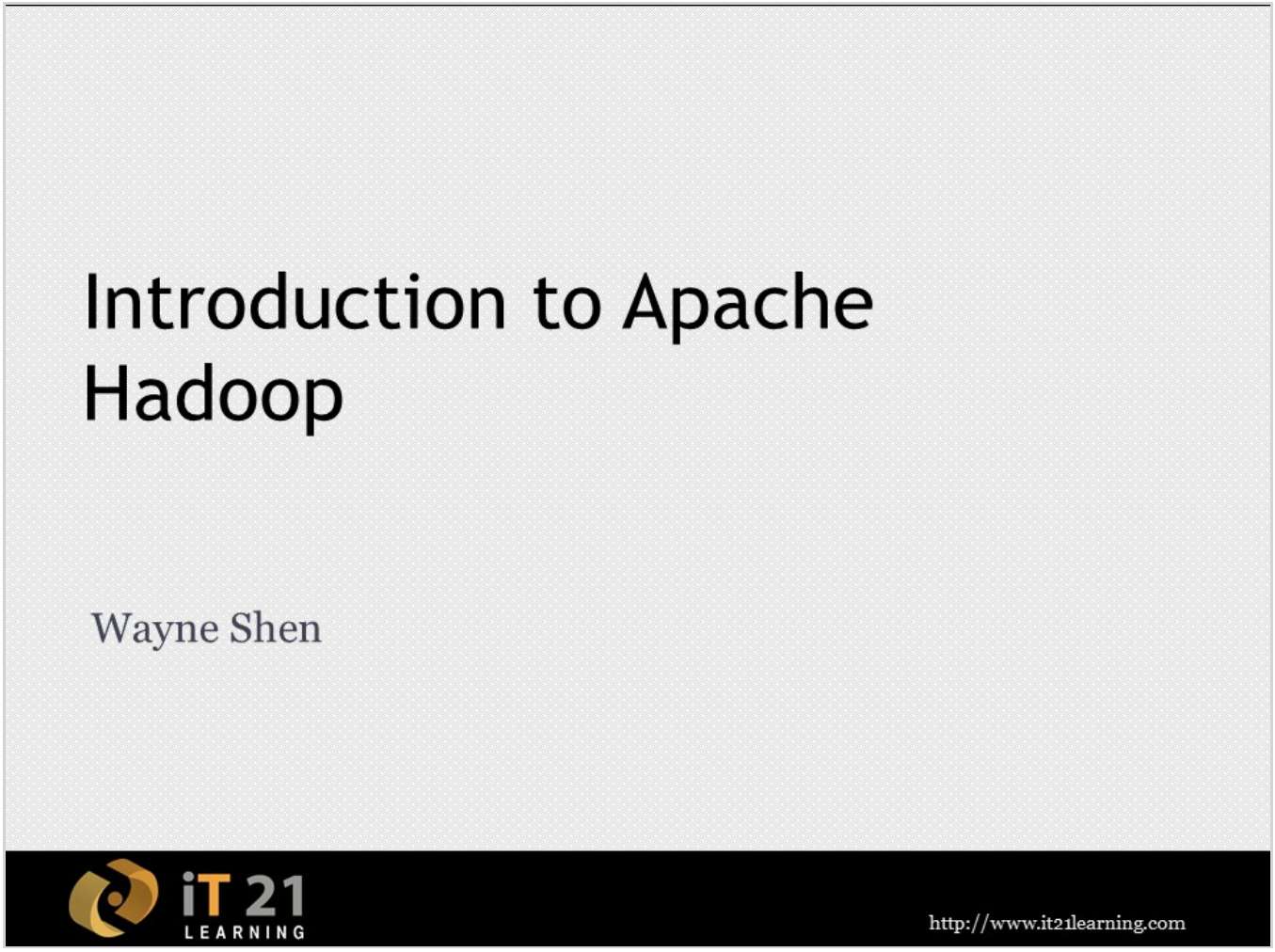
Mastering Hadoop, Spark & HBase for Big Data Solution

Will Du

IT21 Learning Inc.

Introducing Apache Hadoop

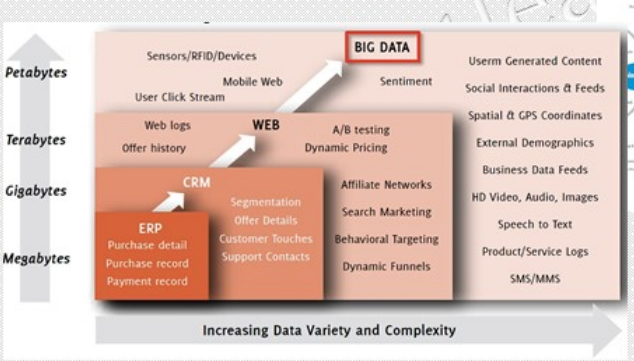

Introduction to Apache Hadoop
Wayne Shen
iT 21 LEARNING
http://www.it21learning.com

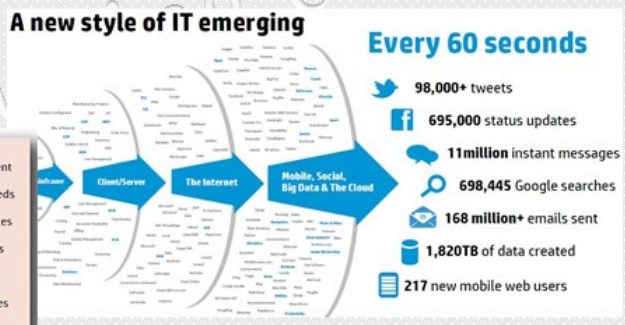
1

iT 21 LEARNING http://www.it21learning.com

What is Big Data?

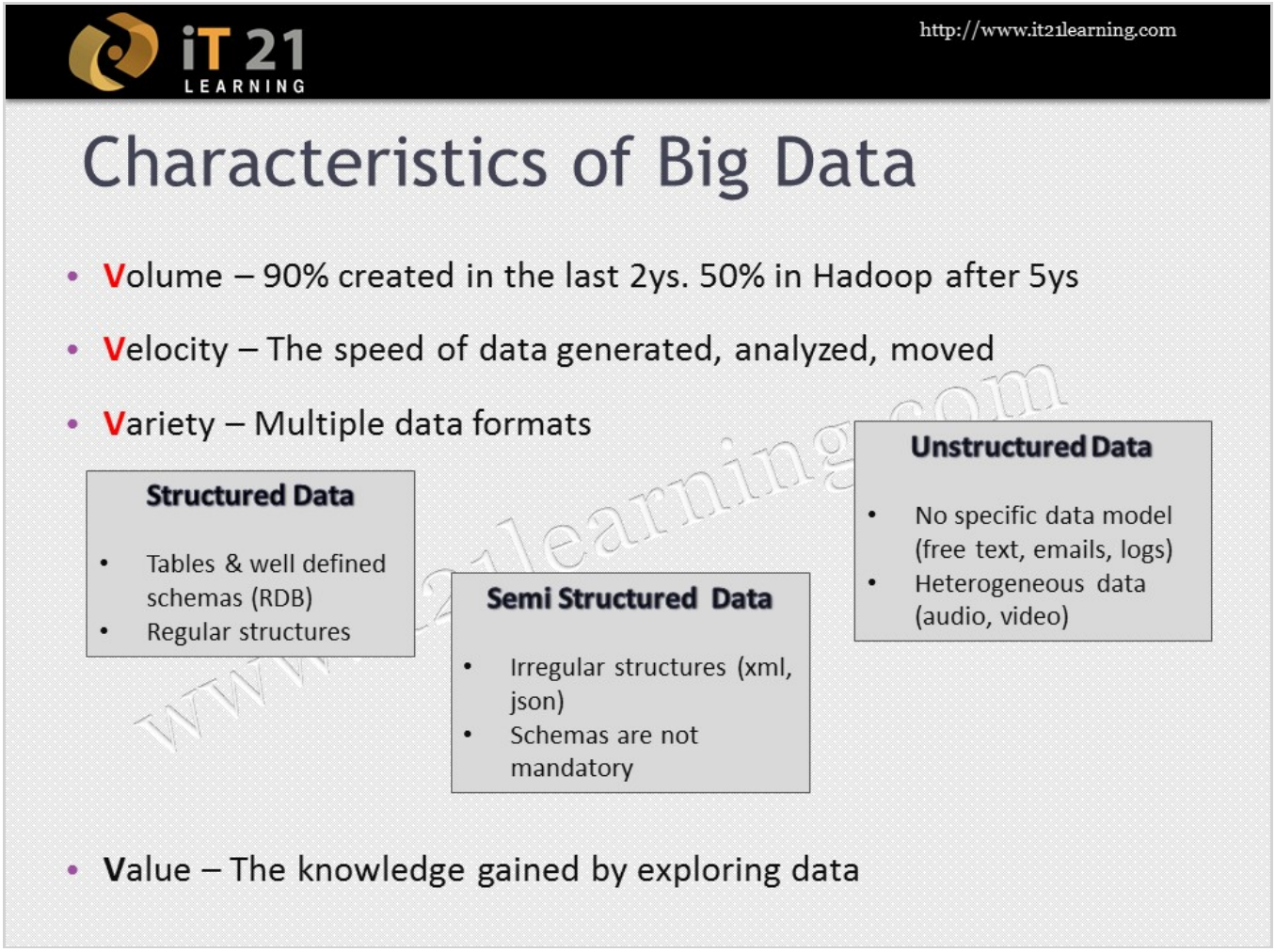
- Big Data is a term that describes **large volumes** of high **velocity**, **complex** and **variable** data that require advanced techniques and technologies to enable the capture, storage, distribution, management, and analysis of the information.





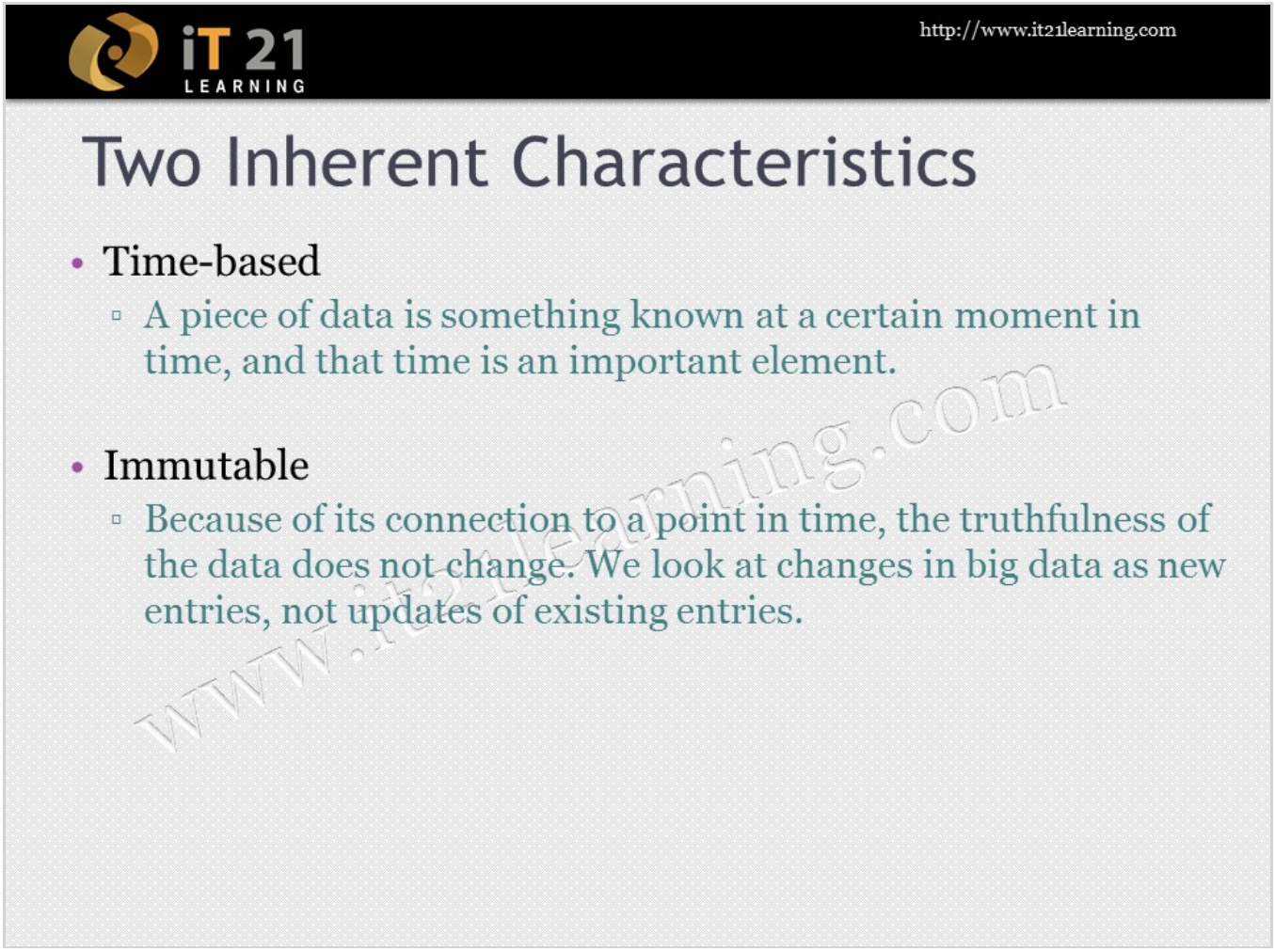
2

Characteristics of Big Data

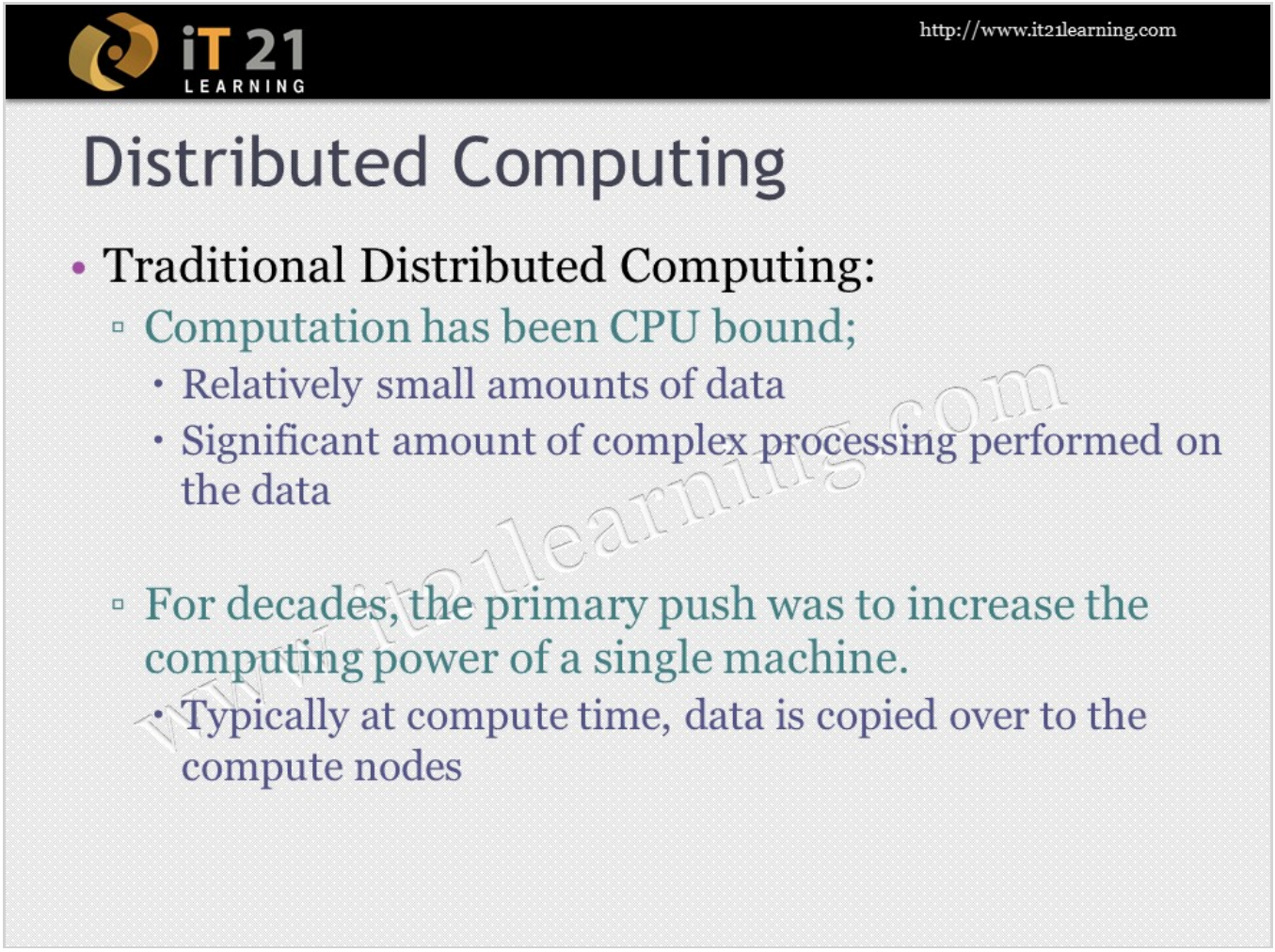


3

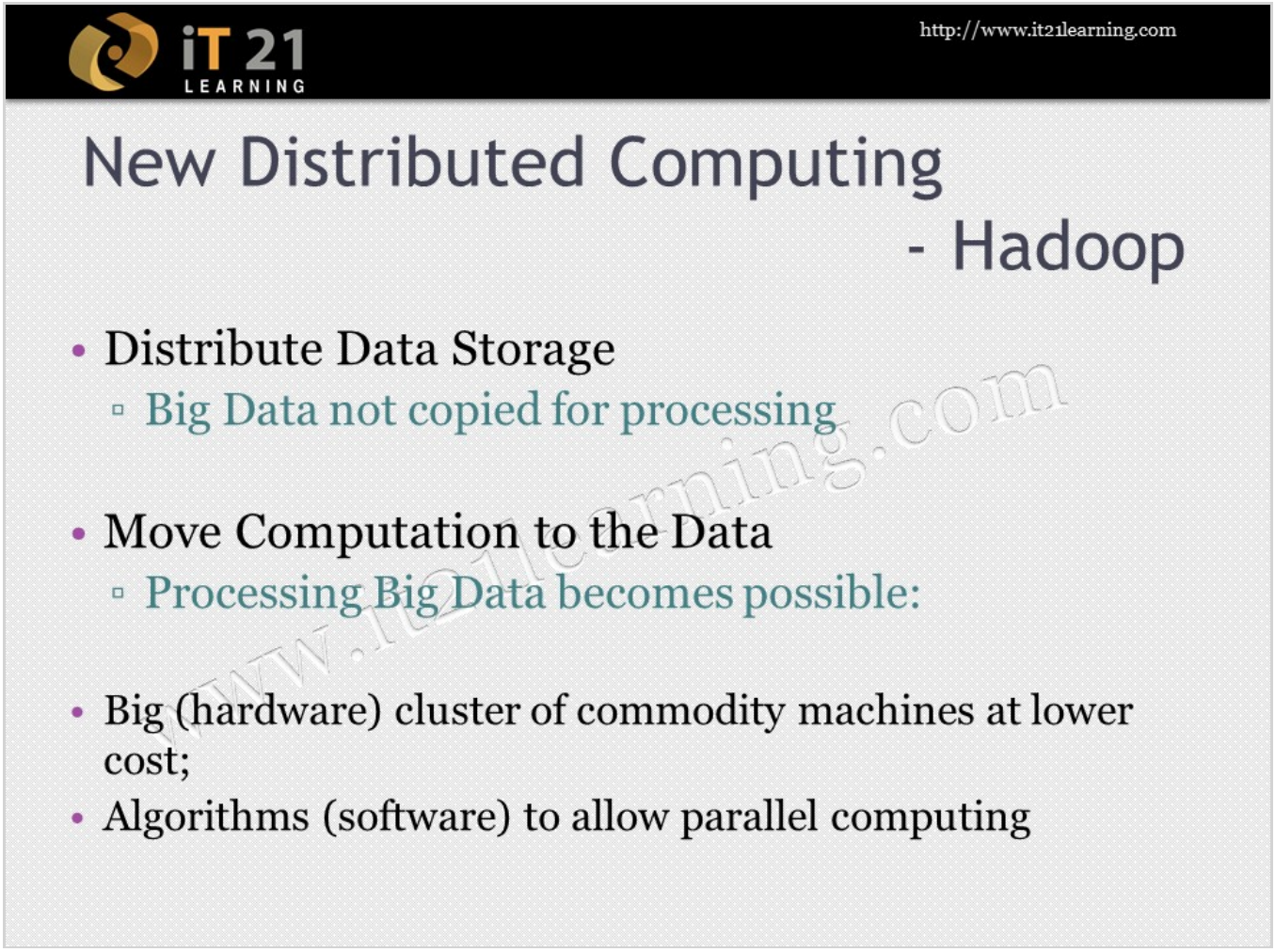
Two Inherent Characteristics



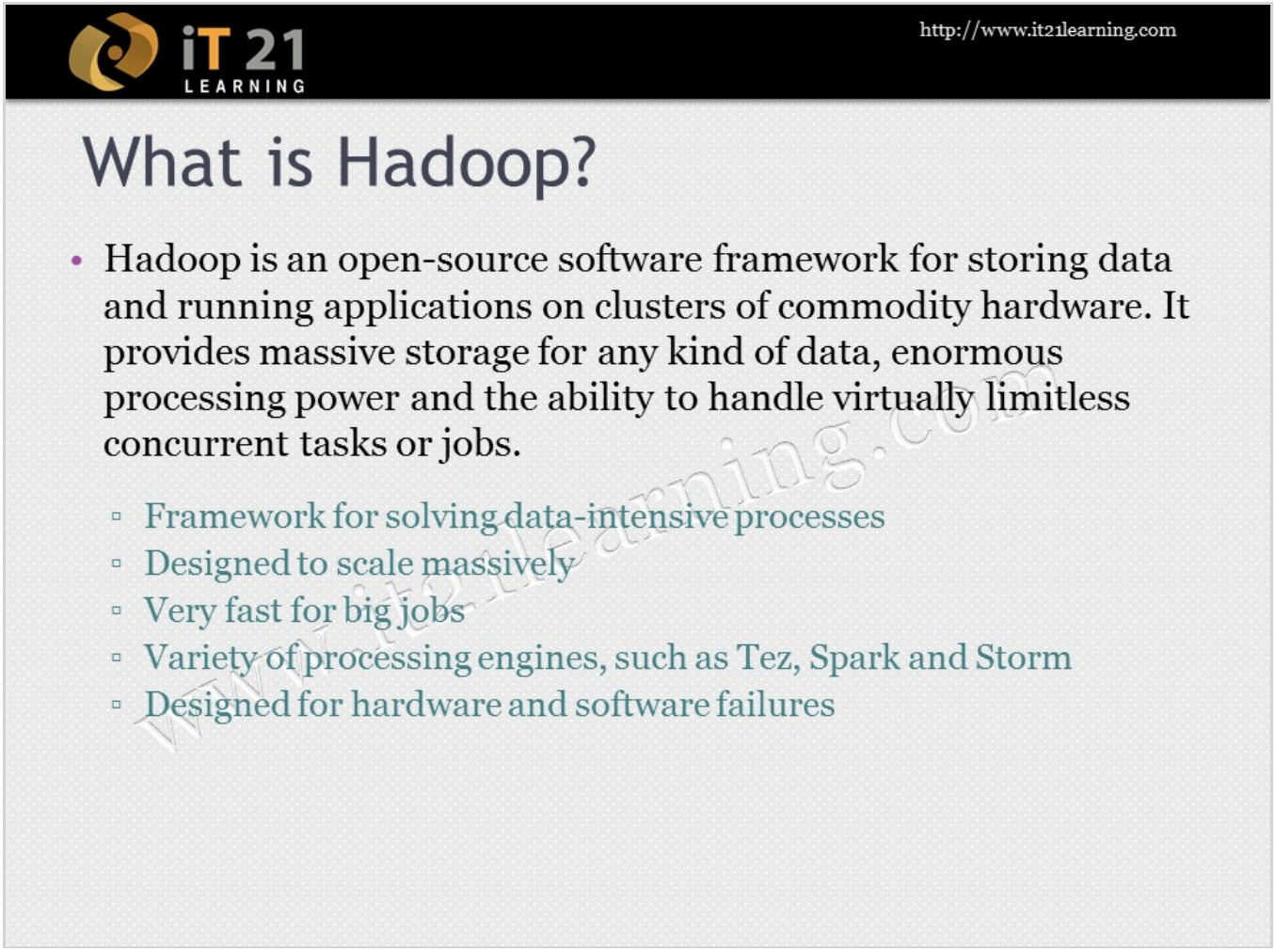
4


iT 21 LEARNING
http://www.it21learning.com
Distributed Computing
• Traditional Distributed Computing:
▫ Computation has been CPU bound;
• Relatively small amounts of data
• Significant amount of complex processing performed on the data
▫ For decades, the primary push was to increase the computing power of a single machine.
• Typically at compute time, data is copied over to the compute nodes
www.it21learning.com

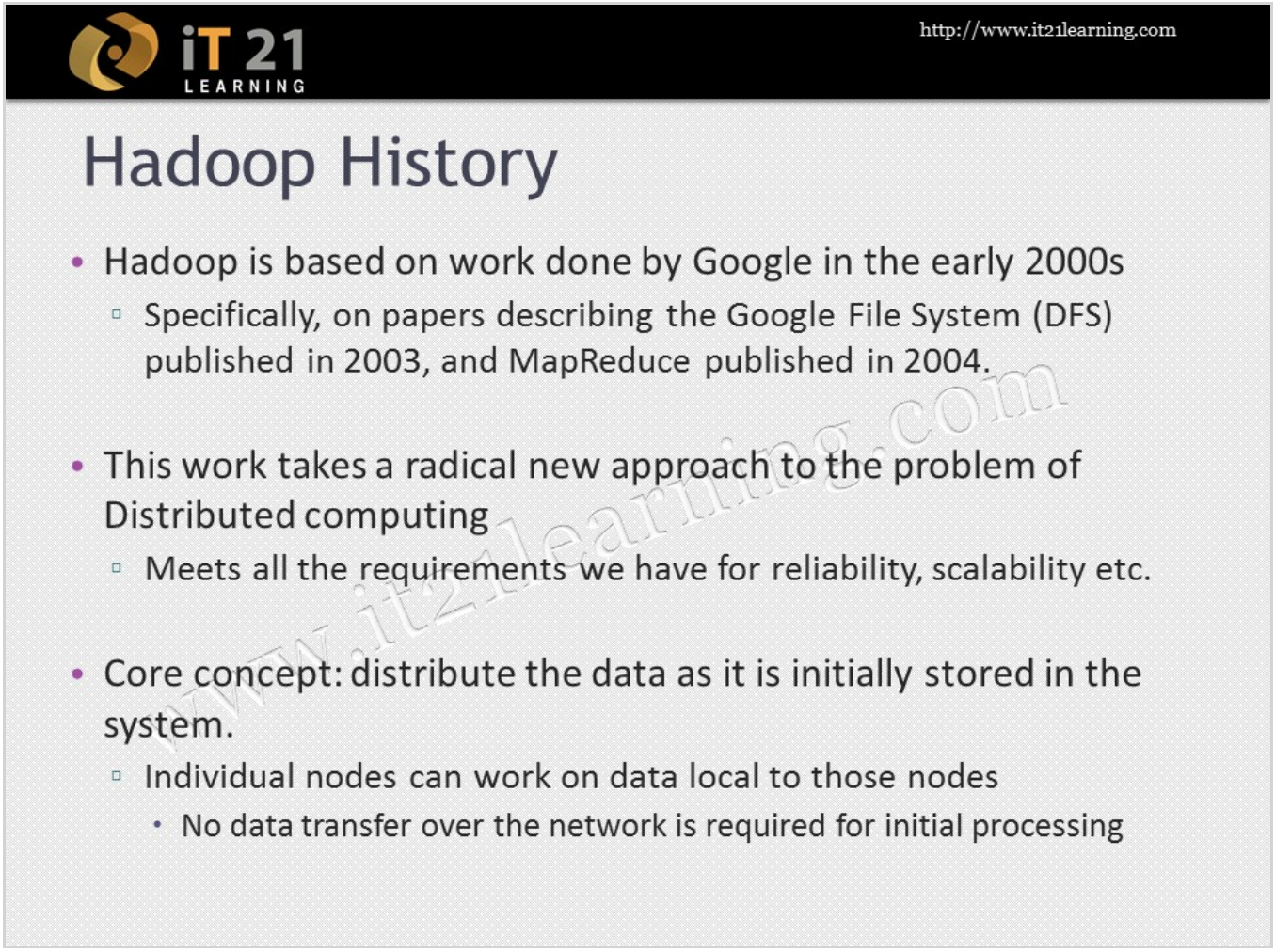
5


iT 21 LEARNING
http://www.it21learning.com
New Distributed Computing
- Hadoop
• Distribute Data Storage
▫ Big Data not copied for processing
• Move Computation to the Data
▫ Processing Big Data becomes possible:
• Big (hardware) cluster of commodity machines at lower cost;
• Algorithms (software) to allow parallel computing
www.it21learning.com

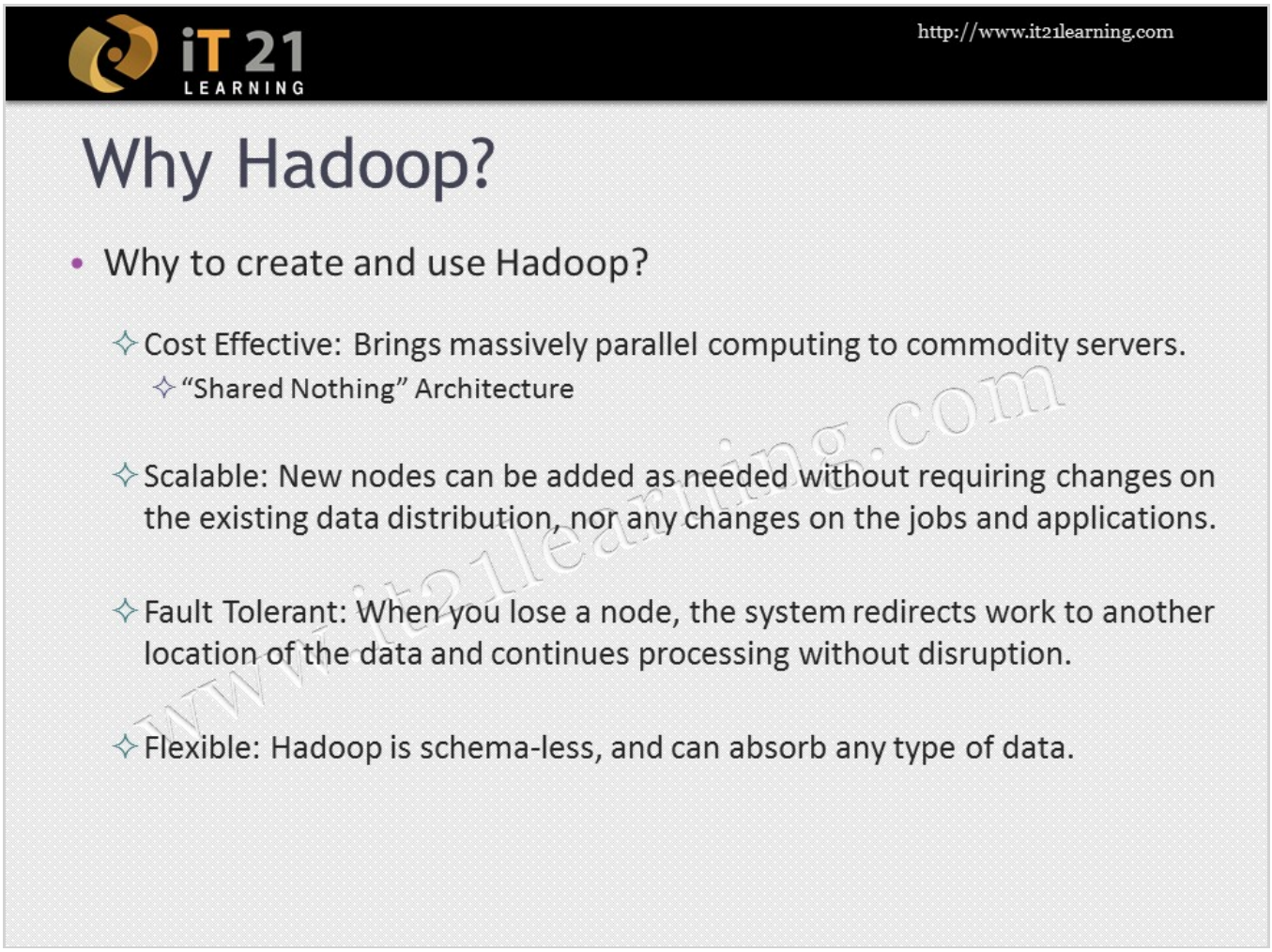
6


iT 21 LEARNING
http://www.it21learning.com
What is Hadoop?
• Hadoop is an open-source software framework for storing data and running applications on clusters of commodity hardware. It provides massive storage for any kind of data, enormous processing power and the ability to handle virtually limitless concurrent tasks or jobs.
▫ Framework for solving data-intensive processes
▫ Designed to scale massively
▫ Very fast for big jobs
▫ Variety of processing engines, such as Tez, Spark and Storm
▫ Designed for hardware and software failures

7

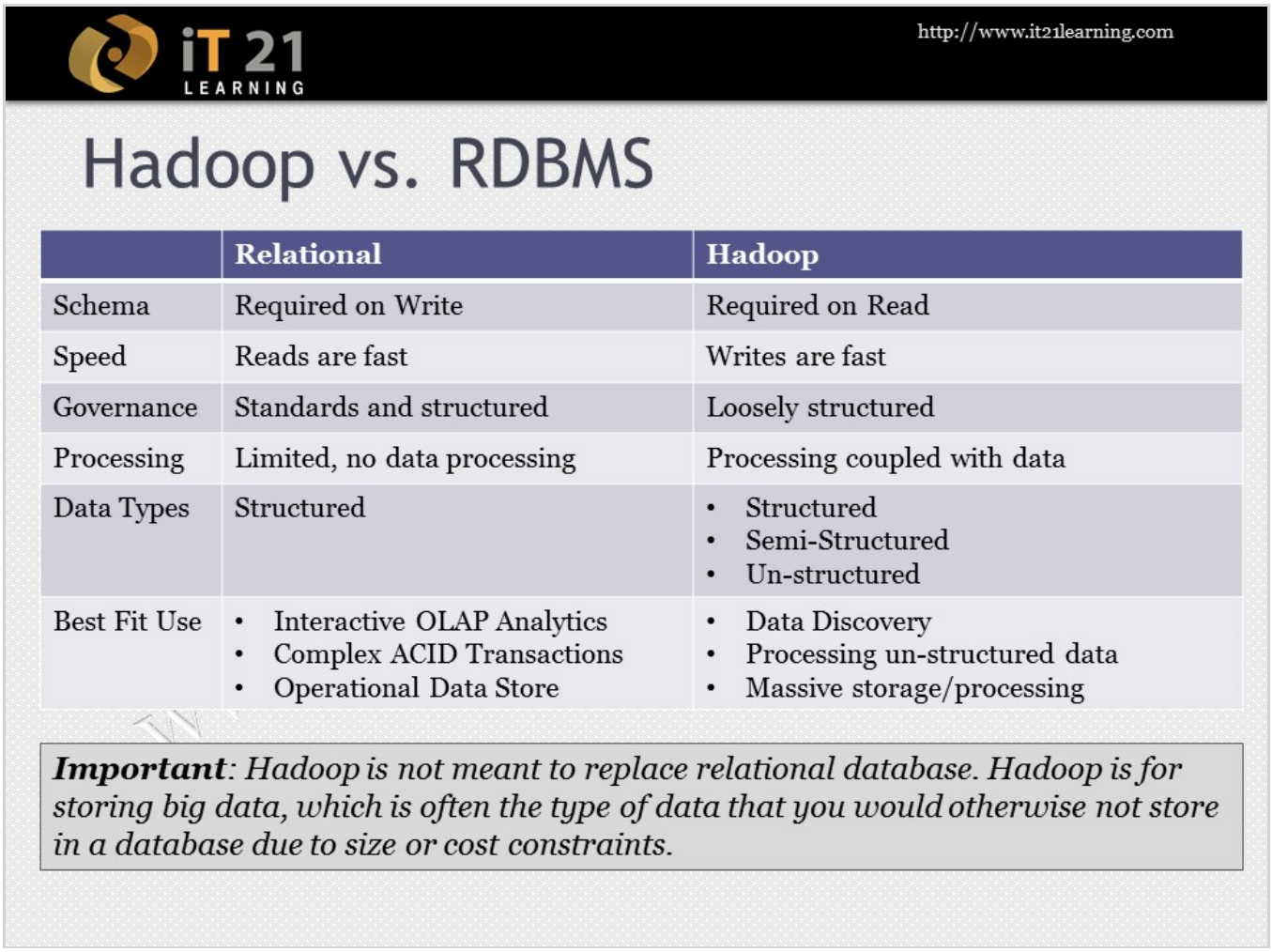

iT 21 LEARNING
http://www.it21learning.com
Hadoop History
• Hadoop is based on work done by Google in the early 2000s
▫ Specifically, on papers describing the Google File System (DFS) published in 2003, and MapReduce published in 2004.
• This work takes a radical new approach to the problem of Distributed computing
▫ Meets all the requirements we have for reliability, scalability etc.
• Core concept: distribute the data as it is initially stored in the system.
▫ Individual nodes can work on data local to those nodes
• No data transfer over the network is required for initial processing
www.it21learning.com

8


iT 21 LEARNING
http://www.it21learning.com
Why Hadoop?
• Why to create and use Hadoop?
✧ Cost Effective: Brings massively parallel computing to commodity servers.
✧ "Shared Nothing" Architecture
✧ Scalable: New nodes can be added as needed without requiring changes on the existing data distribution, nor any changes on the jobs and applications.
✧ Fault Tolerant: When you lose a node, the system redirects work to another location of the data and continues processing without disruption.
✧ Flexible: Hadoop is schema-less, and can absorb any type of data.

9

Hadoop vs. RDBMS

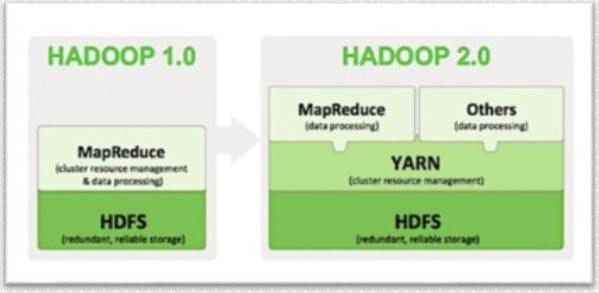


Data governance (DG) refers to the overall management of the availability, usability, integrity, and security of the data employed in an enterprise.

Hadoop Architecture

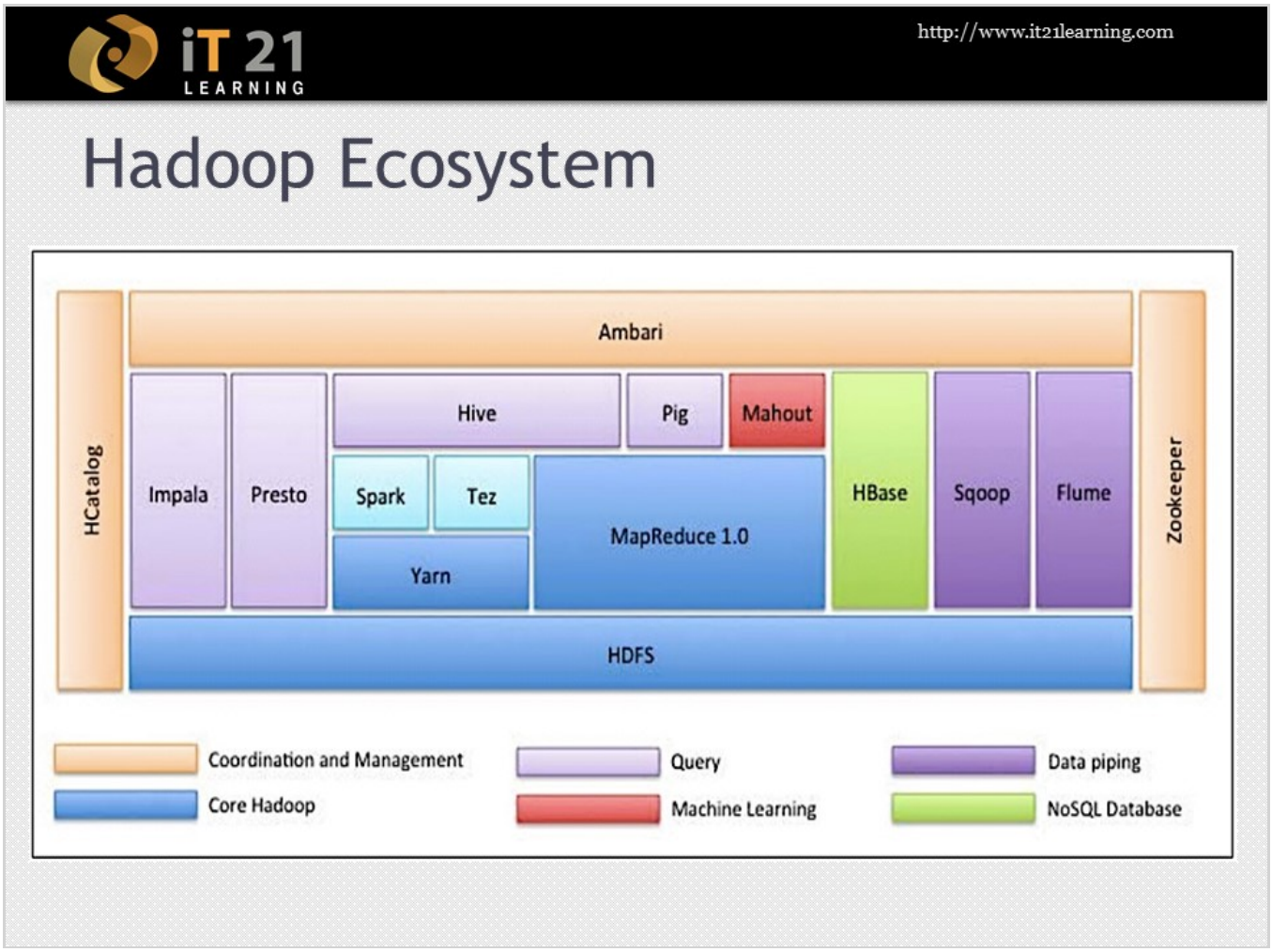
Hadoop Architecture

- **Hadoop Common:**
  - Common Utilities that supports all other modules.
- **Hadoop Distributed File System (HDFS):**
  - File system that spans all the nodes in a Hadoop cluster for data storage.
  - Links the file systems on local nodes to make them into one big file system.
- **Hadoop MapReduce:**
  - Core computing framework available since Hadoop 1.x
- **Hadoop YARN:**
  - New distributed processing framework on Hadoop 2.x.
  - Addresses multiple limitations of MR 1.0
- **Hadoop Ecosystem:**
  - Open-source Apache projects



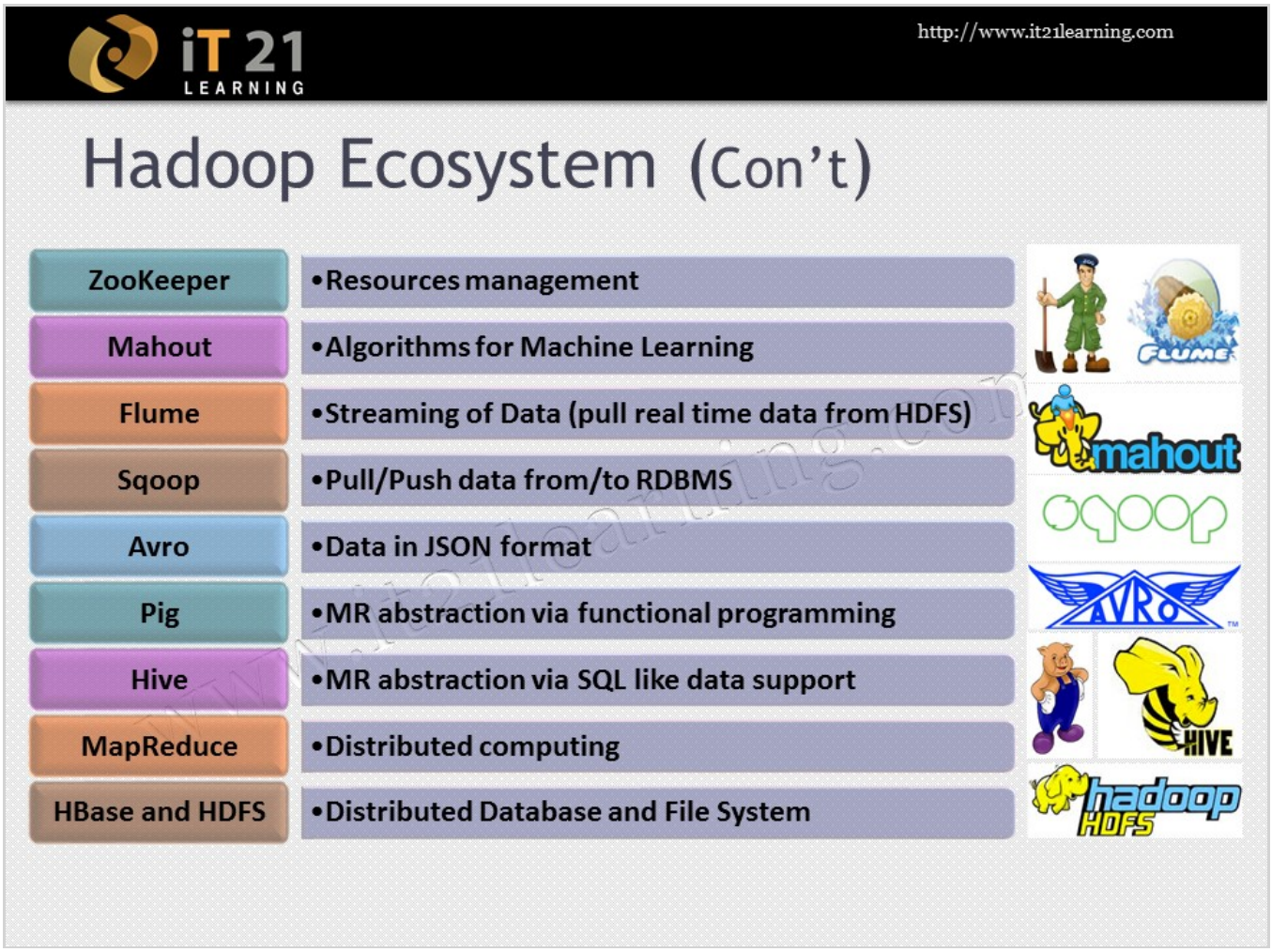
11

Hadoop Ecosystem



12

Hadoop Ecosystem (Con't)

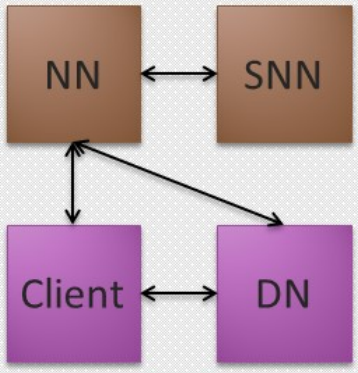


13

Hadoop (HDFS) Roles

Hadoop (HDFS) Roles

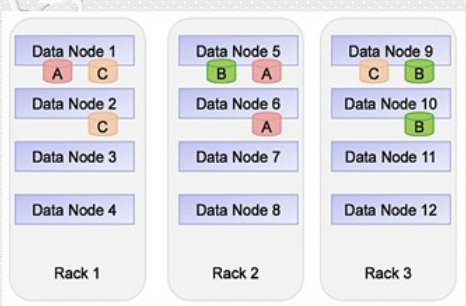
- **NameNode (NN)**
  - Manages the File System's namespace/meta-data/file blocks
  - Runs on 1 machine to several machines
- **Secondary NameNode (SNN)**
  - Performs house keeping work so NameNode doesn't
  - Requires similar hardware as NameNode machine
  - Not used for high-availability – not a backup for NameNode
- **DataNode (DN)**
  - Stores and retrieves data blocks
  - Reports to NameNode
  - Runs on many machines



14

Hadoop (HDFS) Components

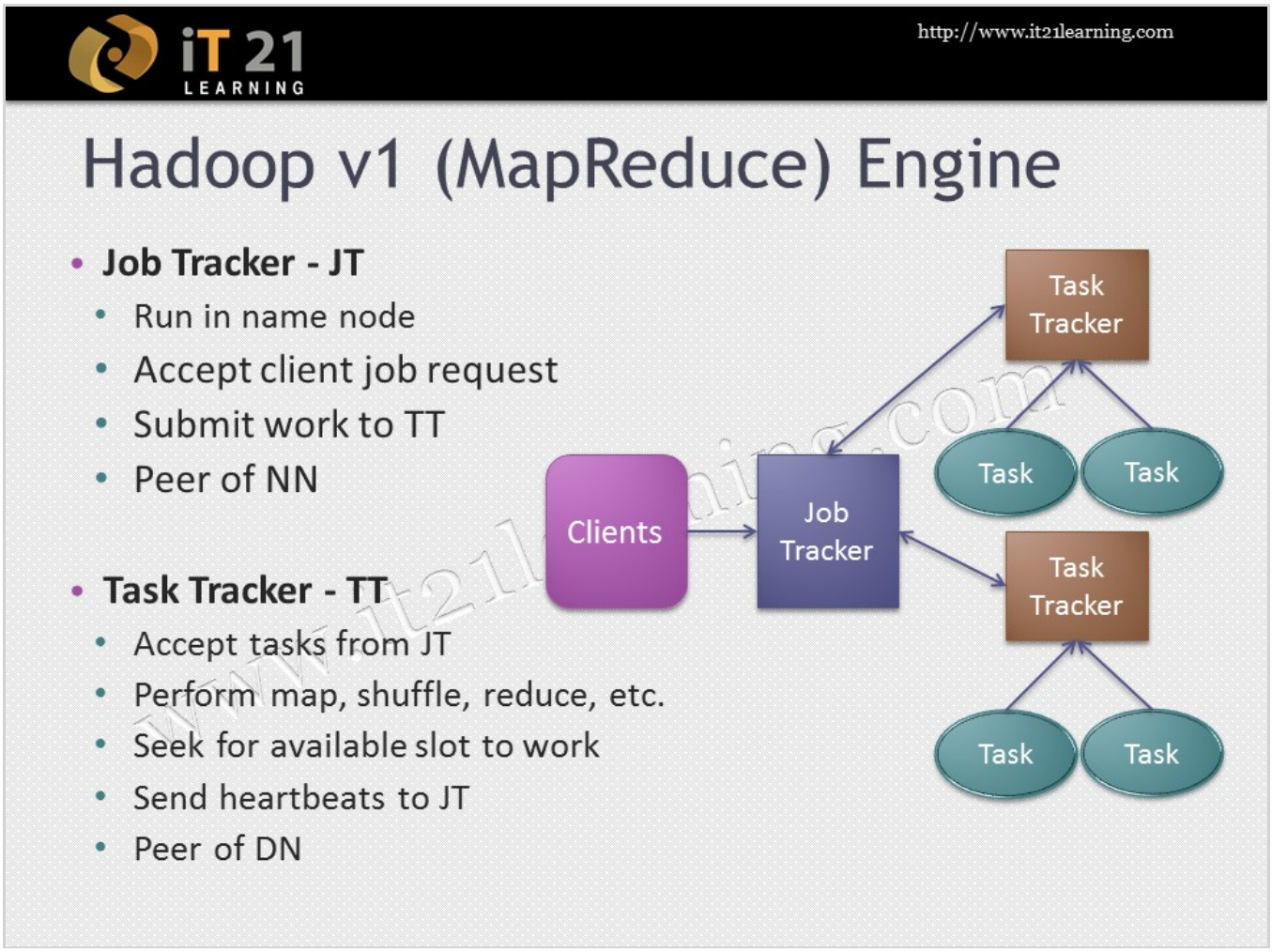
- **Client**
  - User/App interface to interact with cluster, DN
- **Namespace**
  - **Files/Directories** - Same to the regular file systems split into blocks
  - **Blocks**
    - Default: 64M (v1); 128M (v2)
    - Blocks meta data kept in NN
      – Small files issue



- **Block Storage:**
  - **Replications**
    - Default 3 and rebalanced for new added nodes
    - 1st replica on the local. 2nd on the local but different node. 3rd on the different rack

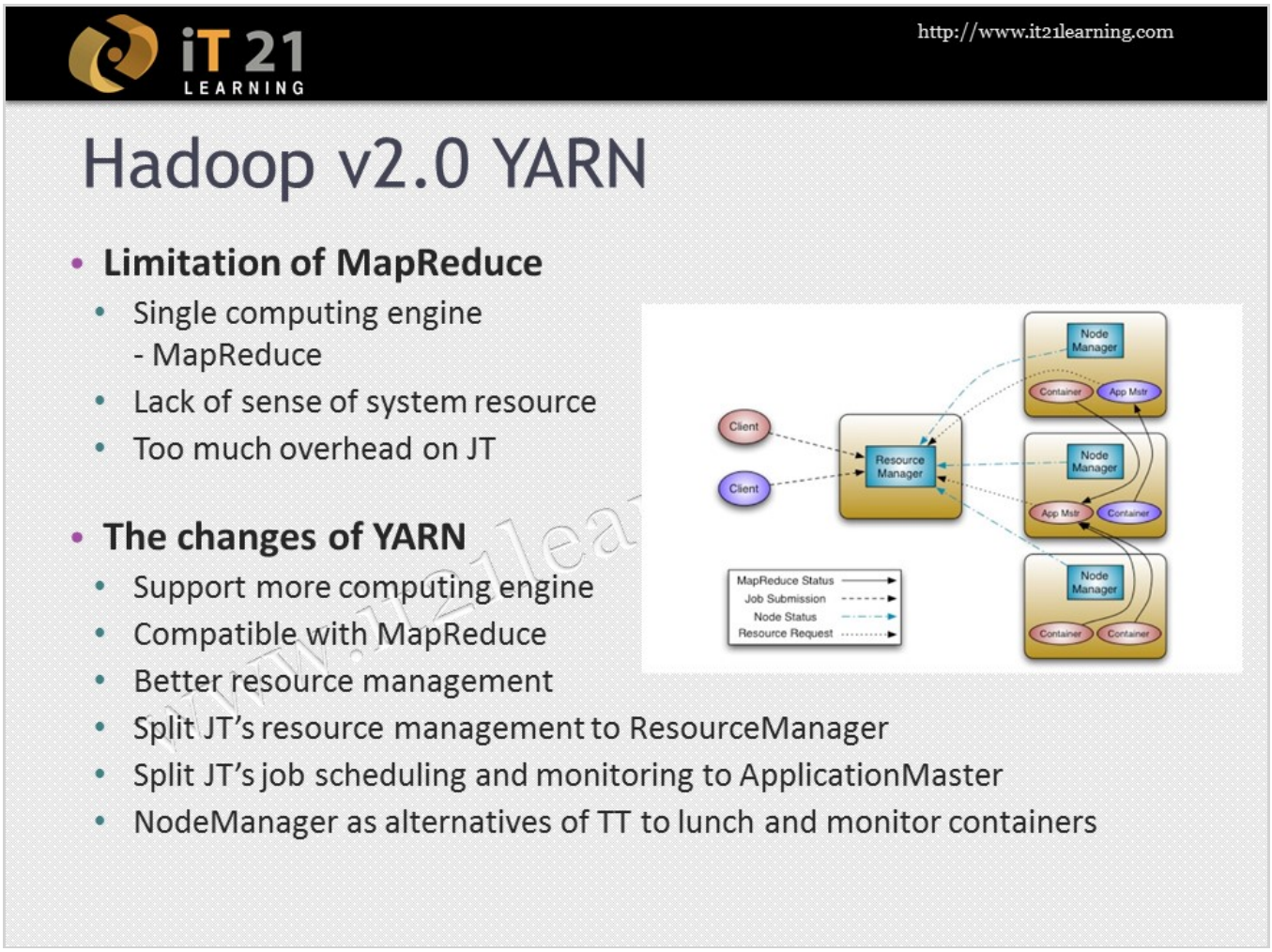
15

Hadoop v1 (MapReduce) Engine



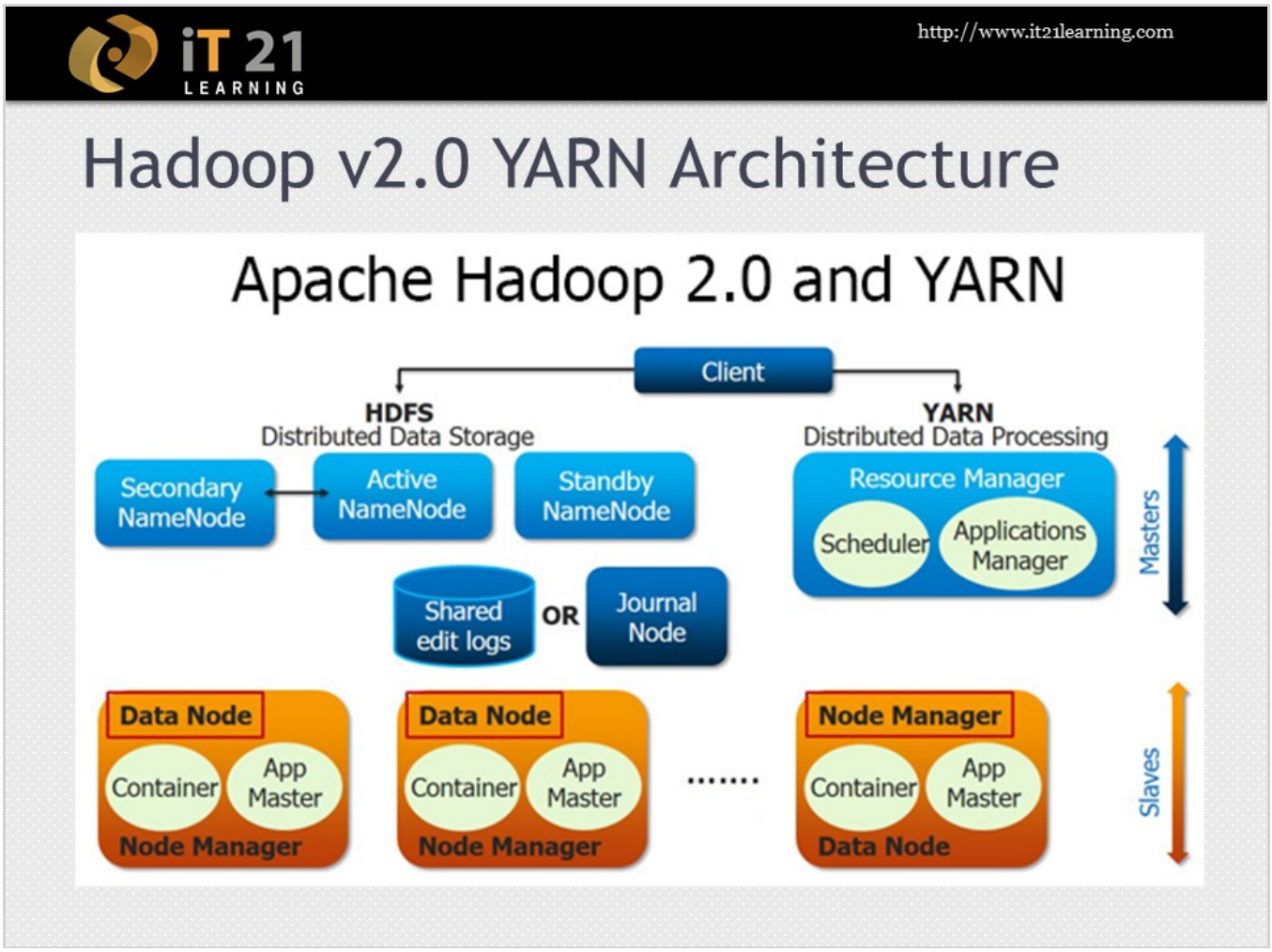
16

Hadoop v2.0 YARN

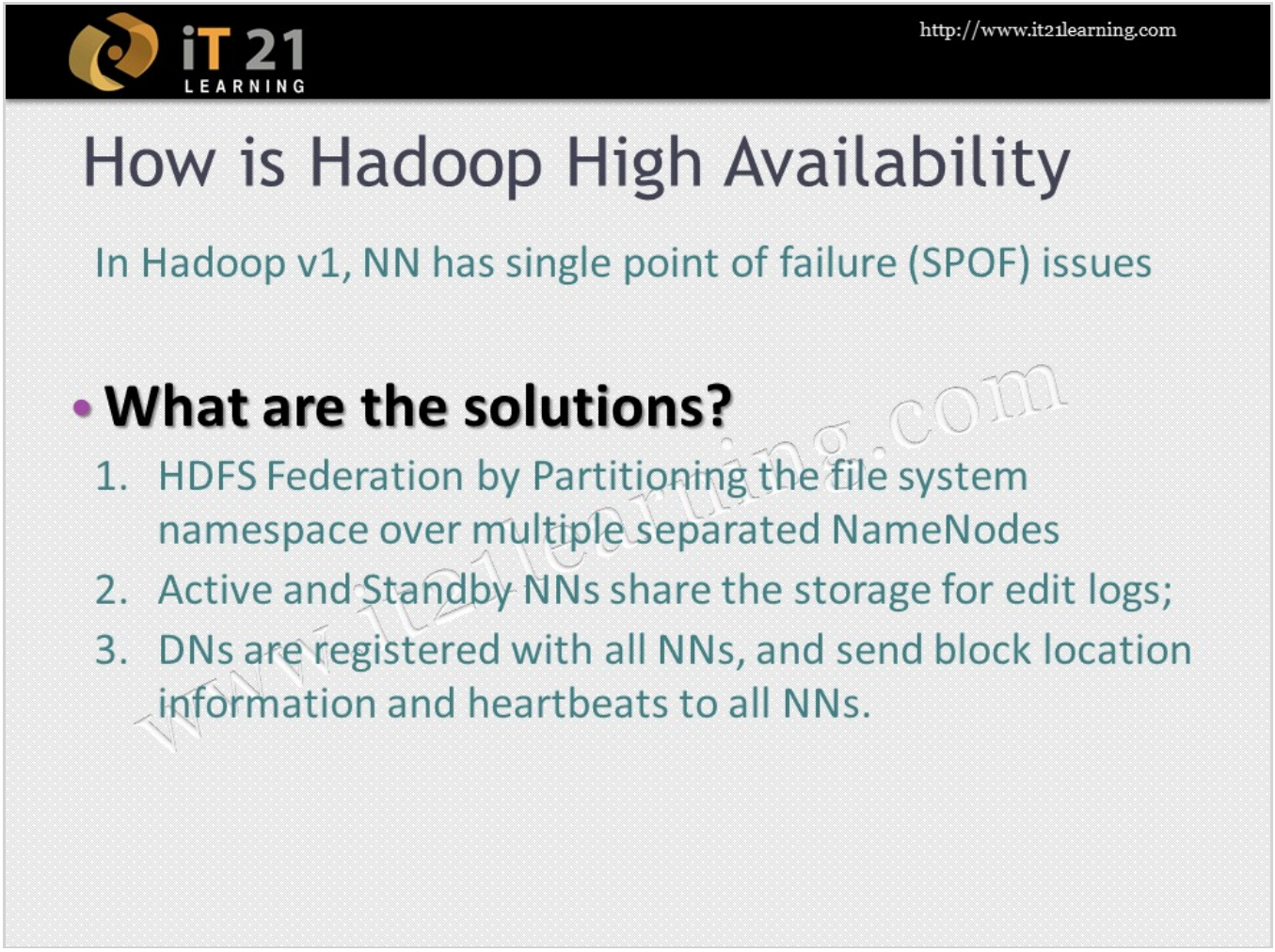


YARN stands for Yet Another Resource Negotiator. Let's walk through an application execution sequence (steps are illustrated in the diagram): A client program submits the application, including the necessary specifications to launch the application-specific ApplicationMaster itself. The ResourceManager assumes the responsibility to negotiate a specified container in which to start the ApplicationMaster and then launches the ApplicationMaster. The ApplicationMaster, on boot-up, registers with the ResourceManager – the registration allows the client program to query the ResourceManager for details, which allow it to directly communicate with its own ApplicationMaster. During normal operation the ApplicationMaster negotiates appropriate resource containers via the resource-request protocol. On successful container allocations, the ApplicationMaster launches the container by providing the container launch specification to the NodeManager. The launch specification, typically, includes the necessary information to allow the container to communicate with the ApplicationMaster itself. The application code executing within the container then provides necessary information (progress, status etc.) to its ApplicationMaster via an application-specific protocol. During the application execution, the client that submitted the program communicates directly with the ApplicationMaster to get status, progress updates etc. via an application-specific protocol. Once the application is complete, and all necessary work has been finished, the ApplicationMaster deregisters with the ResourceManager and shuts down, allowing its own container to be repurposed.

Hadoop v2.0 YARN Architecture

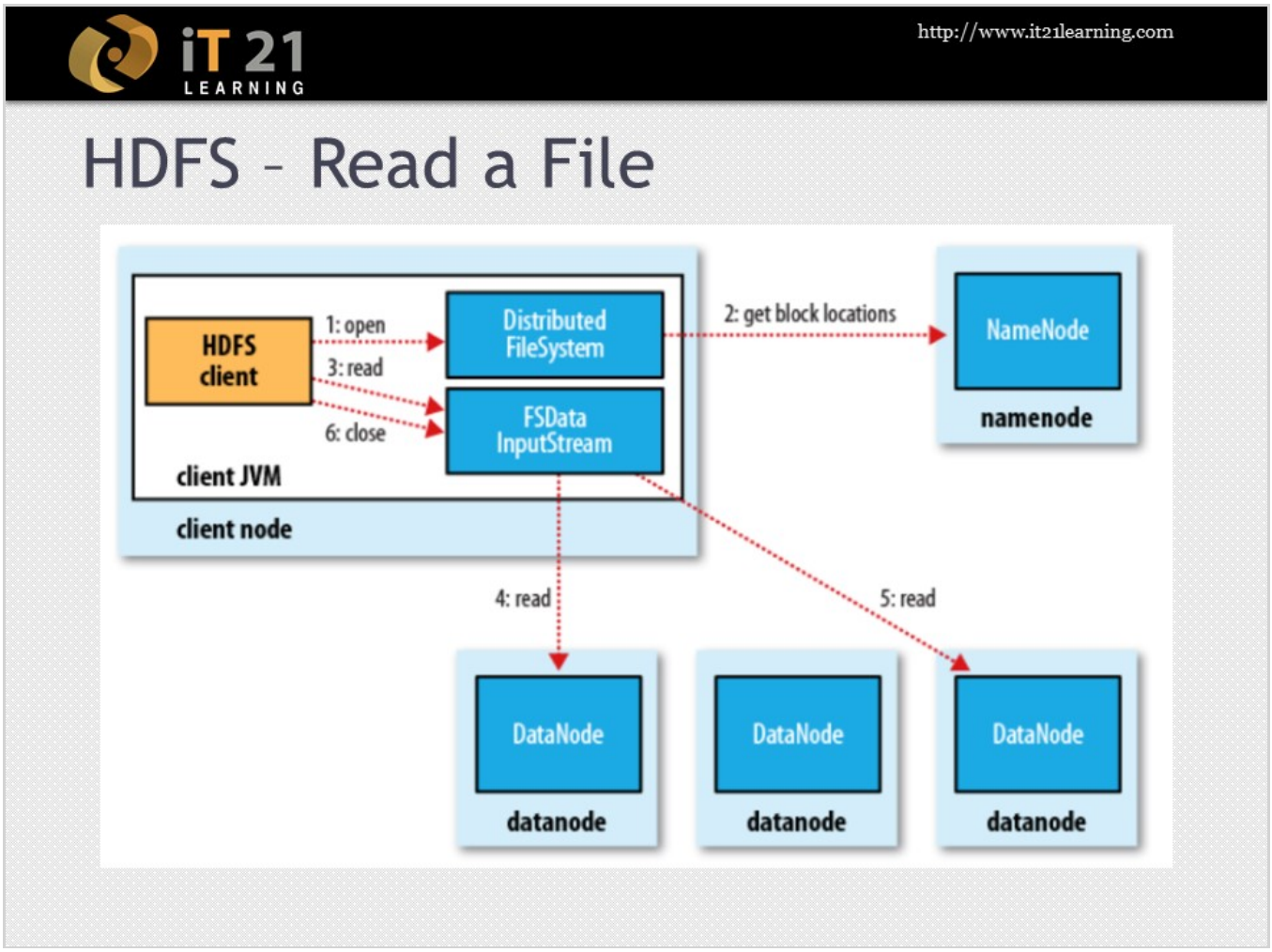


As the data Operating System for Hadoop 2.0, YARN allows multiple applications to co-exist in the same shared cluster. YARN also takes into consideration of Scalability, High utilization, support multiple programming models, flexible resource model, Security, Reliability, and compatibility. YARN separates the JobTracker (Hadoop 1.x) into a cluster level ResourceManager (RM) and application specific ApplicationMaster (AM). YARN architecture follows master-slave architectural model in which ResourceManager as a master and node specific slave NodeManager (NM). NodeManager replaces the TaskTracker in Hadoop 1.x. A Client submits the application to the ResourceManager. In above diagram client 1 submit a MapReduce Request and client 2 submit Shell Script Request. ResourceManager allocates a container to start up the ApplicationMaster for each application request submitted by client – one ApplicationMaster for Shell Script and one for MapReduce application. After starting a ApplicationMaster, it registers application with ResourceManager. After startup of ApplicationMaster, it negotiates with ResourceManager for appropriate resources as per the application requirement. After resource allocation from ResourceManager, ApplicationMaster requests NodeManager to launch the containers allocated by ResourceManager. On successful launching of containers, application code executes within the container. ApplicationManager reports back to ResourceManager with execution status of application. During execution of the application, client can request ApplicationMaster or directly with ResourceManager for application status, progress updates etc. On application execution process completion, ApplicationMaster request ResourceManager to unregister and shut down its own container process.


iT 21 LEARNING
http://www.it21learning.com
How is Hadoop High Availability
In Hadoop v1, NN has single point of failure (SPOF) issues
What are the solutions?
1. HDFS Federation by Partitioning the file system namespace over multiple separated NameNodes
2. Active and Standby NNs share the storage for edit logs;
3. DNs are registered with all NNs, and send block location information and heartbeats to all NNs.
www.it21learning.com

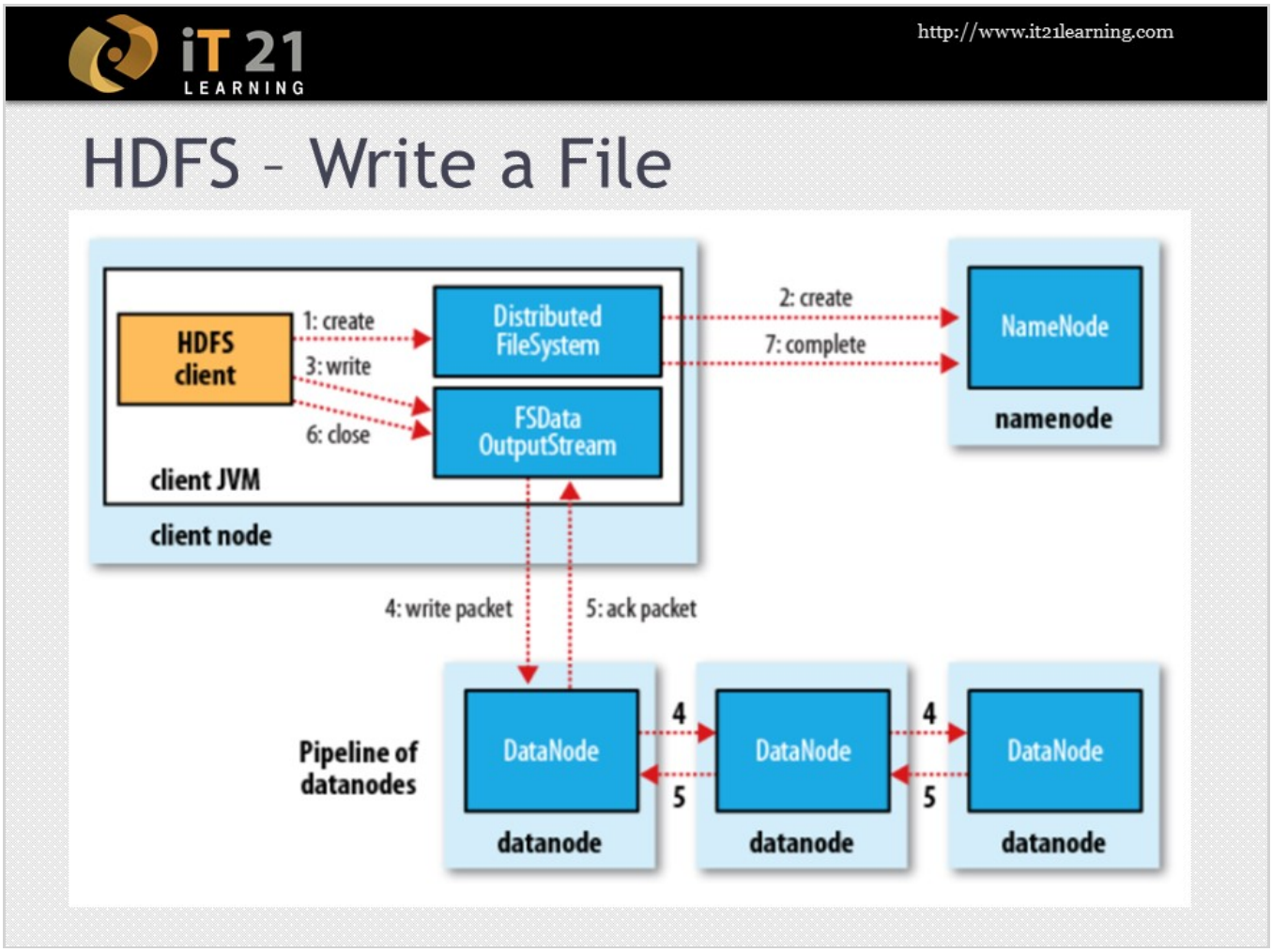
19

HDFS – Read a File



20

HDFS – Write a File

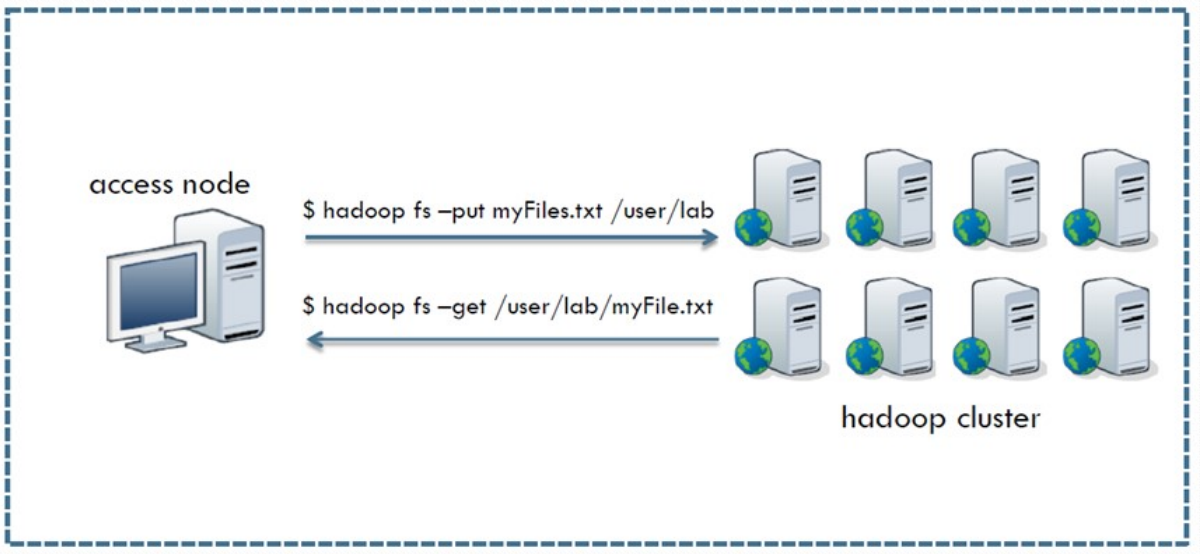


21

HDFS – CLI (Command Line)

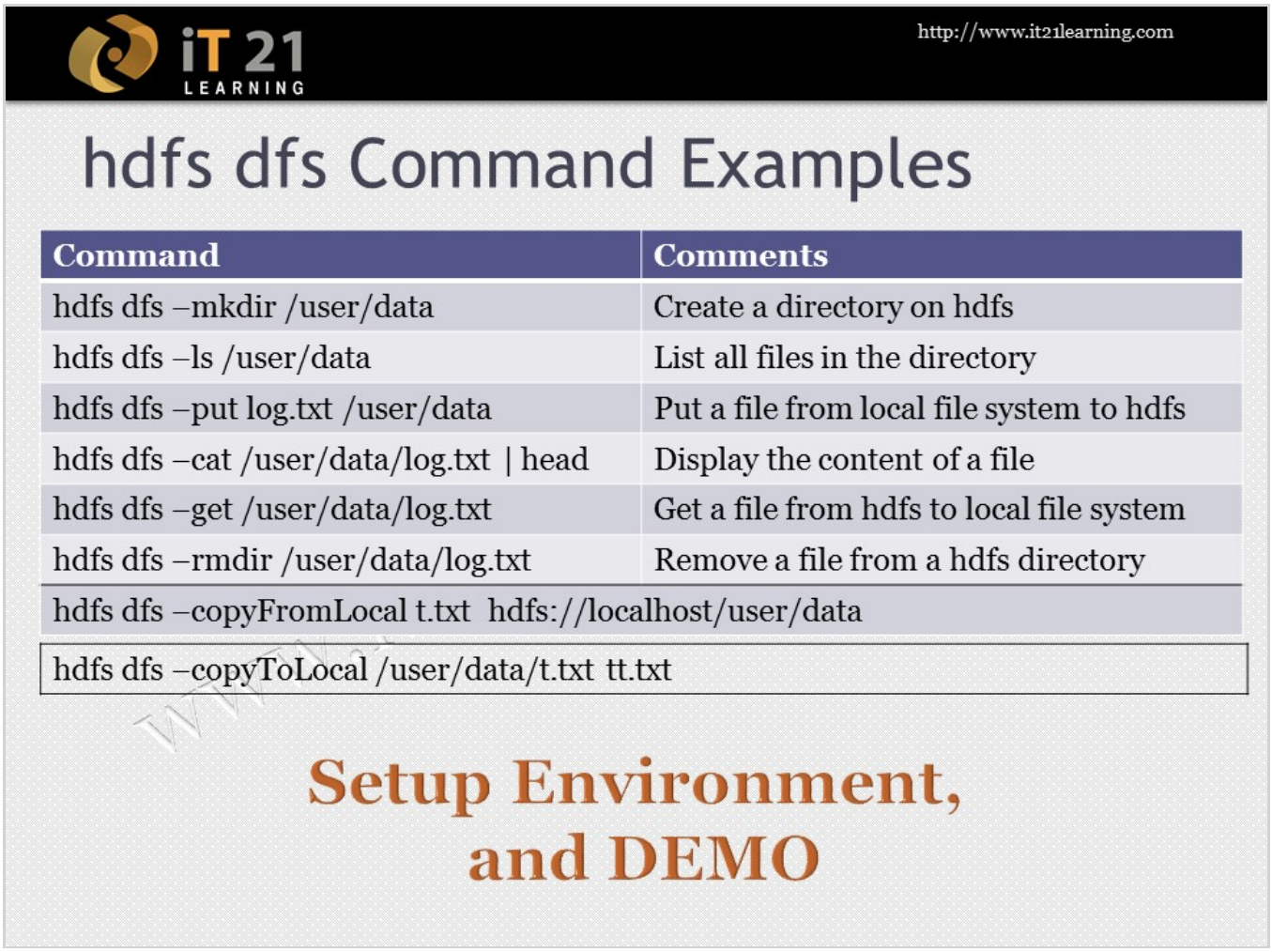
HDFS - CLI (Command Line)

- Users typically access HDFS via hdfs dfs or hadoop fs command
  - *hadoop fs is depreciated.*

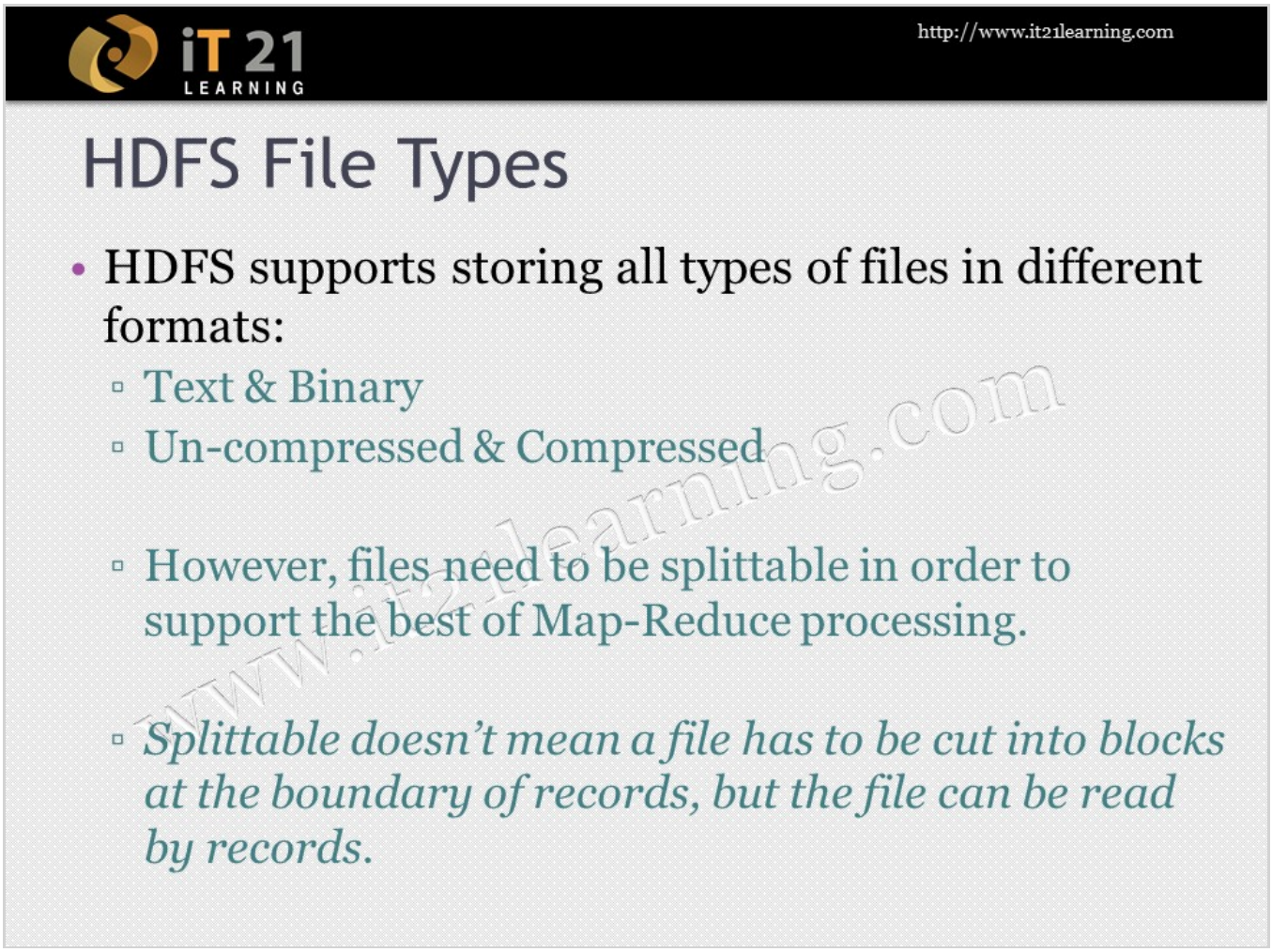


22

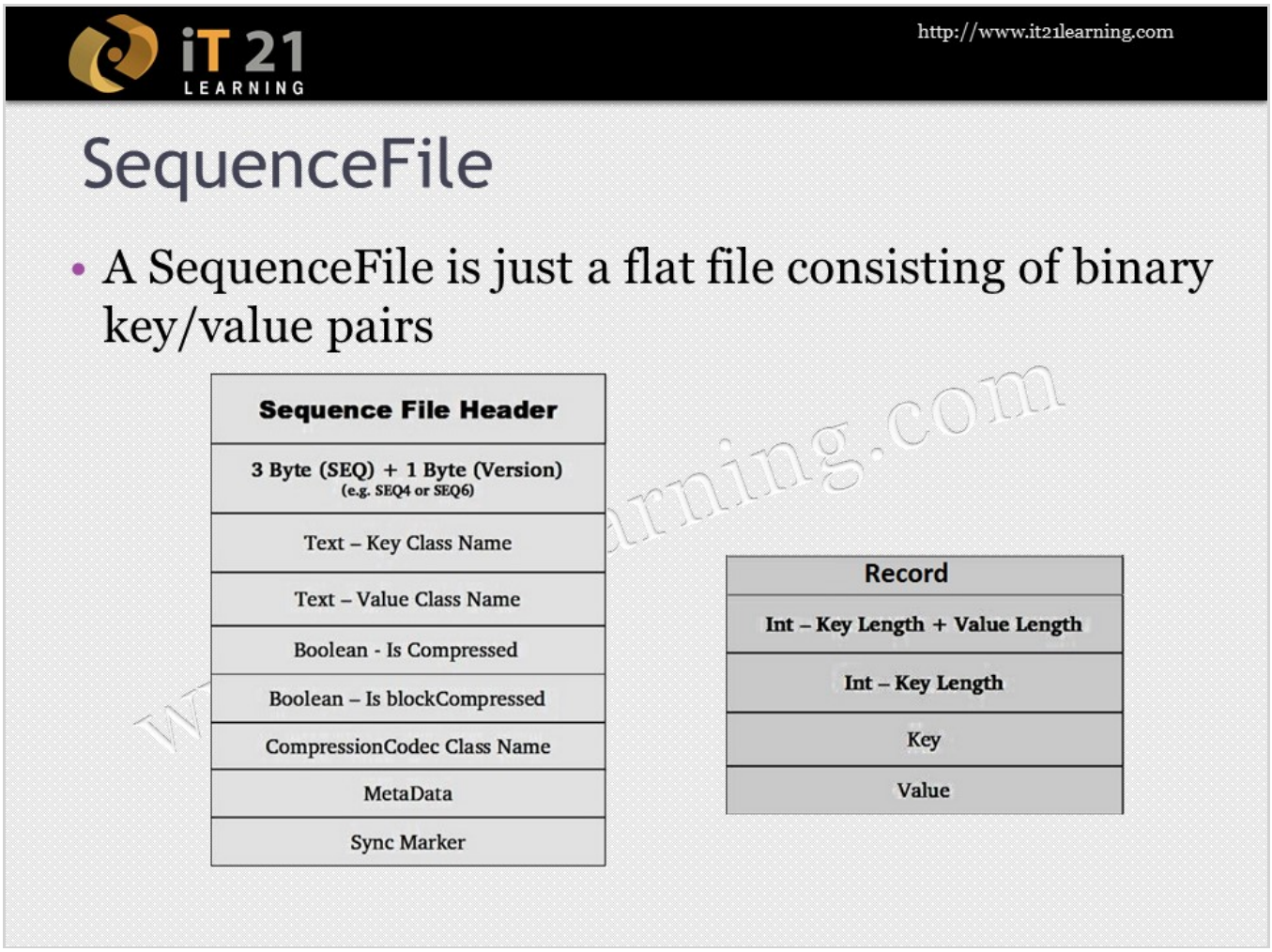
hdfs dfs Command Examples



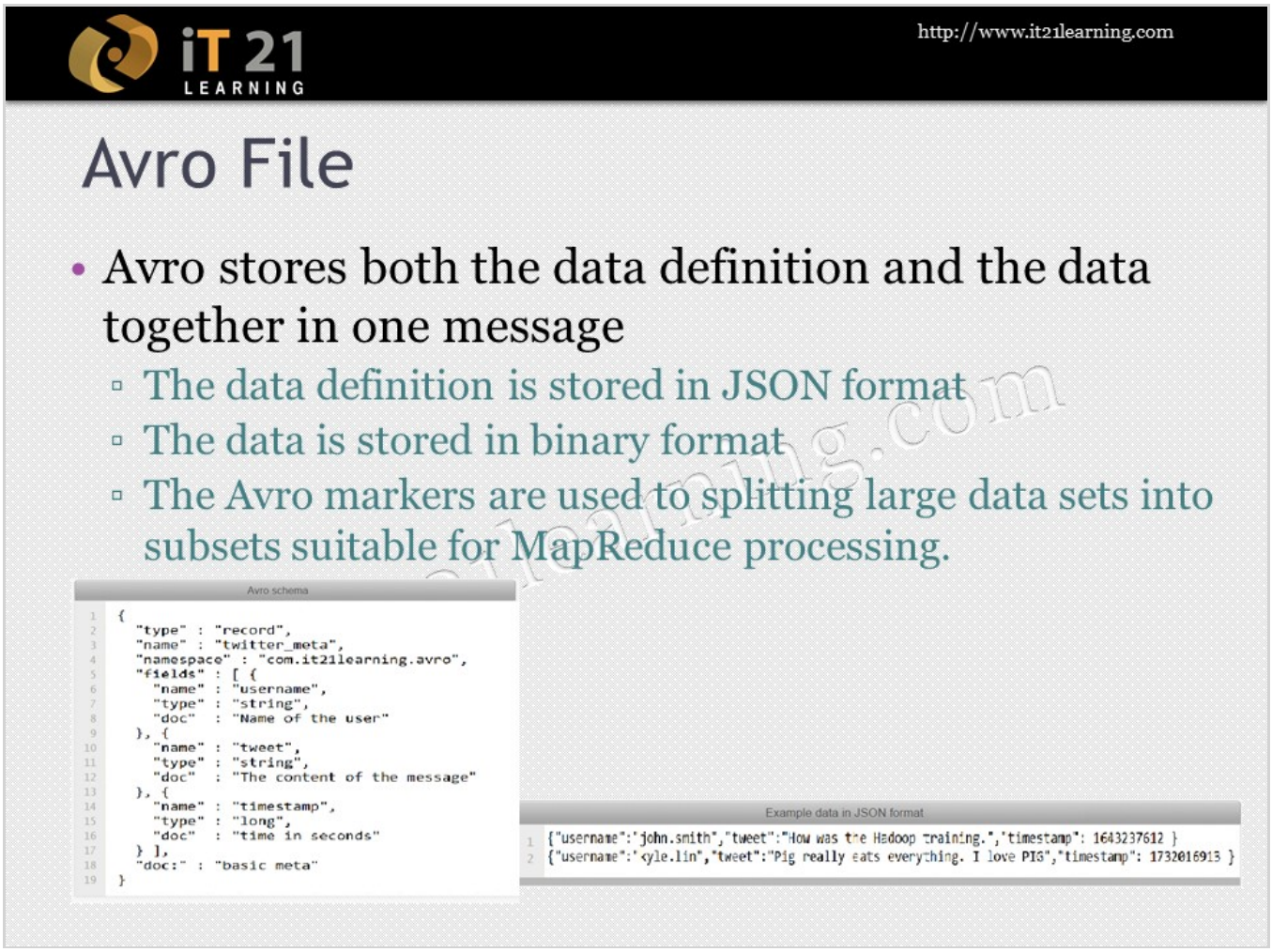
23


iT 21 LEARNING
http://www.it21learning.com
HDFS File Types
• HDFS supports storing all types of files in different formats:
▫ Text & Binary
▫ Un-compressed & Compressed
▫ However, files need to be splittable in order to support the best of Map-Reduce processing.
▫ Splittable doesn't mean a file has to be cut into blocks at the boundary of records, but the file can be read by records.
www.it21learning.com

24


iT 21 LEARNING
http://www.it21learning.com
SequenceFile
• A SequenceFile is just a flat file consisting of binary key/value pairs
Sequence File Header
3 Byte (SEQ) + 1 Byte (Version)
(e.g. SEQ4 or SEQ6)
Text – Key Class Name
Text – Value Class Name
Boolean - Is Compressed
Boolean – Is blockCompressed
CompressionCodec Class Name
MetaData
Sync Marker
Record
Int – Key Length + Value Length
Int – Key Length
Key
Value

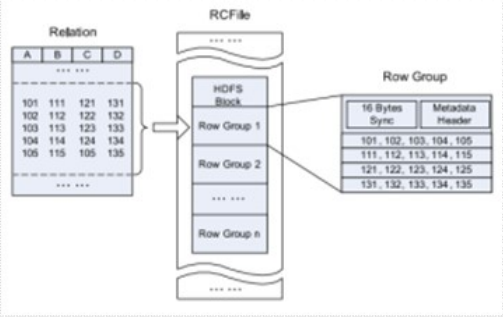
25


iT 21 LEARNING
http://www.it21learning.com
Avro File
• Avro stores both the data definition and the data together in one message
▫ The data definition is stored in JSON format
▫ The data is stored in binary format
▫ The Avro markers are used to splitting large data sets into subsets suitable for MapReduce processing.
Avro schema
{
"type" : "record",
"name" : "twitter_meta",
"namespace" : "com.it21learning.avro",
"fields" : [{
"name" : "username",
"type" : "string",
"doc" : "Name of the user"
}, {
"name" : "tweet",
"type" : "string",
"doc" : "The content of the message"
}, {
"name" : "timestamp",
"type" : "long",
"doc" : "time in seconds"
}],
"doc:" : "basic meta"
}
Example data in JSON format
{"username":"john.smith","tweet":"How was the Hadoop training.","timestamp": 1643237612 }
{"username":"kyle.lin","tweet":"Pig really eats everything. I love PIG","timestamp": 1732016913 }

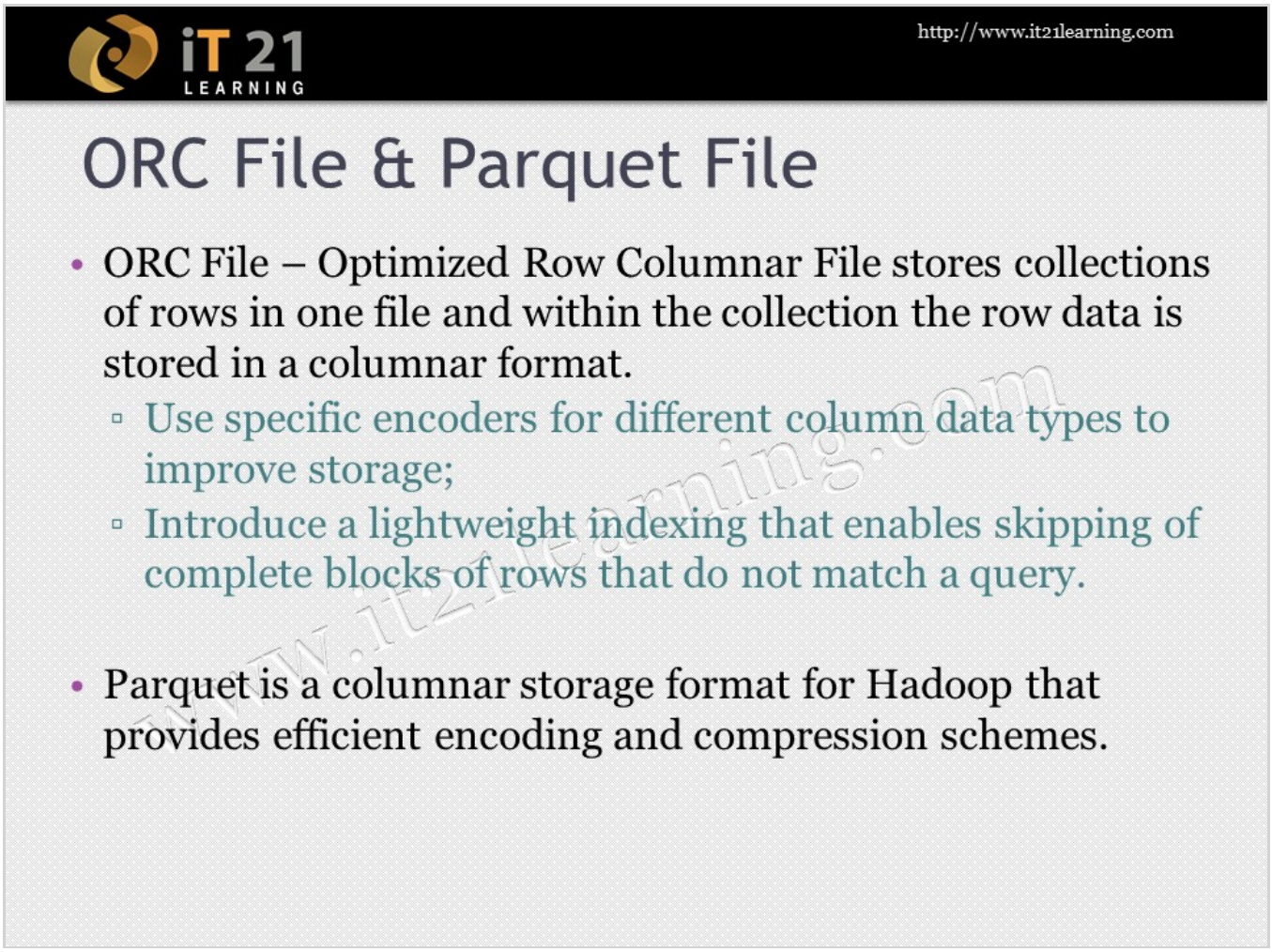
26

RCFile - Record Columnar File

- RCFile has been adopted by the Hive and Pig projects as the core format for table like data storage.
  - The RCFile splits data horizontally into row groups.
    - Rows 1 to 100 are stored in one group and rows 101 to 200 in the next and so on.
  - The RCFile saves each row group data in a columnar format.
    - Instead of storing row one then row two, it stores column one across all rows then column two across all rows and so on.
  - Benefit
    - Each processing node reads only the columns relevant to a query from a file and skips irrelevant ones

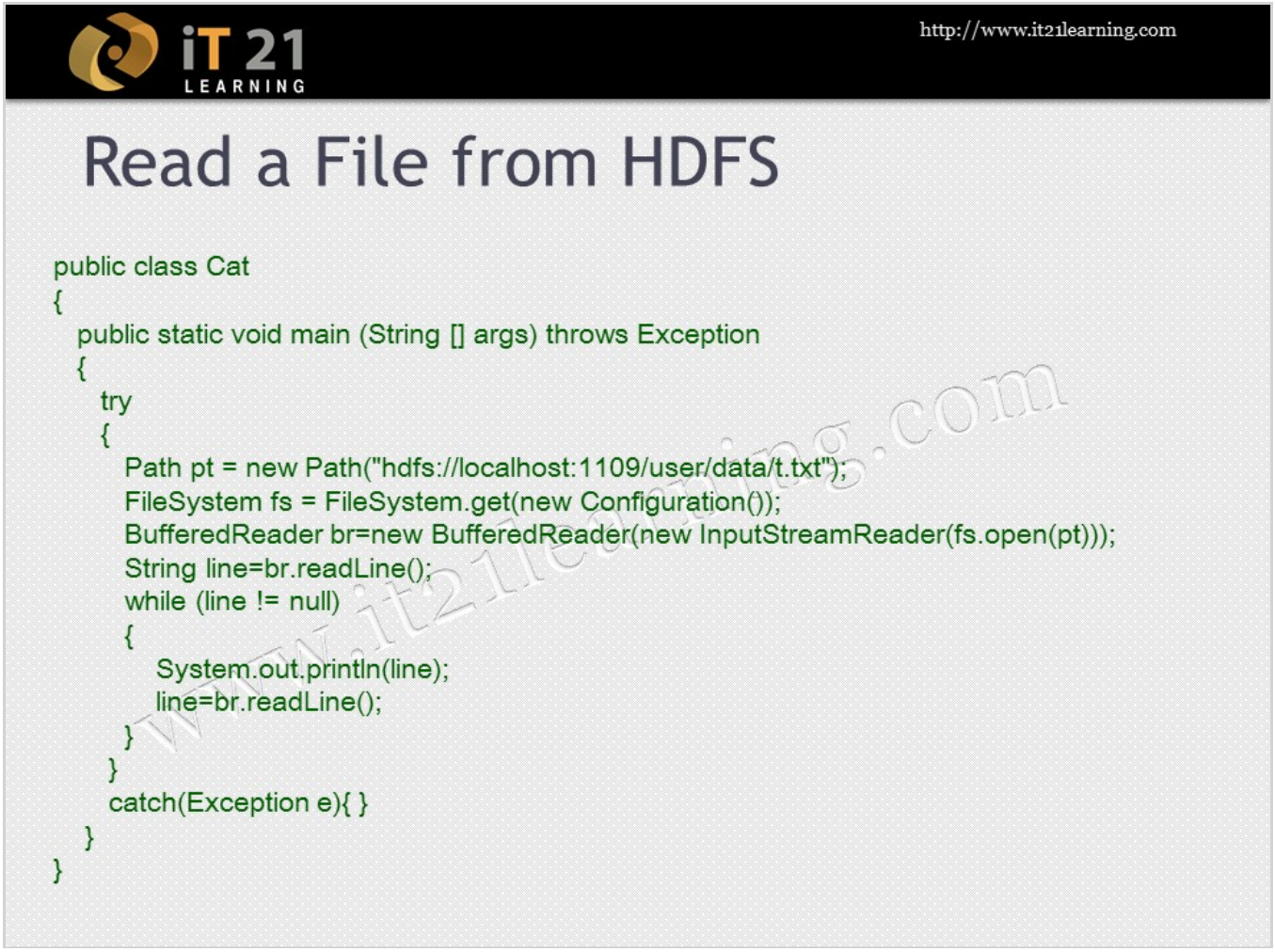


27


iT 21 LEARNING
http://www.it21learning.com
ORC File & Parquet File
• ORC File – Optimized Row Columnar File stores collections of rows in one file and within the collection the row data is stored in a columnar format.
▫ Use specific encoders for different column data types to improve storage;
▫ Introduce a lightweight indexing that enables skipping of complete blocks of rows that do not match a query.
• Parquet is a columnar storage format for Hadoop that provides efficient encoding and compression schemes.
www.it21learning.com

28

Read a File from HDFS



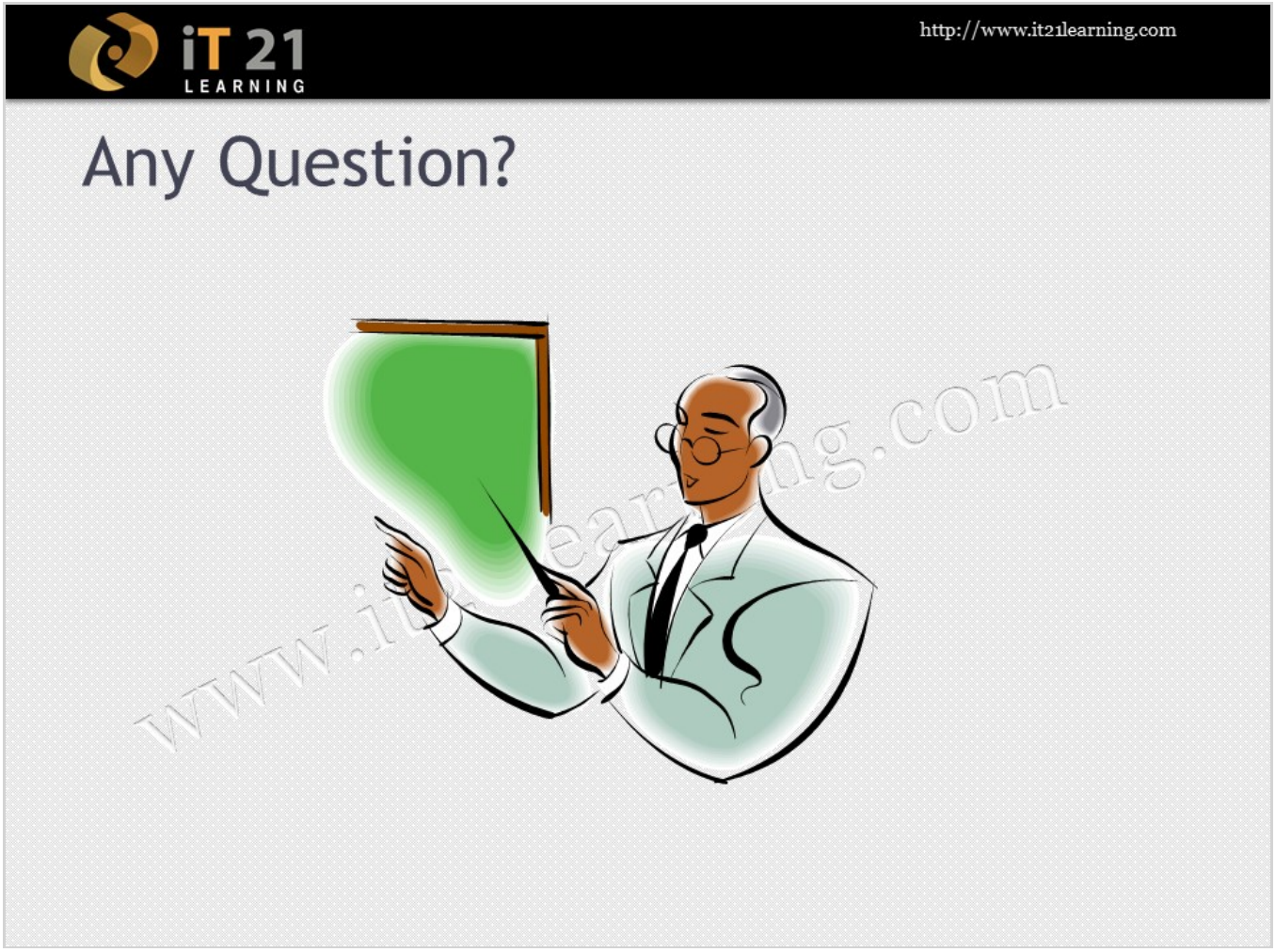
29

Write a File to HDFS

- *Path newFolderPath= new Path("/user/data");*
- *FileSystem hdfs =FileSystem.get(new Configuration());*
  if(hdfs.exists(newFolderPath))
  {
  hdfs.delete(newFolderPath, true);
  }
- *hdfs.mkdirs(newFolderPath);*
- *Path newFilePath=new Path("/user/data/newFile.txt");*
  StringBuilder sb=new StringBuilder();
  for(int i=1;i<=5;i++)
  {
  sb.append("Data");
  sb.append(i);
  sb.append("\n");
  }
  byte[] byt=sb.toString().getBytes();
  FSDataOutputStream fsOutStream = hdfs.create(newFilePath);
  fsOutStream.write(byt);
- *fsOutStream.close();*

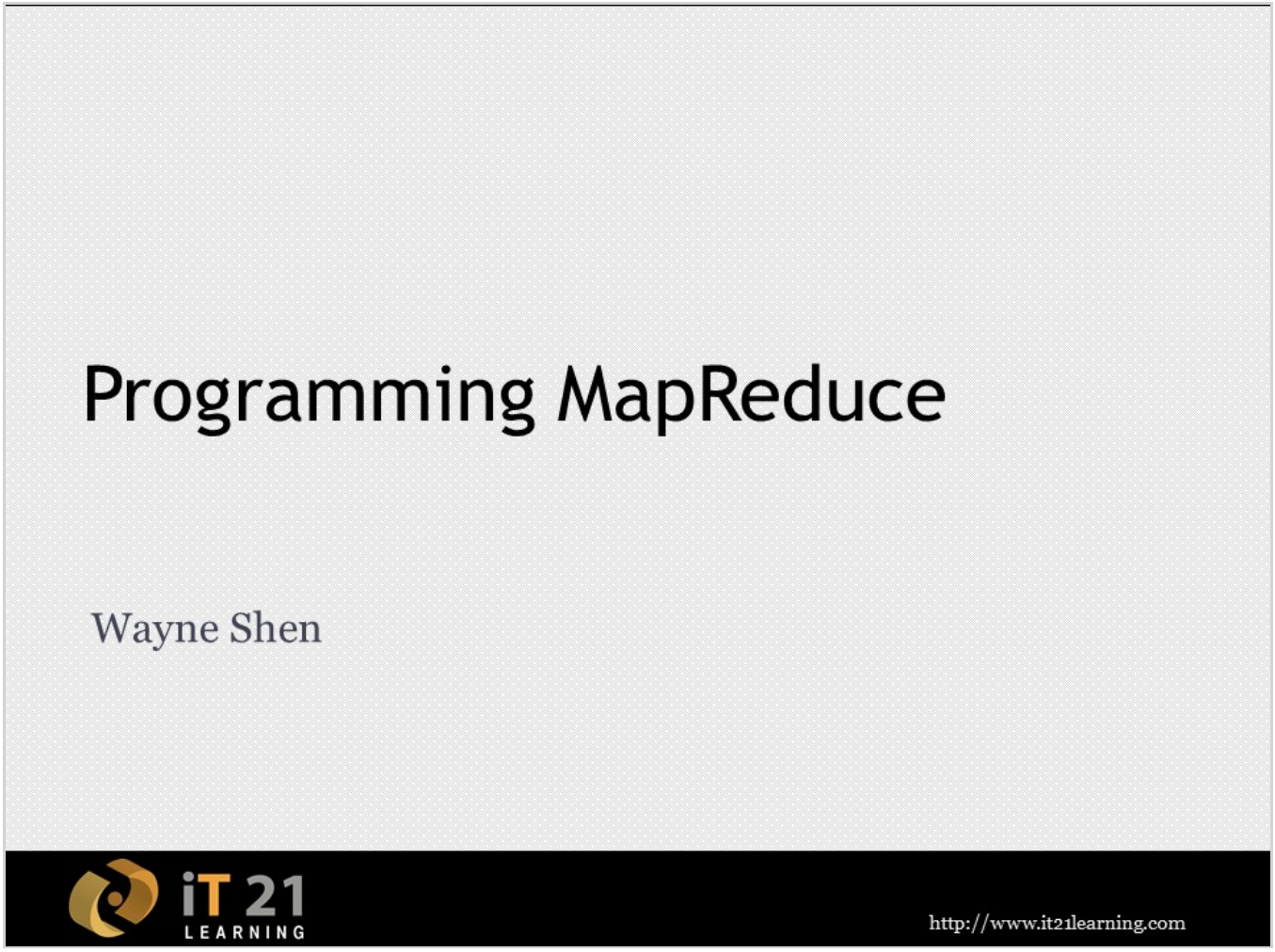
30

Any Question?

