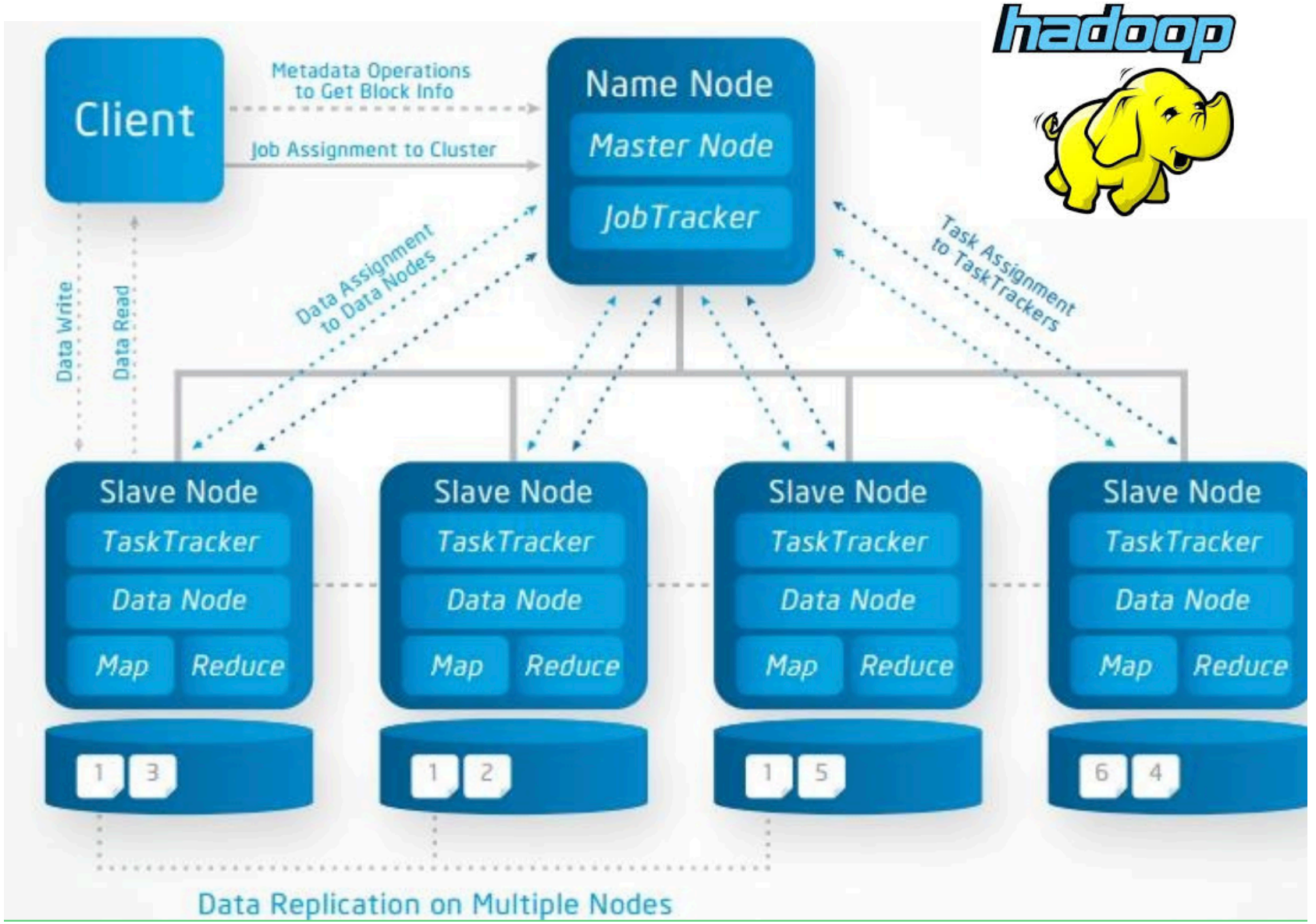
Once you are in the realm of "big data" you need another solution.

Scaling Out: Commodity Servers / Distributed Computing



Hadoop for Big Data applications

- Hadoop is an open-source distributed computing and data processing framework that runs on a cluster of commodity hardware.
- Computing model base on **map-reduce** which divides up large problems into chunks that are distributed to different processing nodes (map) and then aggregates the results (reduce).
- Distributed storage via **HDFS: The Hadoop Distributed File System** which sits on top of each node's local file system and provides a global and fault-tolerant view of the data.