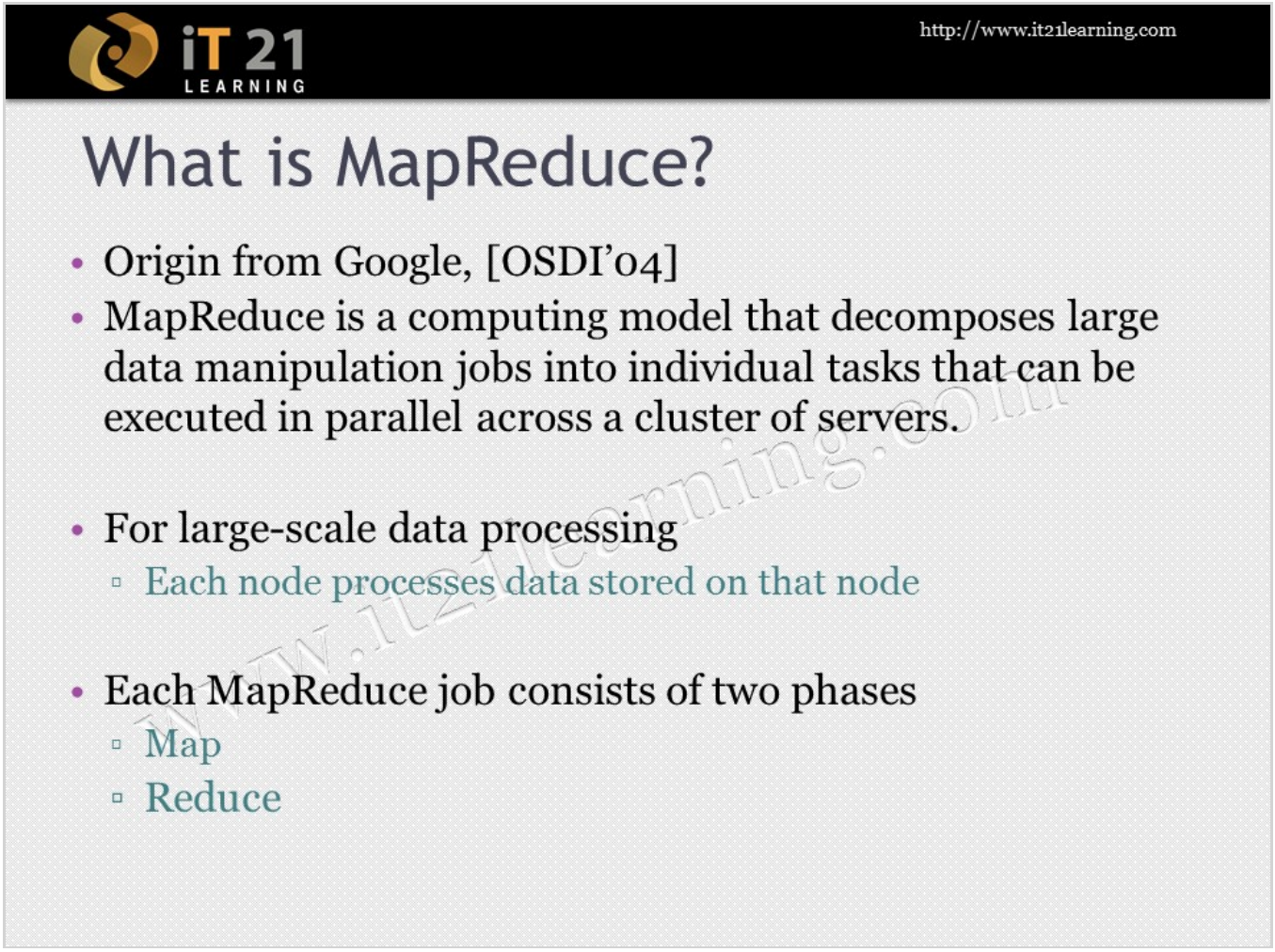


31

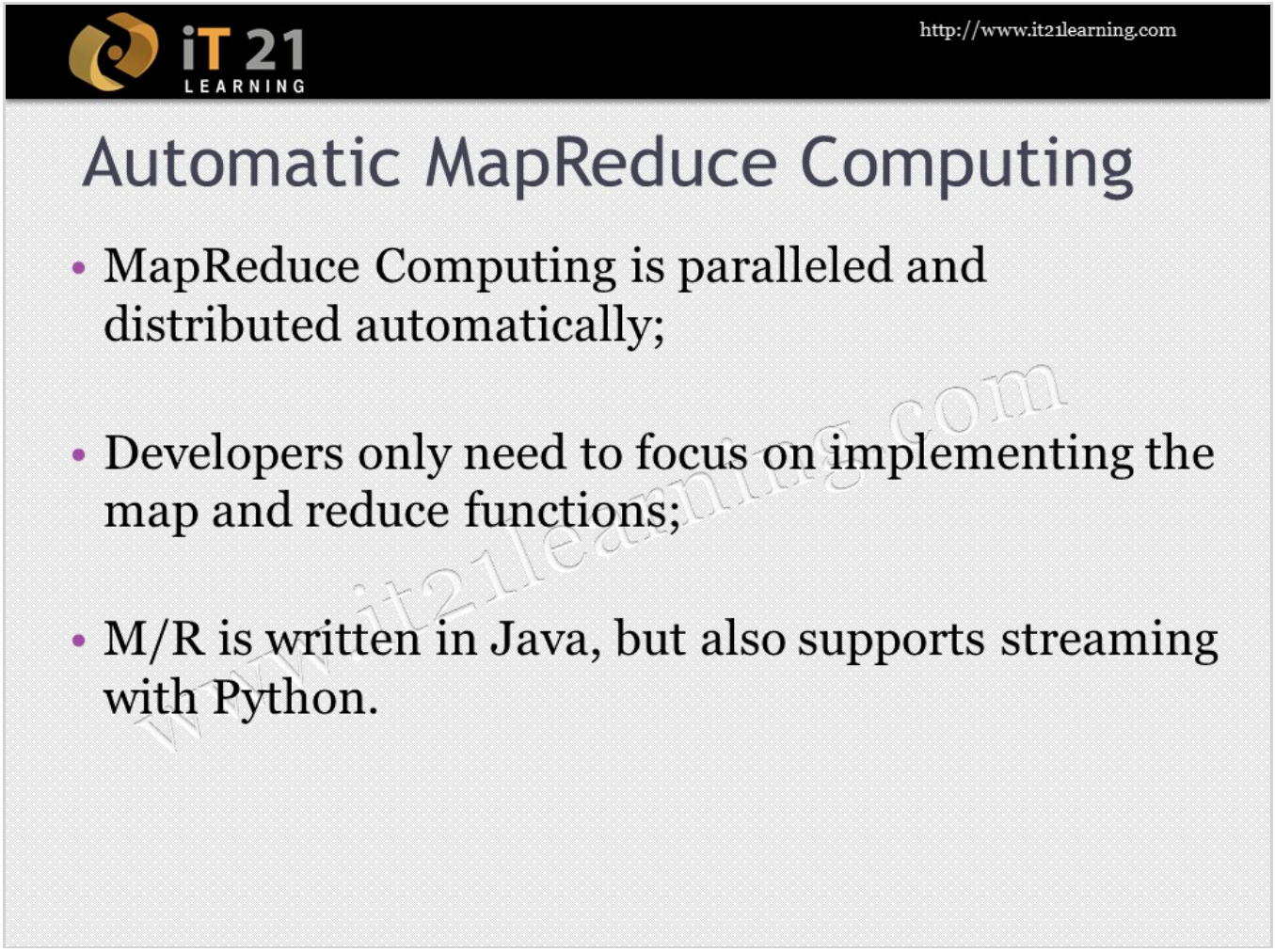
Programming MapReduce


Programming MapReduce
Wayne Shen
iT 21
LEARNING
http://www.it21learning.com

1

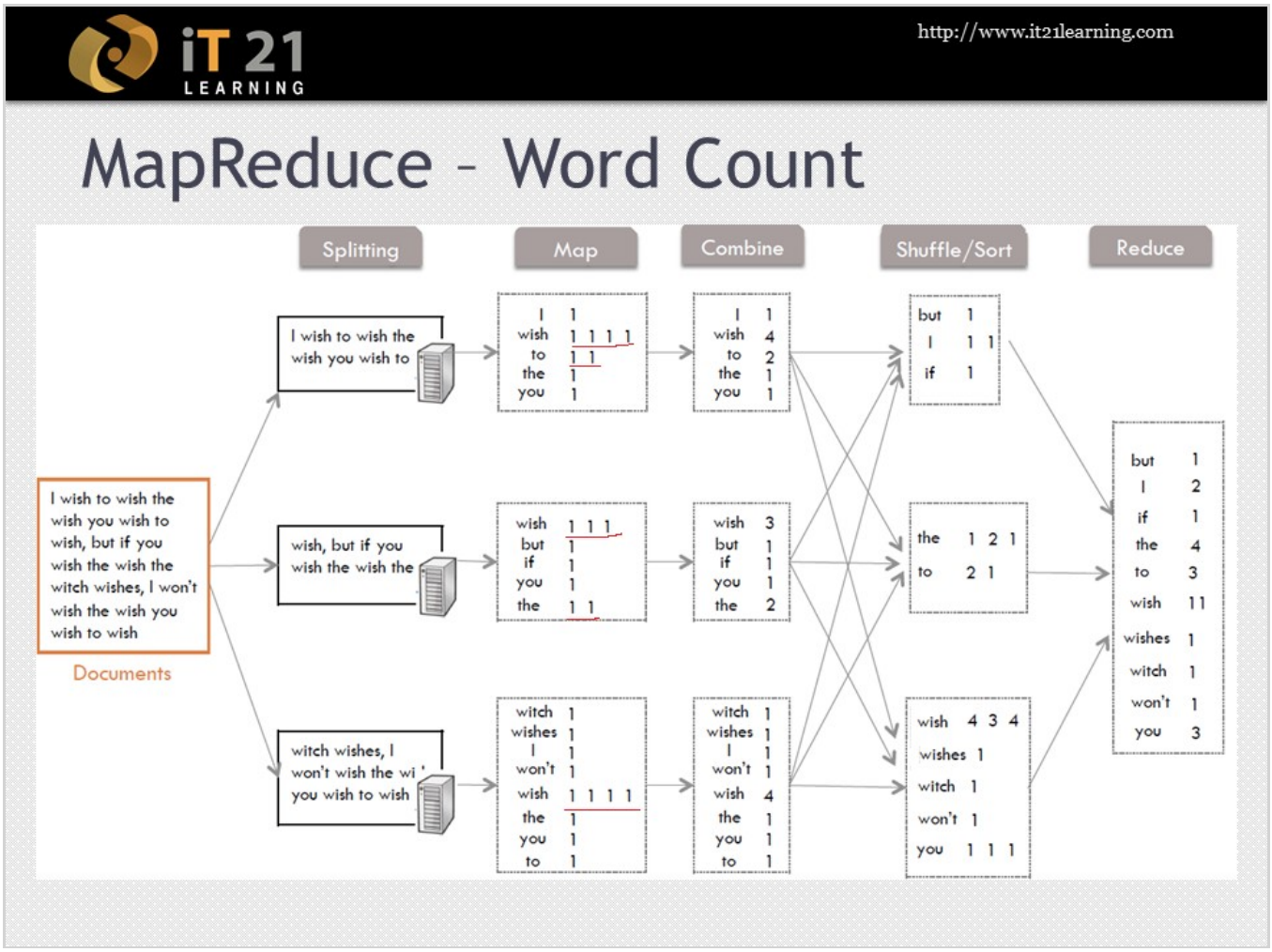

iT 21 LEARNING
http://www.it21learning.com
What is MapReduce?
• Origin from Google, [OSDI'04]
• MapReduce is a computing model that decomposes large data manipulation jobs into individual tasks that can be executed in parallel across a cluster of servers.
• For large-scale data processing
▫ Each node processes data stored on that node
• Each MapReduce job consists of two phases
▫ Map
▫ Reduce
www.it21learning.com

2


iT 21 LEARNING
http://www.it21learning.com
Automatic MapReduce Computing
• MapReduce Computing is paralleled and distributed automatically;
• Developers only need to focus on implementing the map and reduce functions;
• M/R is written in Java, but also supports streaming with Python.
www.it21learning.com

3

MapReduce – Word Count



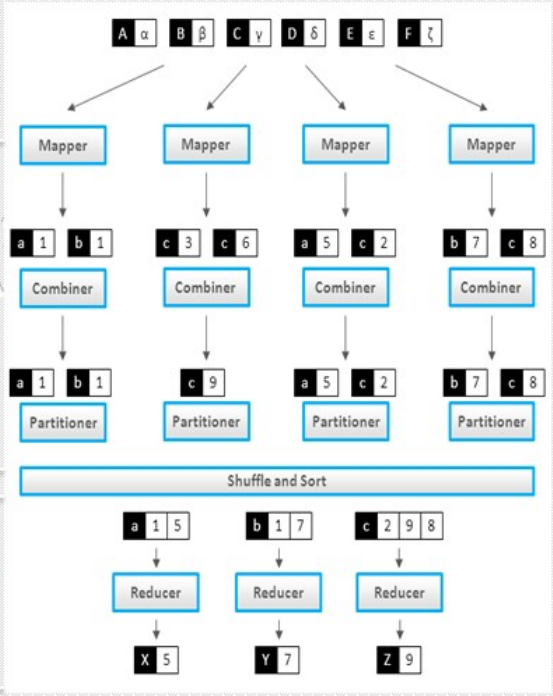
4

MapReduce Framework

Definition:

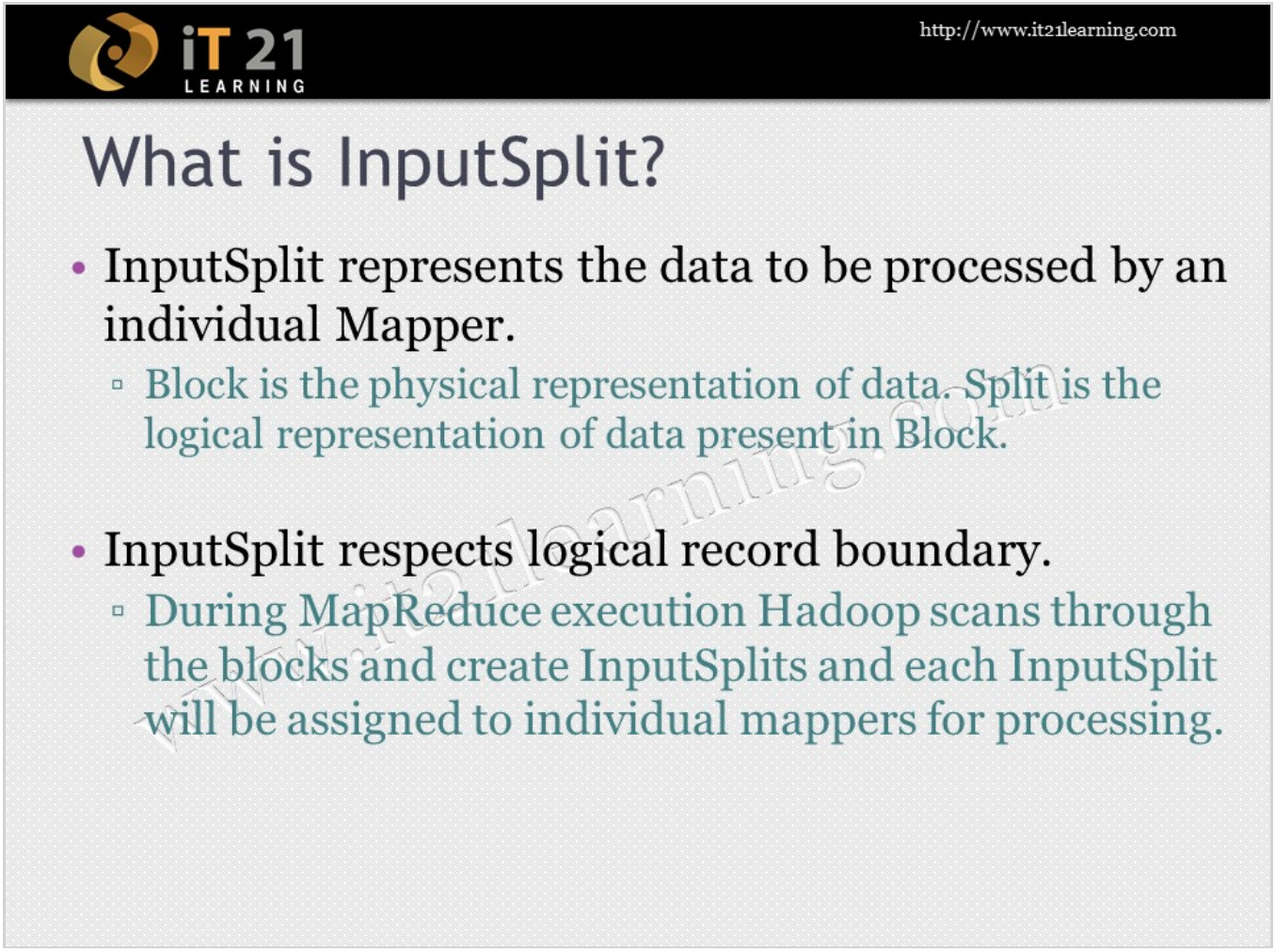
- map: (K1,V1) → list (K2,V2)
- reduce: (K2,list(V2)) → list (K3,V3)

- Mapper (map)
- Combiner (optional)
  Can only be used when commutative(a.b = b.a) and associative {a.(b.c) = (a.b).c}
- Partitioner
  Default is the HashPartitioner that performs a modulo against the numOfPartitions to return the partition number
- Shuffle and Sort
- Reducer (reduce)



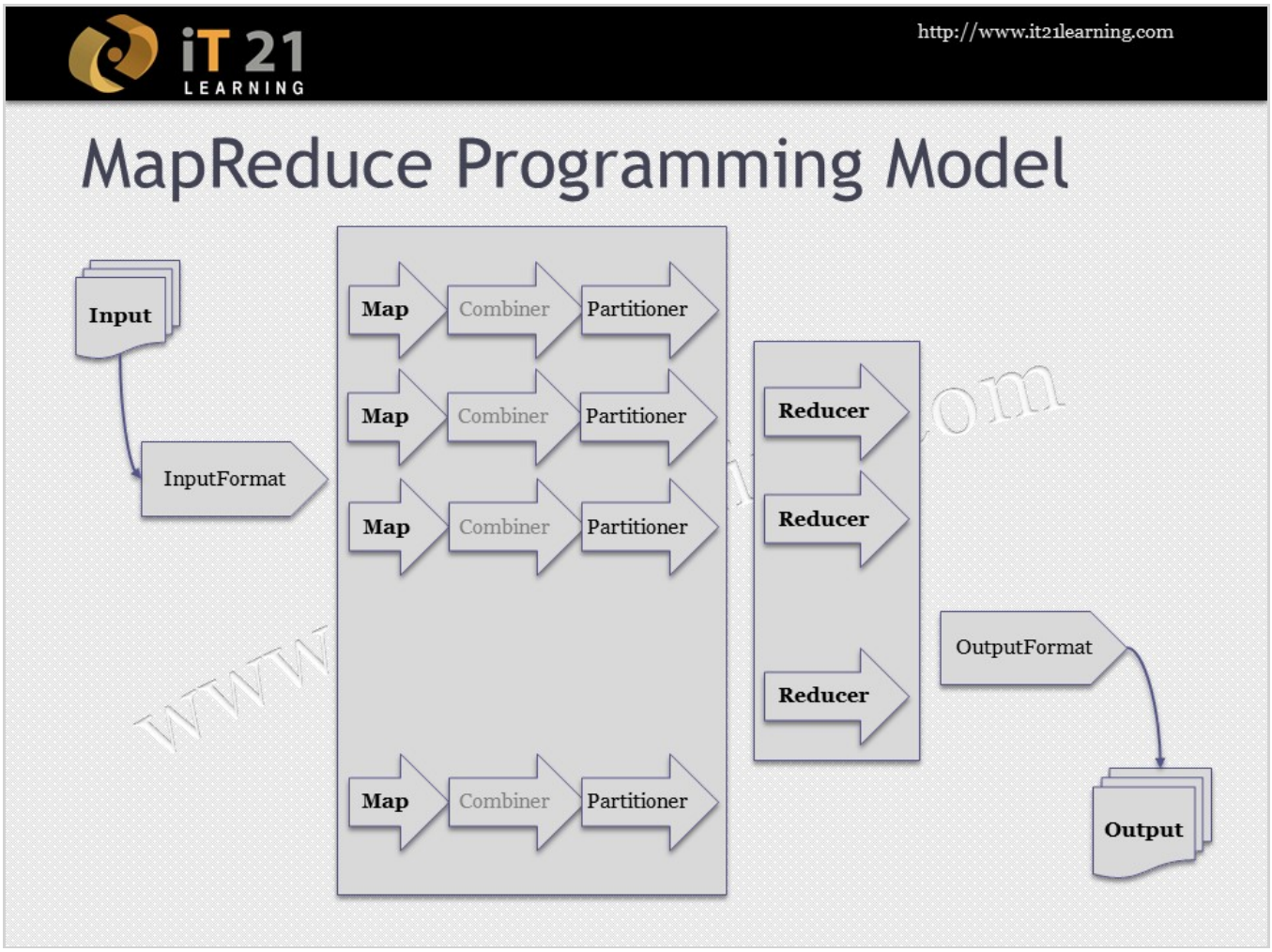
5

What is InputSplit?

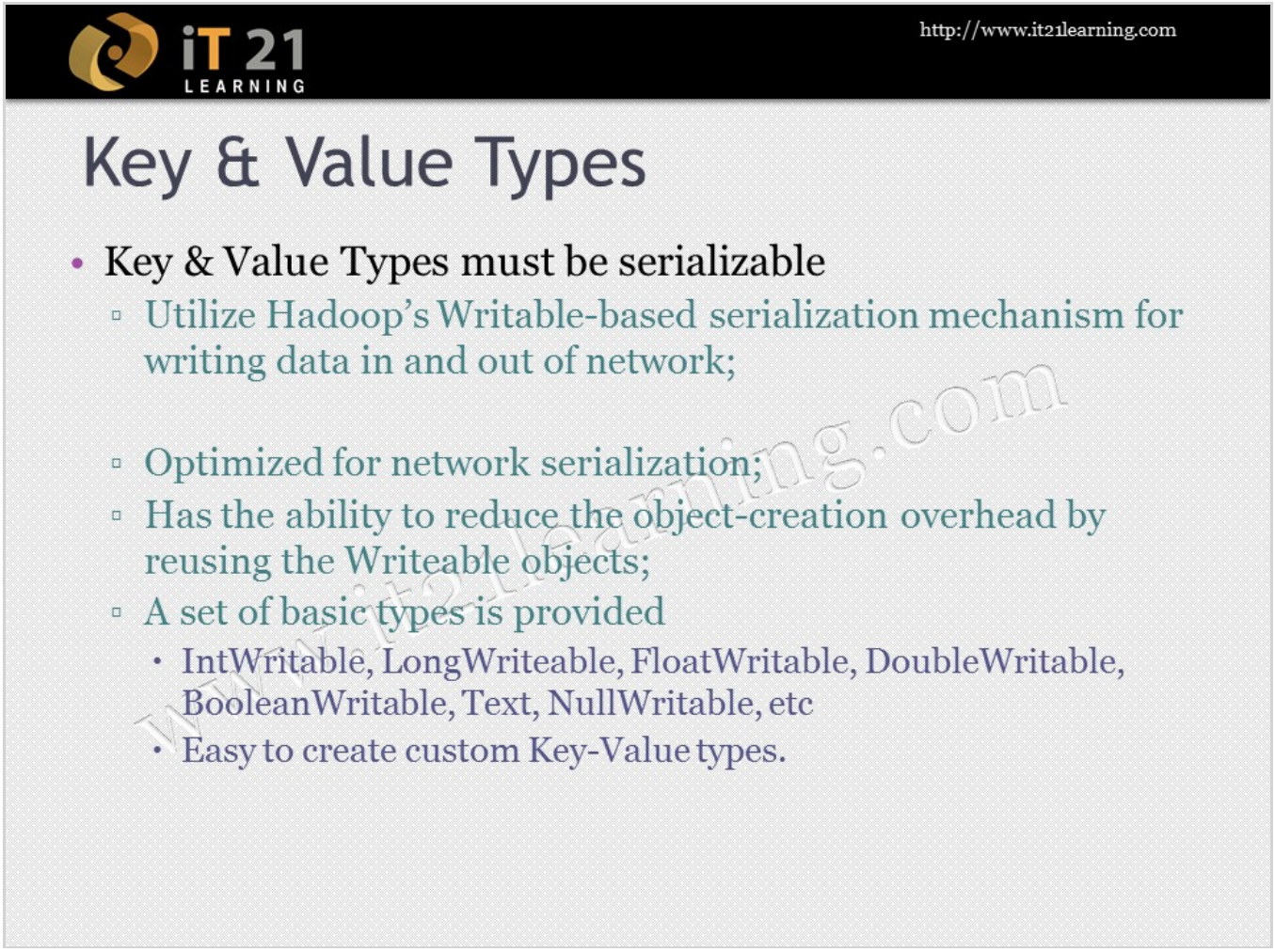


6

MapReduce Programming Model

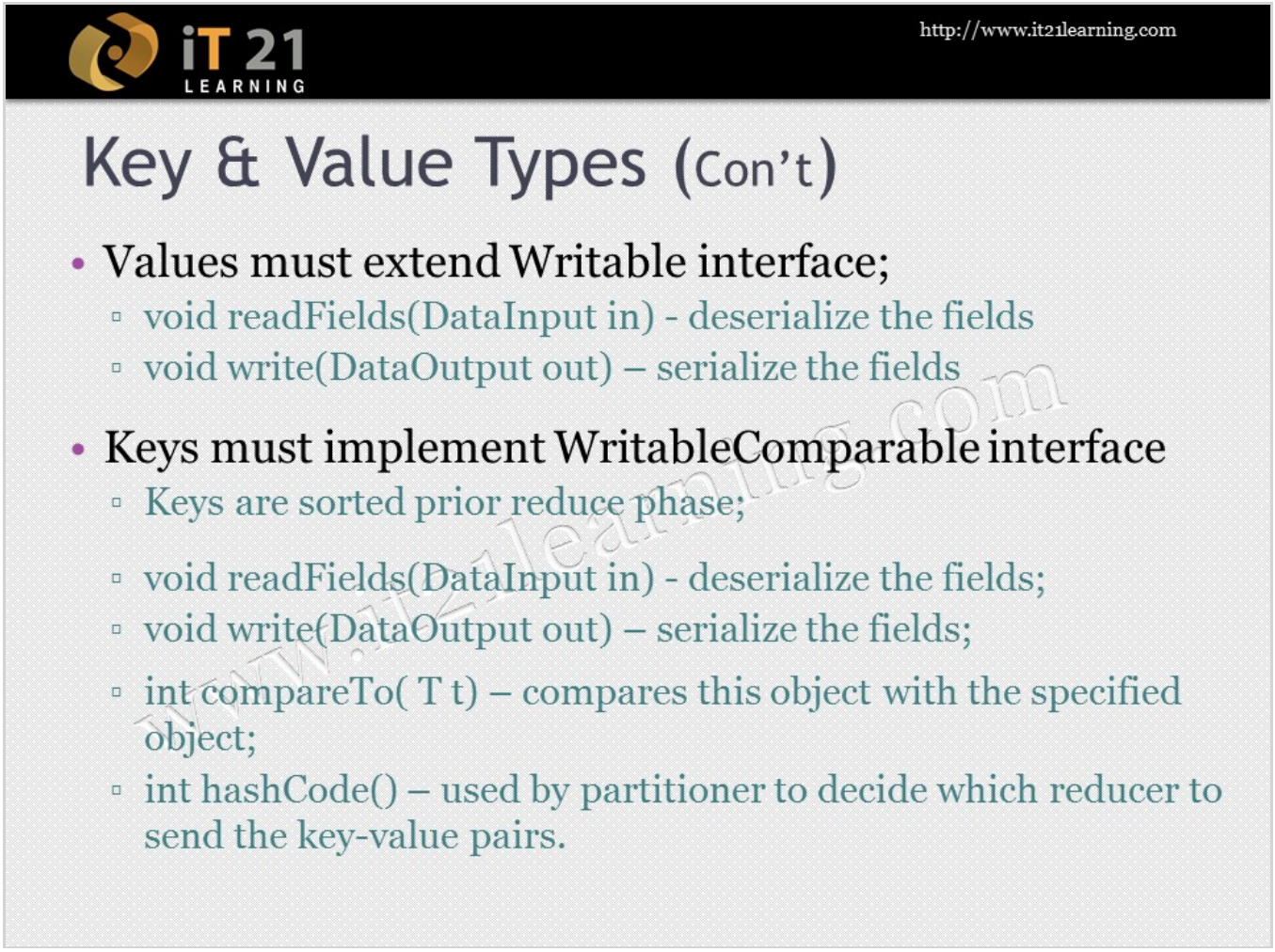


7


iT 21 LEARNING
http://www.it21learning.com
Key & Value Types
• Key & Value Types must be serializable
▫ Utilize Hadoop's Writable-based serialization mechanism for writing data in and out of network;
▫ Optimized for network serialization;
▫ Has the ability to reduce the object-creation overhead by reusing the Writeable objects;
▫ A set of basic types is provided
• IntWritable, LongWriteable, FloatWritable, DoubleWritable, BooleanWritable, Text, NullWritable, etc
• Easy to create custom Key-Value types.
www.it21learning.com

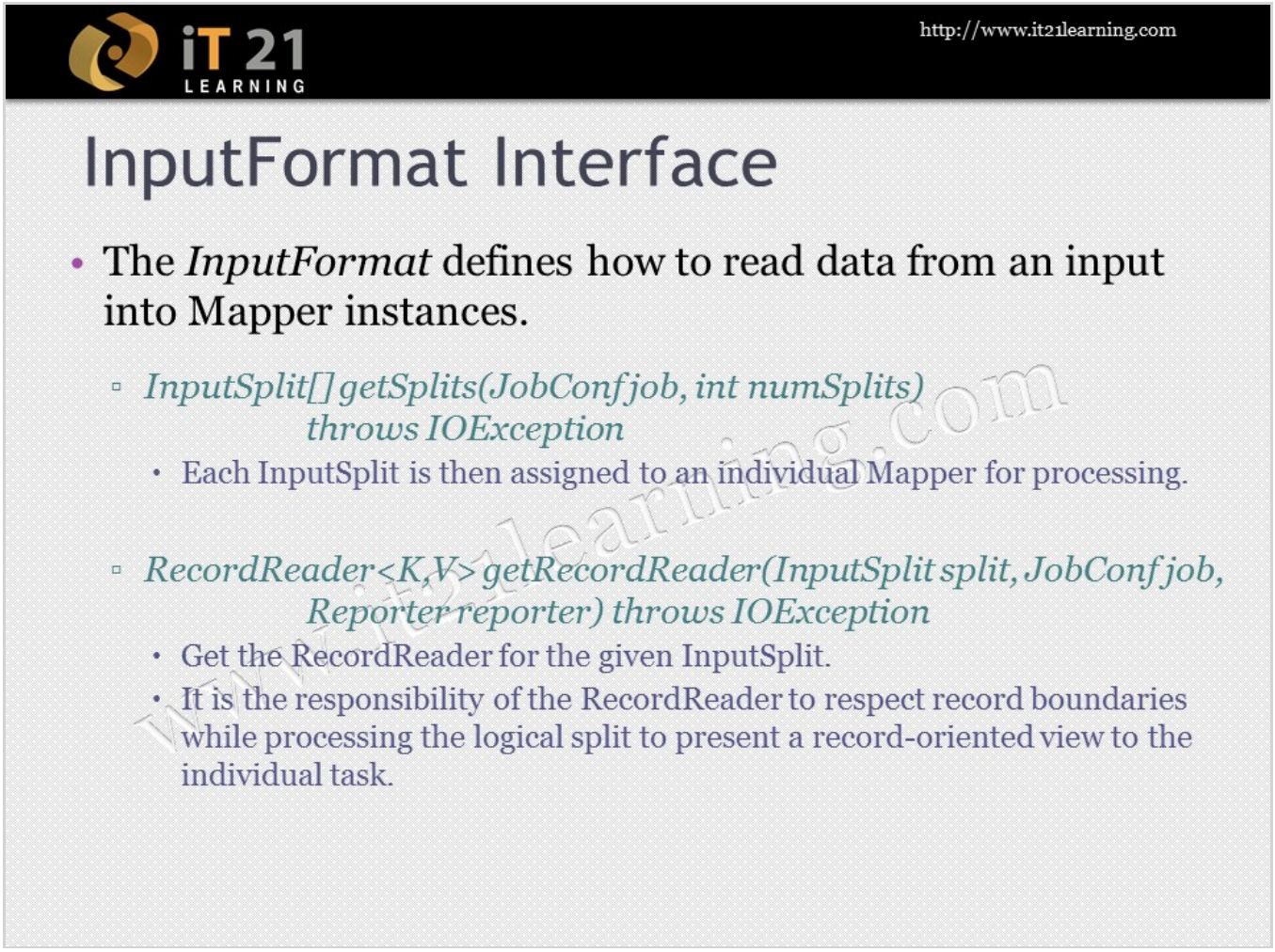
8

Key & Value Types (Con't)



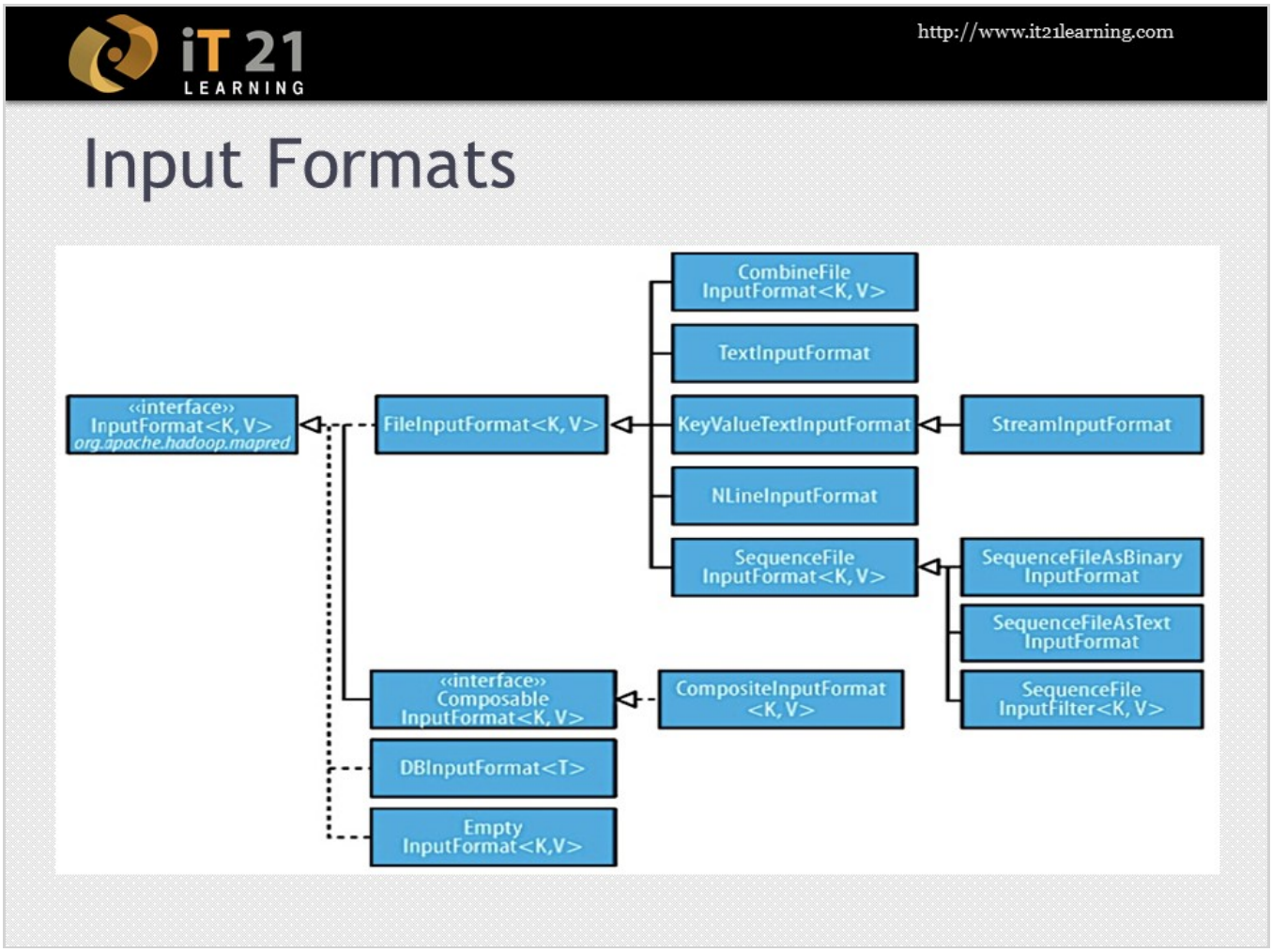
9

InputFormat Interface

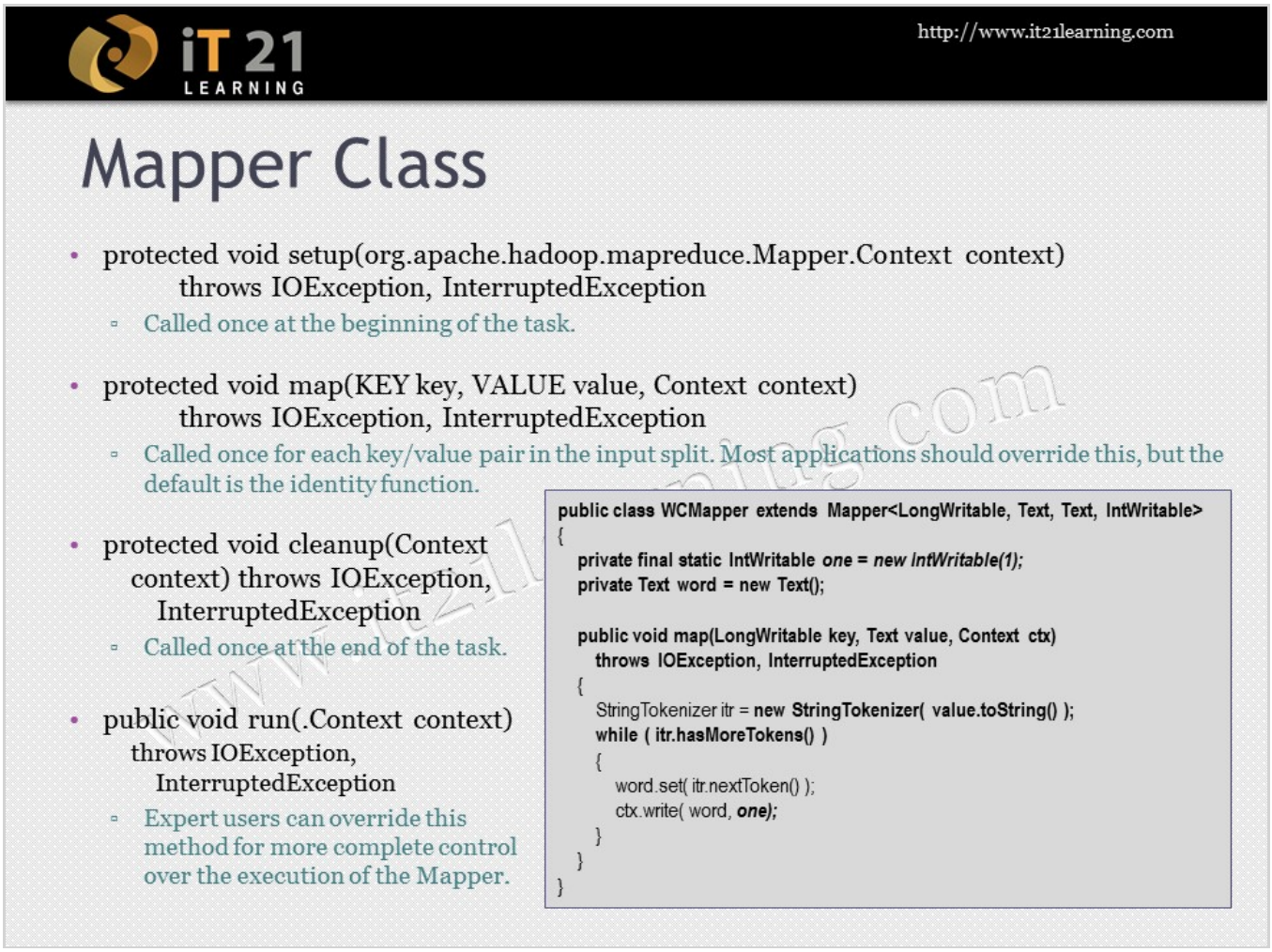


10

Input Formats

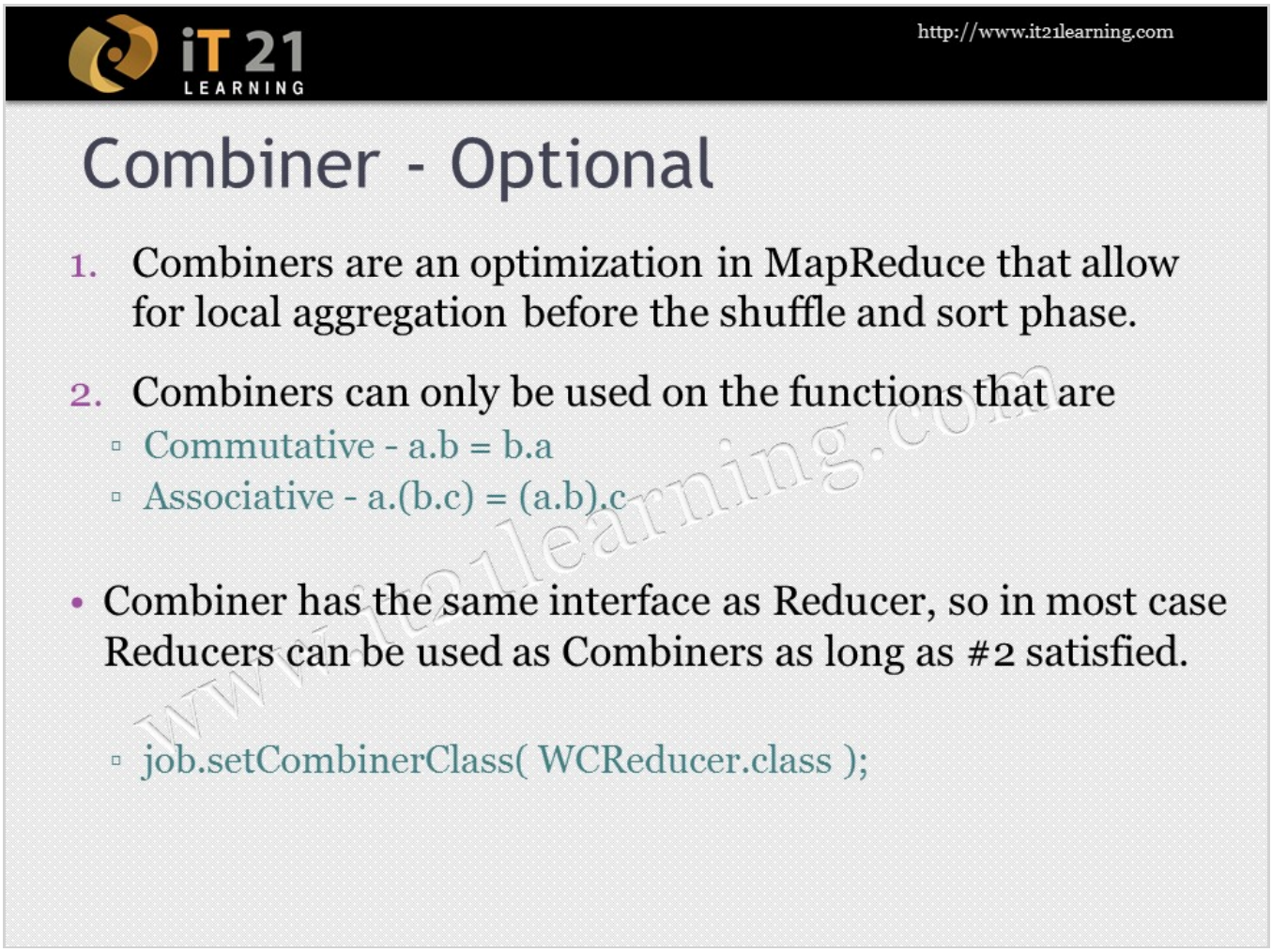


11


iT 21 LEARNING
http://www.it21learning.com
Mapper Class
protected void setup(org.apache.hadoop.mapreduce.Mapper.Context context) throws IOException, InterruptedException
Called once at the beginning of the task.
protected void map(KEY key, VALUE value, Context context) throws IOException, InterruptedException
Called once for each key/value pair in the input split. Most applications should override this, but the default is the identity function.
protected void cleanup(Context context) throws IOException, InterruptedException
Called once at the end of the task.
public void run(.Context context) throws IOException, InterruptedException
Expert users can override this method for more complete control over the execution of the Mapper.
public class WCMapper extends Mapper<LongWritable, Text, Text, IntWritable>
{
private final static IntWritable one = new IntWritable(1);
private Text word = new Text();
public void map(LongWritable key, Text value, Context ctx)
throws IOException, InterruptedException
{
StringTokenizer itr = new StringTokenizer(value.toString());
while (itr.hasMoreTokens())
{
word.set(itr.nextToken());
ctx.write(word, one);
}
}
}

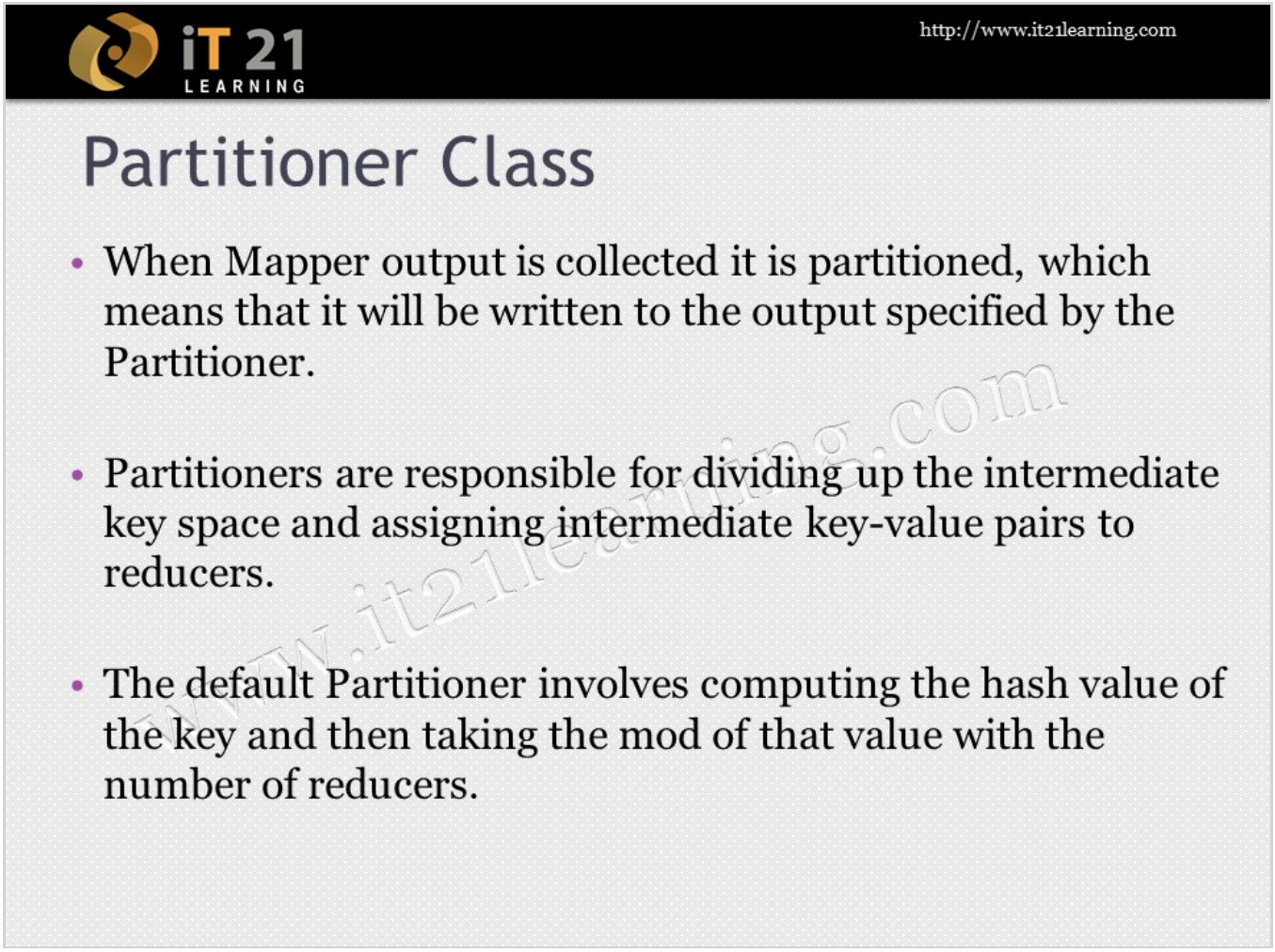
12

Combiner - Optional



13

Partitioner Class



14

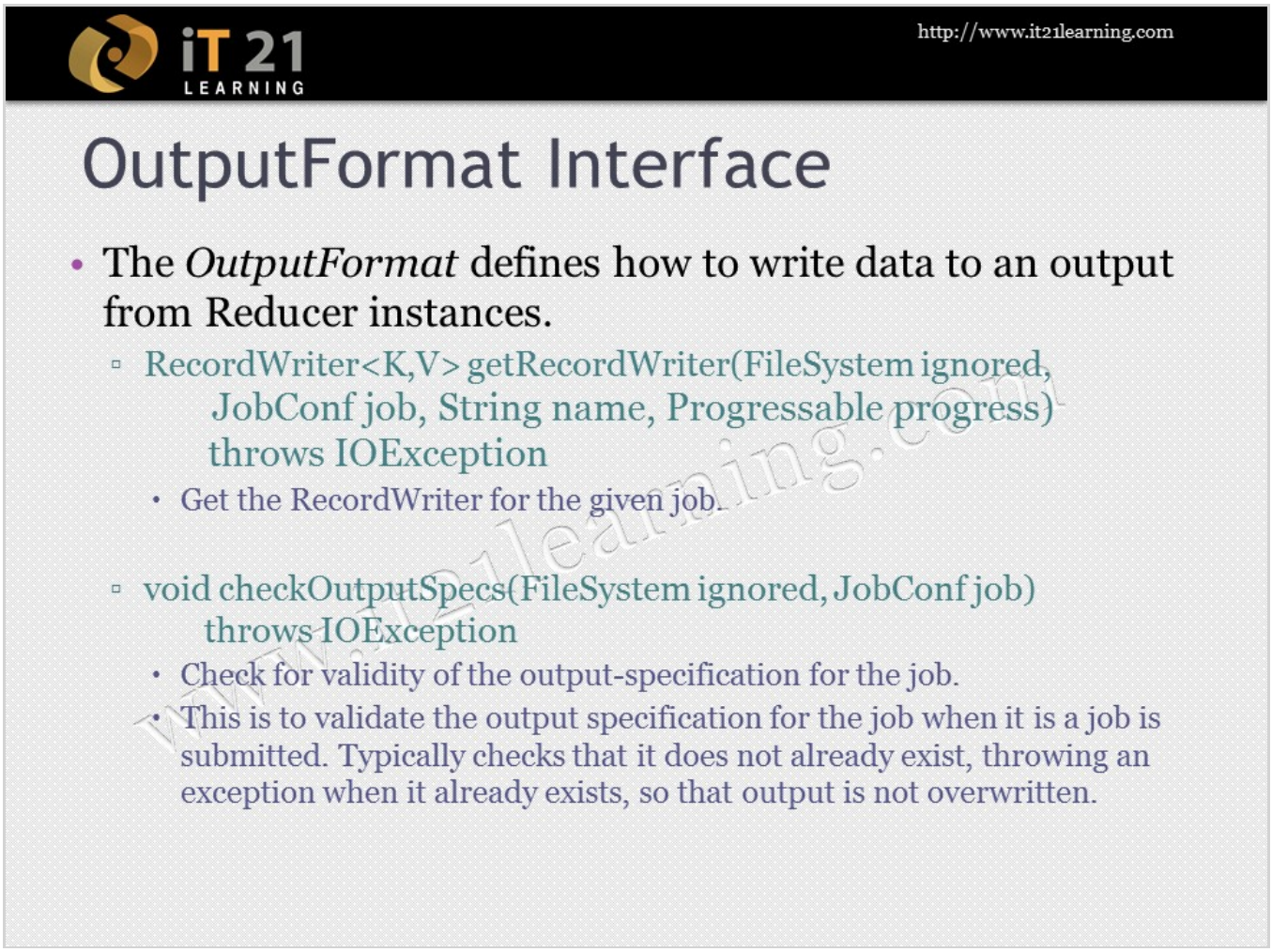
Reducer Class

- protected void setup(org.apache.hadoop.mapreduce.Mapper.Context context) throws IOException, InterruptedException
  - Called once at the beginning of the task.
- protected void reduce(KEY key, Iterable<VALUE> values, Context context) throws IOException, InterruptedException
  - Called once for each key. Most applications should override this, but the default is the identity function.
- protected void cleanup(Context context) throws IOException, InterruptedException
  - Called once at the end of the task.
- public void run(.Context context) throws IOException, InterruptedException
  - Advanced application writers can use the run method to control how the reduce task works.

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

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

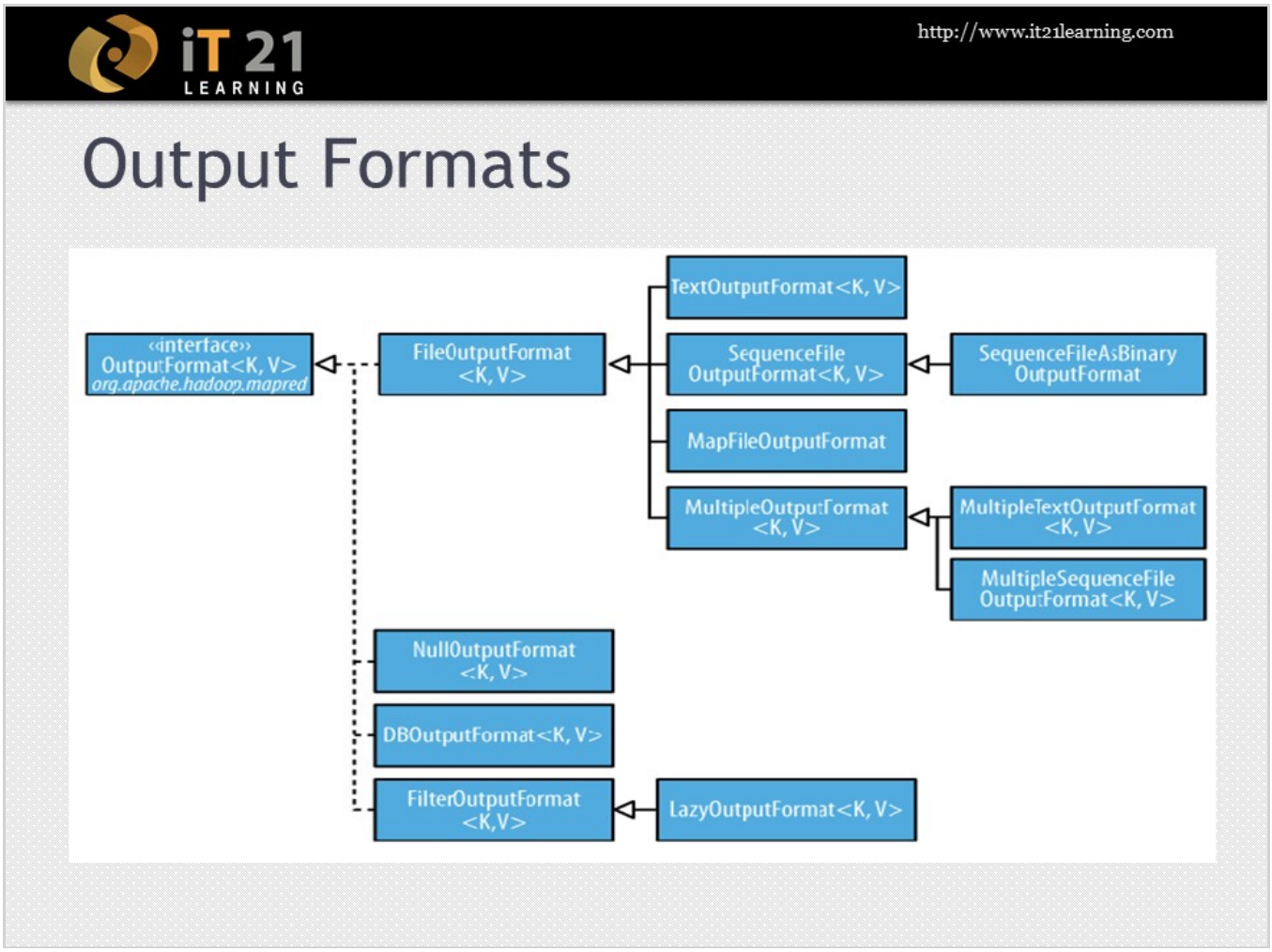
15

OutputFormat Interface



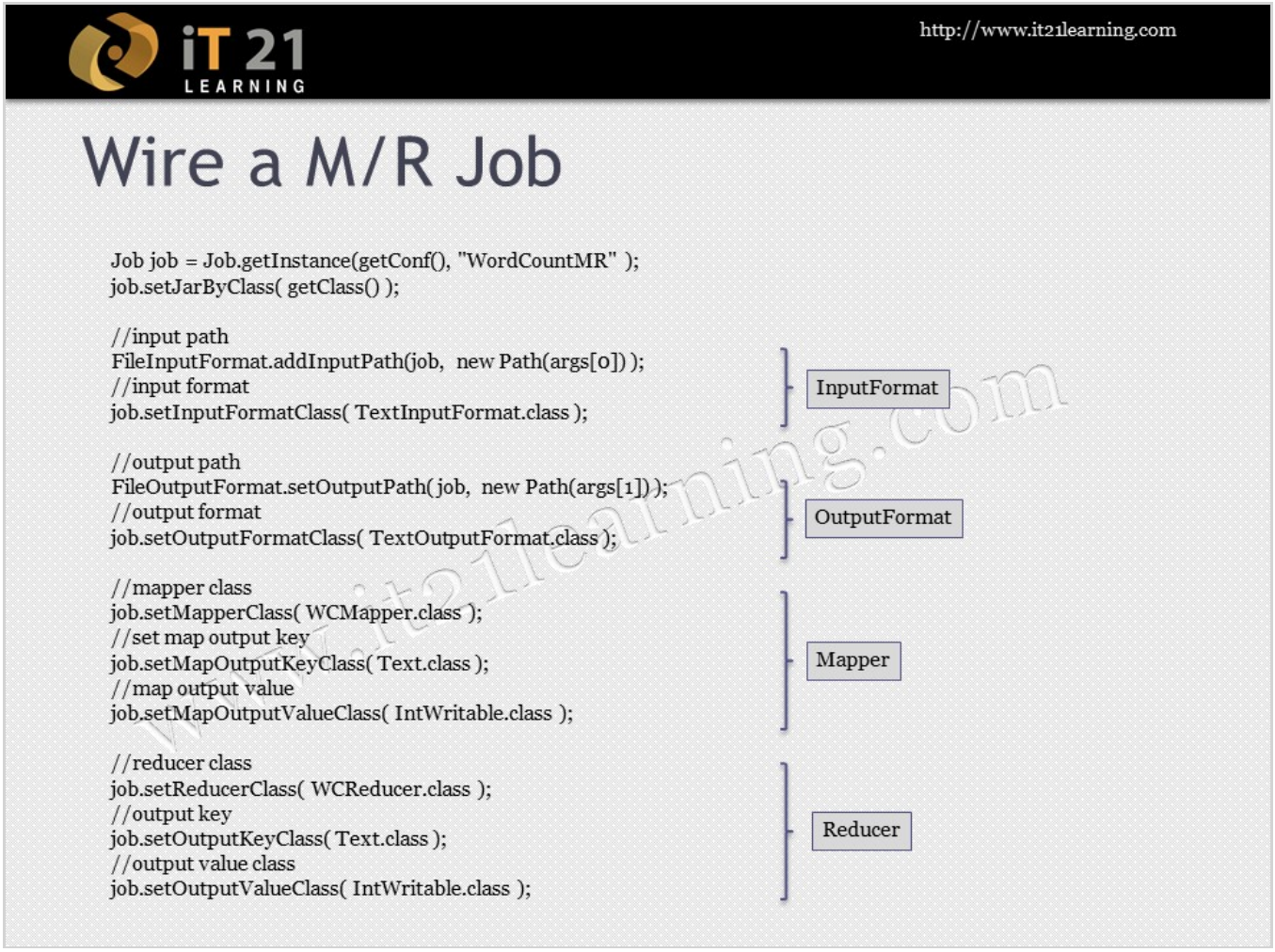
16

Output Formats

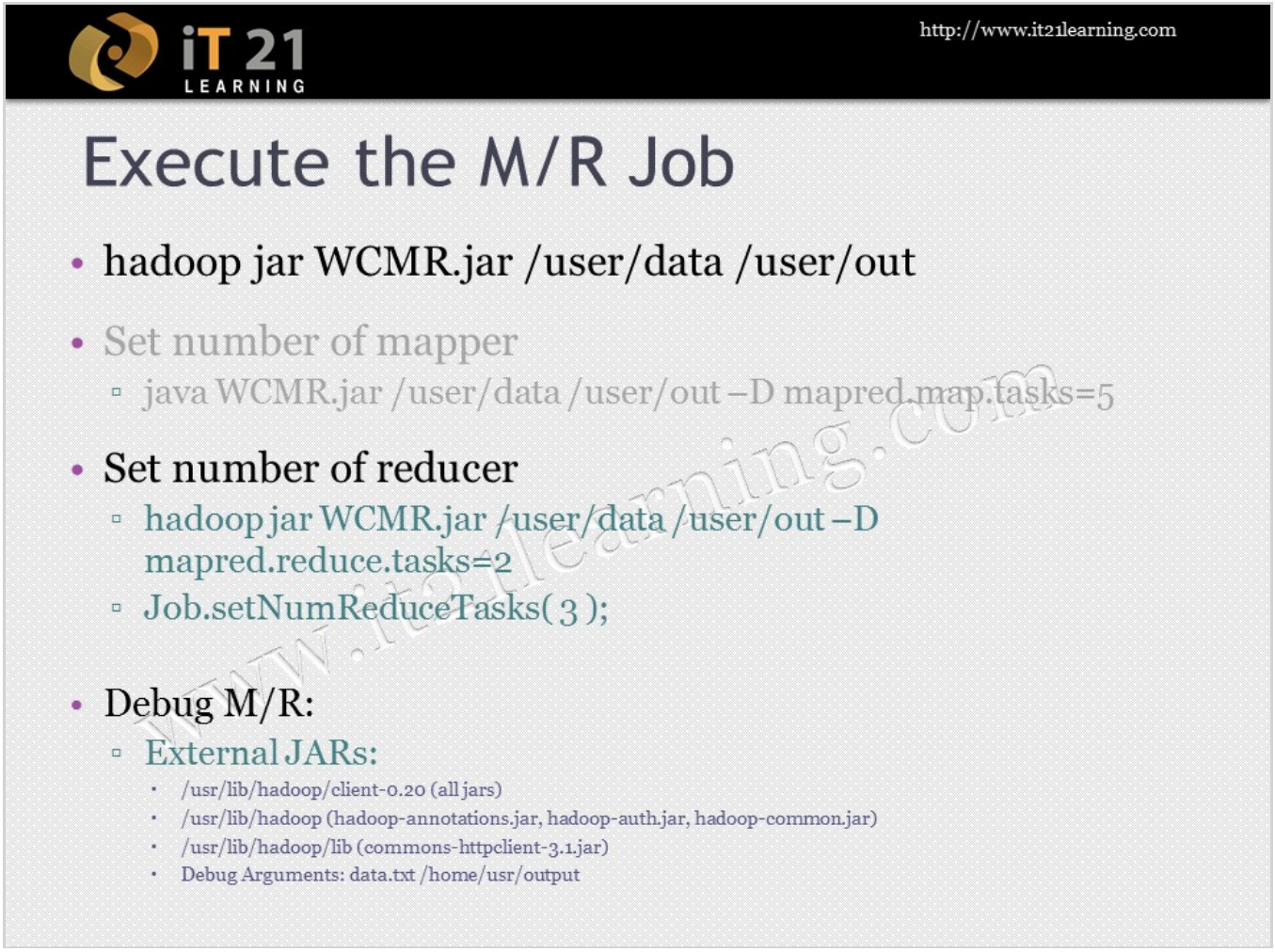


17

Wire a M/R Job

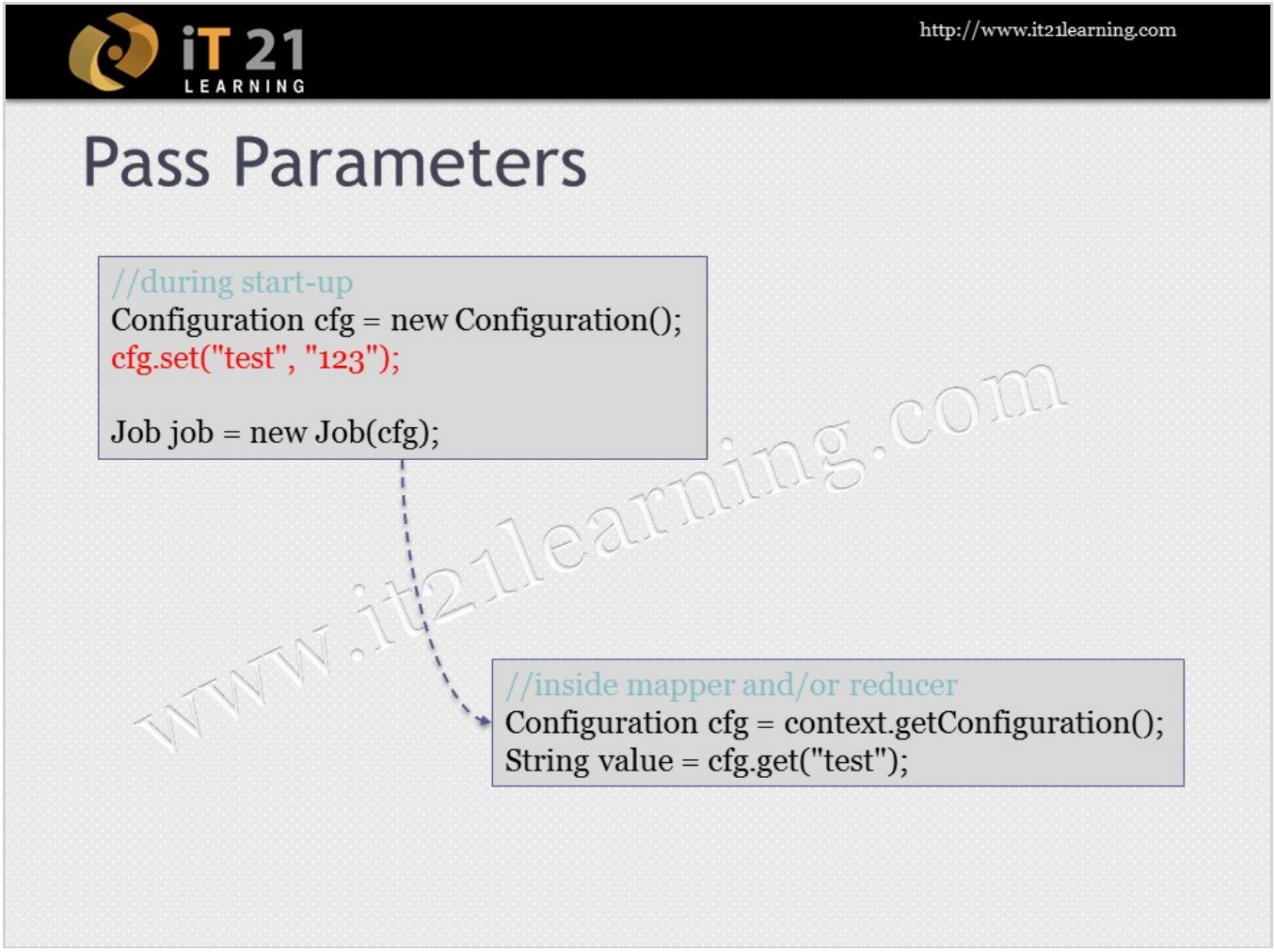


18

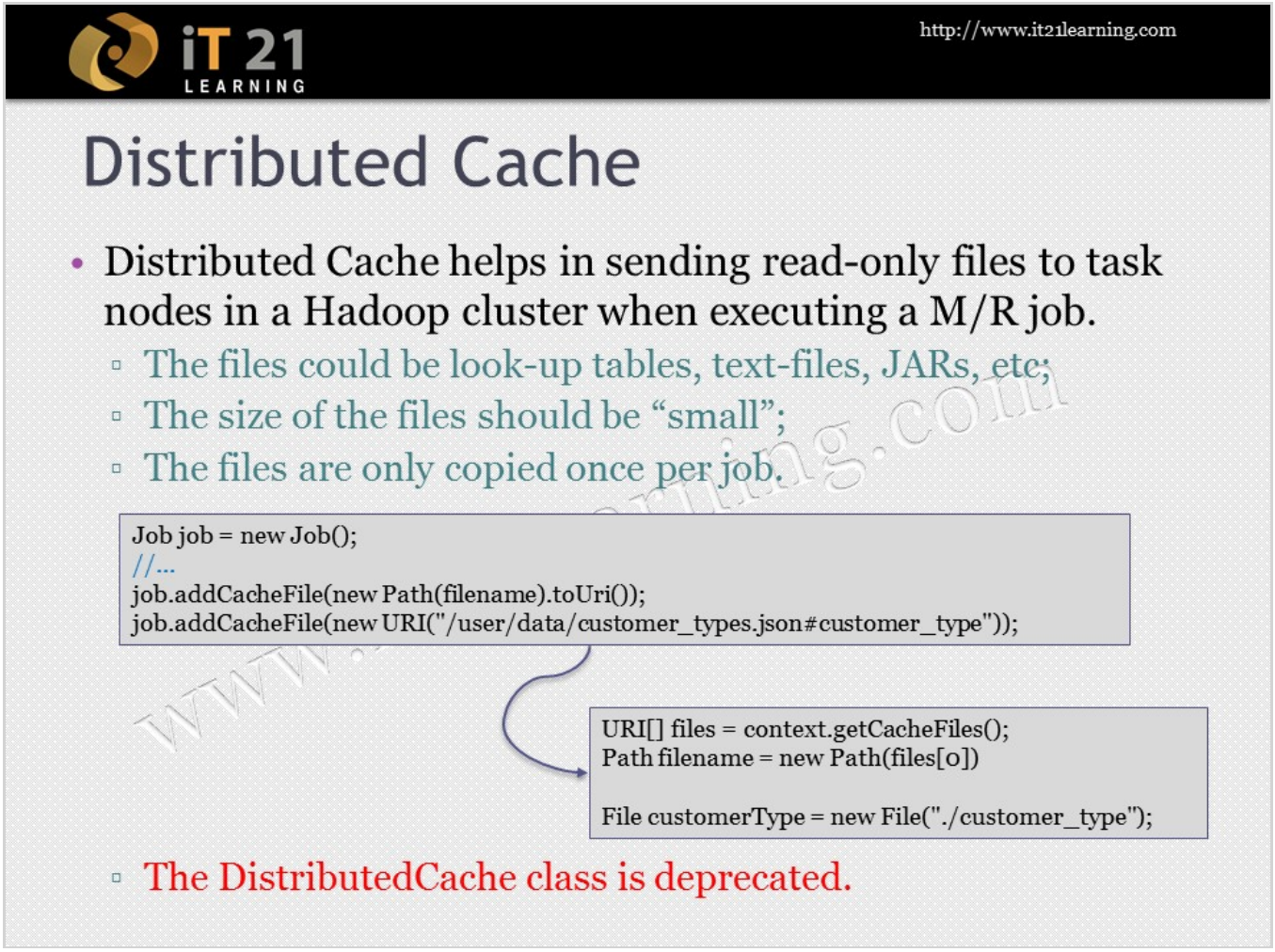

iT 21 LEARNING
http://www.it21learning.com
Execute the M/R Job
• hadoop jar WCMR.jar /user/data /user/out
• Set number of mapper
▫ java WCMR.jar /user/data /user/out –D mapred.map.tasks=5
• Set number of reducer
▫ hadoop jar WCMR.jar /user/data /user/out –D mapred.reduce.tasks=2
▫ Job.setNumReduceTasks(3);
• Debug M/R:
▫ External JARs:
/usr/lib/hadoop/client-0.20 (all jars)
/usr/lib/hadoop (hadoop-annotations.jar, hadoop-auth.jar, hadoop-common.jar)
/usr/lib/hadoop/lib (commons-httpclient-3.1.jar)
Debug Arguments: data.txt /home/usr/output

19

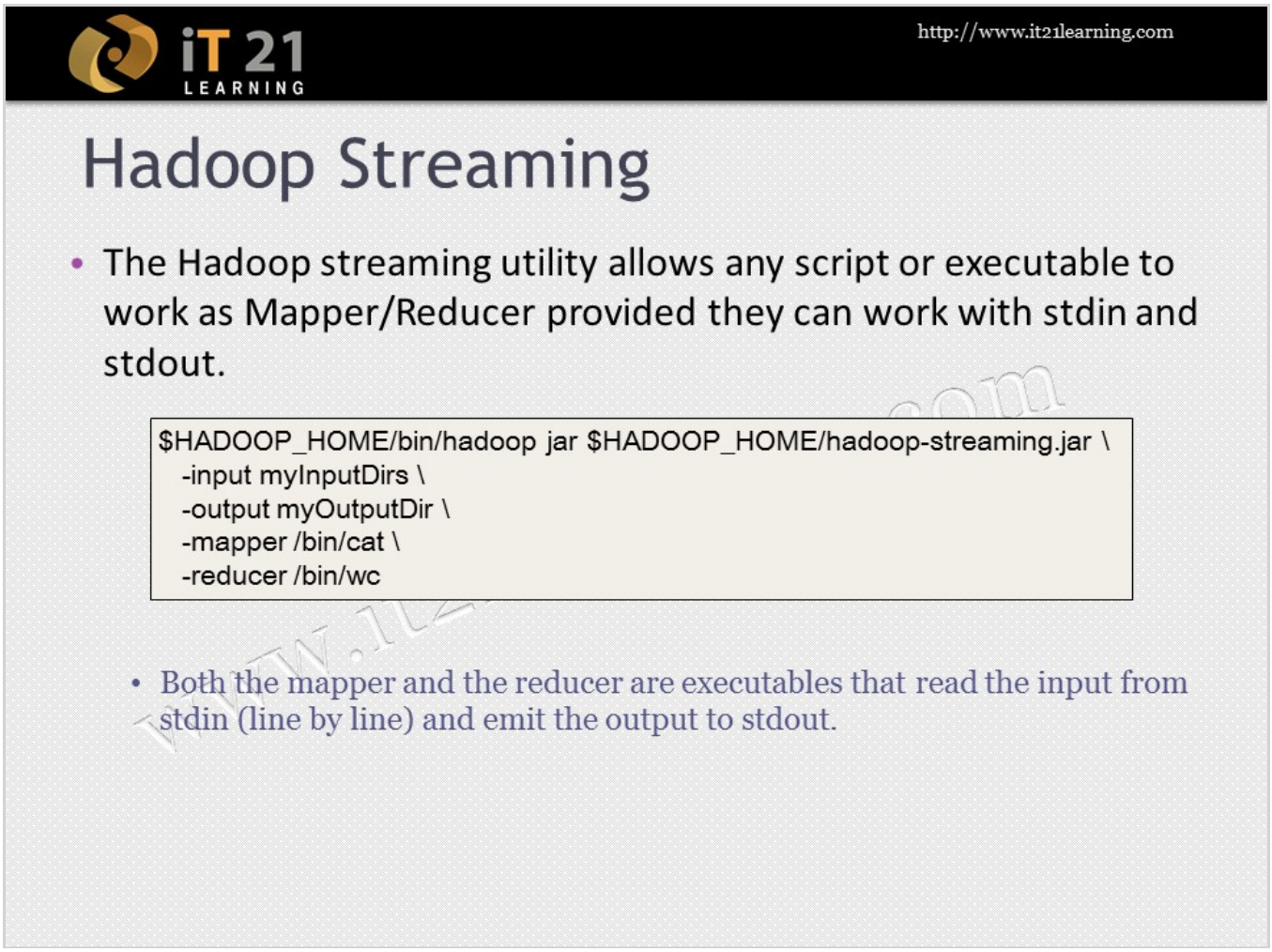
Pass Parameters



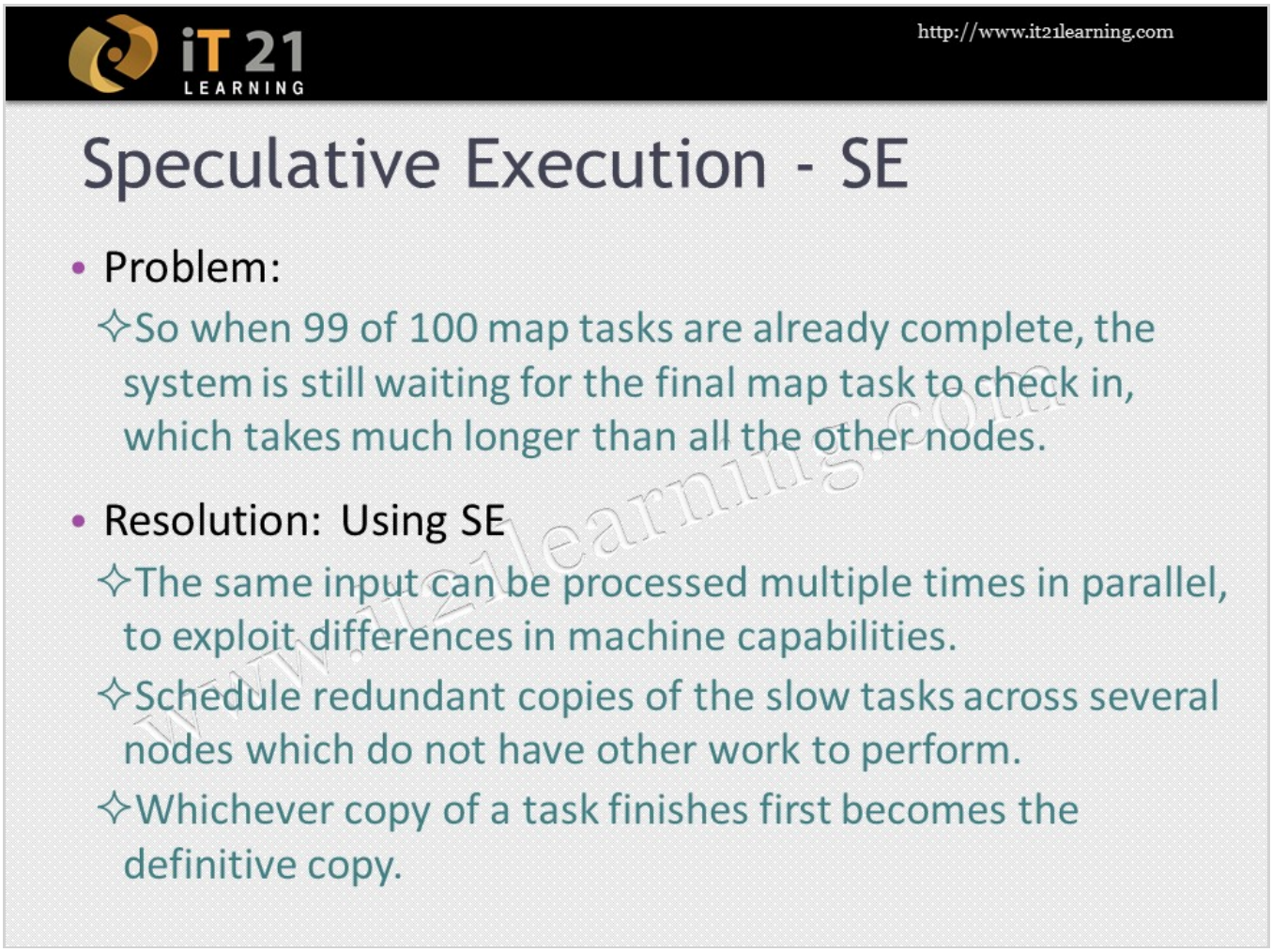
20


iT 21 LEARNING
http://www.it21learning.com
Distributed Cache
Distributed Cache helps in sending read-only files to task nodes in a Hadoop cluster when executing a M/R job.
The files could be look-up tables, text-files, JARs, etc;
The size of the files should be "small";
The files are only copied once per job.
Job job = new Job();
//...
job.addCacheFile(new Path(filename).toUri());
job.addCacheFile(new URI("/user/data/customer_types.json#customer_type"));
URI[] files = context.getCacheFiles();
Path filename = new Path(files[0])
File customerType = new File("./customer_type");
The DistributedCache class is deprecated.

21

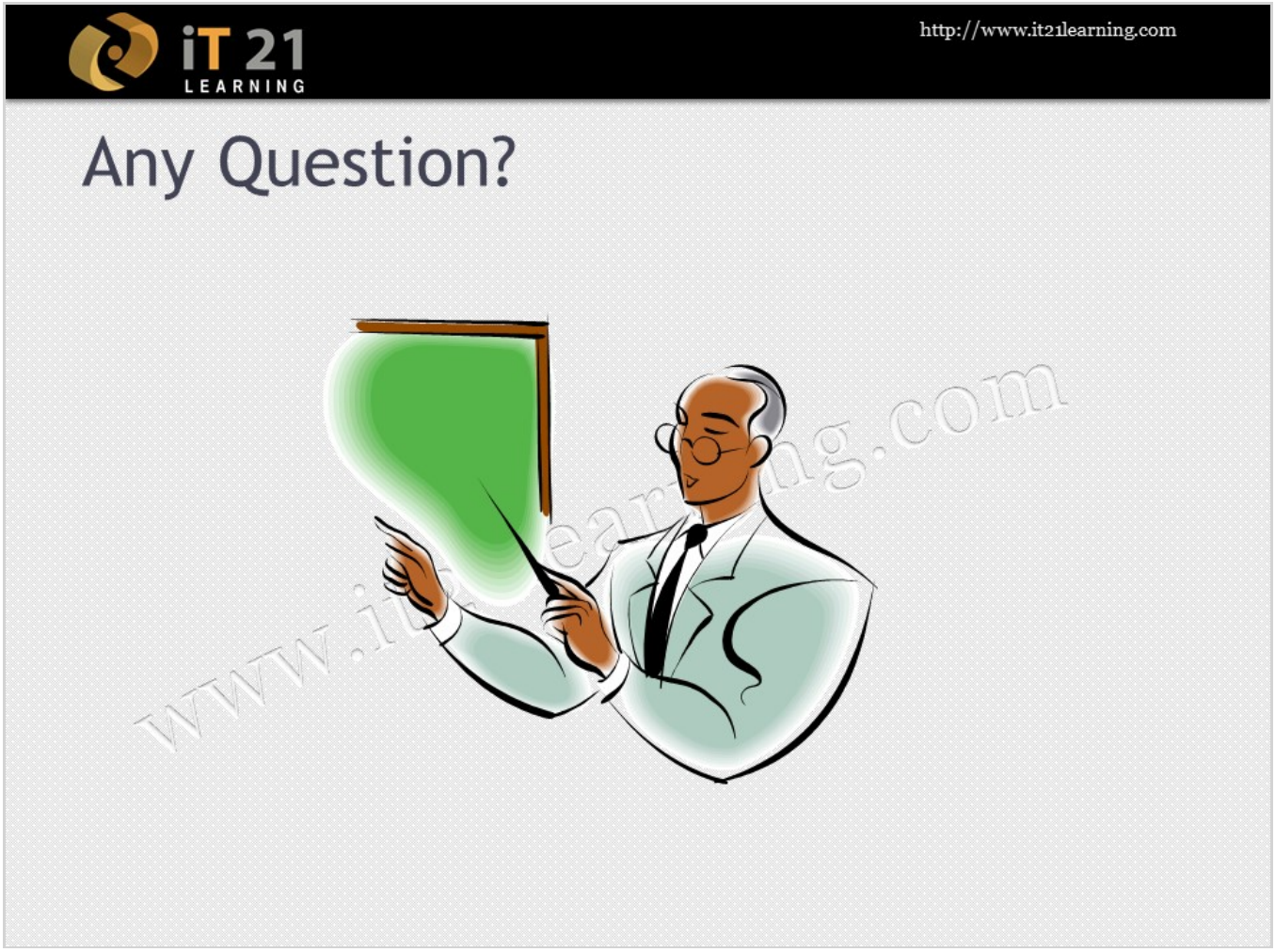

iT 21 LEARNING
http://www.it21learning.com
Hadoop Streaming
• The Hadoop streaming utility allows any script or executable to work as Mapper/Reducer provided they can work with stdin and stdout.
$HADOOP_HOME/bin/hadoop jar $HADOOP_HOME/hadoop-streaming.jar \
-input myInputDirs \
-output myOutputDir \
-mapper /bin/cat \
-reducer /bin/wc
• Both the mapper and the reducer are executables that read the input from stdin (line by line) and emit the output to stdout.

22


iT 21 LEARNING
http://www.it21learning.com
Speculative Execution - SE
• Problem:
✧So when 99 of 100 map tasks are already complete, the system is still waiting for the final map task to check in, which takes much longer than all the other nodes.
• Resolution: Using SE
✧The same input can be processed multiple times in parallel, to exploit differences in machine capabilities.
✧Schedule redundant copies of the slow tasks across several nodes which do not have other work to perform.
✧Whichever copy of a task finishes first becomes the definitive copy.

23

Any Question?



24

Mastering Apache Hive

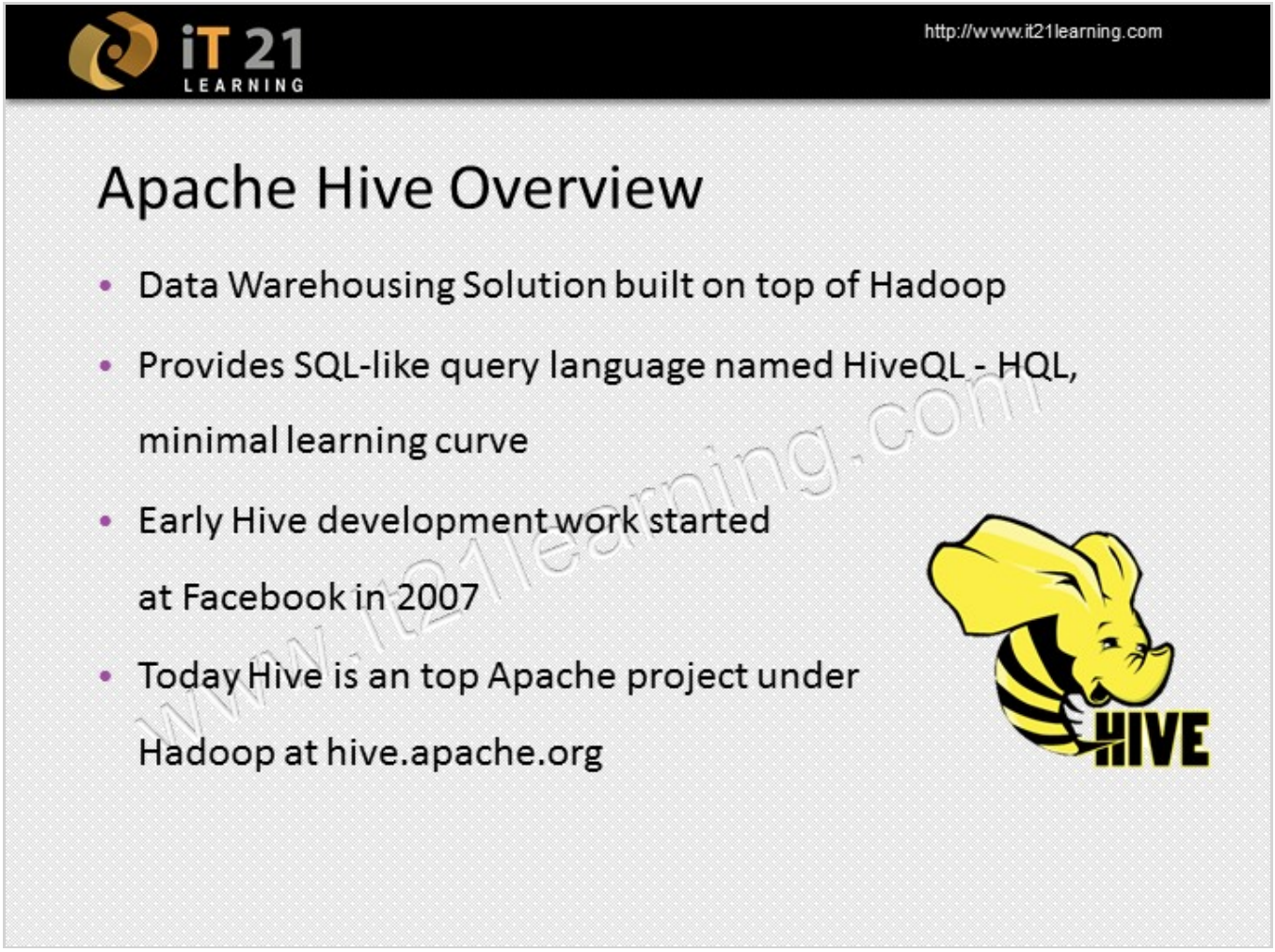

Mastering Apache Hive
Will Du
iT 21
LEARNING
http://www.it21learning.com

1

What do we cover today?

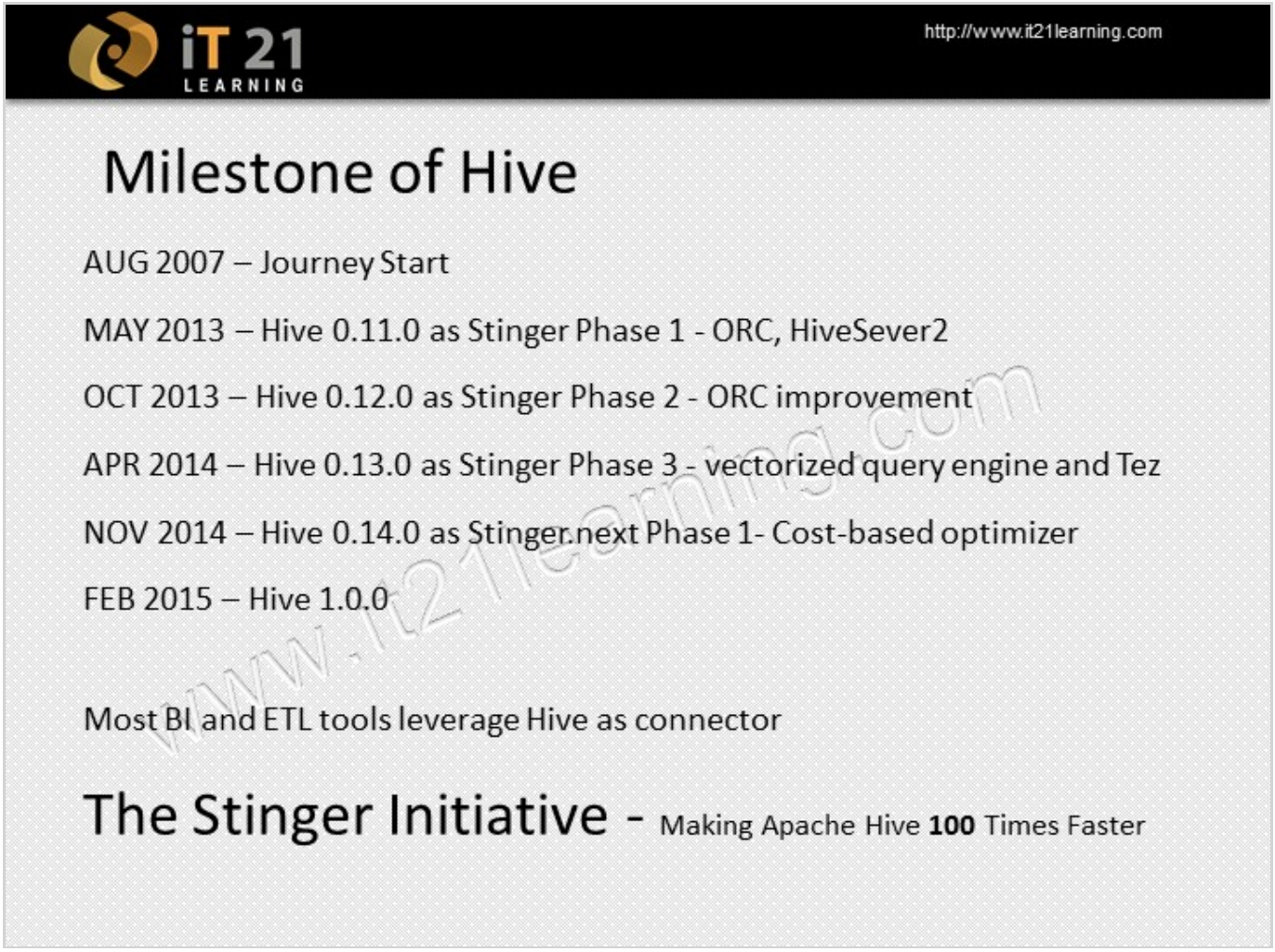
2

Apache Hive Overview



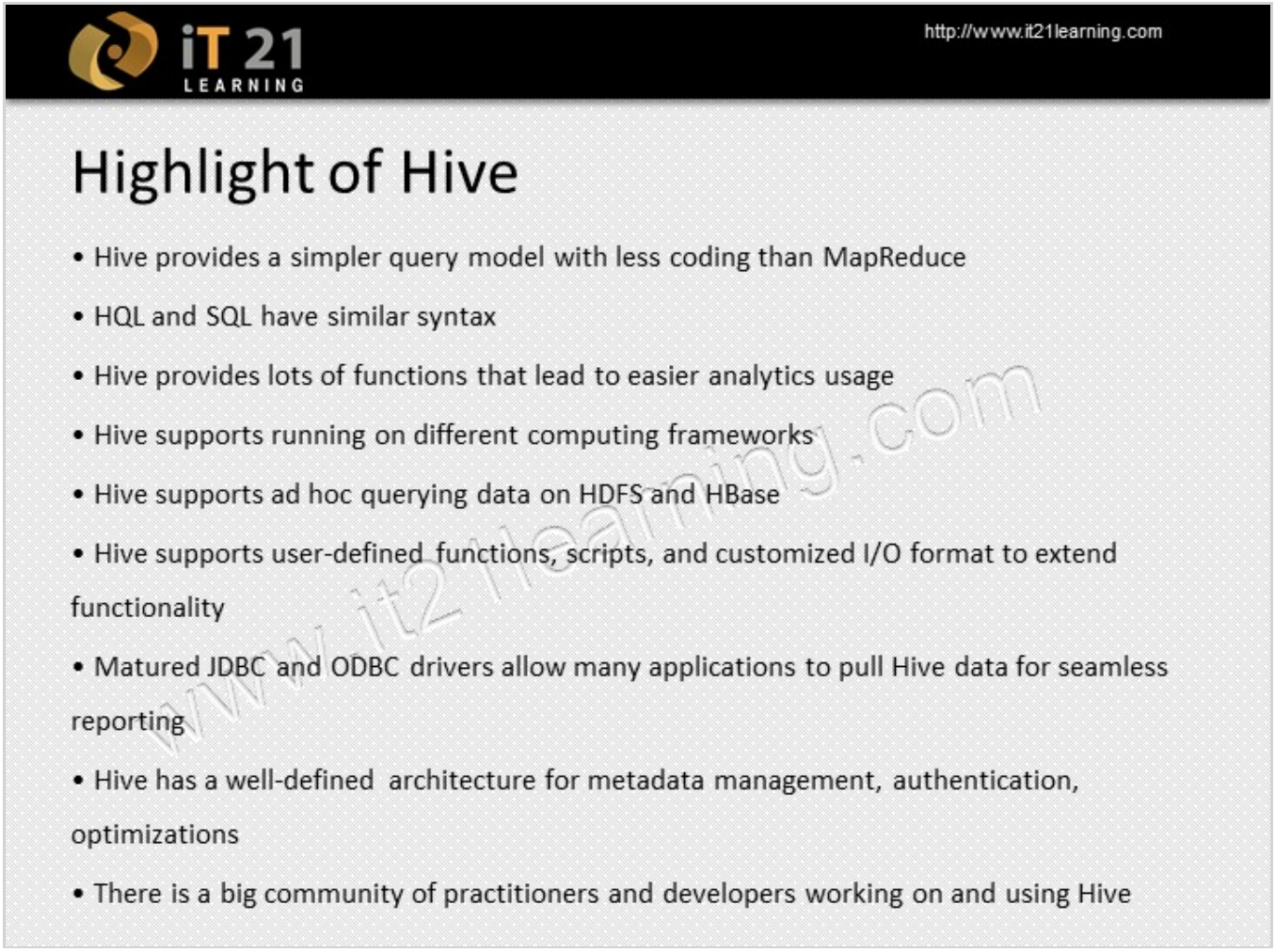
3

Milestone of Hive



4

Highlight of Hive



5

Hive vs. MapReduce – Word Count Again

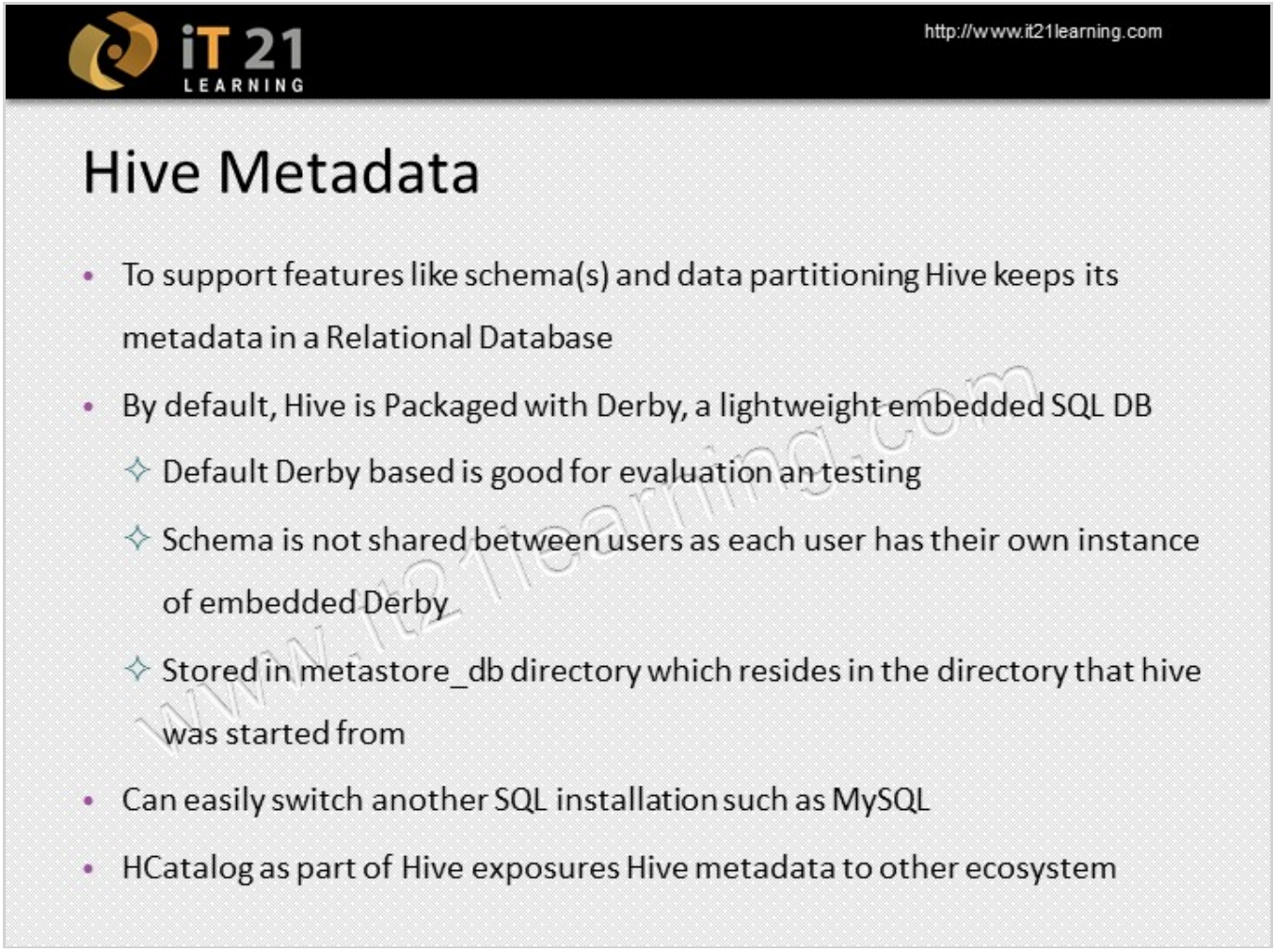
```
package org.apache.hadoop.examples;
import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import 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;
import org.apache.hadoop.util.GenericOptionsParser;

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();
String[] otherArgs = new GenericOptionsParser(conf, args).getRemainingArgs();
if (otherArgs.length != 2) {
  System.err.println("Usage: wordcount <in> <out>");
  System.exit(2);
  }
Job job = new Job(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(otherArgs[0]));
FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));
System.exit(job.waitForCompletion(true) ? 0 : 1);
}
}
```

67 Lin

```
--Define metadata for the source
CREATE EXTERNAL TABLE lines(line STRING);
LOAD DATA INPATH 'book' OVERWRITE INTO TABLE lines;

-- word count
SELECT word, count(*) as world_count
FROM lines
LATERAL VIEW explode(split(text,' ')) t1 as word
GROUP BY word;
```

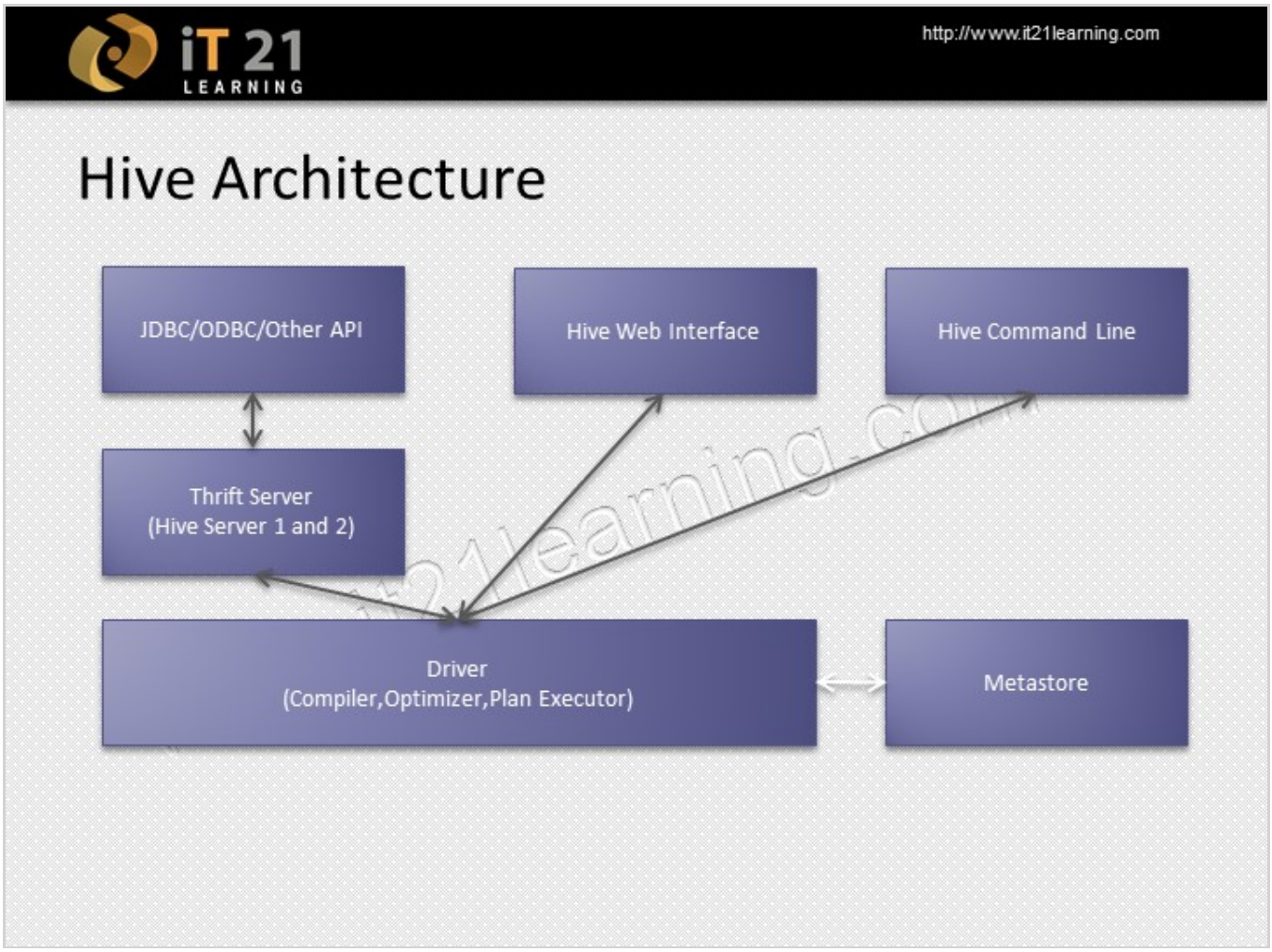
6

Hive Metadata



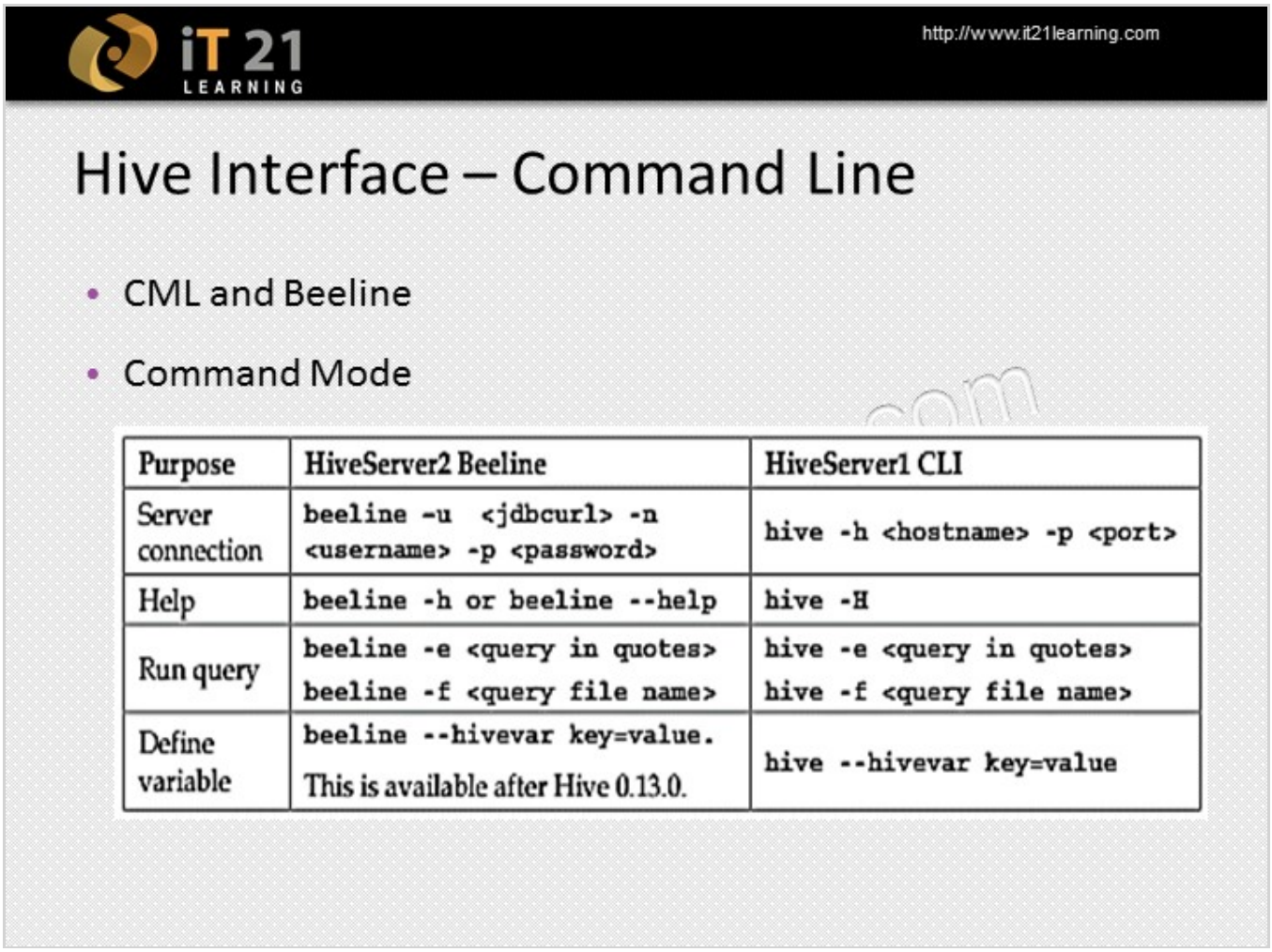
7

Hive Architecture



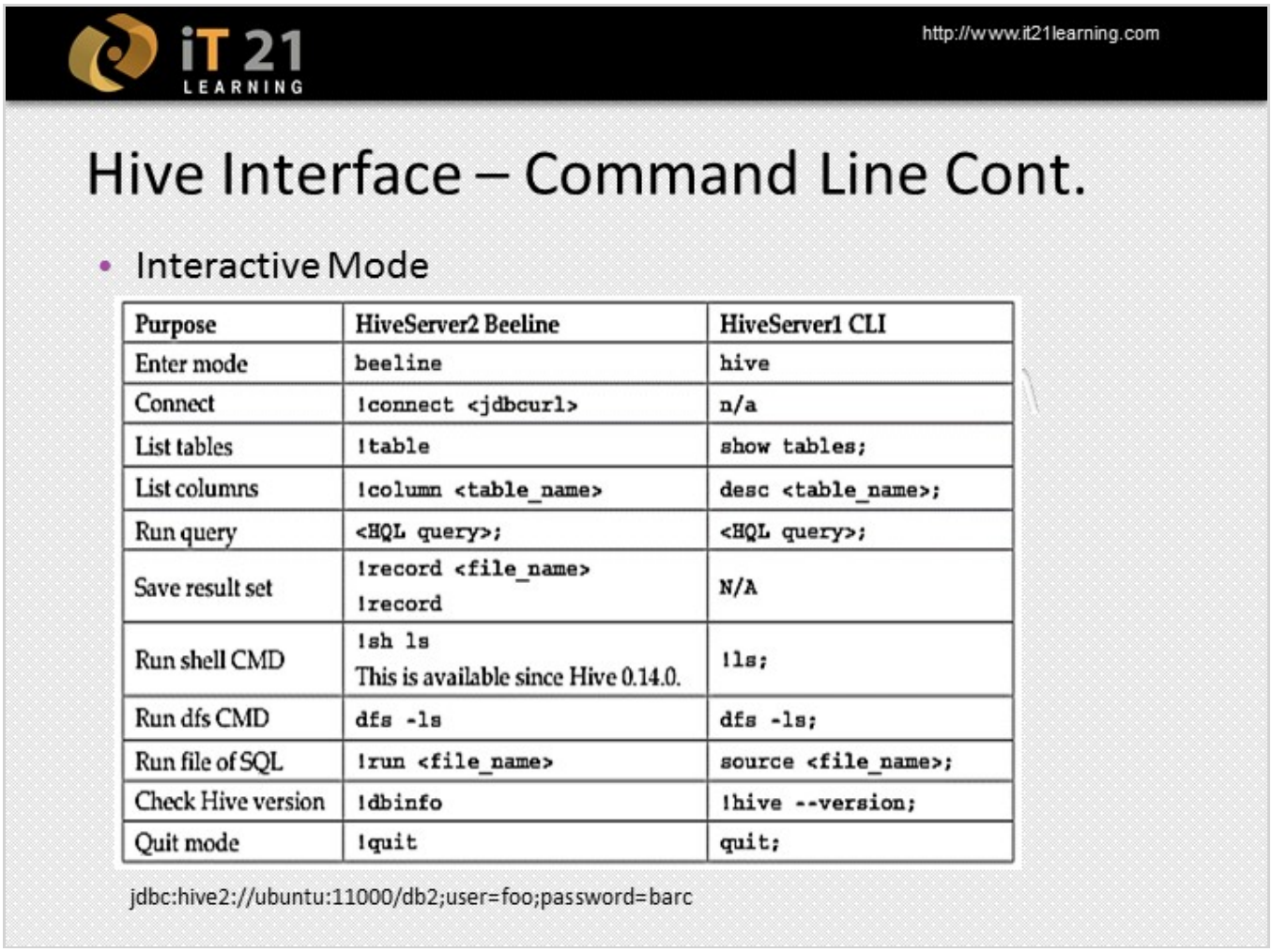
8

Hive Interface – Command Line



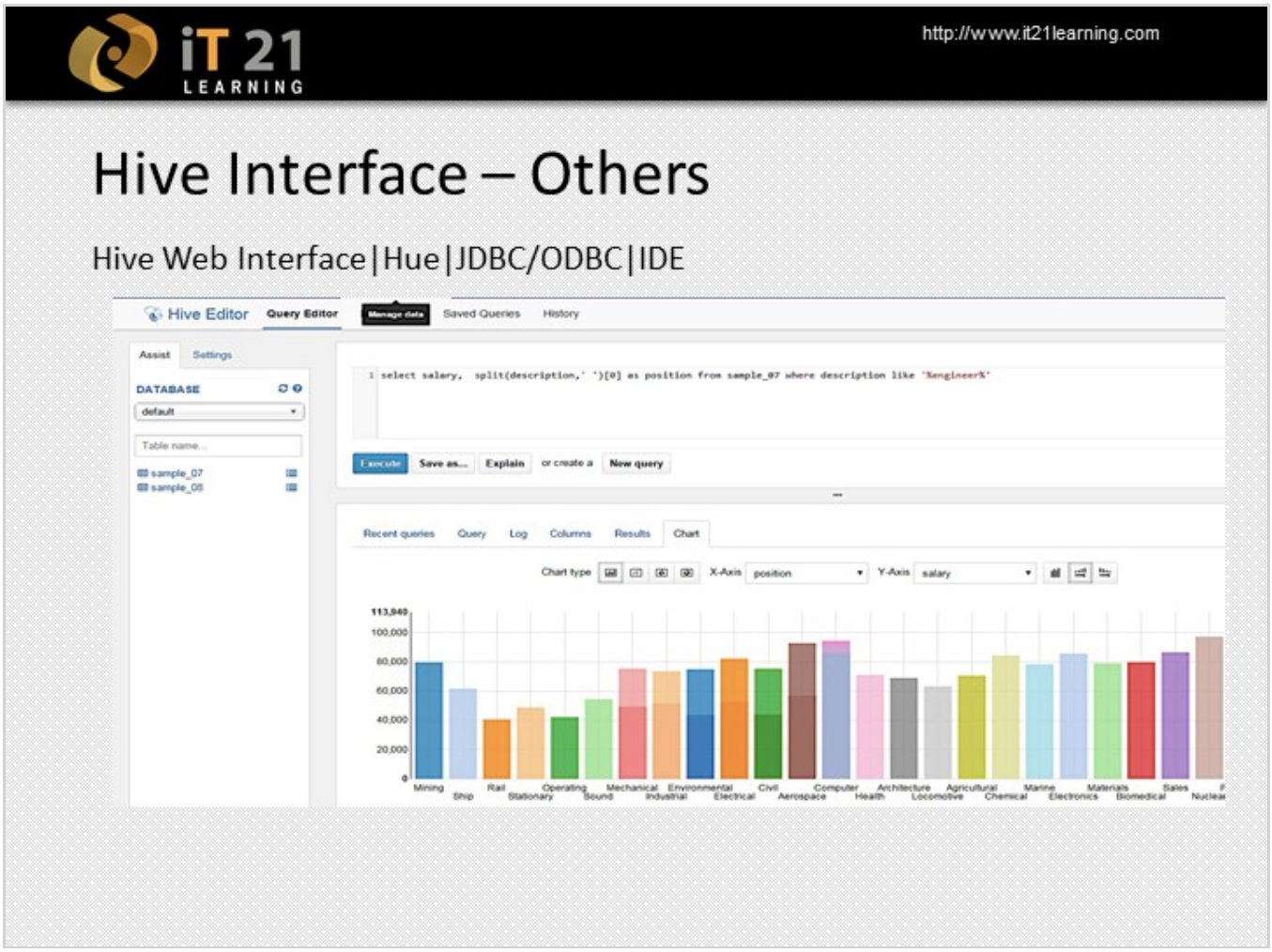
Purpose	HiveServer2 Beeline	HiveServer1 CLI
Server connection	`beeline -u <jdbcurl> -n <username> -p <password>`	`hive -h <hostname> -p <port>`
Help	`beeline -h or beeline --help`	`hive -H`
Run query	`beeline -e <query in quotes>` `beeline -f <query file name>`	`hive -e <query in quotes>` `hive -f <query file name>`
Define variable	`beeline --hivevar key=value.` This is available after Hive 0.13.0.	`hive --hivevar key=value`

You need to use the special hiveconf for variable substitution. e.g. hive> set CURRENT_DATE='2015-06-26'; hive> select * from foo where day >= '${hiveconf:CURRENT_DATE}' similarly, you could pass on command line: % hive -hiveconf CURRENT_DATE='2015-06-26' -f test.hql Note that there are env and system variables as well, so you can reference ${env:USER} for example. To see all the available variables, from the command line, run % hive -e 'set;' or from the hive prompt, run hive> set;



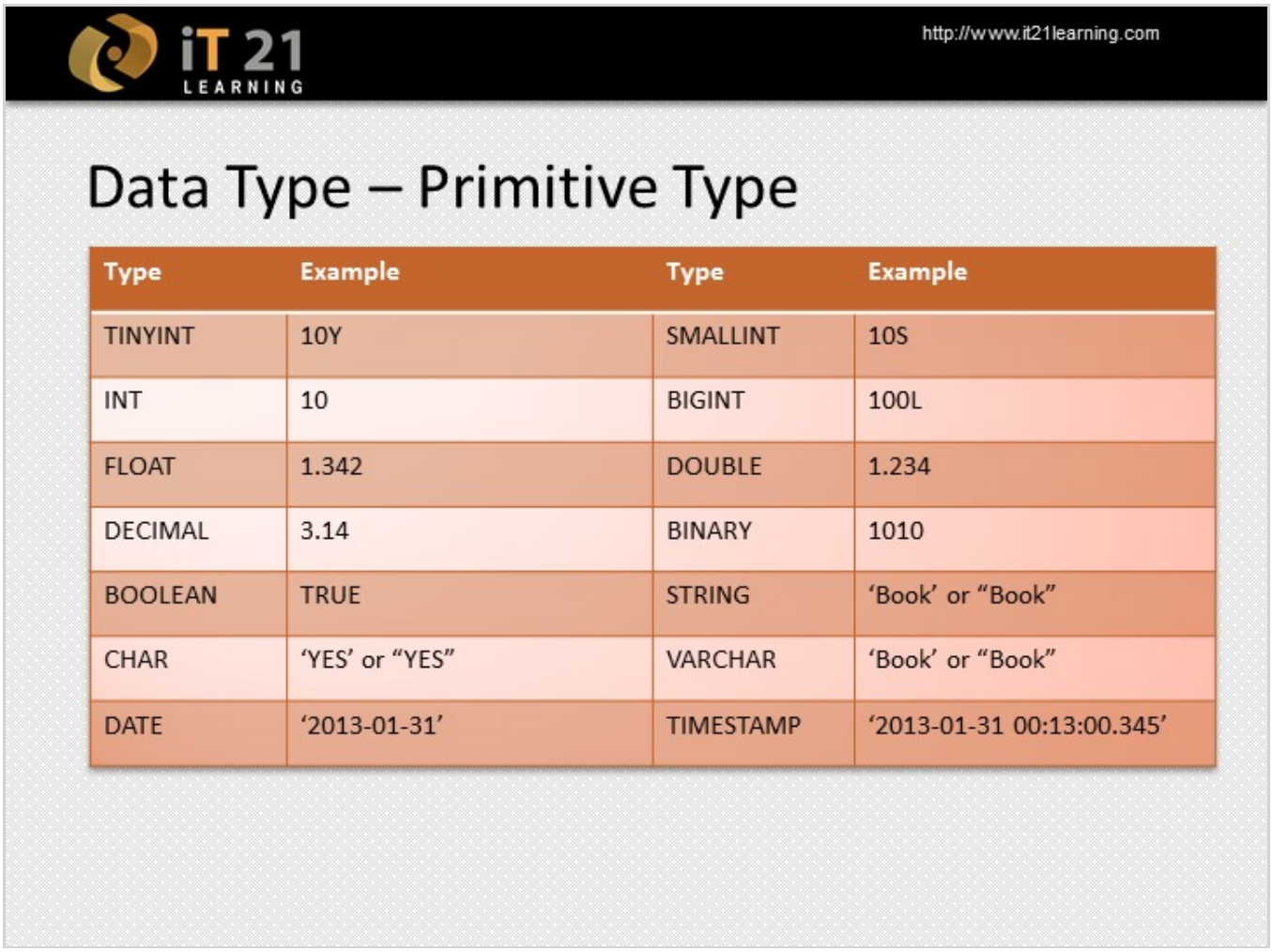
Purpose	HiveServer2 Beeline	HiveServer1 CLI
Enter mode	`beeline`	`hive`
Connect	`!connect <jdbcurl>`	`n/a`
List tables	`!table`	`show tables;`
List columns	`!column <table_name>`	`desc <table_name>;`
Run query	`<HQL query>;`	`<HQL query>;`
Save result set	`!record <file_name>` `!record`	`N/A`
Run shell CMD	`!sh ls` This is available since Hive 0.14.0.	`!ls;`
Run dfs CMD	`dfs -ls`	`dfs -ls;`
Run file of SQL	`!run <file_name>`	`source <file_name>;`
Check Hive version	`!dbinfo`	`!hive --version;`
Quit mode	`!quit`	`quit;`

10

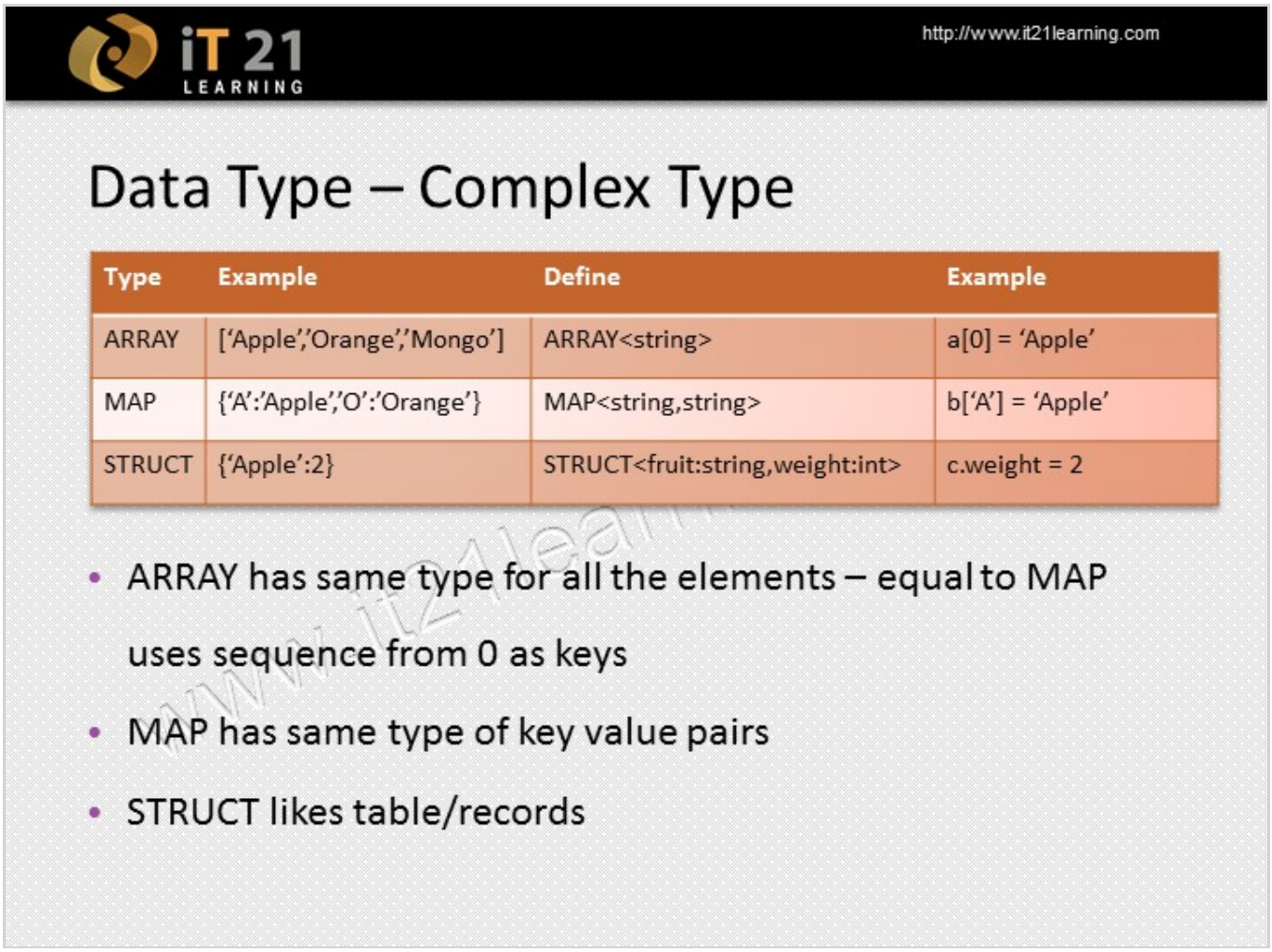

iT 21 LEARNING
http://www.it21learning.com
Hive Interface – Others
Hive Web Interface|Hue|JDBC/ODBC|IDE
Hive Editor
Query Editor
Manage data
Saved Queries
History
Assist
Settings
DATABASE
default
Table name...
sample_07
sample_08
select salary, split(description,' ')[0] as position from sample_07 where description like '%engineer%'
Execute
Save as...
Explain
or create a
New query
Recent queries
Query
Log
Columns
Results
Chart
Chart type
X-Axis
position
Y-Axis
salary
113,940
100,000
80,000
60,000
40,000
20,000
0
Mining
Ship
Rail
Stationary
Operating
Sound
Mechanical
Industrial
Environmental
Electrical
Civil
Aerospace
Computer
Health
Architecture
Locomotive
Agricultural
Chemical
Marine
Electronics
Materials
Biomedical
Sales

11

Data Type – Primitive Type

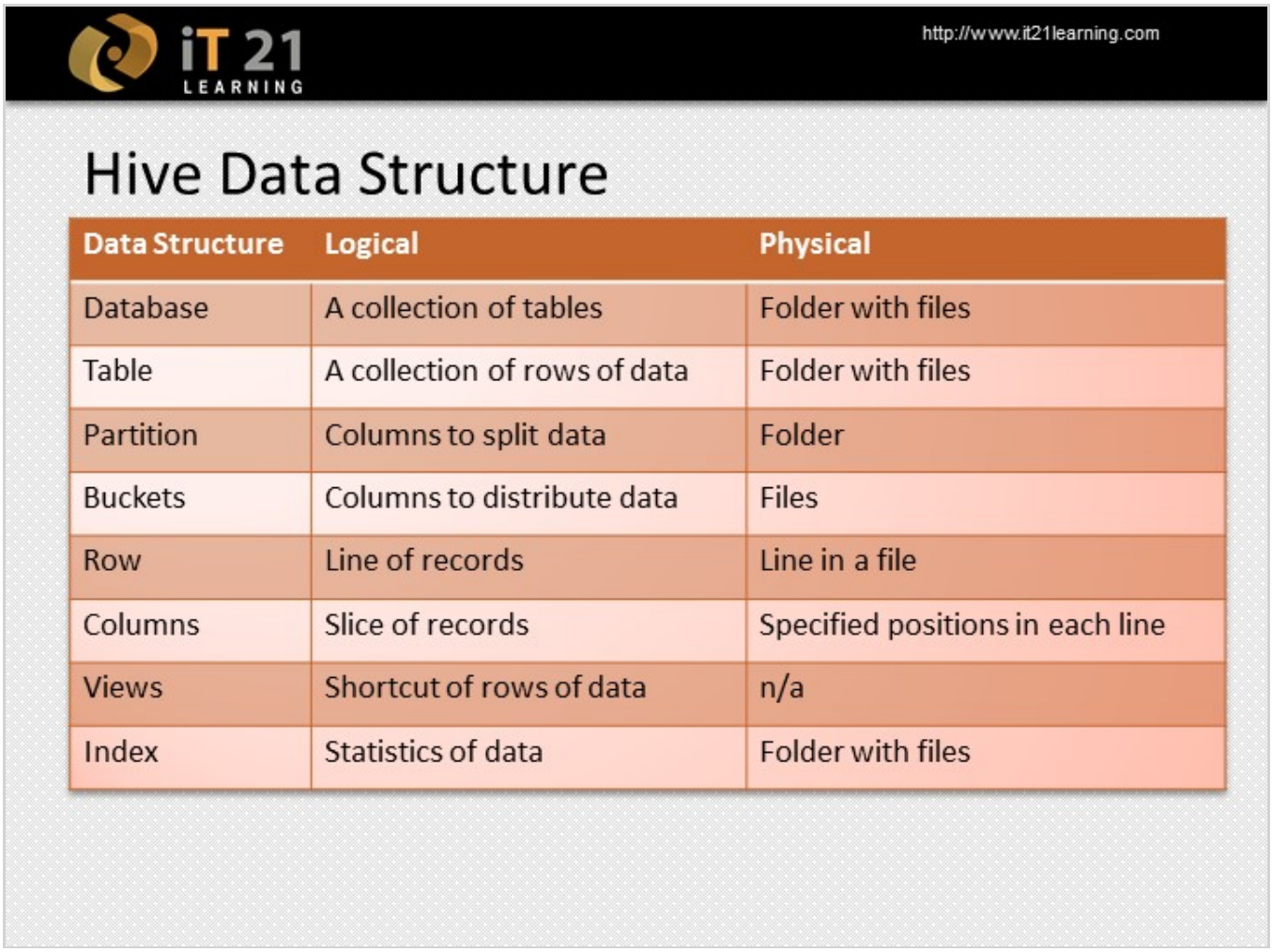


12


iT 21 LEARNING
http://www.it21learning.com
Data Type – Complex Type
Type | Example | Define | Example
ARRAY | ['Apple','Orange','Mongo'] | ARRAY<string> | a[0] = 'Apple'
MAP | {'A':'Apple','O':'Orange'} | MAP<string,string> | b['A'] = 'Apple'
STRUCT | {'Apple':2} | STRUCT<fruit:string,weight:int> | c.weight = 2
• ARRAY has same type for all the elements – equal to MAP uses sequence from 0 as keys
• MAP has same type of key value pairs
• STRUCT likes table/records

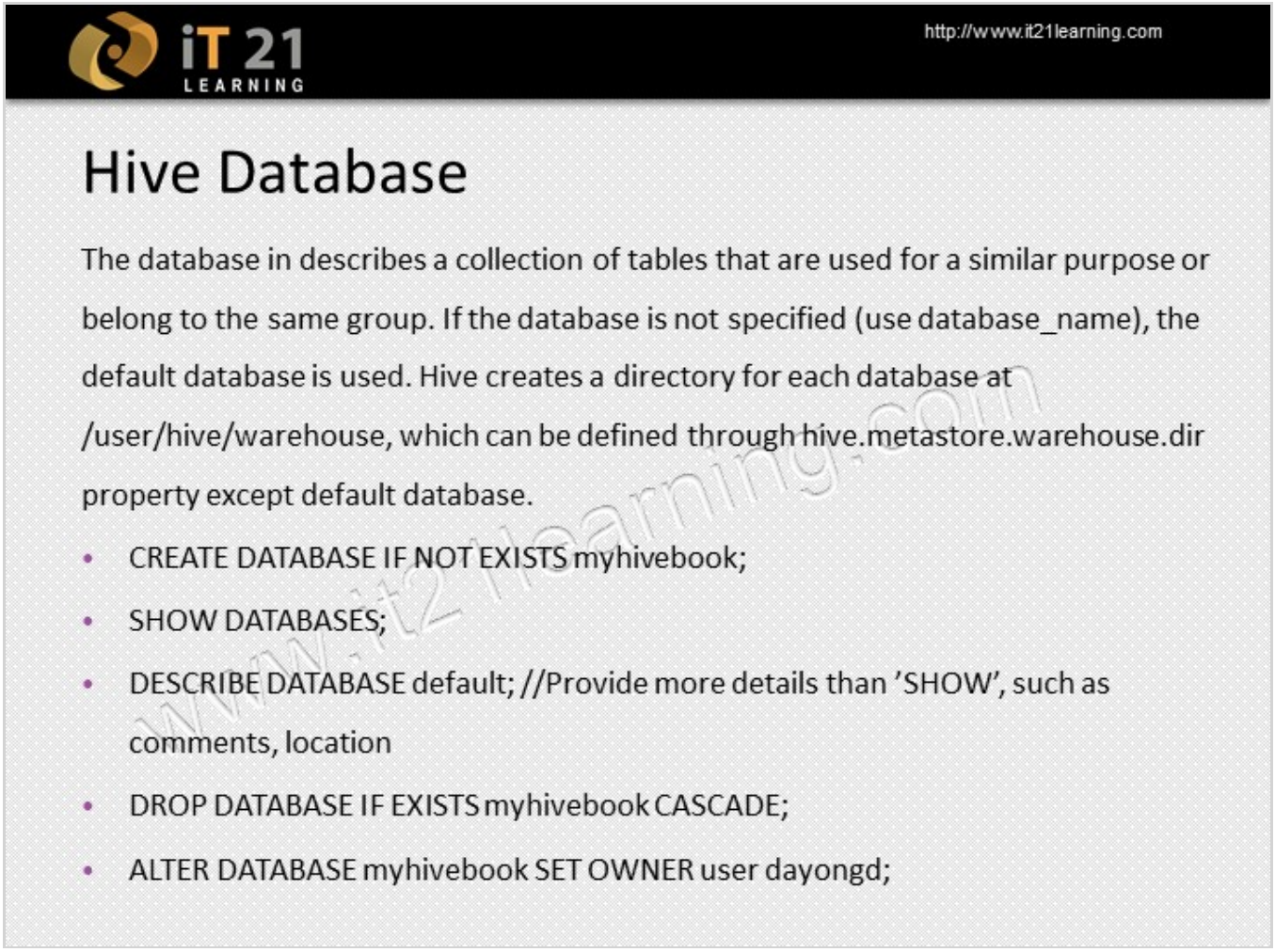
13

Hive Data Structure



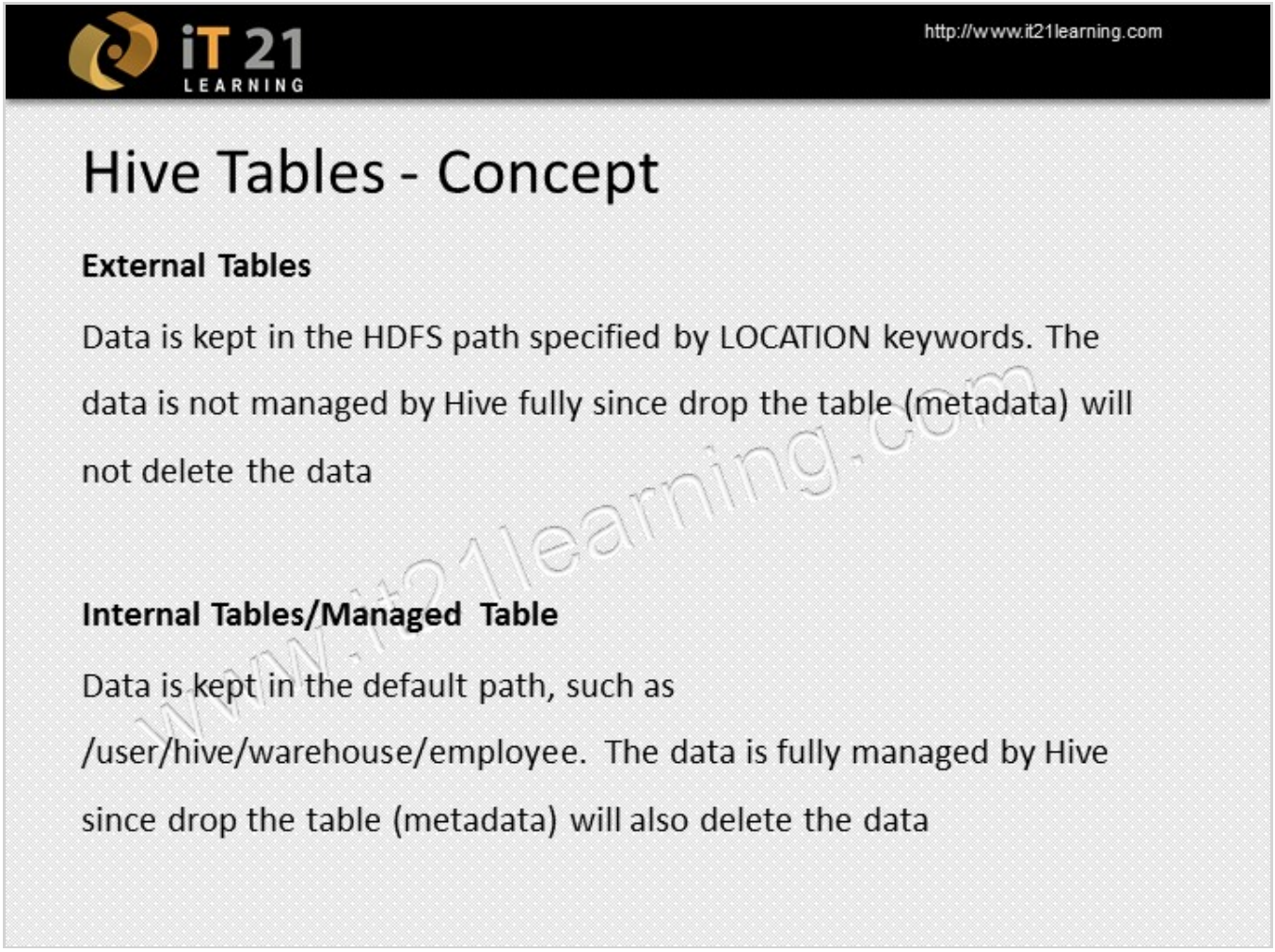
14

Hive Database

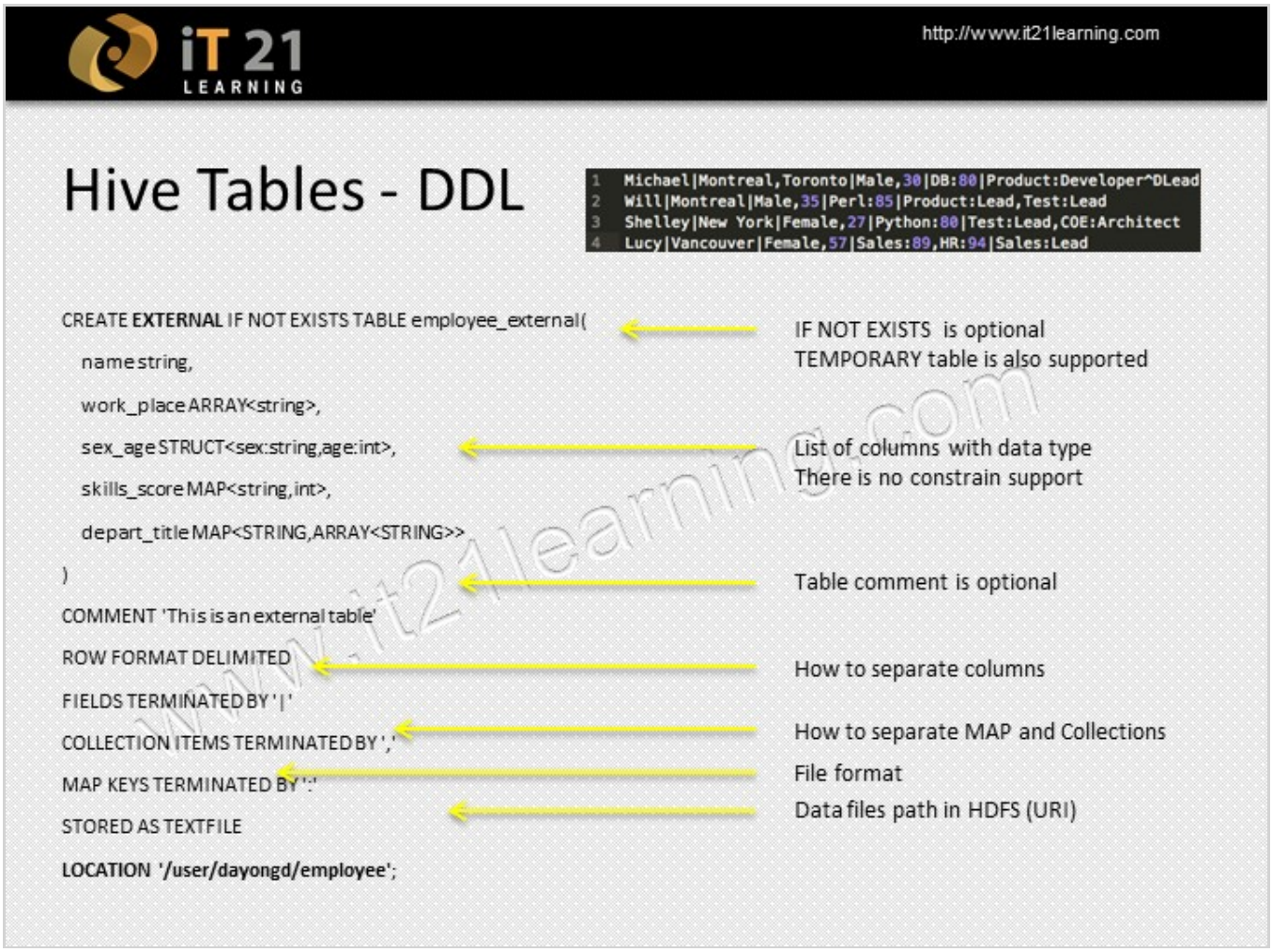


15

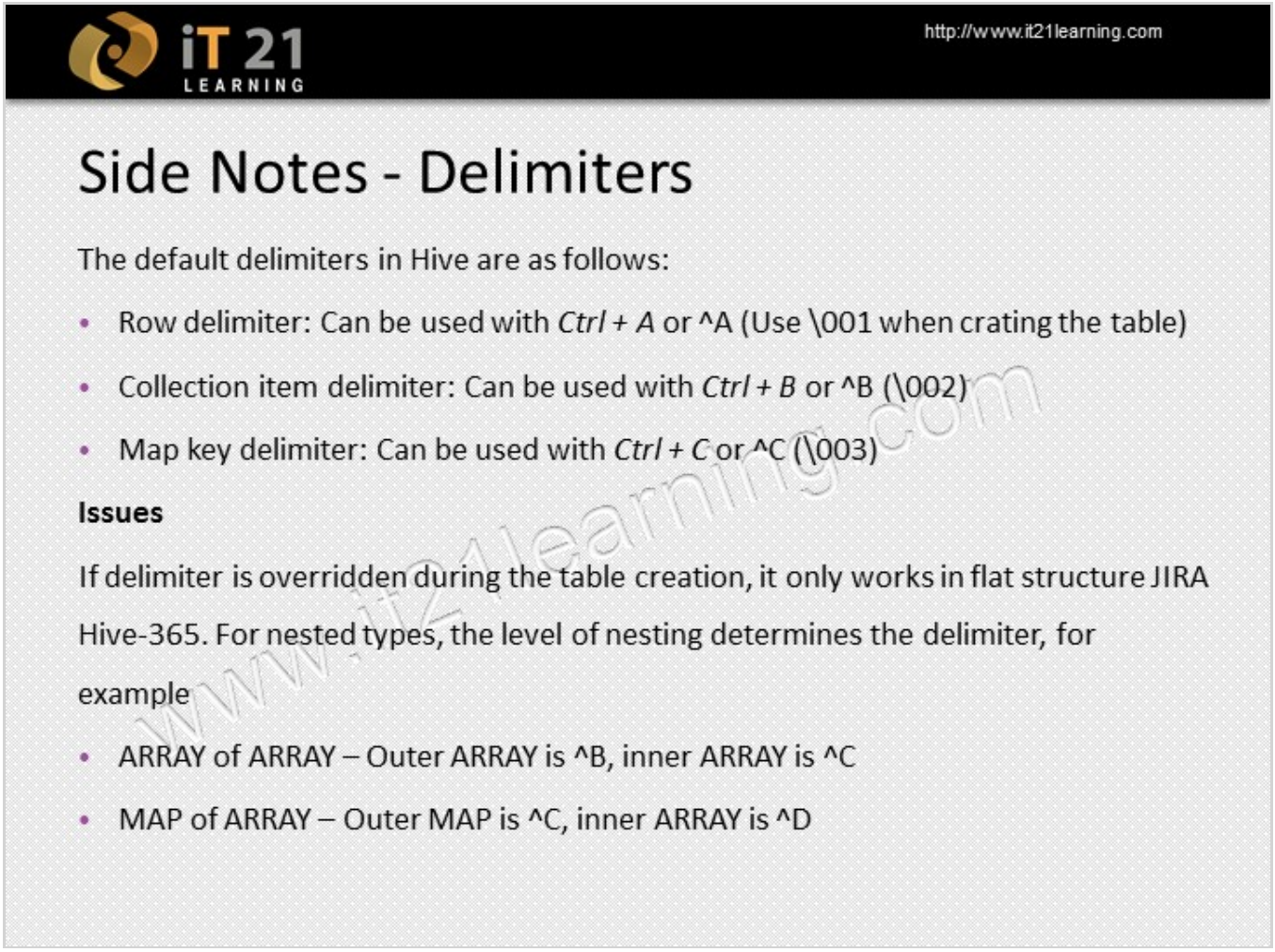
Hive Tables - Concept



16



No sub-directory is allowed in the URI

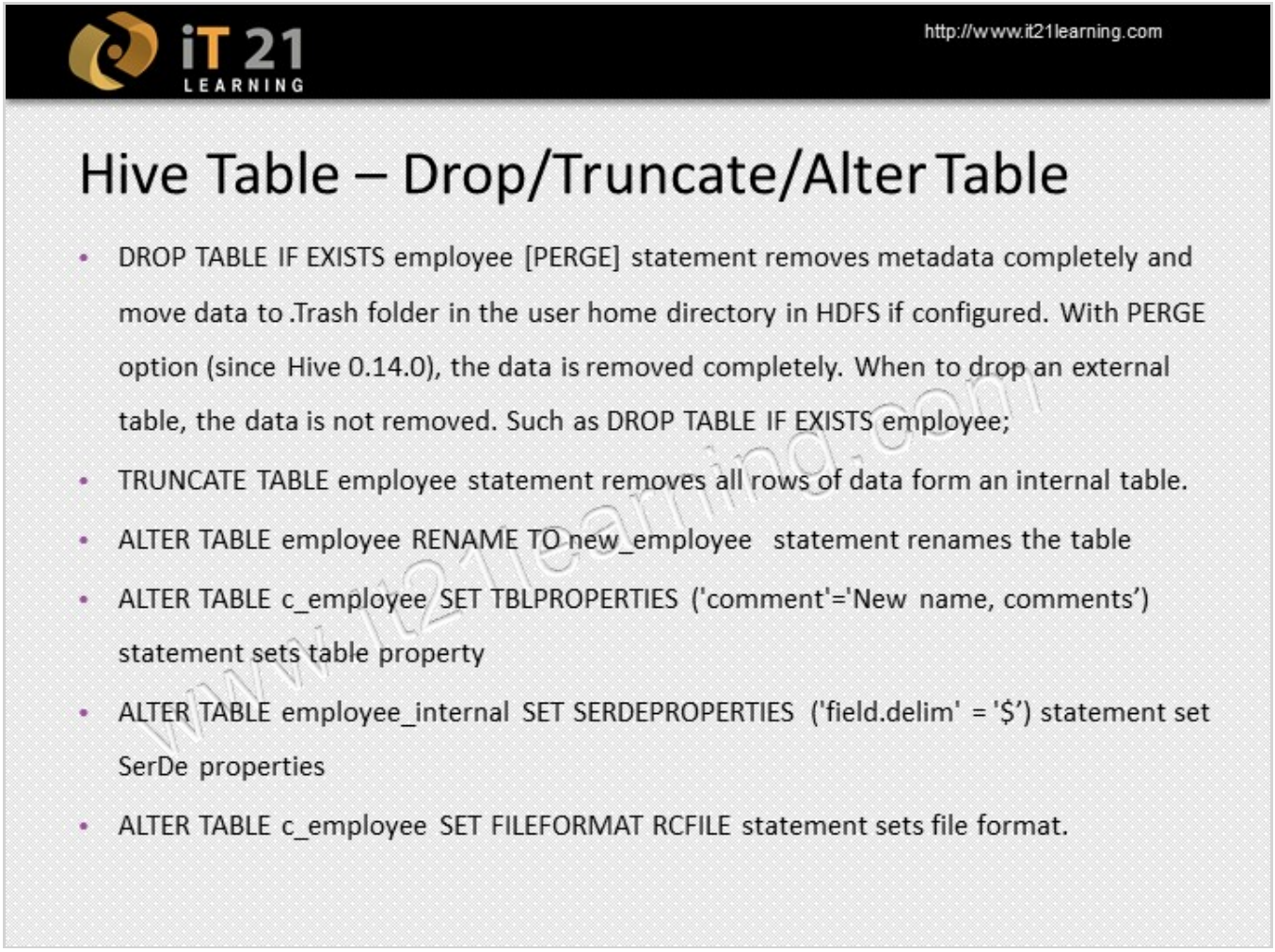

iT 21 LEARNING
http://www.it21learning.com
Side Notes - Delimiters
The default delimiters in Hive are as follows:
• Row delimiter: Can be used with Ctrl + A or ^A (Use \001 when crating the table)
• Collection item delimiter: Can be used with Ctrl + B or ^B (\002)
• Map key delimiter: Can be used with Ctrl + C or ^C (\003)
Issues
If delimiter is overridden during the table creation, it only works in flat structure JIRA Hive-365. For nested types, the level of nesting determines the delimiter, for example
• ARRAY of ARRAY – Outer ARRAY is ^B, inner ARRAY is ^C
• MAP of ARRAY – Outer MAP is ^C, inner ARRAY is ^D

18

Hive Table – DDL – Create Table

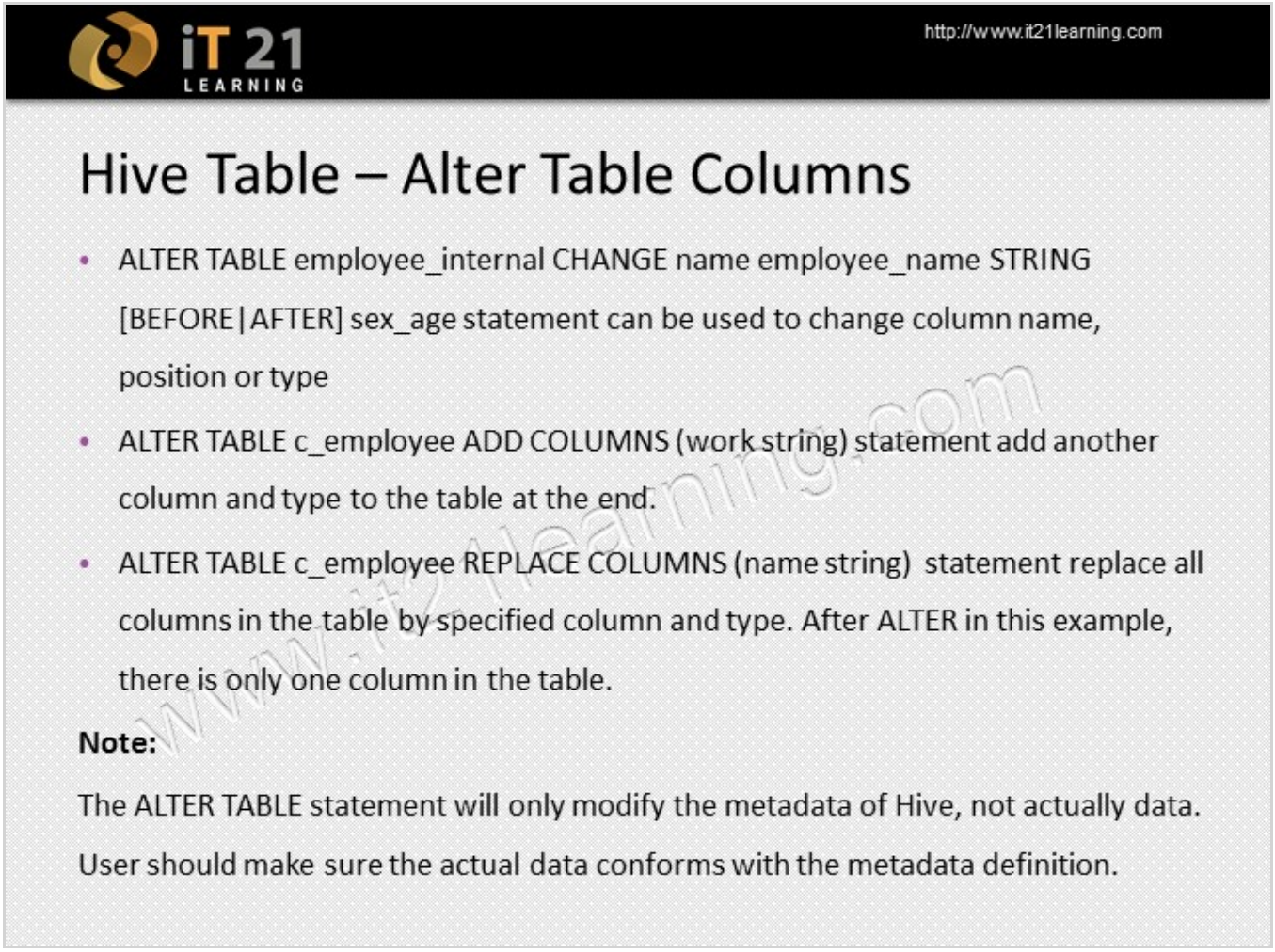
19

Hive Table – Drop/Truncate/Alter Table

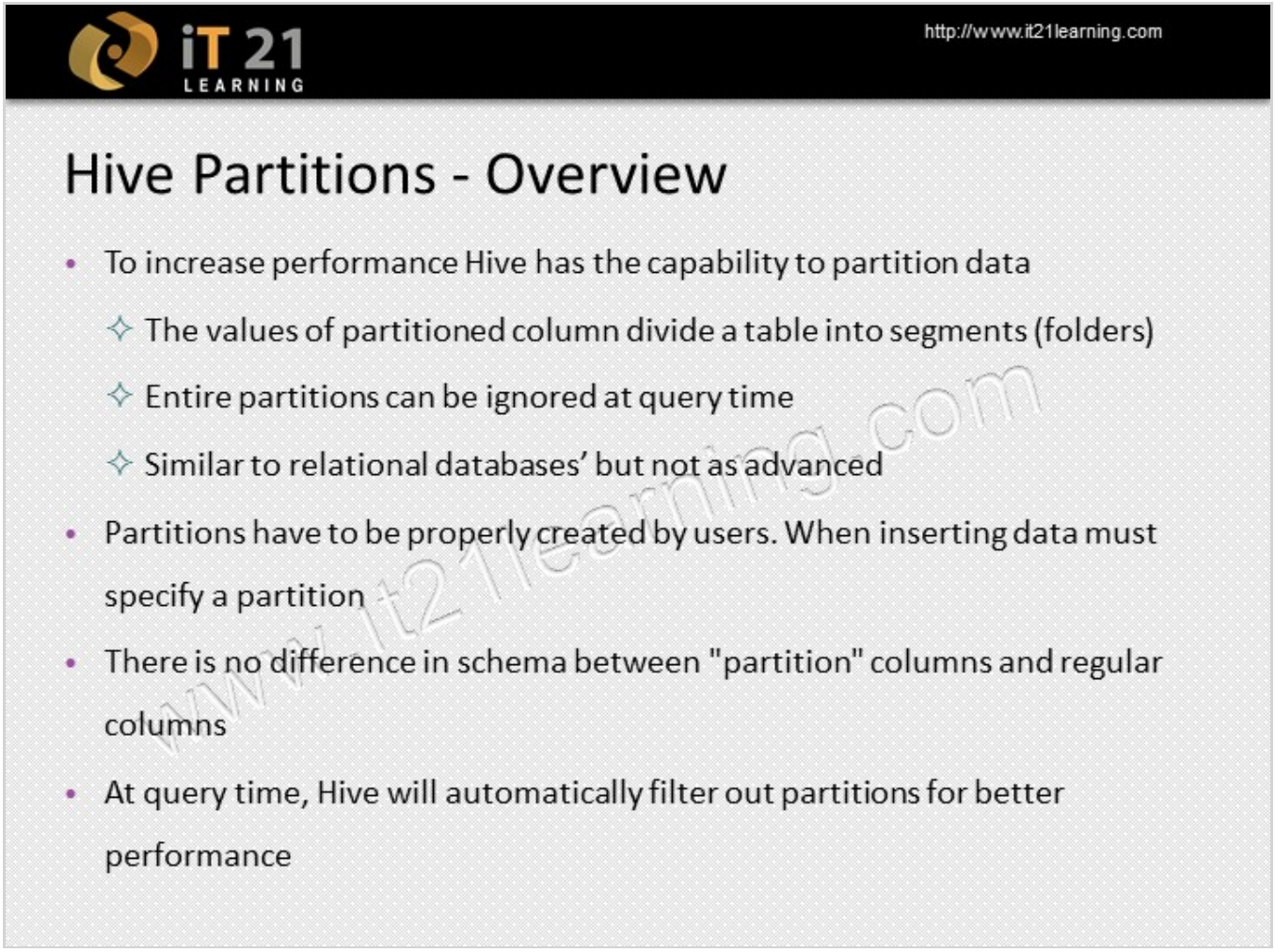


DROP TABLE IF EXISTS empty_ctas_employee;

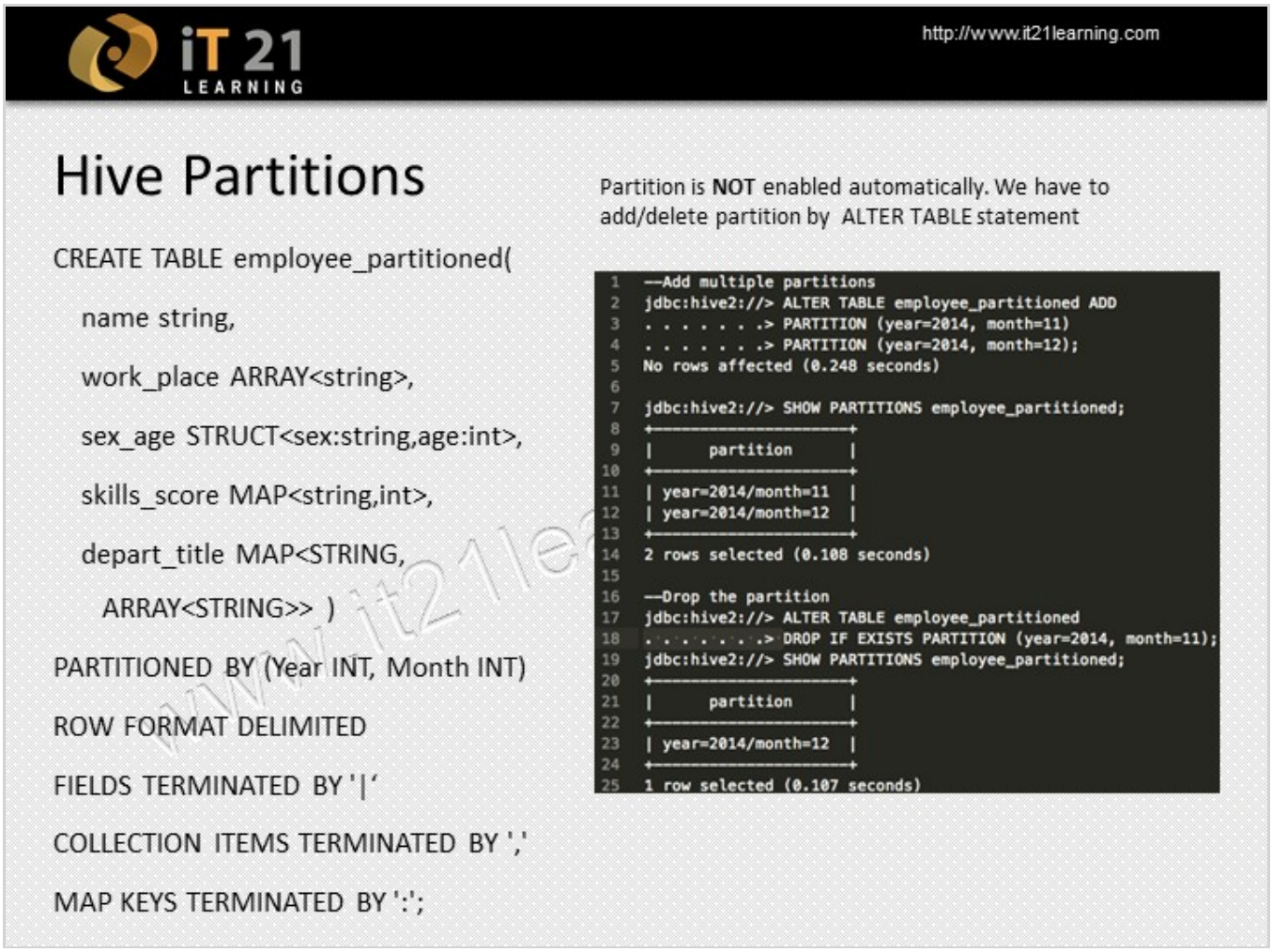
Hive Table – Alter Table Columns



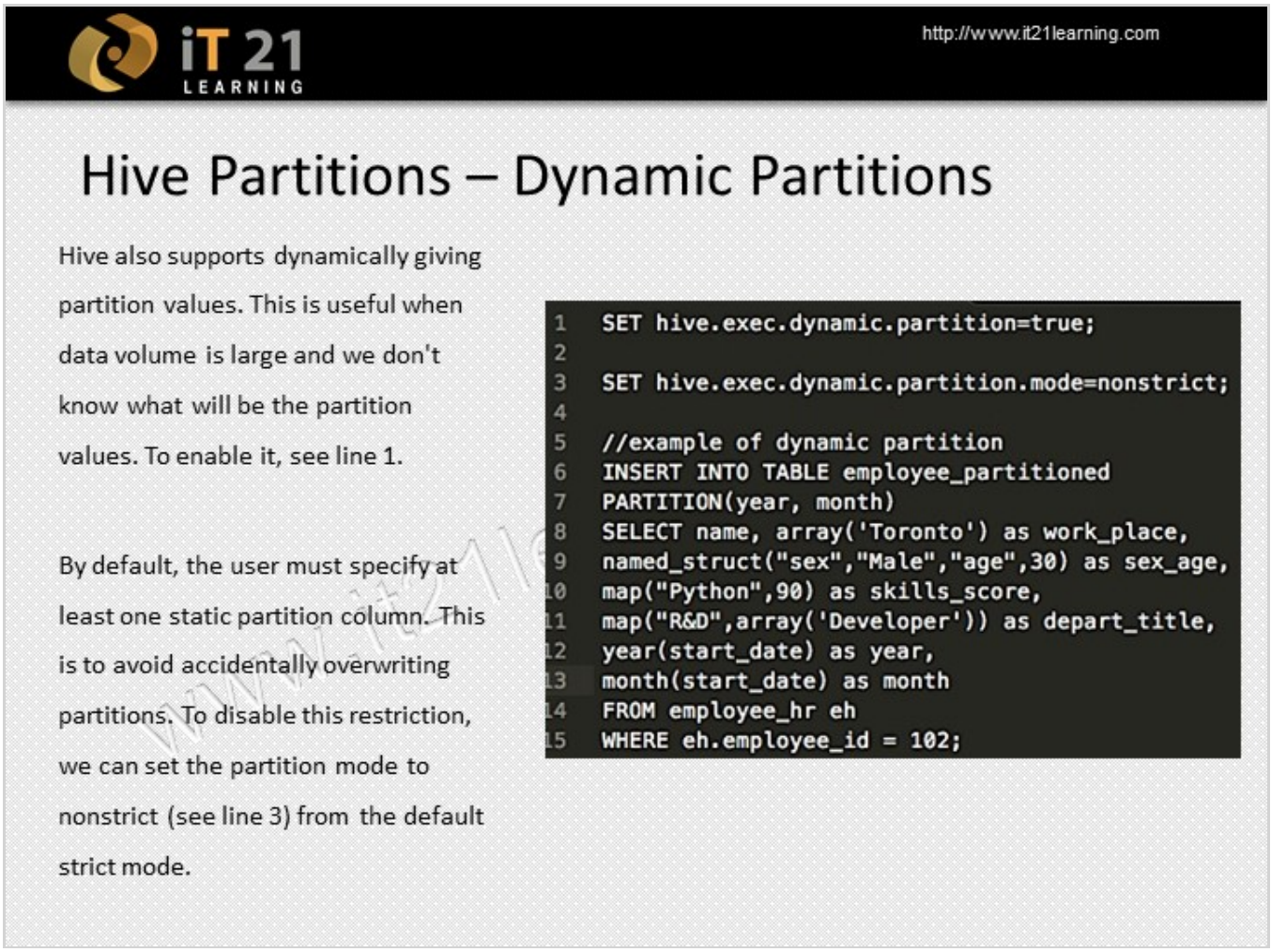
21


iT 21 LEARNING
http://www.it21learning.com
Hive Partitions - Overview
To increase performance Hive has the capability to partition data
The values of partitioned column divide a table into segments (folders)
Entire partitions can be ignored at query time
Similar to relational databases' but not as advanced
Partitions have to be properly created by users. When inserting data must specify a partition
There is no difference in schema between "partition" columns and regular columns
At query time, Hive will automatically filter out partitions for better performance
www.it21learning.com

22

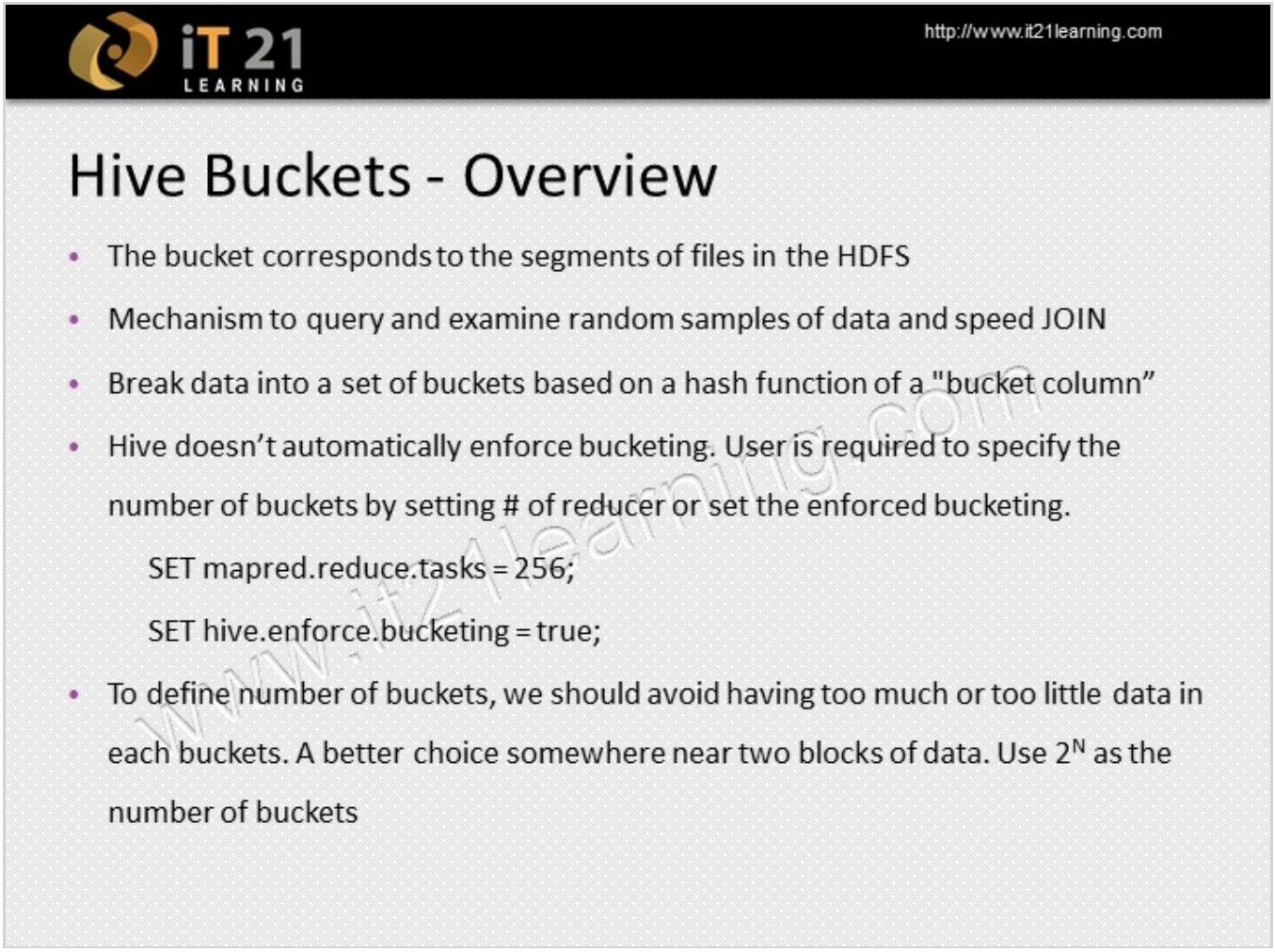

iT 21 LEARNING
http://www.it21learning.com
Hive Partitions
CREATE TABLE employee_partitioned(
name string,
work_place ARRAY<string>,
sex_age STRUCT<sex:string,age:int>,
skills_score MAP<string,int>,
depart_title MAP<STRING,
ARRAY<STRING>>)
PARTITIONED BY (Year INT, Month INT)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '|'
COLLECTION ITEMS TERMINATED BY ','
MAP KEYS TERMINATED BY ':';
Partition is NOT enabled automatically. We have to add/delete partition by ALTER TABLE statement
--Add multiple partitions
jdbc:hive2://> ALTER TABLE employee_partitioned ADD
. . . . . .> PARTITION (year=2014, month=11)
. . . . . .> PARTITION (year=2014, month=12);
No rows affected (0.248 seconds)
jdbc:hive2://> SHOW PARTITIONS employee_partitioned;
| partition |
| year=2014/month=11 |
| year=2014/month=12 |
2 rows selected (0.108 seconds)
--Drop the partition
jdbc:hive2://> ALTER TABLE employee_partitioned
. . . . . .> DROP IF EXISTS PARTITION (year=2014, month=11);
jdbc:hive2://> SHOW PARTITIONS employee_partitioned;
| partition |
| year=2014/month=12 |
1 row selected (0.107 seconds)

23

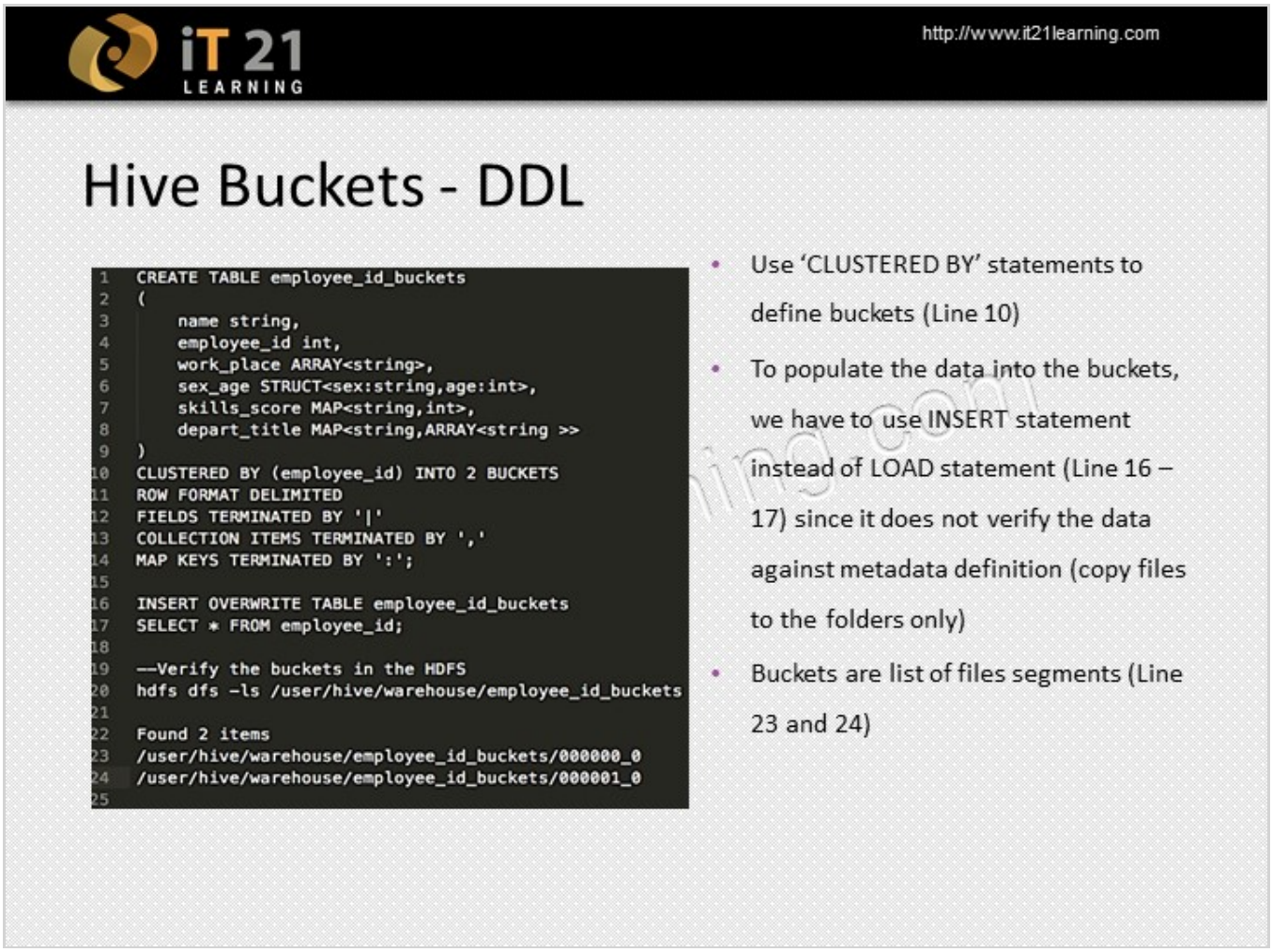

iT 21 LEARNING
http://www.it21learning.com
Hive Partitions – Dynamic Partitions
Hive also supports dynamically giving partition values. This is useful when data volume is large and we don't know what will be the partition values. To enable it, see line 1.
By default, the user must specify at least one static partition column. This is to avoid accidentally overwriting partitions. To disable this restriction, we can set the partition mode to nonstrict (see line 3) from the default strict mode.
1 SET hive.exec.dynamic.partition=true;
2
3 SET hive.exec.dynamic.partition.mode=nonstrict;
4
5 //example of dynamic partition
6 INSERT INTO TABLE employee_partitioned
7 PARTITION(year, month)
8 SELECT name, array('Toronto') as work_place,
9 named_struct("sex","Male","age",30) as sex_age,
10 map("Python",90) as skills_score,
11 map("R&D",array('Developer')) as depart_title,
12 year(start_date) as year,
13 month(start_date) as month
14 FROM employee_hr eh
15 WHERE eh.employee_id = 102;

24

Hive Buckets - Overview

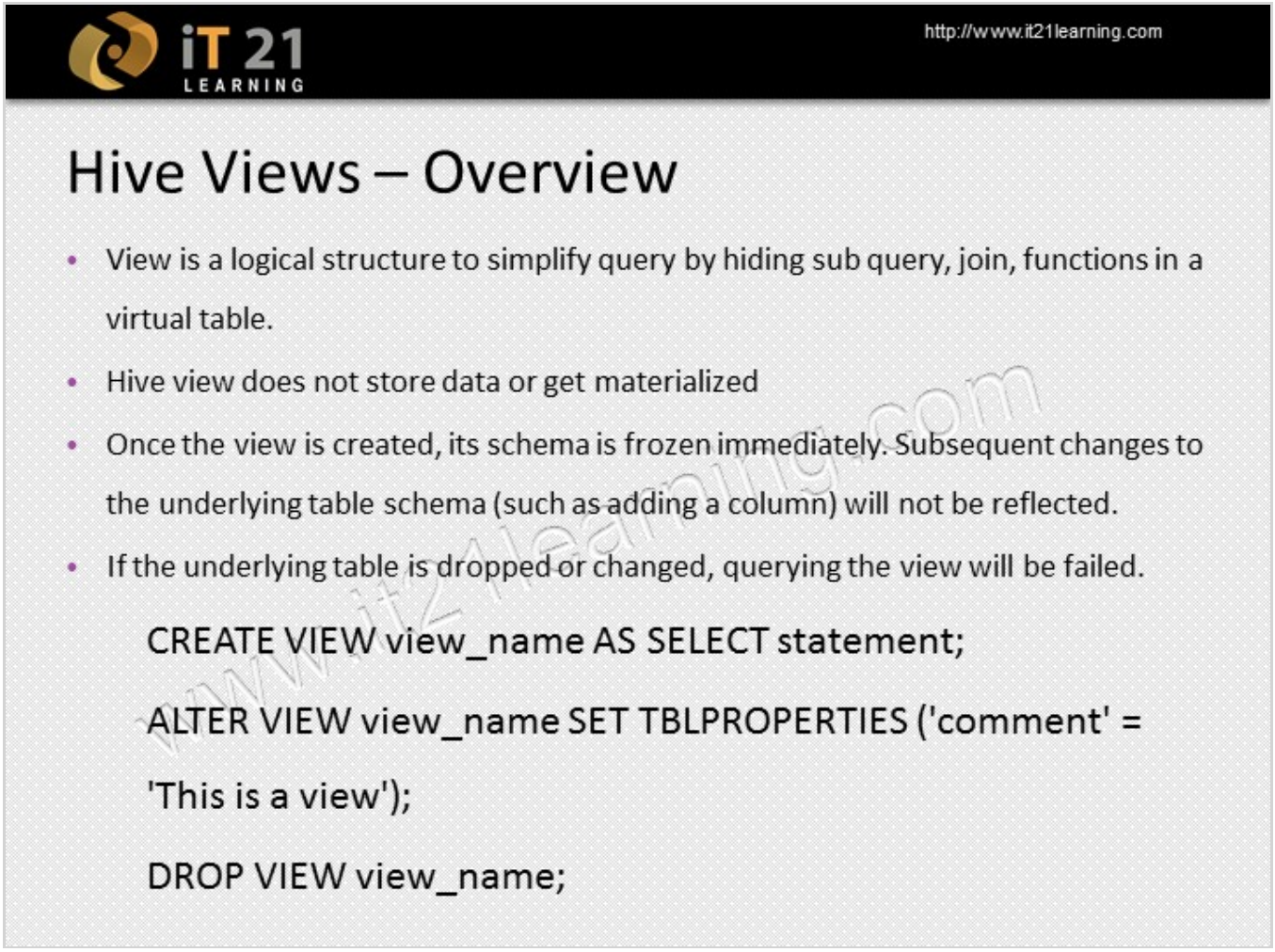


Data in the same buckets can join fast since they are likely have the same group of keys


iT 21 LEARNING
http://www.it21learning.com
Hive Buckets - DDL
1 CREATE TABLE employee_id_buckets
2 (
3 name string,
4 employee_id int,
5 work_place ARRAY<string>,
6 sex_age STRUCT<sex:string,age:int>,
7 skills_score MAP<string,int>,
8 depart_title MAP<string,ARRAY<string >>
9)
10 CLUSTERED BY (employee_id) INTO 2 BUCKETS
11 ROW FORMAT DELIMITED
12 FIELDS TERMINATED BY '|'
13 COLLECTION ITEMS TERMINATED BY ','
14 MAP KEYS TERMINATED BY ':';
15
16 INSERT OVERWRITE TABLE employee_id_buckets
17 SELECT * FROM employee_id;
18
19 --Verify the buckets in the HDFS
20 hdfs dfs -ls /user/hive/warehouse/employee_id_buckets
21
22 Found 2 items
23 /user/hive/warehouse/employee_id_buckets/000000_0
24 /user/hive/warehouse/employee_id_buckets/000001_0
25
• Use 'CLUSTERED BY' statements to define buckets (Line 10)
• To populate the data into the buckets, we have to use INSERT statement instead of LOAD statement (Line 16 – 17) since it does not verify the data against metadata definition (copy files to the folders only)
• Buckets are list of files segments (Line 23 and 24)

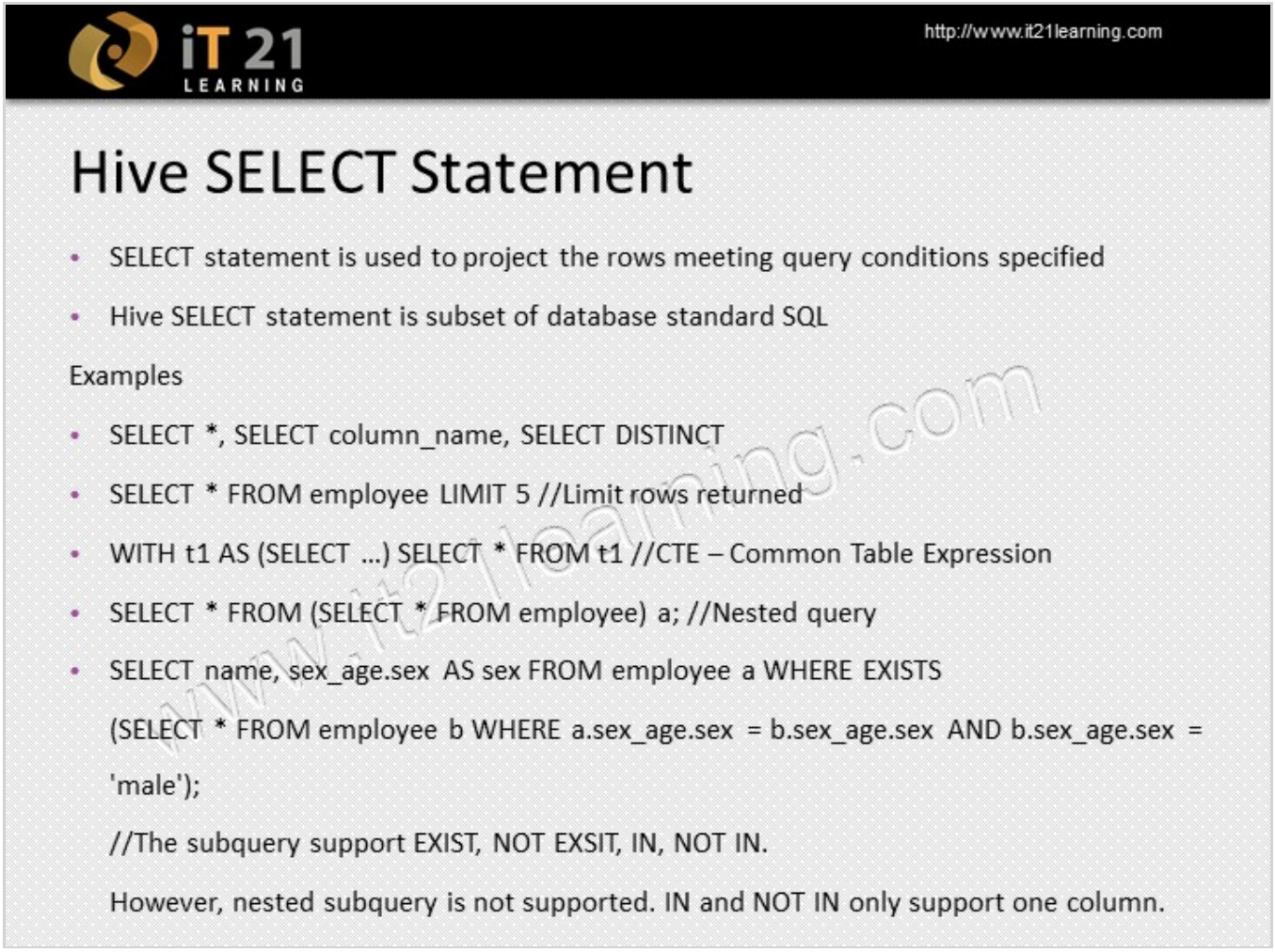
26

Hive Views – Overview



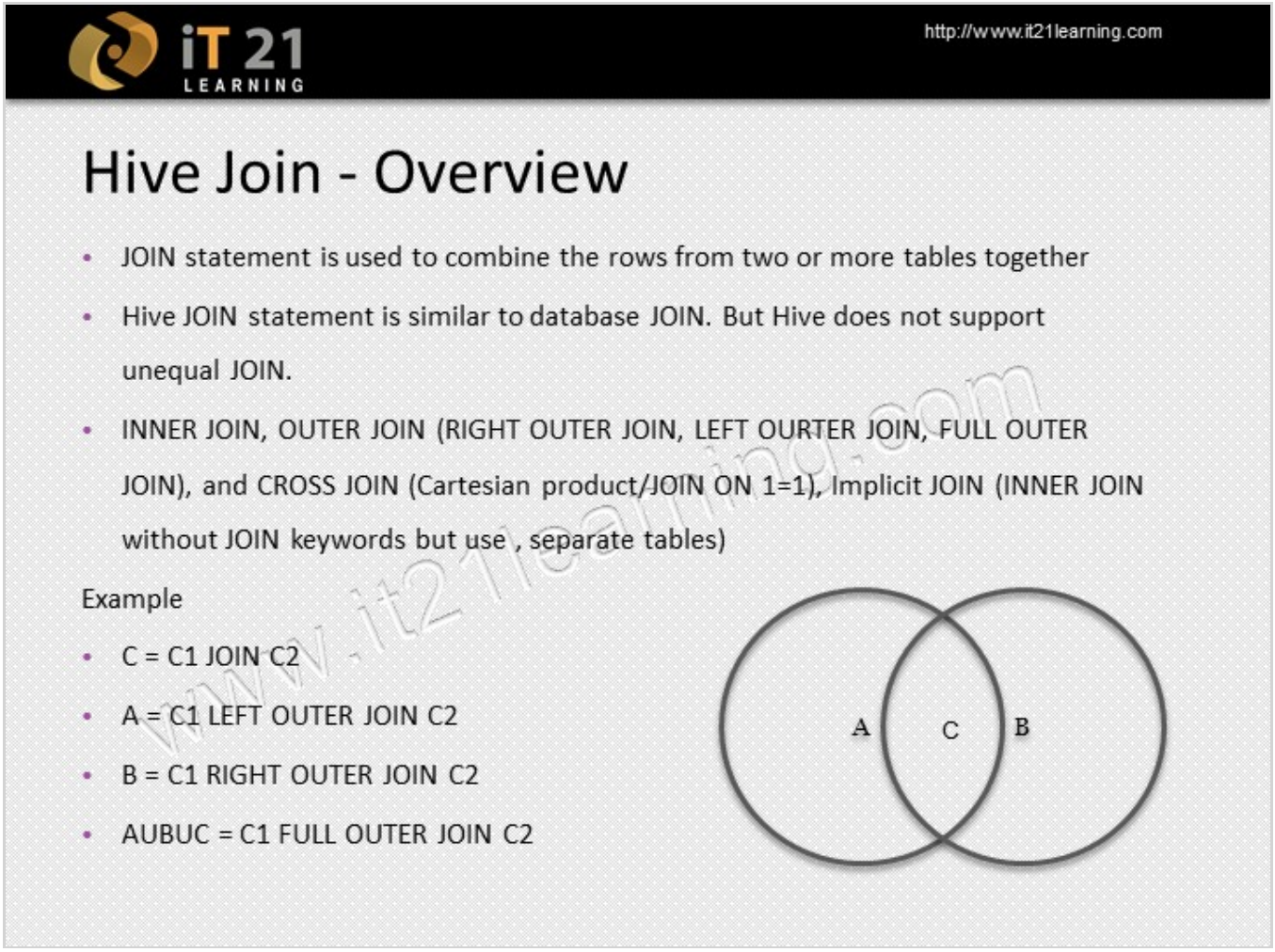
View is not likely optimized by optimizer

Hive SELECT Statement



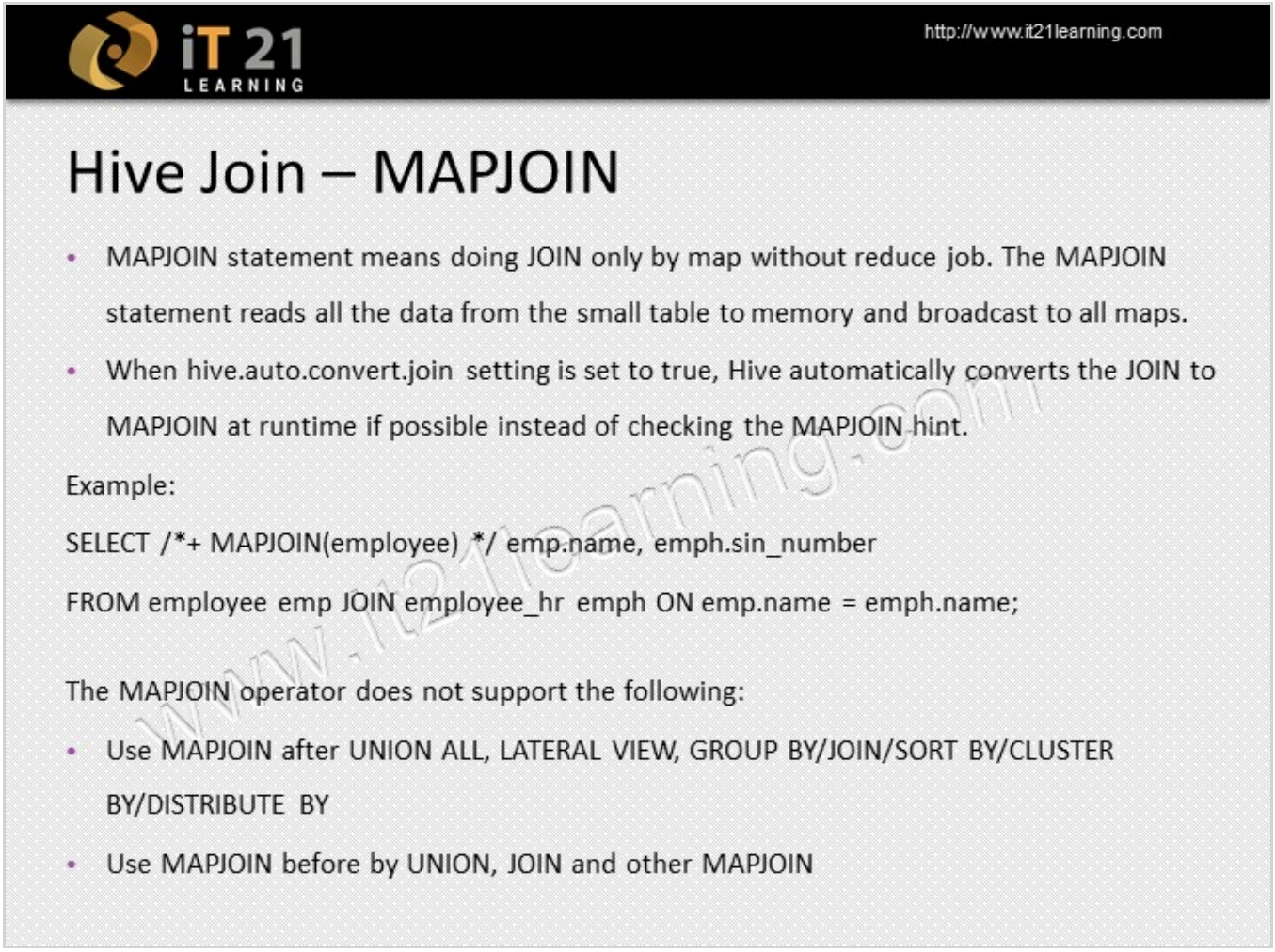
View is not likely optimized by optimizer

Hive Join - Overview

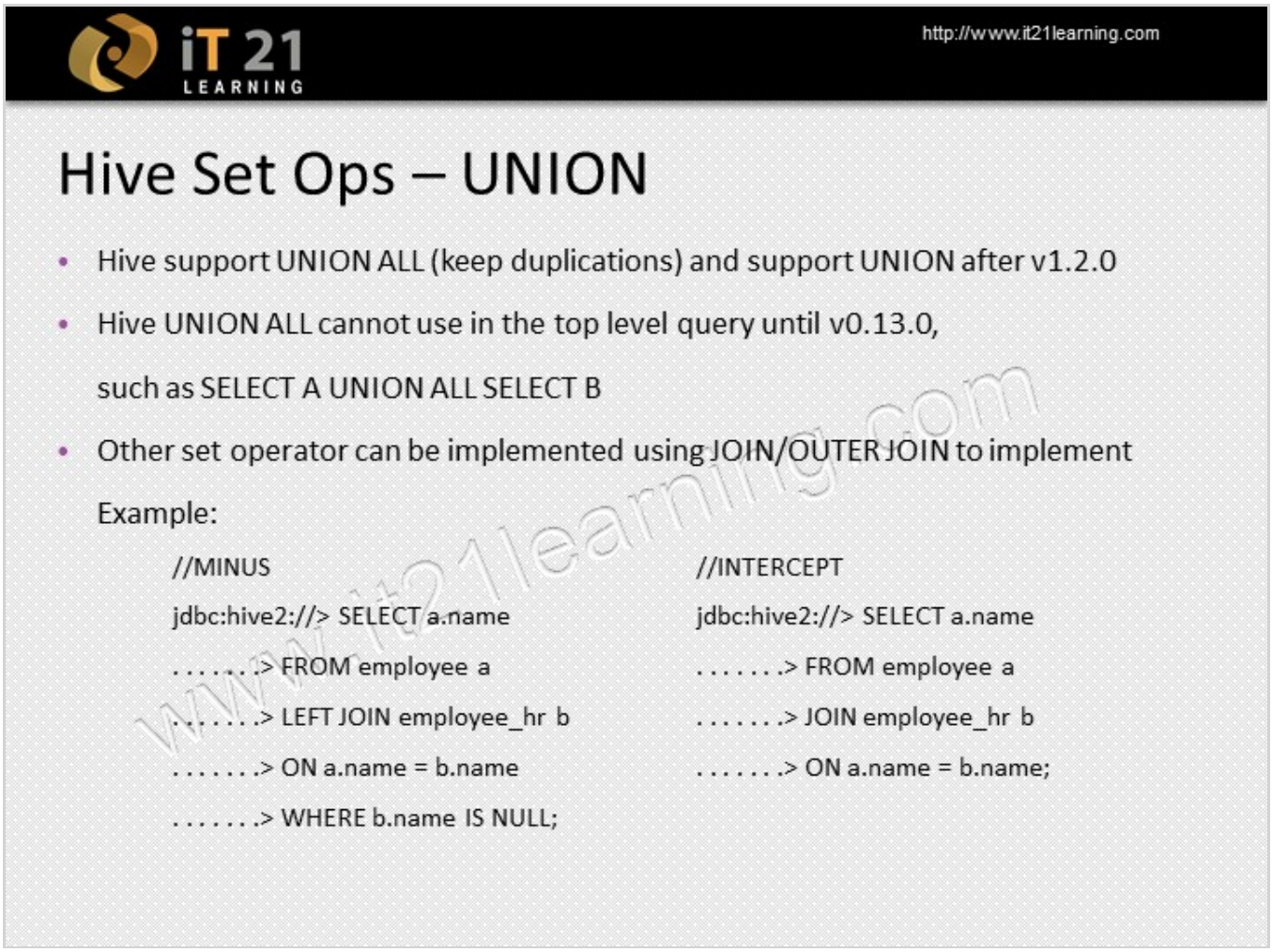


View is not likely optimized by optimizer

Hive Join – MAPJOIN

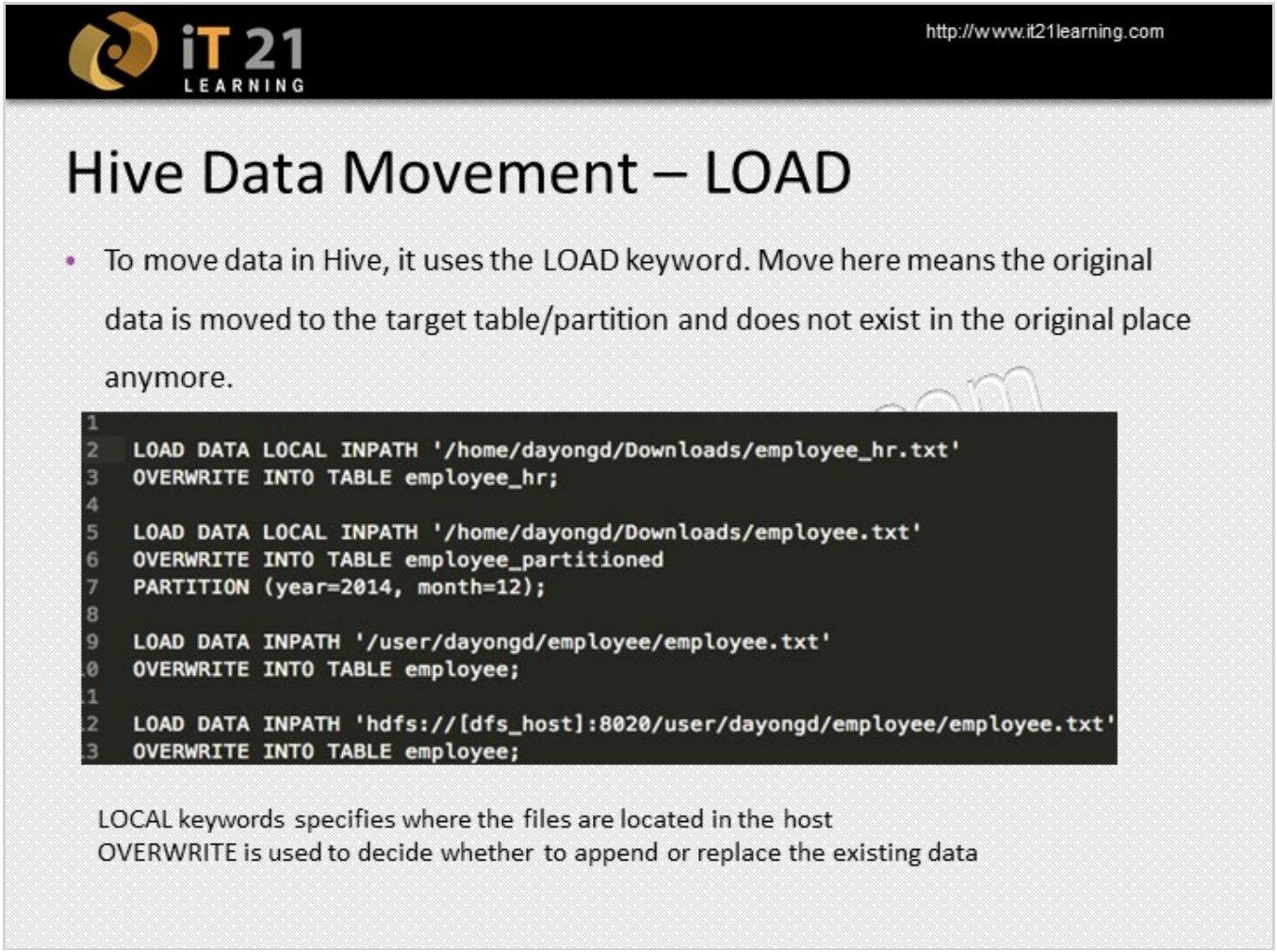


View is not likely optimized by optimizer



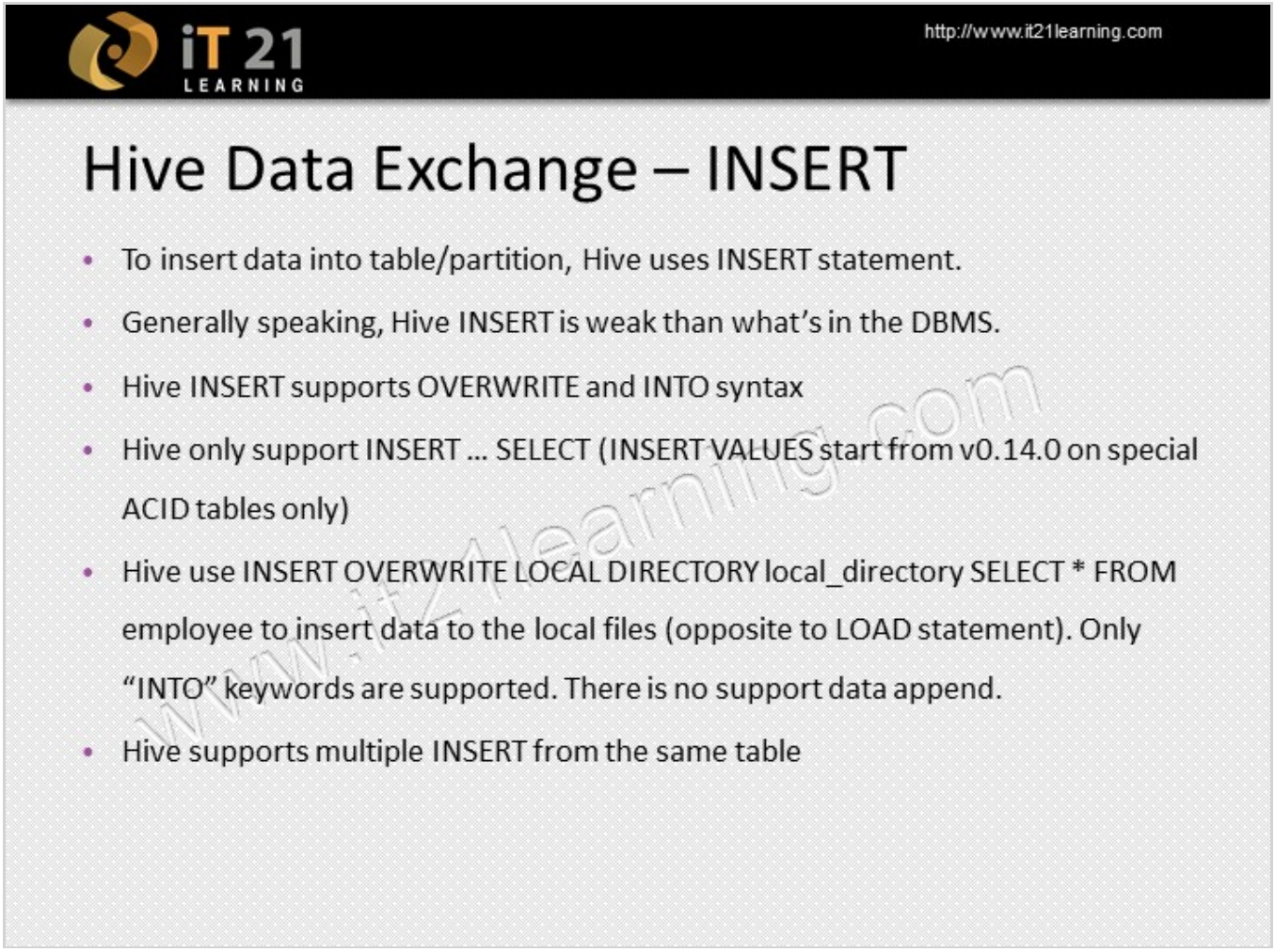
View is not likely optimized by optimizer

Hive Data Movement – LOAD



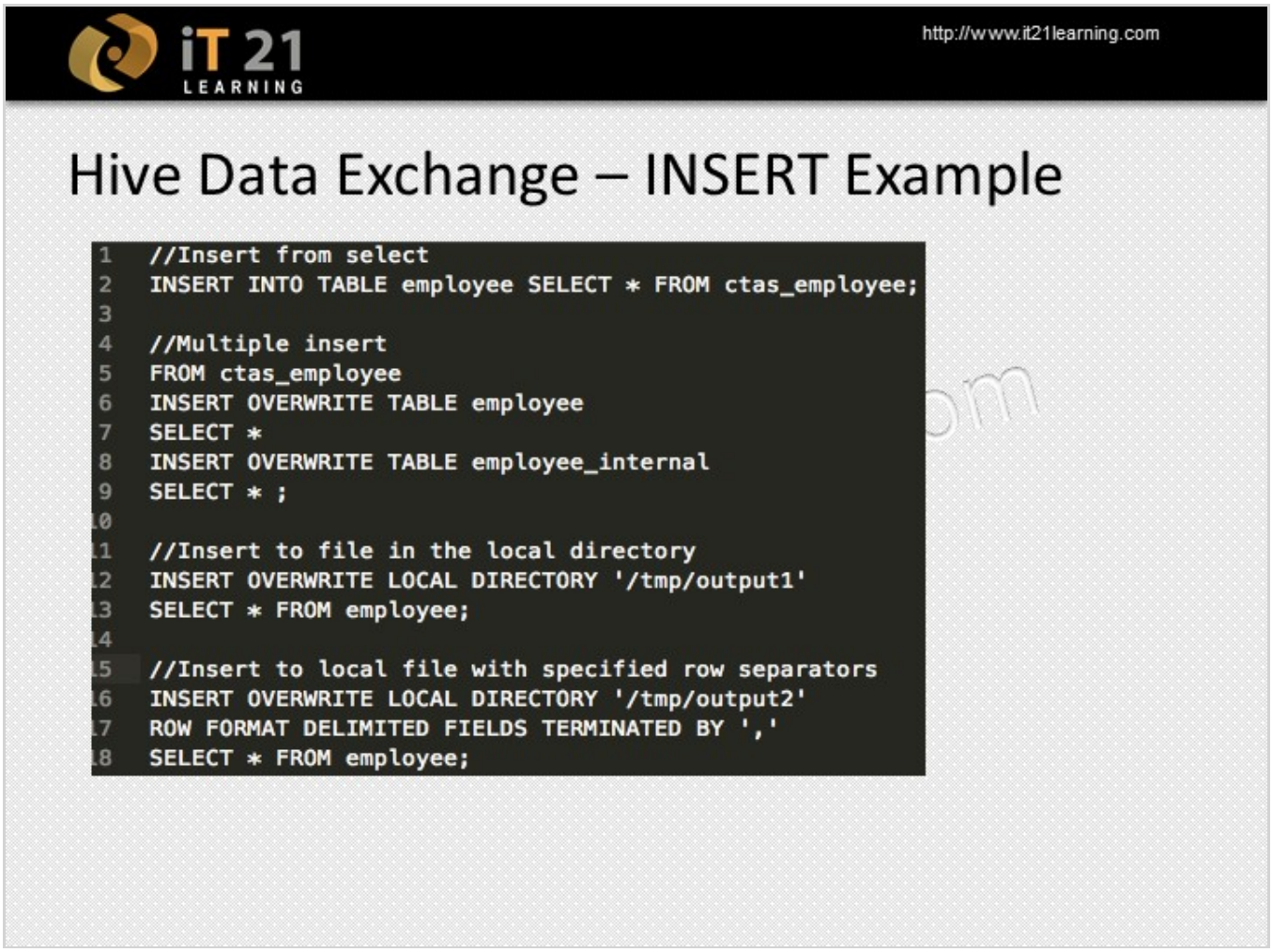
View is not likely optimized by optimizer