Sample Hadoop Code: Word Count

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class WordCount {

  public static class TokenizerMapper
       extends Mapper<Object, Text, Text, IntWritable>{

    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

    public void map(Object key, Text value, Context context
                    ) throws IOException, InterruptedException {
      StringTokenizer itr = new StringTokenizer(value.toString());
      while (itr.hasMoreTokens()) {
        word.set(itr.nextToken());
        context.write(word, one);
      }
    }
  }

  public static class IntSumReducer
       extends Reducer<Text,IntWritable,Text,IntWritable> {
    private IntWritable result = new IntWritable();

    public void reduce(Text key, Iterable<IntWritable> values,
                       Context context
                       ) throws IOException, InterruptedException {
      int sum = 0;
      for (IntWritable val : values) {
        sum += val.get();
      }
      result.set(sum);
      context.write(key, result);
    }
  }

  public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "word count");
    job.setJarByClass(WordCount.class);
    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(IntSumReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

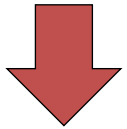
Library imports

Mapper

Reducer

Job setup and submission

WORD COUNT
“to be or not to be”

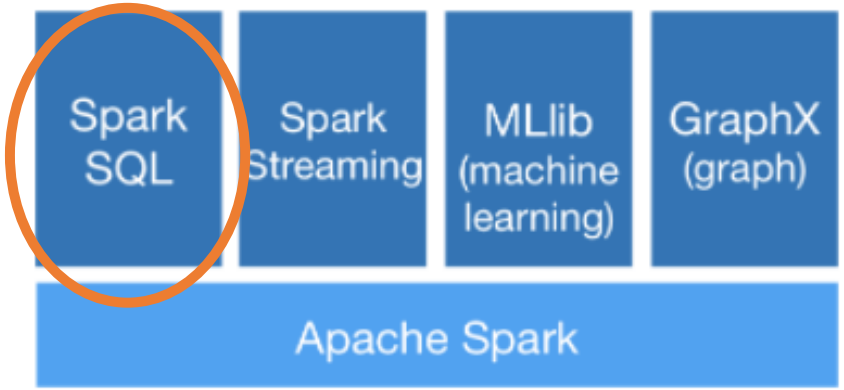


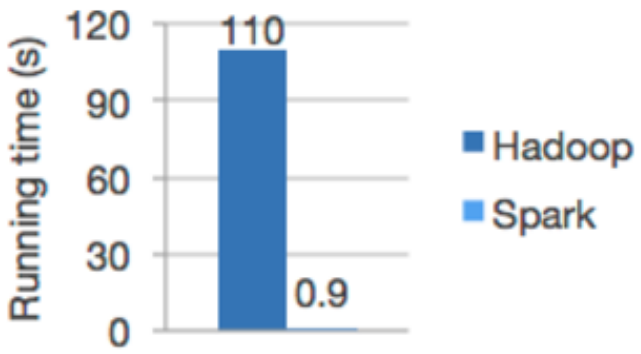
to → 2
be → 2
or → 1
not → 1

What is Apache Spark?

- **Apache Spark** is a fast and general engine for large-scale data processing that runs on a Hadoop cluster.

- With spark, it is *much easier* to write programs, including ETL data processing programs, that leverage the parallel processing capabilities of Hadoop.





Logistic regression in Hadoop and Spark

Hadoop

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class WordCount {

  public static class TokenizerMapper
       extends Mapper<Object, Text, Text, IntWritable>{

    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

    public void map(Object key, Text value, Context context
                    ) throws IOException, InterruptedException {
      StringTokenizer itr = new StringTokenizer(value.toString());
      while (itr.hasMoreTokens()) {
        word.set(itr.nextToken());
        context.write(word, one);
      }
    }
  }

  public static class IntSumReducer
       extends Reducer<Text,IntWritable,Text,IntWritable> {
    private IntWritable result = new IntWritable();

    public void reduce(Text key, Iterable<IntWritable> values,
                       Context context
                       ) throws IOException, InterruptedException {
      int sum = 0;
      for (IntWritable val : values) {
        sum += val.get();
      }
      result.set(sum);
      context.write(key, result);
    }
  }

  public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "word count");
    job.setJarByClass(WordCount.class);
    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(IntSumReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

Spark

```
val textFile = sc.textFile("hdfs://...")
val counts = textFile.flatMap(line => line.split(" "))
                 .map(word => (word, 1))
                 .reduceByKey(_ + _)
```

WORD COUNT

“to be or not to be”

to → 2

be → 2

or → 1

not → 1