Hive Data Exchange – INSERT



View is not likely optimized by optimizer

Hive Data Exchange – INSERT Example

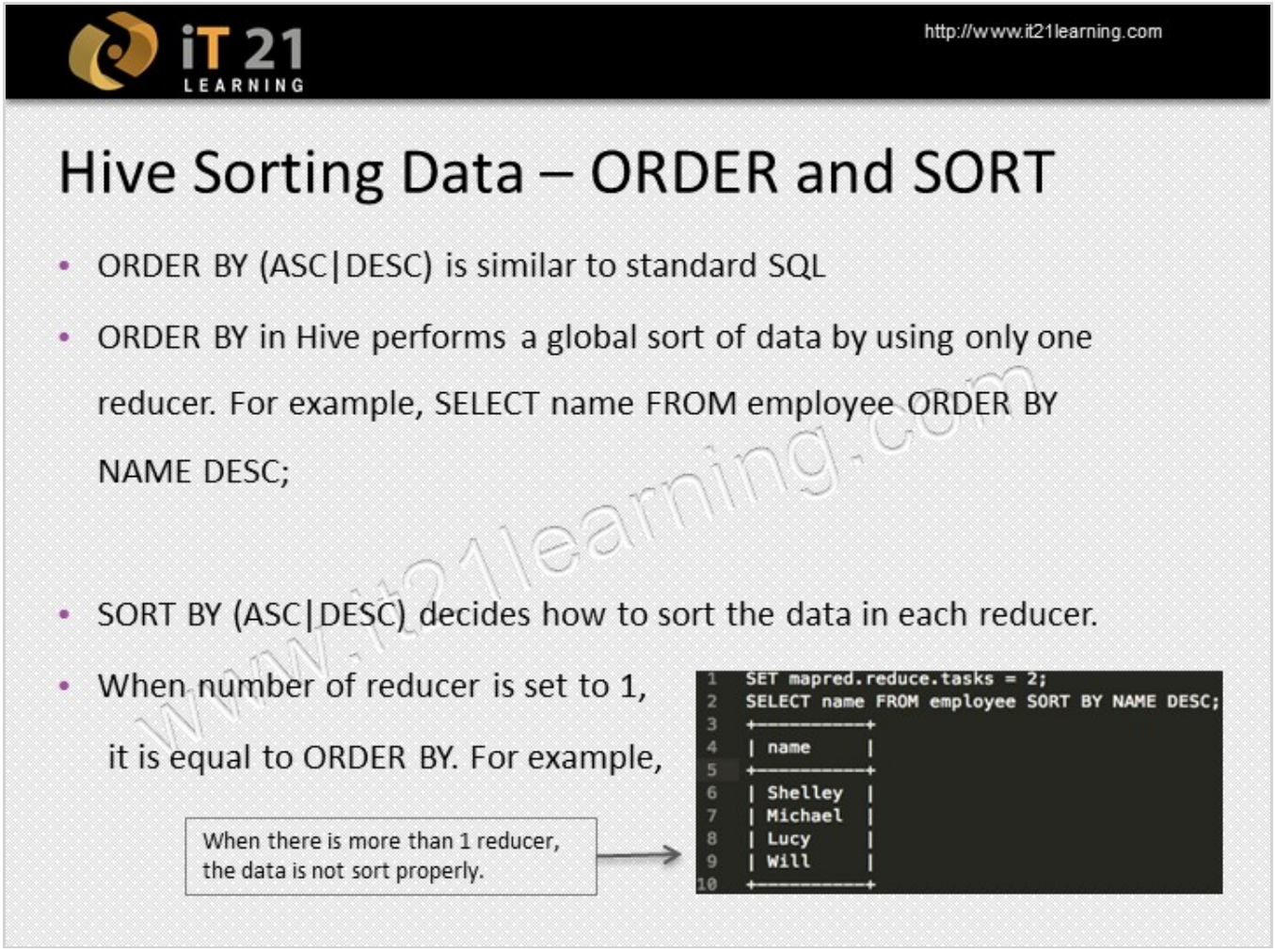


View is not likely optimized by optimizer

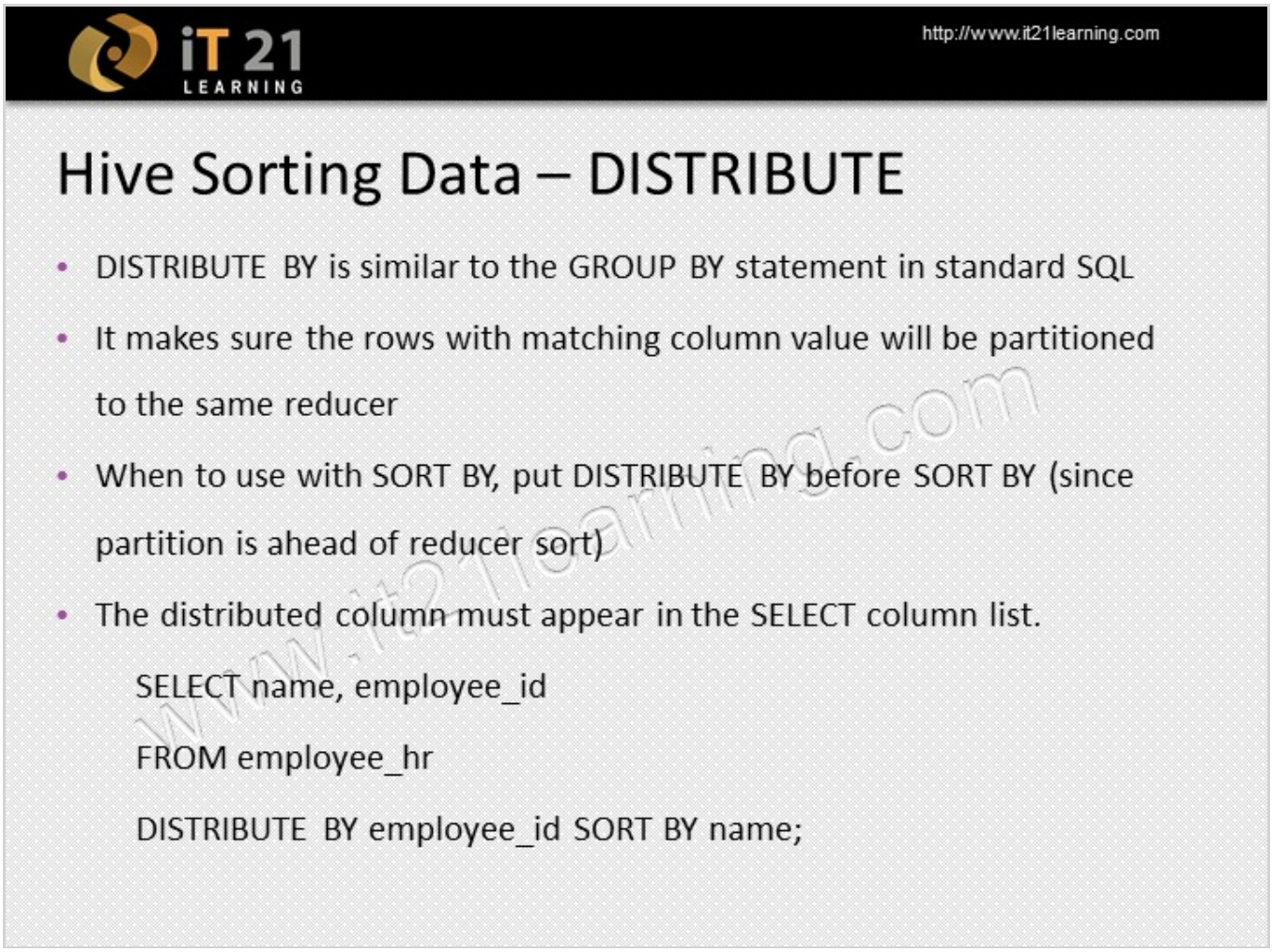
Hive Data Exchange – [EX|IM]PORT

View is not likely optimized by optimizer

Hive Sorting Data – ORDER and SORT

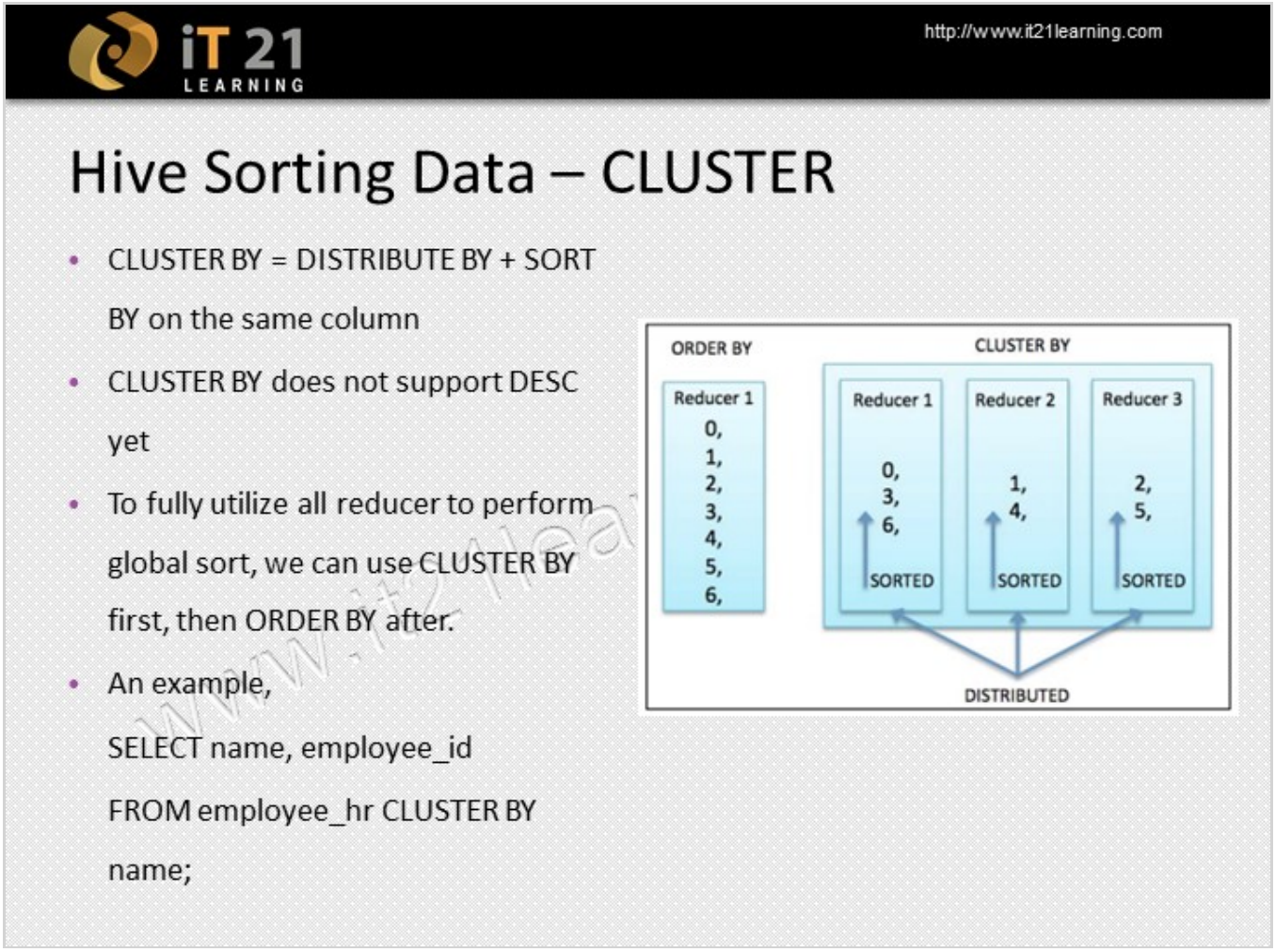


View is not likely optimized by optimizer



View is not likely optimized by optimizer

Hive Sorting Data – CLUSTER



View is not likely optimized by optimizer

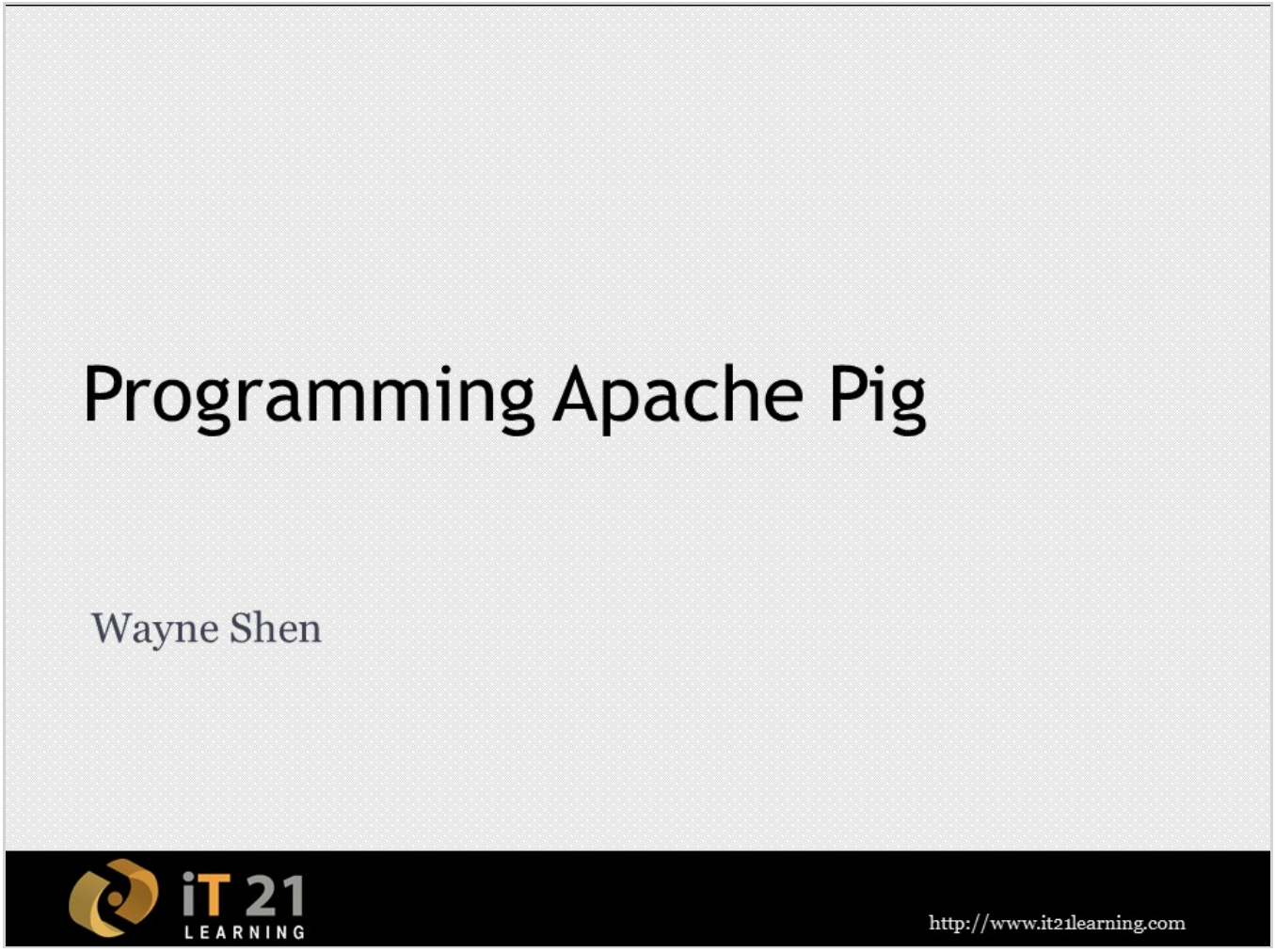
Hive Other Topics – Not Covered

View is not likely optimized by optimizer

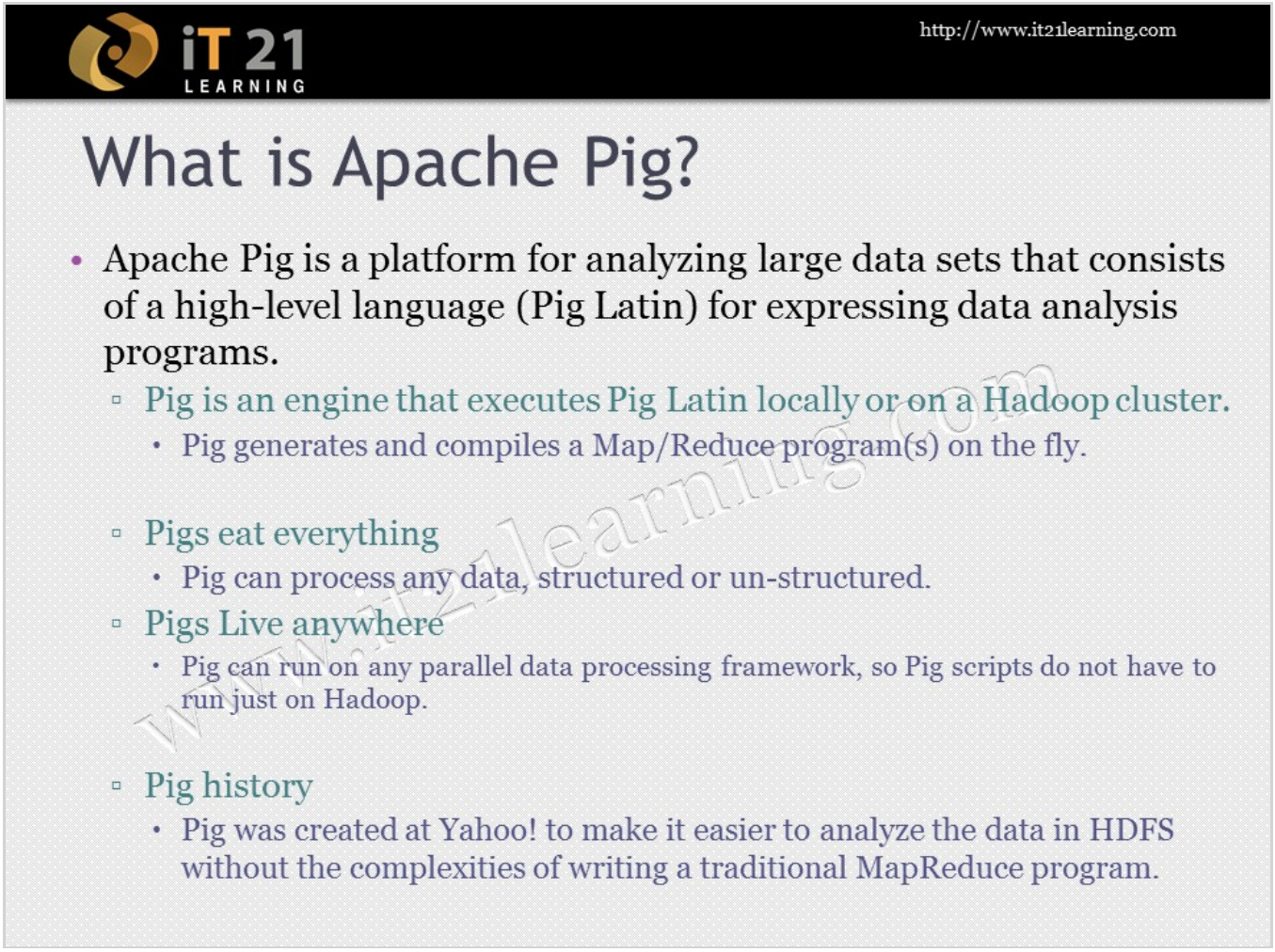

iT 21 LEARNING
http://www.it21learning.com
Q&A
Thank You
www.it21learning.com

40

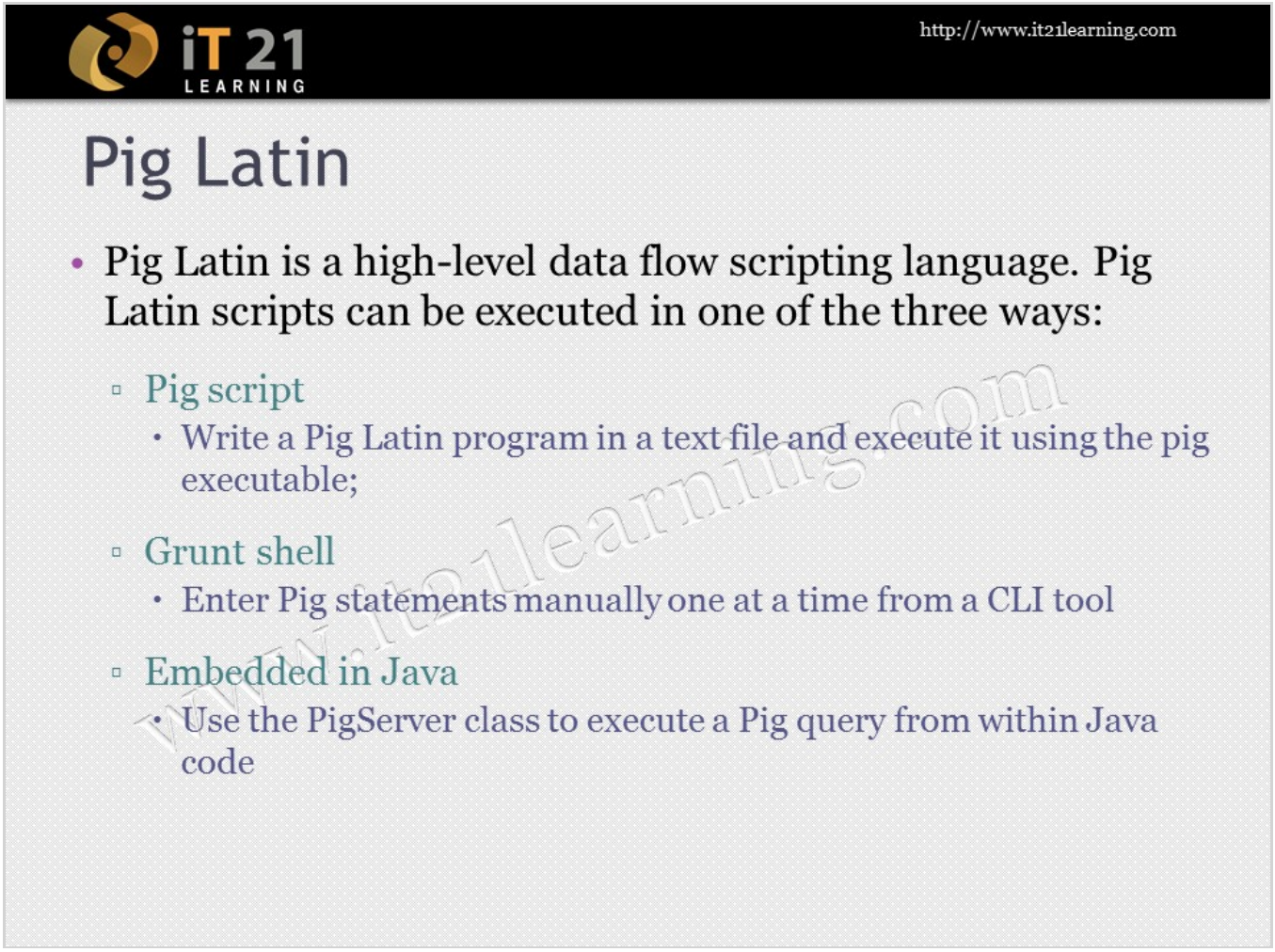
Mastering Apache Pig


Programming Apache Pig
Wayne Shen
iT 21
LEARNING
http://www.it21learning.com

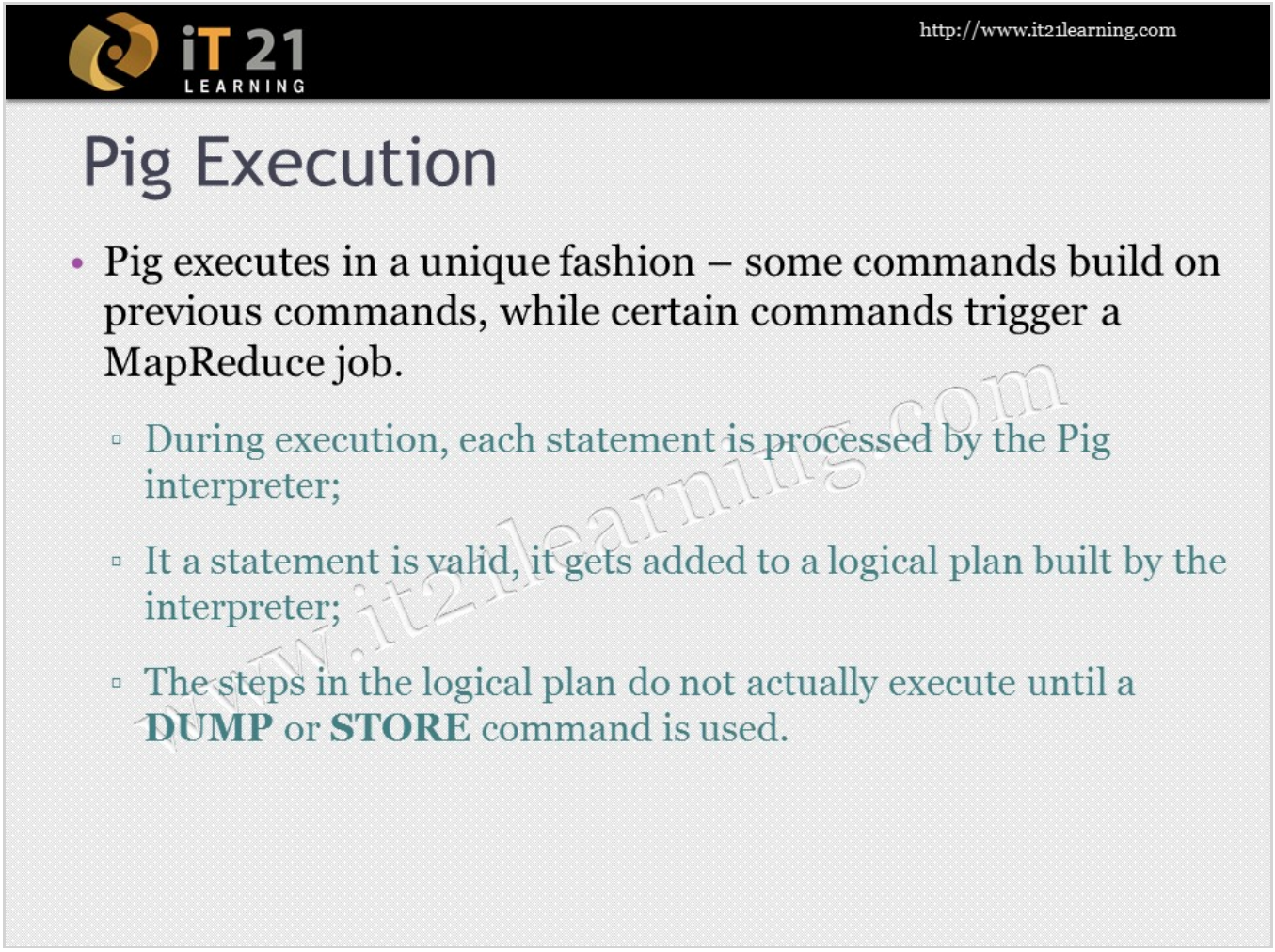
1


iT 21 LEARNING
http://www.it21learning.com
What is Apache Pig?
• Apache Pig is a platform for analyzing large data sets that consists of a high-level language (Pig Latin) for expressing data analysis programs.
▫ Pig is an engine that executes Pig Latin locally or on a Hadoop cluster.
• Pig generates and compiles a Map/Reduce program(s) on the fly.
▫ Pigs eat everything
• Pig can process any data, structured or un-structured.
▫ Pigs Live anywhere
• Pig can run on any parallel data processing framework, so Pig scripts do not have to run just on Hadoop.
▫ Pig history
• Pig was created at Yahoo! to make it easier to analyze the data in HDFS without the complexities of writing a traditional MapReduce program.

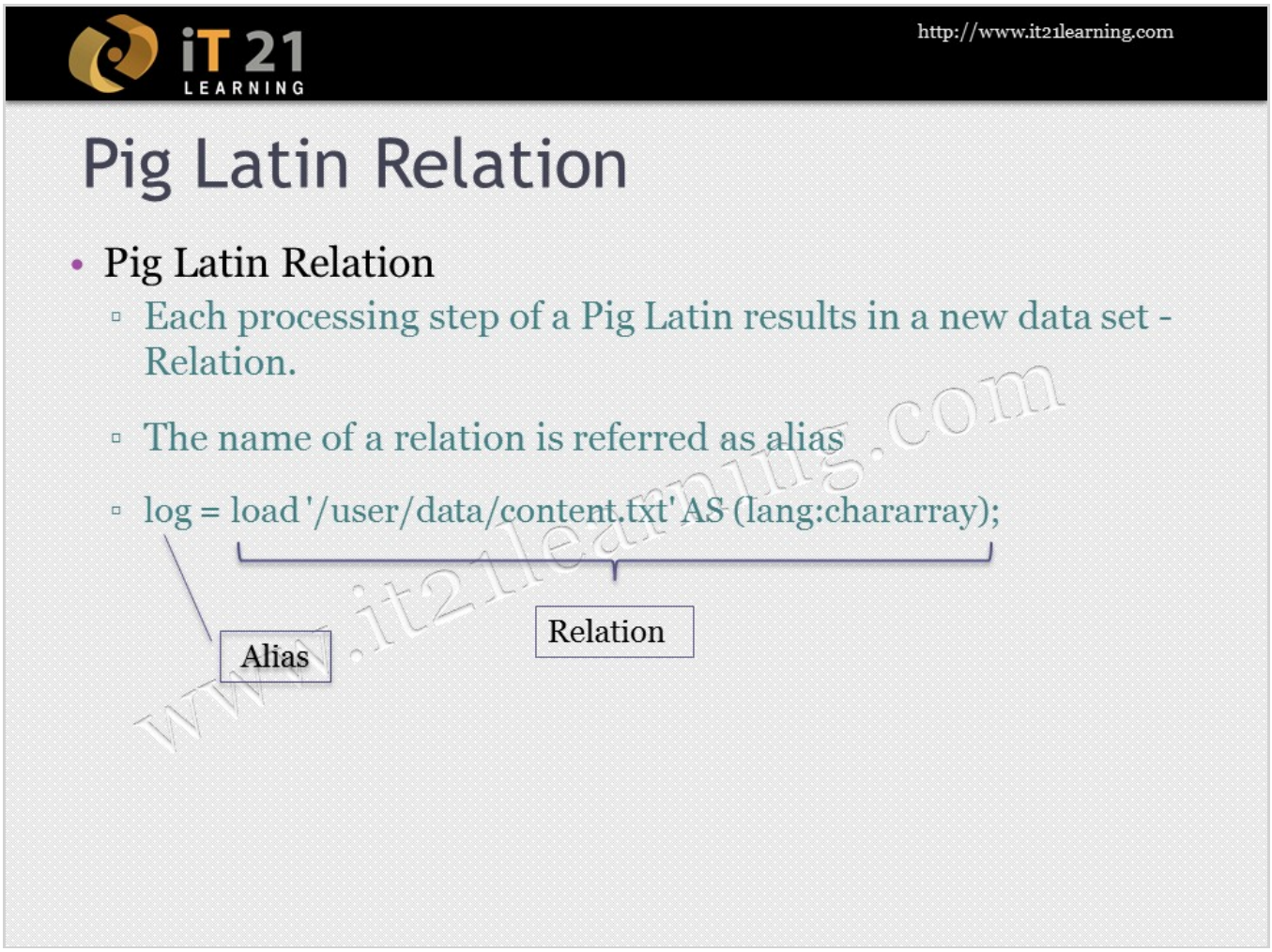
2


iT 21 LEARNING
http://www.it21learning.com
Pig Latin
Pig Latin is a high-level data flow scripting language. Pig Latin scripts can be executed in one of the three ways:
Pig script
Write a Pig Latin program in a text file and execute it using the pig executable;
Grunt shell
Enter Pig statements manually one at a time from a CLI tool
Embedded in Java
Use the PigServer class to execute a Pig query from within Java code
www.it21learning.com

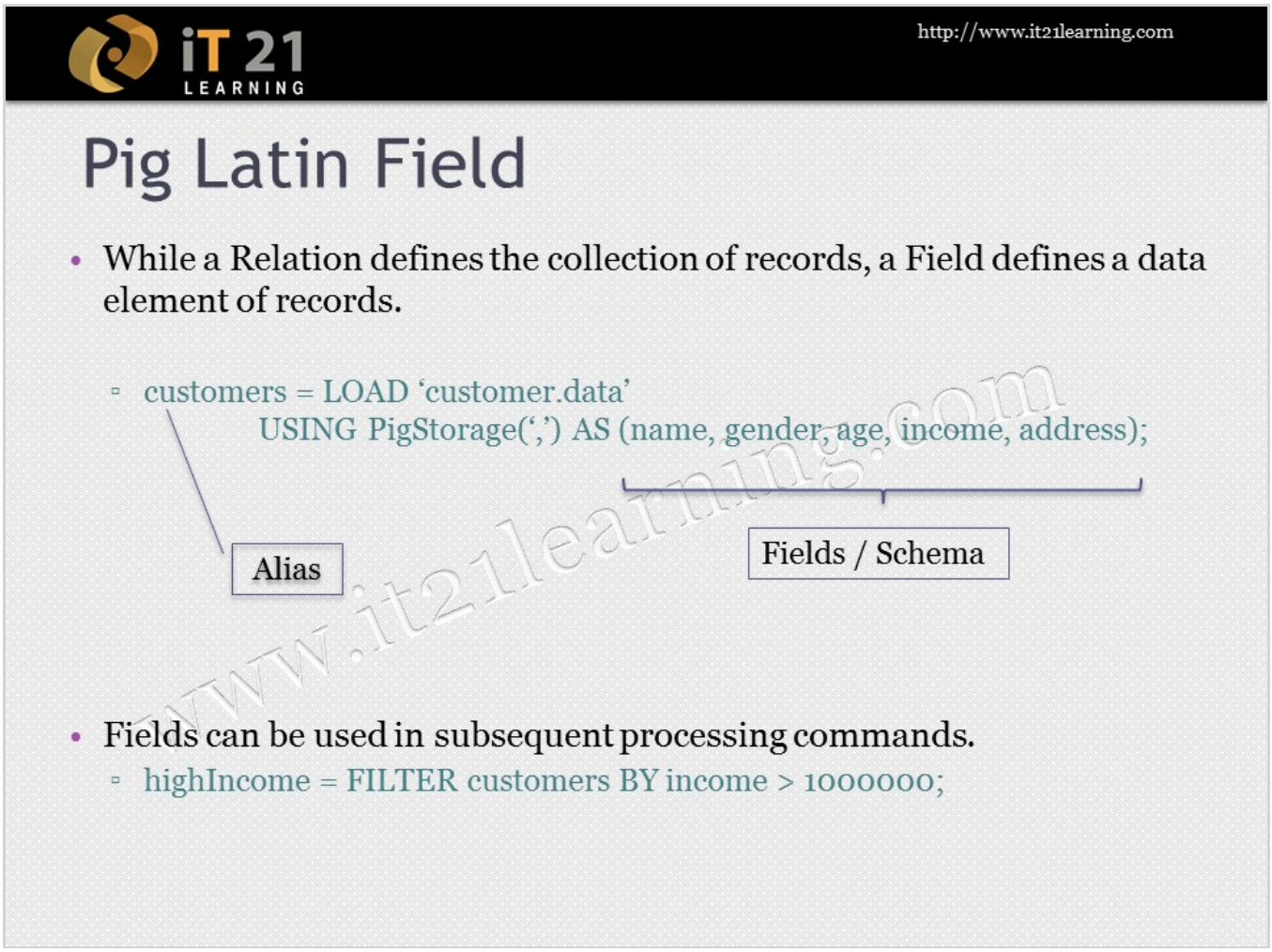
3


iT 21 LEARNING
http://www.it21learning.com
Pig Execution
Pig executes in a unique fashion – some commands build on previous commands, while certain commands trigger a MapReduce job.
During execution, each statement is processed by the Pig interpreter;
It a statement is valid, it gets added to a logical plan built by the interpreter;
The steps in the logical plan do not actually execute until a DUMP or STORE command is used.
www.it21learning.com

4


iT 21 LEARNING
http://www.it21learning.com
Pig Latin Relation
Pig Latin Relation
Each processing step of a Pig Latin results in a new data set - Relation.
The name of a relation is referred as alias
log = load '/user/data/content.txt' AS (lang:chararray);
Alias
Relation
www.it21learning.com

5


iT 21 LEARNING
http://www.it21learning.com
Pig Latin Field
While a Relation defines the collection of records, a Field defines a data element of records.
customers = LOAD 'customer.data'
USING PigStorage(',') AS (name, gender, age, income, address);
Alias
Fields / Schema
Fields can be used in subsequent processing commands.
highIncome = FILTER customers BY income > 1000000;

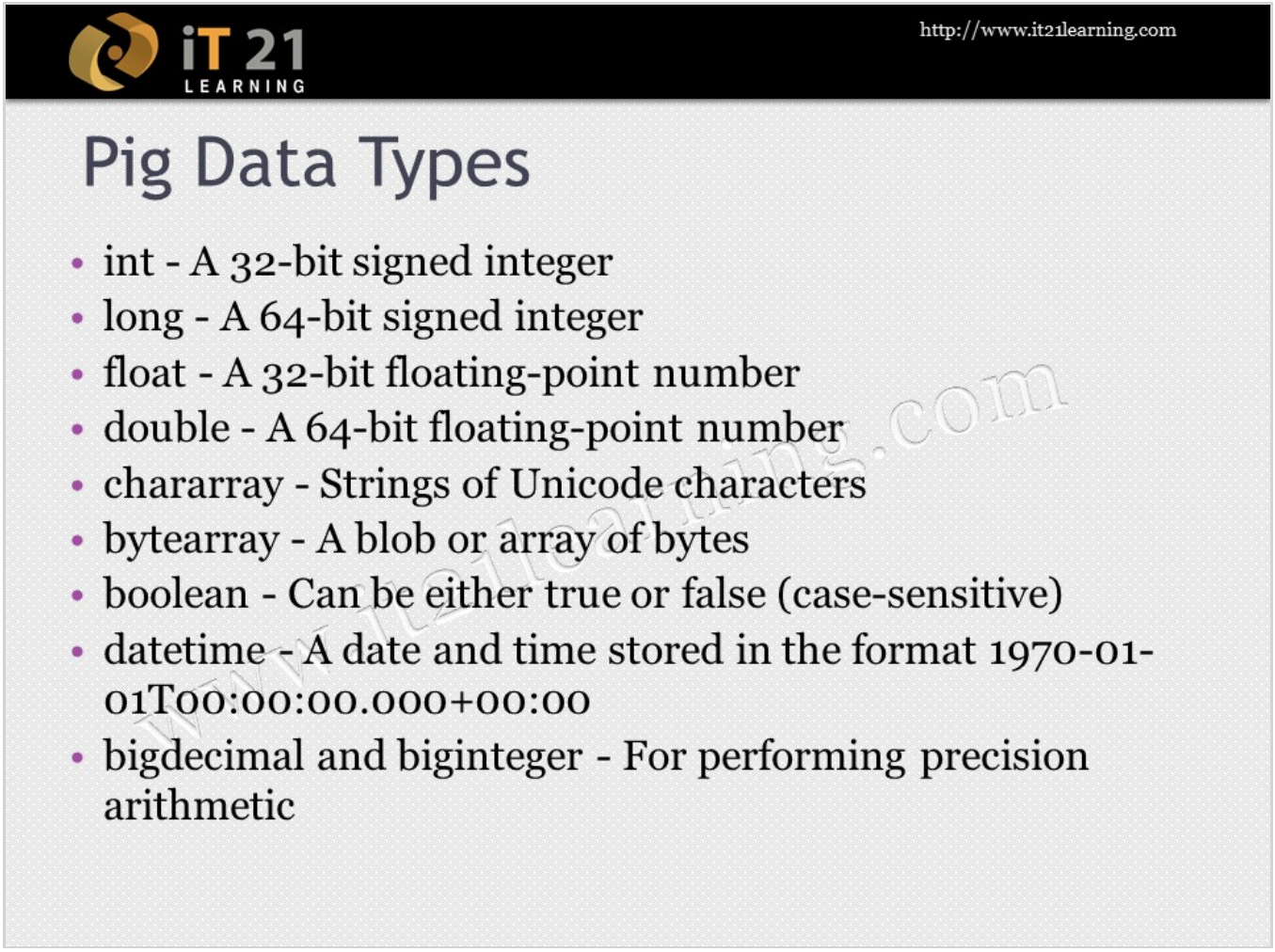
6

Pig Latin DEMO

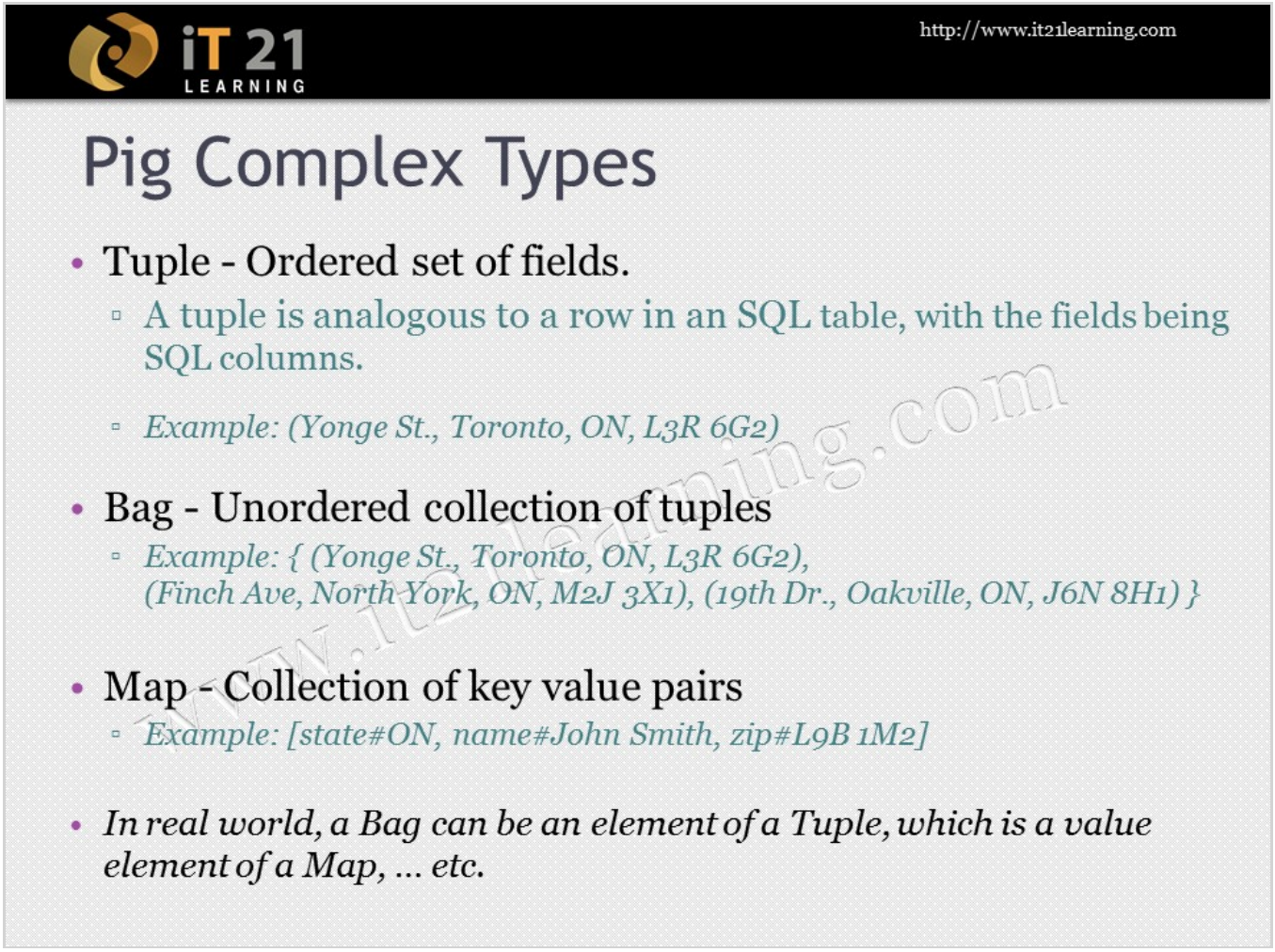
- rs = LOAD '/user/data/full_text.txt' USING PigStorage('\t')
  AS (id:chararray, ts:chararray, location:chararray,
  latitude:float, longitude:float, tweet:chararray);
- tokens = FOREACH rs GENERATE FLATTEN(TOKENIZE(tweet)) AS token;
- grps = GROUP tokens BY token;
- grpcnt = FOREACH grps
  GENERATE group AS token, COUNT(tokens) AS cnt;
- srt = ORDER grpcnt BY cnt DESC;
- lmt = LIMIT srt 50;
- DUMP lmt;
  - Pig doesn't execute until it sees keyword such as DUMP or STORE

7

Pig Data Types

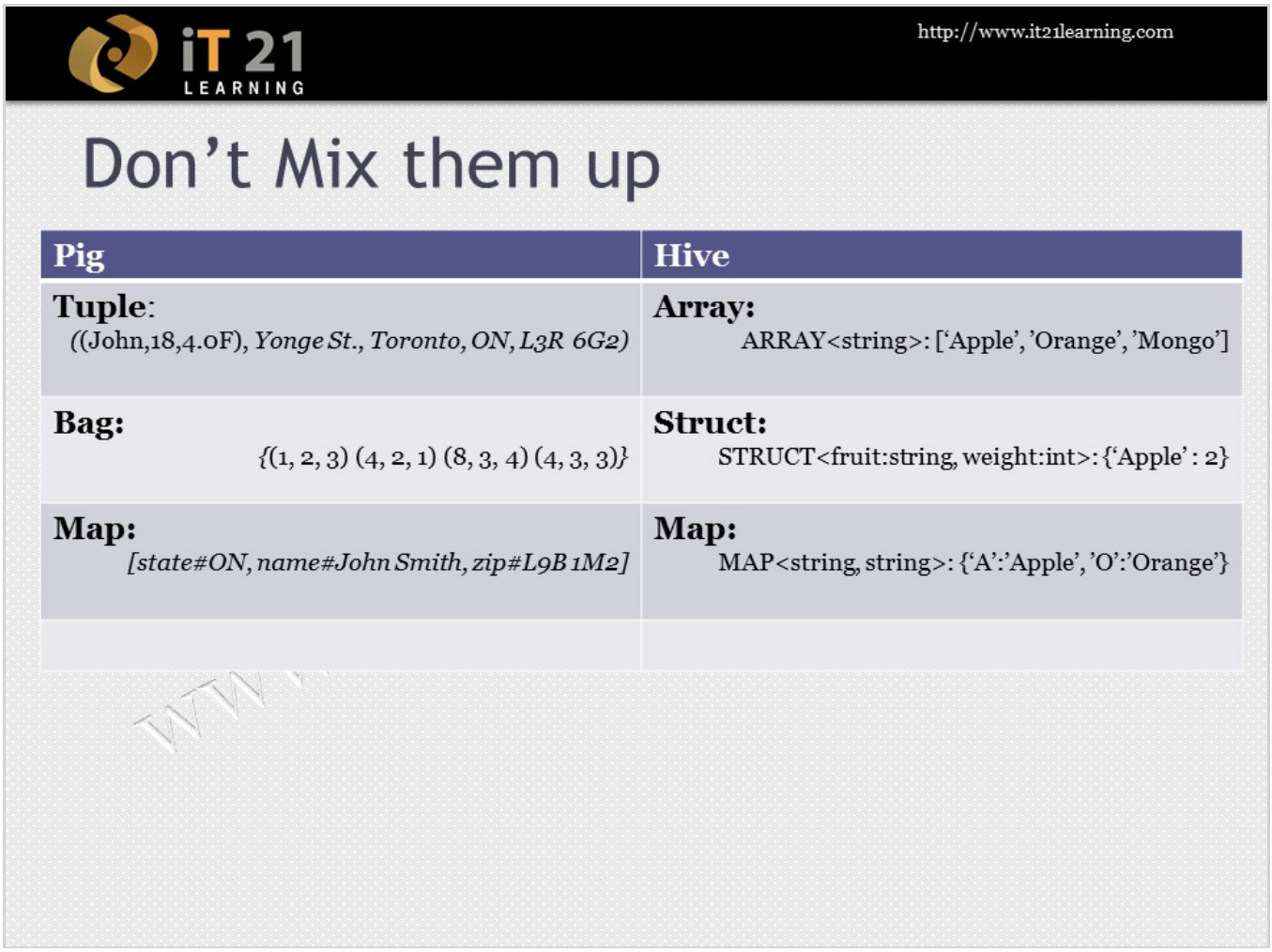


8


iT 21 LEARNING
http://www.it21learning.com
Pig Complex Types
• Tuple - Ordered set of fields.
▫ A tuple is analogous to a row in an SQL table, with the fields being SQL columns.
▫ Example: (Yonge St., Toronto, ON, L3R 6G2)
• Bag - Unordered collection of tuples
▫ Example: { (Yonge St., Toronto, ON, L3R 6G2), (Finch Ave, North York, ON, M2J 3X1), (19th Dr., Oakville, ON, J6N 8H1) }
• Map - Collection of key value pairs
▫ Example: [state#ON, name#John Smith, zip#L9B 1M2]
• In real world, a Bag can be an element of a Tuple, which is a value element of a Map, … etc.

9

Don’t Mix them up



10

Defining a Schema

- When data is well-structured, the schema can be defined when loading the data;
  - customers = LOAD 'customer_data'
    AS (name : chararray, street:chararray, city:chararray, province:chararray);

Fields / Schema

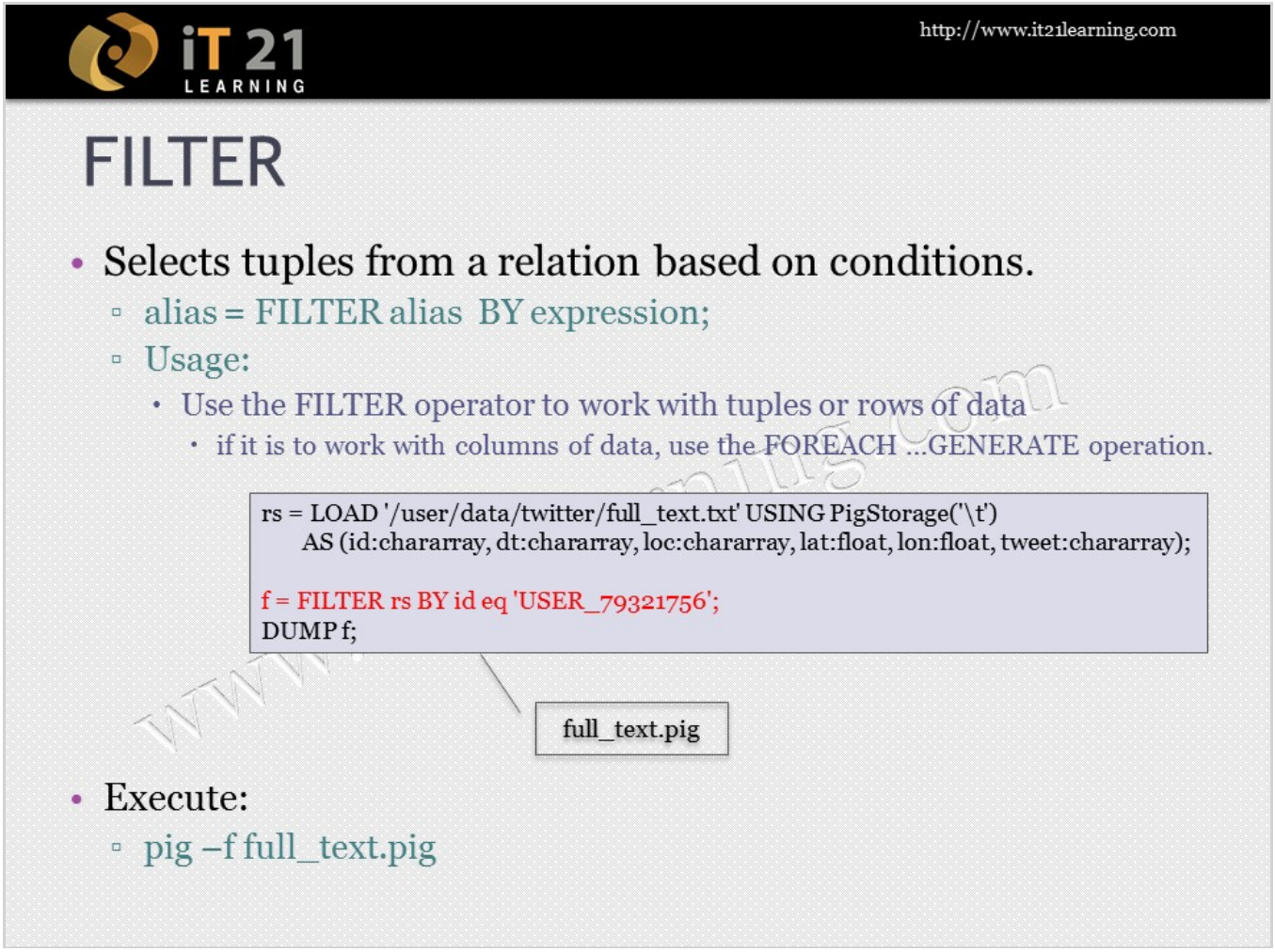
- The schema can also specify complex types.

'John Smith', ('abc Yonge St.', 'Toronto', 'ON', { '416-xxx-2342', '905-123-xxxx' }), 98000
'Mike Doug', ('xyz 19th Ave.', 'Markham', 'ON', { '416-789-xxxx', '905-xxx-1234' }), 79000

```
customers = LOAD 'customer_data' AS
    ( name : chararray,
      contact : tuple { street:chararry, city:chararray, province:chararray, bag: { phone : chararray } ),
      salary : int );
```

- *Once Schema is defined and associated, Pig performs error-checking with it.*

11


iT 21 LEARNING
http://www.it21learning.com
FILTER
• Selects tuples from a relation based on conditions.
▫ alias = FILTER alias BY expression;
▫ Usage:
• Use the FILTER operator to work with tuples or rows of data
• if it is to work with columns of data, use the FOREACH …GENERATE operation.
rs = LOAD '/user/data/twitter/full_text.txt' USING PigStorage('\t')
AS (id:chararray, dt:chararray, loc:chararray, lat:float, lon:float, tweet:chararray);
f = FILTER rs BY id eq 'USER_79321756';
DUMP f;
full_text.pig
• Execute:
▫ pig –f full_text.pig

12

FOREACH … GENERATE

- Generates data transformations based on columns of data.
  - alias = FOREACH alias GENERATE expression [expression ….]
  - Usage:
    - Use the FOREACH …GENERATE operation to work with columns of data
      - if it is to work with tuples or rows of data, use the FILTER operation.
    - FOREACH …GENERATE works with relations (outer bags) as well as inner bags:

```
rs = LOAD '/user/data/twitter/full_text.txt' USING PigStorage('\t')
     AS (id:chararray, dt:chararray, loc:chararray, lat:float, lon:float, tweet:chararray);

c = FOREACH rs GENERATE id, lat, lon;

f = FILTER c BY id eq 'USER_79321756';
DUMP f;
```

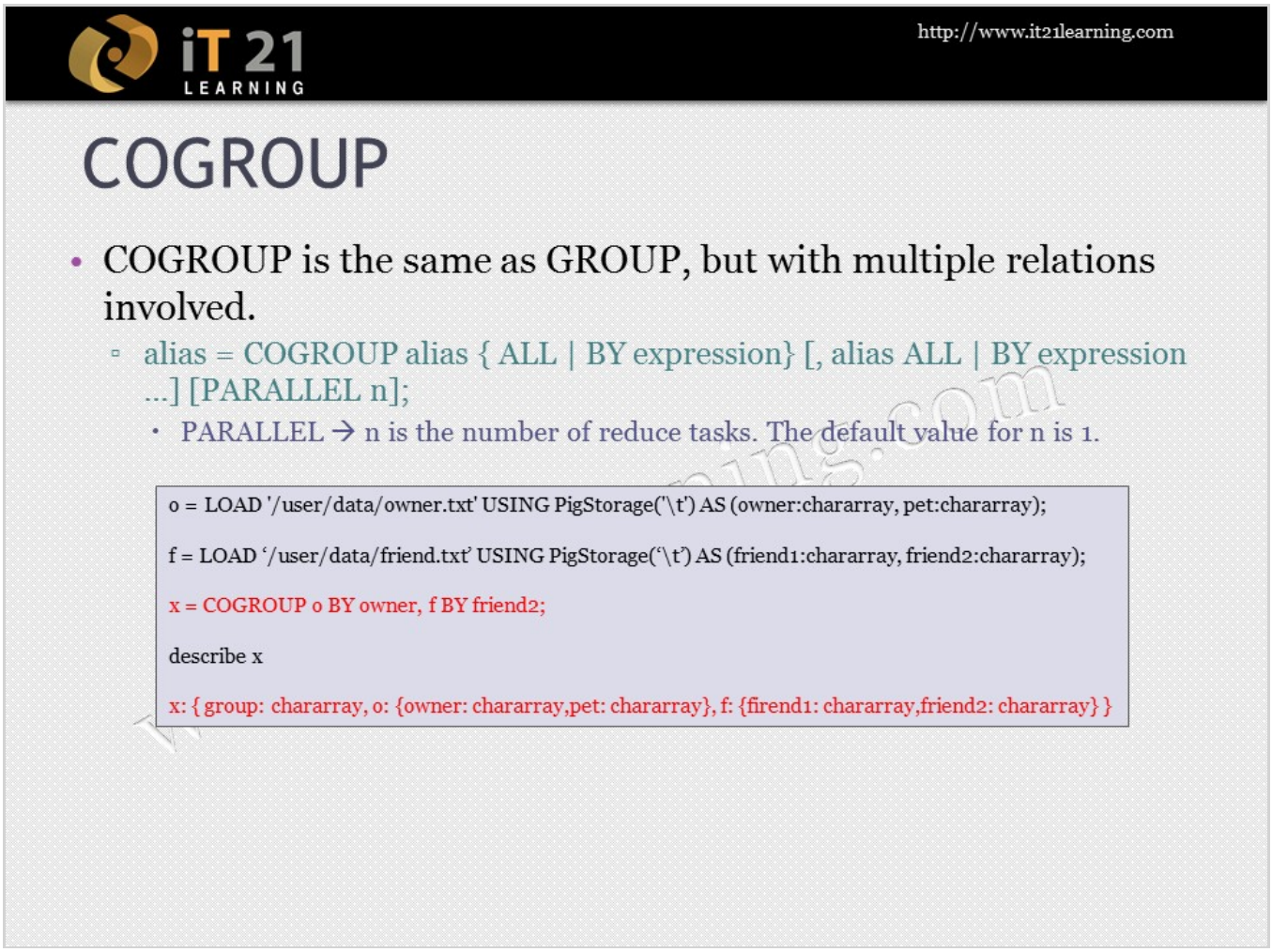
13

GROUP

- Groups the data in one or multiple relations.
  - alias = GROUP alias { ALL | BY expression} [USING 'collected'];
    - When ALL used, all tuples go to a single group.
    - "collected" → Allows for more efficient computation of a group if the loader guarantees that the data for the same key is continuous and is given to a single map. The efficiency is achieved by performing the group operation in map rather than reduce.
  - Usage:
    - The GROUP operator groups together tuples that have the same group key.
    - The result of a GROUP operation is a relation that includes one tuple per group. This tuple contains two fields:
      - The first field is named "group" and is the same type as the group key.
      - The second field takes the name of the original relation and is type bag.


```
rs = LOAD '/user/data/twitter/full_text.txt' USING PigStorage('\t')
    AS (id:chararray, dt:chararray, loc:chararray, lat:float, lon:float, tweet:chararray);
c = FOREACH rs GENERATE id, (lat, lon) AS pos, tweet;
g = GROUP c BY pos;
DUMP g;
n = FOREACH g GENERATE group AS pos, COUNT(c) AS cnt;
DUMP n;
```

14


iT 21 LEARNING
http://www.it21learning.com
COGROUP
• COGROUP is the same as GROUP, but with multiple relations involved.
▫ alias = COGROUP alias { ALL | BY expression} [, alias ALL | BY expression ...] [PARALLEL n];
• PARALLEL → n is the number of reduce tasks. The default value for n is 1.
o = LOAD '/user/data/owner.txt' USING PigStorage('\t') AS (owner:chararray, pet:chararray);
f = LOAD '/user/data/friend.txt' USING PigStorage('\t') AS (friend1:chararray, friend2:chararray);
x = COGROUP o BY owner, f BY friend2;
describe x
x: { group: chararray, o: {owner: chararray,pet: chararray}, f: {firend1: chararray,friend2: chararray} }

15

ORDER

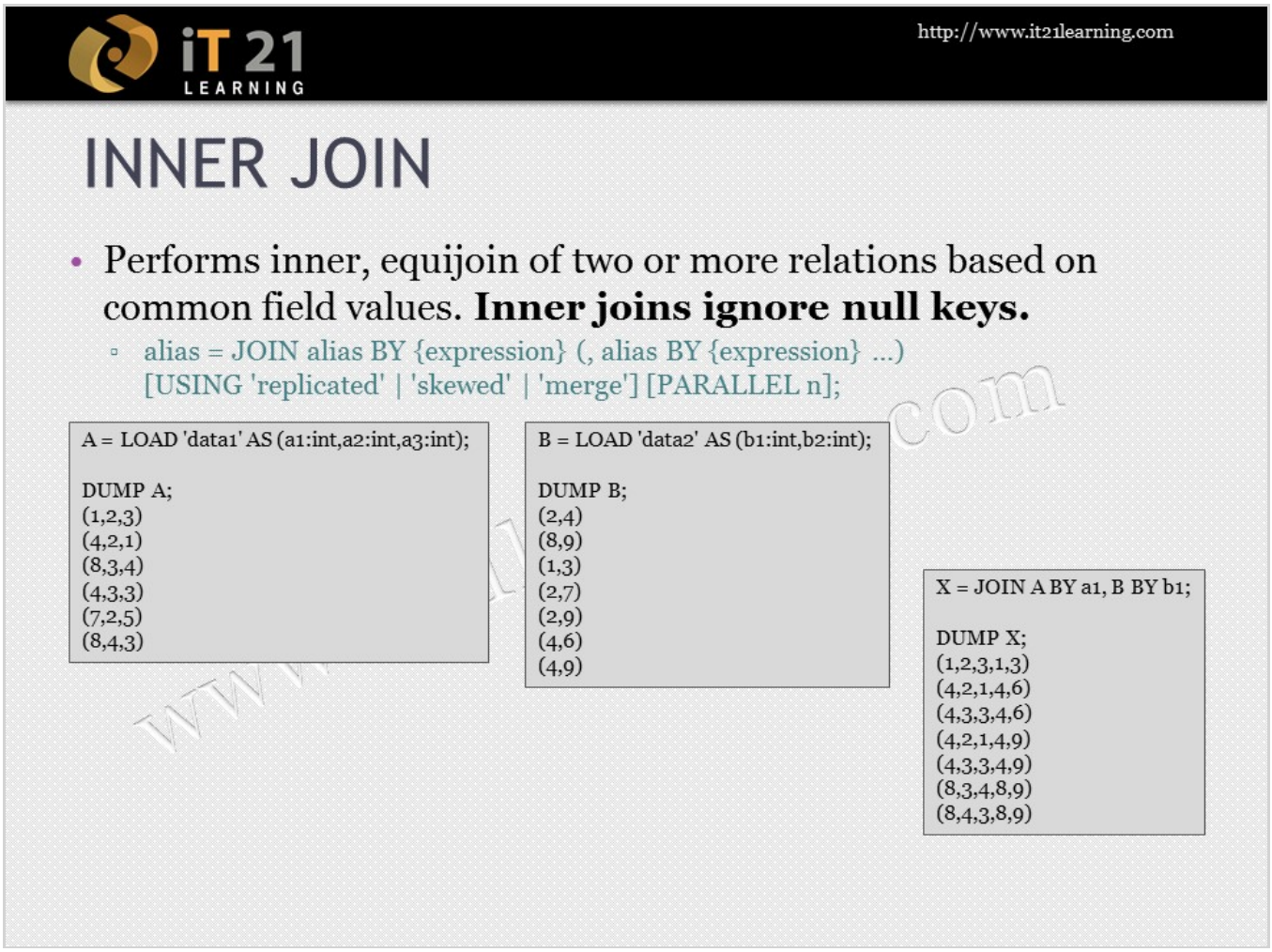
- Sorts a relation based on one or more fields.
  - alias = ORDER alias BY { * [ASC|DESC] | field_alias [ASC|DESC] [, field_alias [ASC|DESC] ...] } [PARALLEL n];

```
A = LOAD 'data' AS (a1:int,a2:int,a3:int);

DUMP A;
(1,2,3)
(4,2,1)
(8,3,4)
(4,3,3)
(7,2,5)
(8,4,3)
```

```
X = ORDER A BY a3 DESC;

DUMP X;
(7,2,5)
(8,3,4)
(1,2,3)
(4,3,3)
(8,4,3)
(4,2,1)
```

  - If relation A is sorted to produce relation X (X = ORDER A BY * DESC;) relations A and X still contain the same thing.
  - If the contents of relation X (DUMP X;) is dumped, the order is guaranteed (descending).
  - However, if relation X (Y = FILTER X BY $0 > 1;) is further processed, there is no guarantee that the contents will be processed in the order originally specified (descending).

16


iT 21 LEARNING
http://www.it21learning.com
INNER JOIN
Performs inner, equijoin of two or more relations based on common field values. Inner joins ignore null keys.
alias = JOIN alias BY {expression} (, alias BY {expression} ...) [USING 'replicated' | 'skewed' | 'merge'] [PARALLEL n];
A = LOAD 'data1' AS (a1:int,a2:int,a3:int);
DUMP A;
(1,2,3)
(4,2,1)
(8,3,4)
(4,3,3)
(7,2,5)
(8,4,3)
B = LOAD 'data2' AS (b1:int,b2:int);
DUMP B;
(2,4)
(8,9)
(1,3)
(2,7)
(2,9)
(4,6)
(4,9)
X = JOIN A BY a1, B BY b1;
DUMP X;
(1,2,3,1,3)
(4,2,1,4,6)
(4,3,3,4,6)
(4,2,1,4,9)
(4,3,3,4,9)
(8,3,4,8,9)
(8,4,3,8,9)

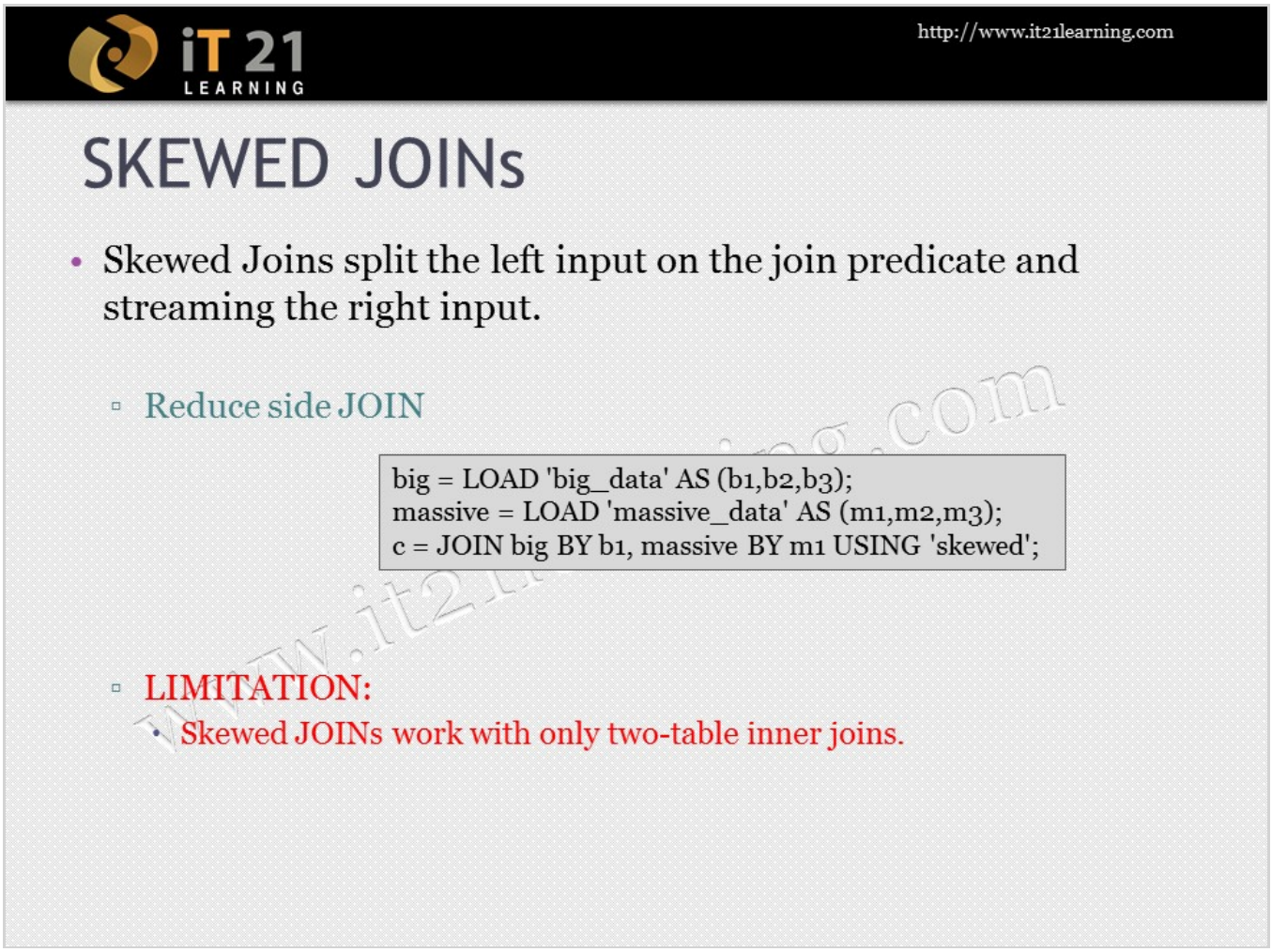
17

REPLICATED JOINs

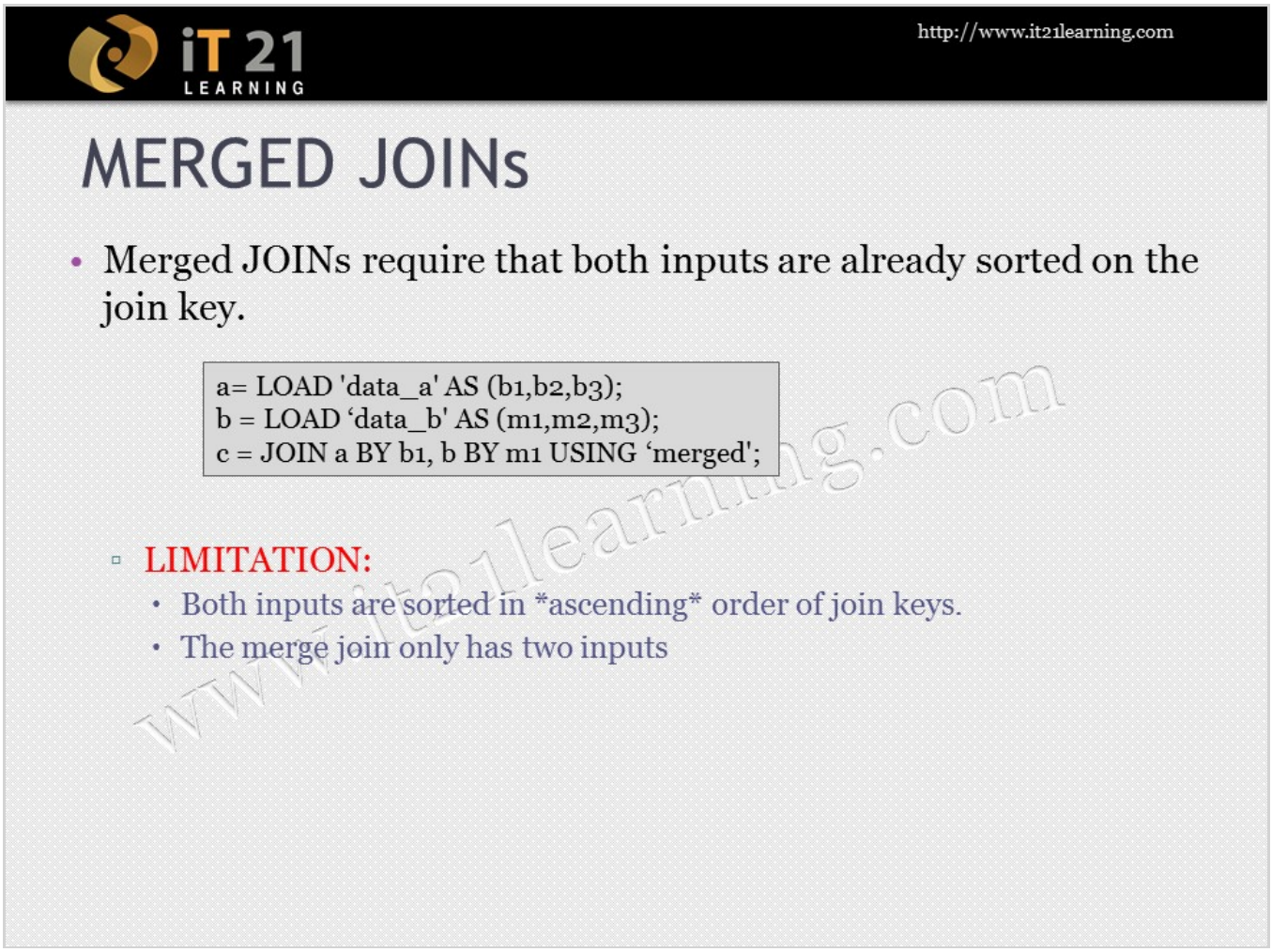
- In REPLICATED JOINs, the large relation is followed by one or more small relations. The small relations must be small enough to fit into main memory; if they don't, the process fails and an error is generated.
  - Map side JOIN

```
big = LOAD 'big_data' AS (b1,b2,b3);

tiny = LOAD 'tiny_data' AS (t1,t2,t3);

mini = LOAD 'mini_data' AS (m1,m2,m3);

c = JOIN big BY b1, tiny BY t1, mini BY m1 USING 'replicated';
```

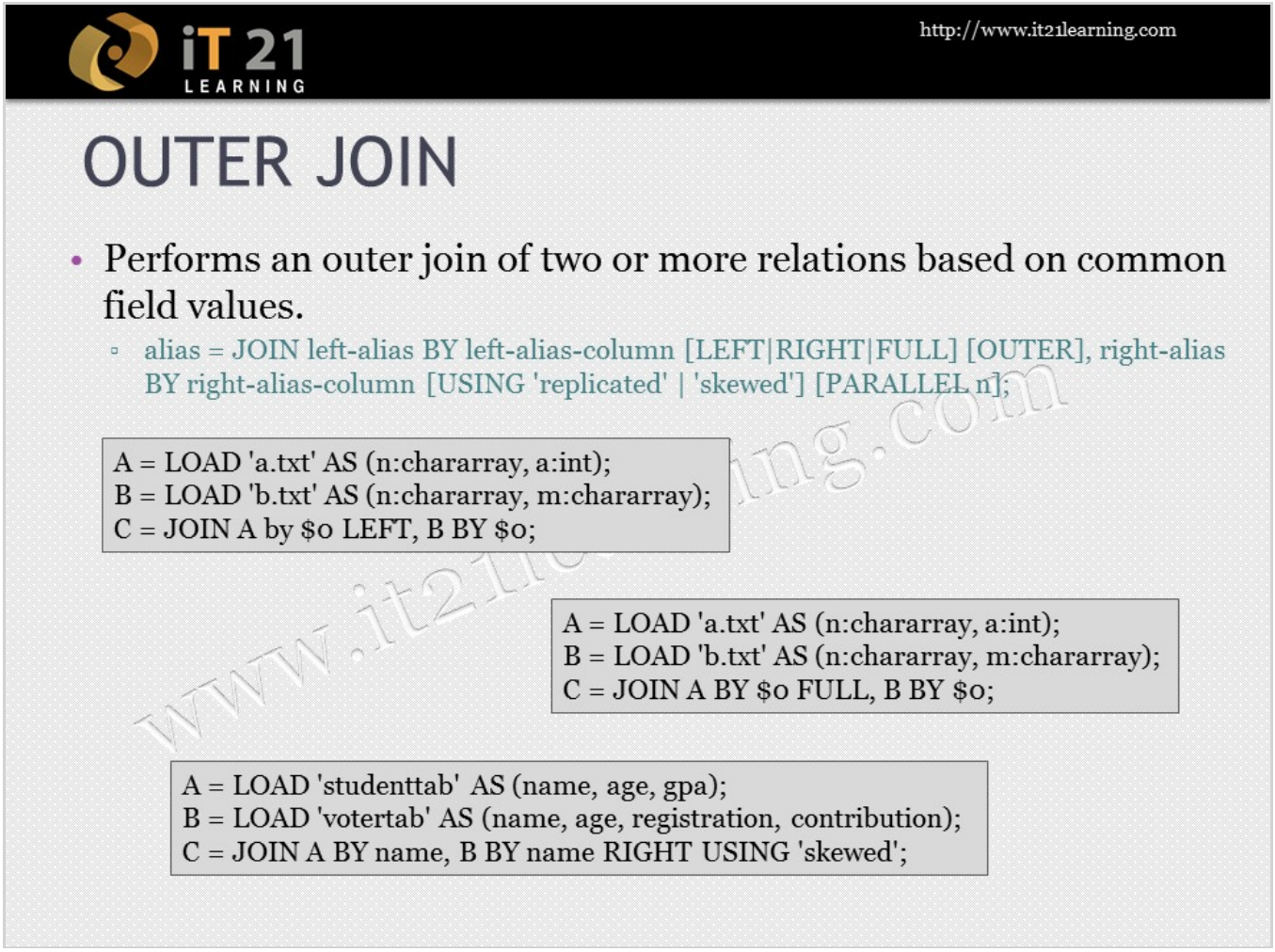

18


iT 21 LEARNING
http://www.it21learning.com
SKEWED JOINs
Skewed Joins split the left input on the join predicate and streaming the right input.
Reduce side JOIN
big = LOAD 'big_data' AS (b1,b2,b3);
massive = LOAD 'massive_data' AS (m1,m2,m3);
c = JOIN big BY b1, massive BY m1 USING 'skewed';
LIMITATION:
Skewed JOINs work with only two-table inner joins.

19

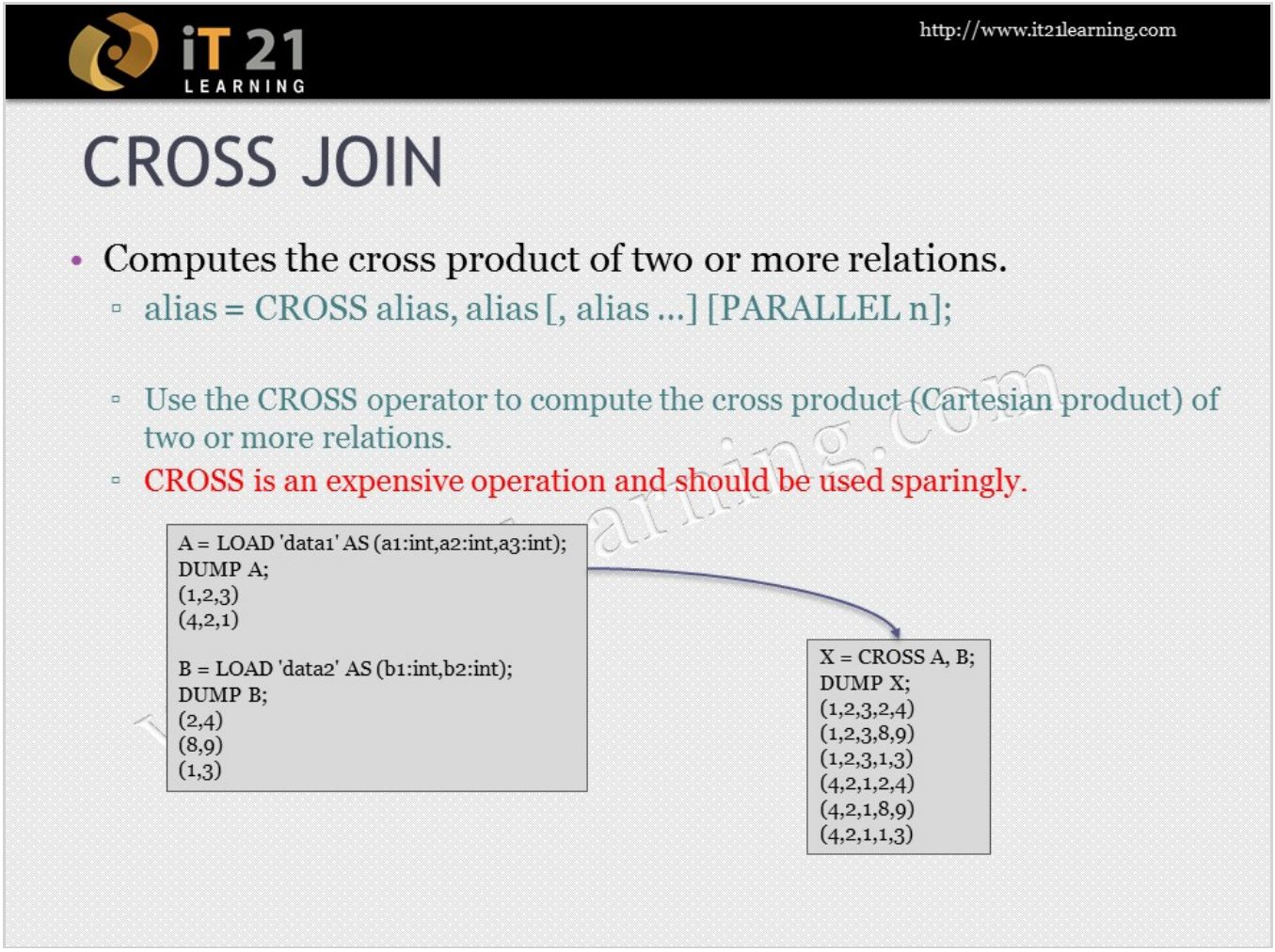

iT 21 LEARNING
http://www.it21learning.com
MERGED JOINs
• Merged JOINs require that both inputs are already sorted on the join key.
a= LOAD 'data_a' AS (b1,b2,b3);
b = LOAD 'data_b' AS (m1,m2,m3);
c = JOIN a BY b1, b BY m1 USING 'merged';
LIMITATION:
• Both inputs are sorted in *ascending* order of join keys.
• The merge join only has two inputs
www.it21learning.com

20


iT 21 LEARNING
http://www.it21learning.com
OUTER JOIN
• Performs an outer join of two or more relations based on common field values.
▫ alias = JOIN left-alias BY left-alias-column [LEFT|RIGHT|FULL] [OUTER], right-alias BY right-alias-column [USING 'replicated' | 'skewed'] [PARALLEL n];
A = LOAD 'a.txt' AS (n:chararray, a:int);
B = LOAD 'b.txt' AS (n:chararray, m:chararray);
C = JOIN A by $0 LEFT, B BY $0;
A = LOAD 'a.txt' AS (n:chararray, a:int);
B = LOAD 'b.txt' AS (n:chararray, m:chararray);
C = JOIN A BY $0 FULL, B BY $0;
A = LOAD 'studenttab' AS (name, age, gpa);
B = LOAD 'votertab' AS (name, age, registration, contribution);
C = JOIN A BY name, B BY name RIGHT USING 'skewed';

21

CROSS JOIN



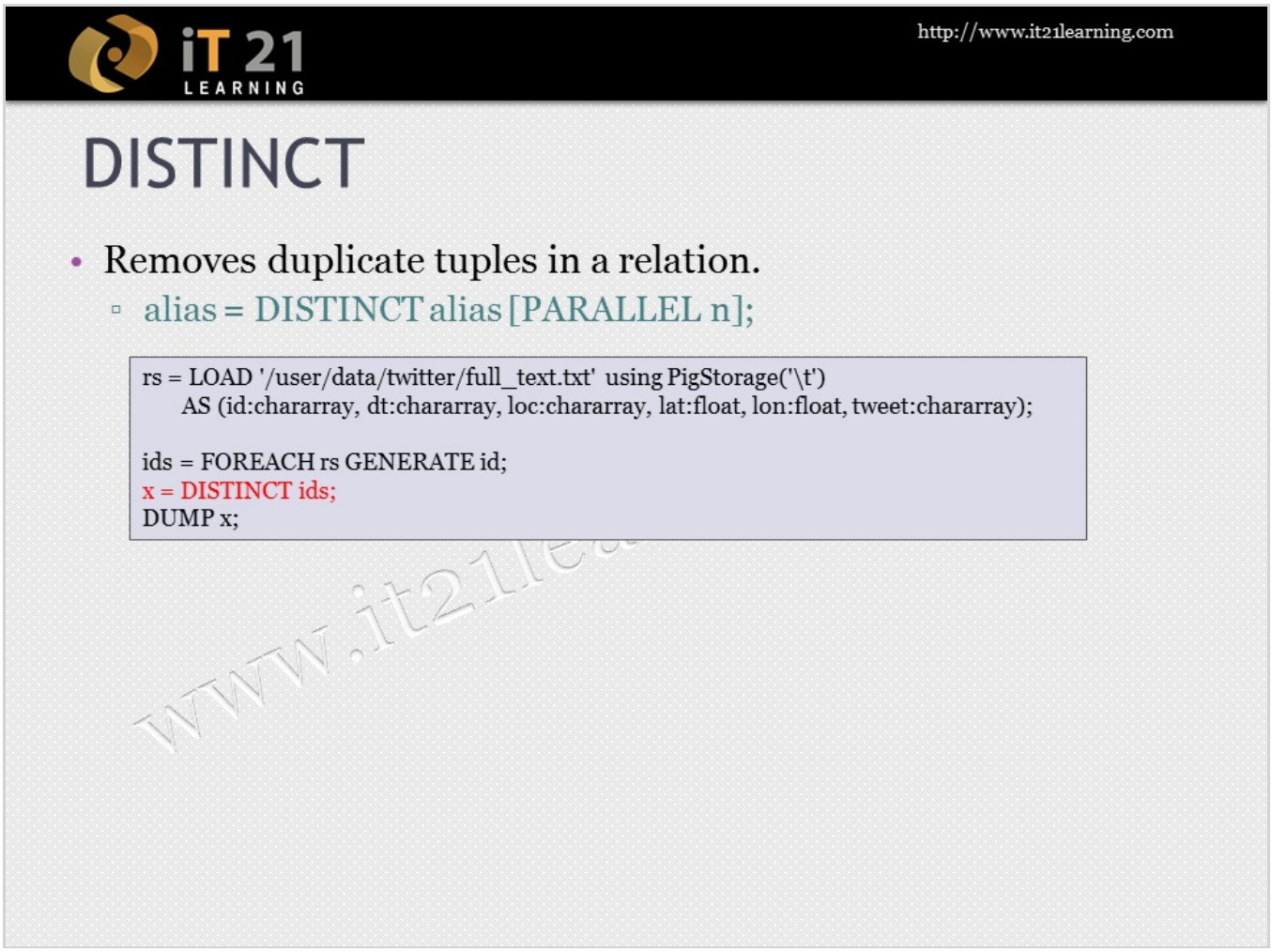
22

UNION

- Computes the union of two or more relations.
  - alias = UNION alias, alias [, alias ...];
  - Does not preserve the order of tuples.
  - Does not ensure that all tuples adhere to the same schema or that they have the same number of fields.
  - Does not eliminate duplicate tuples.

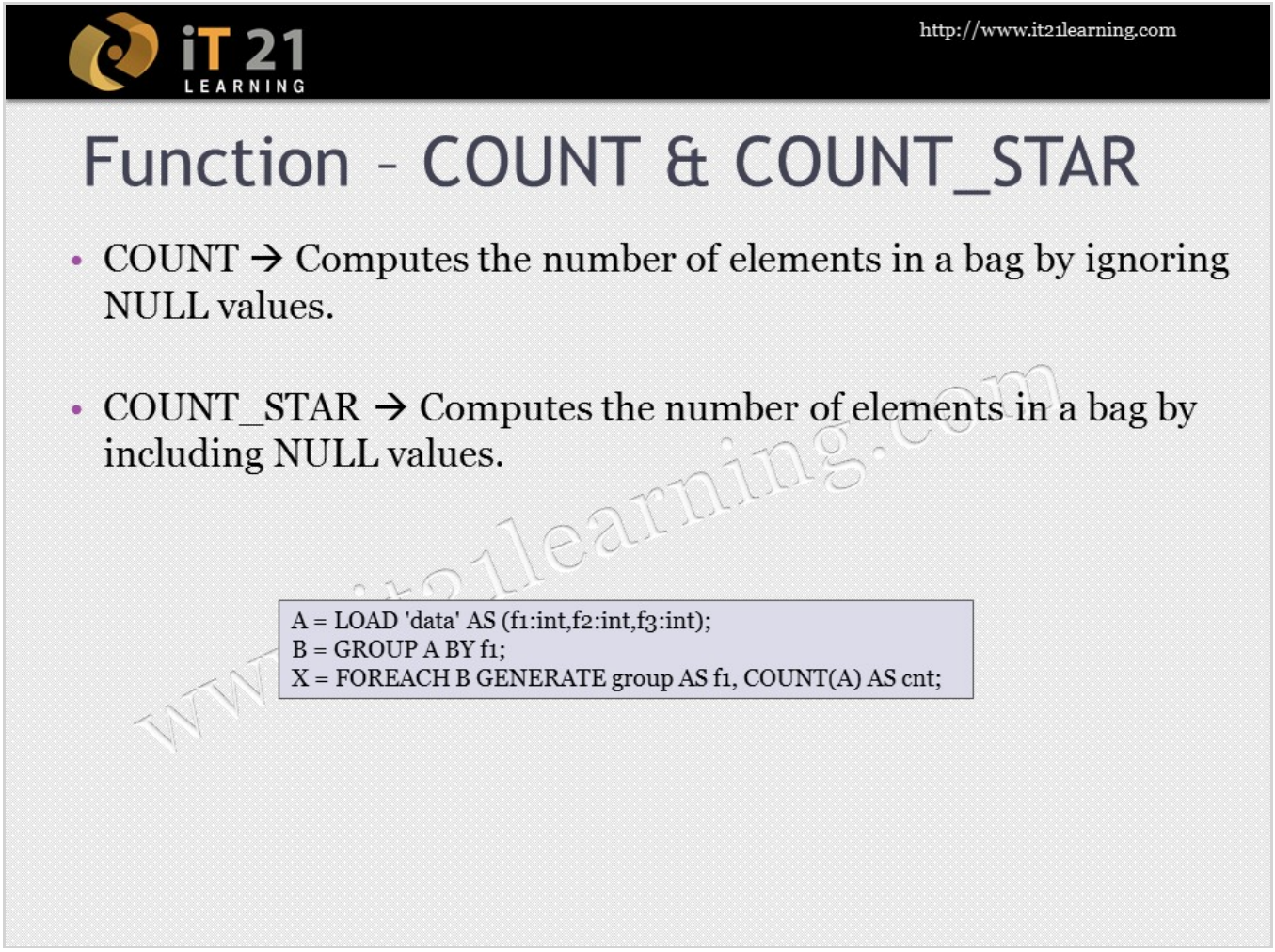

```
rs = LOAD '/user/data/twitter/full_text.txt'  using PigStorage('\t')
     AS (id:chararray, dt:chararray, loc:chararray, lat:float, lon:float, tweet:chararray);

toronto = FILTER rs BY ( lan>= 43.5847 and lon >= -79.6393)  and (lan <= 43.8554 and lon <= -79.1156);
newyork = FILTER rs BY (lan >= 40.4774 and lon >= -74.2589)  and (lan <= 40.9176 and lon <= -73.7004);

t = FOREACH toronto GENERATE id, 'Toronto' AS region;
n = FOREACH newyork GENERATE id, 'NewYork' AS region;
all = UNION t, n;
g = GROUP all BY region;
result = FOREACH g GENERATE group AS region, COUNT_STAR(all) AS cnt;
DUMP result;
```

23

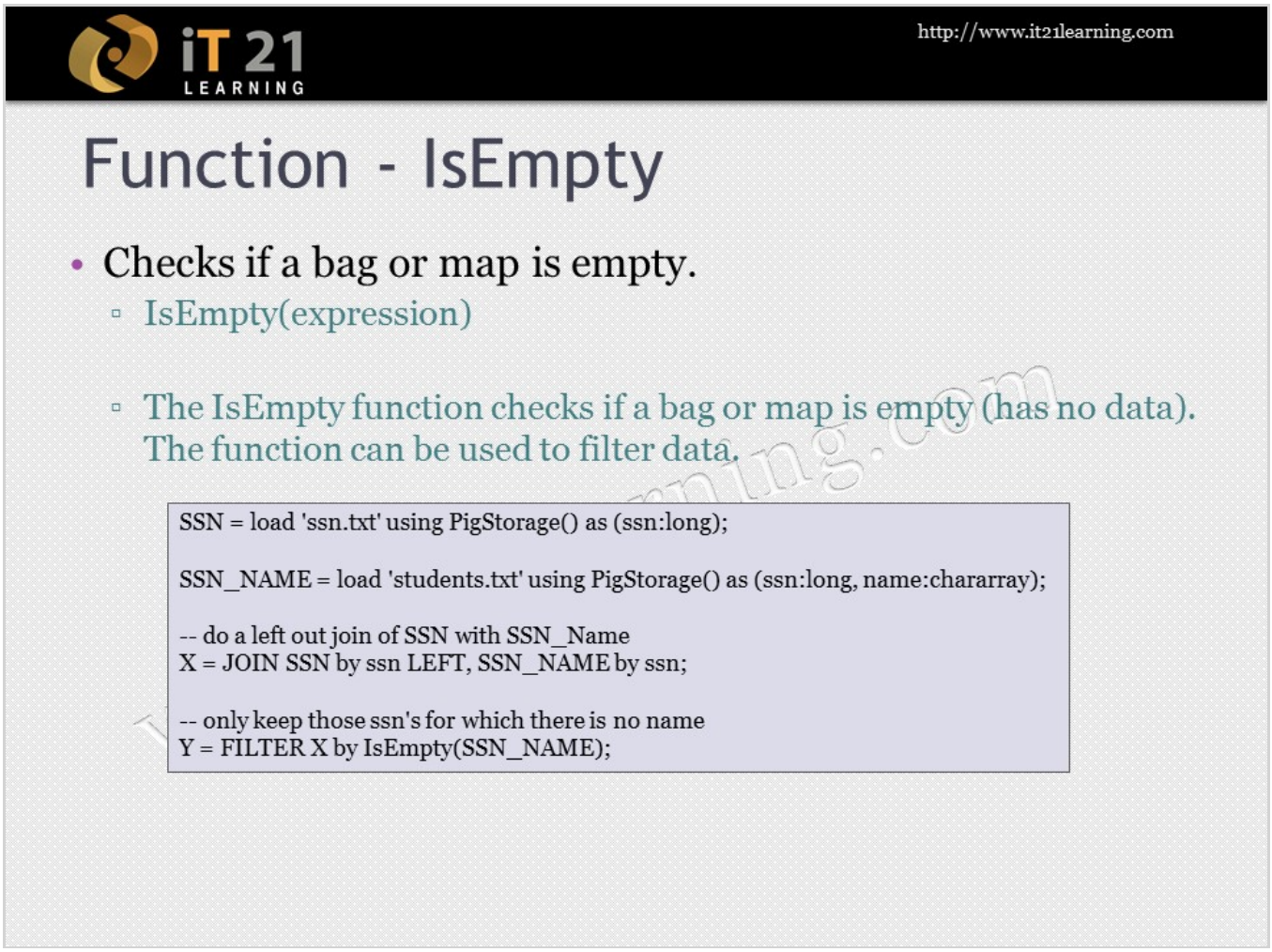

iT 21 LEARNING
http://www.it21learning.com
DISTINCT
• Removes duplicate tuples in a relation.
▫ alias = DISTINCT alias [PARALLEL n];
rs = LOAD '/user/data/twitter/full_text.txt' using PigStorage('\t')
AS (id:chararray, dt:chararray, loc:chararray, lat:float, lon:float, tweet:chararray);
ids = FOREACH rs GENERATE id;
x = DISTINCT ids;
DUMP x;

24

Function – COUNT & COUNT_STAR



25


iT 21 LEARNING
http://www.it21learning.com
Function - IsEmpty
• Checks if a bag or map is empty.
▫ IsEmpty(expression)
▫ The IsEmpty function checks if a bag or map is empty (has no data). The function can be used to filter data.
SSN = load 'ssn.txt' using PigStorage() as (ssn:long);
SSN_NAME = load 'students.txt' using PigStorage() as (ssn:long, name:chararray);
-- do a left out join of SSN with SSN_Name
X = JOIN SSN by ssn LEFT, SSN_NAME by ssn;
-- only keep those ssn's for which there is no name
Y = FILTER X by IsEmpty(SSN_NAME);

26

Function - TOKENIZE

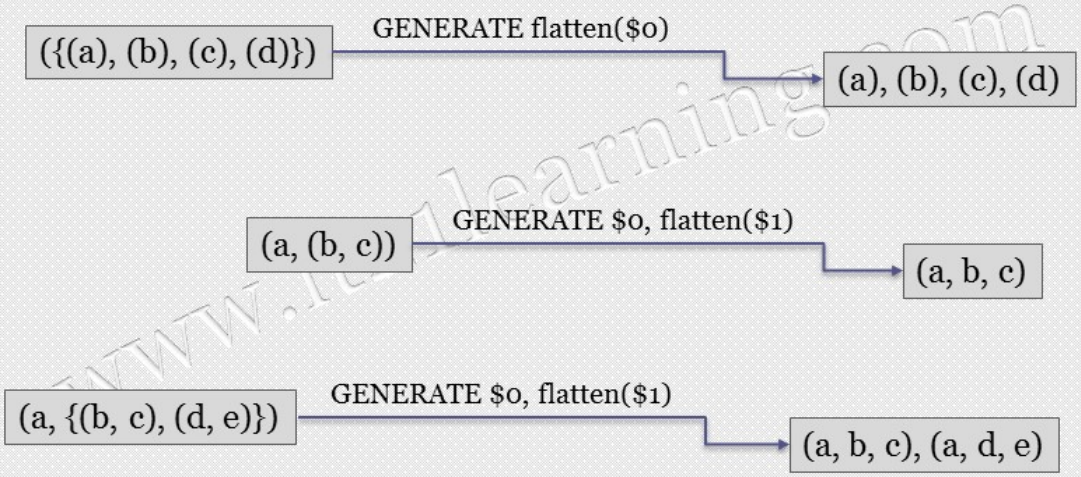
- Splits a string and outputs a bag of words.
  - TOKENIZE(expression)
  - Use the TOKENIZE function to split a string of words (all words in a single tuple) into a bag of words (each word in a single tuple).

```
rs = LOAD '/user/data/twitter/full_text.txt' USING PigStorage('\t')
     AS (id:chararray, dt:chararray, loc:chararray, lat:float, lon:float, tweet:chararray);

tws = FOREACH rs GENERATE tweet;
tw = LIMIT tws 1;

tokens = FOREACH tw GENERATE TOKENIZE($0);
DUMP tokens;
```

27

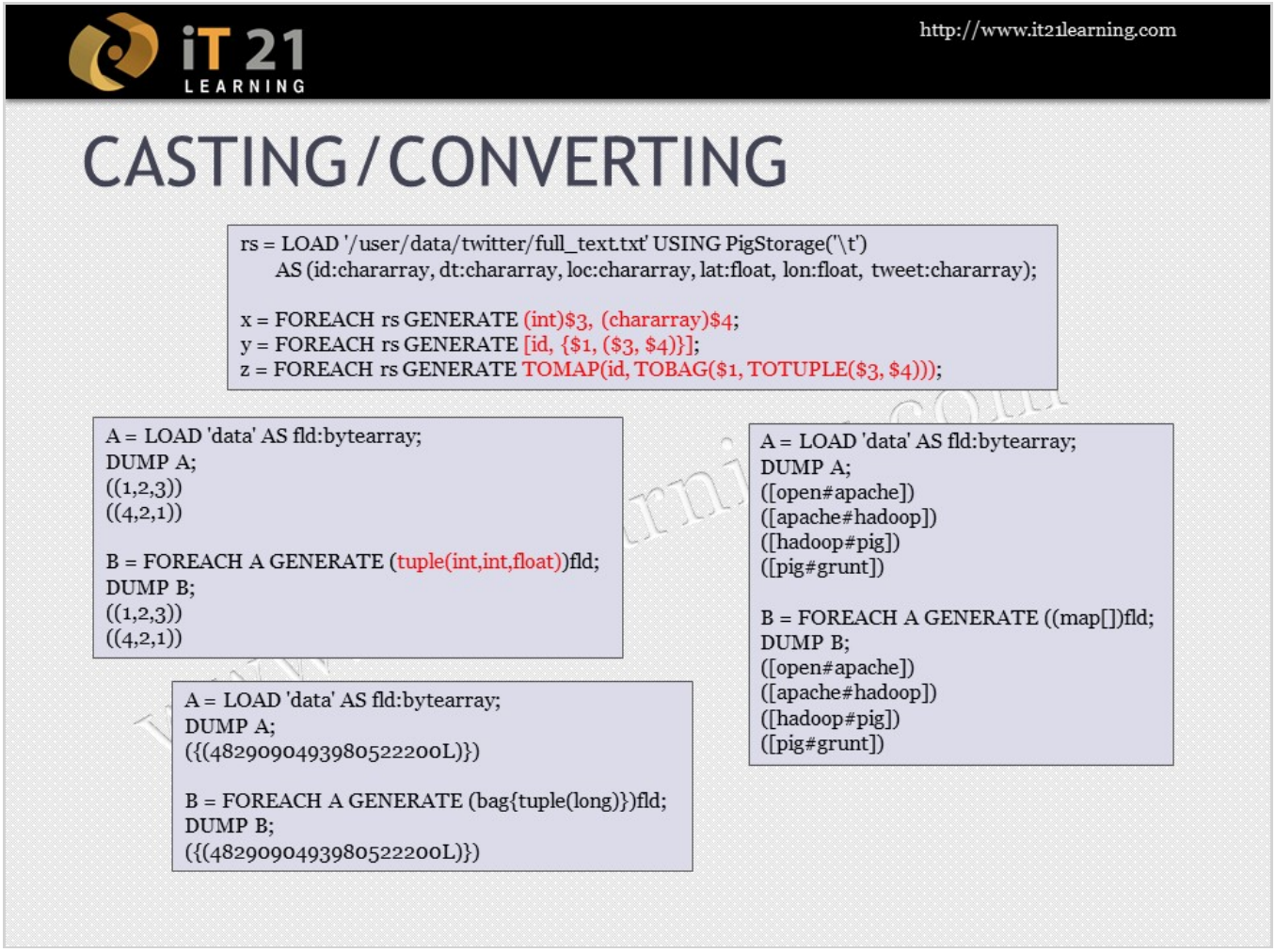
Flatten Operator

- Flatten un-nests tuples as well as bags, so it changes the structure of tuples and bags.

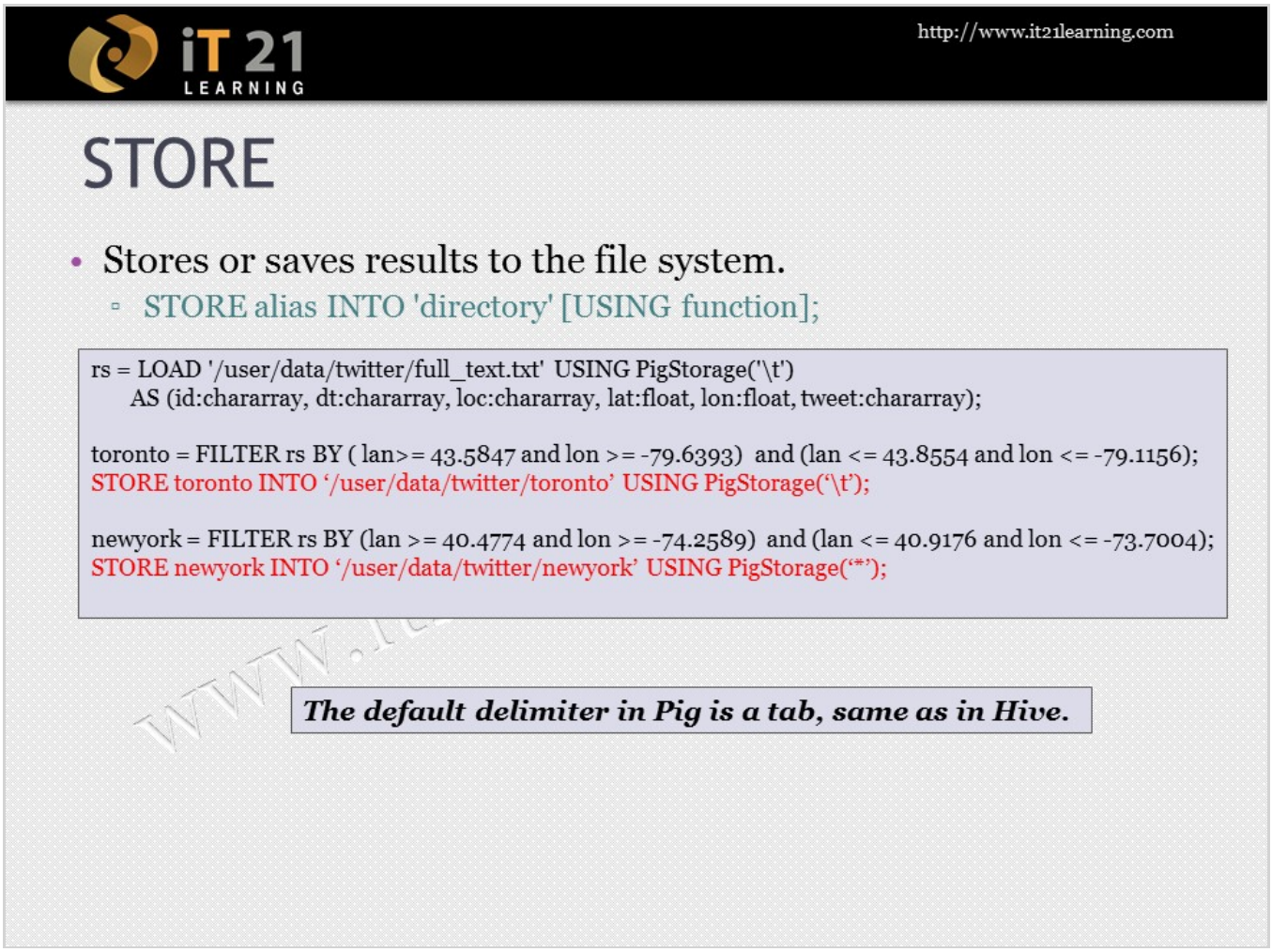


28

CASTING/CONVERTING

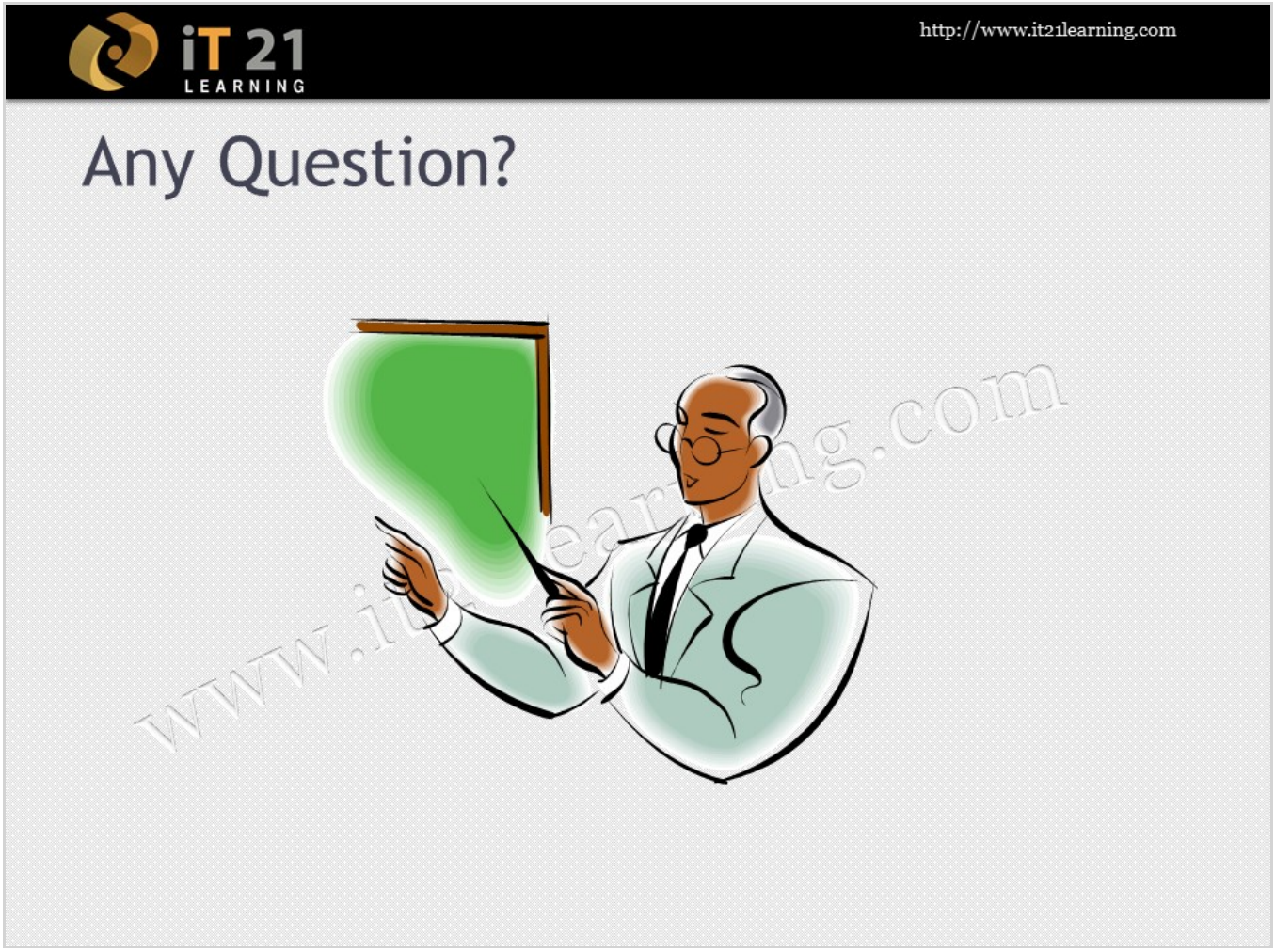


29


iT 21 LEARNING
http://www.it21learning.com
STORE
• Stores or saves results to the file system.
▫ STORE alias INTO 'directory' [USING function];
rs = LOAD '/user/data/twitter/full_text.txt' USING PigStorage('\t')
AS (id:chararray, dt:chararray, loc:chararray, lat:float, lon:float, tweet:chararray);
toronto = FILTER rs BY (lan>= 43.5847 and lon >= -79.6393) and (lan <= 43.8554 and lon <= -79.1156);
STORE toronto INTO '/user/data/twitter/toronto' USING PigStorage('\t');
newyork = FILTER rs BY (lan >= 40.4774 and lon >= -74.2589) and (lan <= 40.9176 and lon <= -73.7004);
STORE newyork INTO '/user/data/twitter/newyork' USING PigStorage('*');
The default delimiter in Pig is a tab, same as in Hive.

30

Any Question?



31

Introducing HBase

Apache HBase Overview

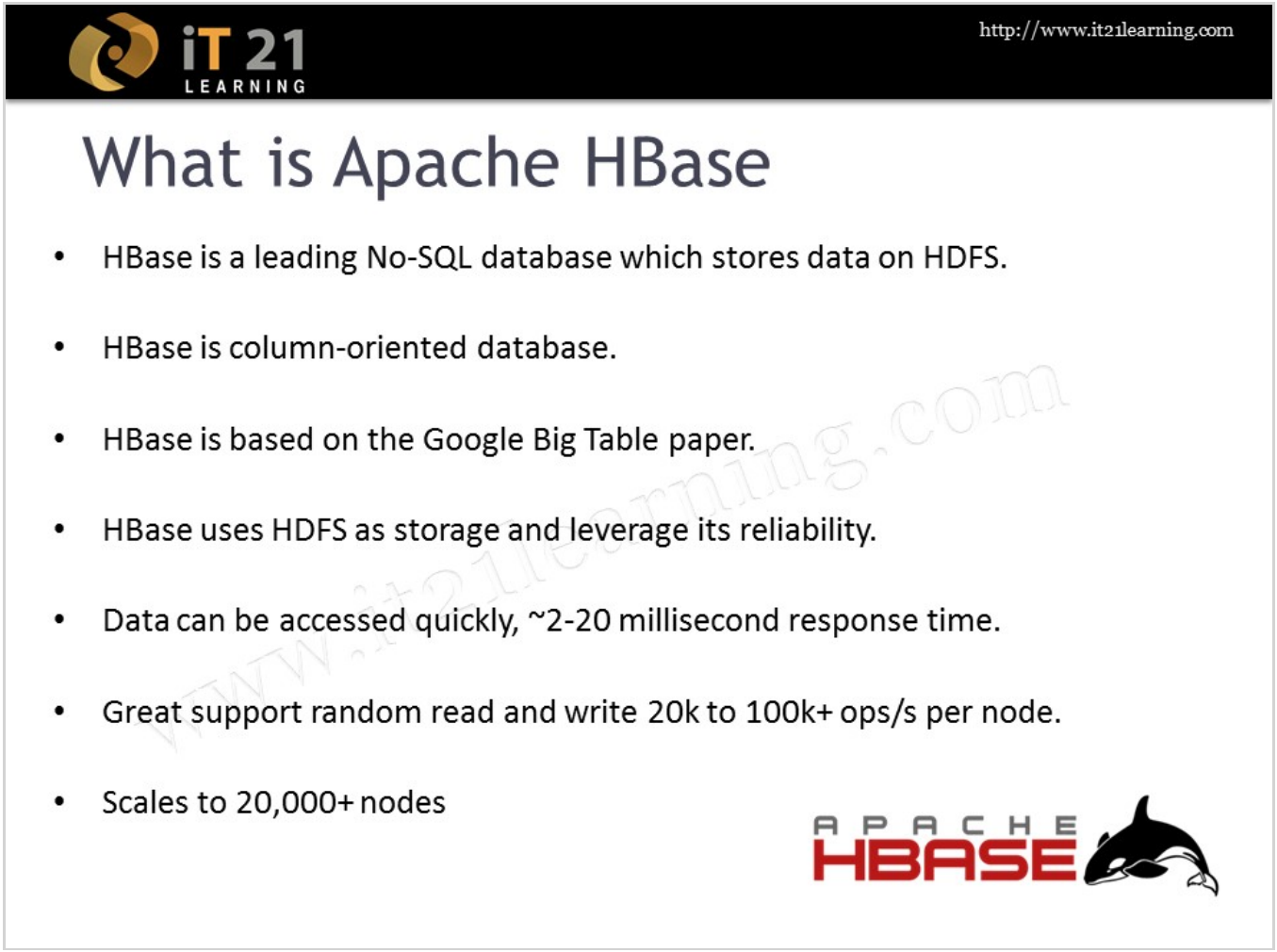
1

Outline

- What is Apache HBase
- HBase History, Use Cases, and User Group
- HBase Architecture
- HBase Shell and Demo
- HBase API
- HBase SQL Access - Phoenix Demo
- HBase Advanced Topics
- Q&A

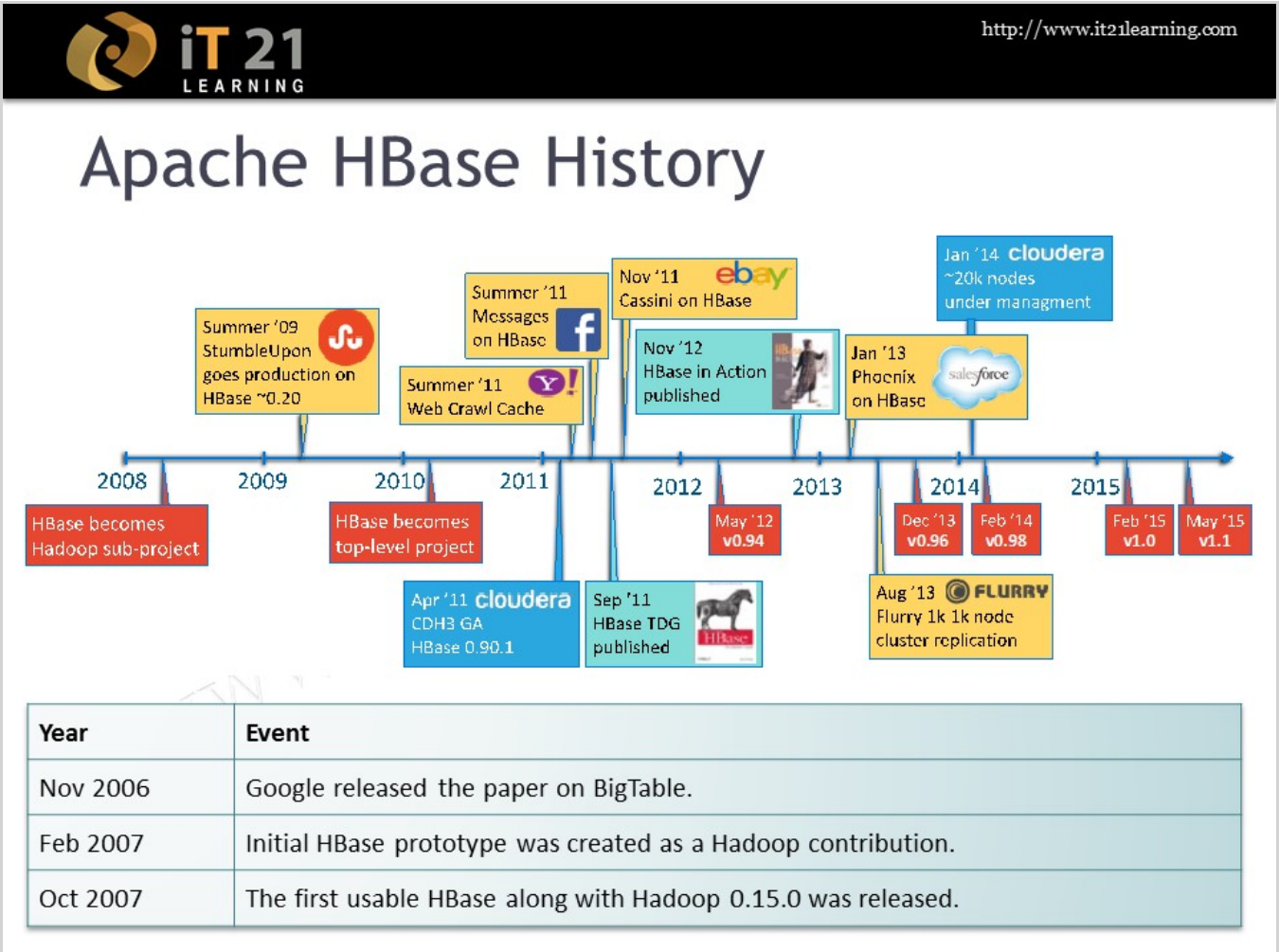
2

What is Apache HBase



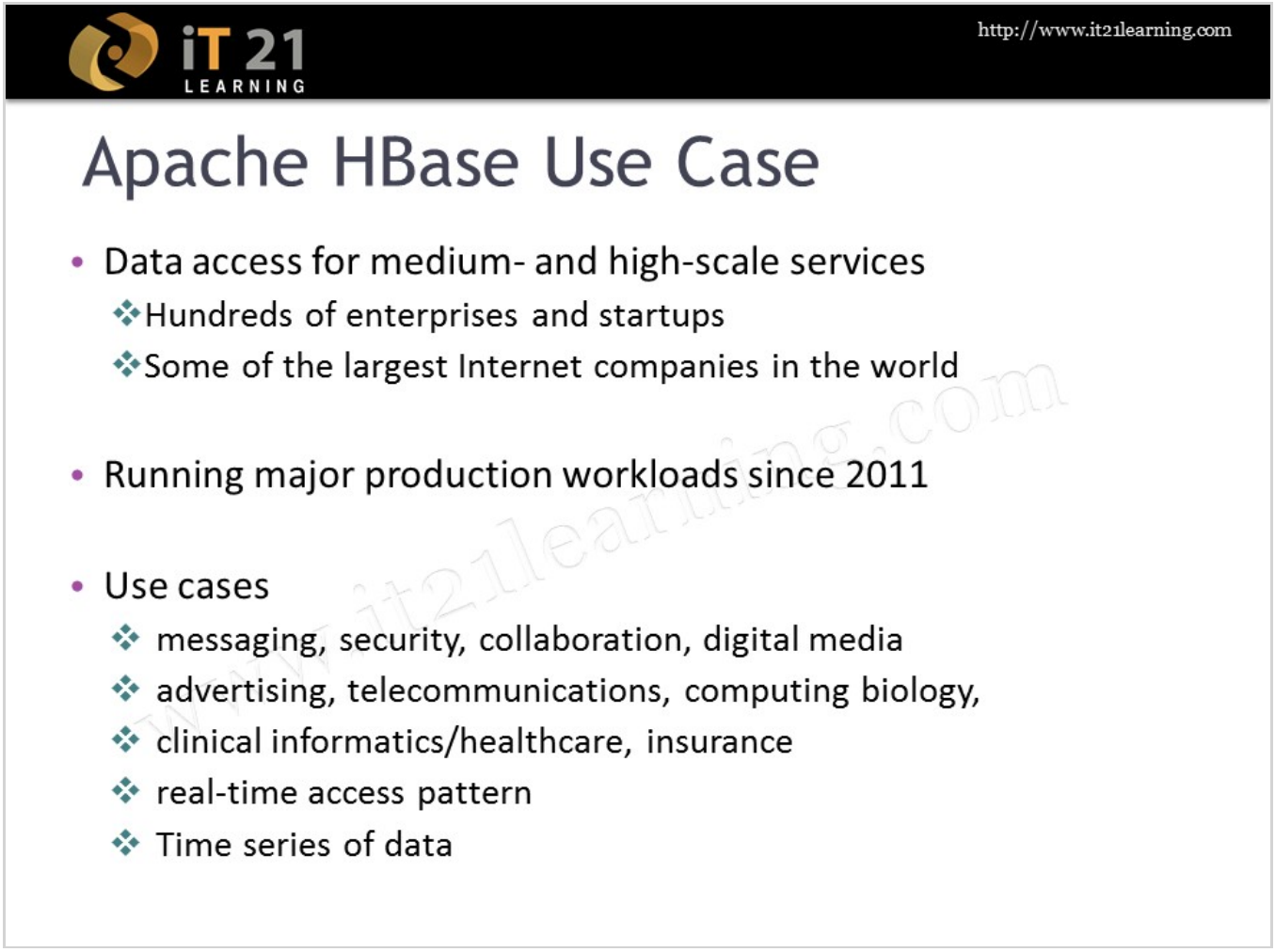
3

Apache HBase History



Year	Event
Nov 2006	Google released the paper on BigTable.
Feb 2007	Initial HBase prototype was created as a Hadoop contribution.
Oct 2007	The first usable HBase along with Hadoop 0.15.0 was released.

eBay presented a keynote at Hadoop World, describing the architecture of its completely rebuilt search engine, Cassini, slated to go live in 2012. It indexes all the content and user metadata to produce better rankings and refreshes indexes hourly. It is built using Apache Hadoop for hourly index updates and Apache HBase to provide random access to item information.

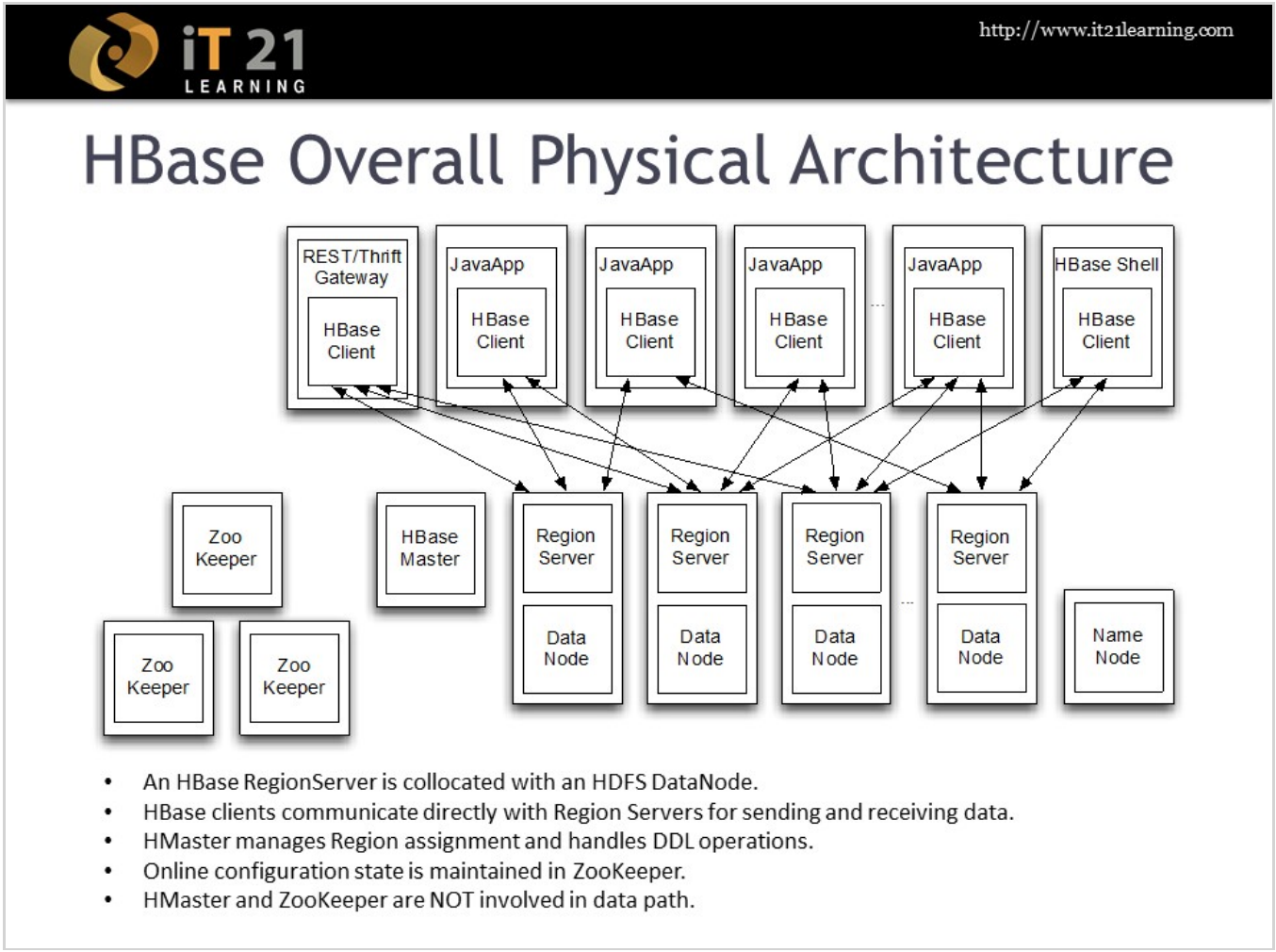

iT 21 LEARNING
http://www.it21learning.com
Apache HBase Use Case
• Data access for medium- and high-scale services
❖ Hundreds of enterprises and startups
❖ Some of the largest Internet companies in the world
• Running major production workloads since 2011
• Use cases
❖ messaging, security, collaboration, digital media
❖ advertising, telecommunications, computing biology,
❖ clinical informatics/healthcare, insurance
❖ real-time access pattern
❖ Time series of data

5

Apache HBase User Groups

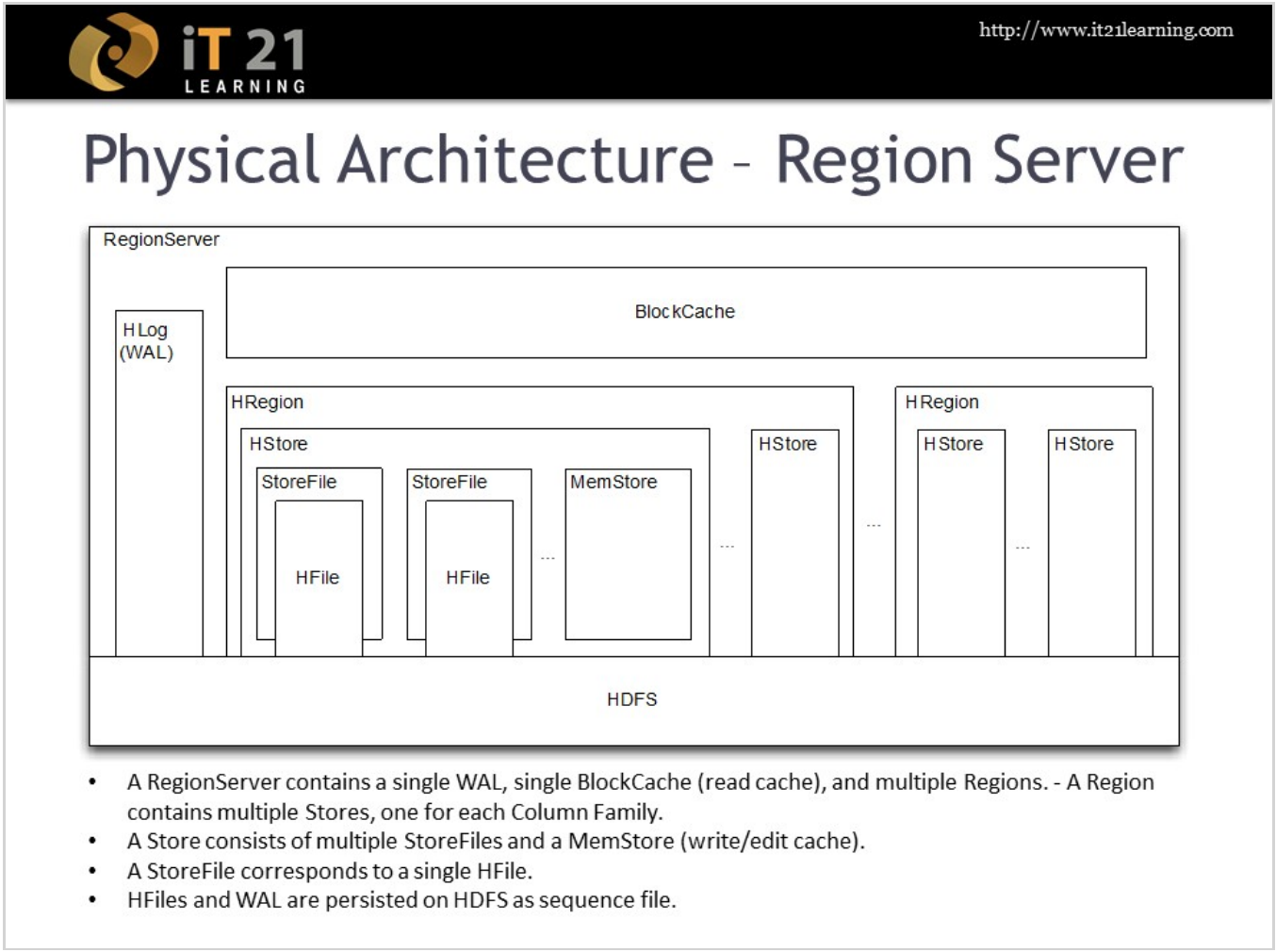
6

HBase Overall Physical Architecture



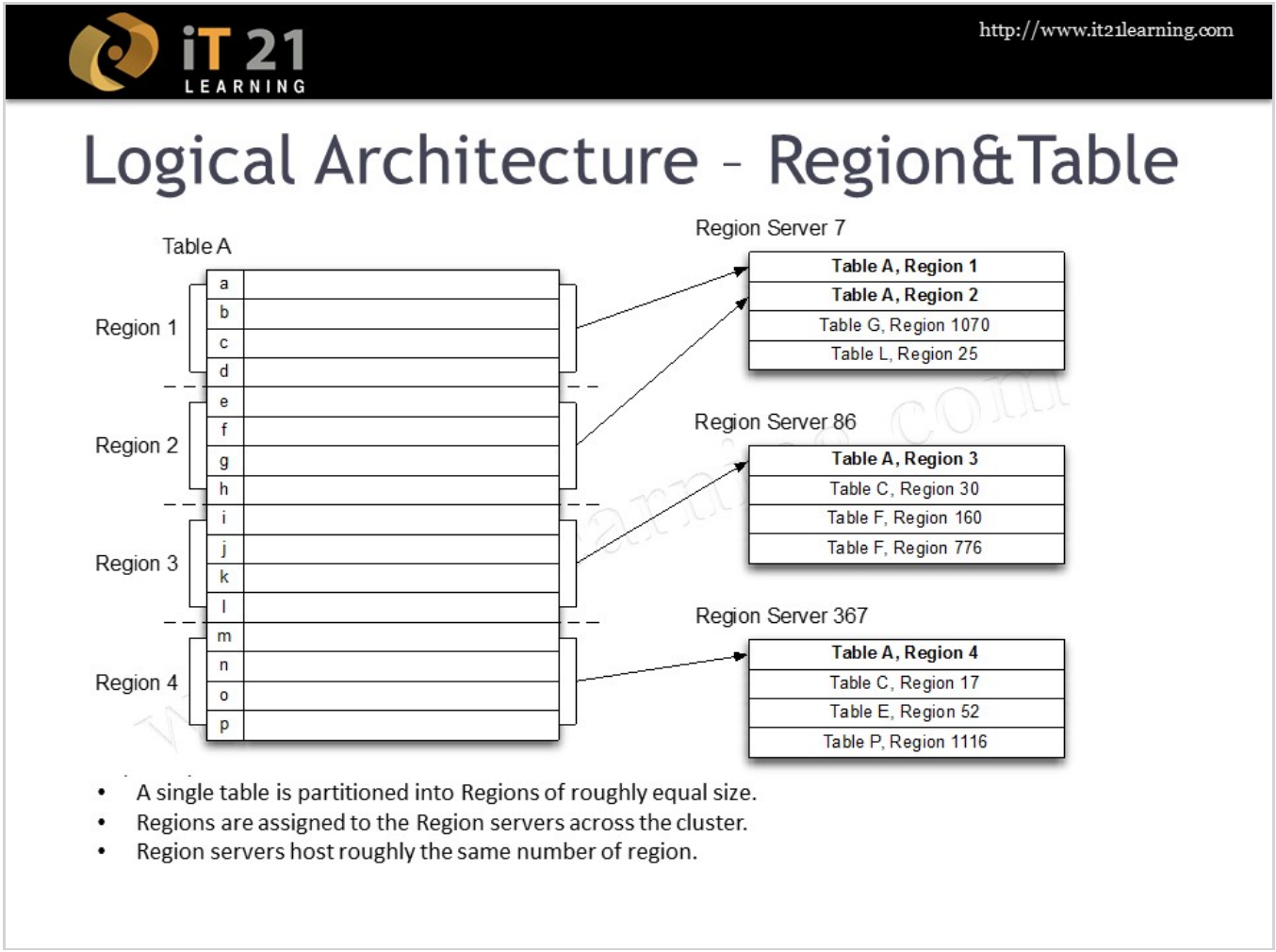
7

Physical Architecture – Region Server



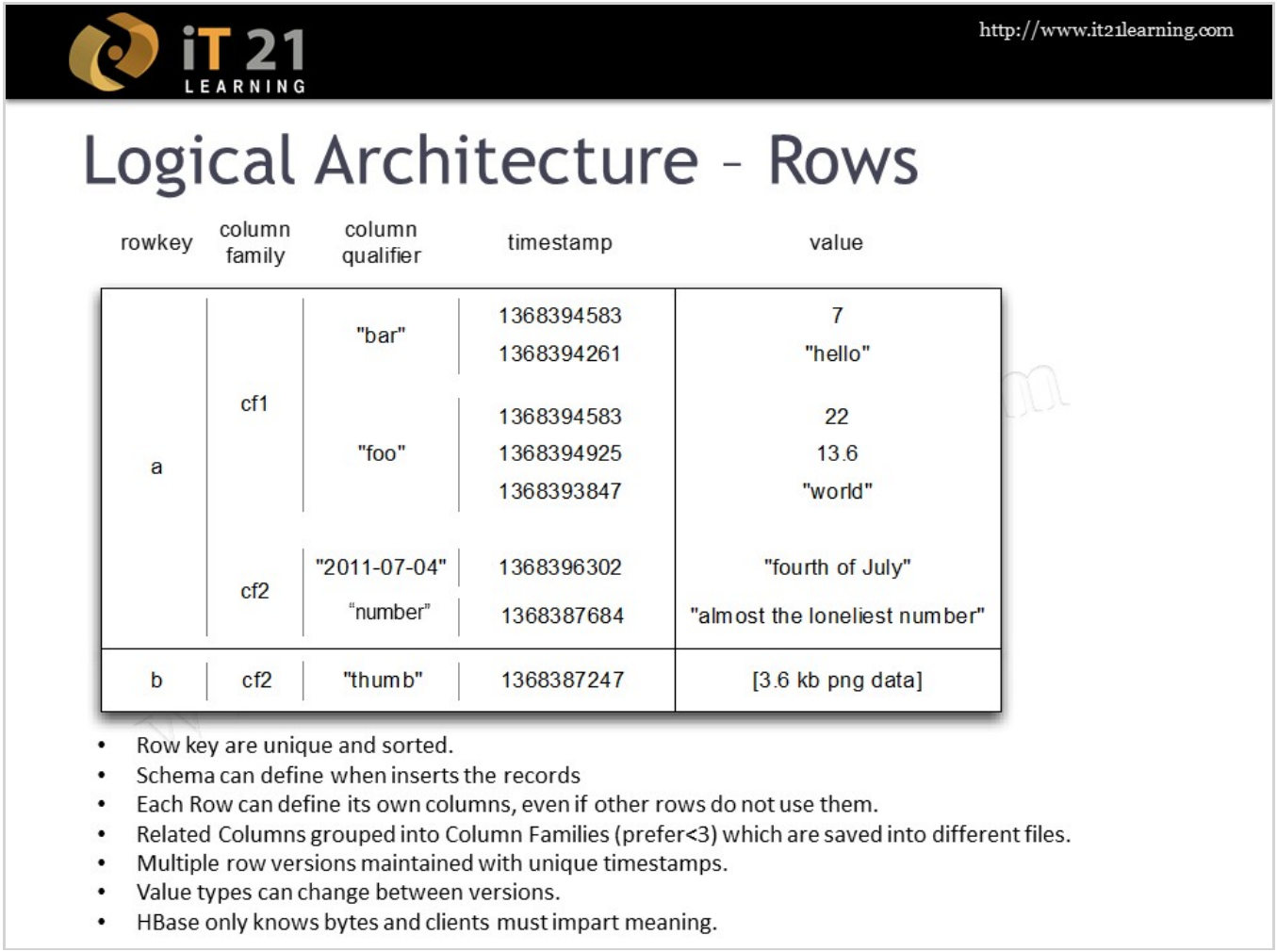
When data is added to HBase, it is first written to the WAL (Write ahead log) called Hlog – A sequence file. Once the write is done, it is then written to an in memory called MemStore Once the memory exceeds a certain threshold, it flushes to disk as an HFile Over time HBase merges smaller HFiles into larger ones. This process is called compaction WAL is used to recover.

Logical Architecture – Region&Table

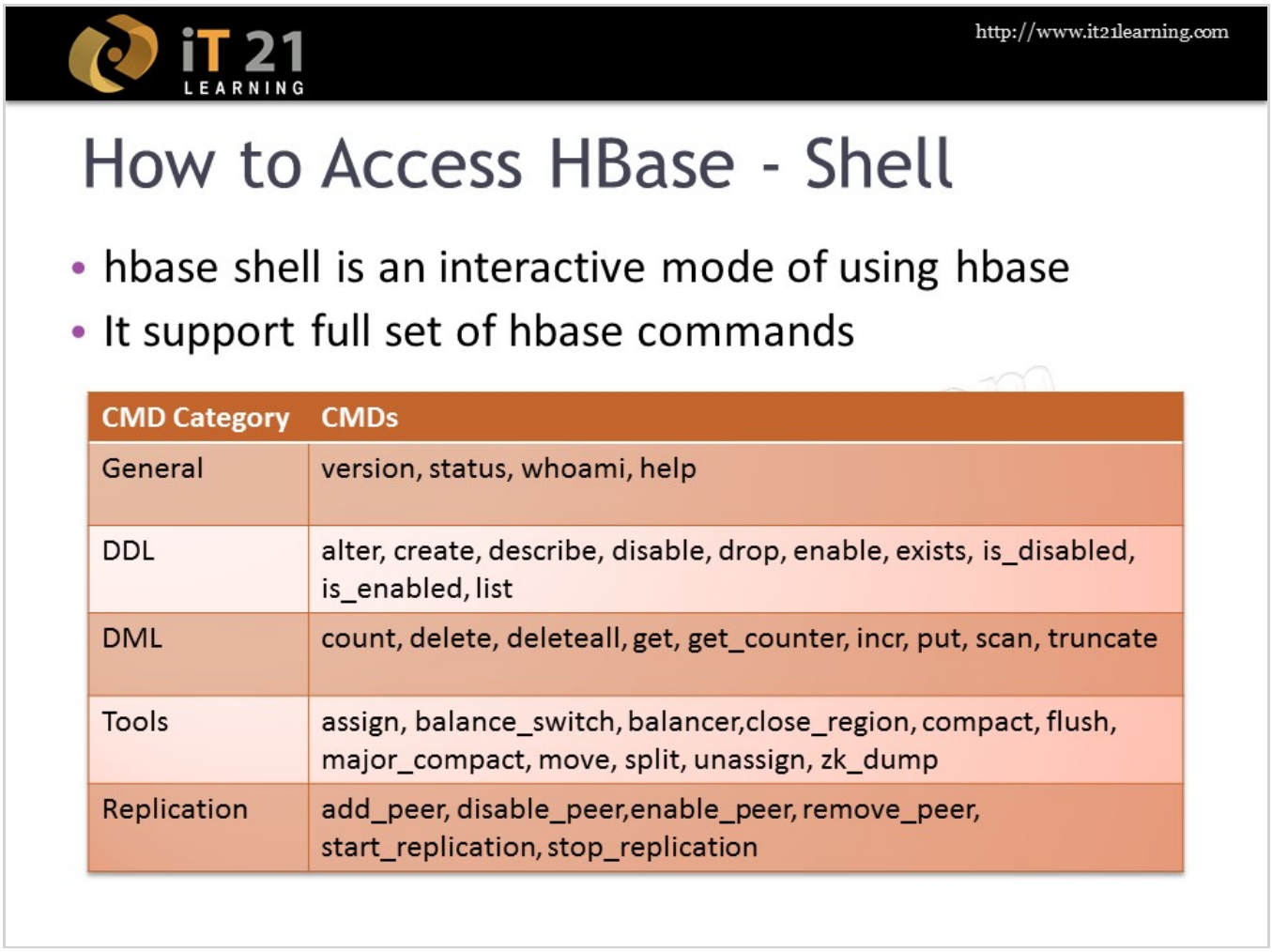


9

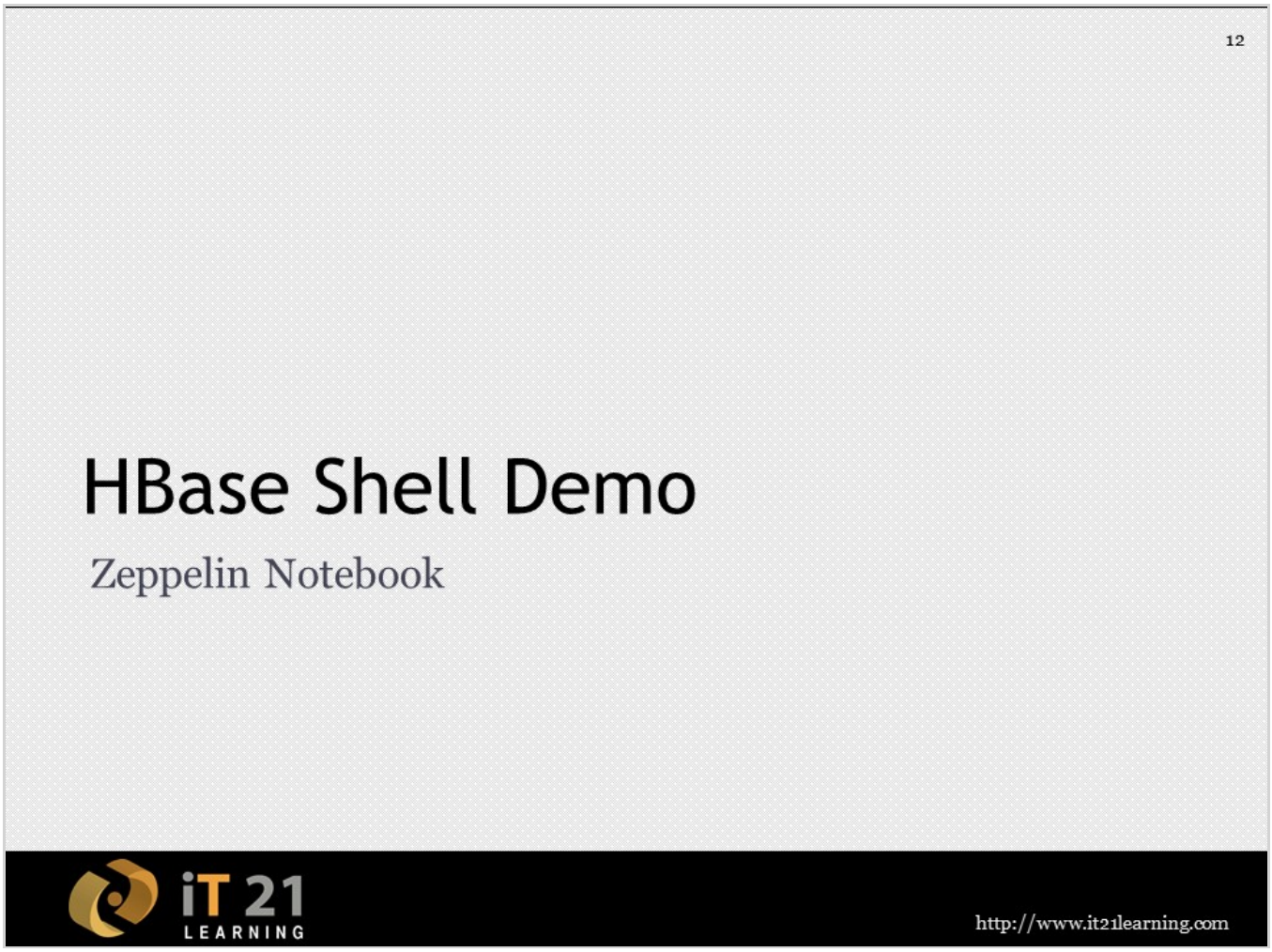
Logical Architecture – Rows



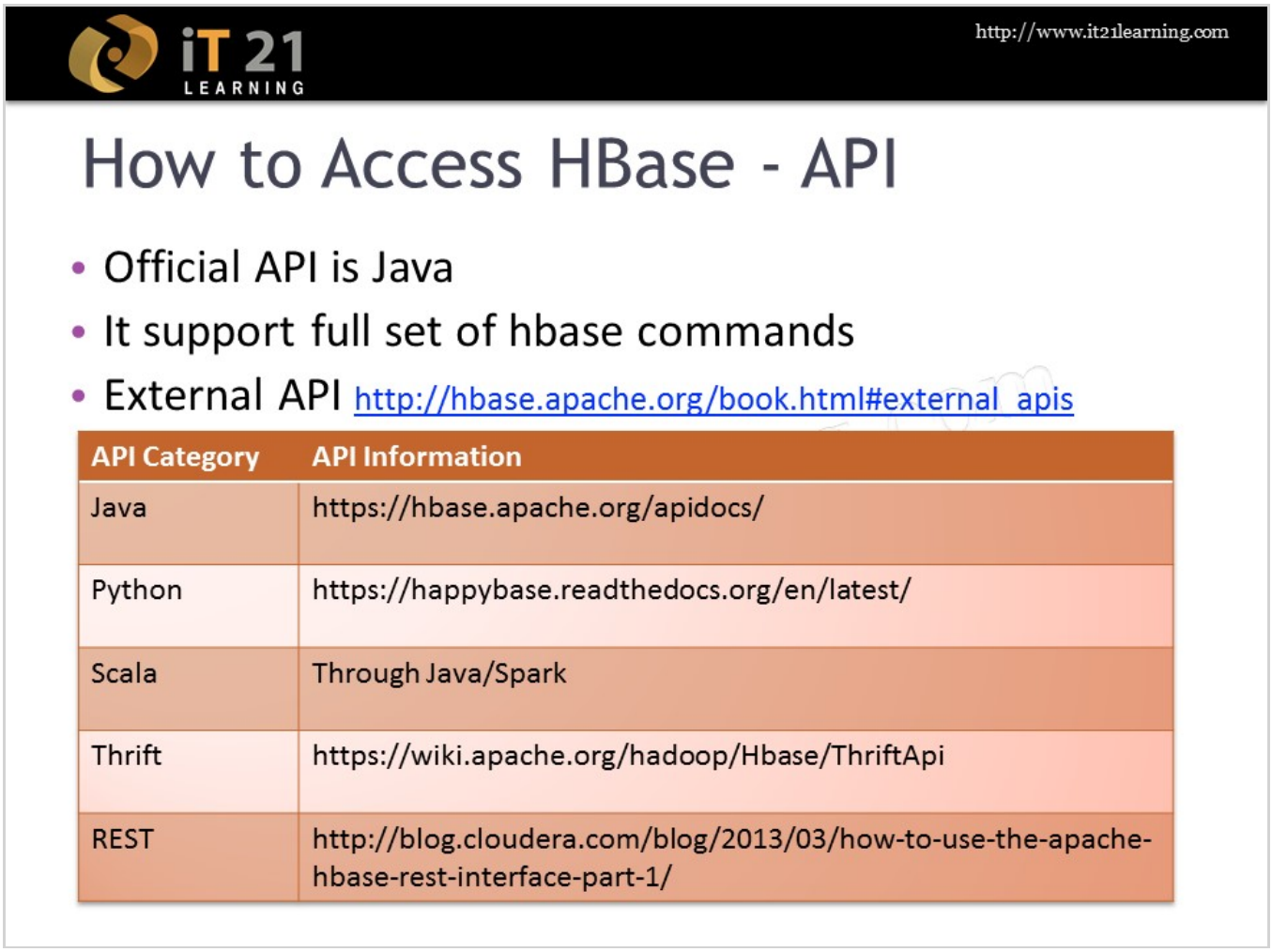
Data is stored... in a big table • Sorted map datastore Tables consist of sorted rows, each of which has a primary row key Each row has a set of columns A column is specified as a column family and column qualifier pair A given cell (row, column family:column qualifier) can have different time- stamped values


iT 21 LEARNING
http://www.it21learning.com
How to Access HBase - Shell
• hbase shell is an interactive mode of using hbase
• It support full set of hbase commands
CMD Category
CMDs
General
version, status, whoami, help
DDL
alter, create, describe, disable, drop, enable, exists, is_disabled, is_enabled, list
DML
count, delete, deleteall, get, get_counter, incr, put, scan, truncate
Tools
assign, balance_switch, balancer,close_region, compact, flush, major_compact, move, split, unassign, zk_dump
Replication
add_peer, disable_peer,enable_peer, remove_peer, start_replication, stop_replication

11

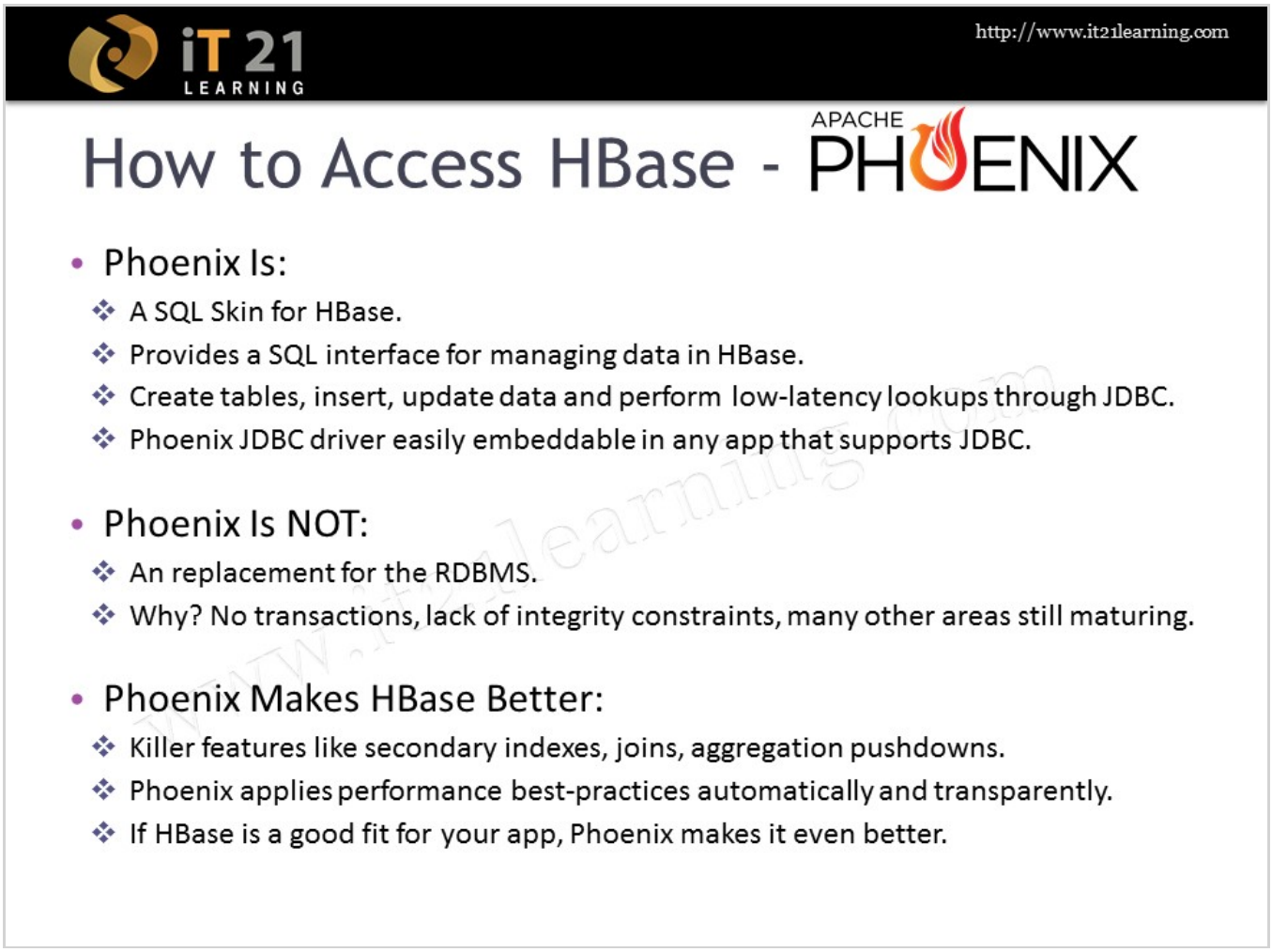

12
HBase Shell Demo
Zeppelin Notebook
iT 21
LEARNING
http://www.it21learning.com

12


iT 21 LEARNING
http://www.it21learning.com
How to Access HBase - API
• Official API is Java
• It support full set of hbase commands
• External API http://hbase.apache.org/book.html#external_apis
API Category
API Information
Java
https://hbase.apache.org/apidocs/
Python
https://happybase.readthedocs.org/en/latest/
Scala
Through Java/Spark
Thrift
https://wiki.apache.org/hadoop/Hbase/ThriftApi
REST
http://blog.cloudera.com/blog/2013/03/how-to-use-the-apache-hbase-rest-interface-part-1/

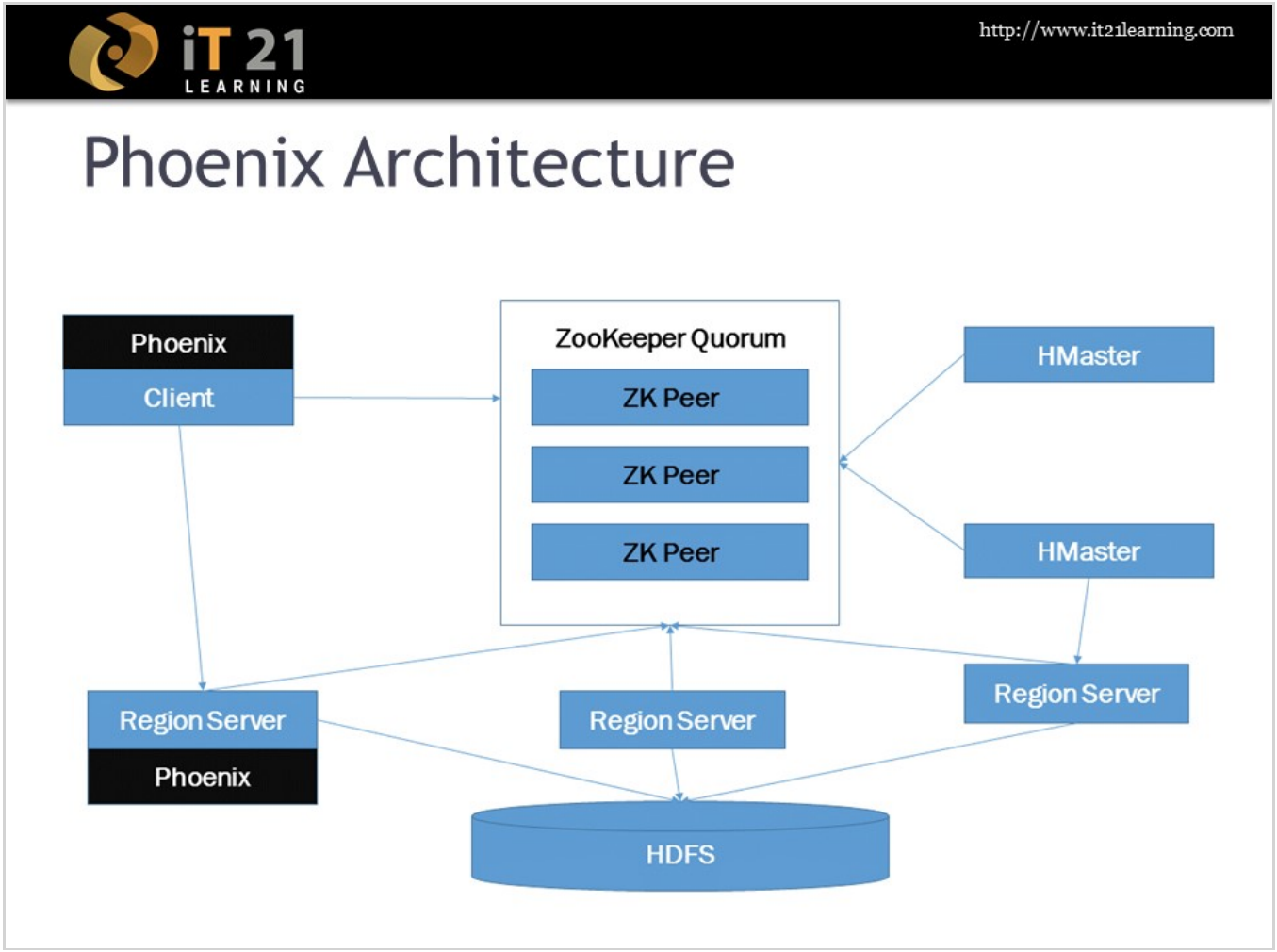
13

How to Access HBase -



14

Phoenix Architecture



15

Phoenix Provides SQL-like Syntax

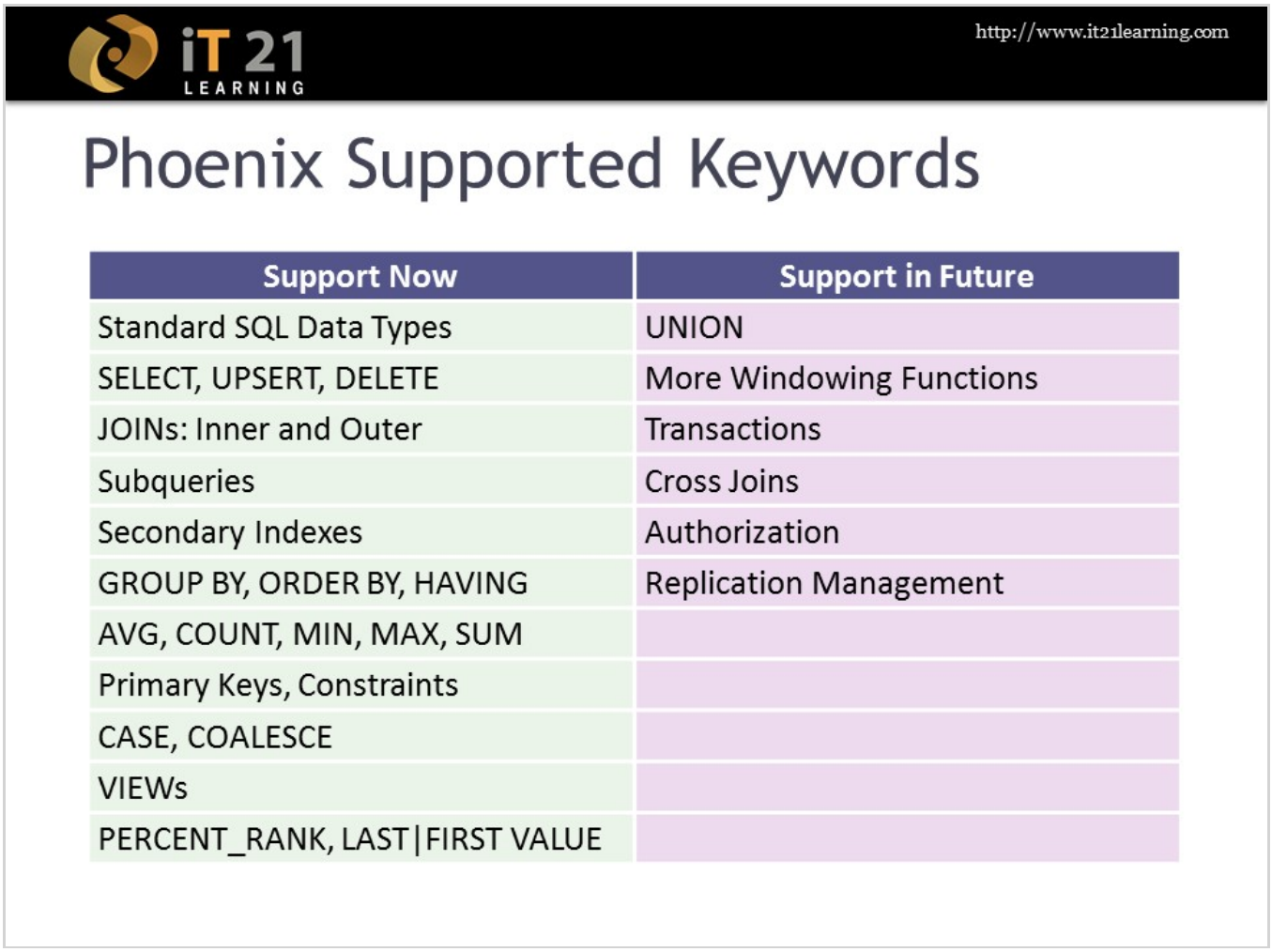
API Code	Phoenix DDL

HBaseAdmin hbase = new HBaseAdmin(conf);
HTableDescriptor desc = new HTableDescriptor("us_population");
HColumnDescriptor state = new HColumnDescriptor("state".getBytes());
HColumnDescriptor city = new HColumnDescriptor("city".getBytes());
HColumnDescriptor population = new HColumnDescriptor("population".getBytes());
desc.addFamily(state);
desc.addFamily(city);
desc.addFamily(population);
hbase.createTable(desc);
	CREATE TABLE us_population (
	state CHAR(2) NOT NULL,
	city VARCHAR NOT NULL,
	population BIGINT
	CONSTRAINT my_pk PRIMARY KEY (state, city)
	);

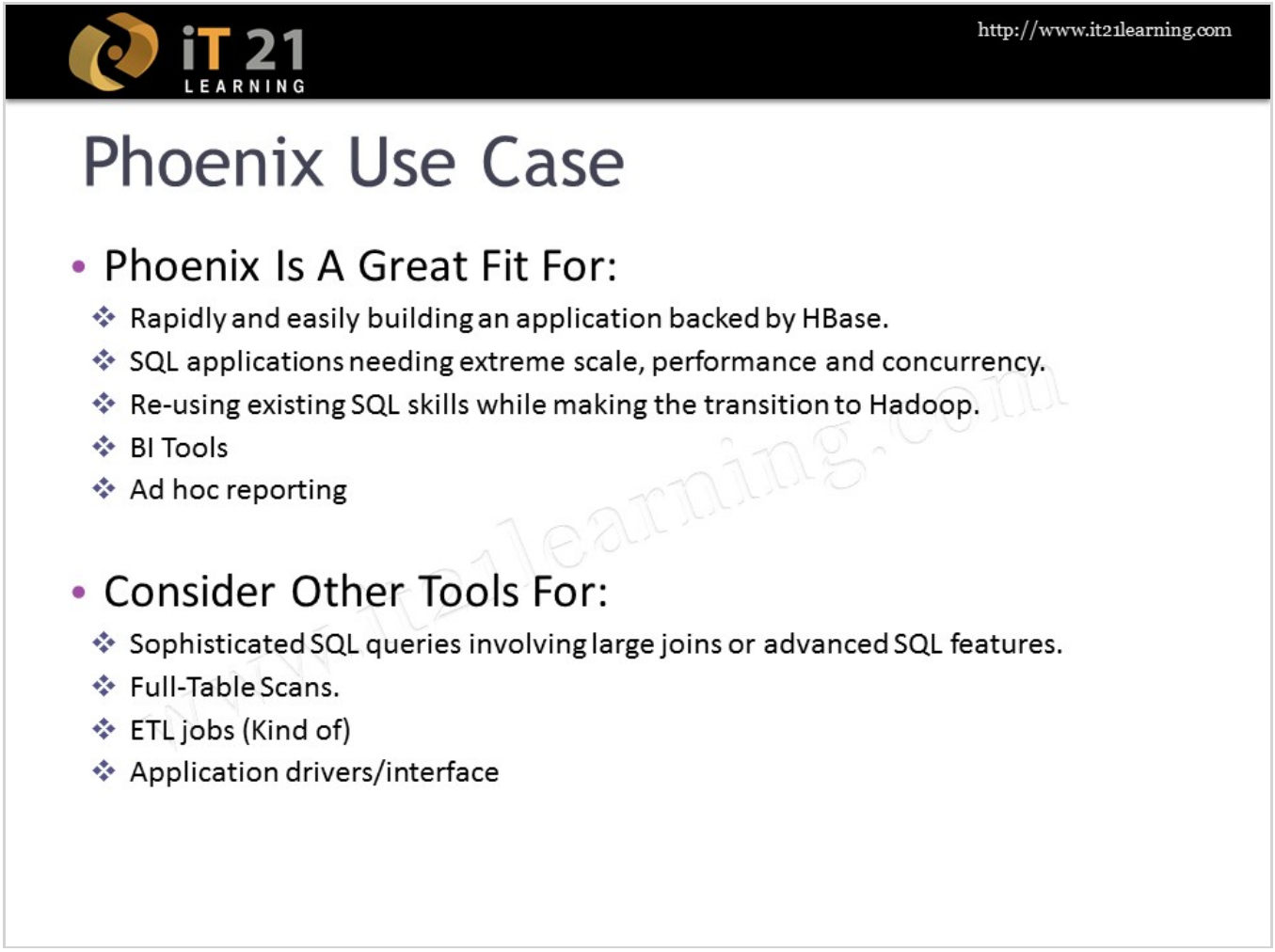
• More flexible • Integration	• Familiar SQL syntax. • Provides additional constraint checking.

16

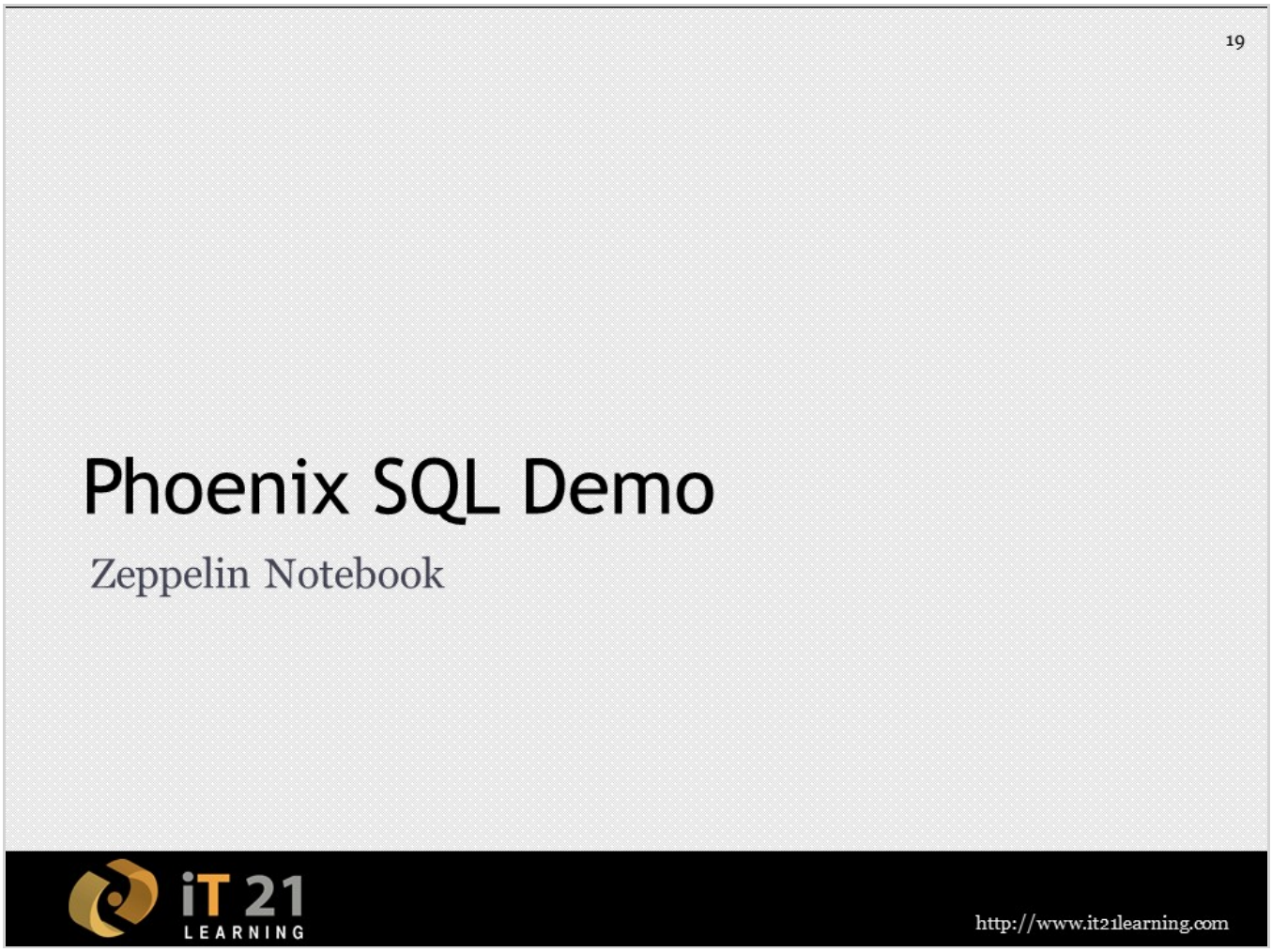
Phoenix Supported Keywords



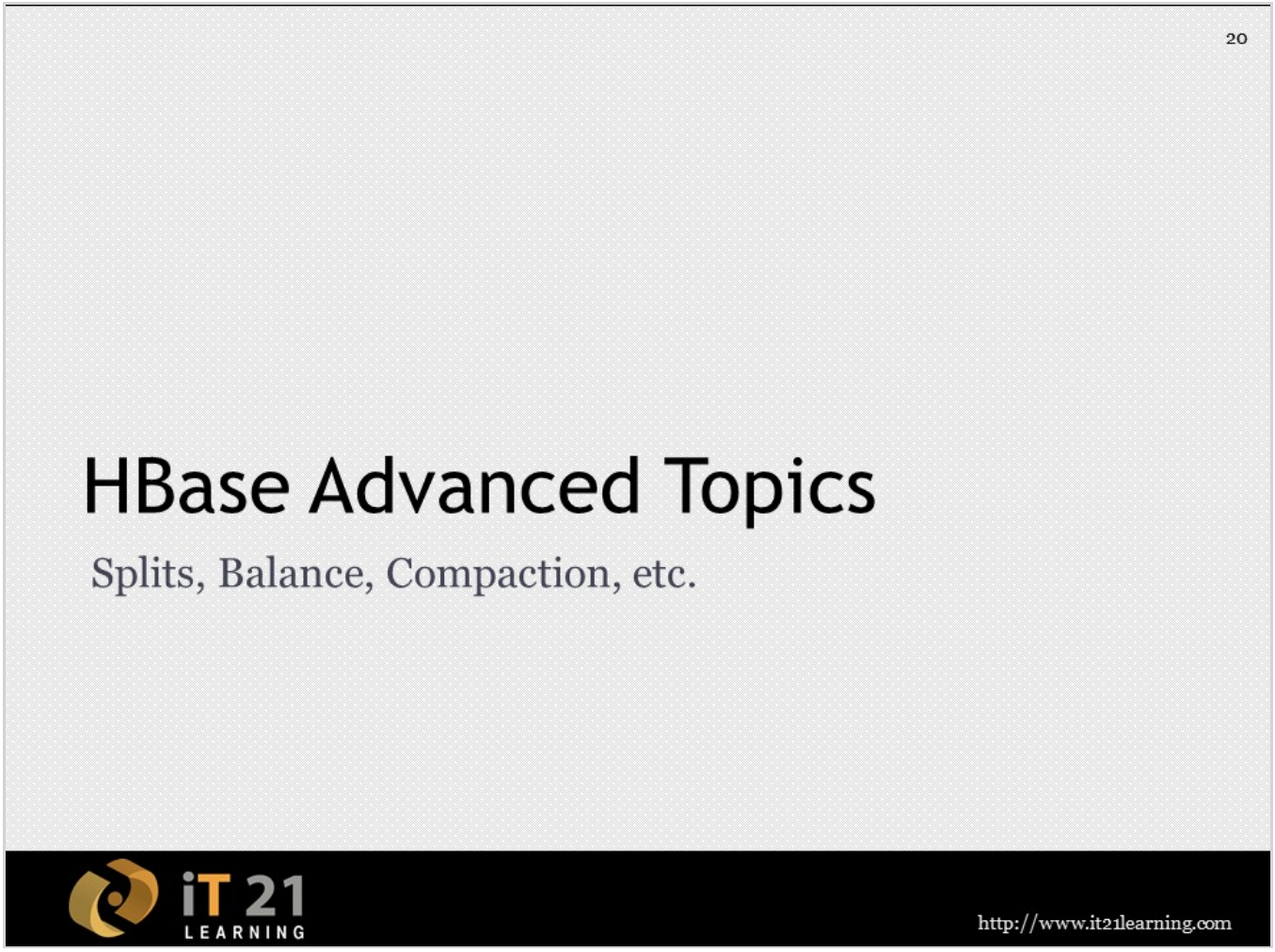
17


iT 21 LEARNING
http://www.it21learning.com
Phoenix Use Case
• Phoenix Is A Great Fit For:
❖ Rapidly and easily building an application backed by HBase.
❖ SQL applications needing extreme scale, performance and concurrency.
❖ Re-using existing SQL skills while making the transition to Hadoop.
❖ BI Tools
❖ Ad hoc reporting
• Consider Other Tools For:
❖ Sophisticated SQL queries involving large joins or advanced SQL features.
❖ Full-Table Scans.
❖ ETL jobs (Kind of)
❖ Application drivers/interface

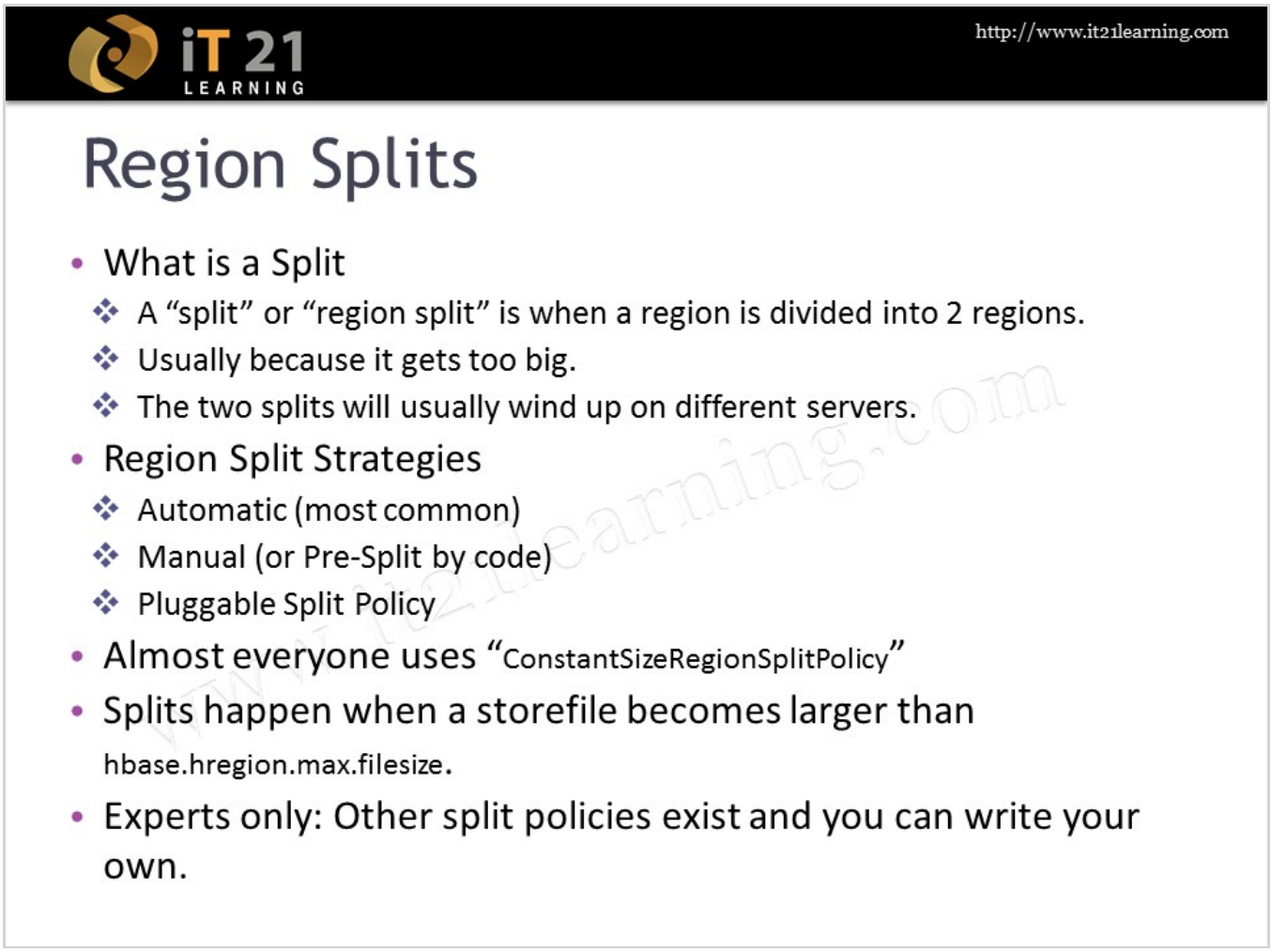
18


19
Phoenix SQL Demo
Zeppelin Notebook
iT 21
LEARNING
http://www.it21learning.com

19

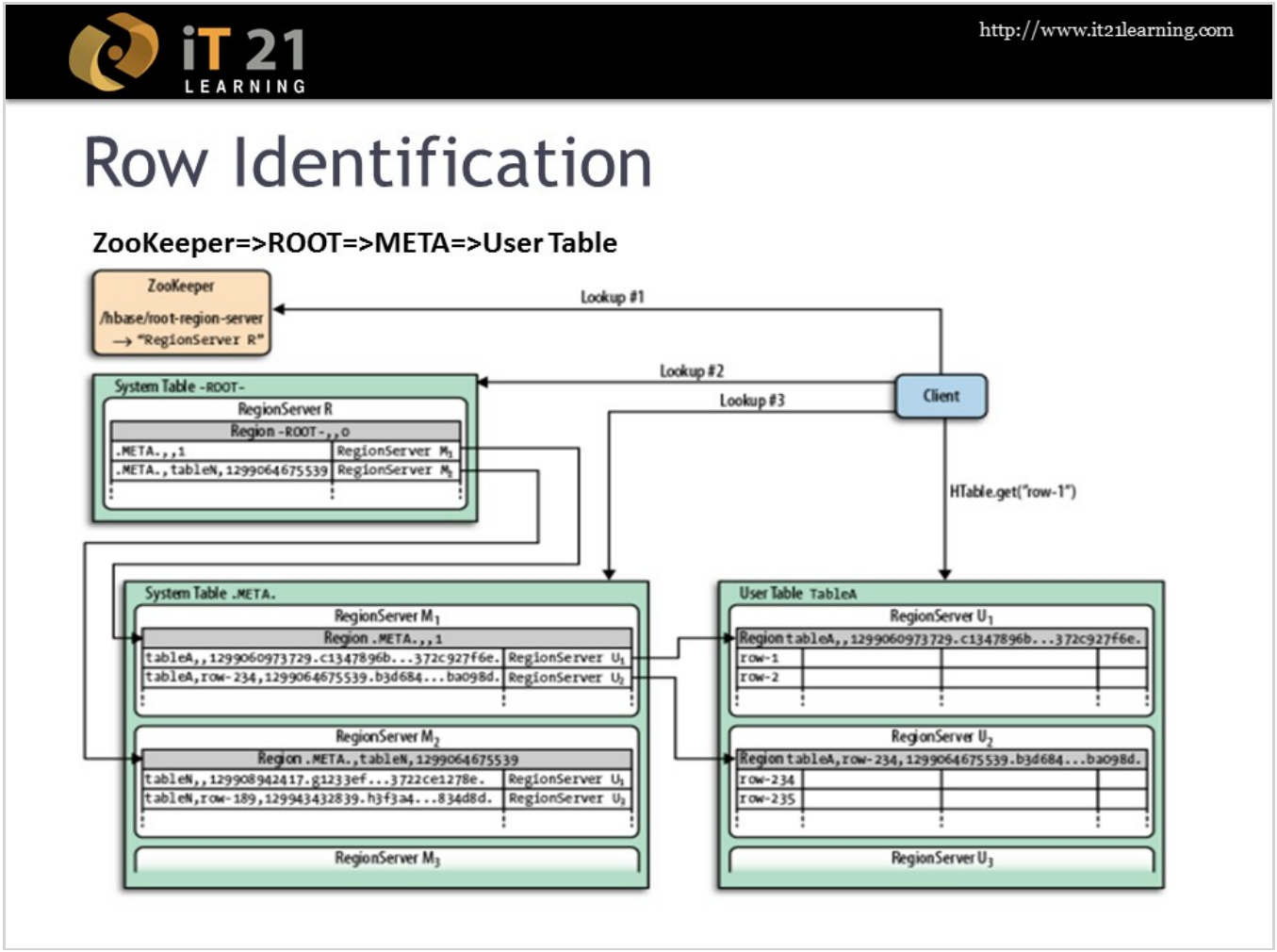

20
HBase Advanced Topics
Splits, Balance, Compaction, etc.
iT 21
LEARNING
http://www.it21learning.com

20


iT 21 LEARNING
http://www.it21learning.com
Region Splits
• What is a Split
❖ A “split” or “region split” is when a region is divided into 2 regions.
❖ Usually because it gets too big.
❖ The two splits will usually wind up on different servers.
• Region Split Strategies
❖ Automatic (most common)
❖ Manual (or Pre-Split by code)
❖ Pluggable Split Policy
• Almost everyone uses “ConstantSizeRegionSplitPolicy”
• Splits happen when a storefile becomes larger than hbase.hregion.max.filesize.
• Experts only: Other split policies exist and you can write your own.

21

Row Identification

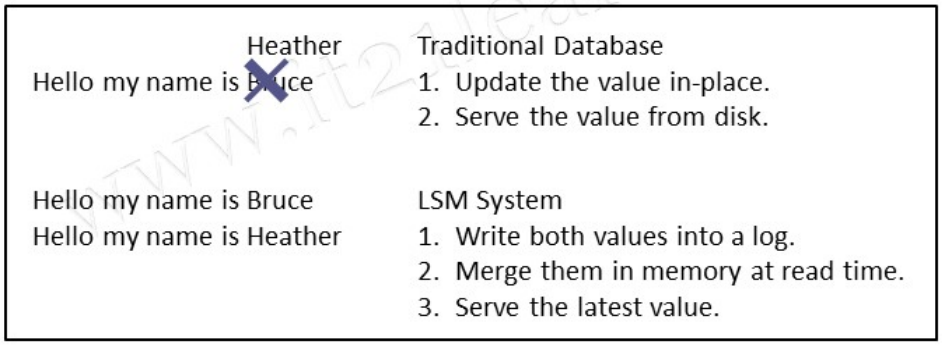


22

Compactions: Motivation

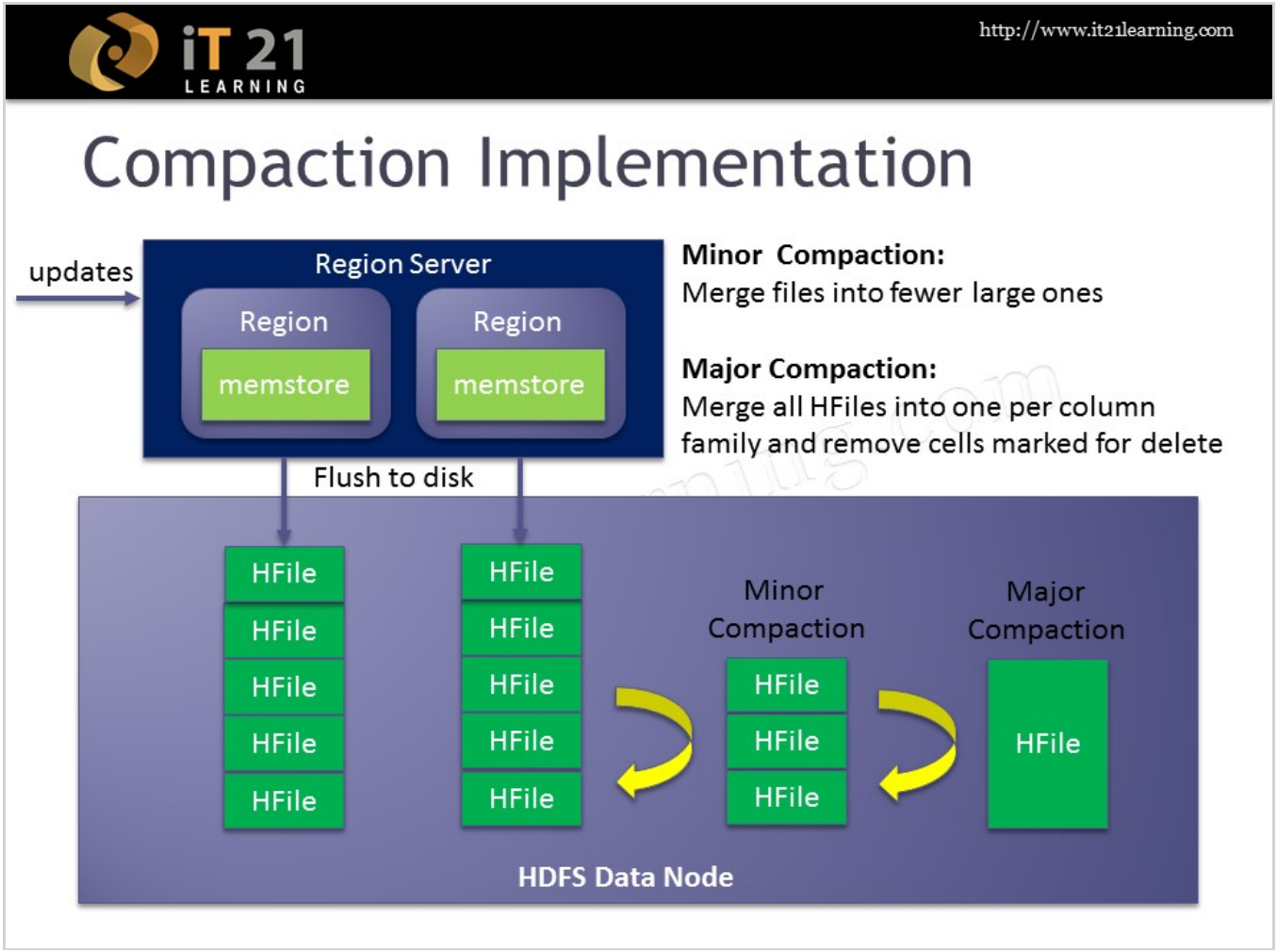
Log-Structured Merge (LSM)

- Traditional databases are architected to update data in-place.
- Most modern databases use some sort of Log-Structured Merge (LSM).
- That means just write values to the end of a log and sort it out later.
- Pro: Inserts and updates are extremely fast.
- Con: Uses lots more space, so we need to do compaction.

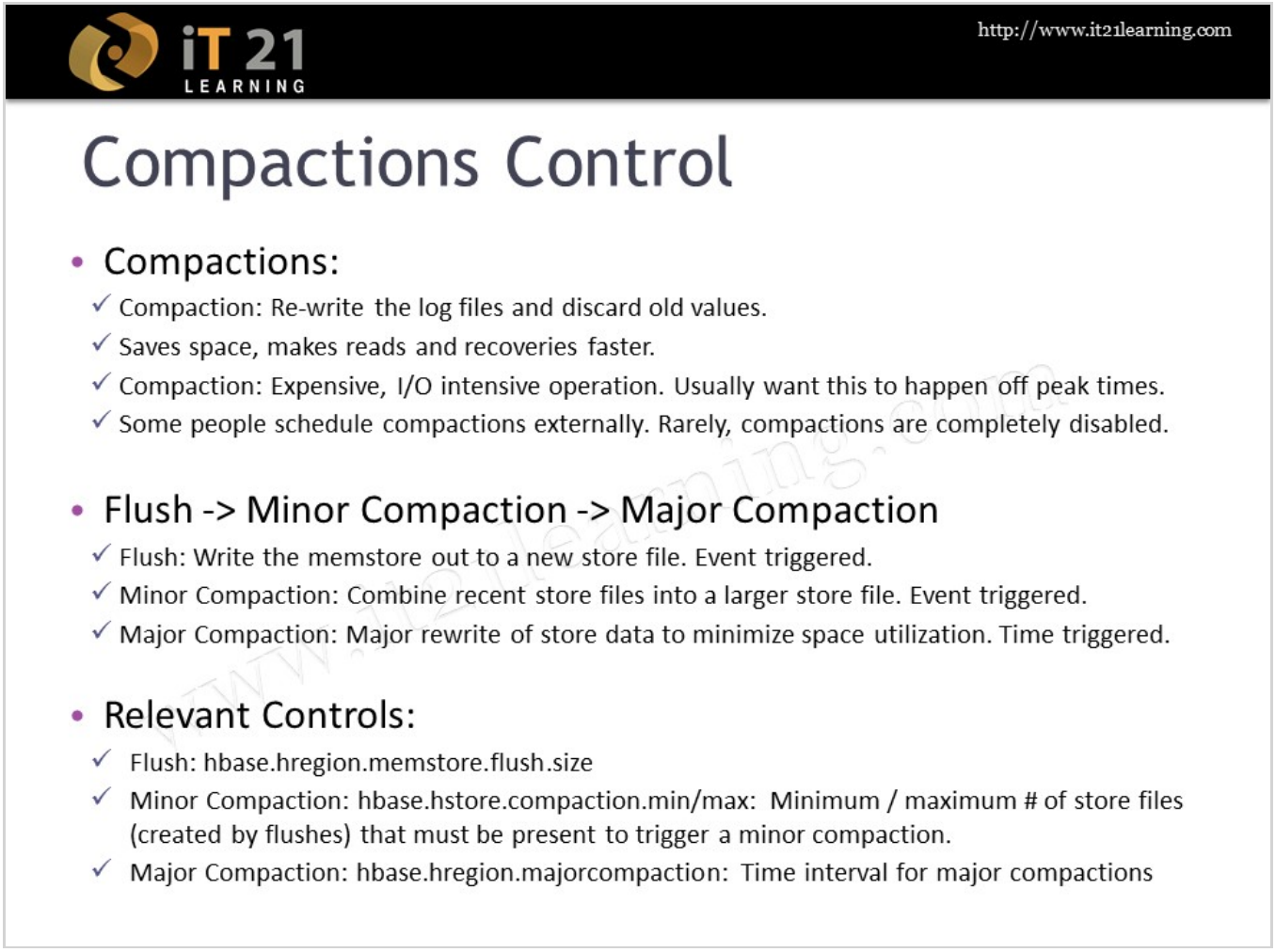


23

Compaction Implementation



24


iT 21 LEARNING
http://www.it21learning.com
Compactions Control
• Compactions:
✓ Compaction: Re-write the log files and discard old values.
✓ Saves space, makes reads and recoveries faster.
✓ Compaction: Expensive, I/O intensive operation. Usually want this to happen off peak times.
✓ Some people schedule compactions externally. Rarely, compactions are completely disabled.
• Flush -> Minor Compaction -> Major Compaction
✓ Flush: Write the memstore out to a new store file. Event triggered.
✓ Minor Compaction: Combine recent store files into a larger store file. Event triggered.
✓ Major Compaction: Major rewrite of store data to minimize space utilization. Time triggered.
• Relevant Controls:
✓ Flush: hbase.hregion.memstore.flush.size
✓ Minor Compaction: hbase.hstore.compaction.min/max: Minimum / maximum # of store files (created by flushes) that must be present to trigger a minor compaction.
✓ Major Compaction: hbase.hregion.majorcompaction: Time interval for major compactions

25

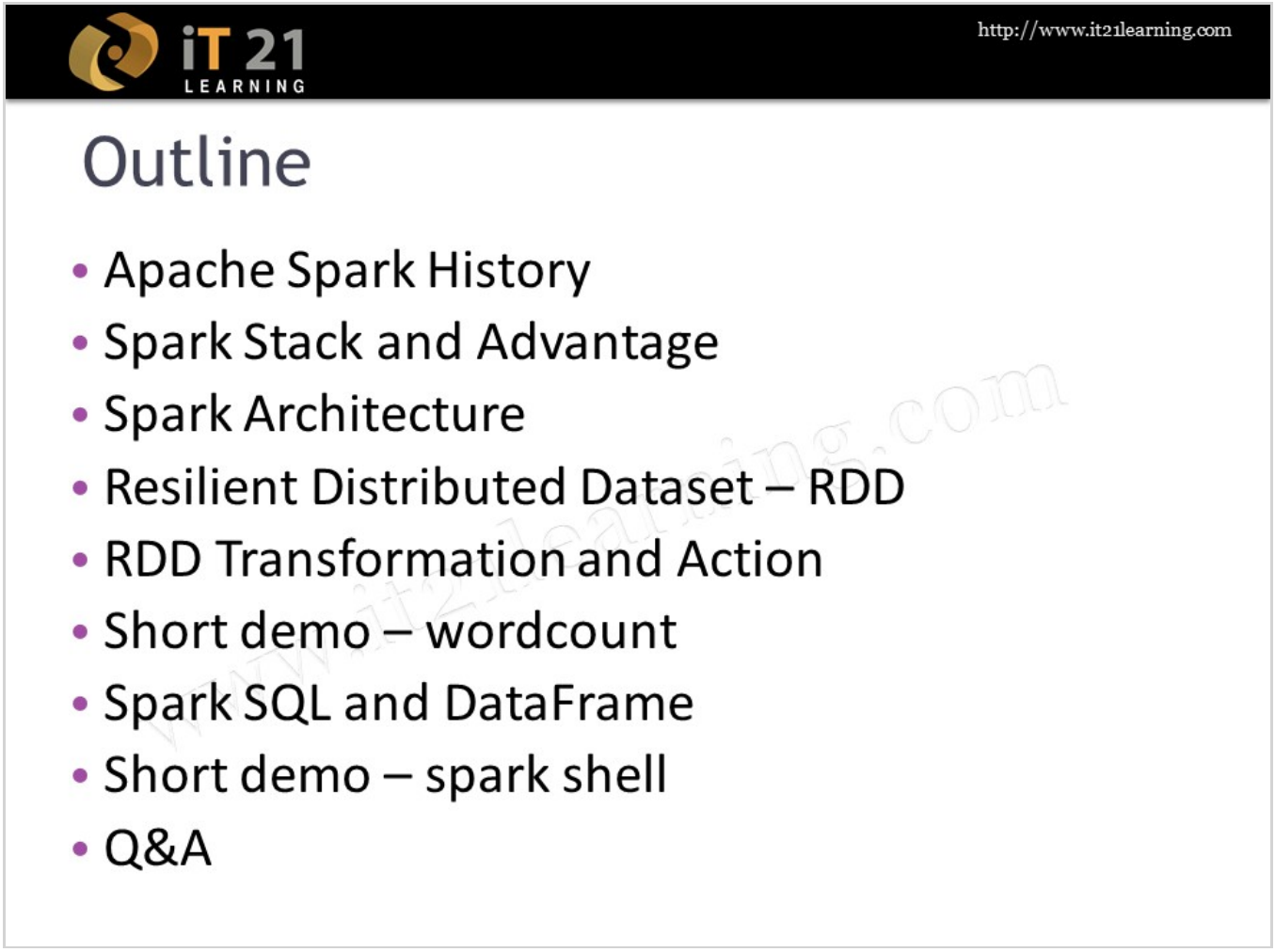
Thank You

26

Introducing Apache Spark

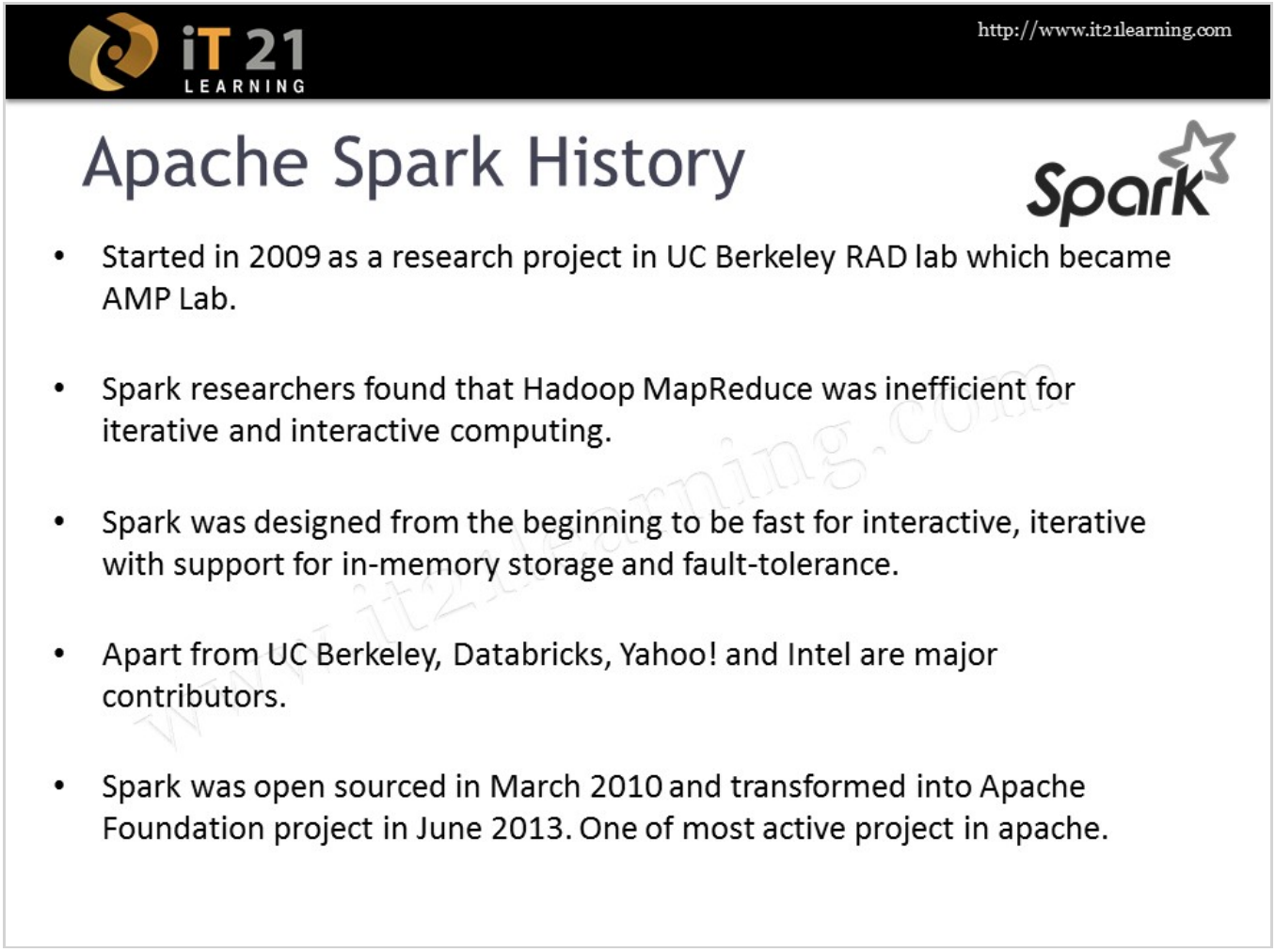
Apache Spark Overview

1


iT 21 LEARNING
http://www.it21learning.com
Outline
• Apache Spark History
• Spark Stack and Advantage
• Spark Architecture
• Resilient Distributed Dataset – RDD
• RDD Transformation and Action
• Short demo – wordcount
• Spark SQL and DataFrame
• Short demo – spark shell
• Q&A

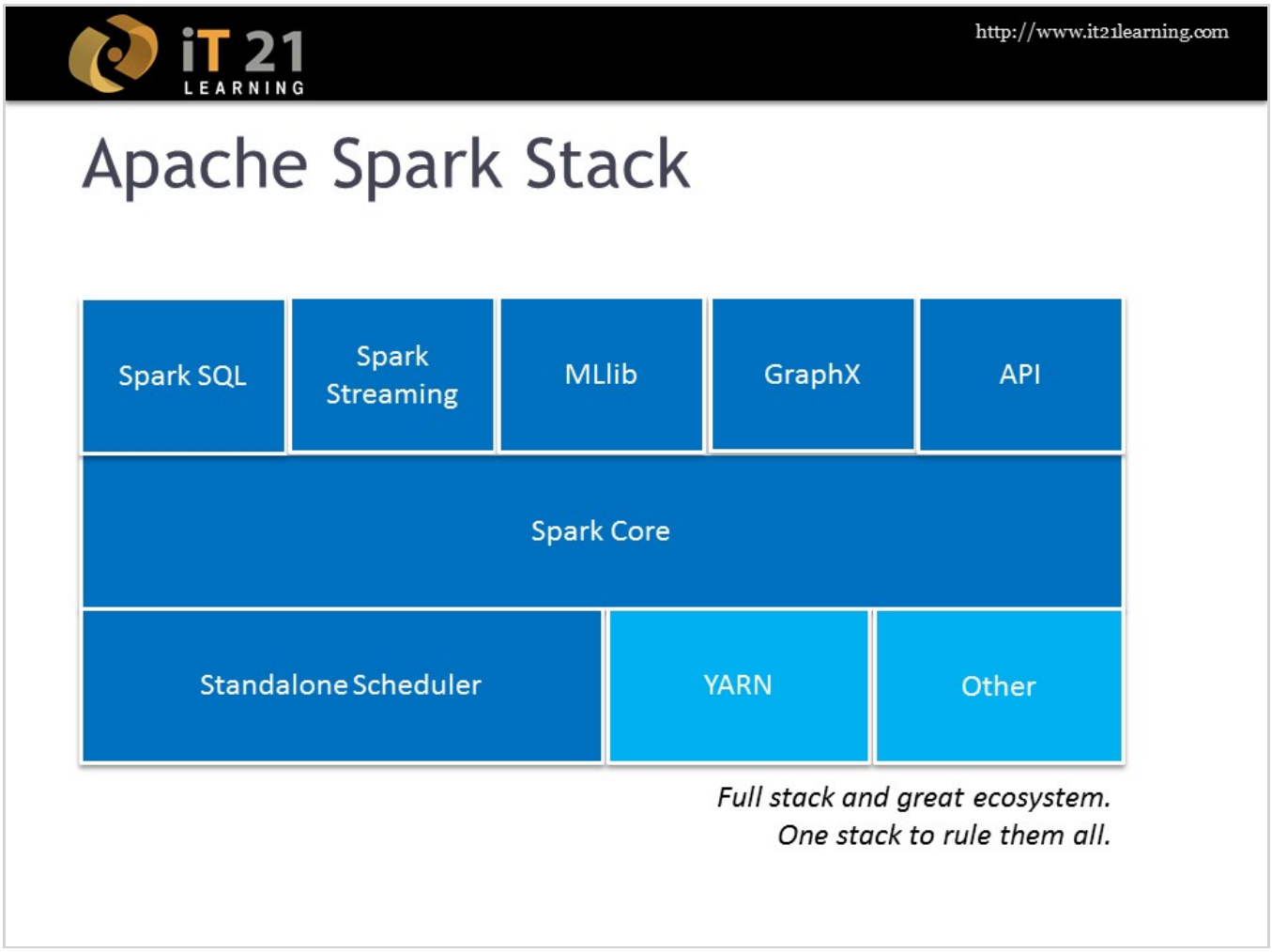
2

Apache Spark History



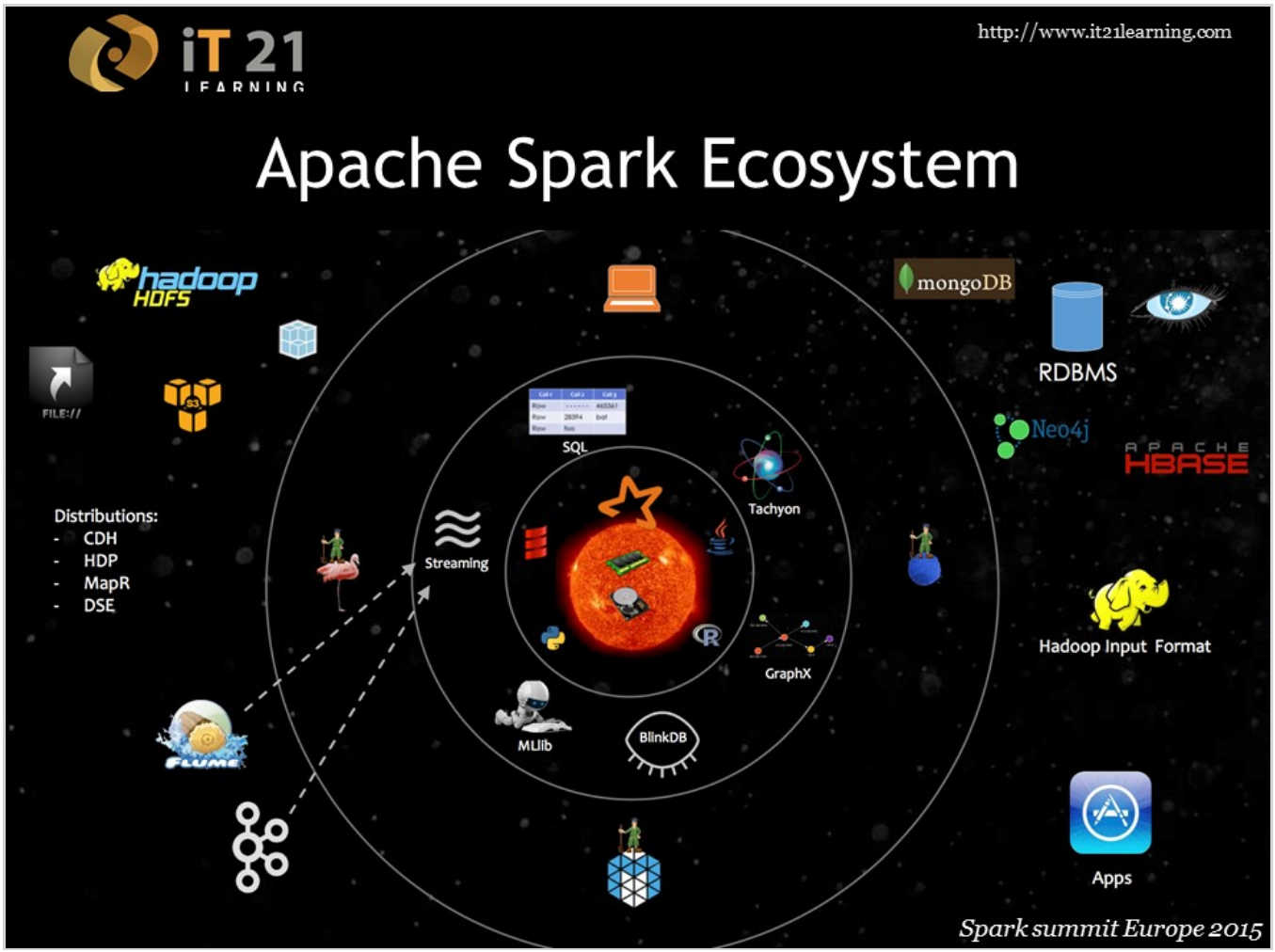
Resilient Distributed Dataset

Apache Spark Stack



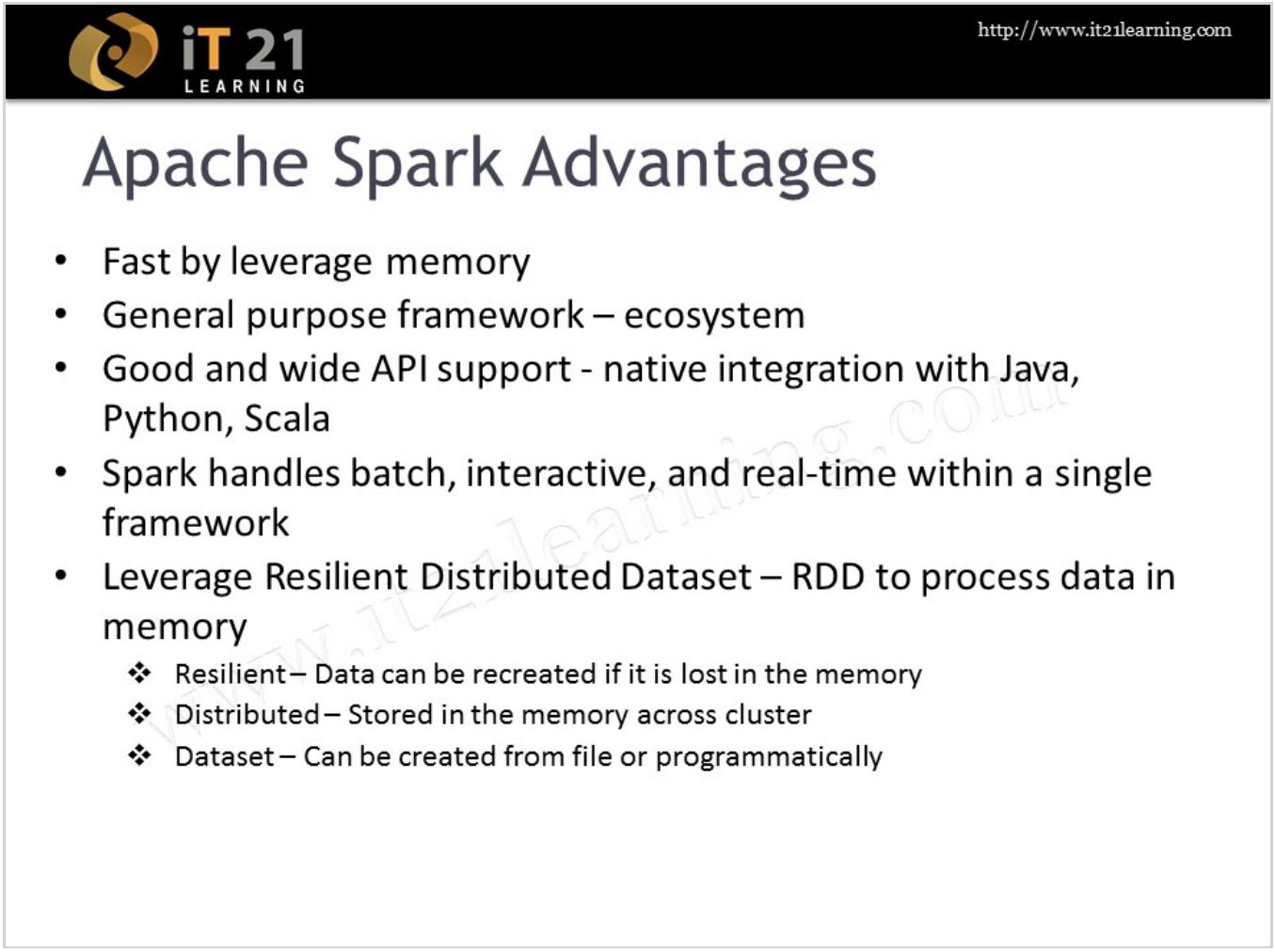
Tachyon Mesos Blinkdb – 17T 100nodes 2sec 2-10 error

Apache Spark Ecosystem



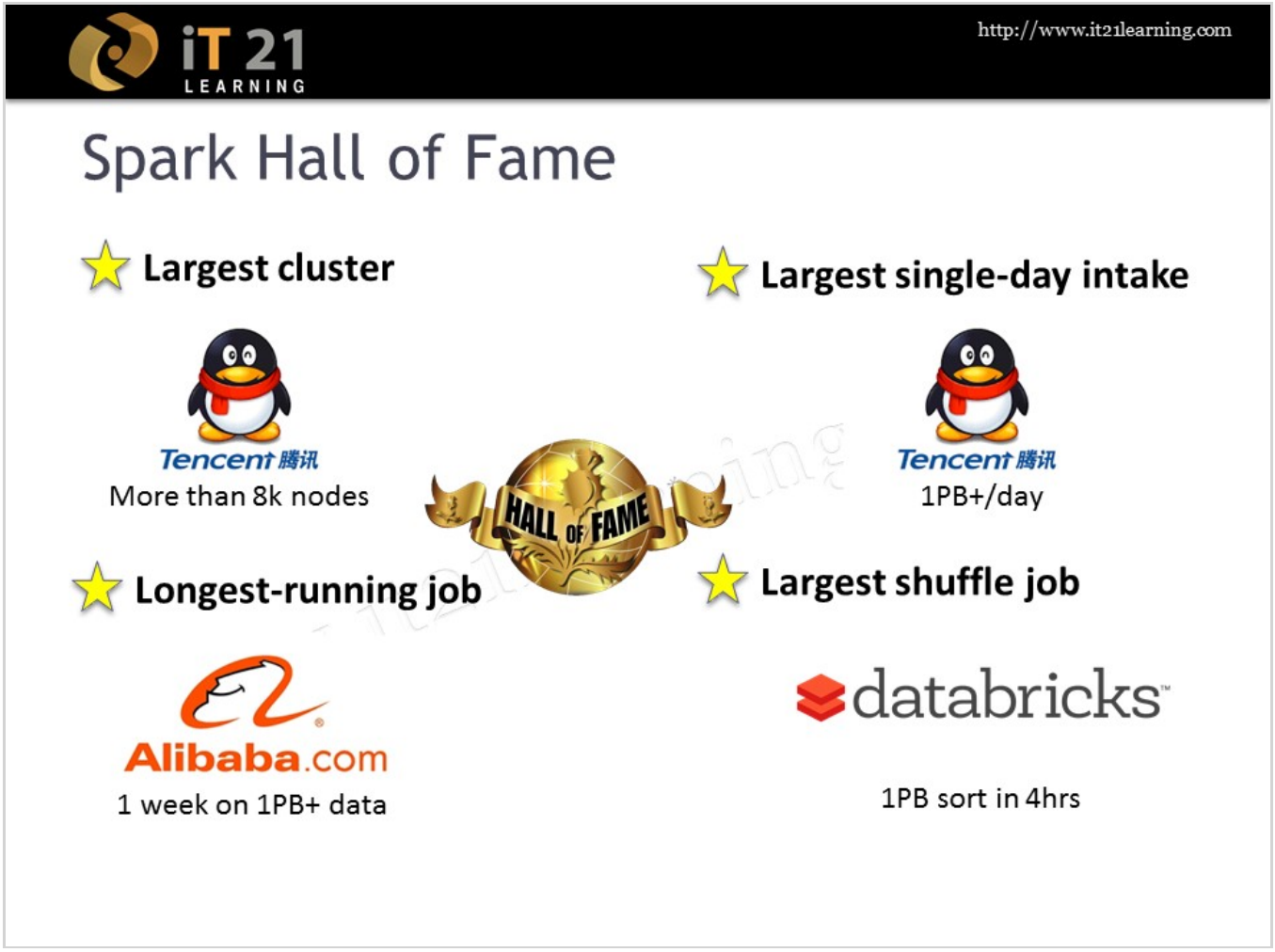
5

Apache Spark Advantages

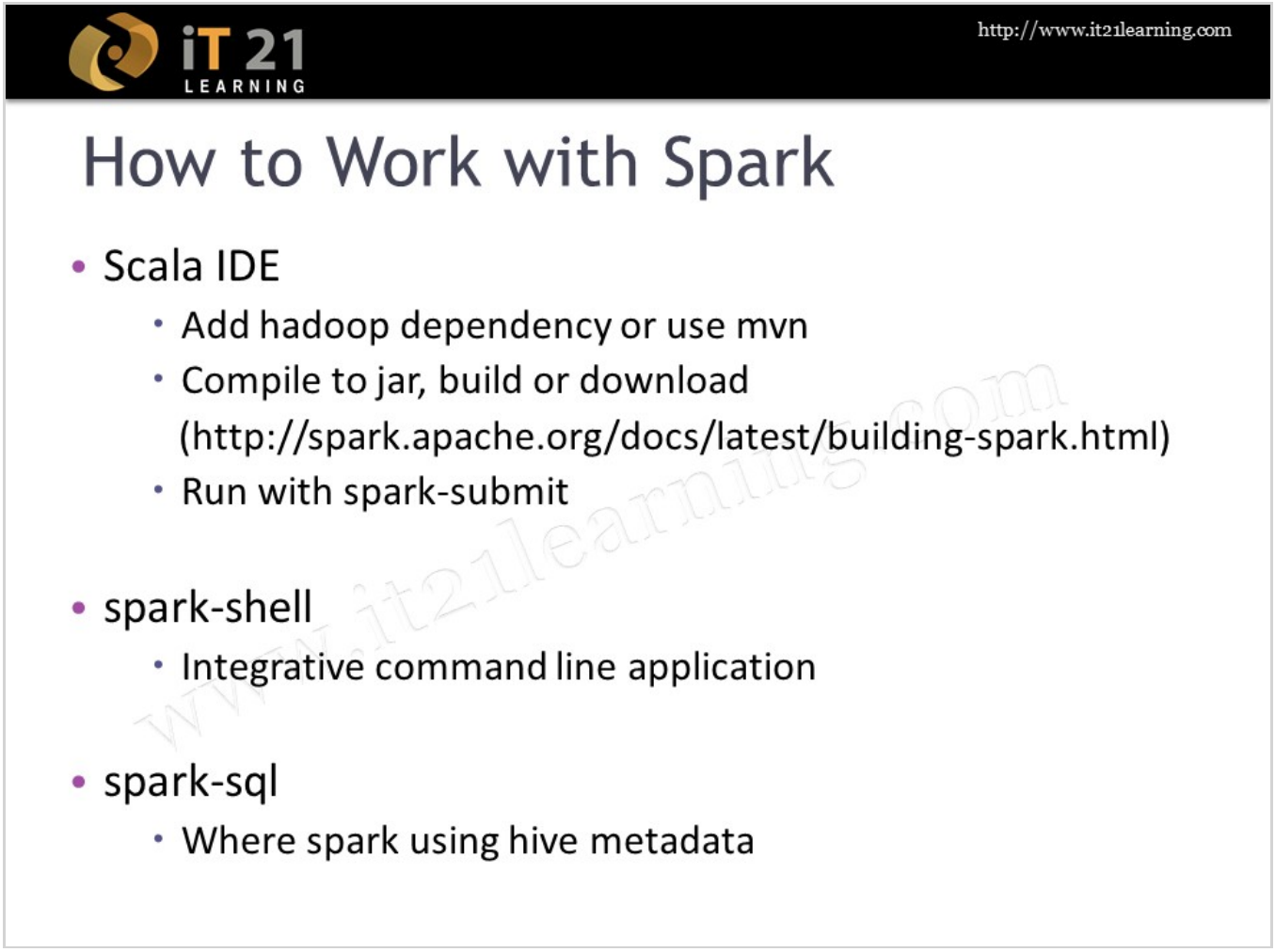


Resilient Distributed Dataset

Spark Hall of Fame



LARGEST CLUSTER Tencent (8000+ nodes) LARGEST SINGLE-DAY INTAKE Tencent (1PB+ /day) LONGEST-RUNNING JOB Alibaba(1 week on 1PB+ data) LARGEST SHUFFLE Databricks PB Sort (1PB) MOST INTERESTING APP Jeremy Freeman Mapping the Brain at Scale (with lasers!)


iT 21 LEARNING
http://www.it21learning.com
How to Work with Spark
• Scala IDE
• Add hadoop dependency or use mvn
• Compile to jar, build or download (http://spark.apache.org/docs/latest/building-spark.html)
• Run with spark-submit
• spark-shell
• Integrative command line application
• spark-sql
• Where spark using hive metadata

8

Demo – Word Count

Demo - Word Count

MR code

```
package org.apache.hadoop.examples;
import java.io.IOException;
import java.util.StringTokenizer;
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;
import org.apache.hadoop.util.GenericOptionsParser;

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();
String[] otherArgs = new GenericOptionsParser(conf, args).getRemainingArgs();
if (otherArgs.length != 2) {
  System.err.println("Usage: wordcount <in> <out>");
  System.exit(2);
  }
Job job = new Job(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(otherArgs[0]));
FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));
System.exit(job.waitForCompletion(true) ? 0 : 1);
}
}
```

Hive code

```
select word, count(*) as word_cnt
from wordcount
lateral view explode(split(line,' ')) t1 as word
group by word
order by word_cnt desc
limit 10
```

How about spark code?

9

Demo – Word Count Cont.

Demo – Word Count *Cont.*

MR code

```
package org.apache.hadoop.examples;
import java.io.IOException;
import java.util.StringTokenizer;
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;
import org.apache.hadoop.util.GenericOptionsParser;

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();
String[] otherArgs = new GenericOptionsParser(conf, args).getRemainingArgs();
if (otherArgs.length != 2) {
  System.err.println("Usage: wordcount <in> <out>");
  System.exit(2);
  }
Job job = new Job(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(otherArgs[0]));
FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));
System.exit(job.waitForCompletion(true) ? 0 : 1);
}
}
```

Hive code

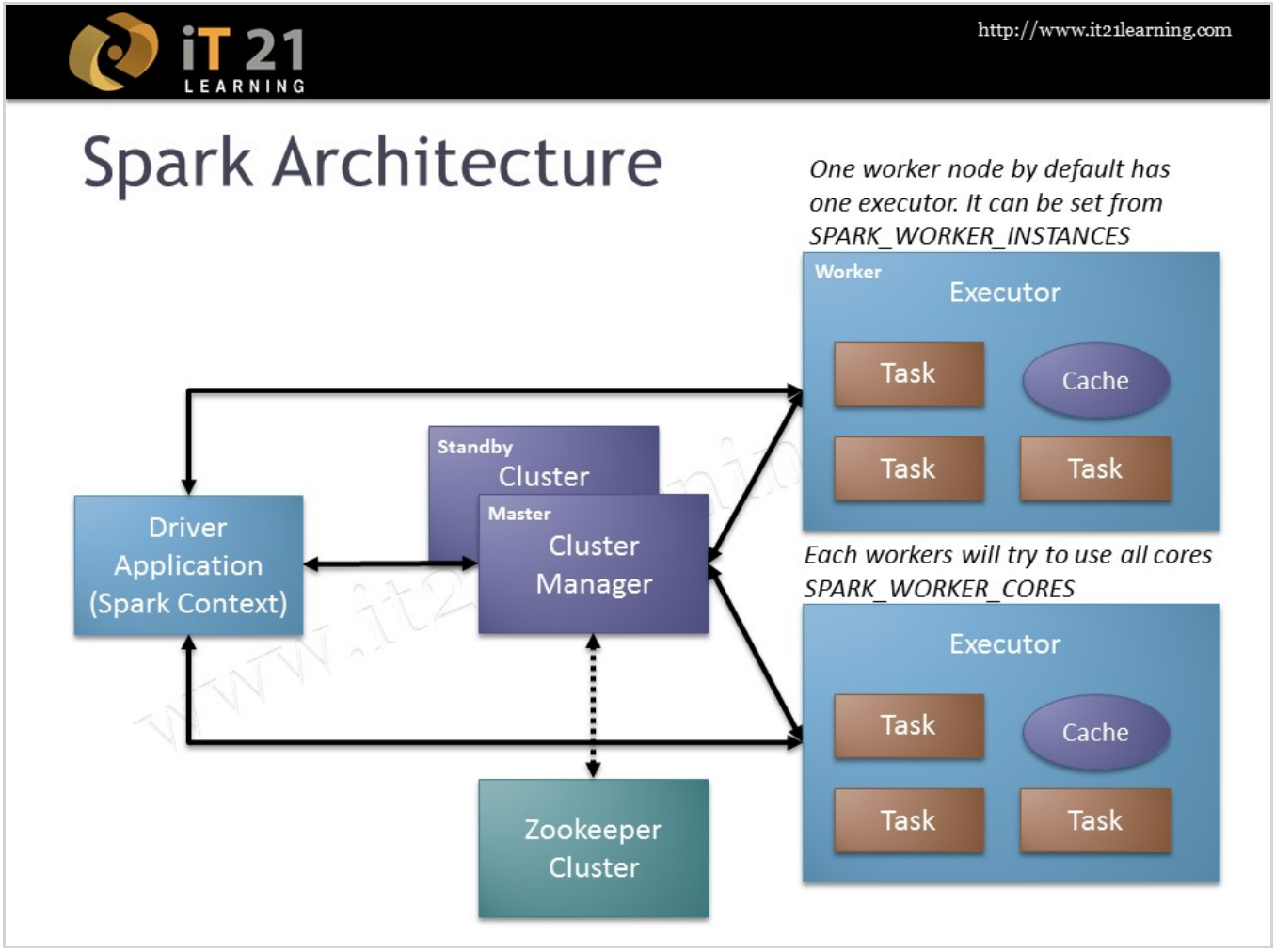
```
select word, count(*) as word_cnt
from wordcount
lateral view explode(split(line,' ')) t1 as word
group by word
order by word_cnt desc
limit 10
```

How about spark code?

```
val wordCounts = sc.textFile("/data/data_words/wordcount.txt").
flatMap(_.split(" ")).filter(_.length>1).map(word=>(word,1)).
reduceByKey(_+_).map(item => item.swap).sortByKey(false);

val result = wordCounts.take(10).foreach(x=>println(x));
```

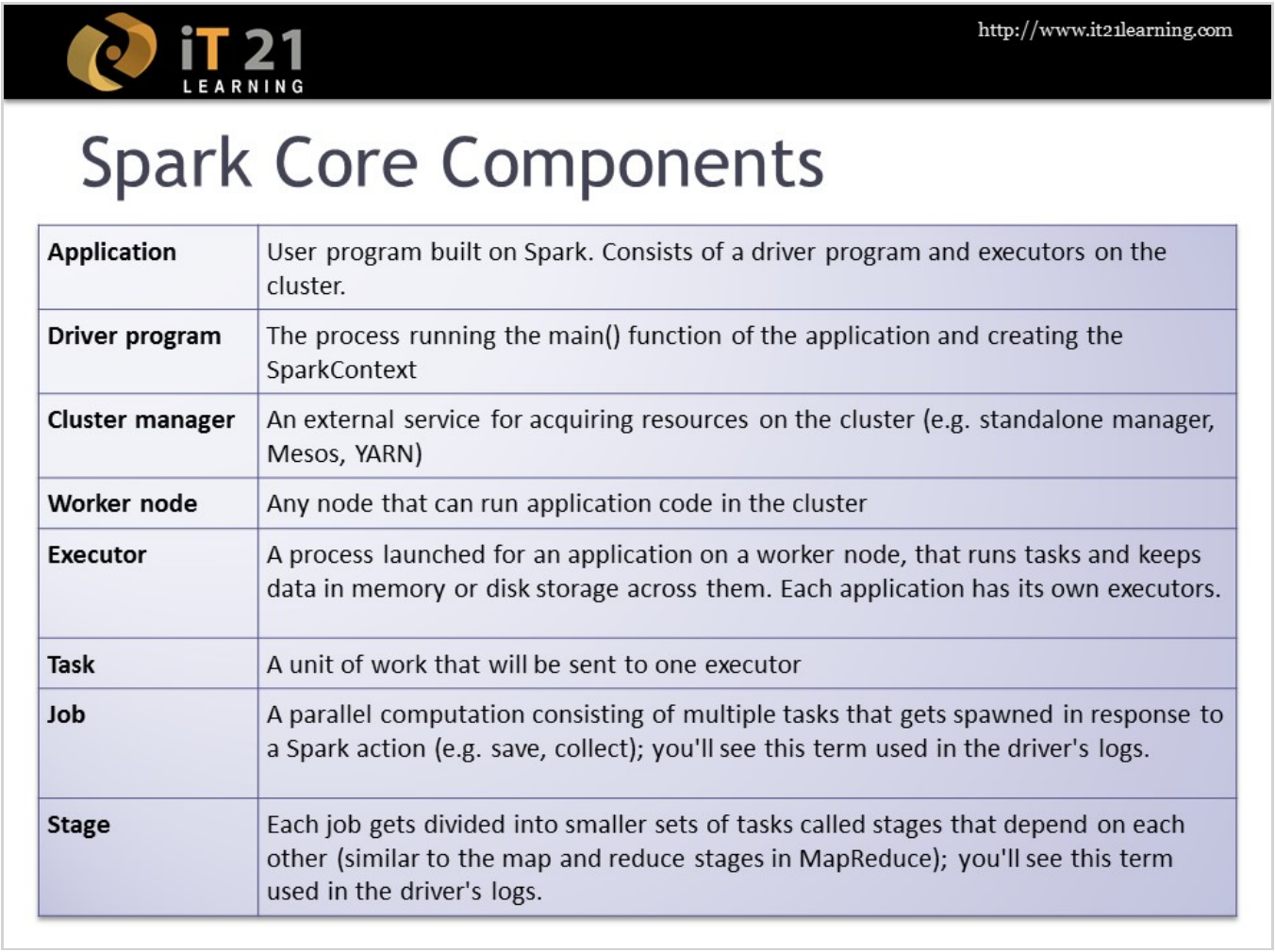
10

Spark Architecture



11

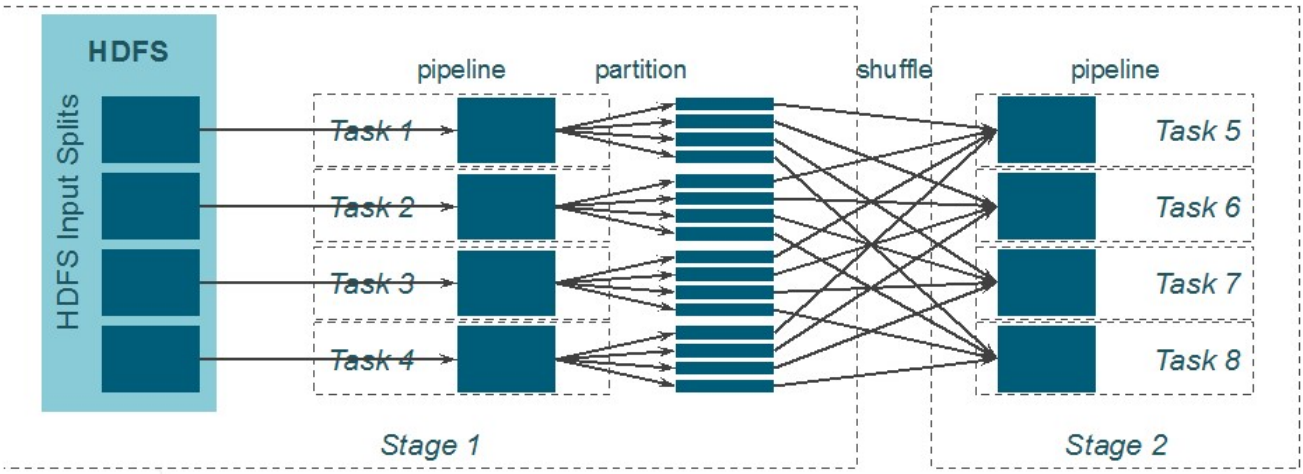
Spark Core Components



12

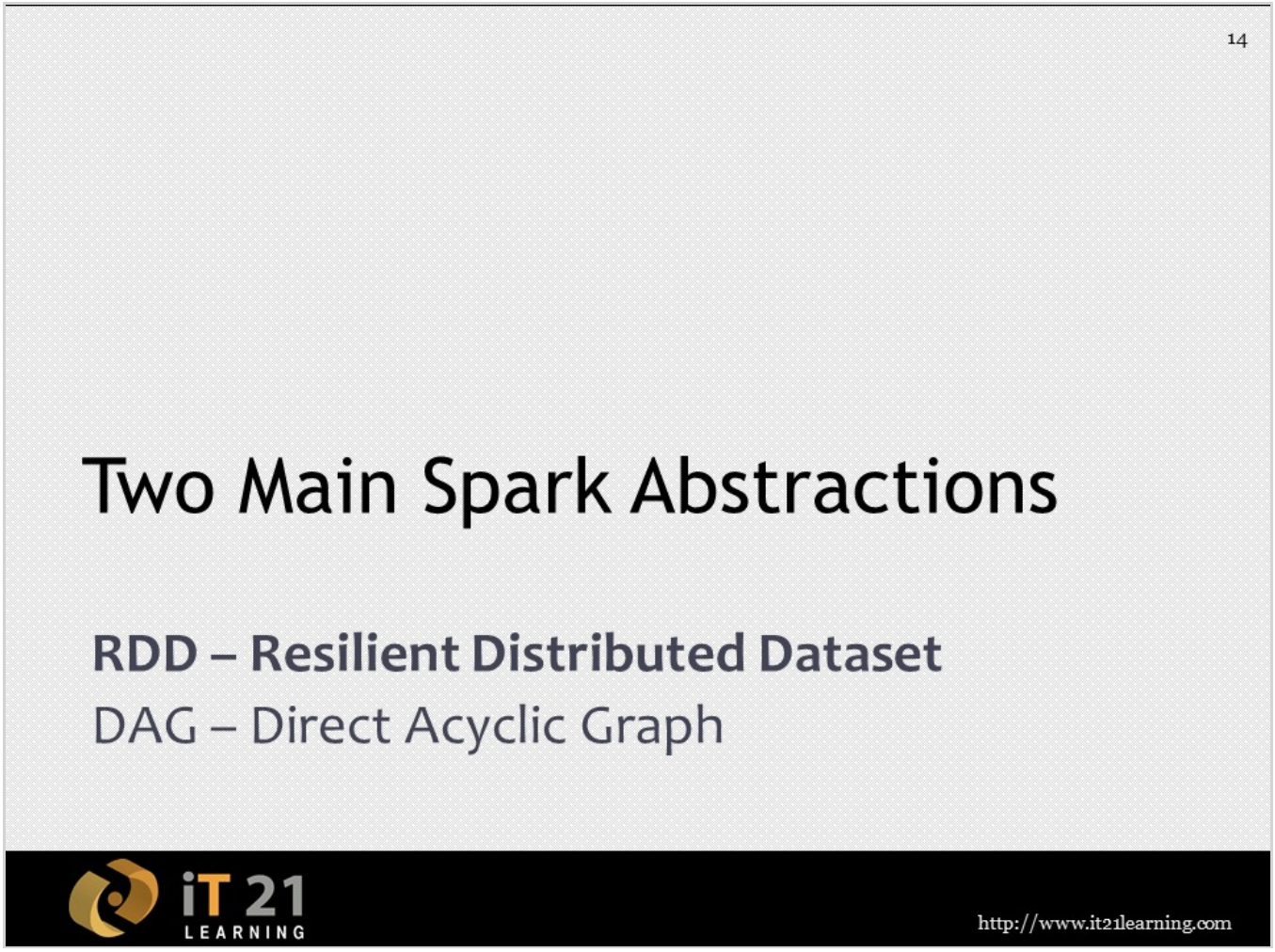
Example of Partial Word Count

Example of Partial Word Count

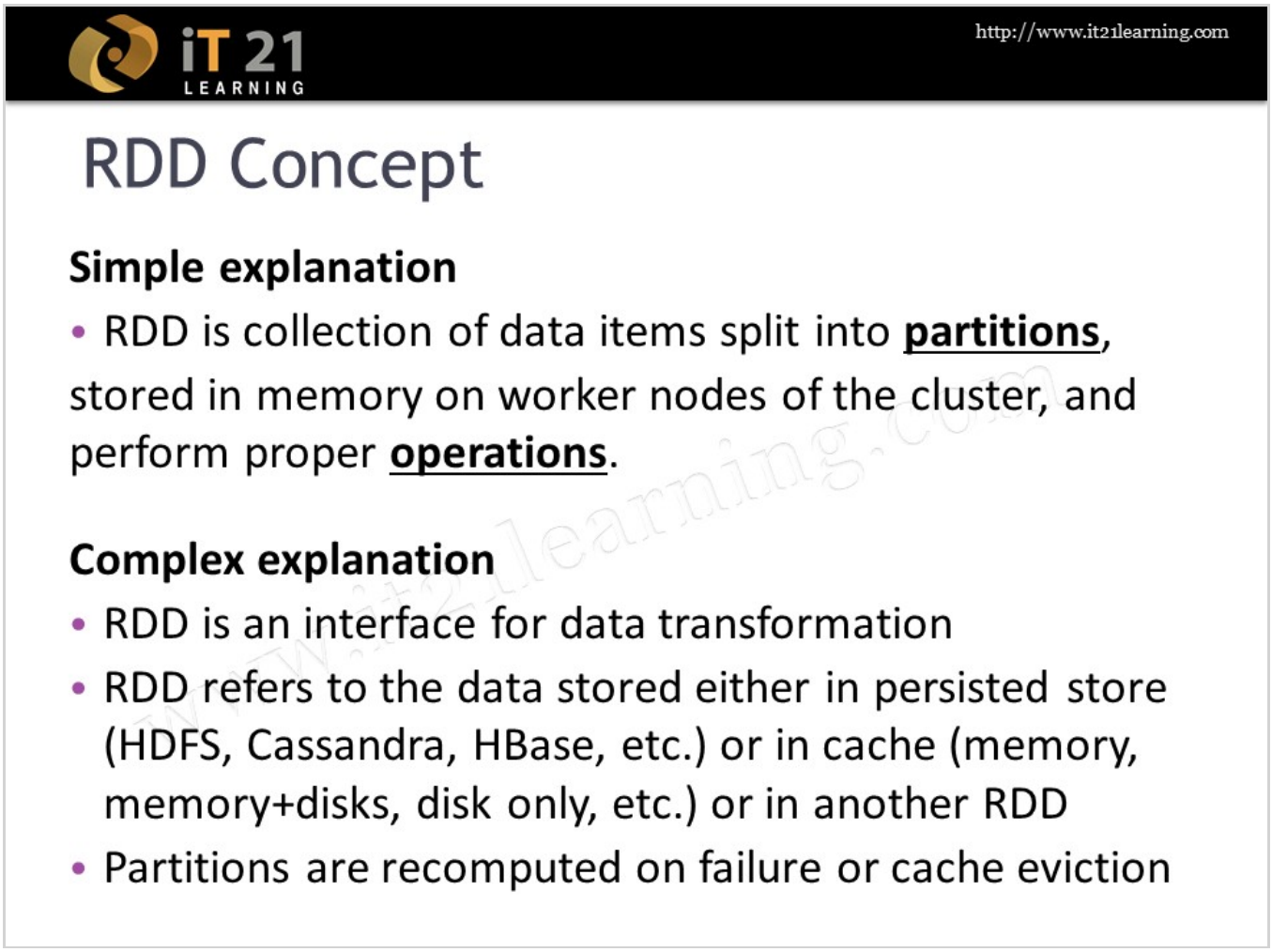


- *Each core by default execute one task a time (spark.task.cpus)*
- *One task per partition, one partition per split*
- *No. of stages is coming from shuffle/wide transformations*

13

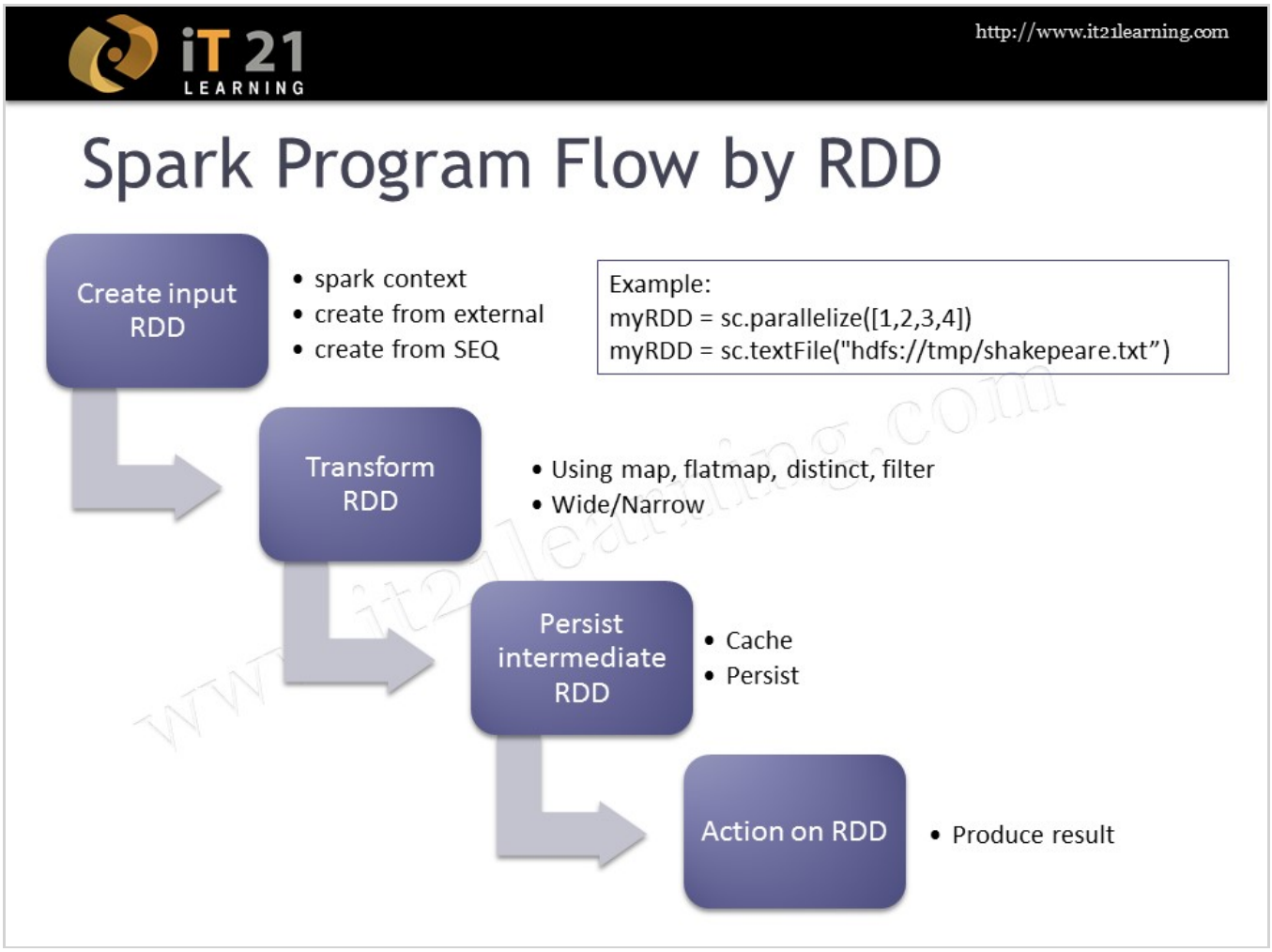

14
Two Main Spark Abstractions
RDD – Resilient Distributed Dataset
DAG – Direct Acyclic Graph
iT 21 LEARNING
http://www.it21learning.com

14


iT 21 LEARNING
http://www.it21learning.com
RDD Concept
Simple explanation
• RDD is collection of data items split into partitions, stored in memory on worker nodes of the cluster, and perform proper operations.
Complex explanation
• RDD is an interface for data transformation
• RDD refers to the data stored either in persisted store (HDFS, Cassandra, HBase, etc.) or in cache (memory, memory+disks, disk only, etc.) or in another RDD
• Partitions are recomputed on failure or cache eviction

15

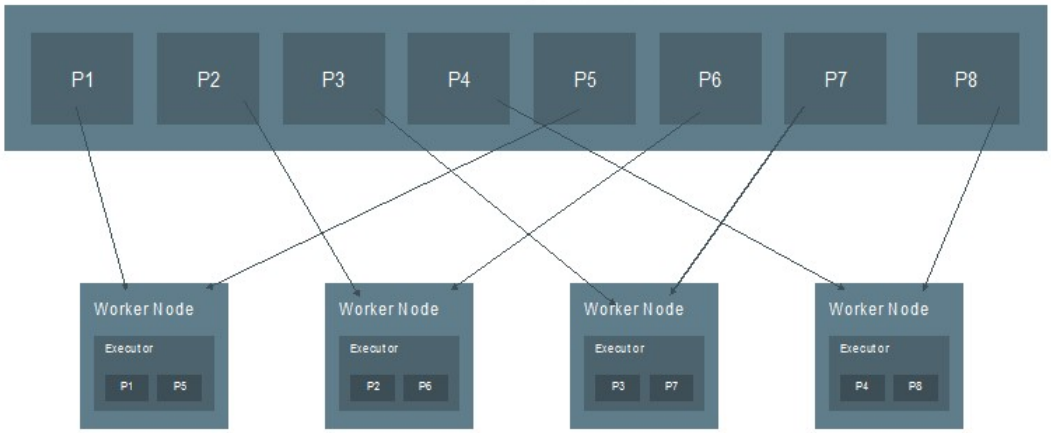
Spark Program Flow by RDD



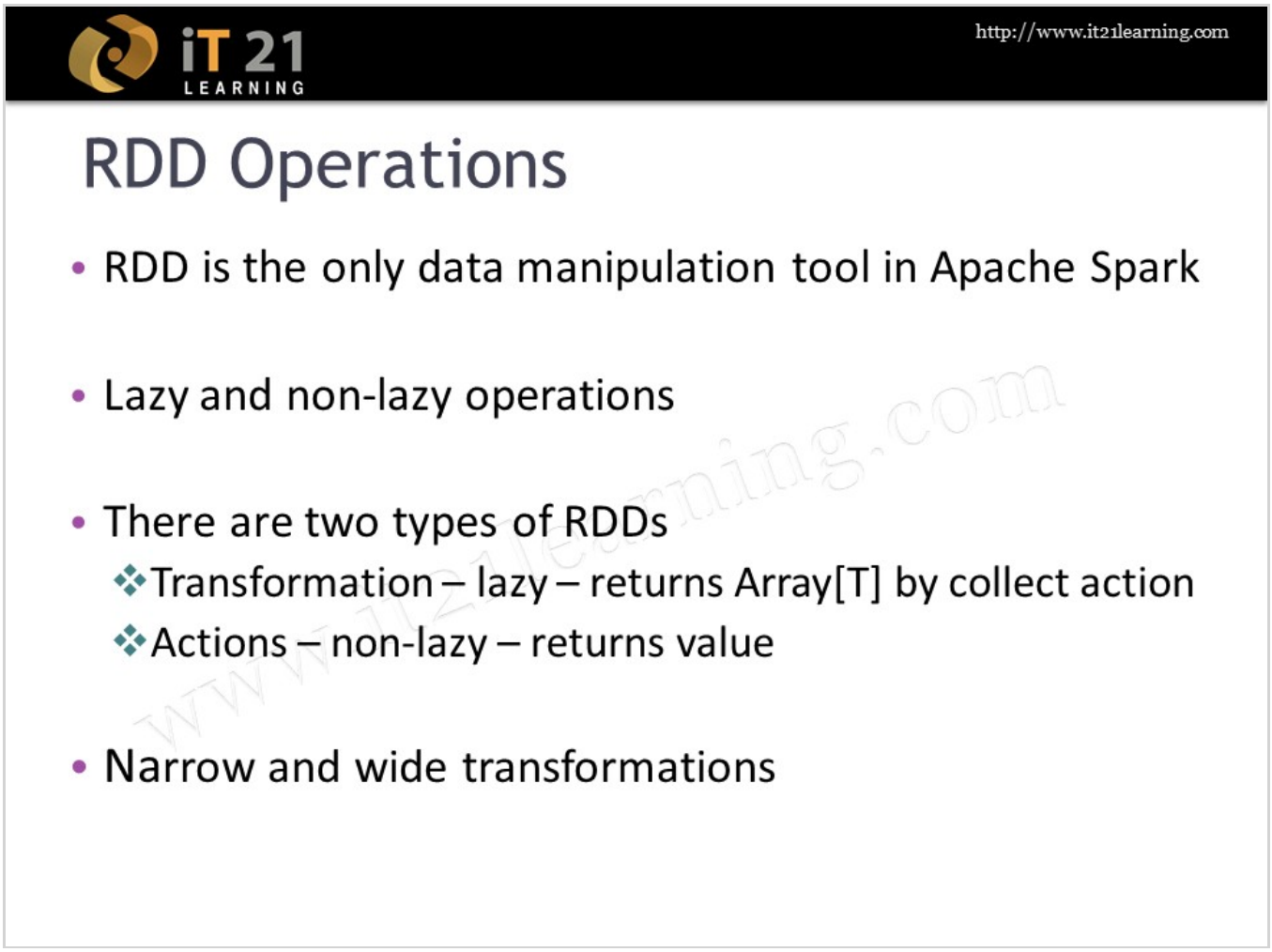
16

RDD Partitions

- A **Partition** is one of the different chunks that a RDD is splitted on and that is sent to a node
- The more partitions we have, the more **parallelism** we get
- Each partition is candidate to be spread out to different **worker nodes**



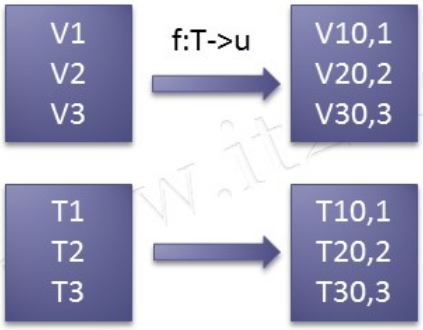
17


iT 21 LEARNING
http://www.it21learning.com
RDD Operations
• RDD is the only data manipulation tool in Apache Spark
• Lazy and non-lazy operations
• There are two types of RDDs
❖Transformation – lazy – returns Array[T] by collect action
❖Actions – non-lazy – returns value
• Narrow and wide transformations

18

RDD Transformations - map

- Applies a transformation function on each item of the RDD and returns the result as a new RDD.
- A narrow transformation

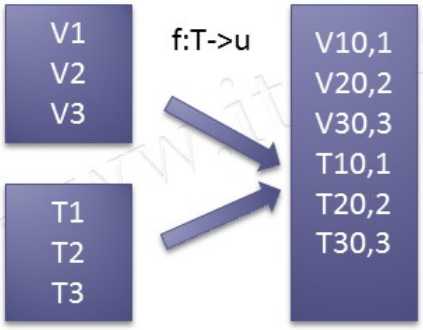



```
val a = sc.parallelize(List("dog", "salmon", "pig"), 3)
val b = a.map(x=>(x,x.length))
b.collect()
b.partitions.size
```


19

RDD Transformations - flatMap

- Similar to map, but each input item can be mapped to 0 or more output items.
- A narrow transformation

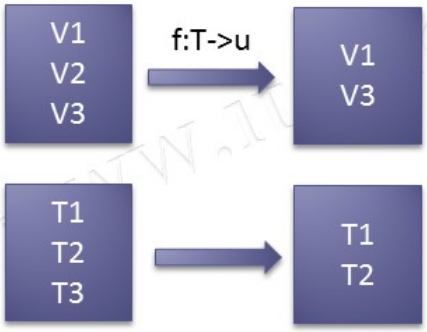



```
val a = sc.parallelize(List("dog", "salmon", "pig"), 3)
val c = a.flatMap(x=>List(x,x))
c.collect
val d = a.map(x=>List(x,x)).collect
```

20

RDD Transformations - filter

- Create a new RDD by passing in a function used to filter the results. It is similar to where clause in SQL language
- A narrow transformation



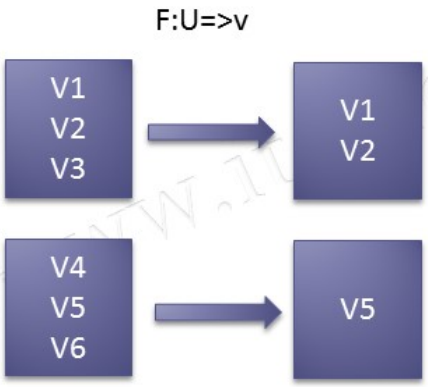

```
val a = sc.parallelize(List("dog", "salmon", "pig"), 3)
val e = a.filter(x=>x.length>3)
e.collect
```


21

RDD Transformations - sample

- Return a random sample subset RDD of the input RDD
- A narrow transformation

def sample(withReplacement: Boolean, fraction: Double, seed: Int): RDD[T]

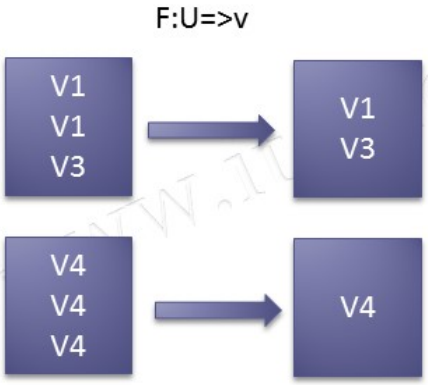



```
val sam = sc.parallelize(1 to 10, 3)
sam.sample(false,  0.1, 0).collect
sam.sample(true,  0.3, 1).collect
```

22

RDD Transformations - distinct

- Return a new RDD with distinct elements within a source RDD with specific partitions
- A narrow/wide transformation

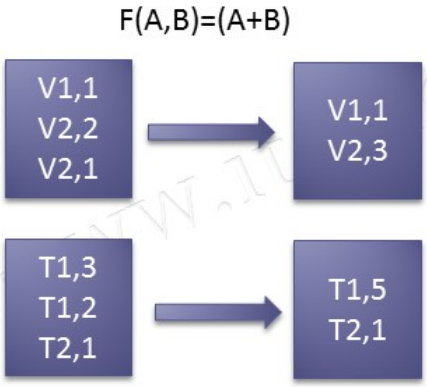



```
Val dis = sc.parallelize(List(1,2,3,4,5,6,7,8,9,9,2,6))
dis.distinct.collect
dis.distinct(2).partitions.length
```

23

RDD Transformations - reduceByKey

- Operates on (K,V) pairs of course, but the func must be of type (V,V) => V
- A narrow/wide transformation



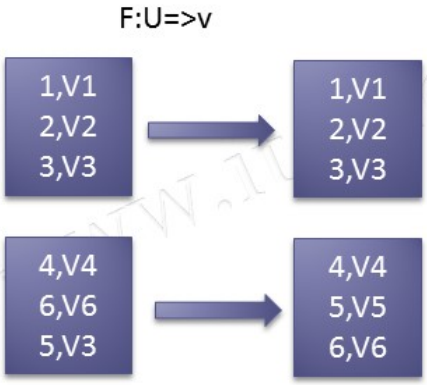

```
val a = sc.parallelize(List("dog", "salmon", "pig"), 3)
val f = a.map(x=>(x.length,x))
f.reduceByKey((a,b)=>(a+b)).collect
f.reduceByKey(_+_).collect
f.groupByKey.collect
```

In terms of group, ReduceByKey is more efficient with combiner in the map phase.

24

RDD Transformations - sortByKey

- This simply sorts the (K,V) pair by K. Default is asc. False as desc.
- A narrow/wide transformation

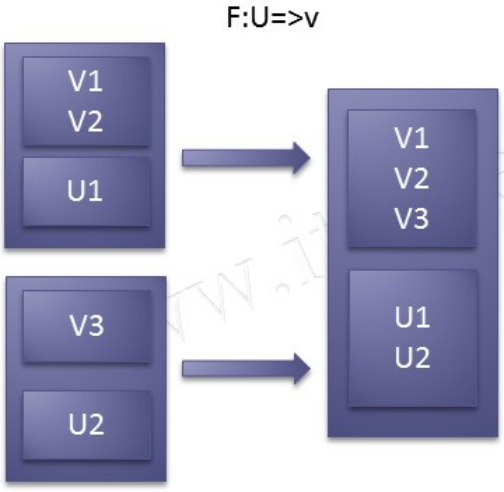



```
val a = sc.parallelize(List("dog", "salmon", "pig"), 3)
val f = a.map(x=>(x.length,x))
f.sortByKey().collect
f.sortByKey(false).collect
```

25

RDD Transformations - union

- Returns the union of two RDDs to a new RDD
- A narrow transformation

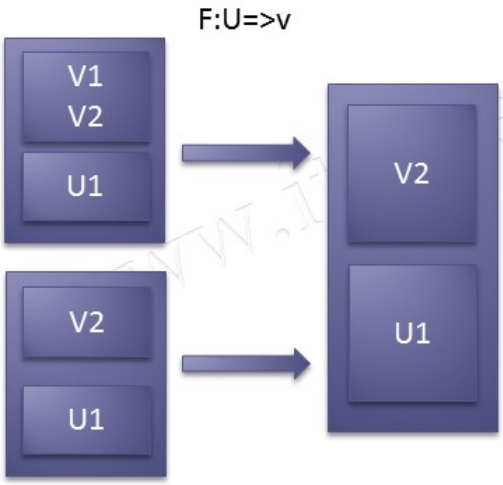



```
val u1 = sc.parallelize(1 to 3)
val u2 = sc.parallelize(3 to 4)
u1.union(u2).collect
(u1 ++ u2).collect
```


26

RDD Transformations - intersection

- Returns a new RDD containing only common elements from source RDD and argument.
- A narrow/wide transformation

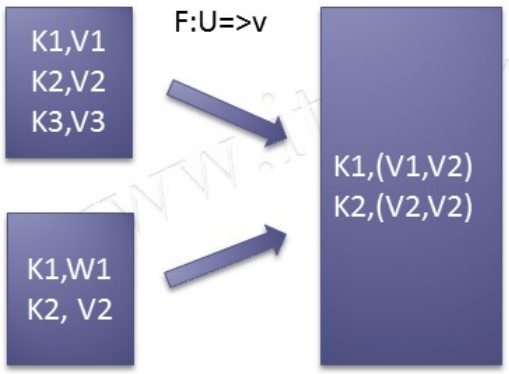



```
val u1 = sc.parallelize(1 to 3)
val u2 = sc.parallelize(3 to 4)
u1.intersection(u2).collect
```

27

RDD Transformations - join

- When invoked on (K,V) and (K,W), this operation creates a new RDD of (K, (V,W)). Must apply on key-value pairs.
- A wide transformation




```
val j1 = sc.parallelize(List("abe",  "abby",
"apple")).map(a  => (a, 1))
val j2 = sc.parallelize(List("apple",  "beatty",
"beatrice")).map(a  => (a, 1))
j1.join(j2).collect
j1.leftOuterJoin(j2).collect
j1.rightOuterJoin(j2).collect
```


28

RDD Actions - count

- Get the number of data elements in the RDD

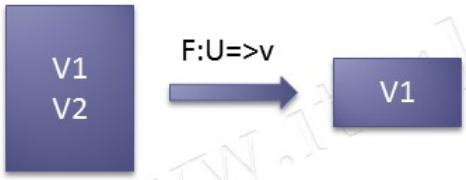
F:U=>v → 3

```
val u1 = sc.parallelize(1 to 3)
val u2 = sc.parallelize(3 to 10)
u1.count
u2.count
```


29

RDD Actions - take

- Fetch number of data elements from the RDD

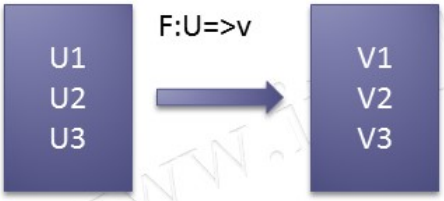



```
val t1 = sc.parallelize(3 to 10)
t1.take(3)
```


30

RDD Actions - foreach

- Execute function for each data element in RDD.




```
val c = sc.parallelize(List("cat",  "dog", "tiger", "lion"))
c.foreach(x => println(x + "s are fun"))
```


31

RDD Actions - saveAsTextFile

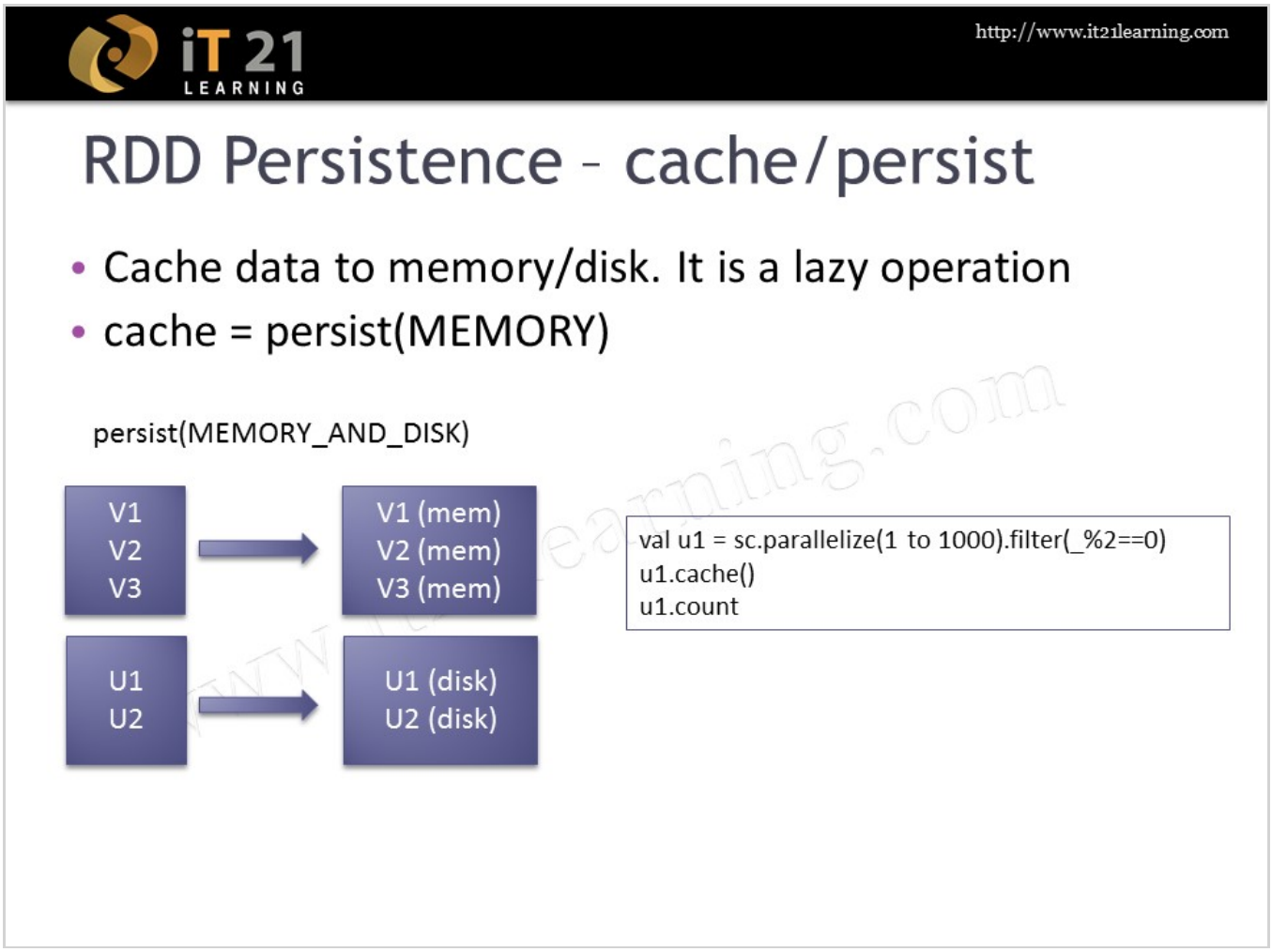
- Writes the content of RDD to a text file or a set of text files to local file system/HDFS. The default path is HDFS or hdfs://

```
val a = sc.parallelize(1 to 10000, 3)
a.saveAsTextFile("file:/Users/will/Downloads/mydata")
```

```
[root@sandbox Downloads]# ls -al /Users/will/Downloads/mydata
total 80
drwxr-xr-x 2 root root  4096 Dec  6 02:48 .
drwxr-xr-x 3 root root  4096 Dec  6 02:48 ..
-rw-r--r-- 1 root root     8 Dec  6 02:48 ._SUCCESS.crc
-rw-r--r-- 1 root root   132 Dec  6 02:48 .part-00000.crc
-rw-r--r-- 1 root root   140 Dec  6 02:48 .part-00001.crc
-rw-r--r-- 1 root root   140 Dec  6 02:48 .part-00002.crc
-rw-r--r-- 1 root root     0 Dec  6 02:48 _SUCCESS
-rw-r--r-- 1 root root 15558 Dec  6 02:48 part-00000
-rw-r--r-- 1 root root 16665 Dec  6 02:48 part-00001
-rw-r--r-- 1 root root 16671 Dec  6 02:48 part-00002
```

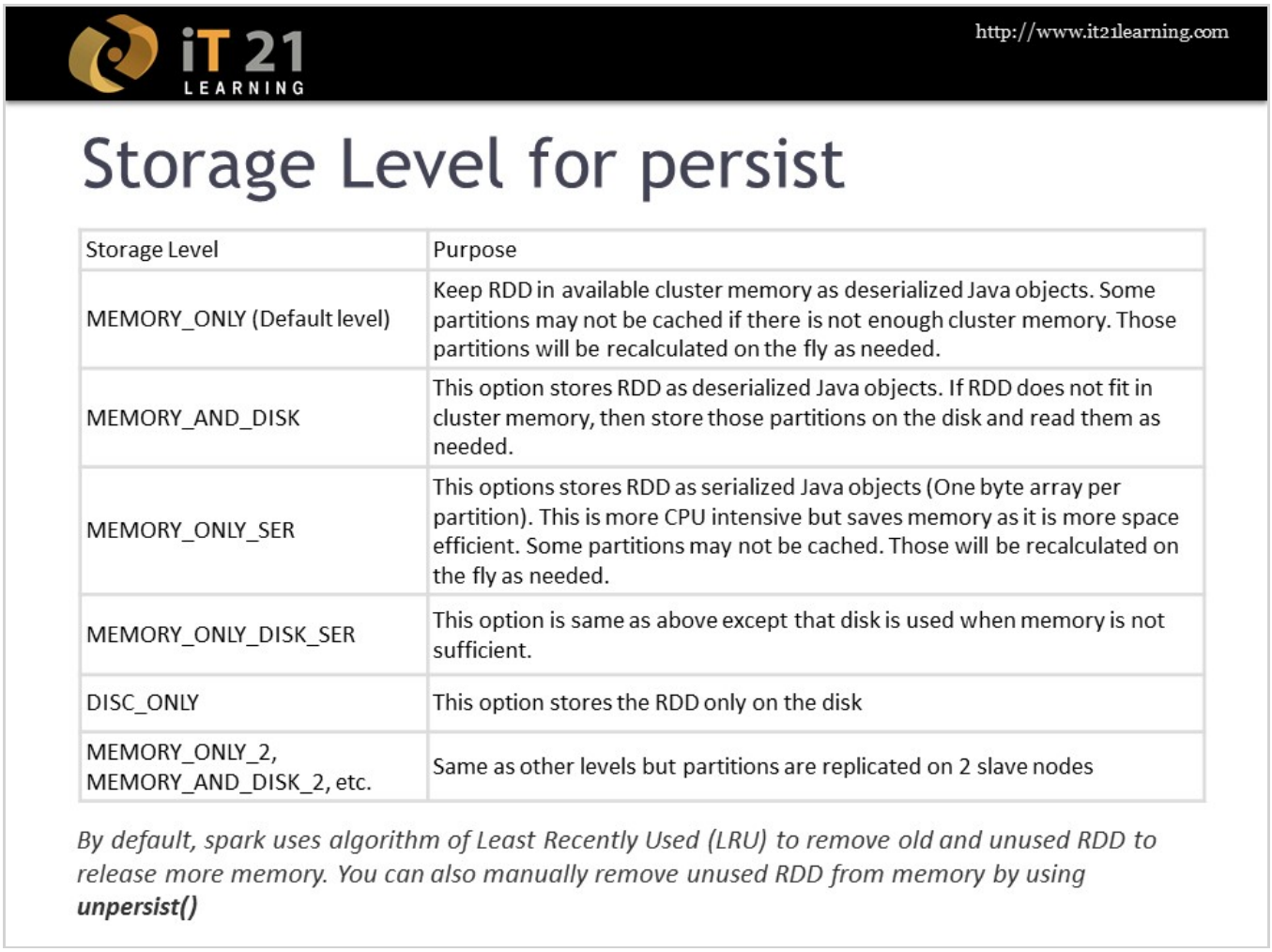
32

RDD Persistence – cache/persist

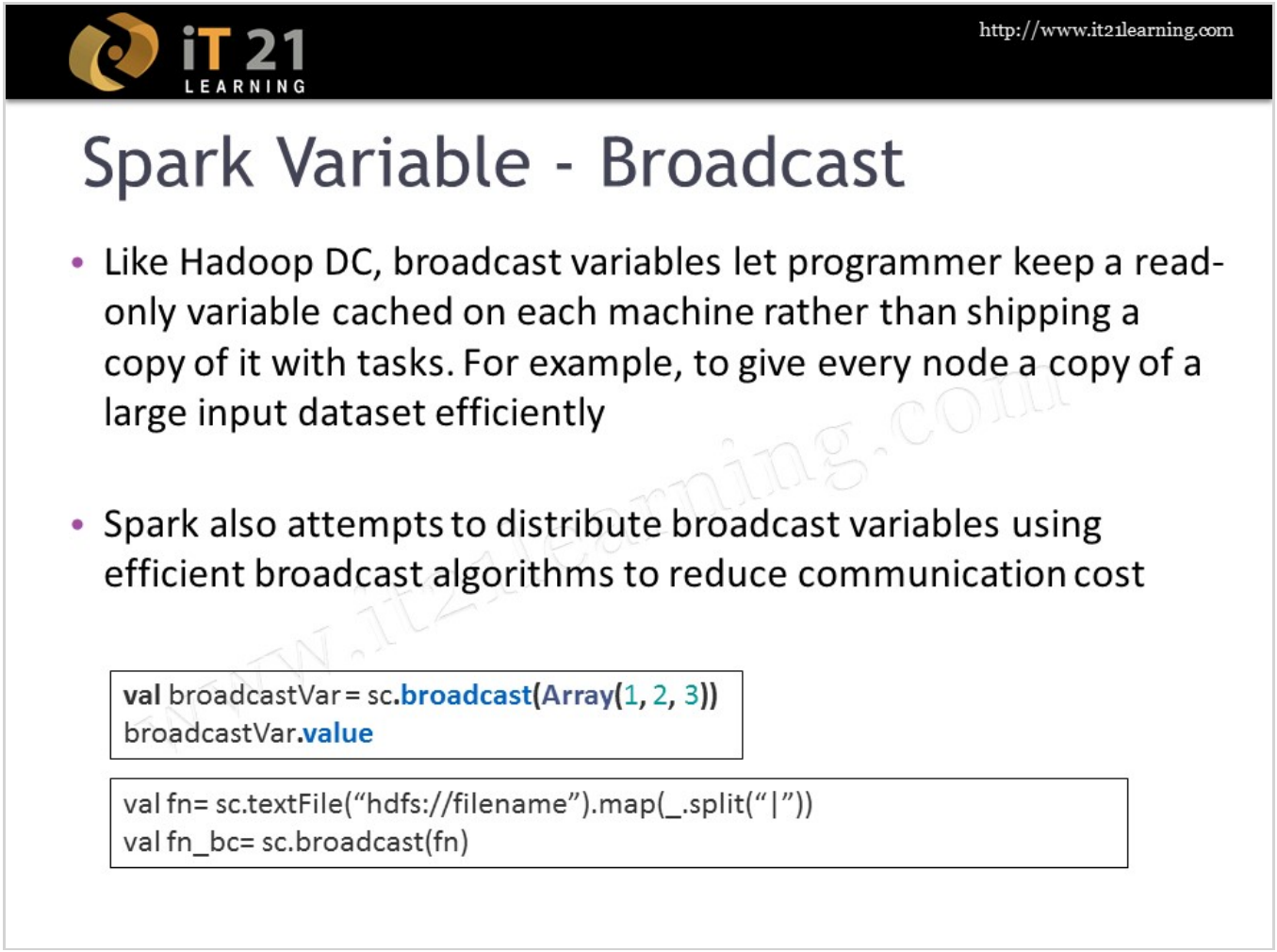


RDD Persistence

Storage Level for persist



34


iT 21 LEARNING
http://www.it21learning.com
Spark Variable - Broadcast
• Like Hadoop DC, broadcast variables let programmer keep a read-only variable cached on each machine rather than shipping a copy of it with tasks. For example, to give every node a copy of a large input dataset efficiently
• Spark also attempts to distribute broadcast variables using efficient broadcast algorithms to reduce communication cost
val broadcastVar = sc.broadcast(Array(1, 2, 3))
broadcastVar.value
val fn= sc.textFile("hdfs://filename").map(_.split("|"))
val fn_bc= sc.broadcast(fn)

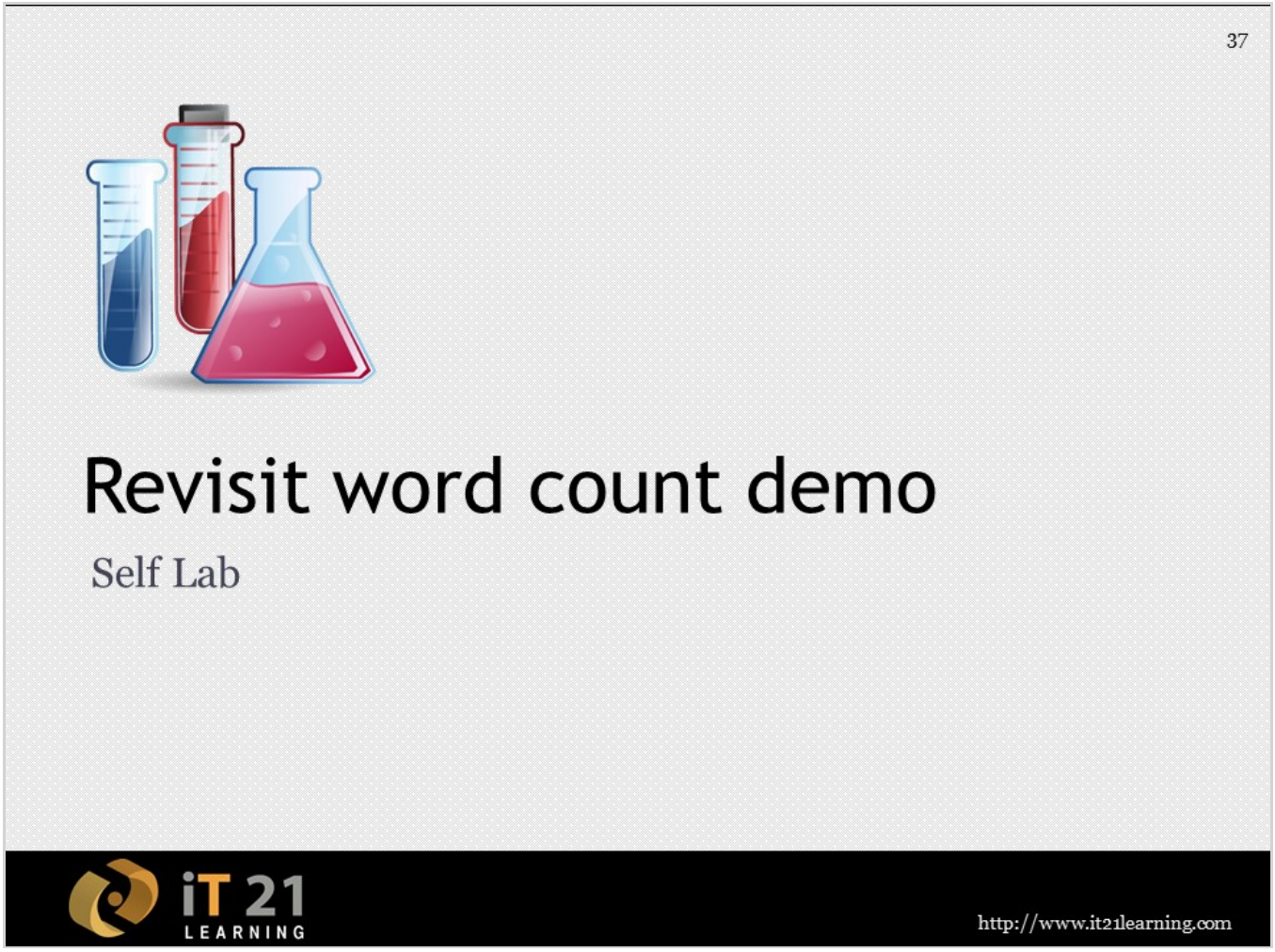
35

Spark Variable - Accumulators

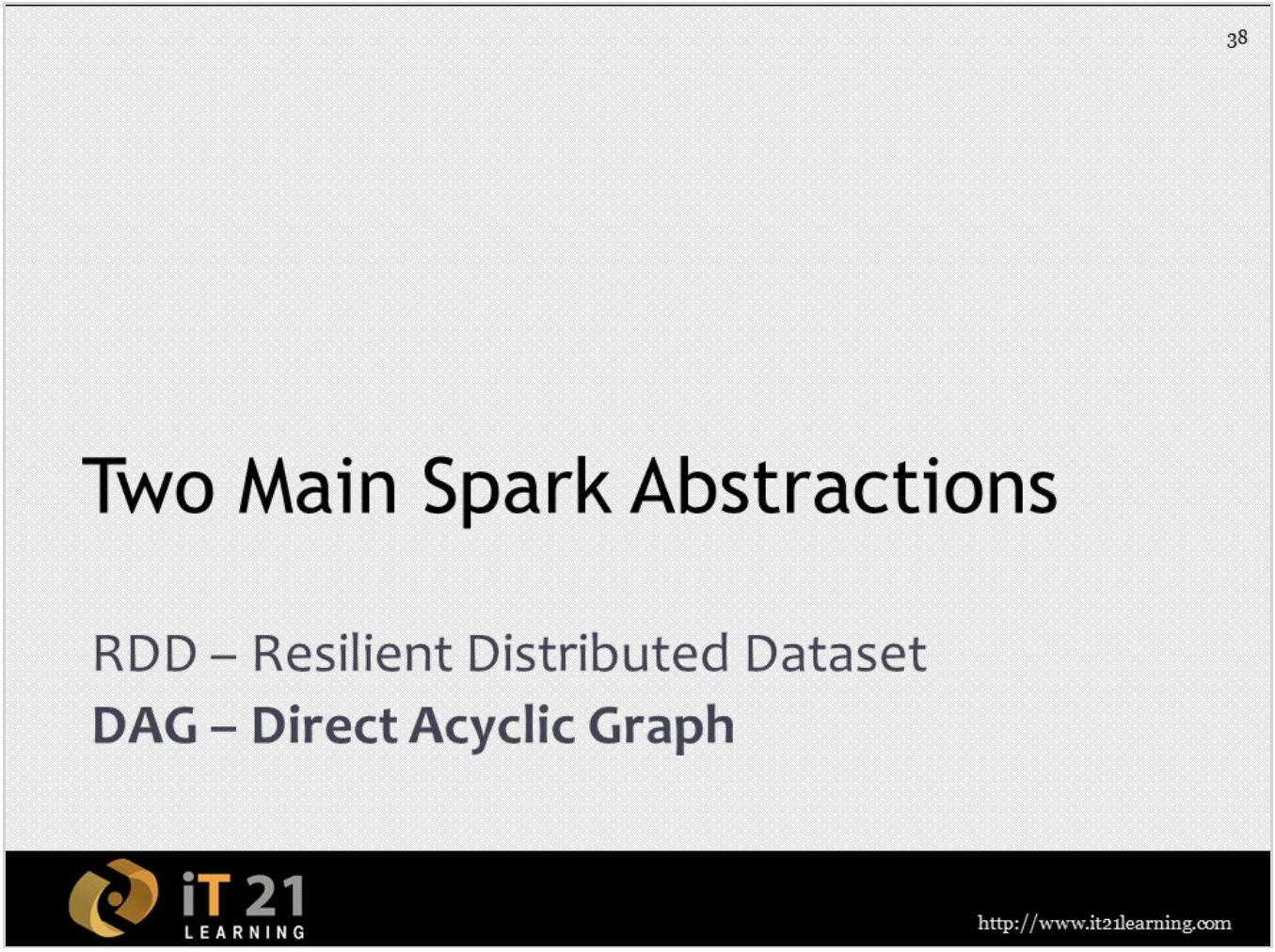
- Accumulators are variables that can only be “added” to through an *associative* operation Used to implement counters and sums, efficiently in parallel
- Spark natively supports accumulators of numeric value types and standard mutable collections, and programmers can extend for new types
- Only the driver program can read an accumulator’s value, not the tasks

```
val accum = sc.accumulator(0)
sc.parallelize(Array(1, 2, 3, 4)).foreach(x => accum += x)
accum.value
```


36


37
Revisit word count demo
Self Lab
iT 21
LEARNING
http://www.it21learning.com

37


38
Two Main Spark Abstractions
RDD – Resilient Distributed Dataset
DAG – Direct Acyclic Graph
iT 21 LEARNING
http://www.it21learning.com

38

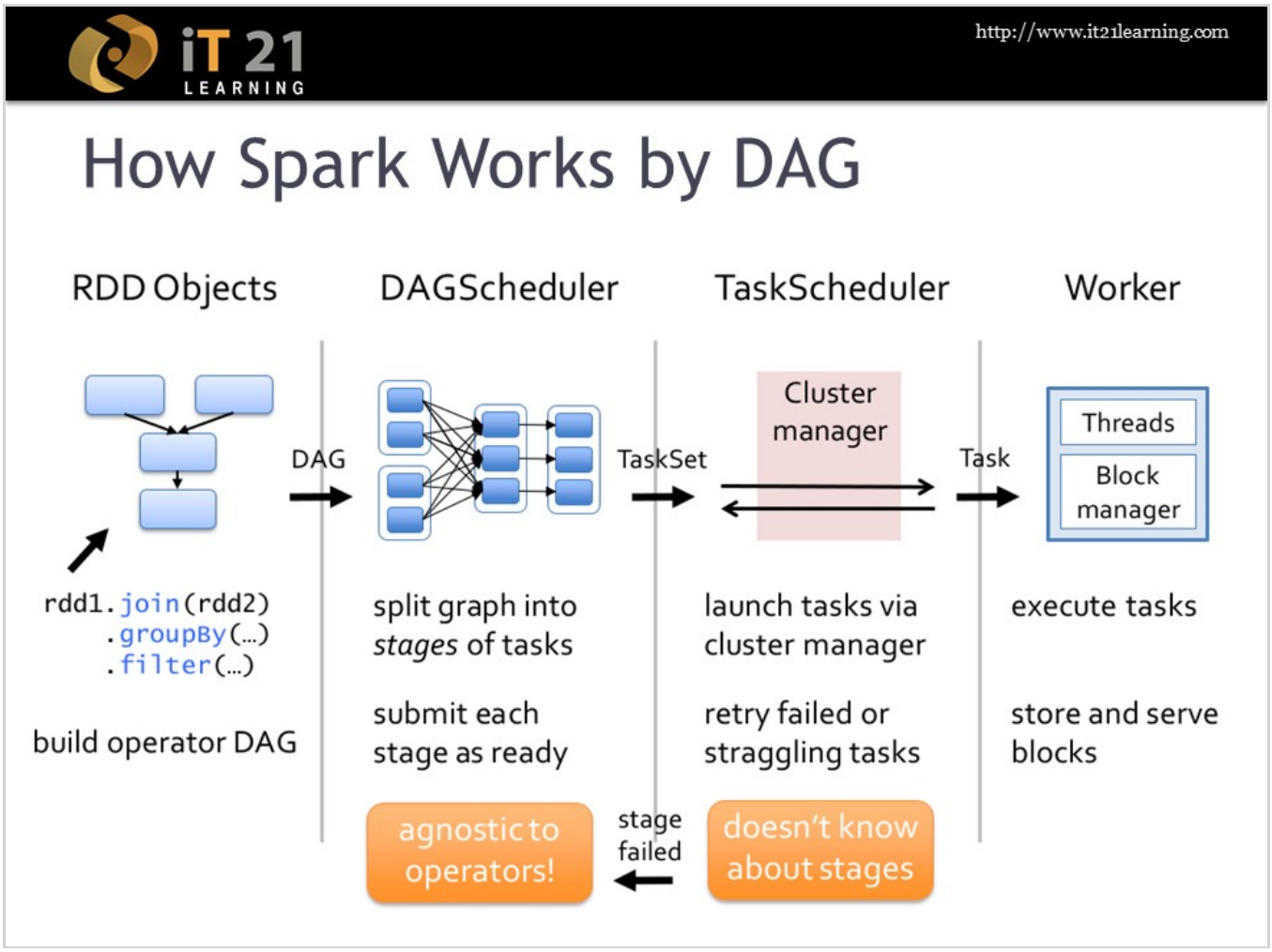
DAG

- ***Direct Acyclic Graph*** – sequence of computations performed on data

- *Node*: RDD partition
- *Edge*: Transformation on top of data
- *Acyclic*: Graph cannot return to the older partition
- *Direct*: transformation is an action that transitions data partition state (from A to B)

39

How Spark Works by DAG

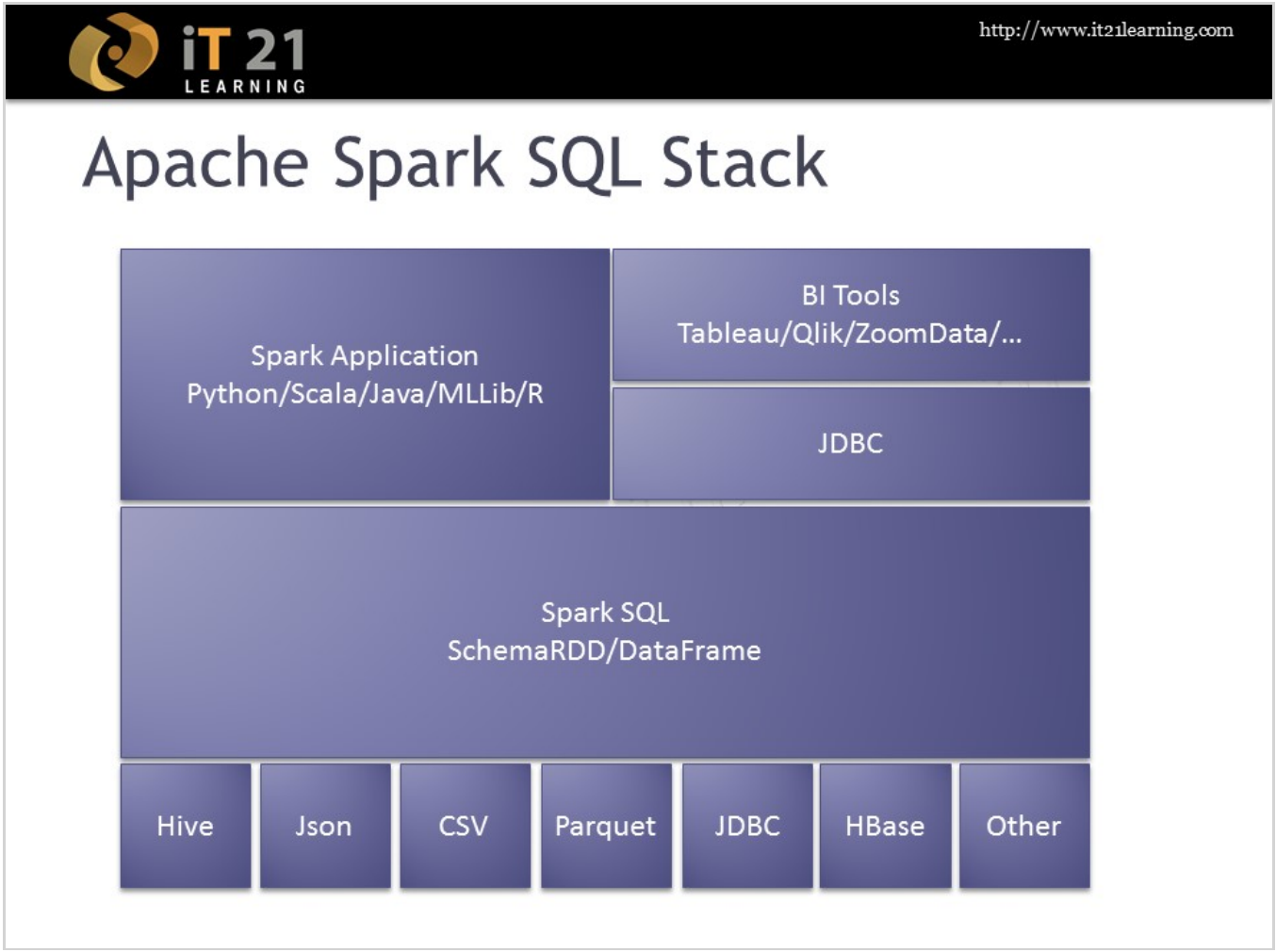


The first layer is the interpreter, Spark uses a Scala interpreter, with some modifications. As you enter your code in spark console (creating RDD's and applying operators), Spark creates a operator graph. When the user runs an action (like collect), the Graph is submitted to a DAG Scheduler. The DAG scheduler divides operator graph into (map and reduce) stages. A stage is comprised of tasks based on partitions of the input data. The DAG scheduler pipelines operators together to optimize the graph. For e.g. Many map operators can be scheduled in a single stage. This optimization is key to Sparks performance. The final result of a DAG scheduler is a set of stages. The stages are passed on to the Task Scheduler. The task scheduler launches tasks via cluster manager. (Spark Standalone/Yarn/Mesos). The task scheduler doesn't know about dependencies among stages. The Worker executes the tasks. A new JVM is started per job. The worker knows only about the code that is passed to it.


41
Spark SQL
SQL over Apache Spark
iT 21
LEARNING
http://www.it21learning.com

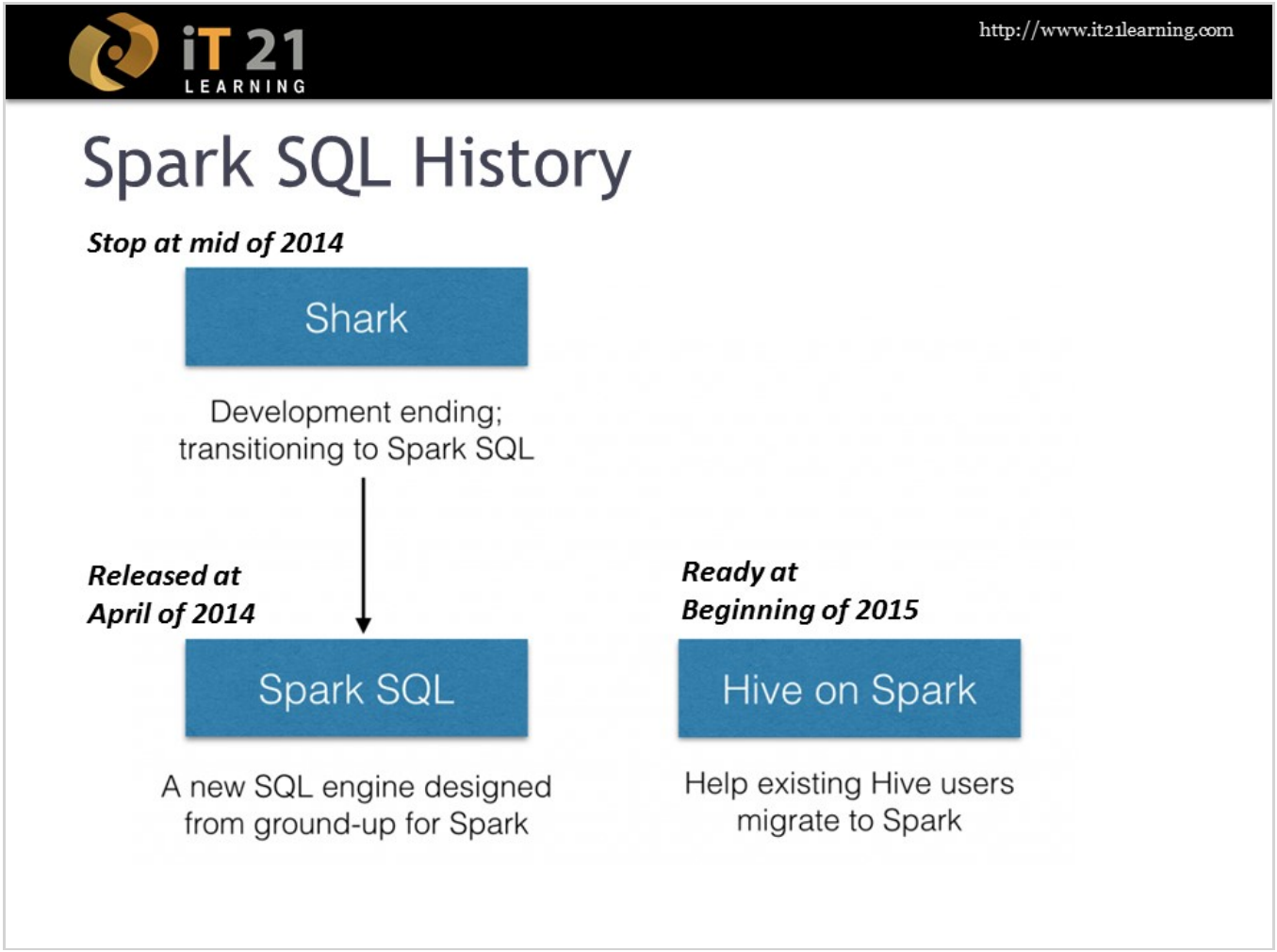
41

Apache Spark SQL Stack



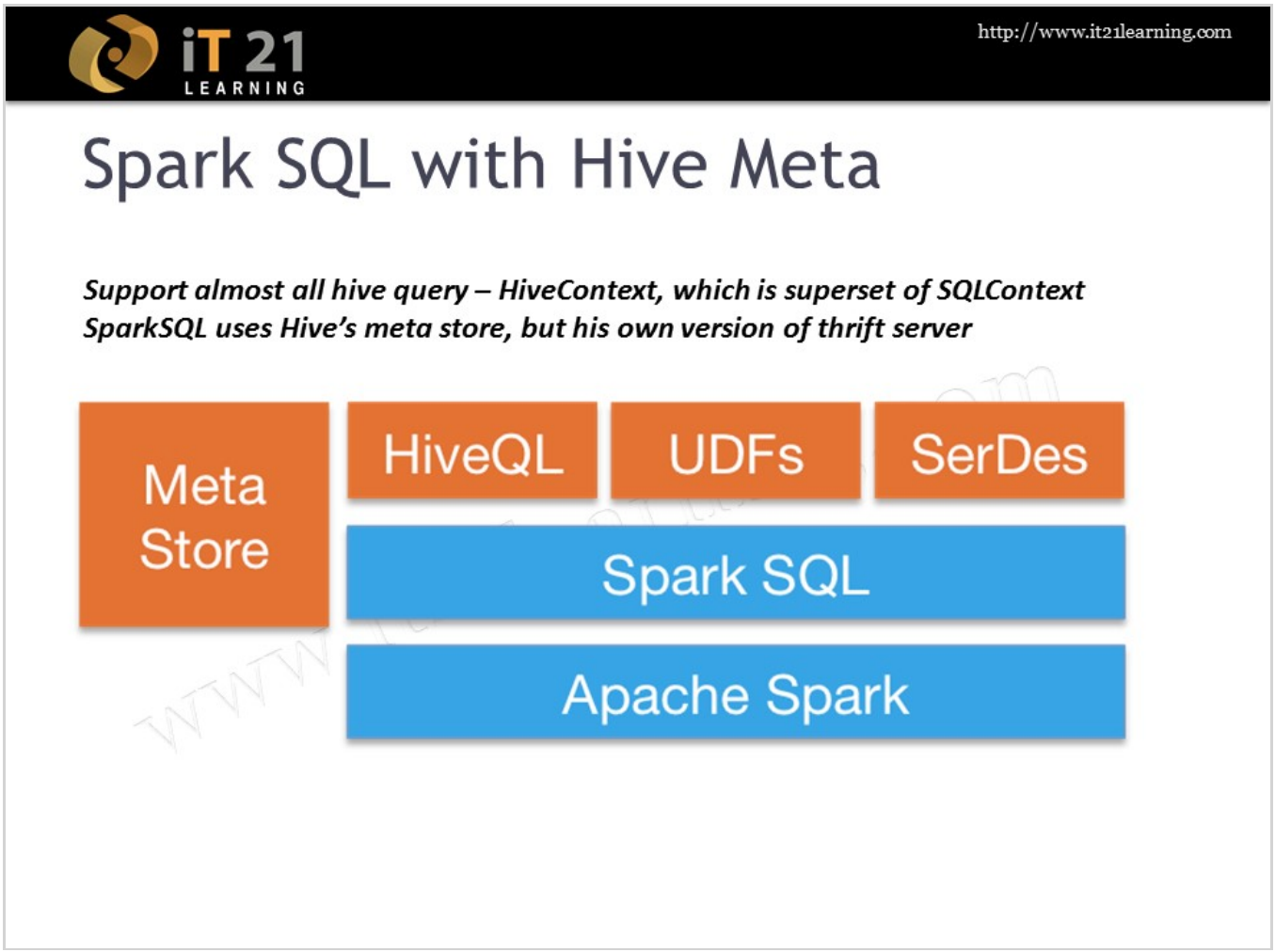
42

Spark SQL History



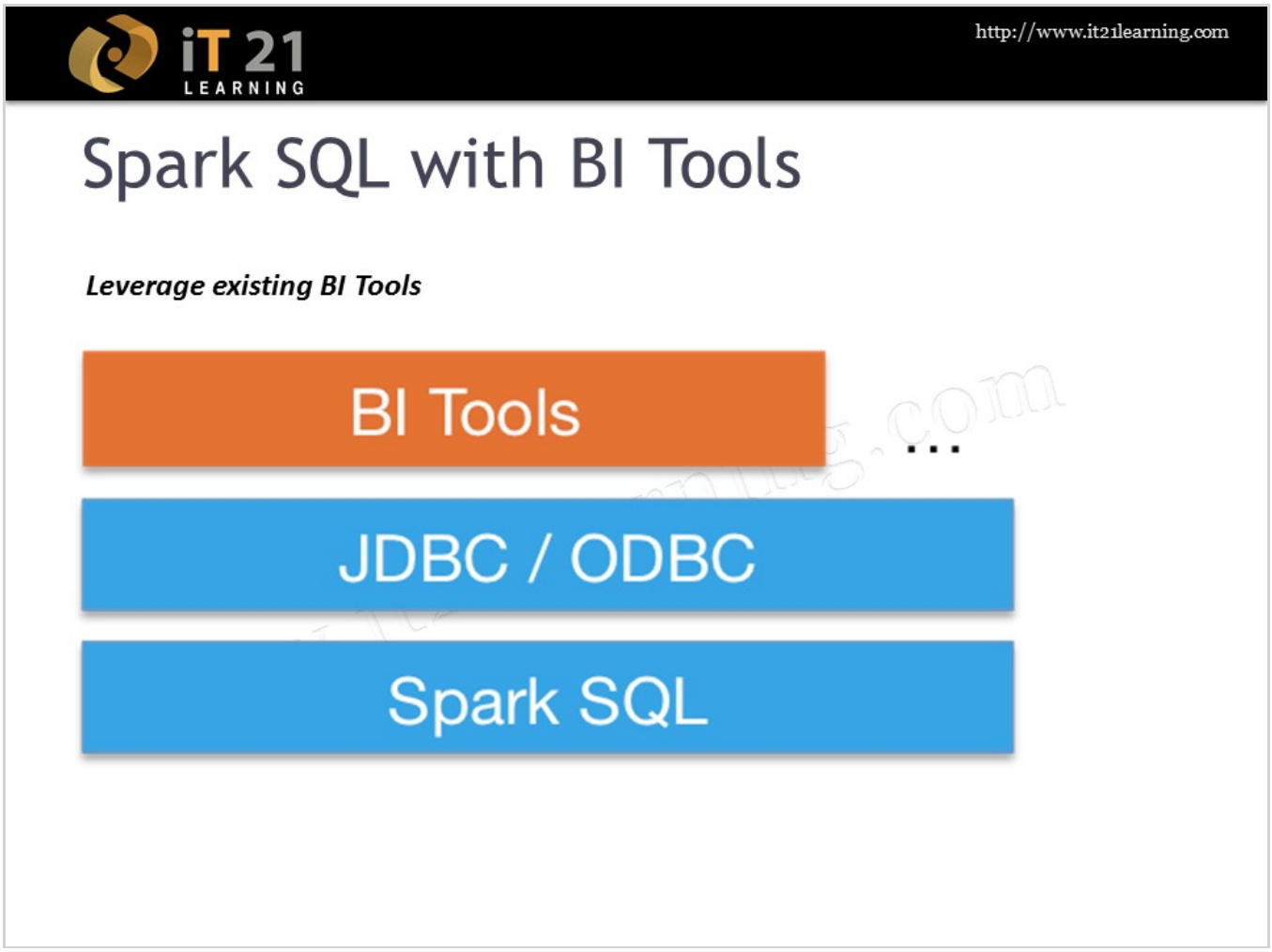
43

Spark SQL with Hive Meta



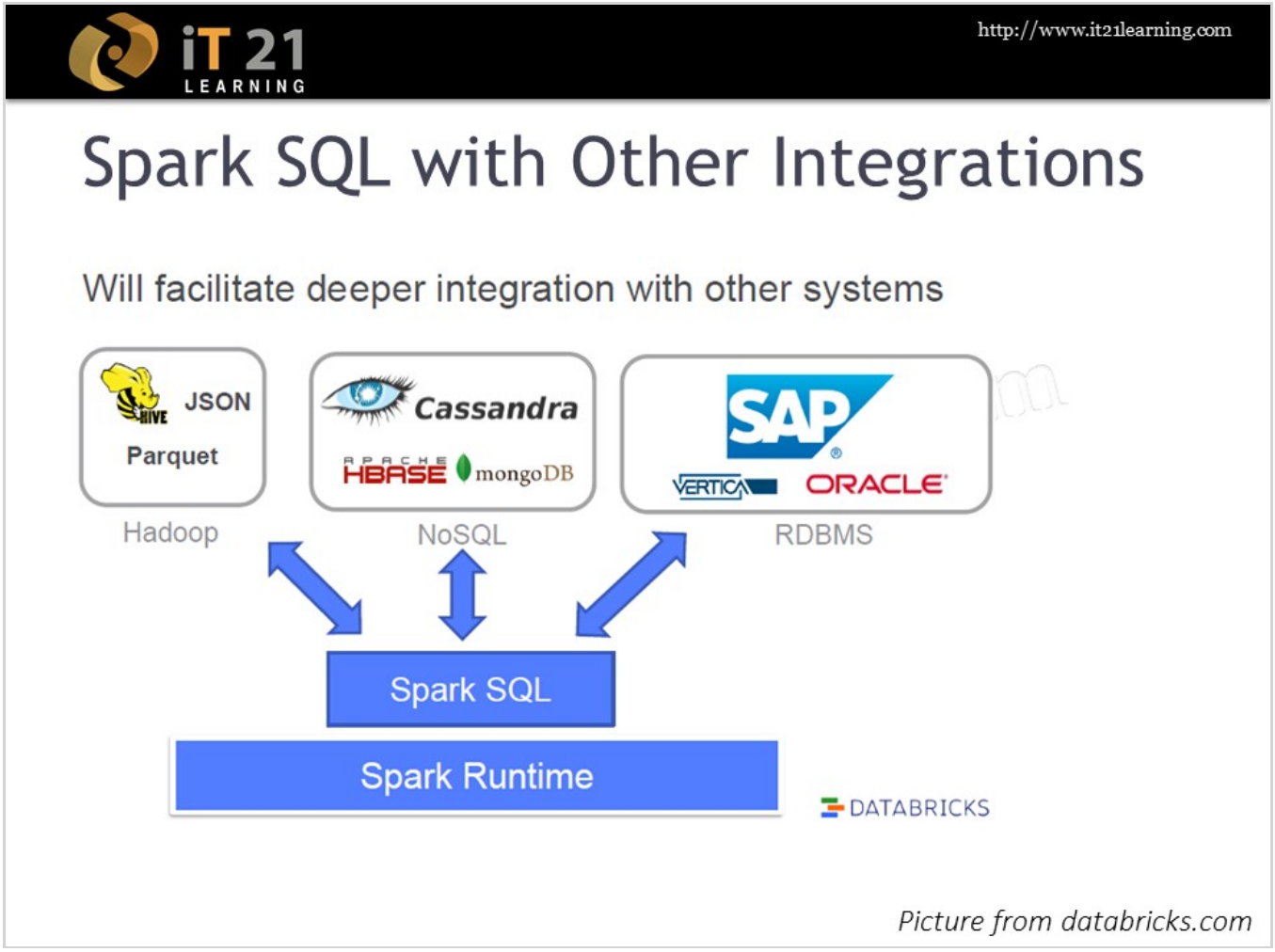
44

Spark SQL with BI Tools



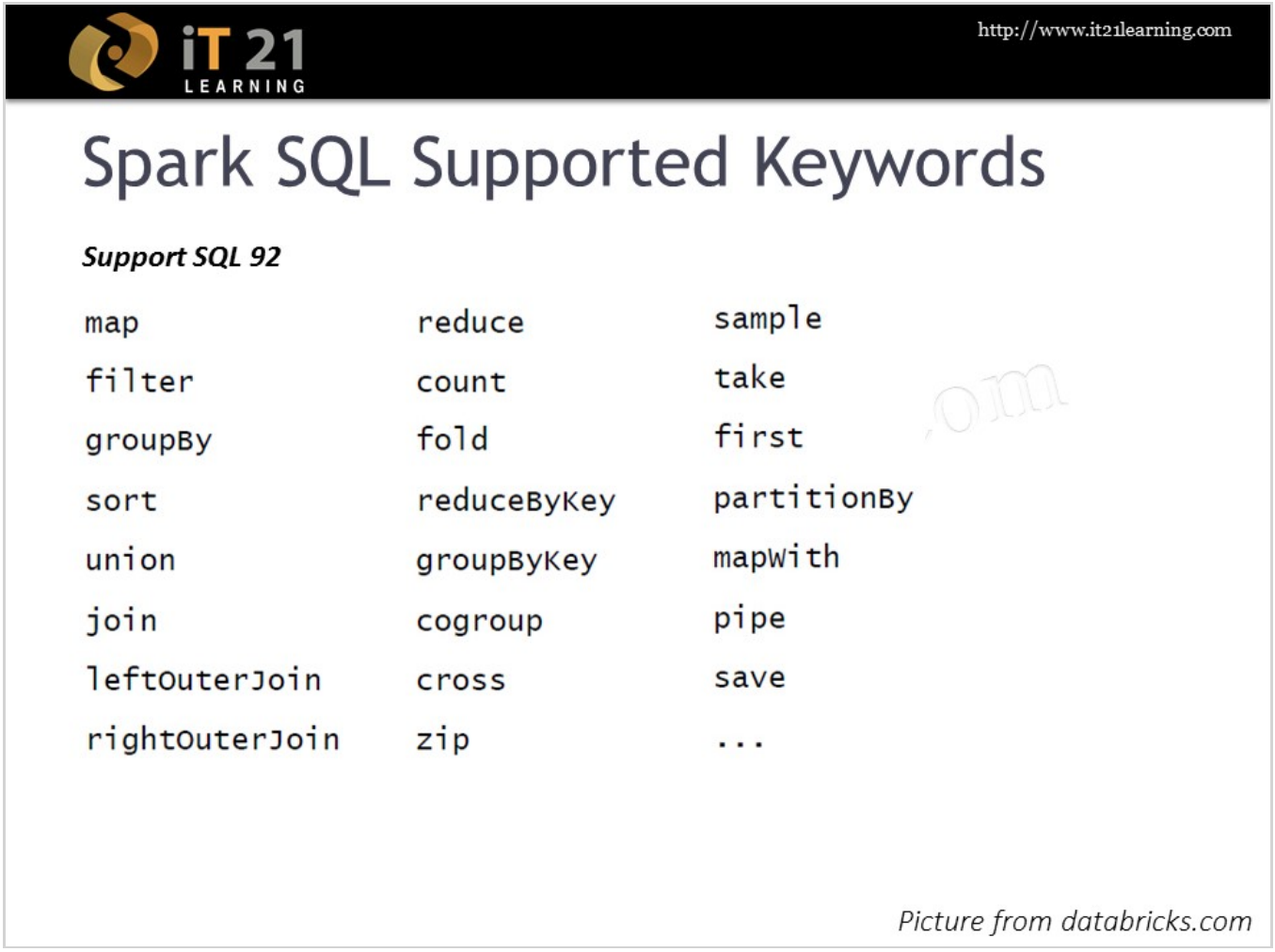
45

Spark SQL with Other Integrations



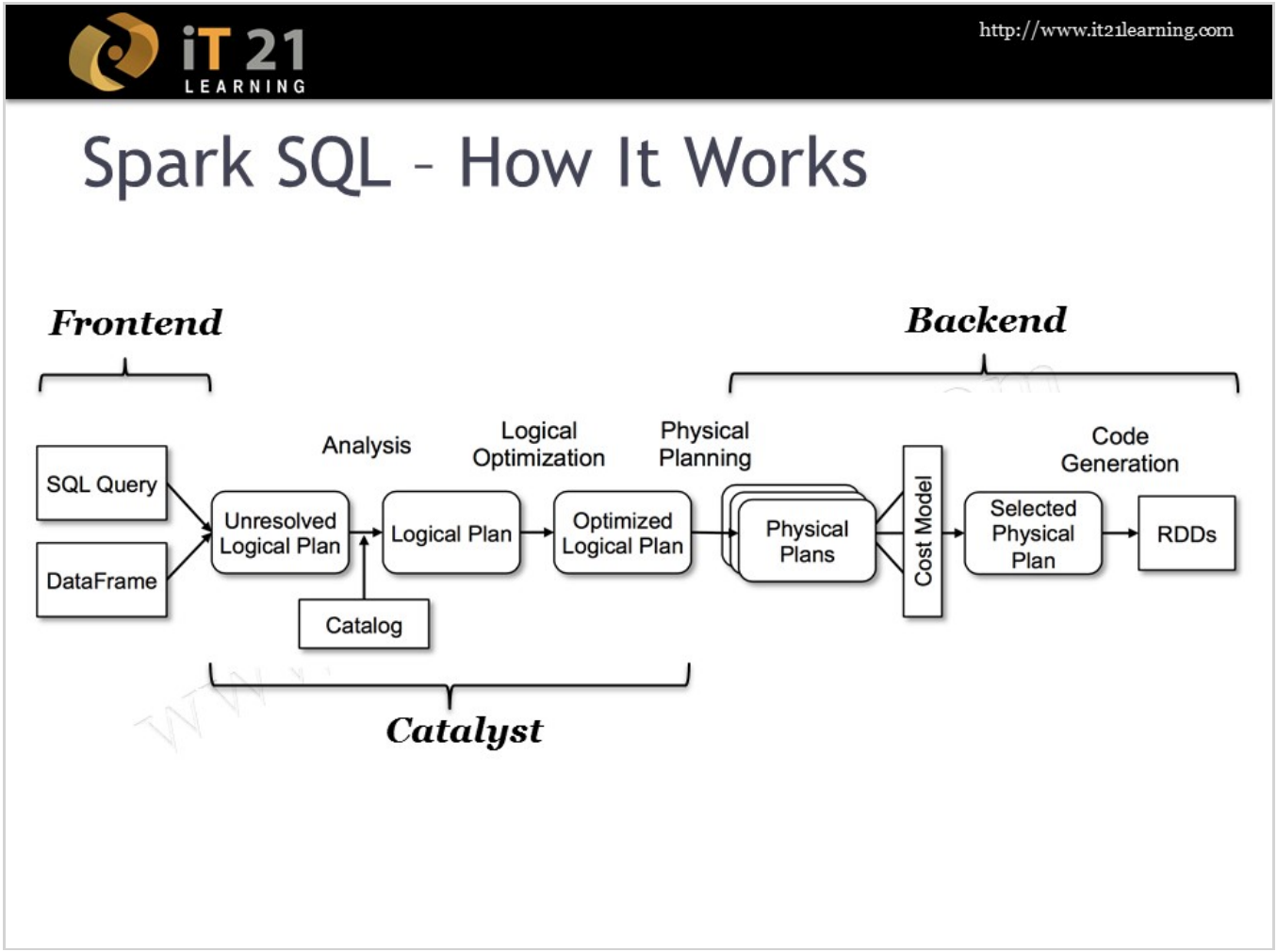
46

Spark SQL Supported Keywords



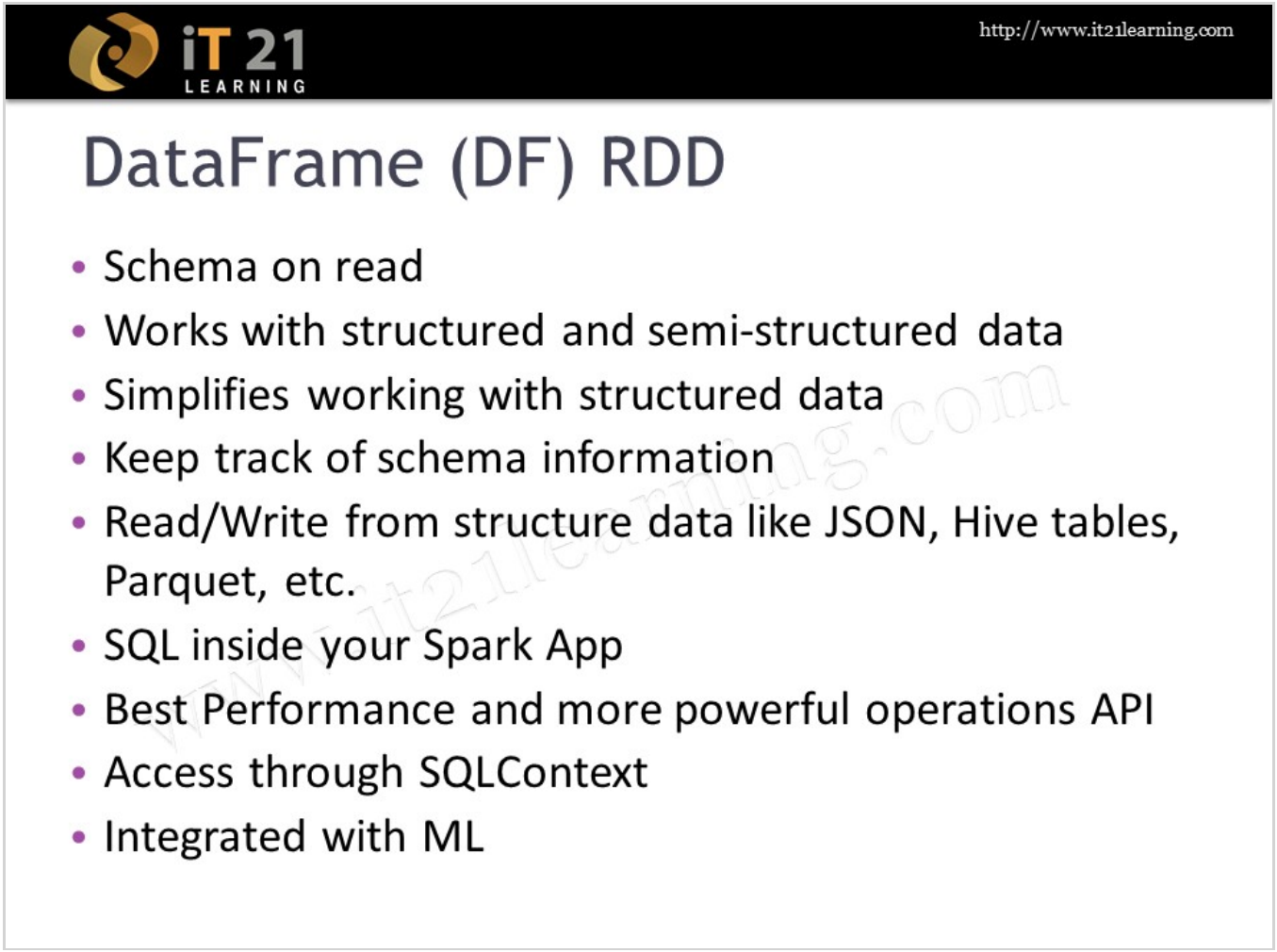
47

Spark SQL – How It Works



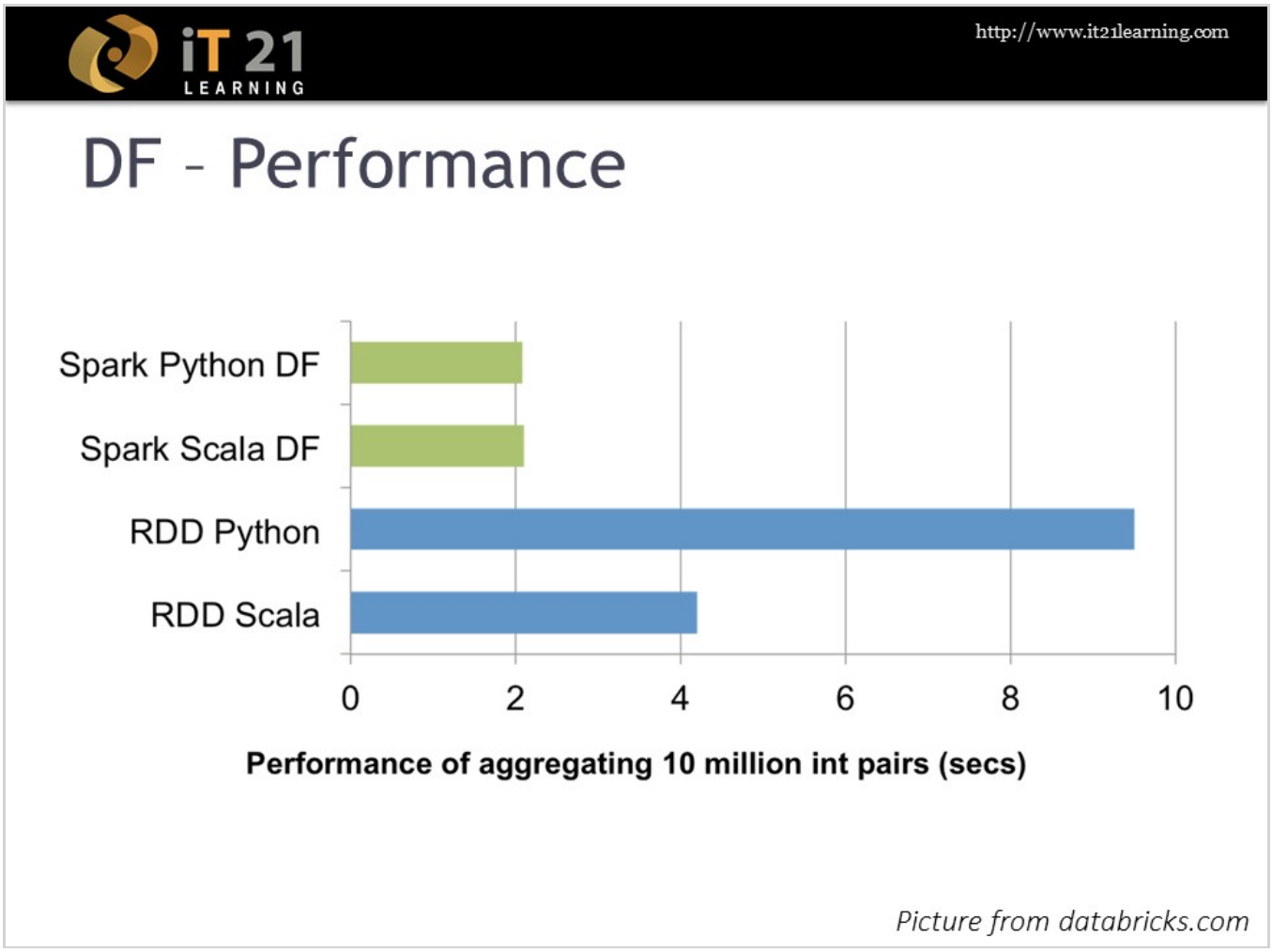
Optimizing query planning for Spark Takes Dataframe operations and 'compiles' them down to RDD operations Often faster than writing RDD code manually Use Dataframes whenever possible (v1.4+)

DataFrame (DF) RDD



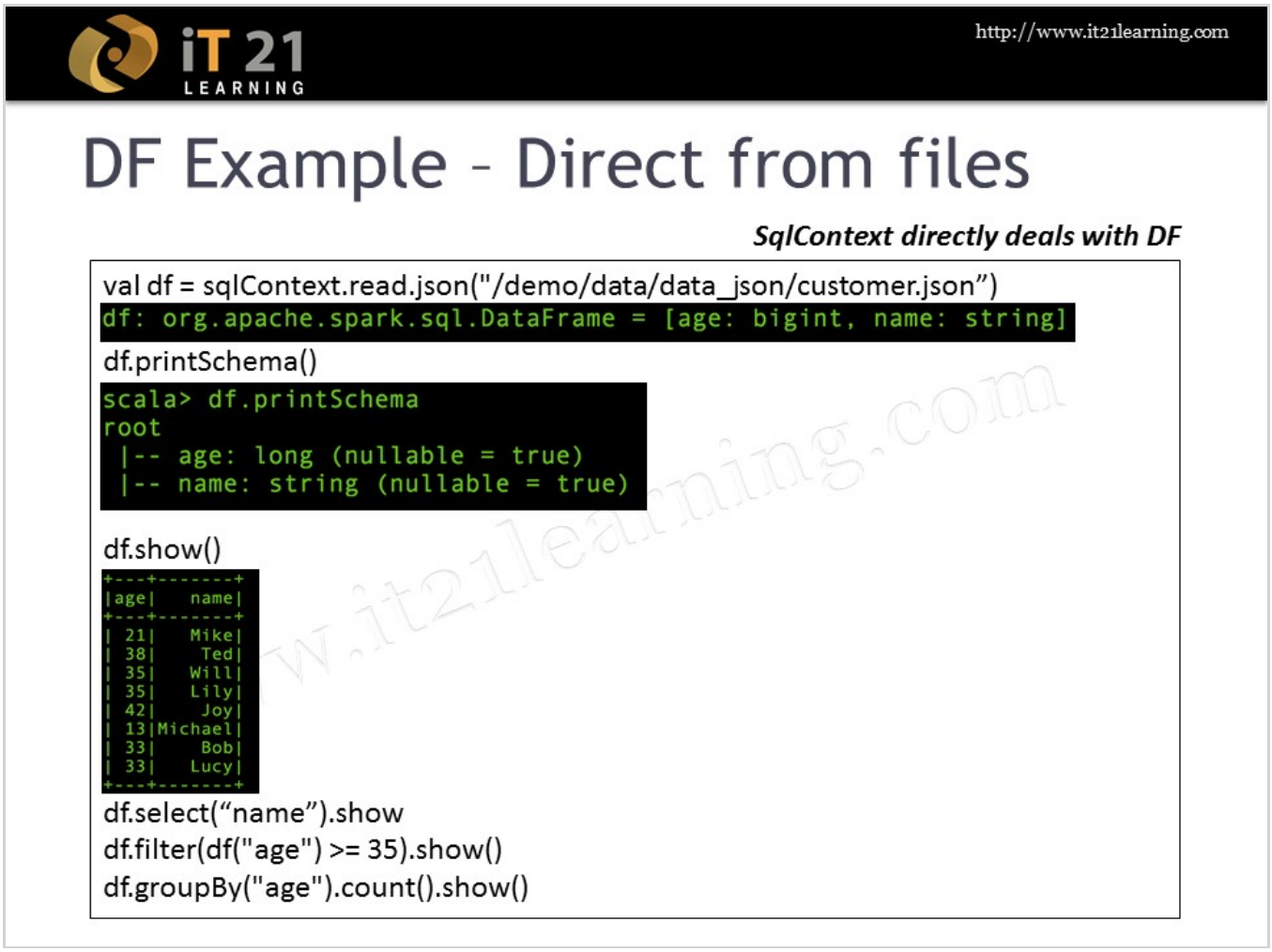
Works with structured and semi-structured dataDataFrame simplifies working with structured dataRead/Write from structure data like JSON, Hive tables, Parquet, etc. SQL inside your Spark AppBest Performance and more powerful operations API

DF – Performance



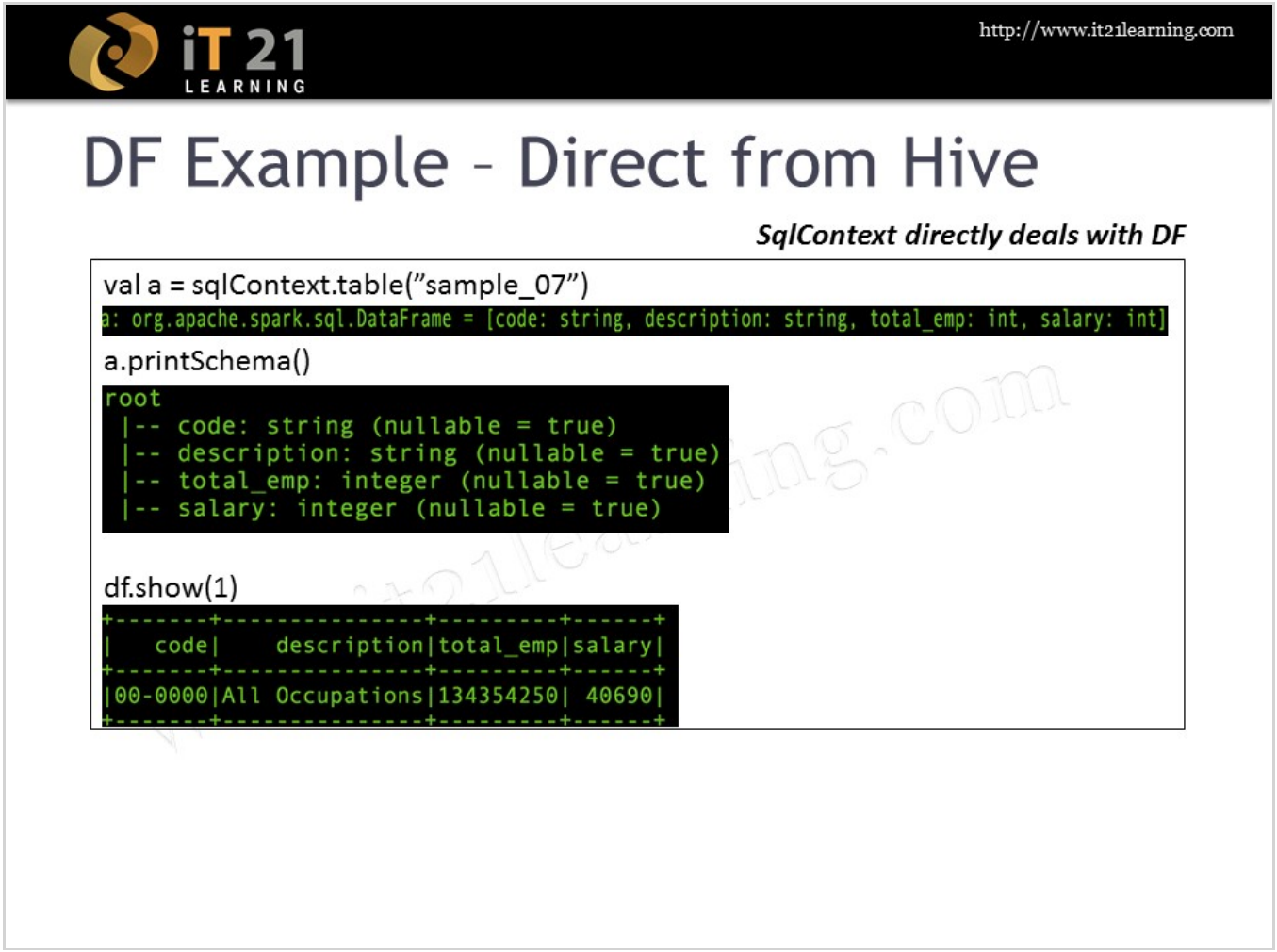
50

DF Example – Direct from files



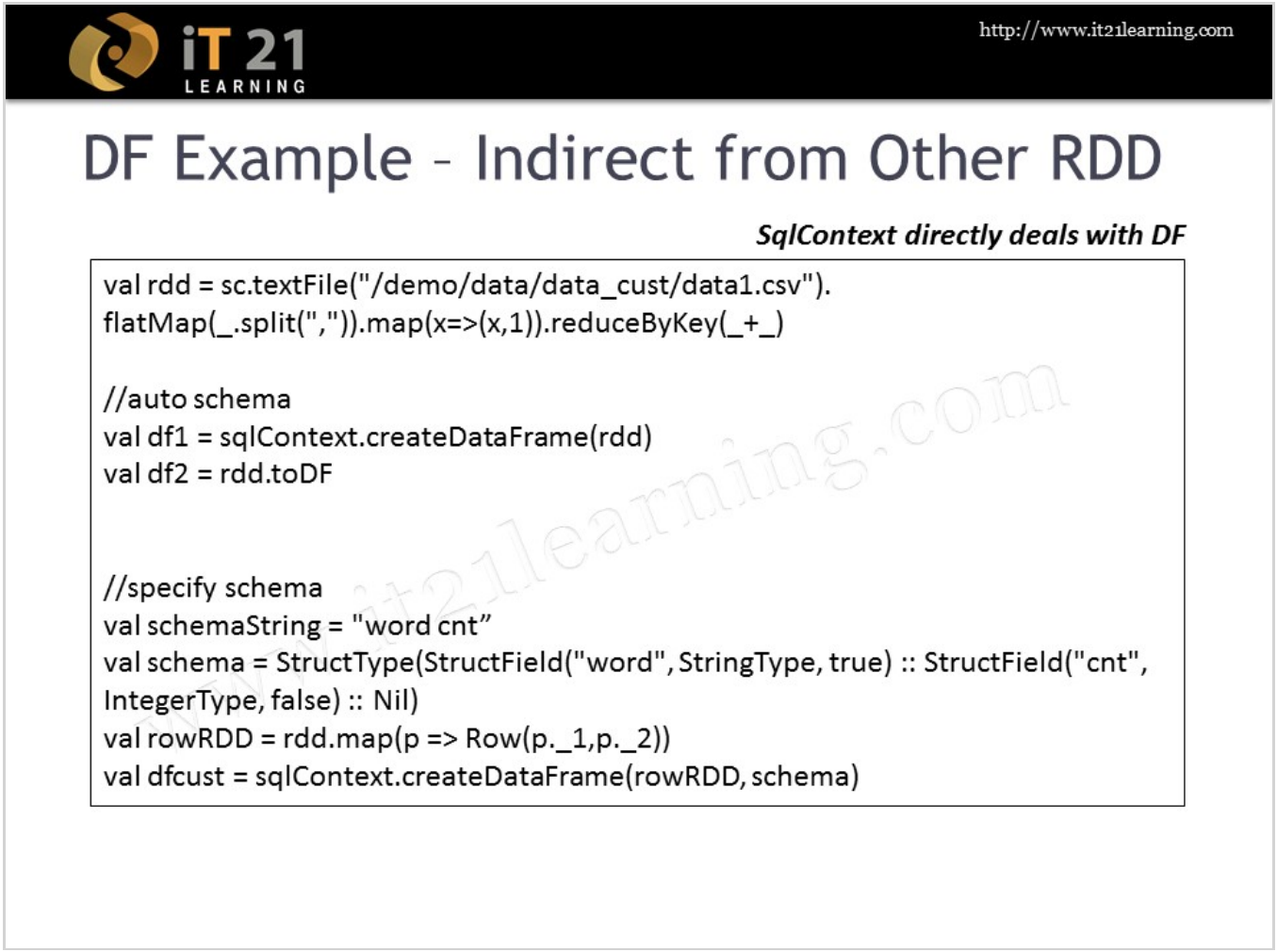
51

DF Example – Direct from Hive



52

DF Example – Indirect from Other RDD

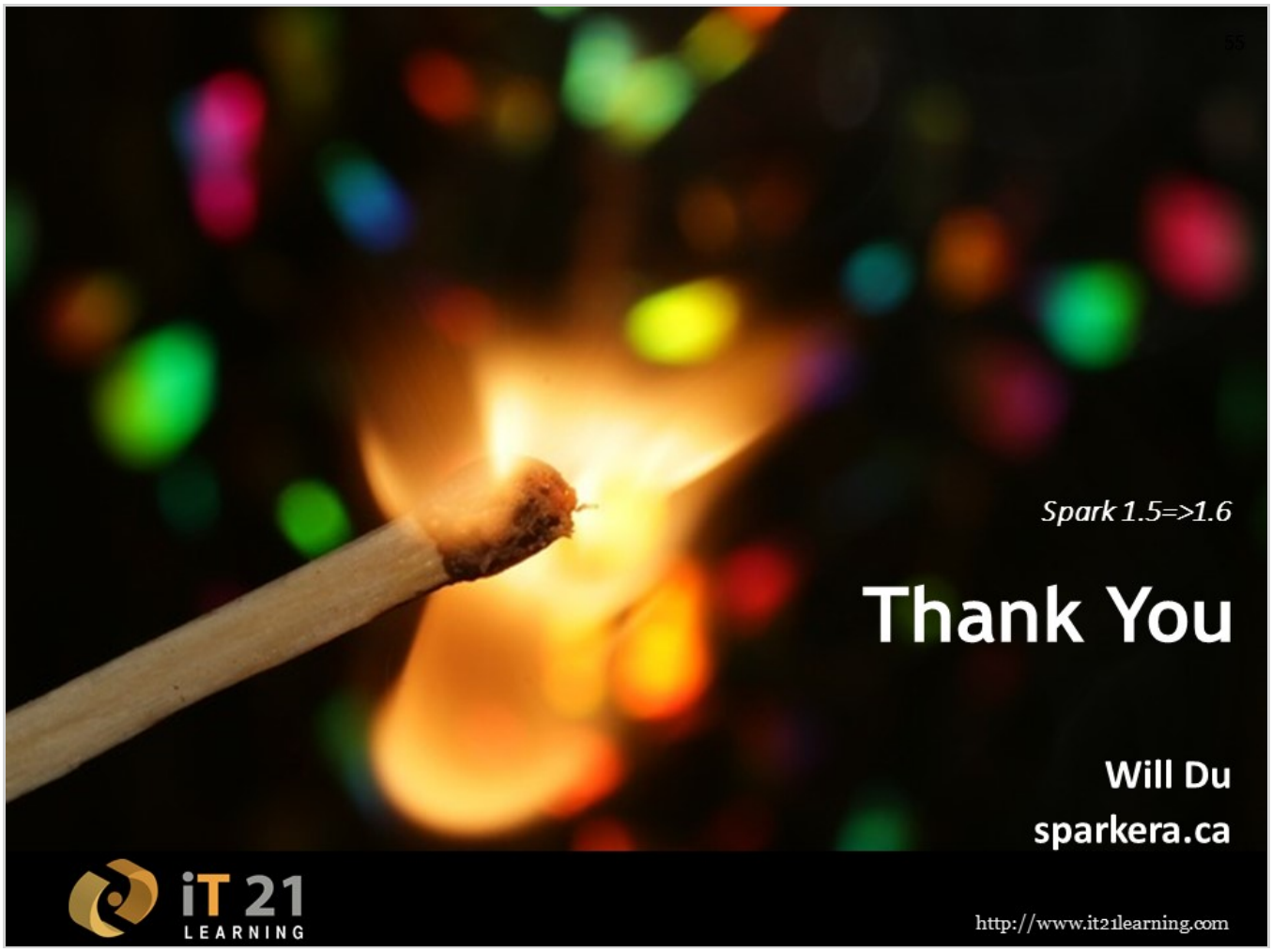


53


54
Spark SQL Shell Demo
Self Practice
iT 21
LEARNING
http://www.it21learning.com

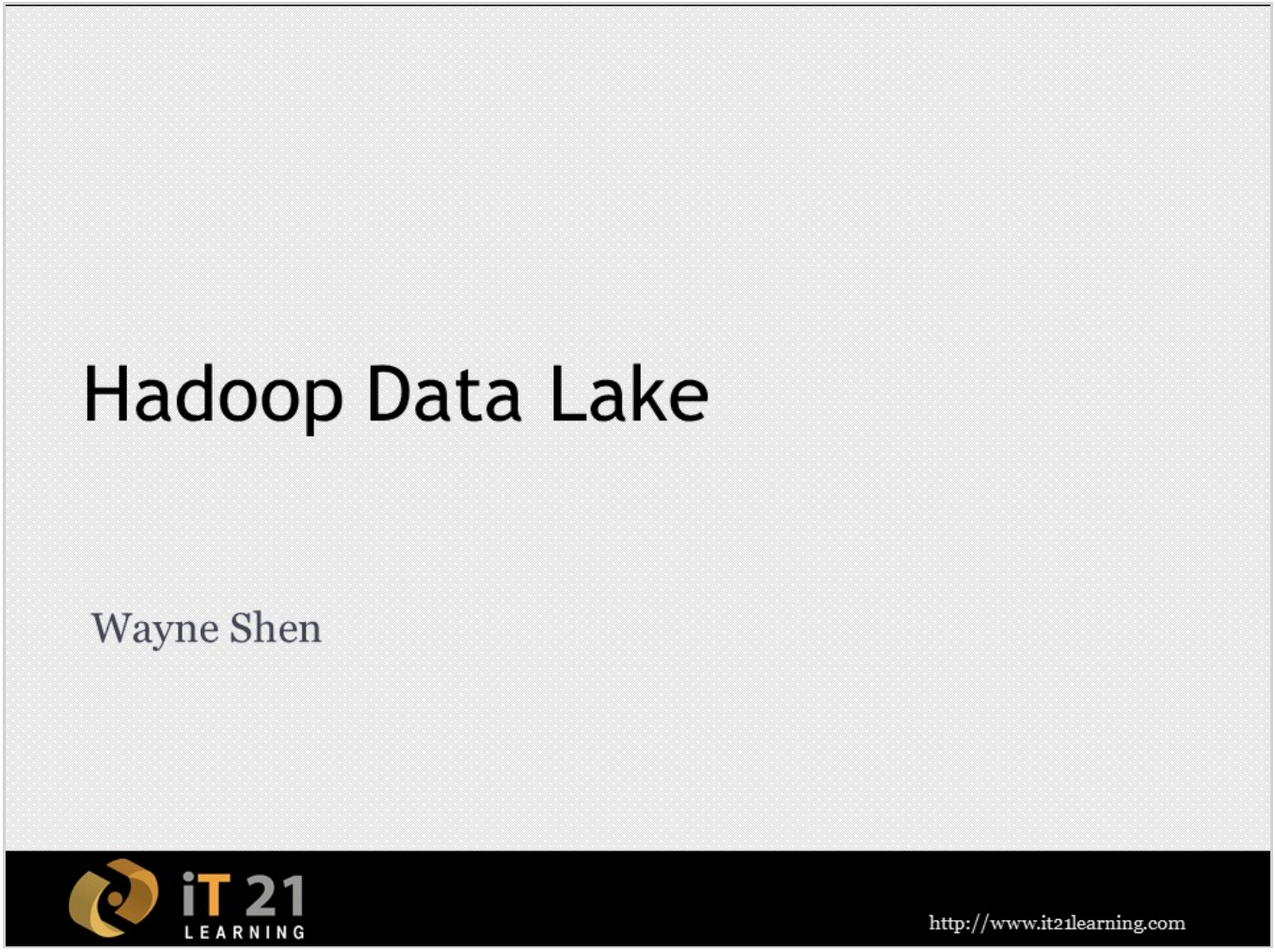
54

Thank You

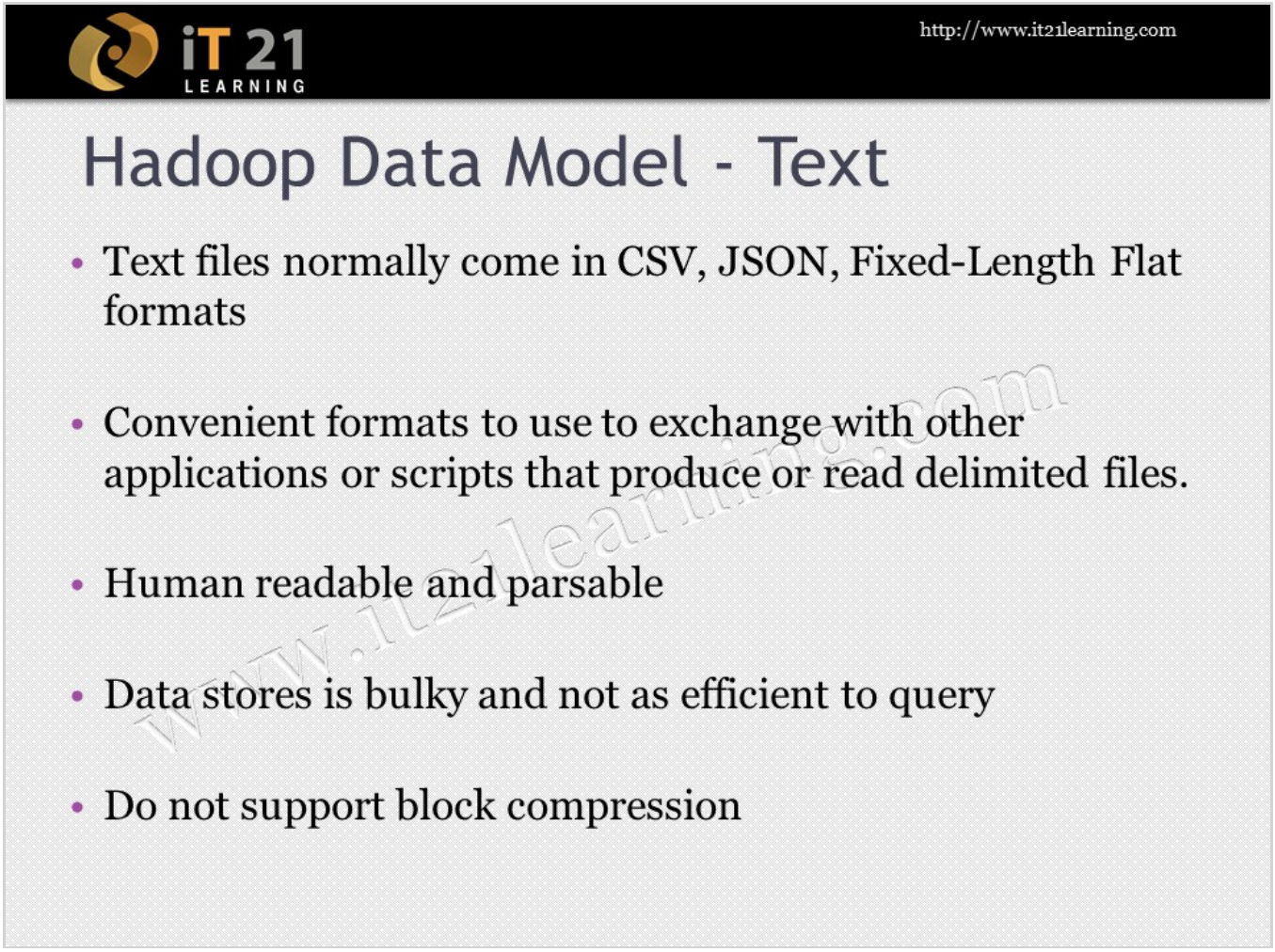

55
Spark 1.5=>1.6
Thank You
Will Du
sparkera.ca
iT 21
LEARNING
http://www.it21learning.com

55

Hadoop Data Model & Application Architecture

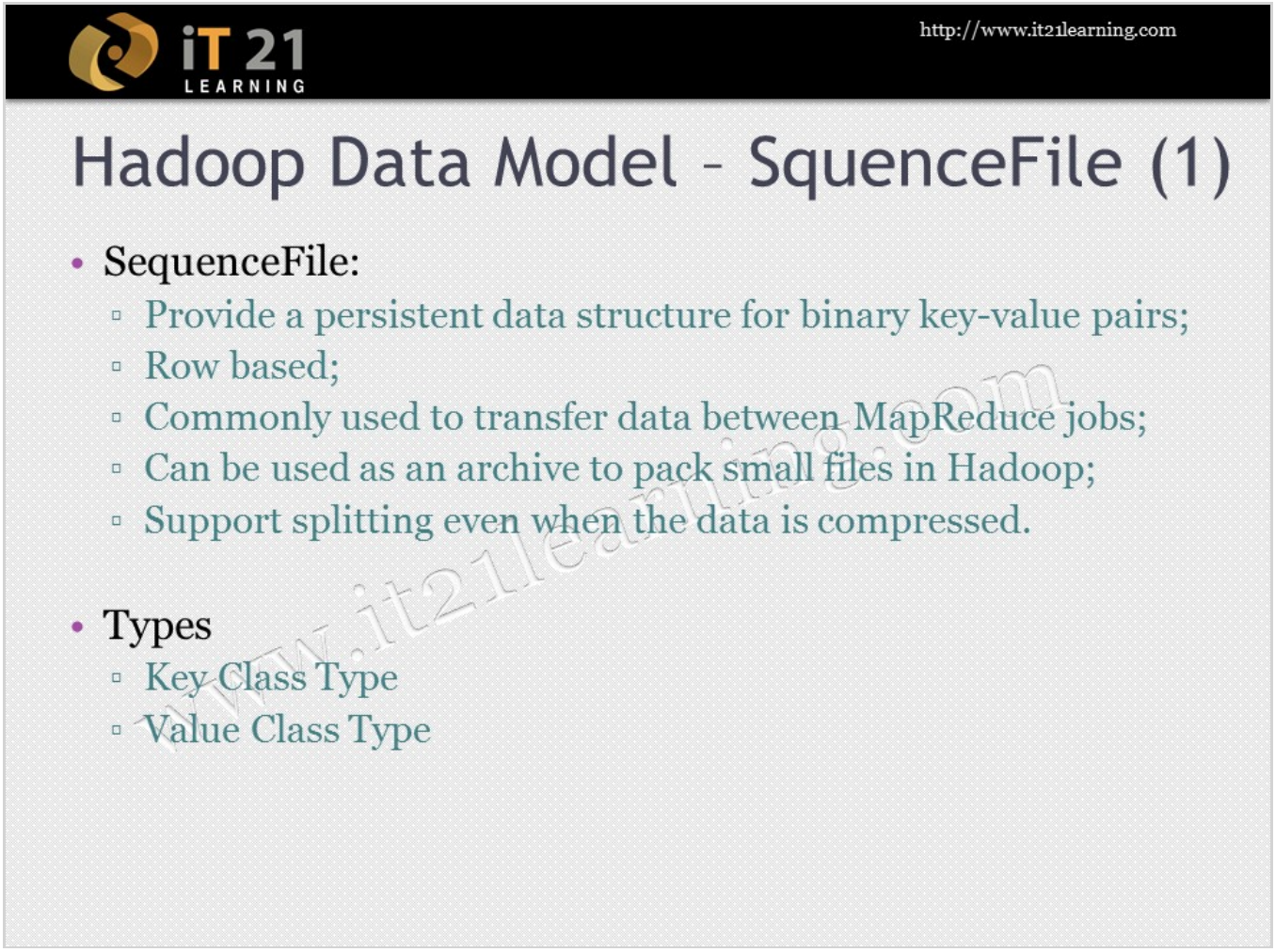


http://www.slideshare.net/StampedeCon/choosing-an-hdfs-data-storage-format-avro-vs-parquet-and-more-stampedecon-2015

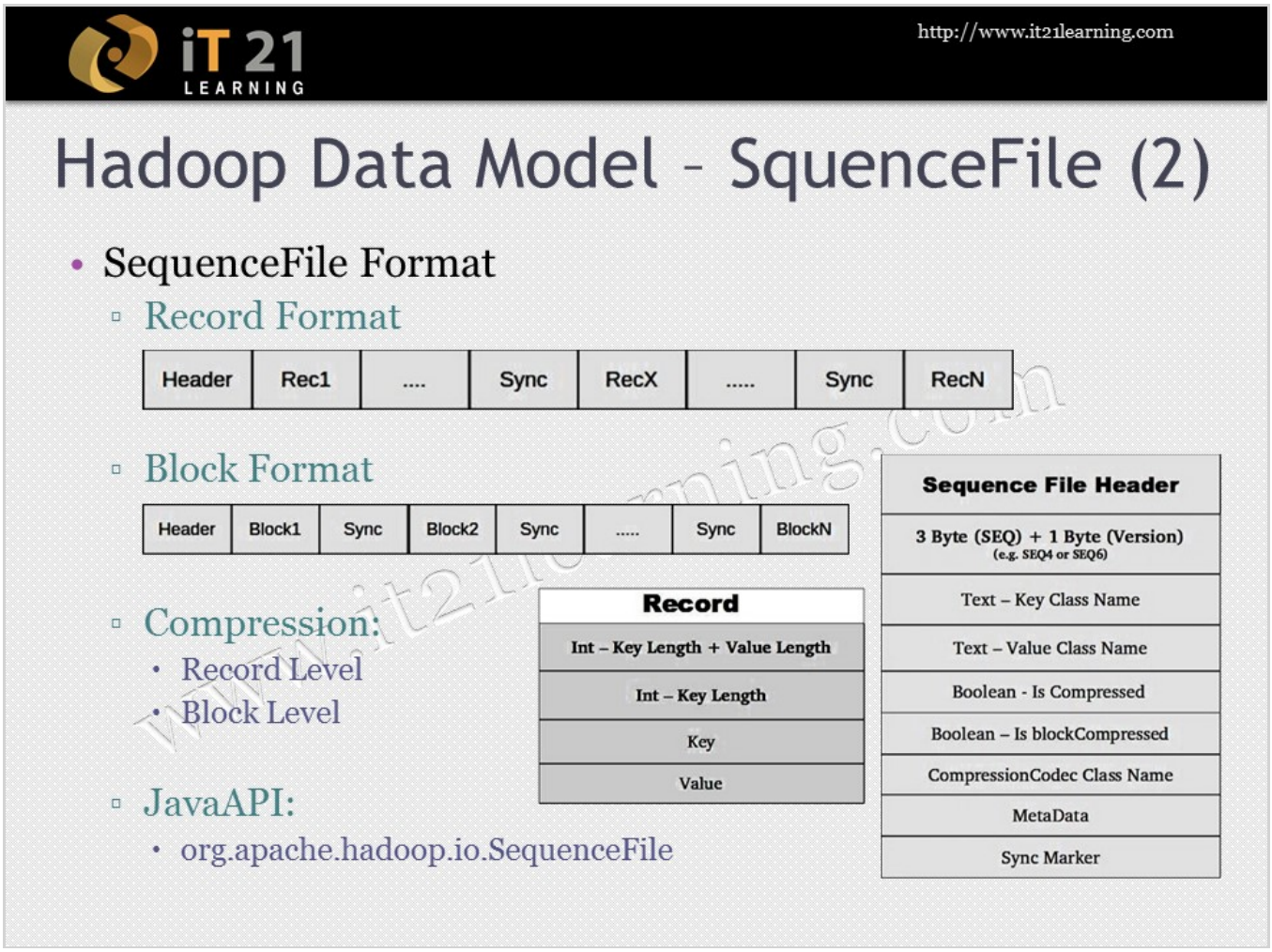

iT 21 LEARNING
http://www.it21learning.com
Hadoop Data Model - Text
• Text files normally come in CSV, JSON, Fixed-Length Flat formats
• Convenient formats to use to exchange with other applications or scripts that produce or read delimited files.
• Human readable and parsable
• Data stores is bulky and not as efficient to query
• Do not support block compression
www.it21learning.com

2

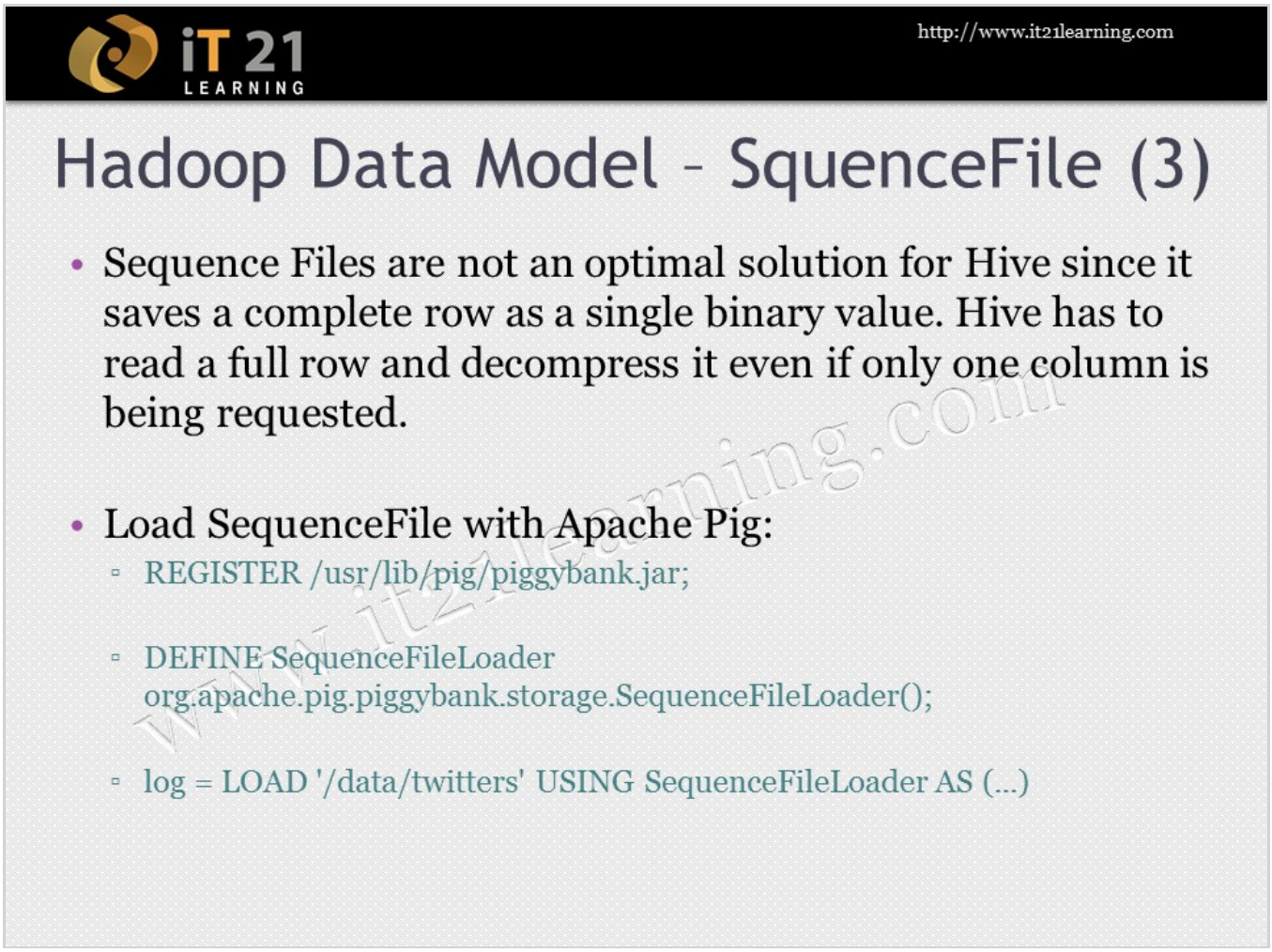
Hadoop Data Model – SquenceFile (1)



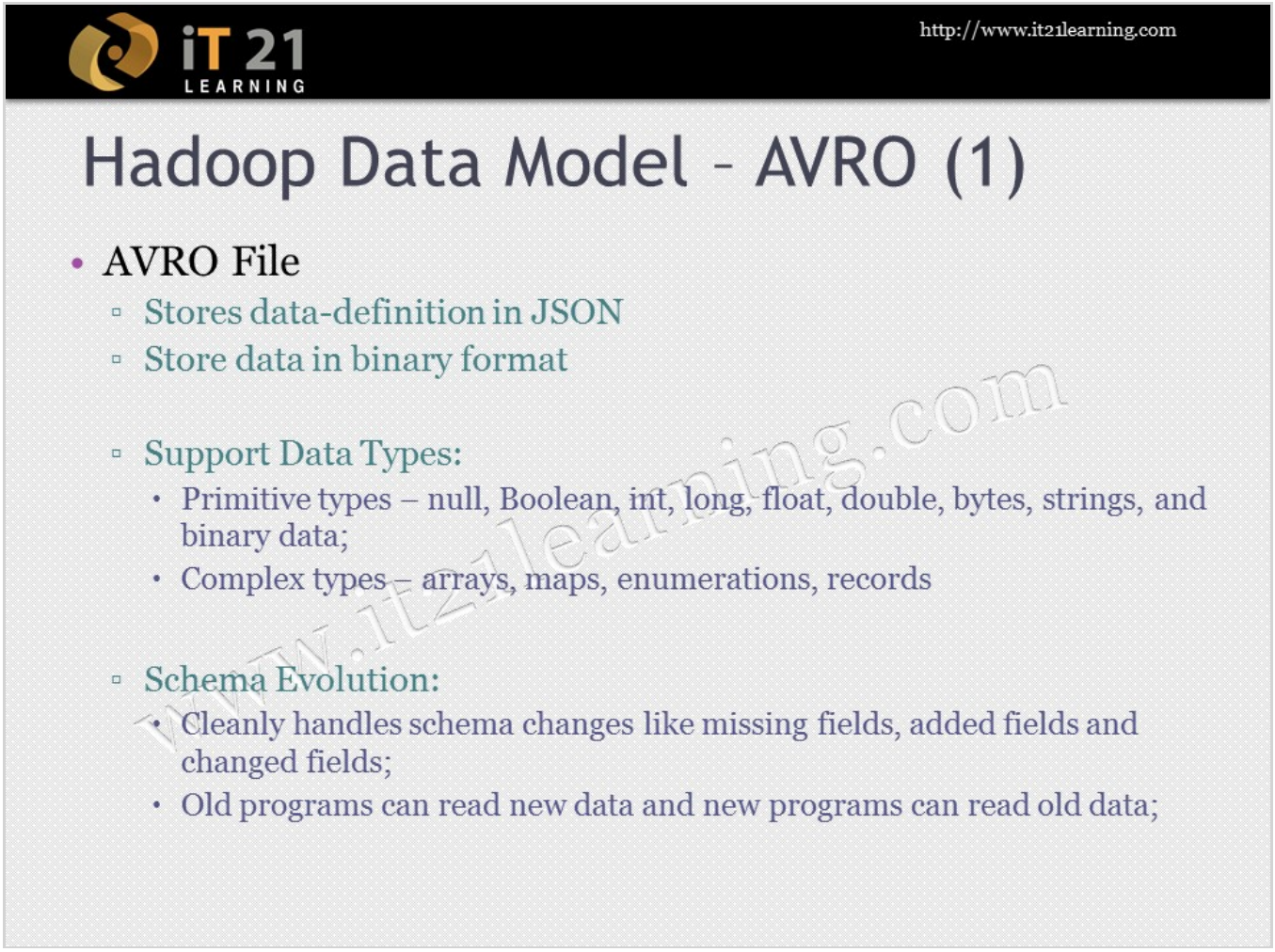
3


iT 21 LEARNING
http://www.it21learning.com
Hadoop Data Model - SquenceFile (2)
SequenceFile Format
Record Format
Header
Rec1
....
Sync
RecX
.....
Sync
RecN
Block Format
Header
Block1
Sync
Block2
Sync
.....
Sync
BlockN
Compression:
Record Level
Block Level
JavaAPI:
org.apache.hadoop.io.SequenceFile
Record
Int – Key Length + Value Length
Int – Key Length
Key
Value
Sequence File Header
3 Byte (SEQ) + 1 Byte (Version)
(e.g. SEQ4 or SEQ6)
Text – Key Class Name
Text – Value Class Name
Boolean - Is Compressed
Boolean – Is blockCompressed
CompressionCodec Class Name
MetaData
Sync Marker

4

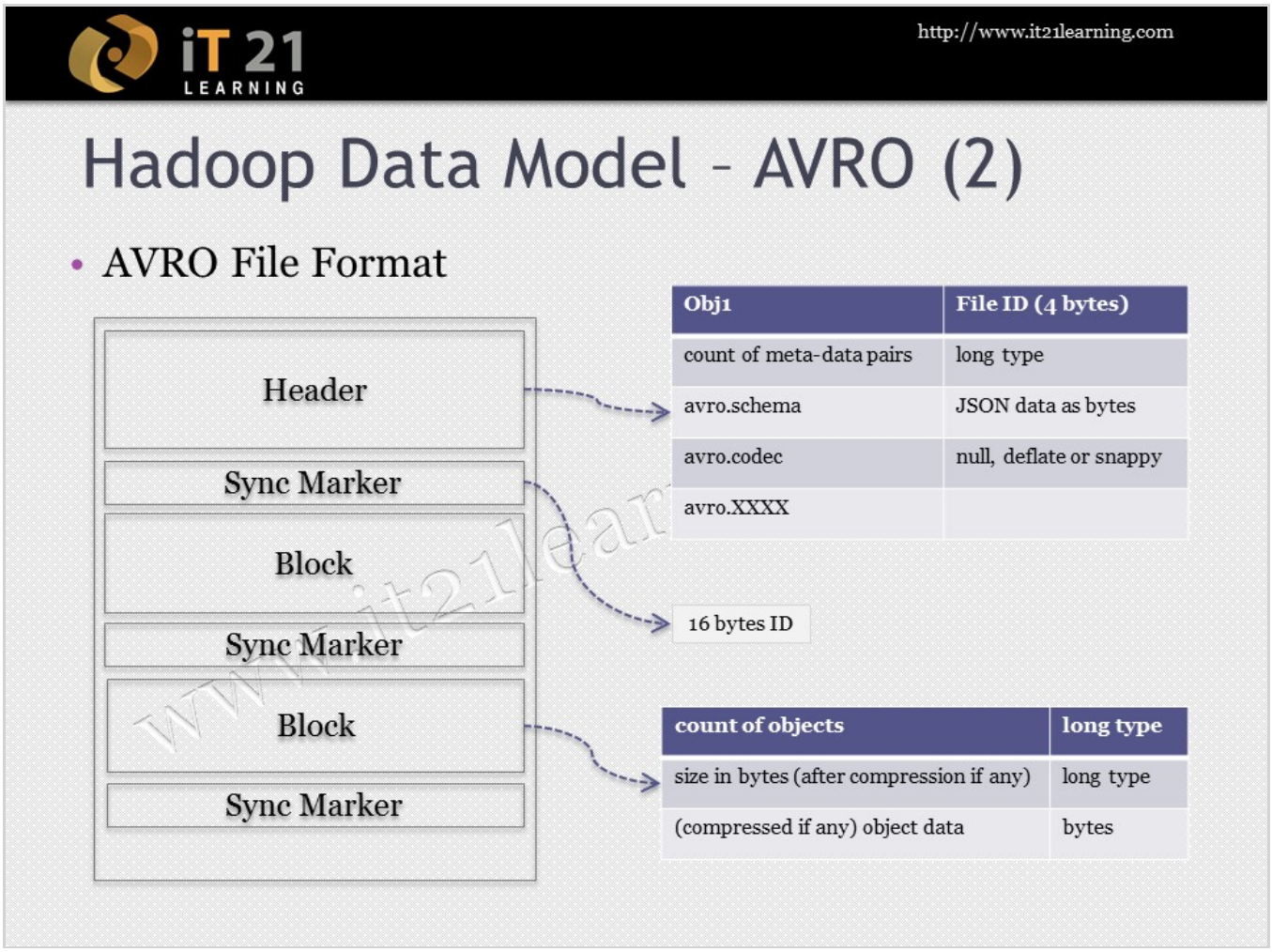

iT 21 LEARNING
http://www.it21learning.com
Hadoop Data Model – SquenceFile (3)
• Sequence Files are not an optimal solution for Hive since it saves a complete row as a single binary value. Hive has to read a full row and decompress it even if only one column is being requested.
• Load SequenceFile with Apache Pig:
▫ REGISTER /usr/lib/pig/piggybank.jar;
▫ DEFINE SequenceFileLoader org.apache.pig.piggybank.storage.SequenceFileLoader();
▫ log = LOAD '/data/twitters' USING SequenceFileLoader AS (...)
www.it21learning.com

5

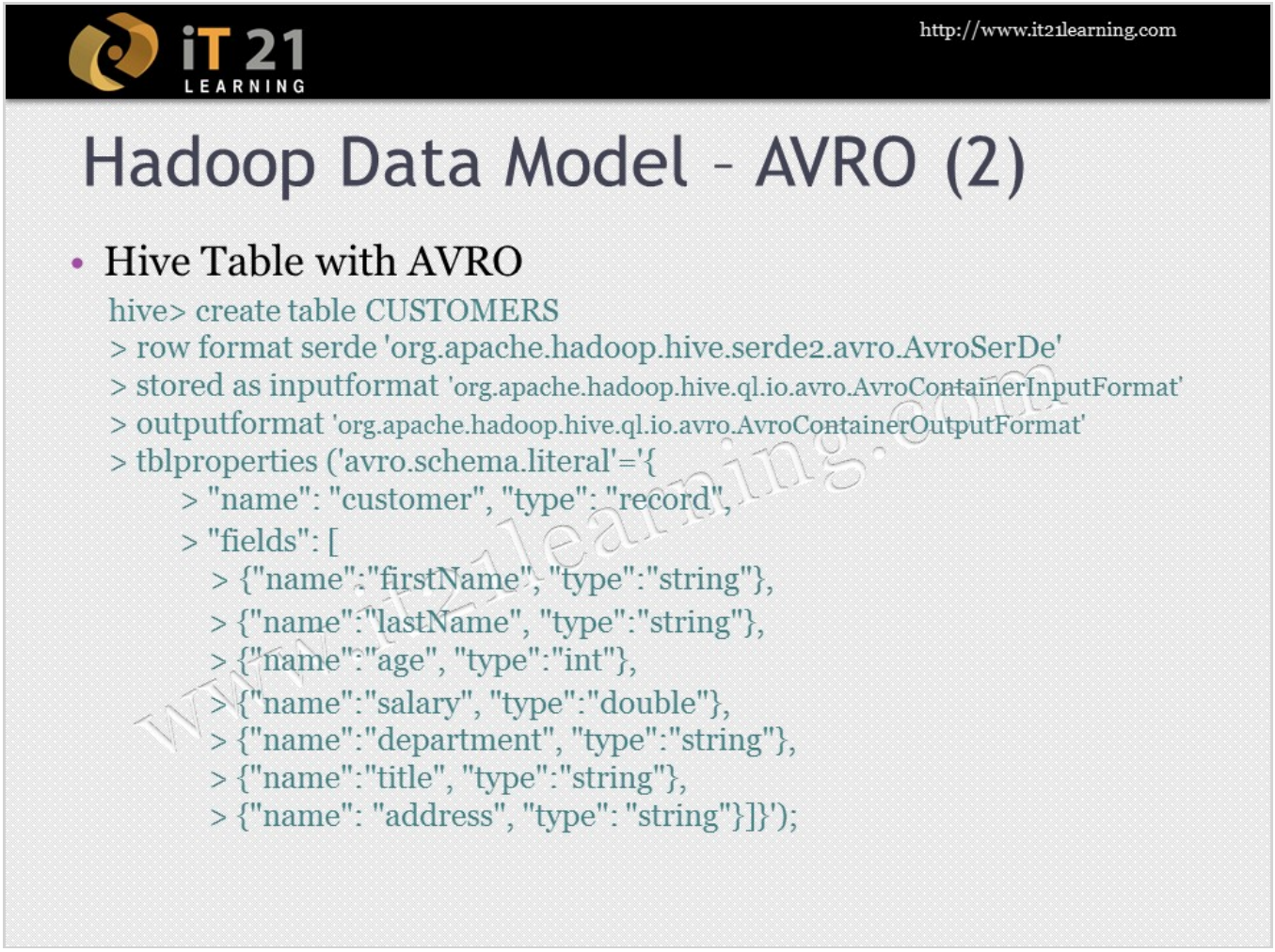

iT 21 LEARNING
http://www.it21learning.com
Hadoop Data Model - AVRO (1)
AVRO File
Stores data-definition in JSON
Store data in binary format
Support Data Types:
Primitive types – null, Boolean, int, long, float, double, bytes, strings, and binary data;
Complex types – arrays, maps, enumerations, records
Schema Evolution:
Cleanly handles schema changes like missing fields, added fields and changed fields;
Old programs can read new data and new programs can read old data;
www.it21learning.com

6

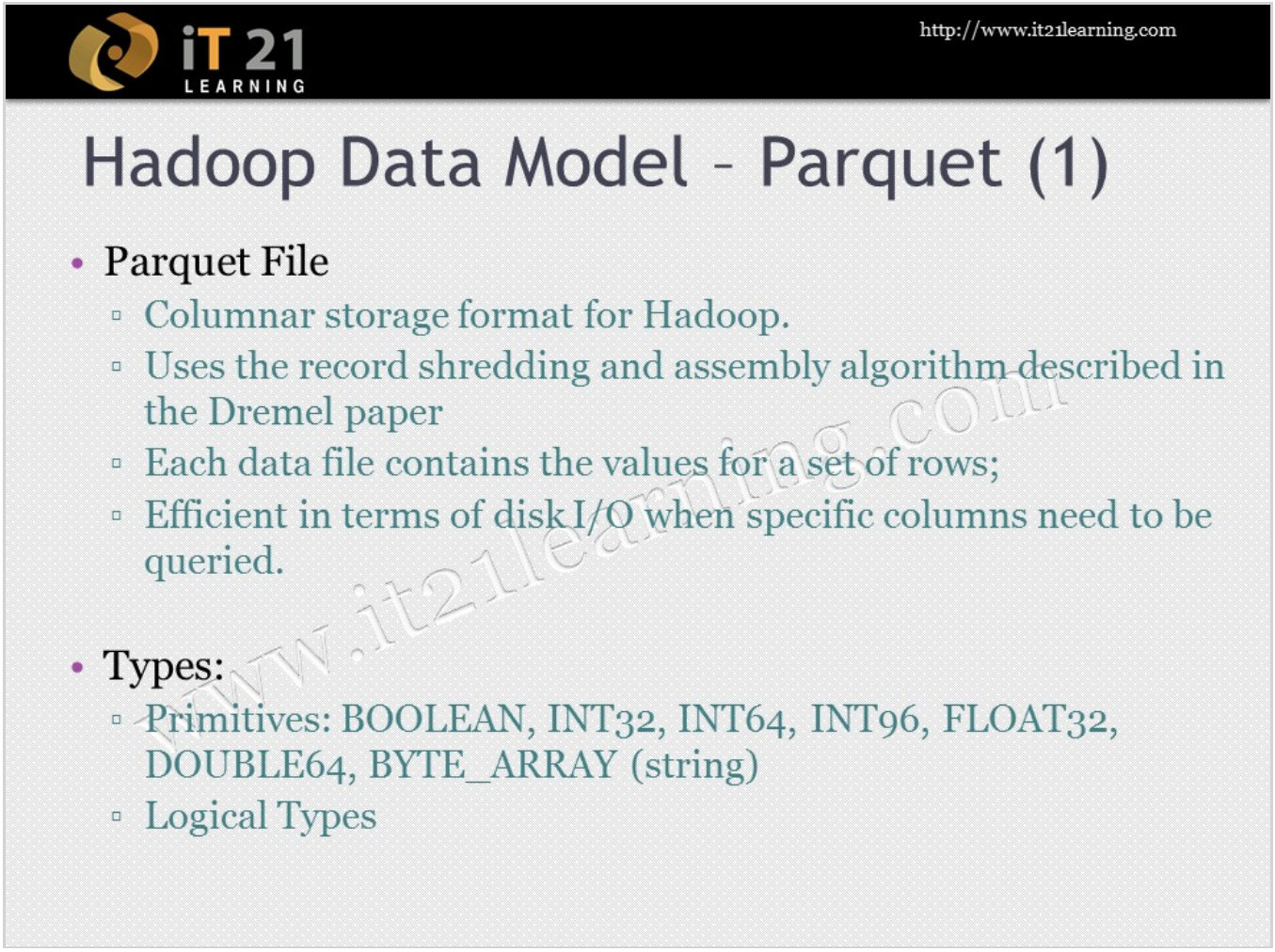
Hadoop Data Model – AVRO (2)



7


iT 21 LEARNING
http://www.it21learning.com
Hadoop Data Model - AVRO (2)
Hive Table with AVRO
hive> create table CUSTOMERS
> row format serde 'org.apache.hadoop.hive.serde2.avro.AvroSerDe'
> stored as inputformat 'org.apache.hadoop.hive.ql.io.avro.AvroContainerInputFormat'
> outputformat 'org.apache.hadoop.hive.ql.io.avro.AvroContainerOutputFormat'
> tblproperties ('avro.schema.literal'='{
> "name": "customer", "type": "record",
> "fields": [
> {"name":"firstName", "type":"string"},
> {"name":"lastName", "type":"string"},
> {"name":"age", "type":"int"},
> {"name":"salary", "type":"double"},
> {"name":"department", "type":"string"},
> {"name":"title", "type":"string"},
> {"name": "address", "type": "string"}]}');

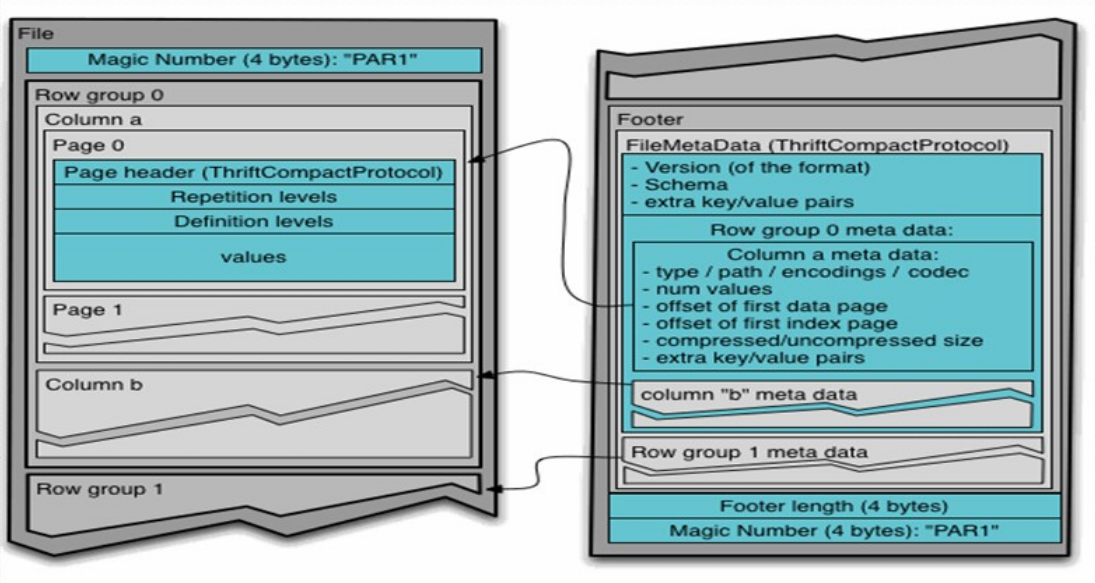
8


iT 21 LEARNING
http://www.it21learning.com
Hadoop Data Model - Parquet (1)
• Parquet File
▫ Columnar storage format for Hadoop.
▫ Uses the record shredding and assembly algorithm described in the Dremel paper
▫ Each data file contains the values for a set of rows;
▫ Efficient in terms of disk I/O when specific columns need to be queried.
• Types:
▫ Primitives: BOOLEAN, INT32, INT64, INT96, FLOAT32, DOUBLE64, BYTE_ARRAY (string)
▫ Logical Types
www.it21learning.com

9

Hadoop Data Model - Parquet (2)

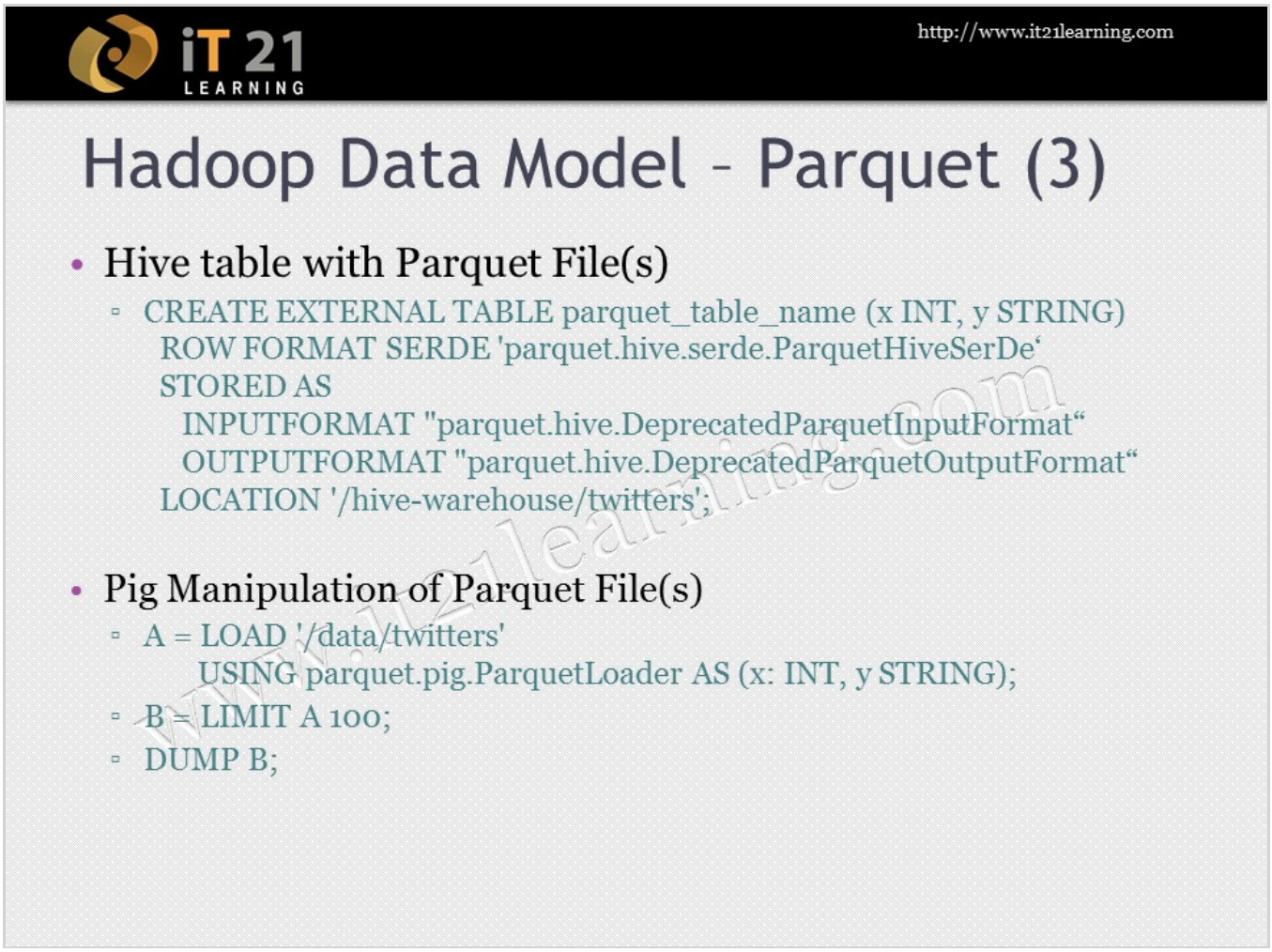
- Parquet File Format



- Format details - https://parquet.apache.org/documentation/latest/

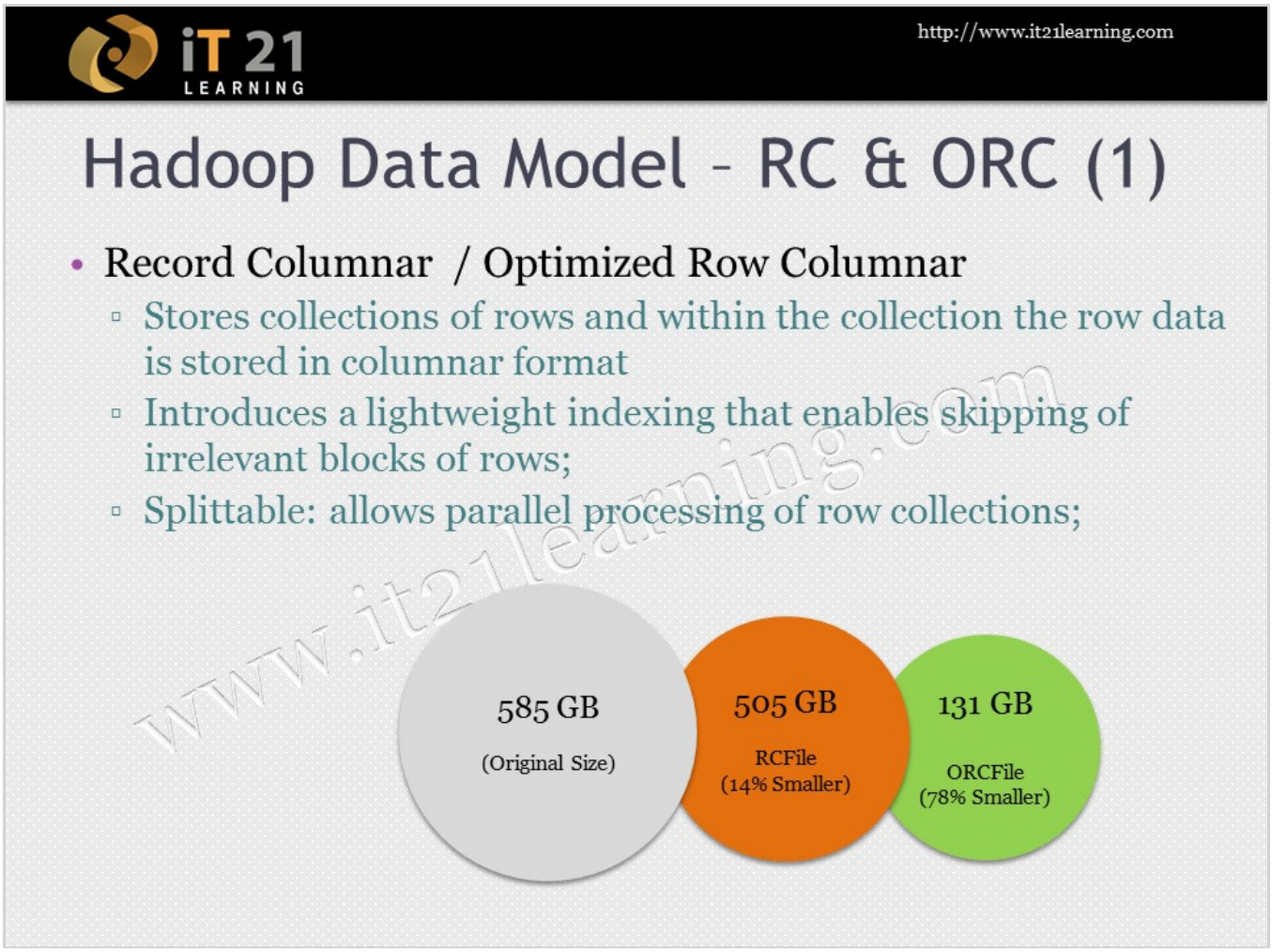
10

Hadoop Data Model – Parquet (3)



11

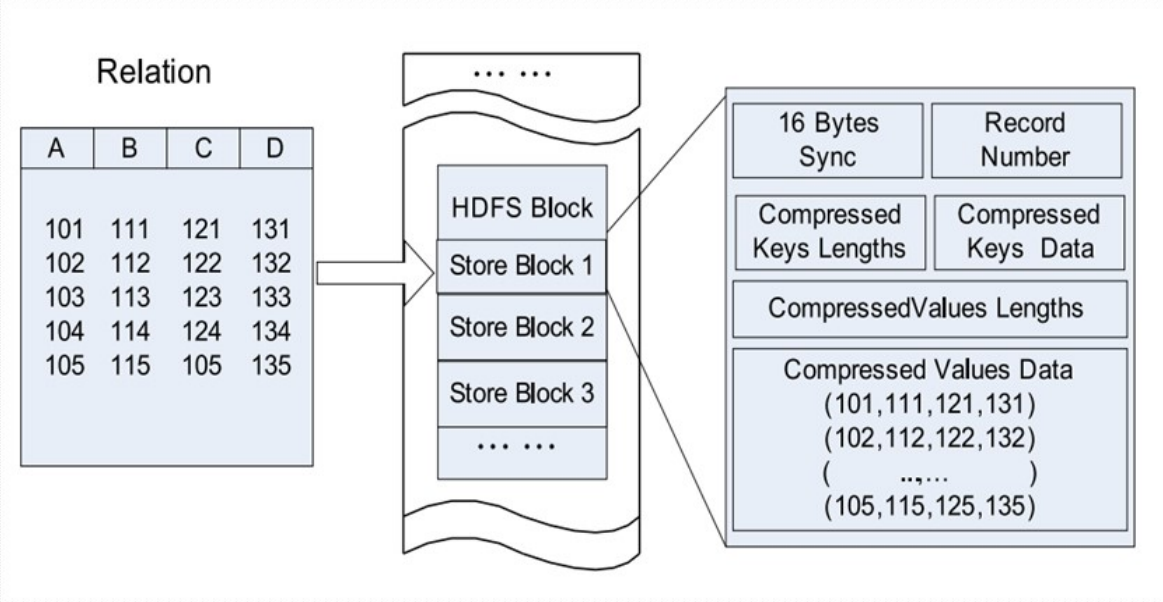
Hadoop Data Model – RC & ORC (1)



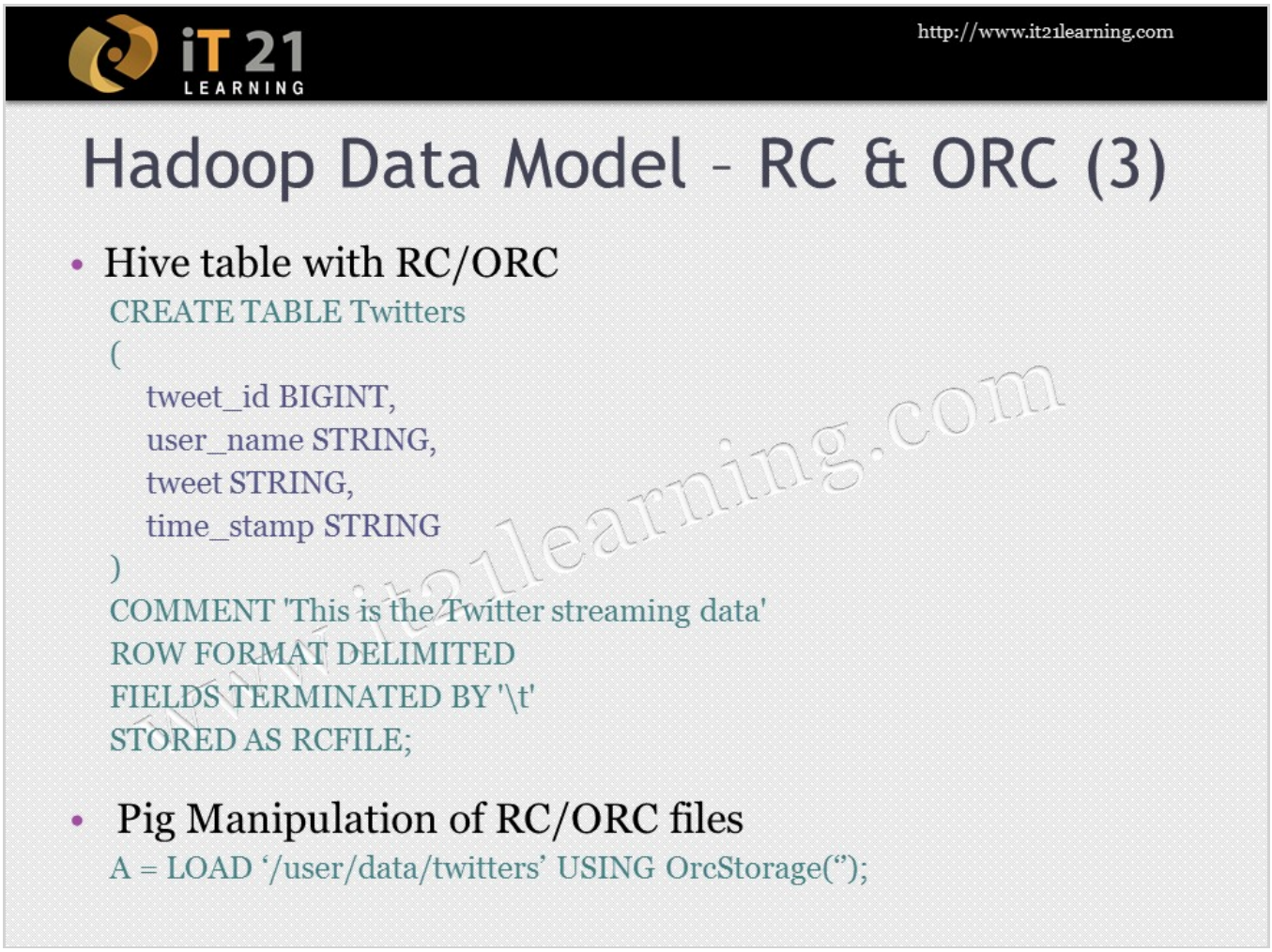
Additionally, compression on a column base is more efficient. It can take advantage of similarity of the data in a column.

Hadoop Data Model – RC & ORC (2)

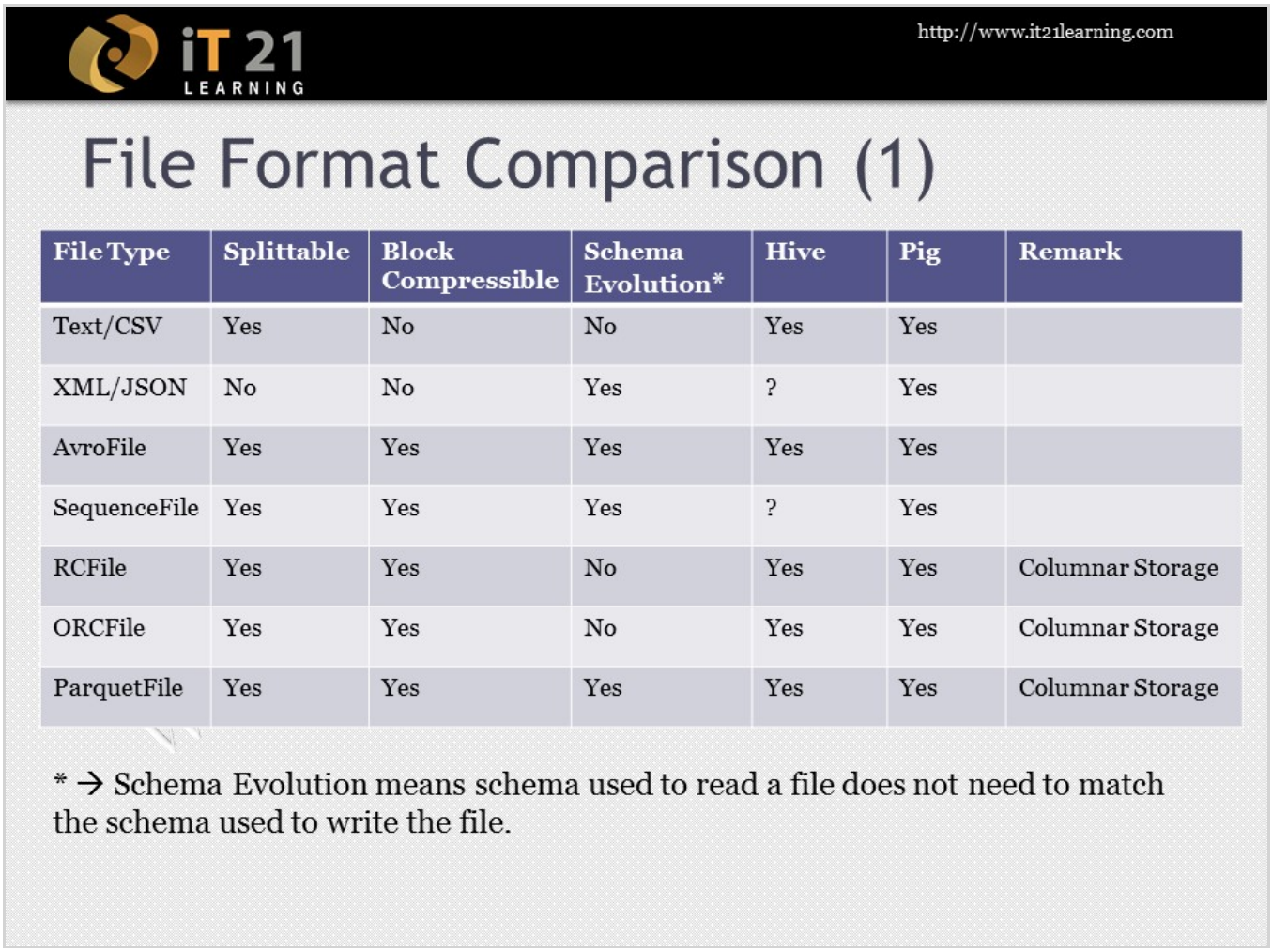
- RCFile Format



13

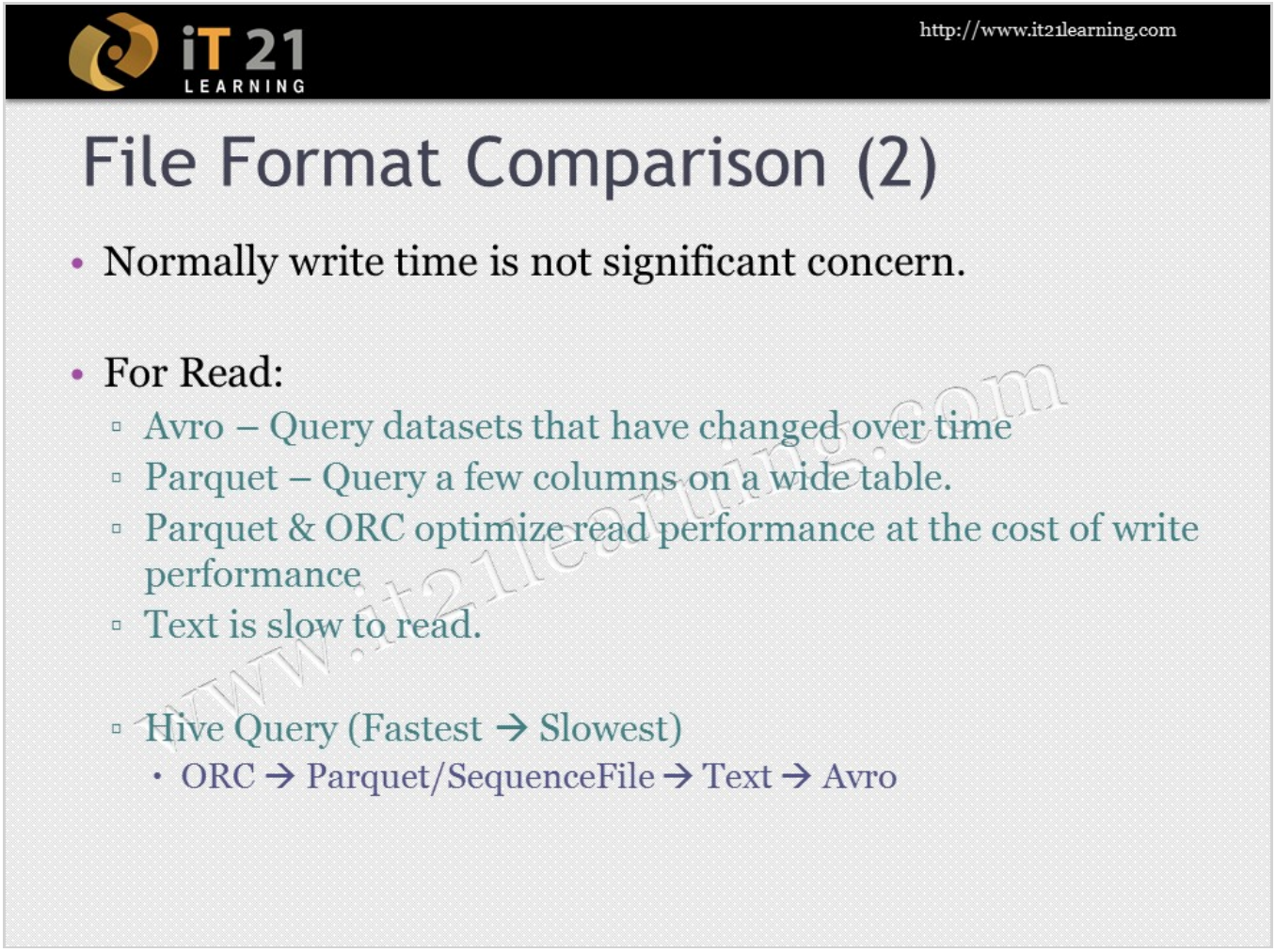

iT 21 LEARNING
http://www.it21learning.com
Hadoop Data Model - RC & ORC (3)
• Hive table with RC/ORC
CREATE TABLE Twitters
(
tweet_id BIGINT,
user_name STRING,
tweet STRING,
time_stamp STRING
)
COMMENT 'This is the Twitter streaming data'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
STORED AS RCFILE;
• Pig Manipulation of RC/ORC files
A = LOAD '/user/data/twitters' USING OrcStorage('');
www.it21learning.com

14


iT 21 LEARNING
http://www.it21learning.com
File Format Comparison (1)
File Type | Splittable | Block Compressible | Schema Evolution* | Hive | Pig | Remark
Text/CSV | Yes | No | No | Yes | Yes |
XML/JSON | No | No | Yes | ? | Yes |
AvroFile | Yes | Yes | Yes | Yes | Yes |
SequenceFile | Yes | Yes | Yes | ? | Yes |
RCFile | Yes | Yes | No | Yes | Yes | Columnar Storage
ORCFile | Yes | Yes | No | Yes | Yes | Columnar Storage
ParquetFile | Yes | Yes | Yes | Yes | Yes | Columnar Storage
* → Schema Evolution means schema used to read a file does not need to match the schema used to write the file.

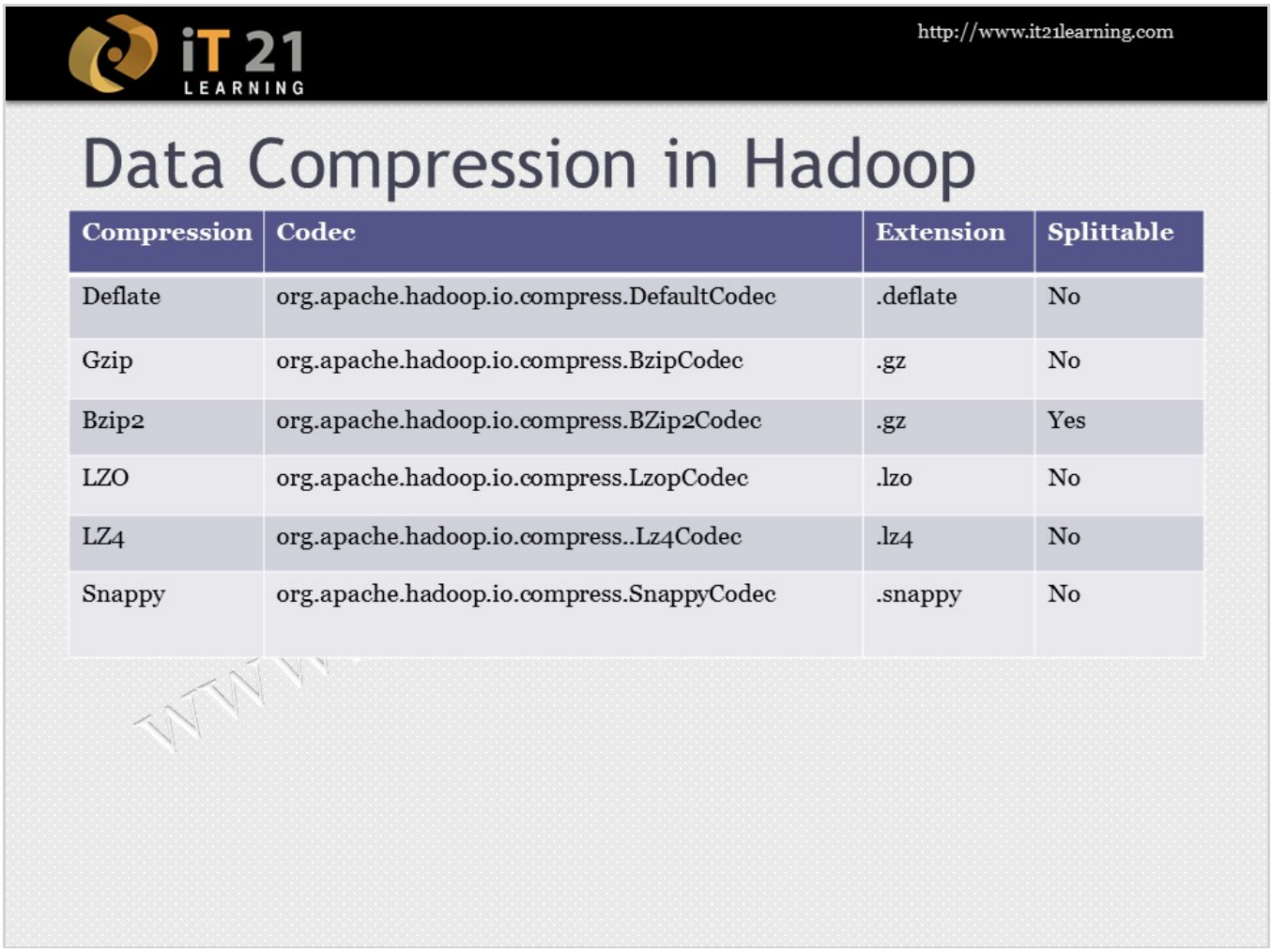
15

File Format Comparison (2)

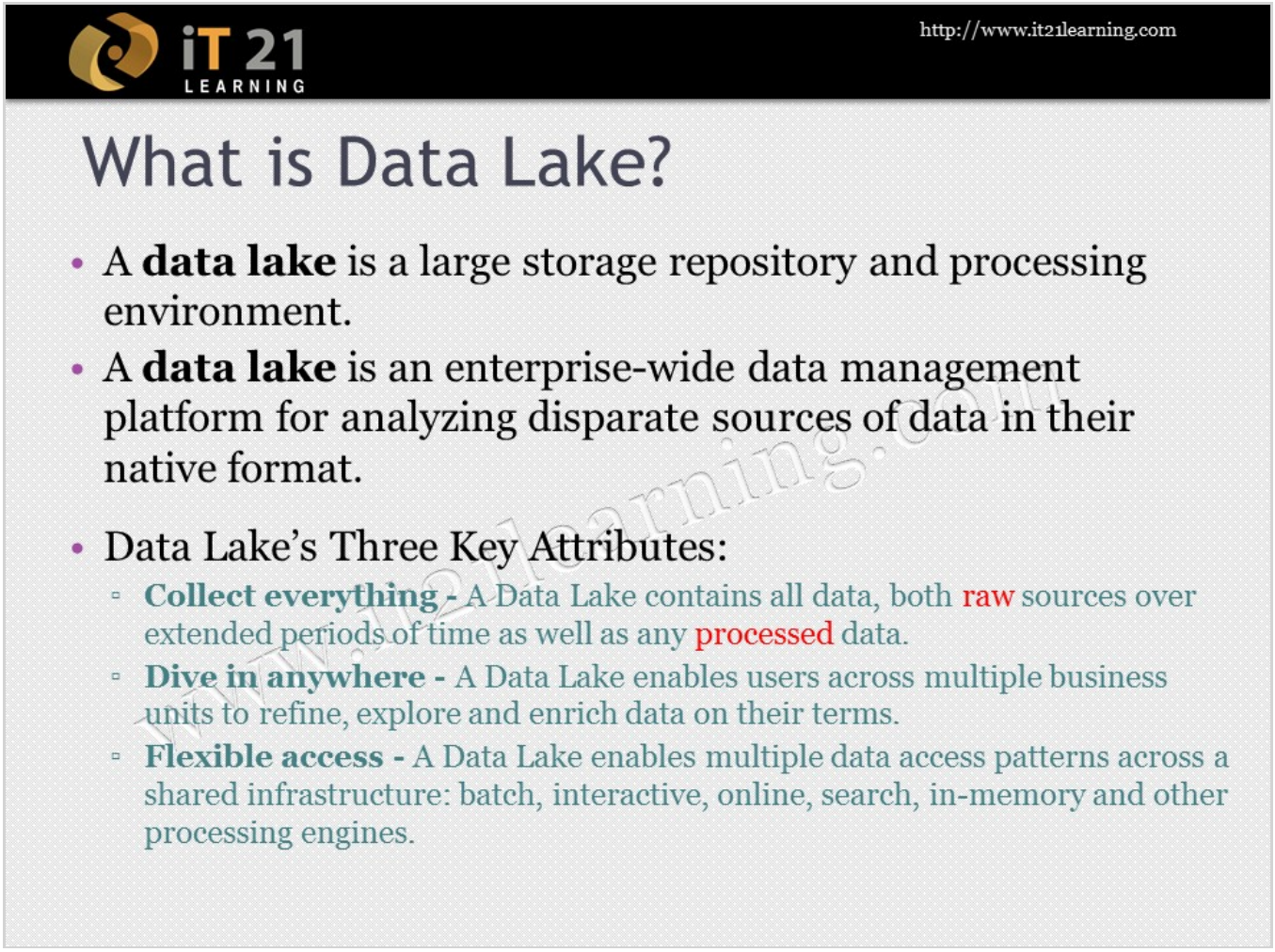


16

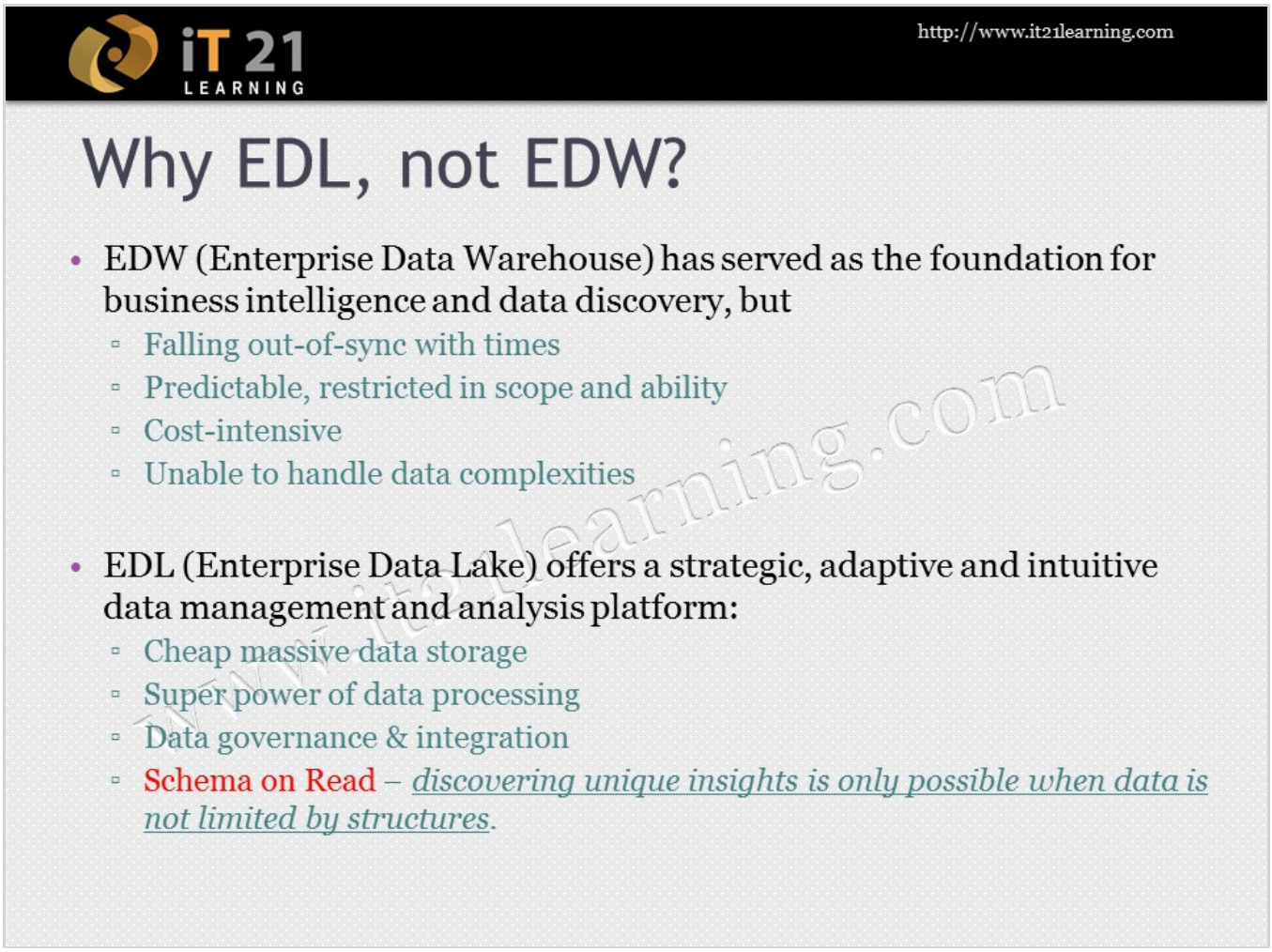
Data Compression in Hadoop



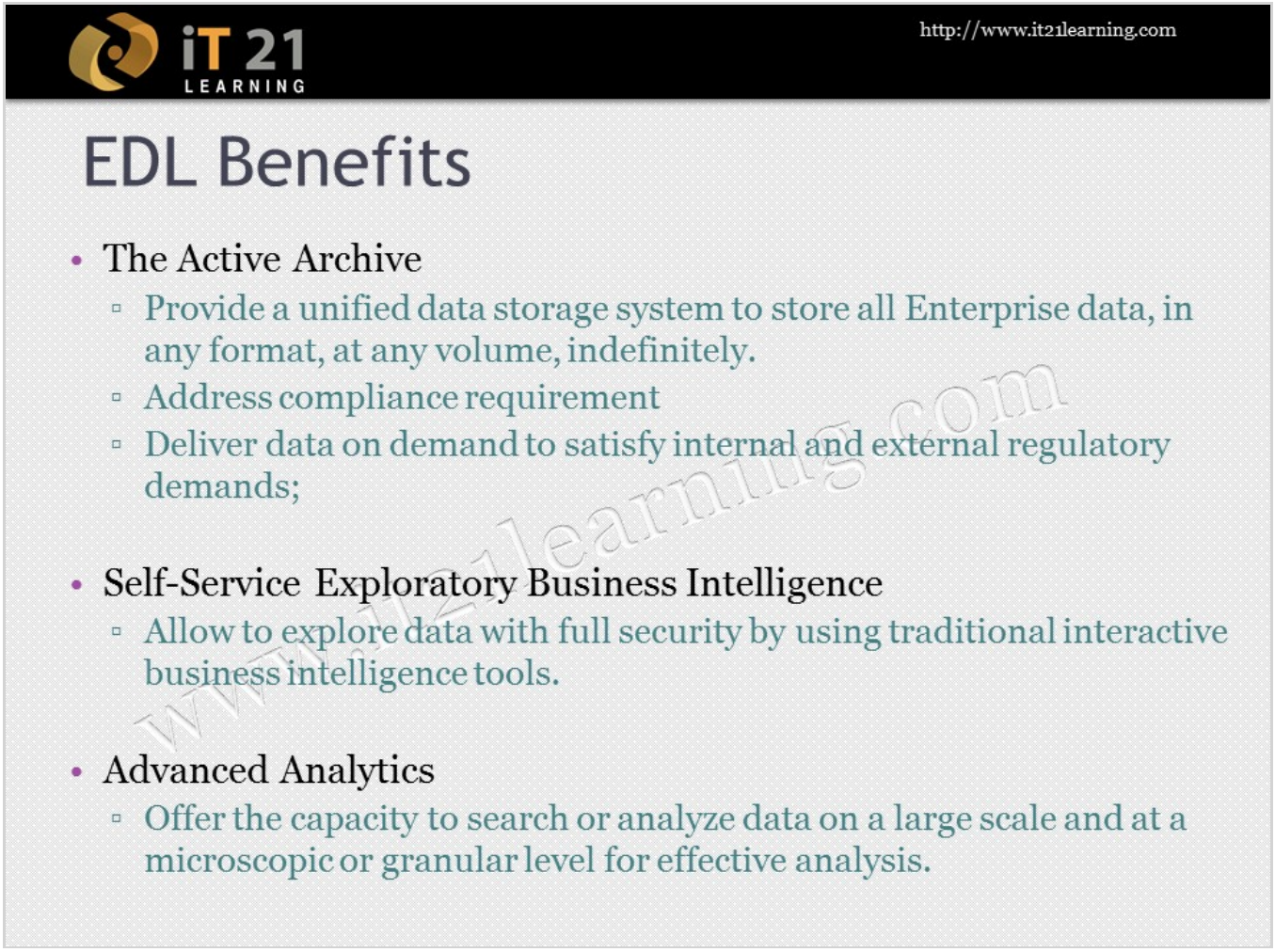
17


iT 21 LEARNING
http://www.it21learning.com
What is Data Lake?
• A data lake is a large storage repository and processing environment.
• A data lake is an enterprise-wide data management platform for analyzing disparate sources of data in their native format.
• Data Lake's Three Key Attributes:
▫ Collect everything - A Data Lake contains all data, both raw sources over extended periods of time as well as any processed data.
▫ Dive in anywhere - A Data Lake enables users across multiple business units to refine, explore and enrich data on their terms.
▫ Flexible access - A Data Lake enables multiple data access patterns across a shared infrastructure: batch, interactive, online, search, in-memory and other processing engines.

18

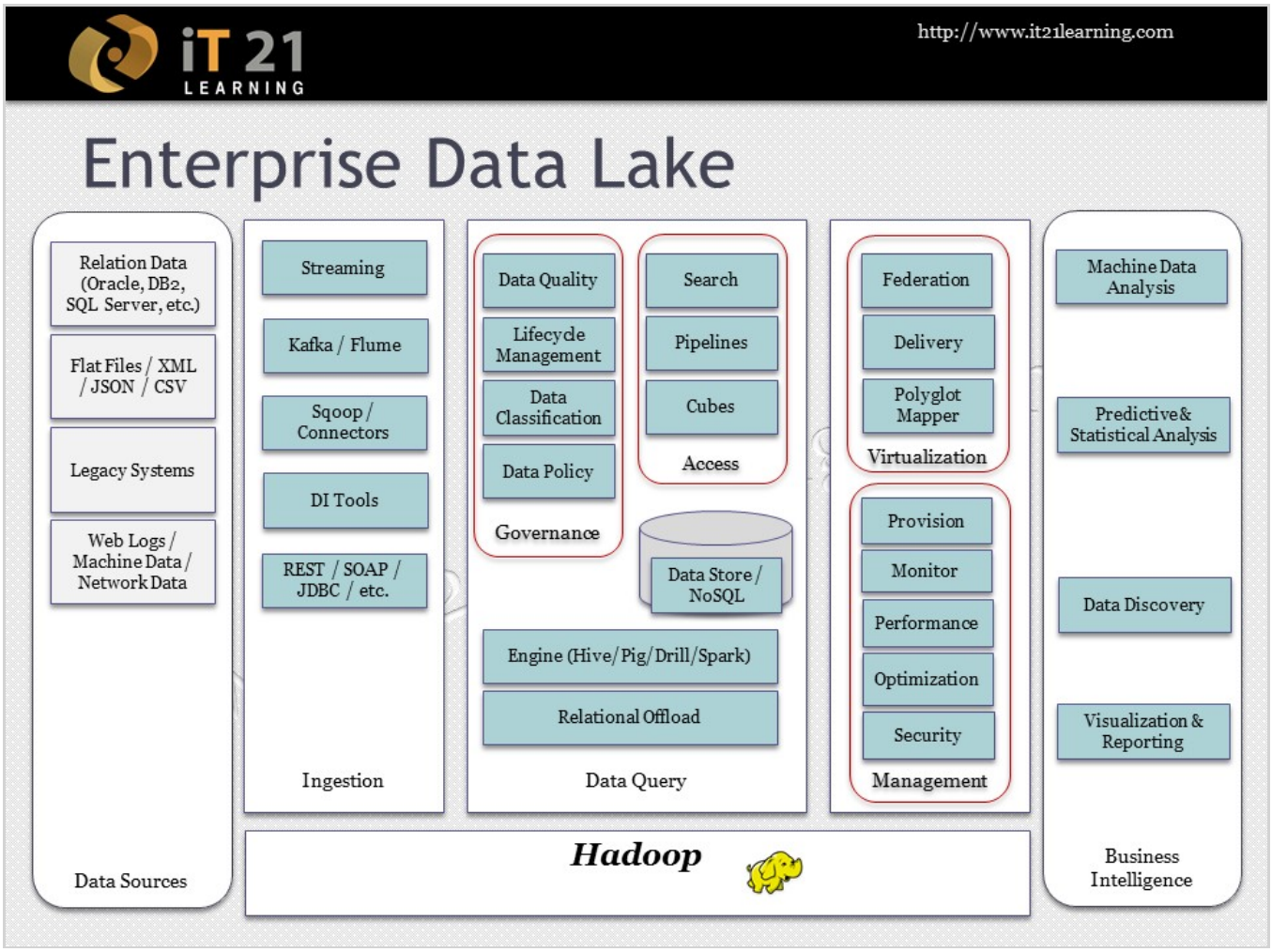

iT 21 LEARNING
http://www.it21learning.com
Why EDL, not EDW?
• EDW (Enterprise Data Warehouse) has served as the foundation for business intelligence and data discovery, but
▫ Falling out-of-sync with times
▫ Predictable, restricted in scope and ability
▫ Cost-intensive
▫ Unable to handle data complexities
• EDL (Enterprise Data Lake) offers a strategic, adaptive and intuitive data management and analysis platform:
▫ Cheap massive data storage
▫ Super power of data processing
▫ Data governance & integration
▫ Schema on Read – discovering unique insights is only possible when data is not limited by structures.

19


iT 21 LEARNING
http://www.it21learning.com
EDL Benefits
• The Active Archive
▫ Provide a unified data storage system to store all Enterprise data, in any format, at any volume, indefinitely.
▫ Address compliance requirement
▫ Deliver data on demand to satisfy internal and external regulatory demands;
• Self-Service Exploratory Business Intelligence
▫ Allow to explore data with full security by using traditional interactive business intelligence tools.
• Advanced Analytics
▫ Offer the capacity to search or analyze data on a large scale and at a microscopic or granular level for effective analysis.

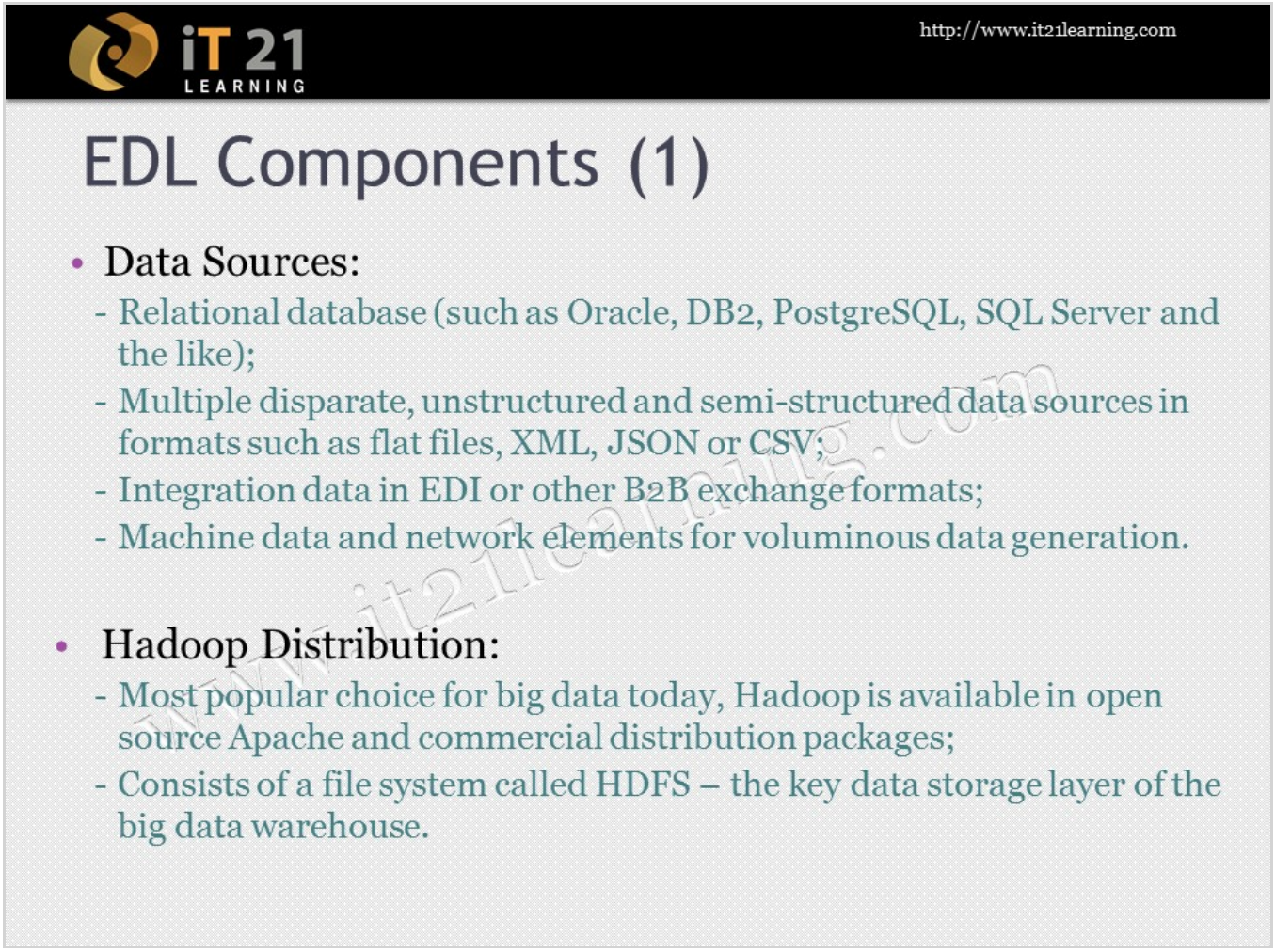
20

Enterprise Data Lake

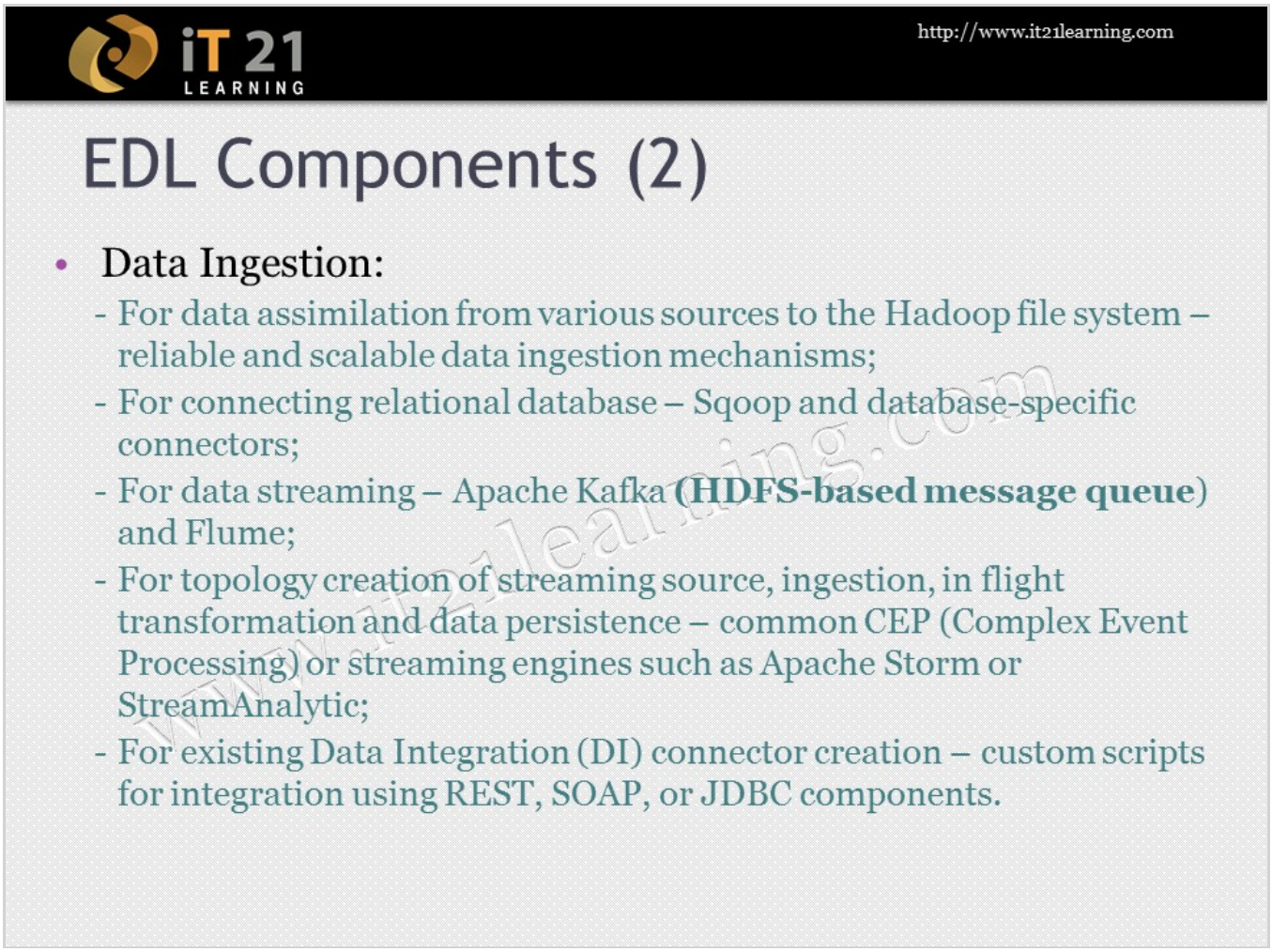


Apache Kafka is HDFS based Message Queuing; Apache Drill is a SQL engine for Hadoop, NoSQL and Cloud Storage. A query can join a HDFS file, with data from NoSQL such as MongoDB etc. Cubes – Cubes functions; Relational Offload – Tools and solutions to help enterprises offload expensive computing from relational data warehouses to big data warehouses;

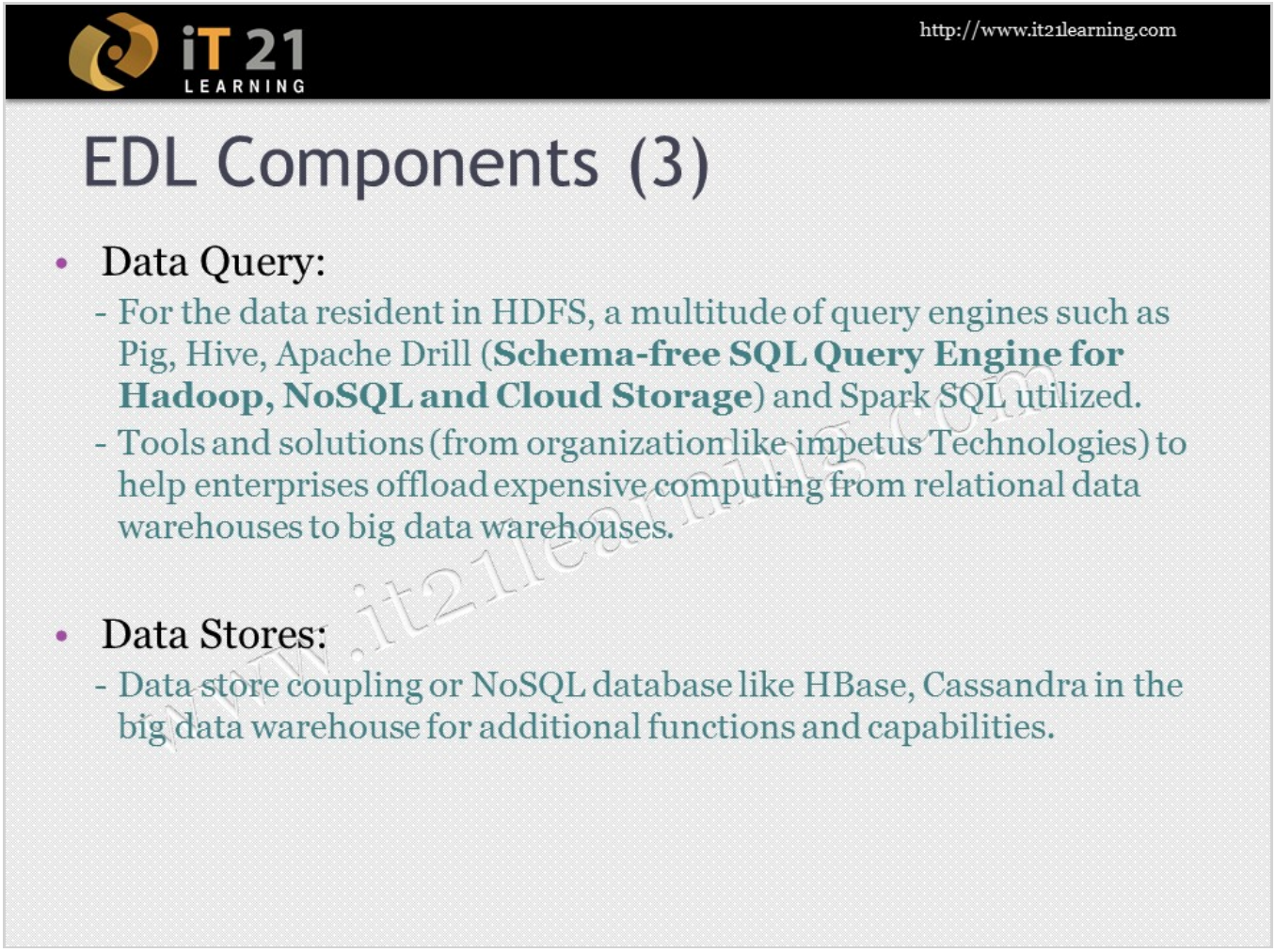
EDL Components (1)



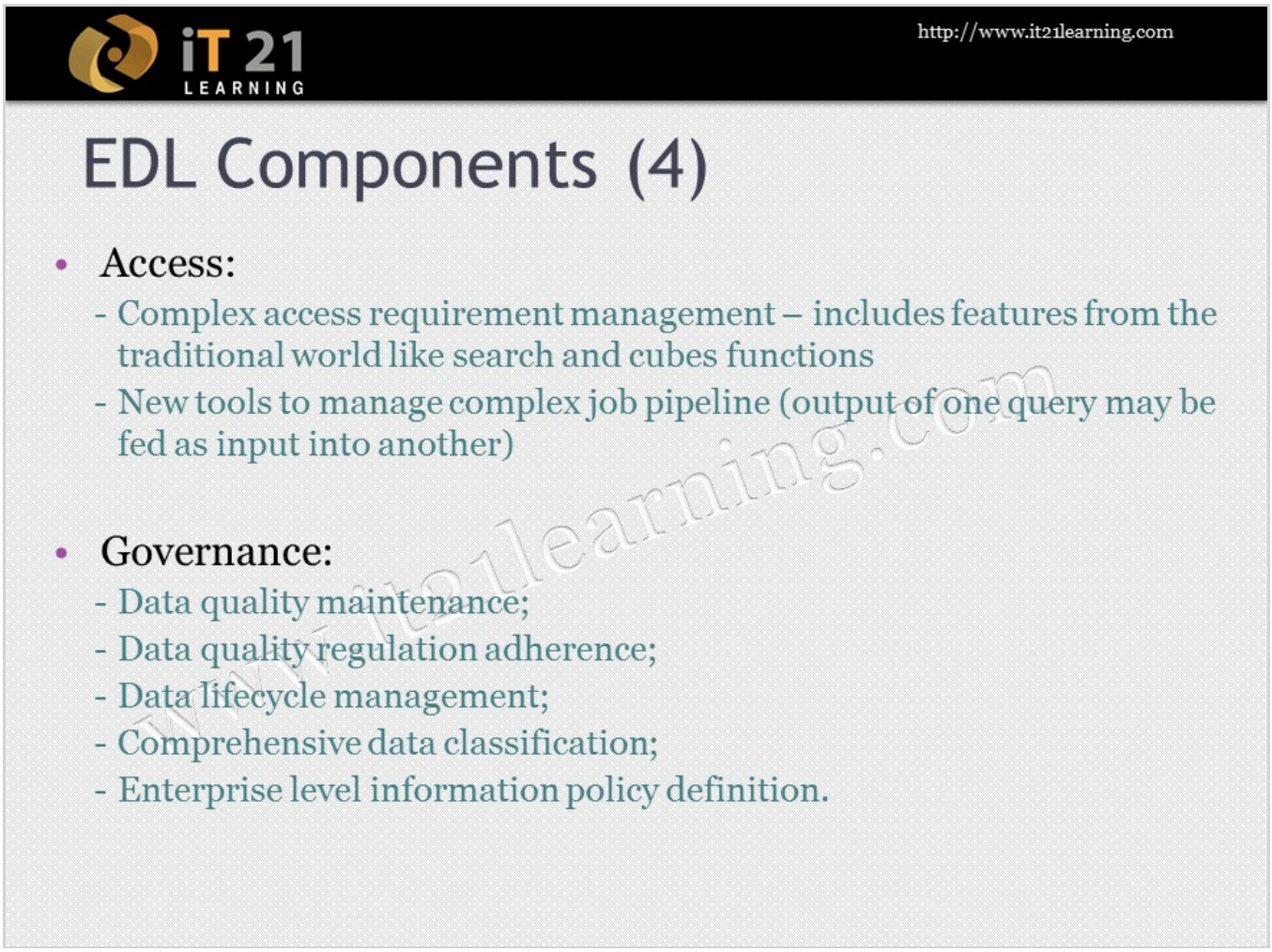
22


iT 21 LEARNING
http://www.it21learning.com
EDL Components (2)
• Data Ingestion:
- For data assimilation from various sources to the Hadoop file system – reliable and scalable data ingestion mechanisms;
- For connecting relational database – Sqoop and database-specific connectors;
- For data streaming – Apache Kafka (HDFS-based message queue) and Flume;
- For topology creation of streaming source, ingestion, in flight transformation and data persistence – common CEP (Complex Event Processing) or streaming engines such as Apache Storm or StreamAnalytic;
- For existing Data Integration (DI) connector creation – custom scripts for integration using REST, SOAP, or JDBC components.

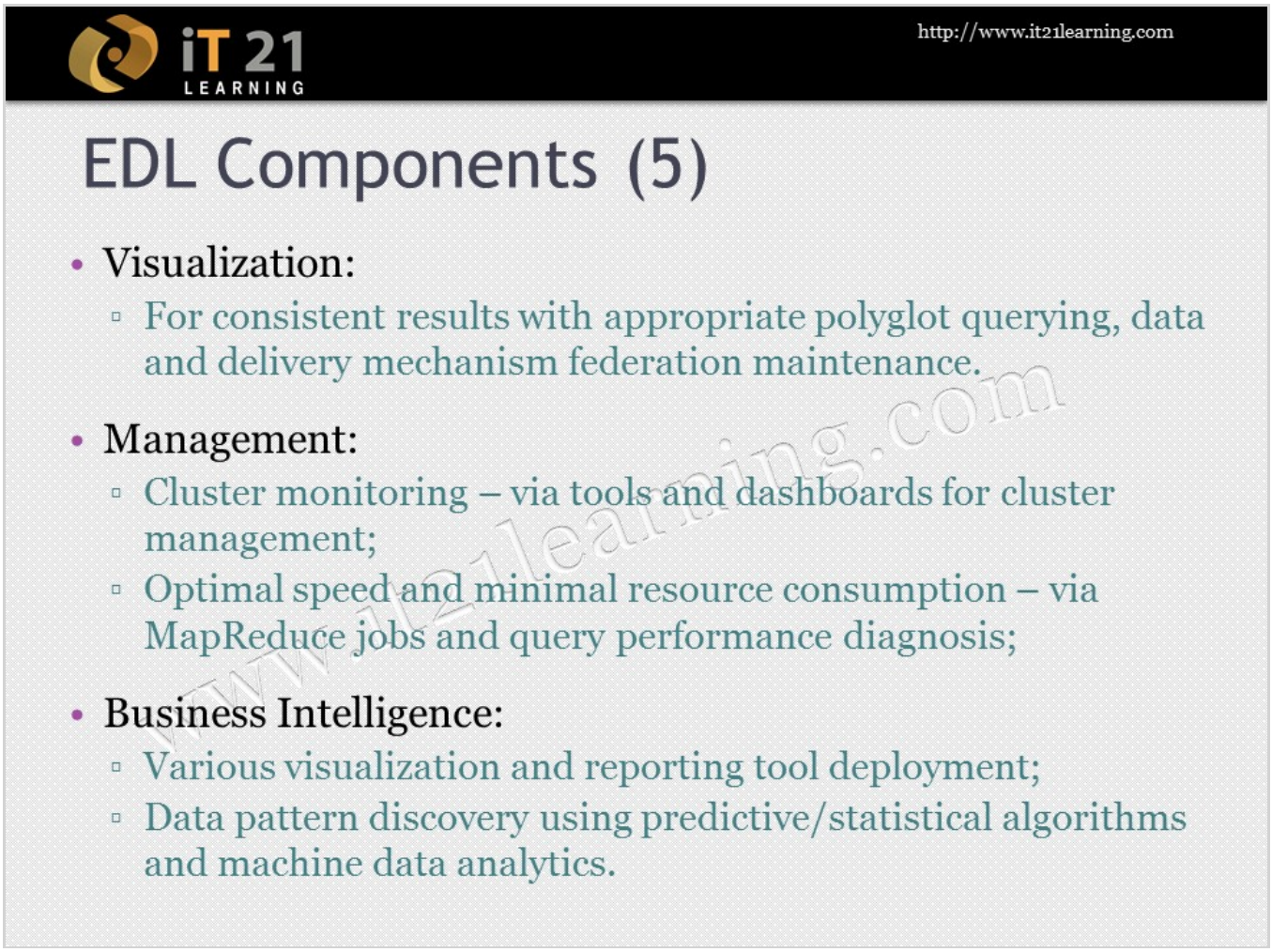
23


iT 21 LEARNING
http://www.it21learning.com
EDL Components (3)
• Data Query:
- For the data resident in HDFS, a multitude of query engines such as Pig, Hive, Apache Drill (Schema-free SQL Query Engine for Hadoop, NoSQL and Cloud Storage) and Spark SQL utilized.
- Tools and solutions (from organization like impetus Technologies) to help enterprises offload expensive computing from relational data warehouses to big data warehouses.
• Data Stores:
- Data store coupling or NoSQL database like HBase, Cassandra in the big data warehouse for additional functions and capabilities.

24

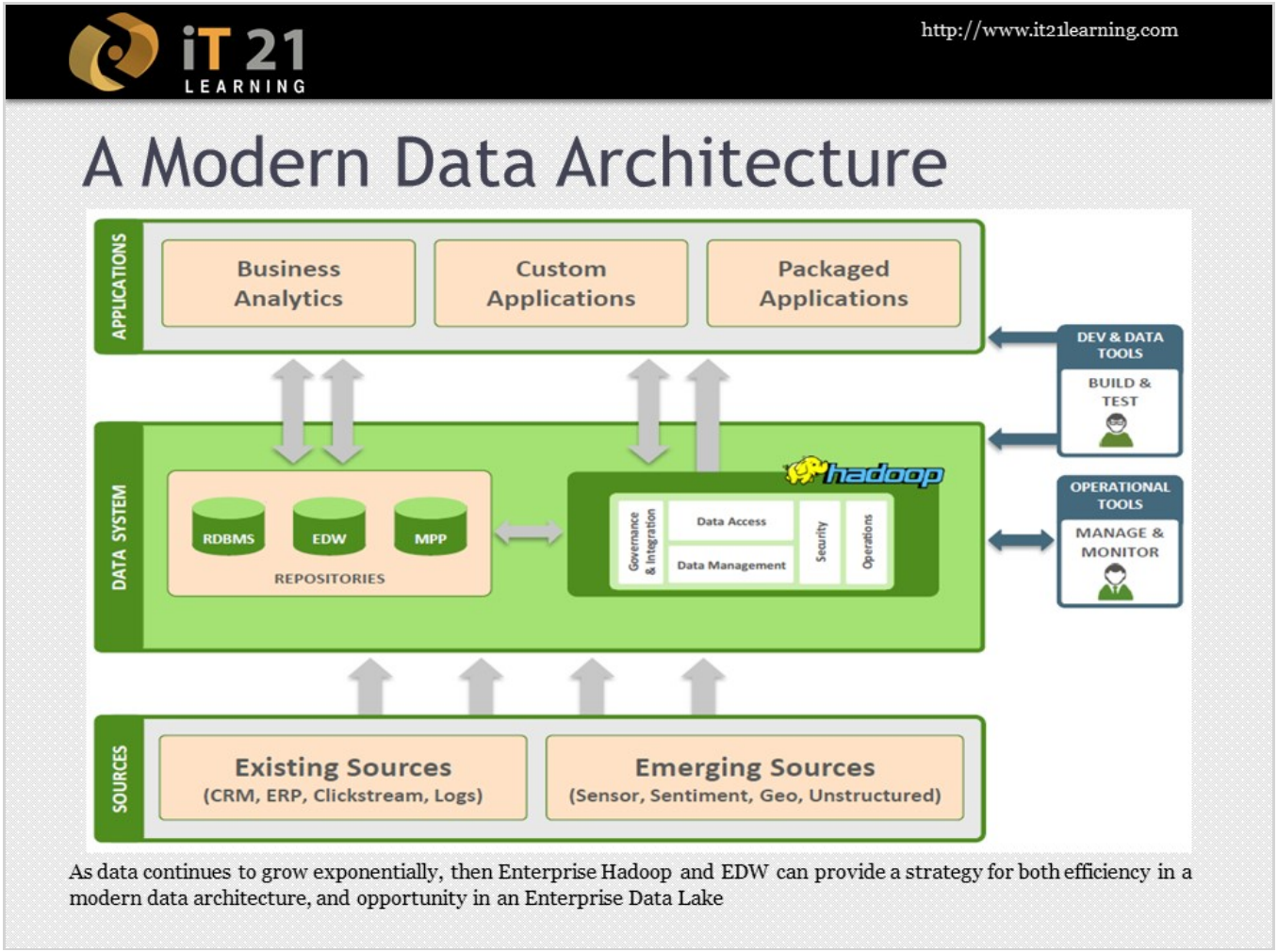

iT 21 LEARNING
http://www.it21learning.com
EDL Components (4)
Access:
- Complex access requirement management – includes features from the traditional world like search and cubes functions
- New tools to manage complex job pipeline (output of one query may be fed as input into another)
Governance:
- Data quality maintenance;
- Data quality regulation adherence;
- Data lifecycle management;
- Comprehensive data classification;
- Enterprise level information policy definition.
www.it21learning.com

25


iT 21 LEARNING
http://www.it21learning.com
EDL Components (5)
• Visualization:
▫ For consistent results with appropriate polyglot querying, data and delivery mechanism federation maintenance.
• Management:
▫ Cluster monitoring – via tools and dashboards for cluster management;
▫ Optimal speed and minimal resource consumption – via MapReduce jobs and query performance diagnosis;
• Business Intelligence:
▫ Various visualization and reporting tool deployment;
▫ Data pattern discovery using predictive/statistical algorithms and machine data analytics.
www.it21learning.com

26

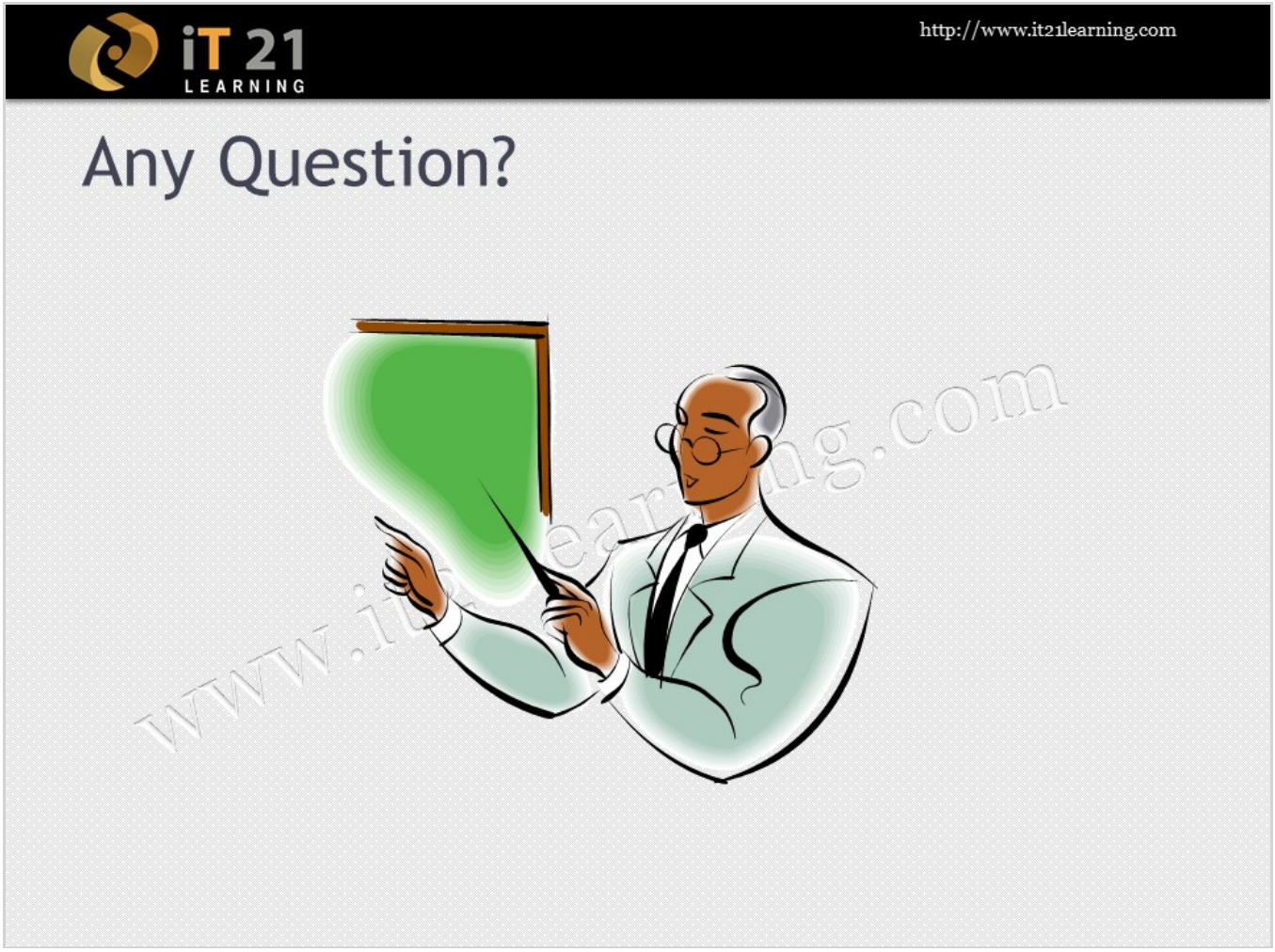
A Modern Data Architecture



As data continues to grow exponentially, then Enterprise Hadoop and EDW can provide a strategy for both efficiency in a modern data architecture, and opportunity in an Enterprise Data Lake

27

Any Question?



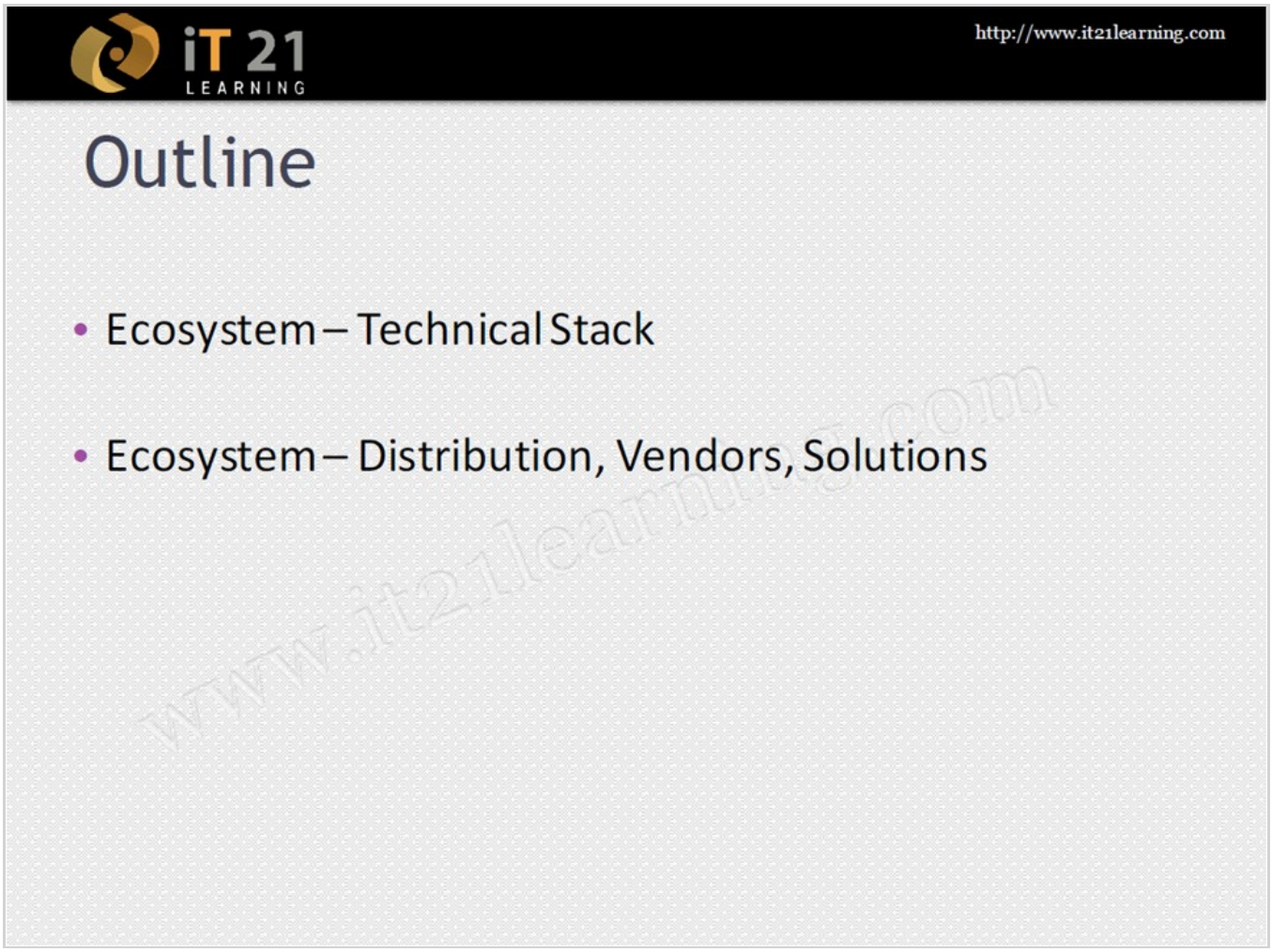
28

Appendix A - Hadoop Ecosystem


Hadoop Ecosystem Overview
Will Du
Oct. 29, 2015@Toronto
iT 21
LEARNING
http://www.it21learning.com

1

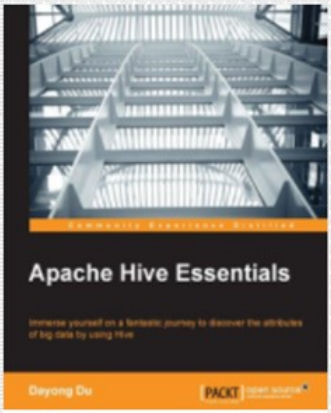
Outline


iT 21 LEARNING
http://www.it21learning.com
Outline
• Ecosystem – Technical Stack
• Ecosystem – Distribution, Vendors, Solutions

2

Who Am I - Will Du

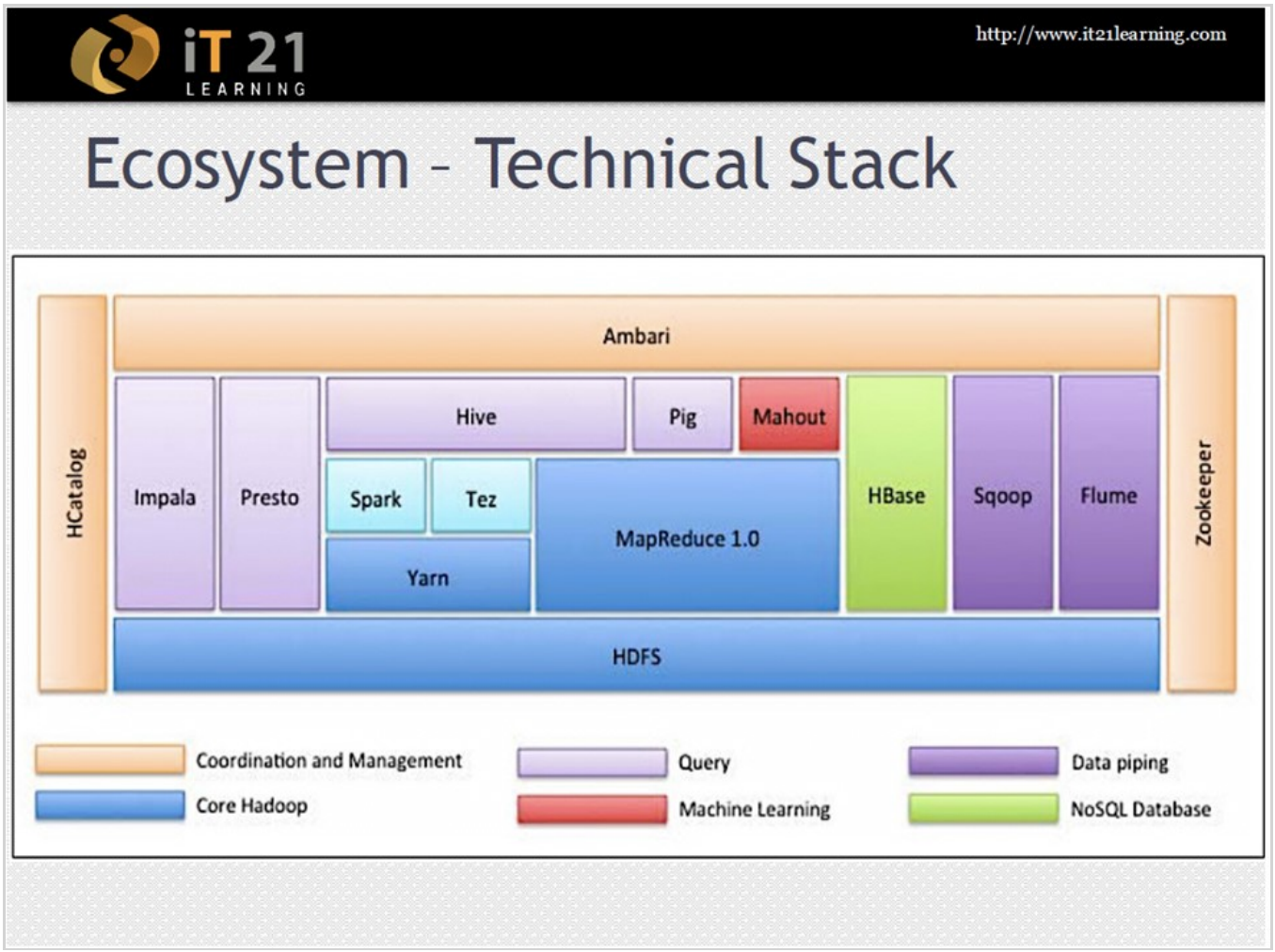
- A big data practitioner, leader, and developer
- 10+ years of experience in enterprise data warehouse and big data architecture
- The author of *Apache Hive Essentials*, the 1st book on Hive v1.0 in the world
- The first few certified Cloudera Hadoop Developer for CDH 4&5
- www.sparkera.ca



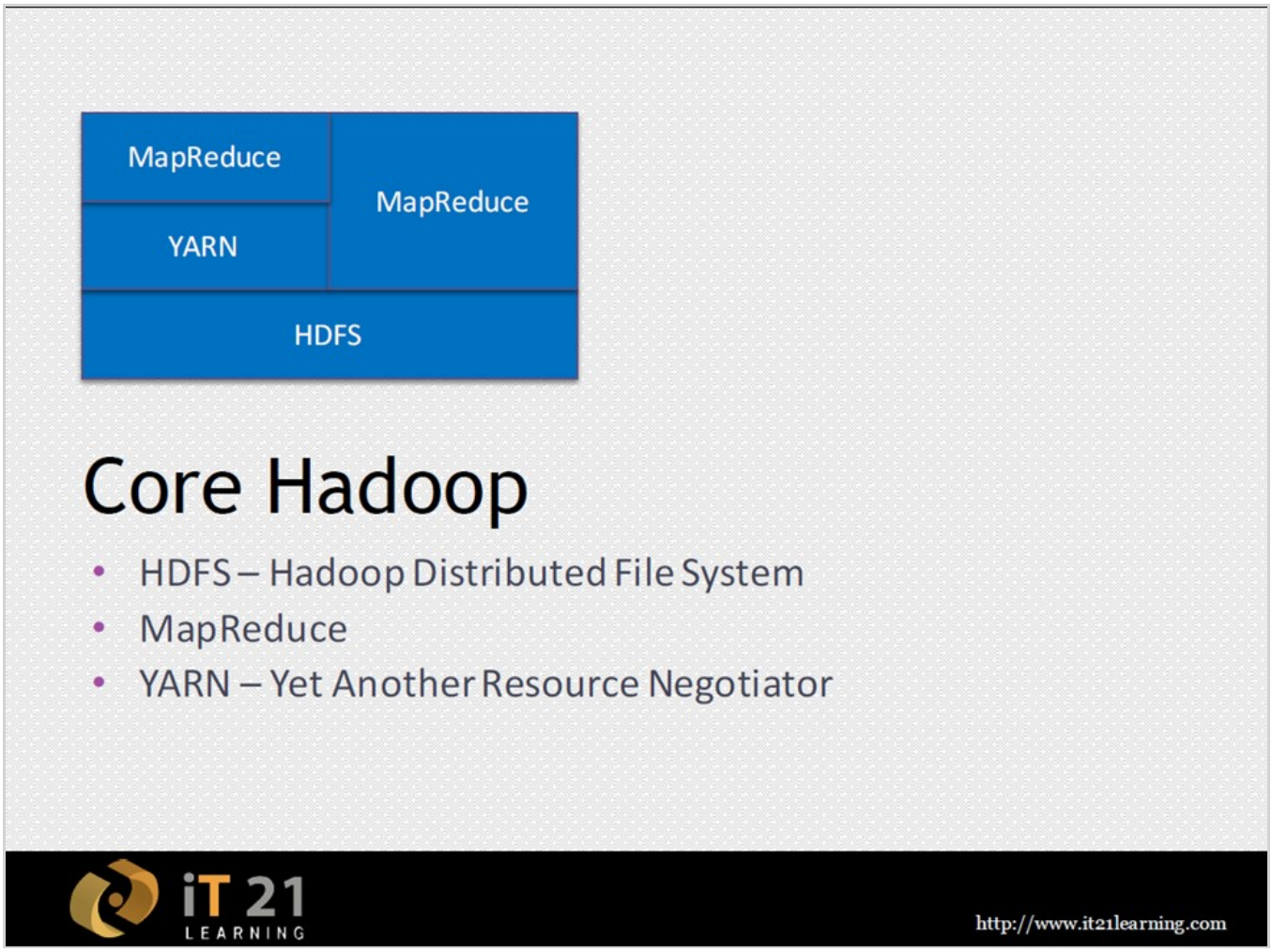
Sparkera
ignites big data imaginations

3

Ecosystem - Technical Stack

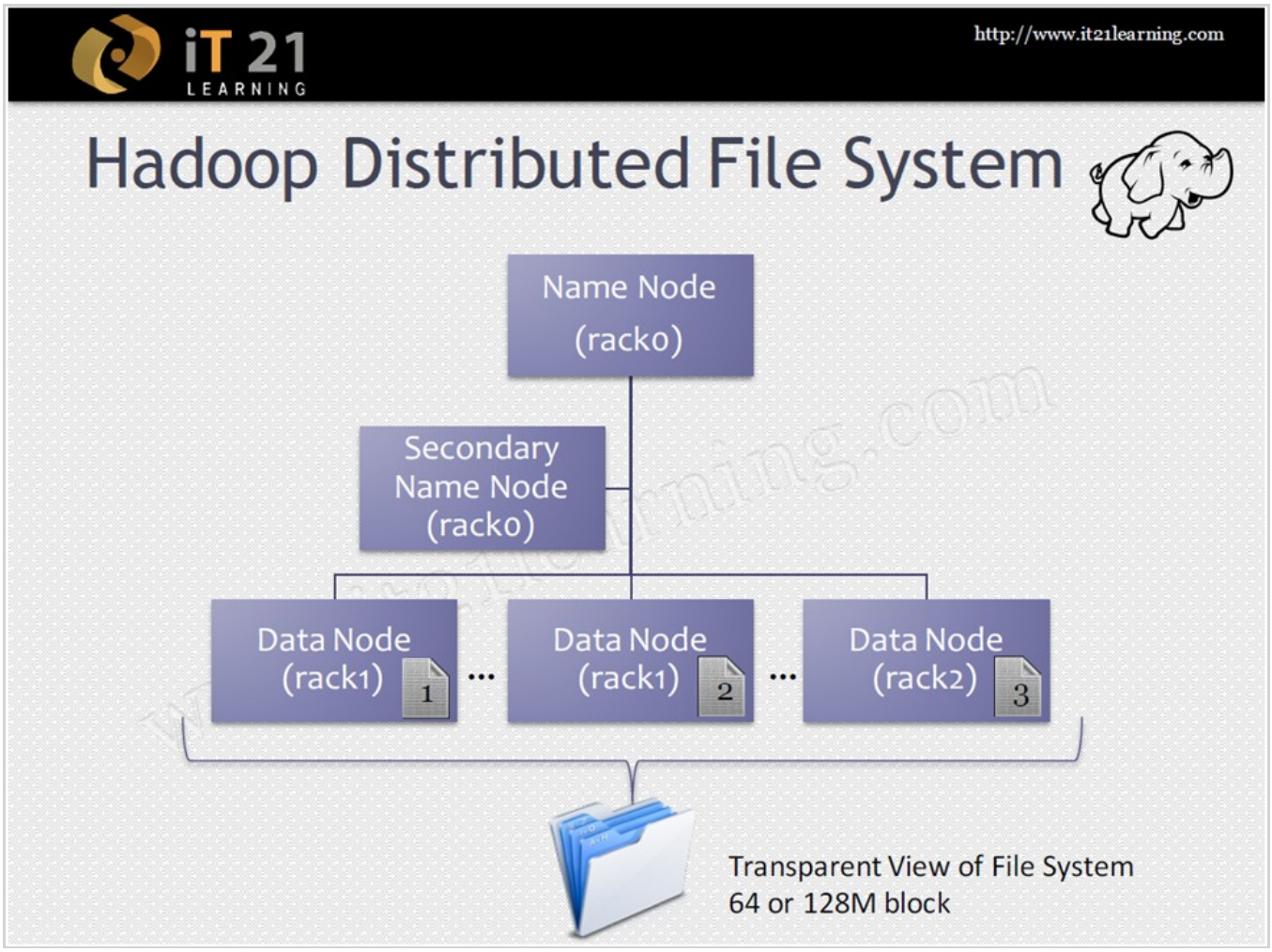


4

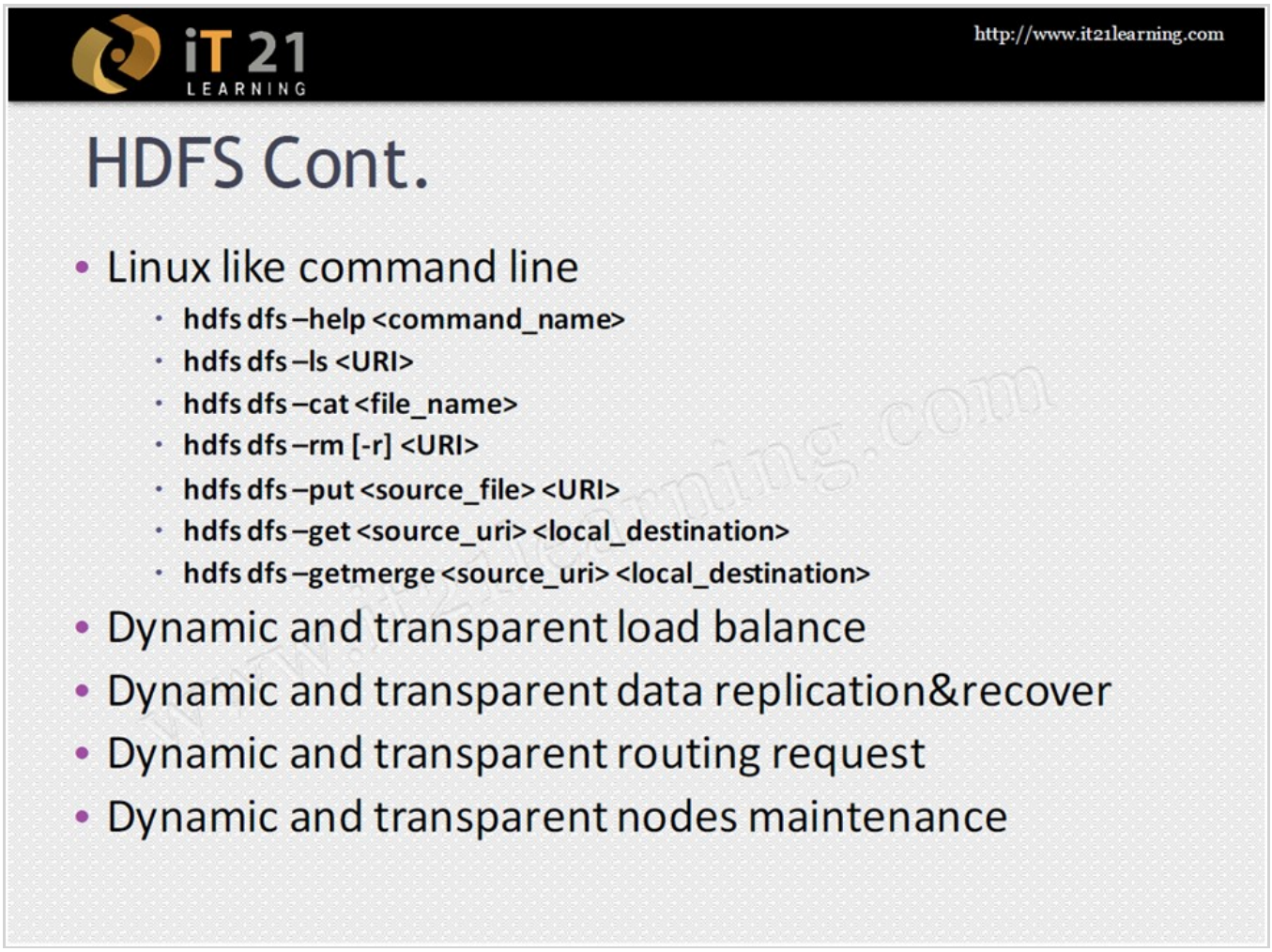

MapReduce
YARN
MapReduce
HDFS
Core Hadoop
• HDFS – Hadoop Distributed File System
• MapReduce
• YARN – Yet Another Resource Negotiator
iT 21 LEARNING
http://www.it21learning.com

5

Hadoop Distributed File System

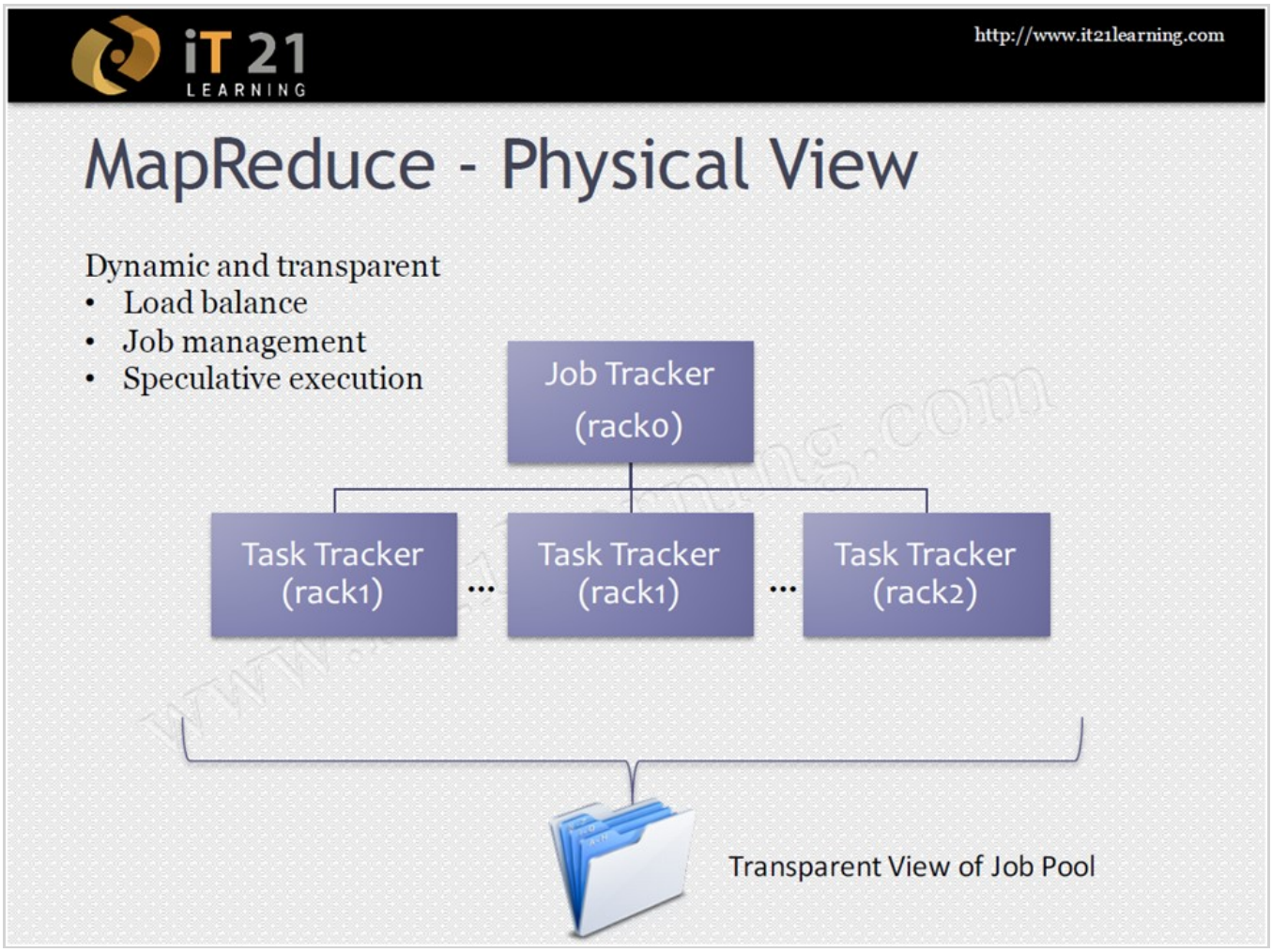


6


iT 21 LEARNING
http://www.it21learning.com
HDFS Cont.
• Linux like command line
• hdfs dfs –help <command_name>
• hdfs dfs –ls <URI>
• hdfs dfs –cat <file_name>
• hdfs dfs –rm [-r] <URI>
• hdfs dfs –put <source_file> <URI>
• hdfs dfs –get <source_uri> <local_destination>
• hdfs dfs –getmerge <source_uri> <local_destination>
• Dynamic and transparent load balance
• Dynamic and transparent data replication&recover
• Dynamic and transparent routing request
• Dynamic and transparent nodes maintenance

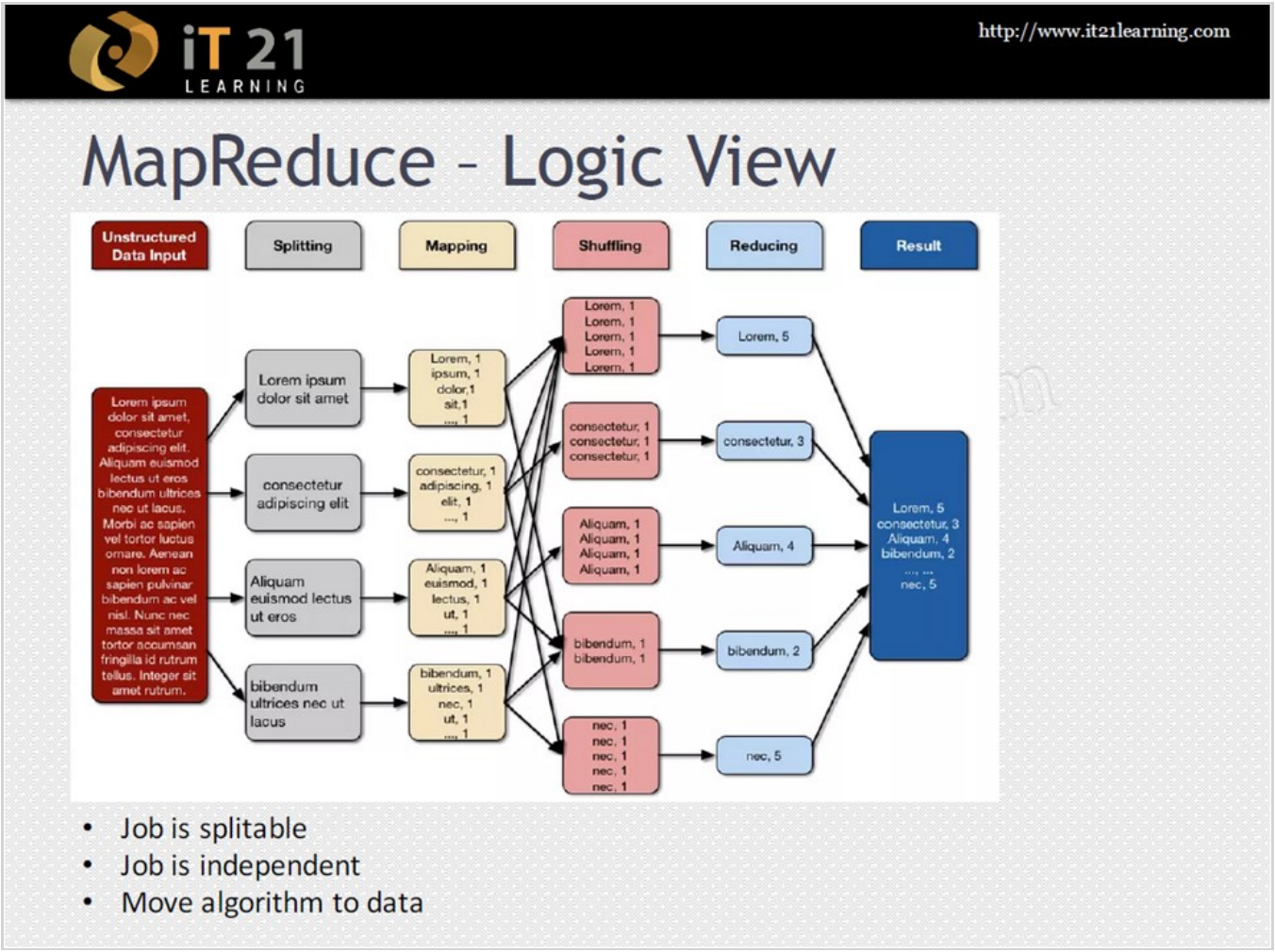
7

MapReduce - Physical View



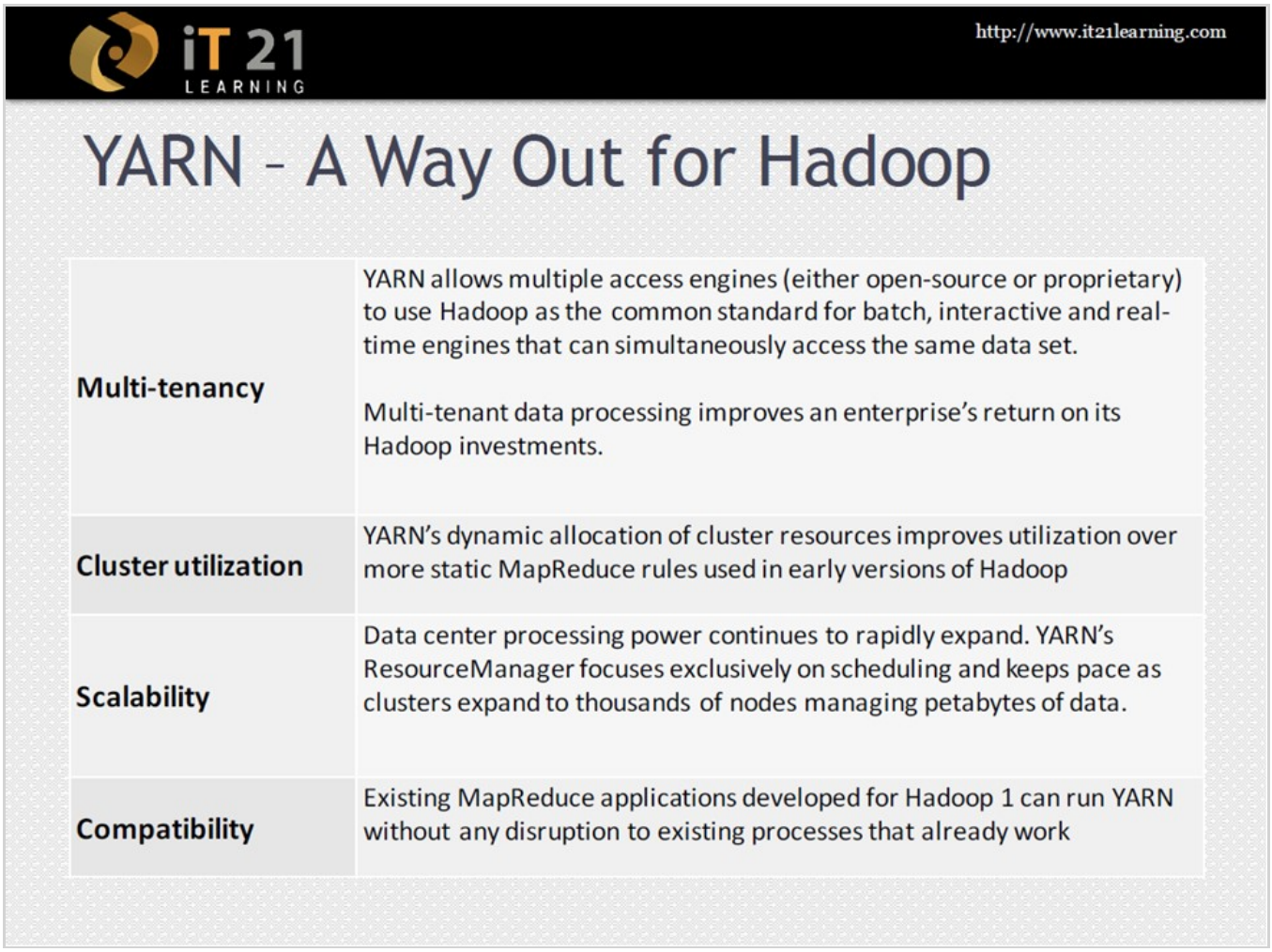
8

MapReduce - Logic View

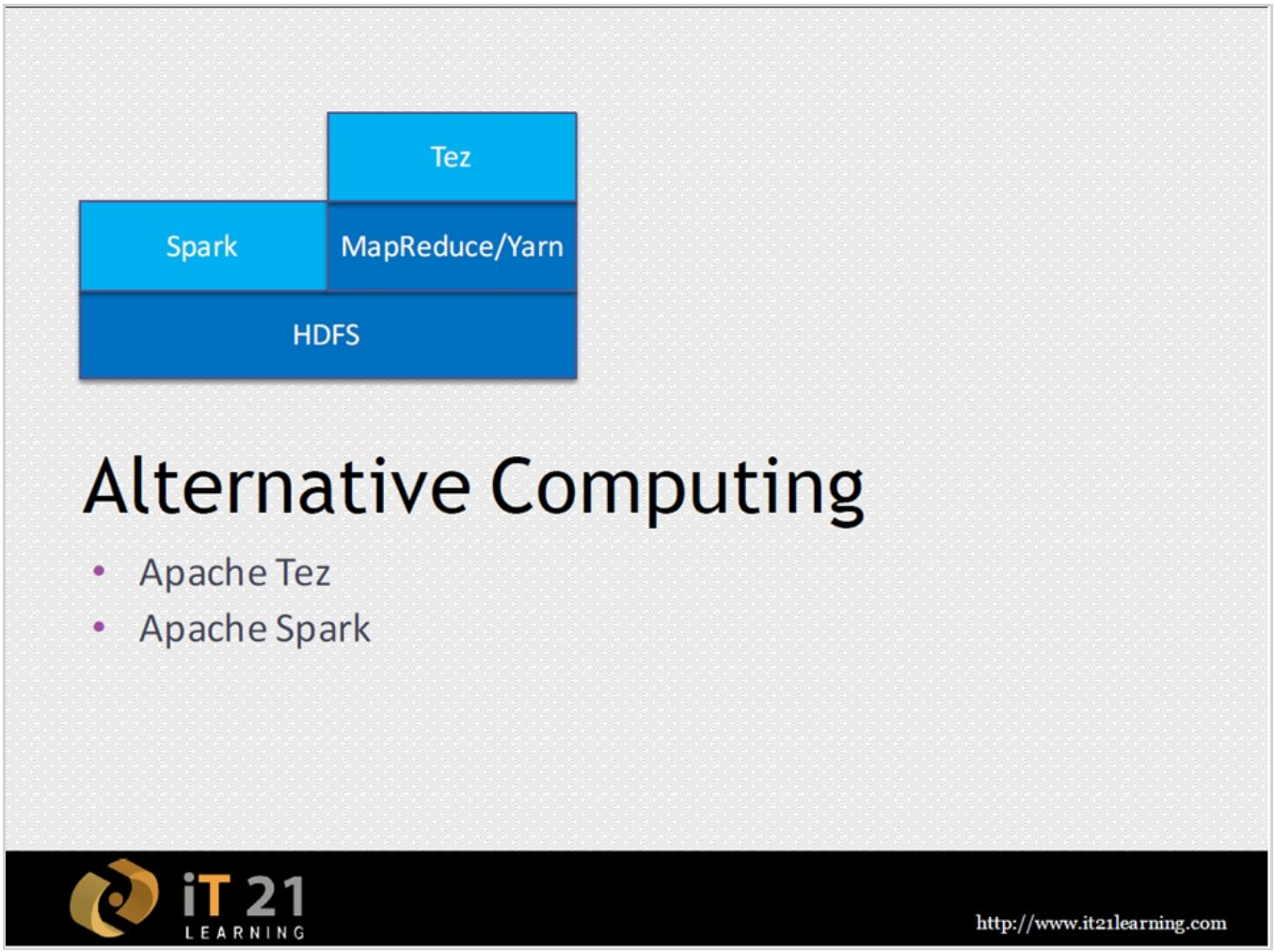


9

YARN - A Way Out for Hadoop

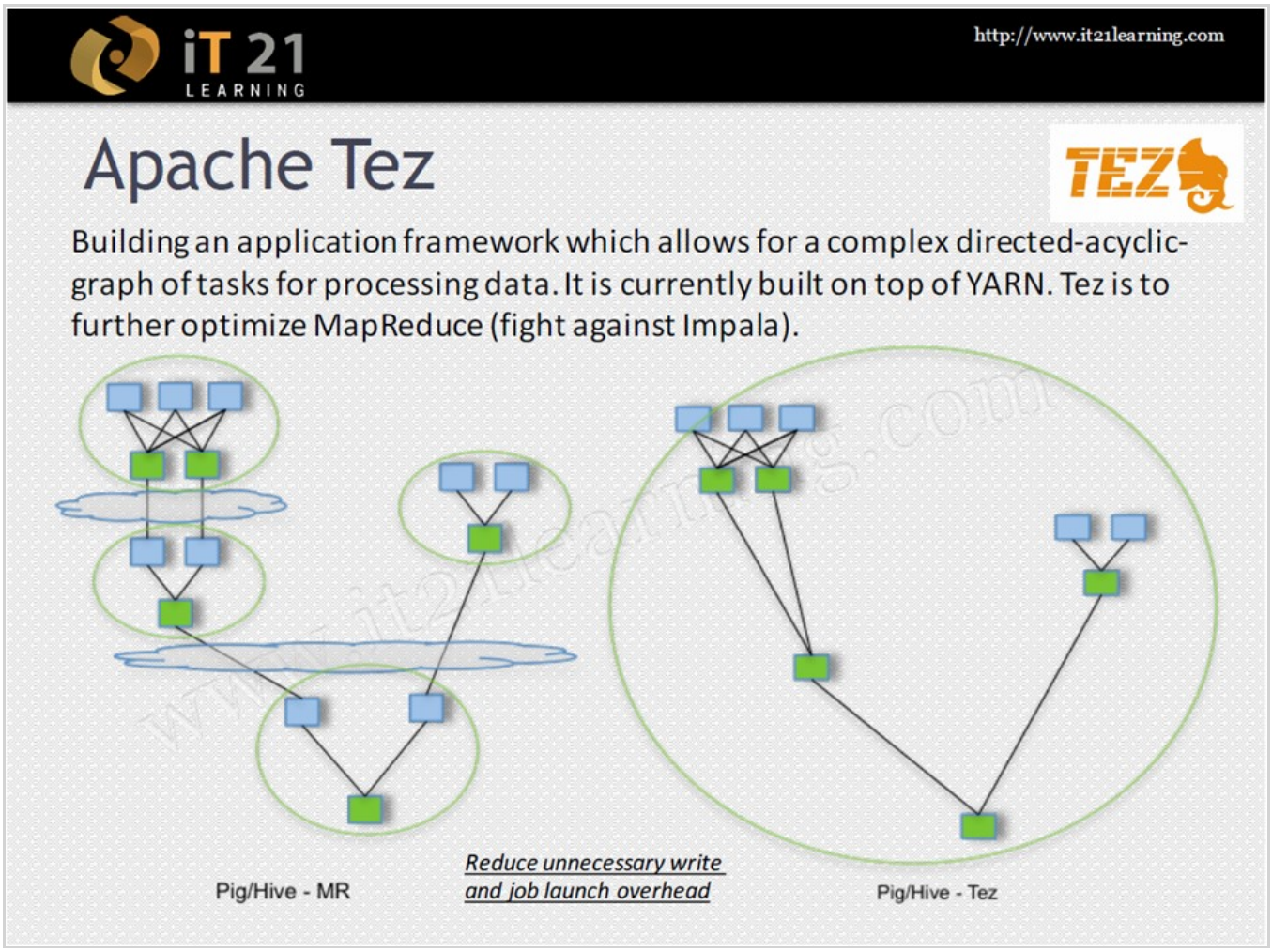


10

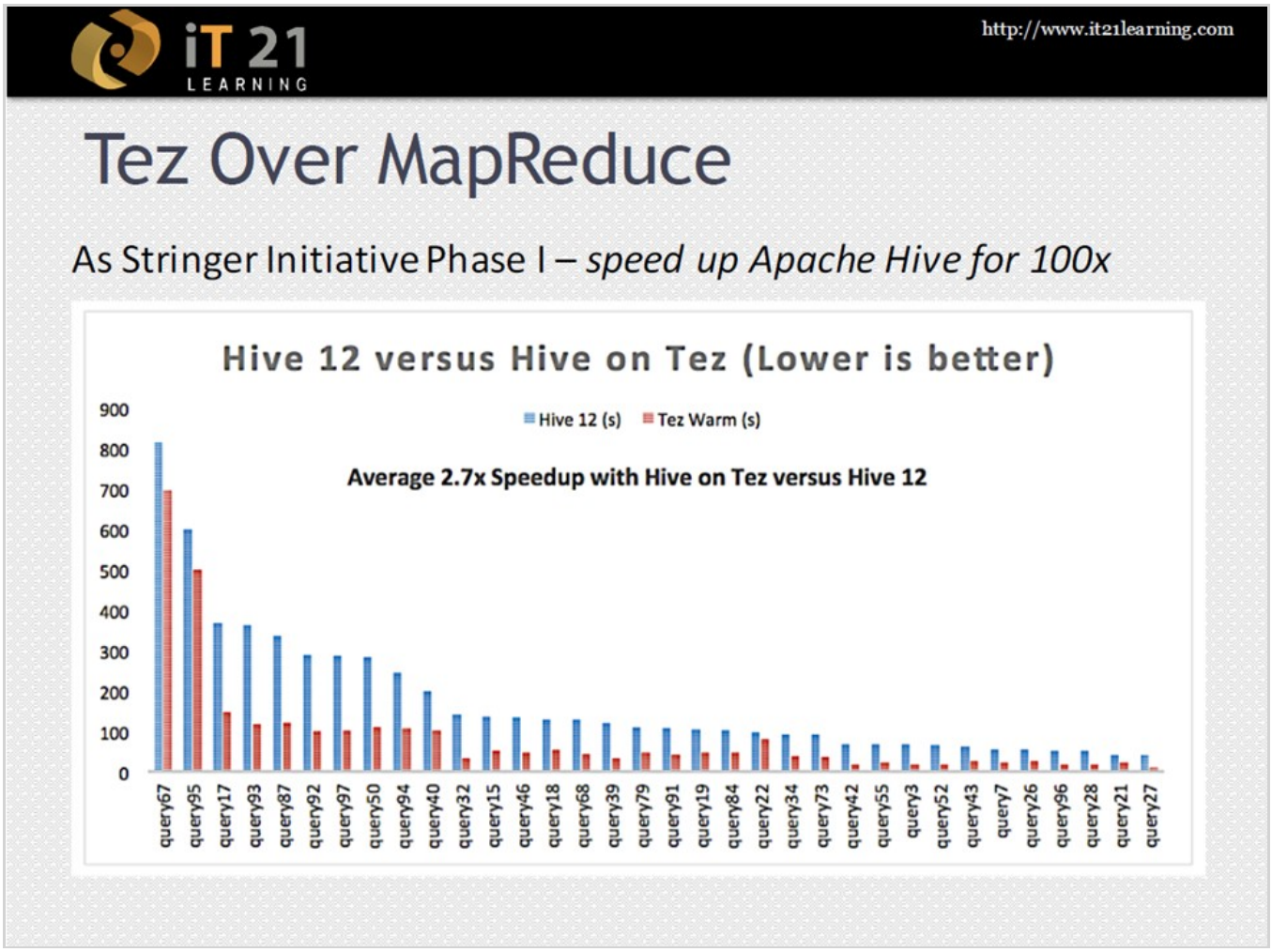

Tez
Spark
MapReduce/Yarn
HDFS
Alternative Computing
• Apache Tez
• Apache Spark
iT 21 LEARNING
http://www.it21learning.com

11

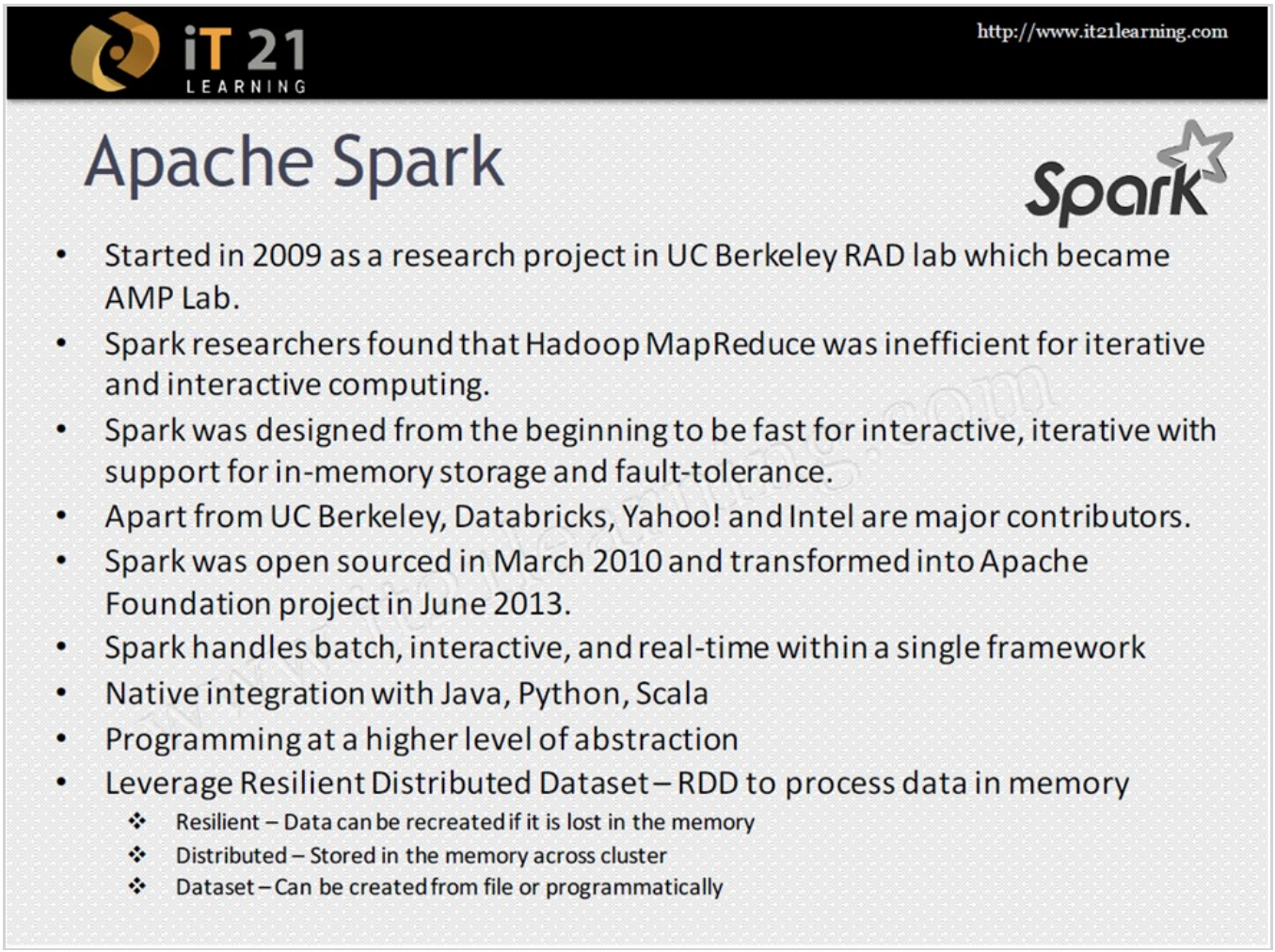
Apache Tez



12

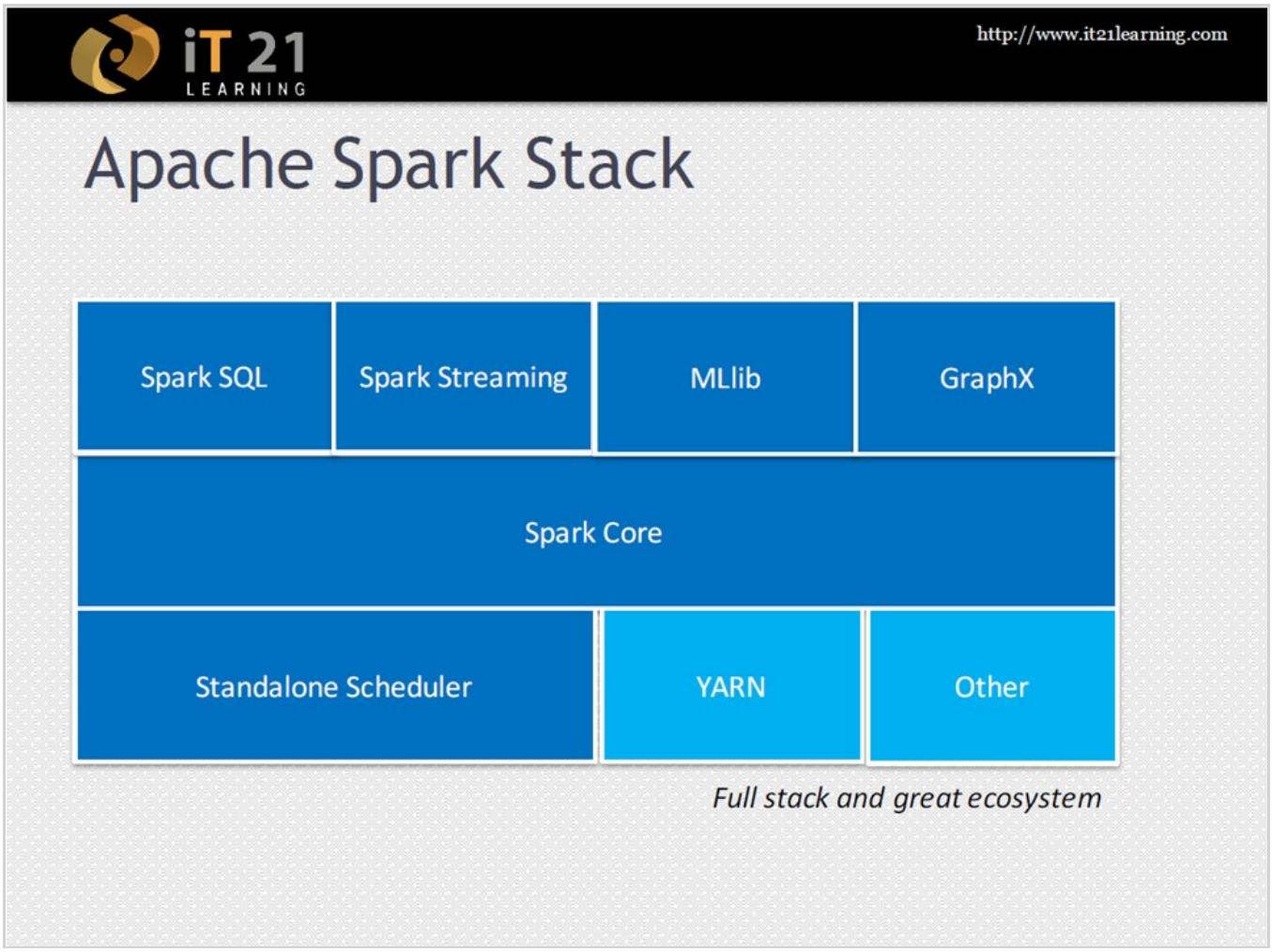

iT 21 LEARNING
http://www.it21learning.com
Tez Over MapReduce
As Stringer Initiative Phase I – speed up Apache Hive for 100x
Hive 12 versus Hive on Tez (Lower is better)
Hive 12 (s)
Tez Warm (s)
Average 2.7x Speedup with Hive on Tez versus Hive 12
900
800
700
600
500
400
300
200
100
0
query67
query95
query17
query93
query87
query92
query97
query50
query94
query40
query32
query15
query46
query18
query68
query39
query79
query91
query19
query84
query22
query34
query73
query42
query55
query3
query52
query43
query7
query26
query96
query28
query21
query27

13

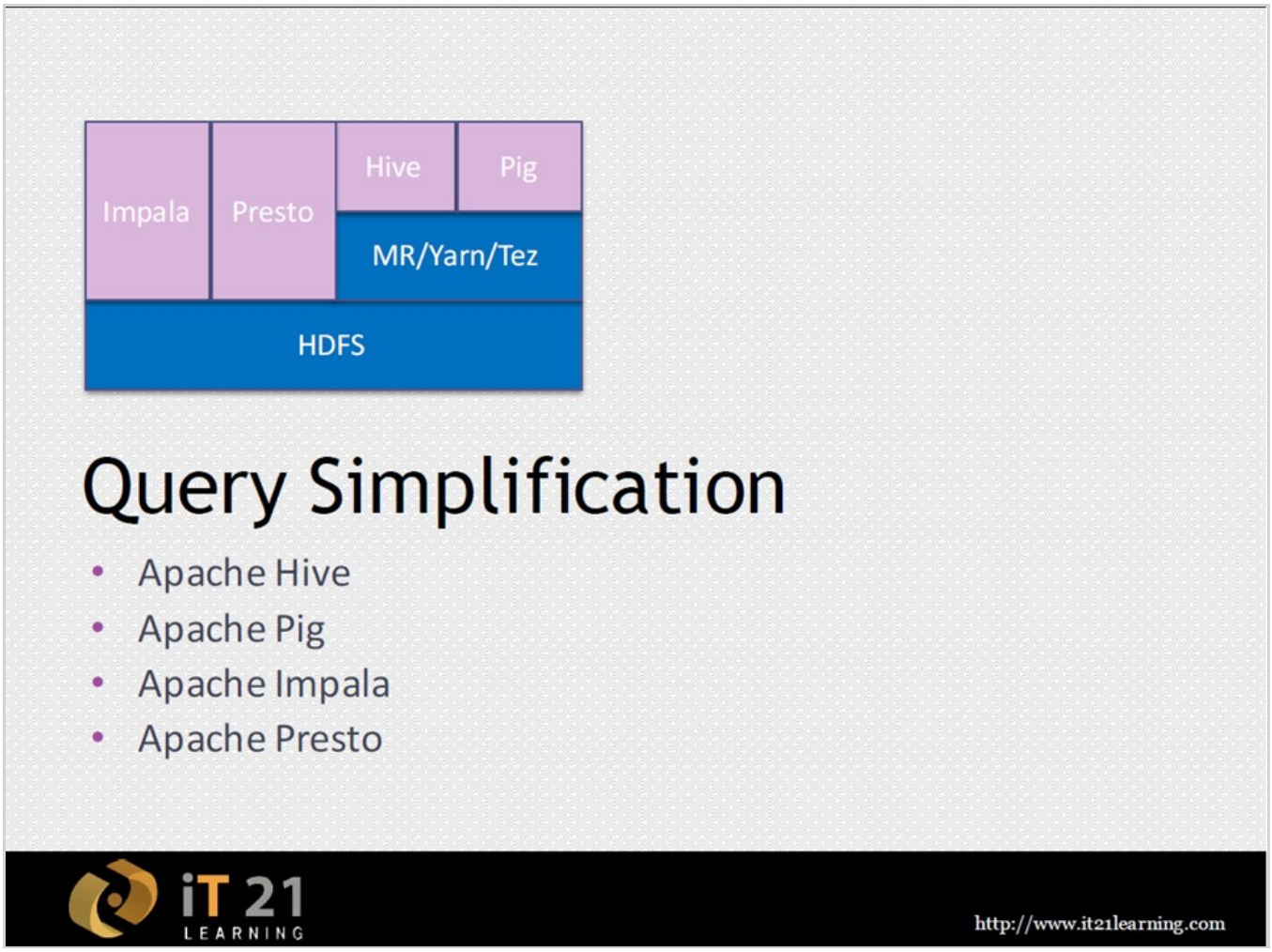

iT 21 LEARNING
http://www.it21learning.com
Apache Spark
Spark
• Started in 2009 as a research project in UC Berkeley RAD lab which became AMP Lab.
• Spark researchers found that Hadoop MapReduce was inefficient for iterative and interactive computing.
• Spark was designed from the beginning to be fast for interactive, iterative with support for in-memory storage and fault-tolerance.
• Apart from UC Berkeley, Databricks, Yahoo! and Intel are major contributors.
• Spark was open sourced in March 2010 and transformed into Apache Foundation project in June 2013.
• Spark handles batch, interactive, and real-time within a single framework
• Native integration with Java, Python, Scala
• Programming at a higher level of abstraction
• Leverage Resilient Distributed Dataset – RDD to process data in memory
❖ Resilient – Data can be recreated if it is lost in the memory
❖ Distributed – Stored in the memory across cluster
❖ Dataset – Can be created from file or programmatically

14

Apache Spark Stack



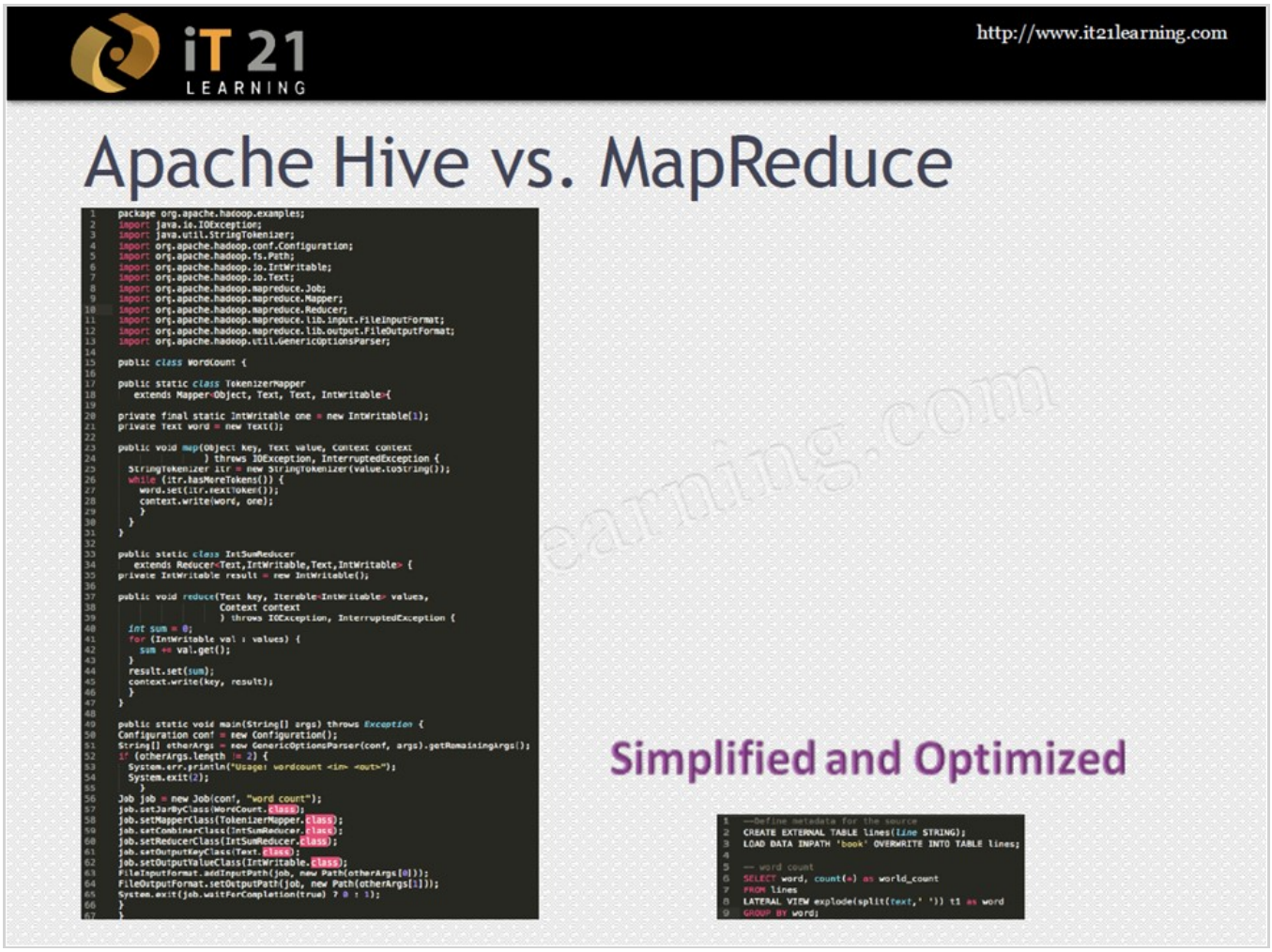
15


Impala
Presto
Hive
Pig
MR/Yarn/Tez
HDFS
Query Simplification
• Apache Hive
• Apache Pig
• Apache Impala
• Apache Presto
iT 21 LEARNING
http://www.it21learning.com

16

Apache Hive

17

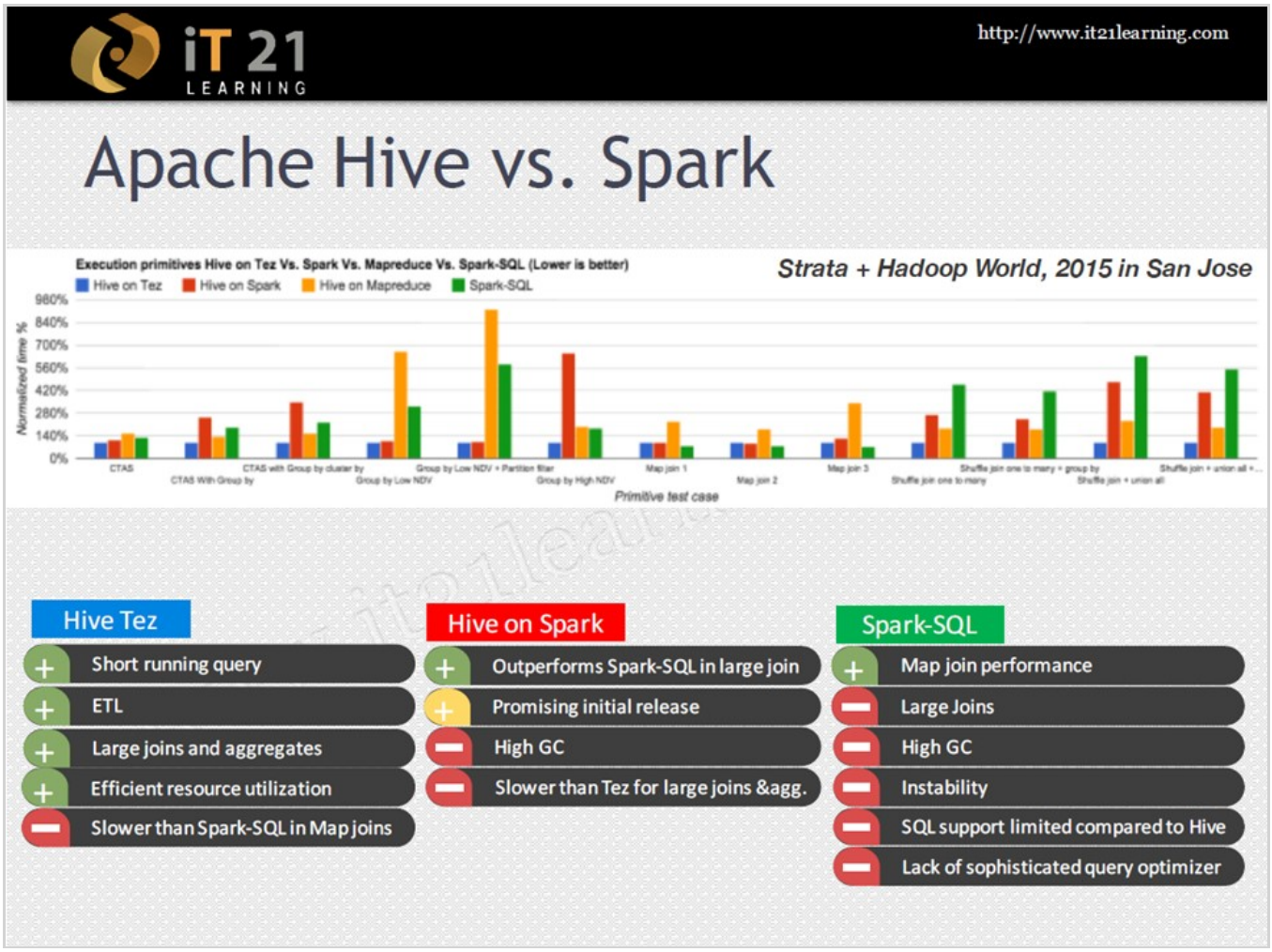



```java
package org.apache.hadoop.examples;
import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import 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;
import org.apache.hadoop.util.GenericOptionsParser;

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();
String[] otherArgs = new GenericOptionsParser(conf, args).getRemainingArgs();
if (otherArgs.length != 2) {
  System.err.println("Usage: wordcount <in> <out>");
  System.exit(2);
  }
Job job = new Job(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(otherArgs[0]));
FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));
System.exit(job.waitForCompletion(true) ? 0 : 1);
}
}
```

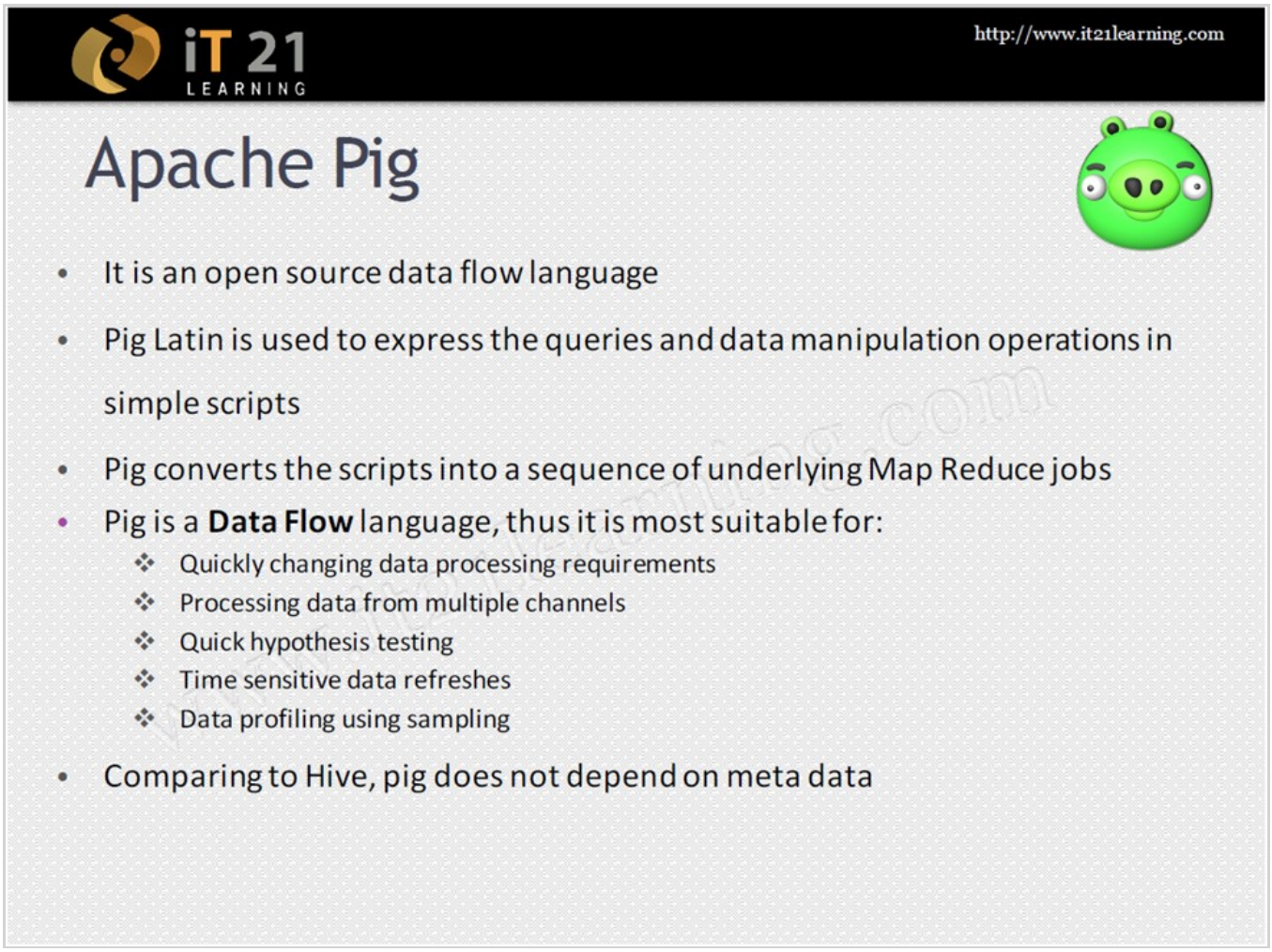
```
--Define metadata for the source
CREATE EXTERNAL TABLE lines(line STRING);
LOAD DATA INPATH 'book' OVERWRITE INTO TABLE lines;

-- word count
SELECT word, count(*) as world_count
FROM lines
LATERAL VIEW explode(split(text,' ')) t1 as word
GROUP BY word;
```

18

Apache Hive vs. Spark



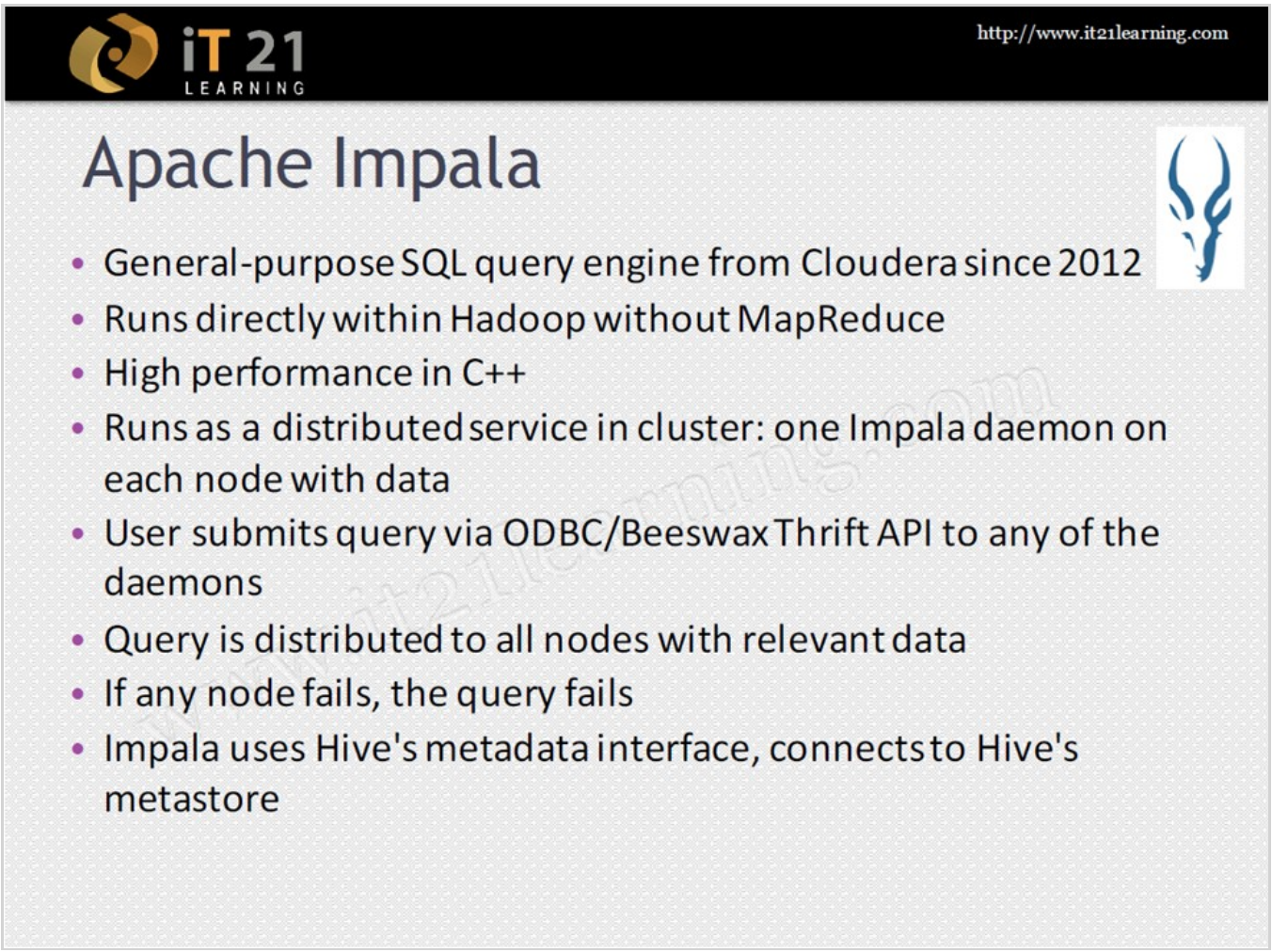
19


iT 21
LEARNING
http://www.it21learning.com
Apache Pig
It is an open source data flow language
Pig Latin is used to express the queries and data manipulation operations in simple scripts
Pig converts the scripts into a sequence of underlying Map Reduce jobs
Pig is a Data Flow language, thus it is most suitable for:
Quickly changing data processing requirements
Processing data from multiple channels
Quick hypothesis testing
Time sensitive data refreshes
Data profiling using sampling
Comparing to Hive, pig does not depend on meta data

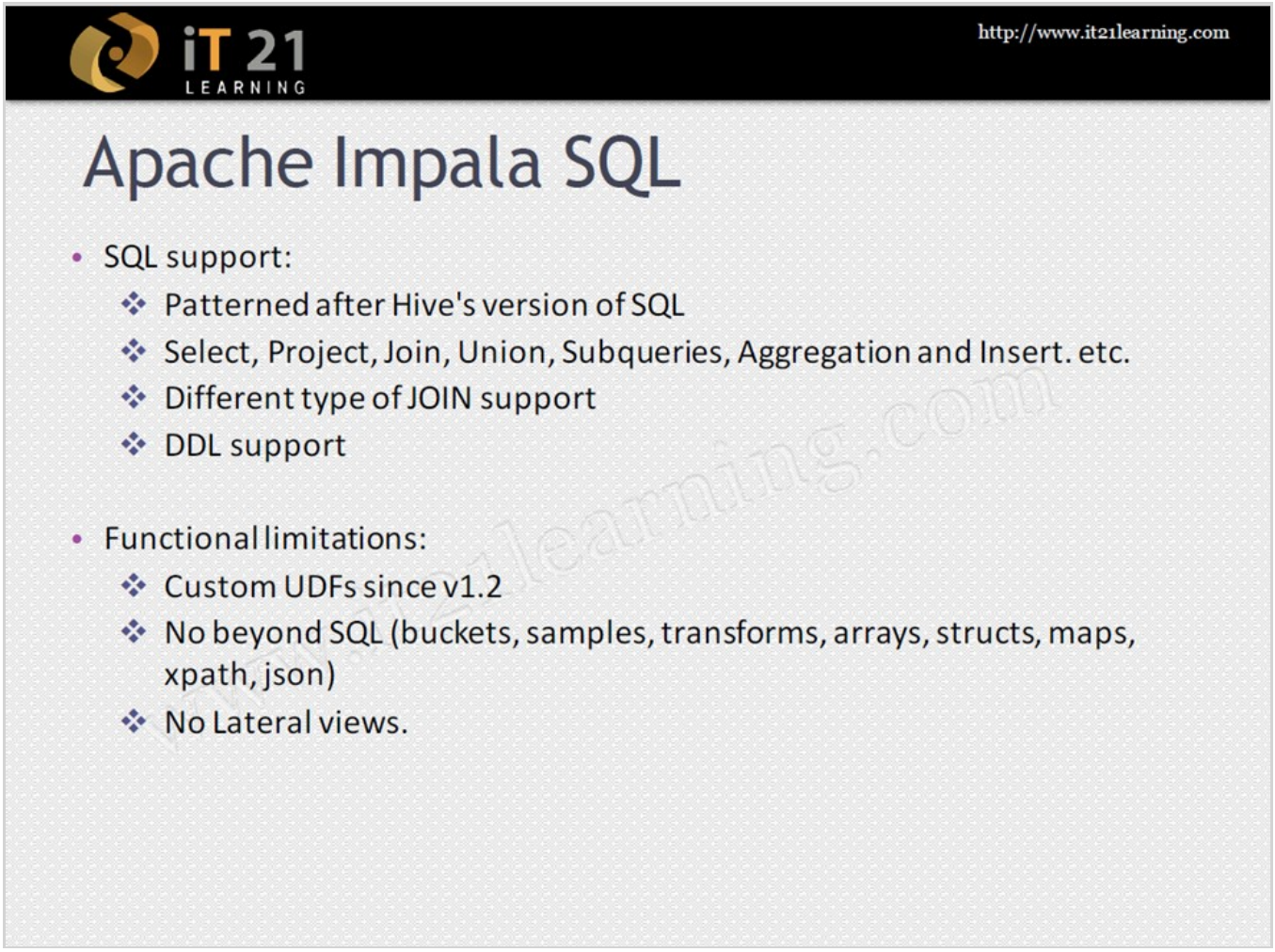
20

Apache Pig - Simple Example

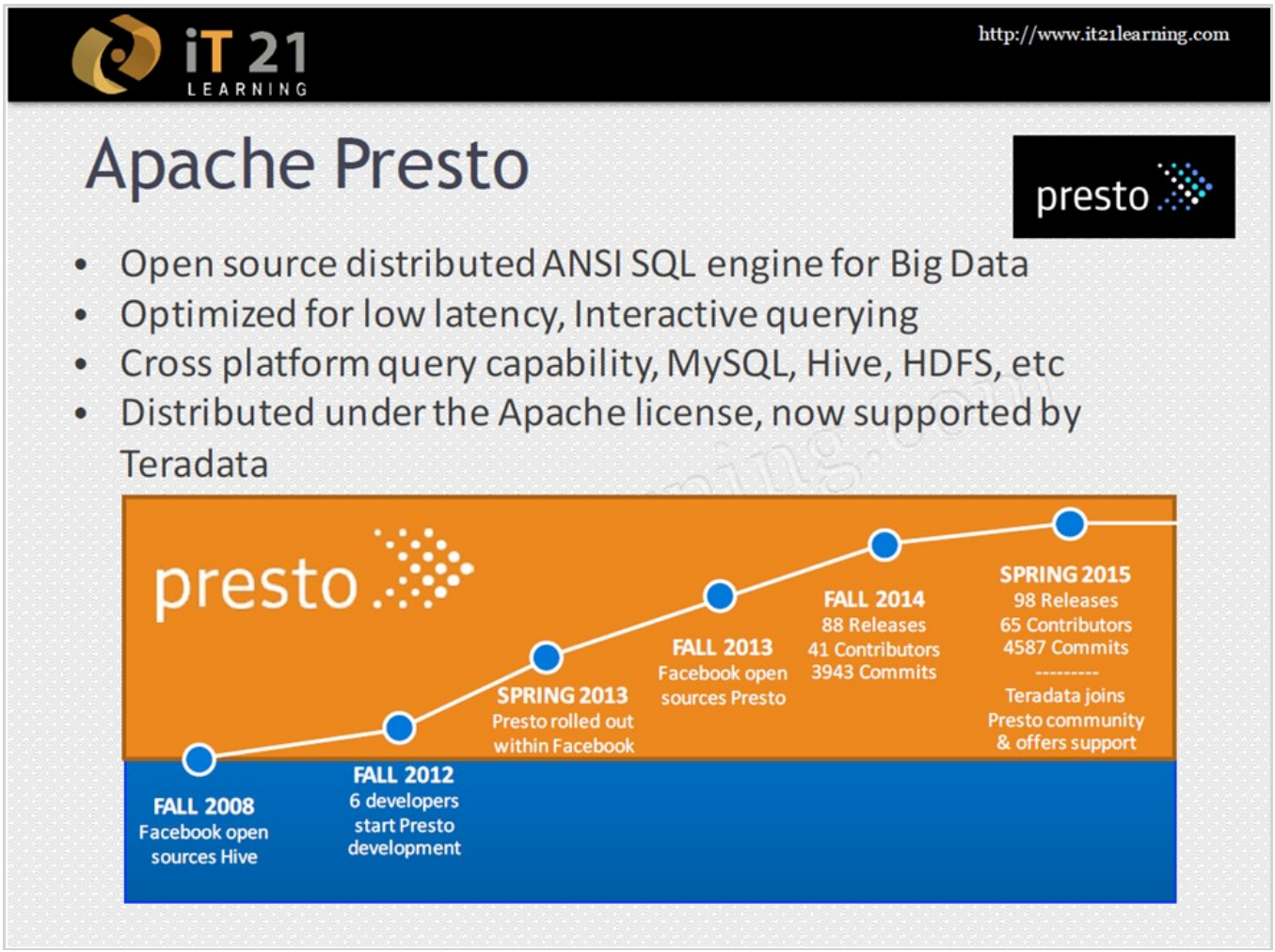
21


iT 21 LEARNING
http://www.it21learning.com
Apache Impala
• General-purpose SQL query engine from Cloudera since 2012
• Runs directly within Hadoop without MapReduce
• High performance in C++
• Runs as a distributed service in cluster: one Impala daemon on each node with data
• User submits query via ODBC/Beeswax Thrift API to any of the daemons
• Query is distributed to all nodes with relevant data
• If any node fails, the query fails
• Impala uses Hive's metadata interface, connects to Hive's metastore

22

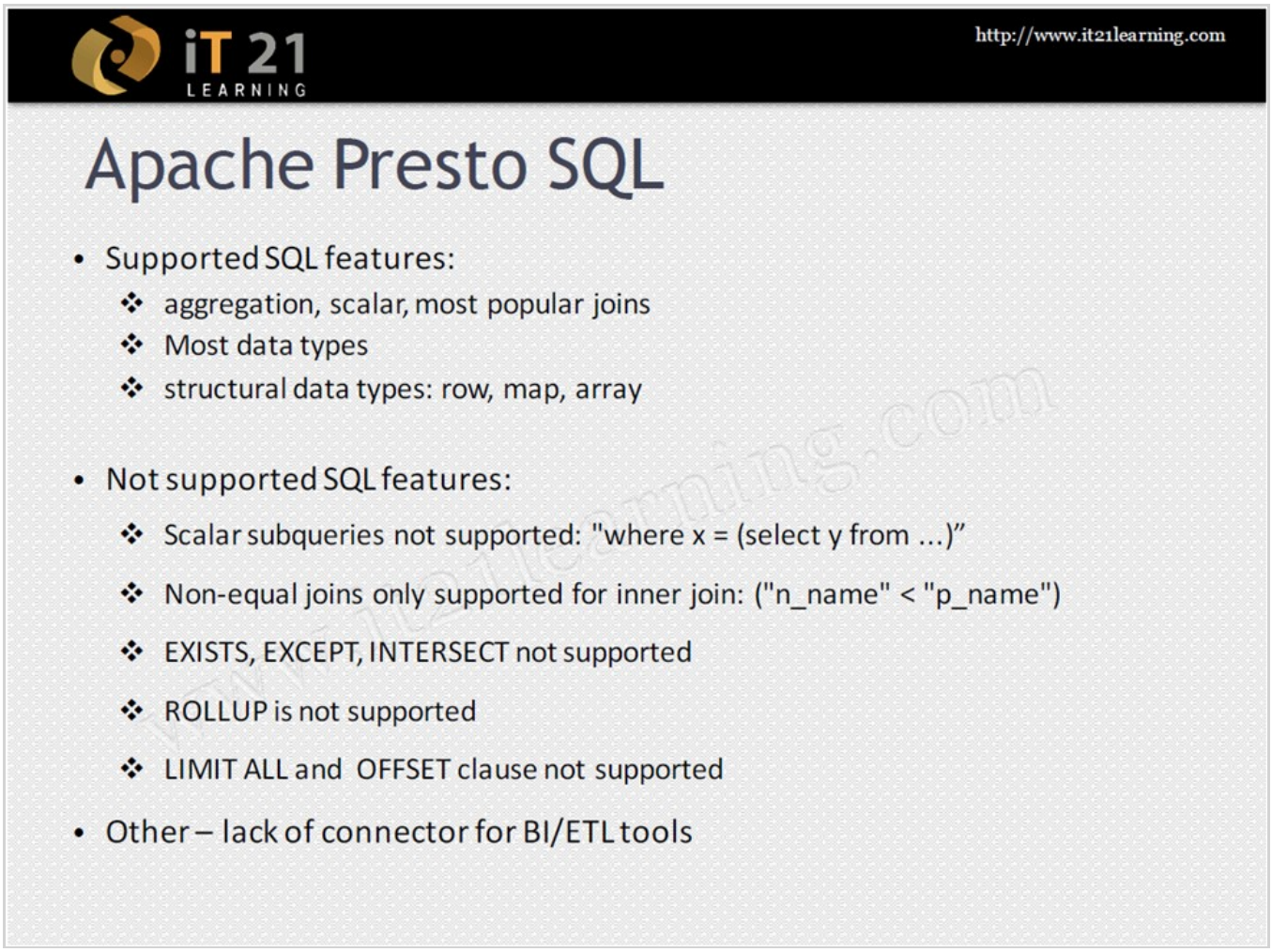

iT 21 LEARNING
http://www.it21learning.com
Apache Impala SQL
• SQL support:
❖ Patterned after Hive's version of SQL
❖ Select, Project, Join, Union, Subqueries, Aggregation and Insert. etc.
❖ Different type of JOIN support
❖ DDL support
• Functional limitations:
❖ Custom UDFs since v1.2
❖ No beyond SQL (buckets, samples, transforms, arrays, structs, maps, xpath, json)
❖ No Lateral views.

23

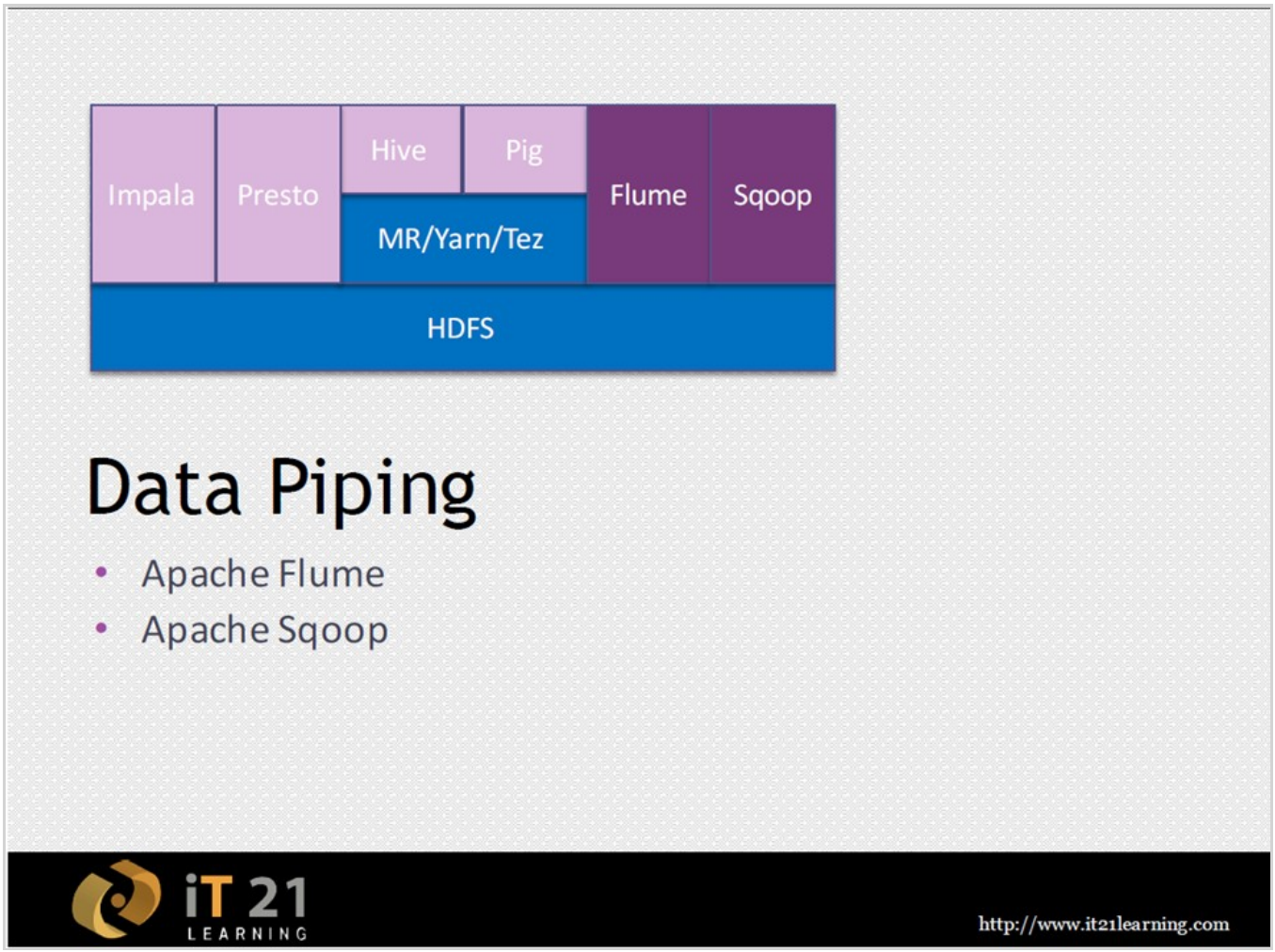

iT 21 LEARNING
http://www.it21learning.com
Apache Presto
presto
• Open source distributed ANSI SQL engine for Big Data
• Optimized for low latency, Interactive querying
• Cross platform query capability, MySQL, Hive, HDFS, etc
• Distributed under the Apache license, now supported by Teradata
presto
FALL 2008
Facebook open sources Hive
FALL 2012
6 developers start Presto development
SPRING 2013
Presto rolled out within Facebook
FALL 2013
Facebook open sources Presto
FALL 2014
88 Releases
41 Contributors
3943 Commits
SPRING 2015
98 Releases
65 Contributors
4587 Commits

Teradata joins Presto community & offers support

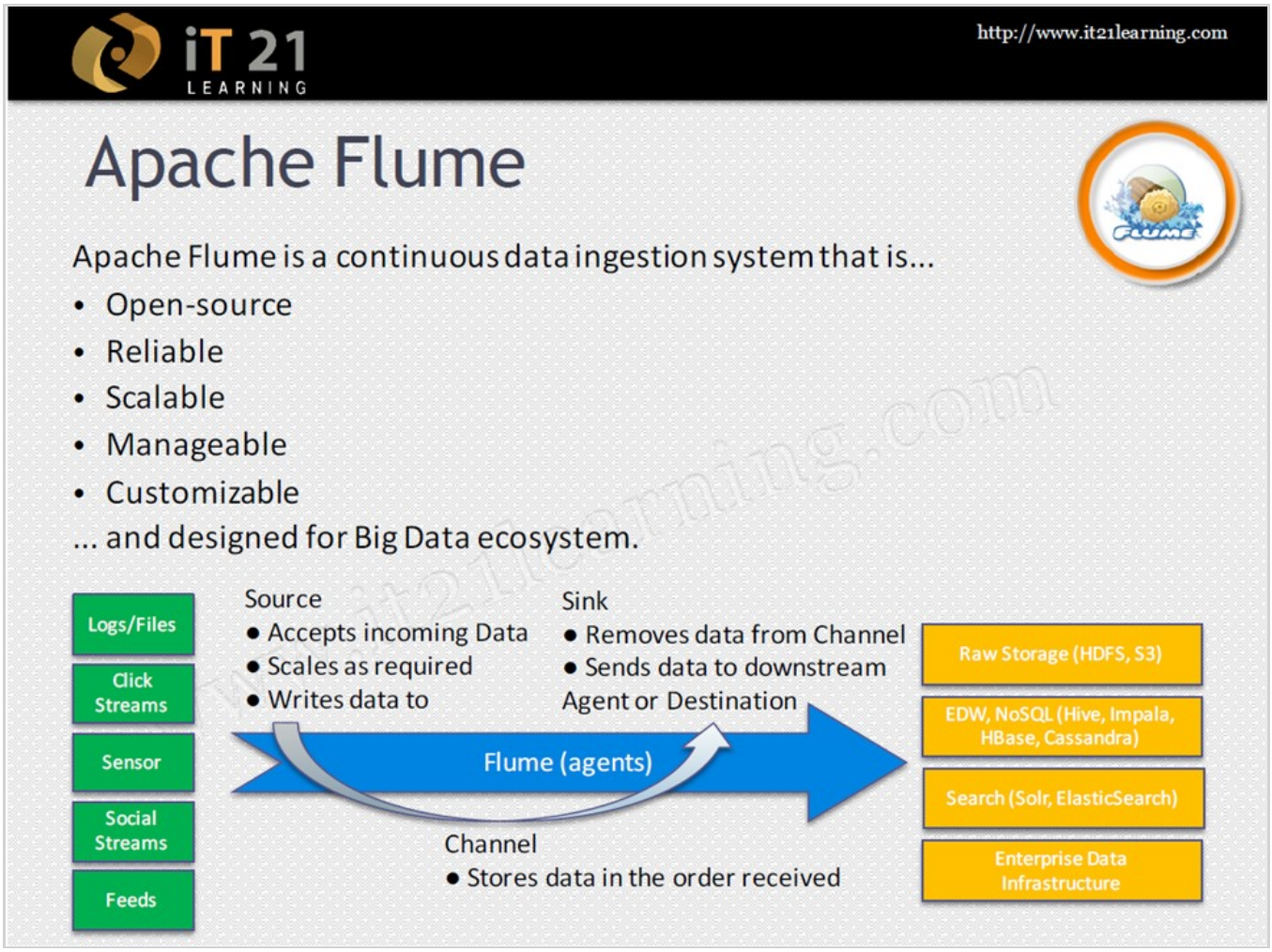
24


iT 21 LEARNING
http://www.it21learning.com
Apache Presto SQL
• Supported SQL features:
❖ aggregation, scalar, most popular joins
❖ Most data types
❖ structural data types: row, map, array
• Not supported SQL features:
❖ Scalar subqueries not supported: "where x = (select y from ...)"
❖ Non-equal joins only supported for inner join: ("n_name" < "p_name")
❖ EXISTS, EXCEPT, INTERSECT not supported
❖ ROLLUP is not supported
❖ LIMIT ALL and OFFSET clause not supported
• Other – lack of connector for BI/ETL tools

25


Impala
Presto
Hive
Pig
MR/Yarn/Tez
Flume
Sqoop
HDFS
Data Piping
• Apache Flume
• Apache Sqoop
iT 21 LEARNING
http://www.it21learning.com

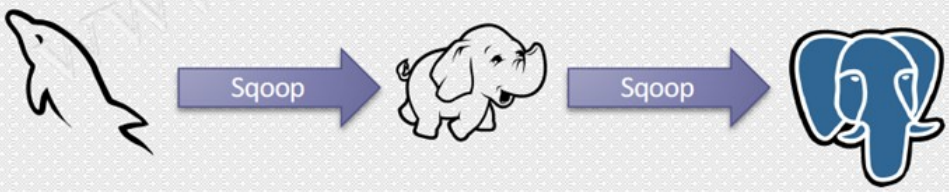
26


iT 21 LEARNING
http://www.it21learning.com
Apache Flume
Apache Flume is a continuous data ingestion system that is...
• Open-source
• Reliable
• Scalable
• Manageable
• Customizable
... and designed for Big Data ecosystem.
Logs/Files
Click Streams
Sensor
Social Streams
Feeds
Source
• Accepts incoming Data
• Scales as required
• Writes data to
Sink
• Removes data from Channel
• Sends data to downstream Agent or Destination
Flume (agents)
Channel
• Stores data in the order received
Raw Storage (HDFS, S3)
EDW, NoSQL (Hive, Impala, HBase, Cassandra)
Search (Solr, ElasticSearch)
Enterprise Data Infrastructure

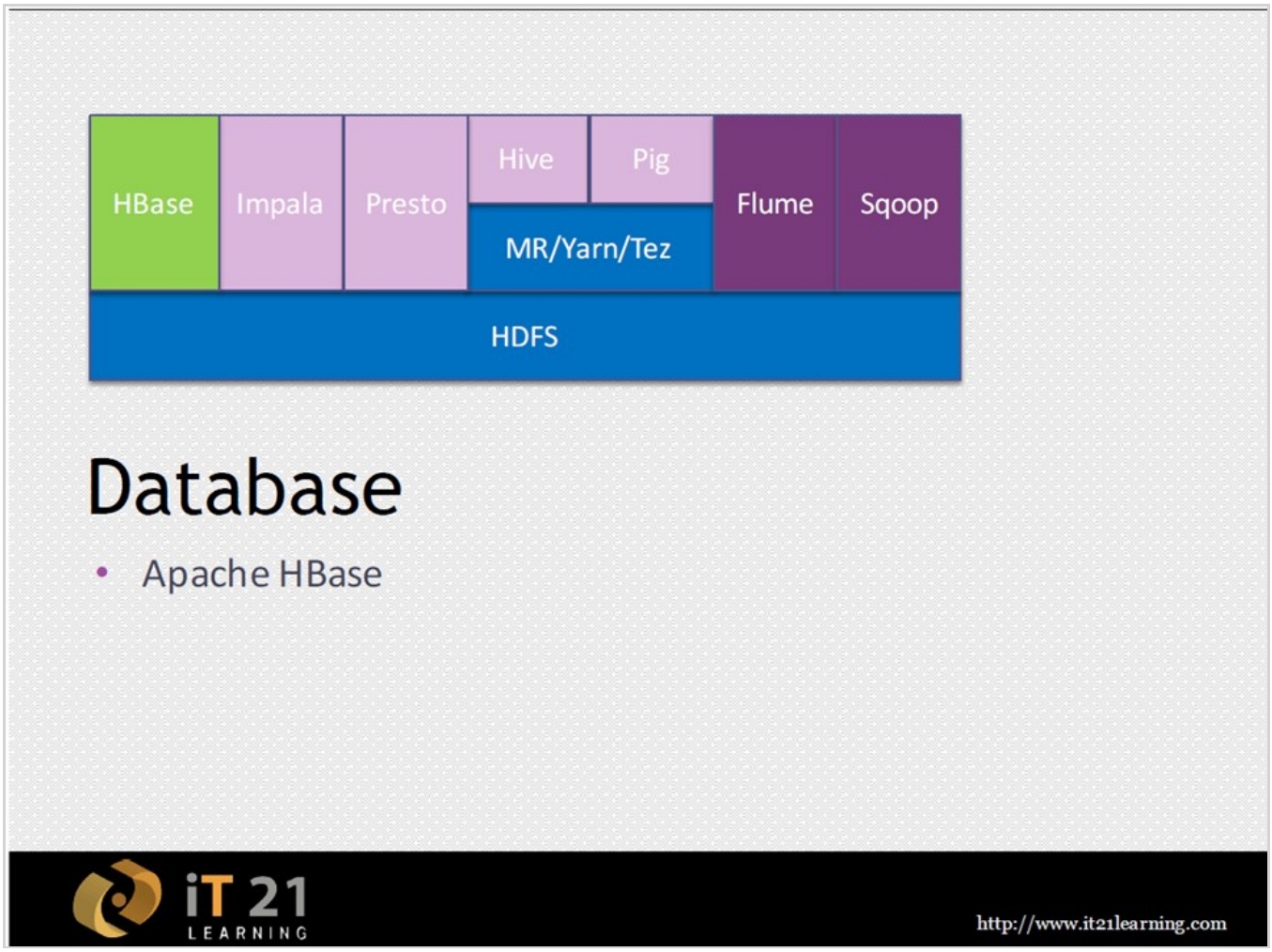
27

Apache Sqoop

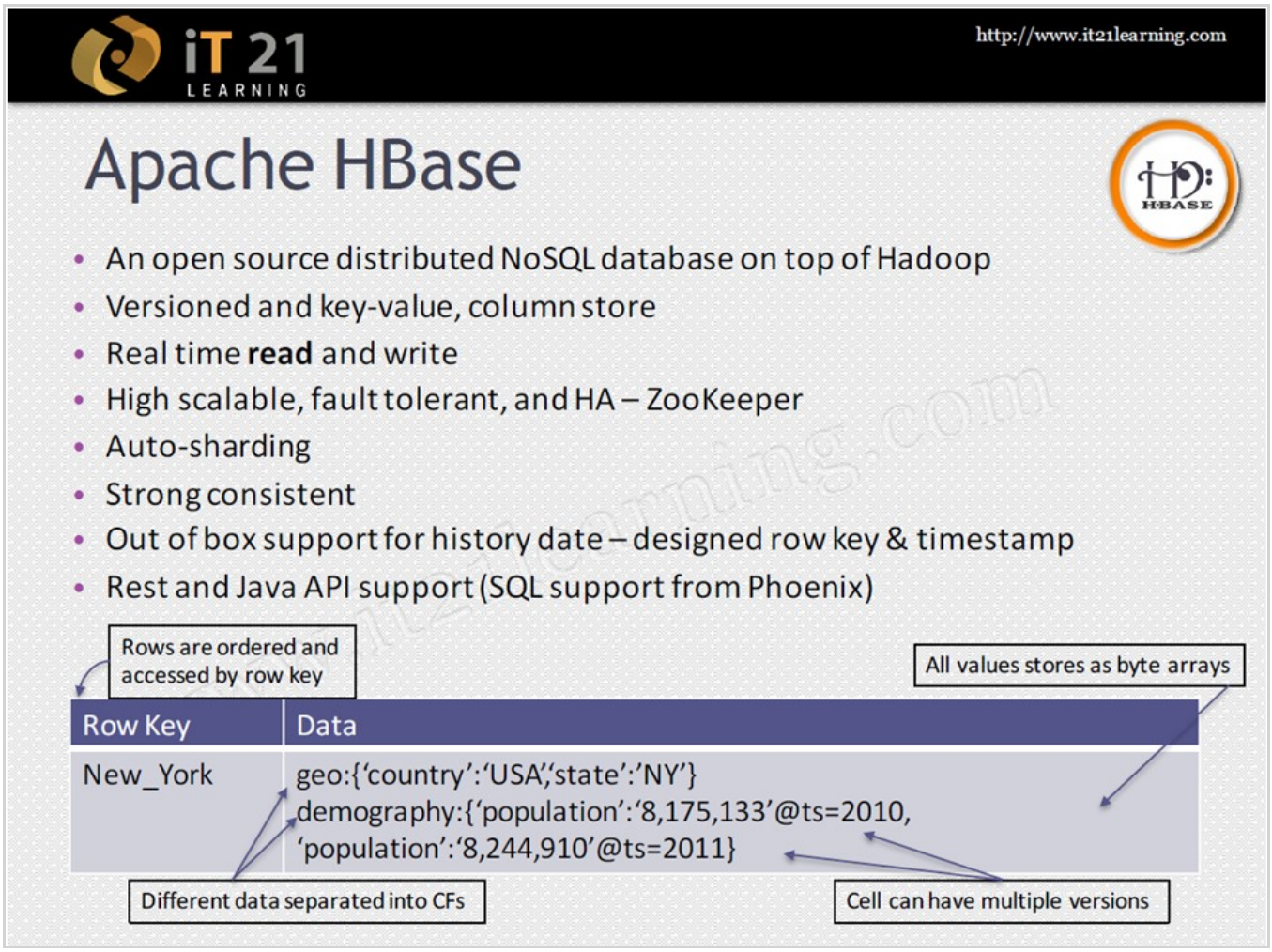
- Apache Top-Level Project
- Data bridge to Hadoop
- Tool to export/import data from relational databases to Hadoop
- Teradata, MySQL, PostgreSQL, Oracle, Netezza, DB2 are supported
- HDFS (text, sequence file), Hive, HBase, Avro are supported
- Sqoop convert job into mapreduce jobs
- Sqoop has good performance for transfer big volume data
- Not an ETL tool, but support simple transfer tracking support



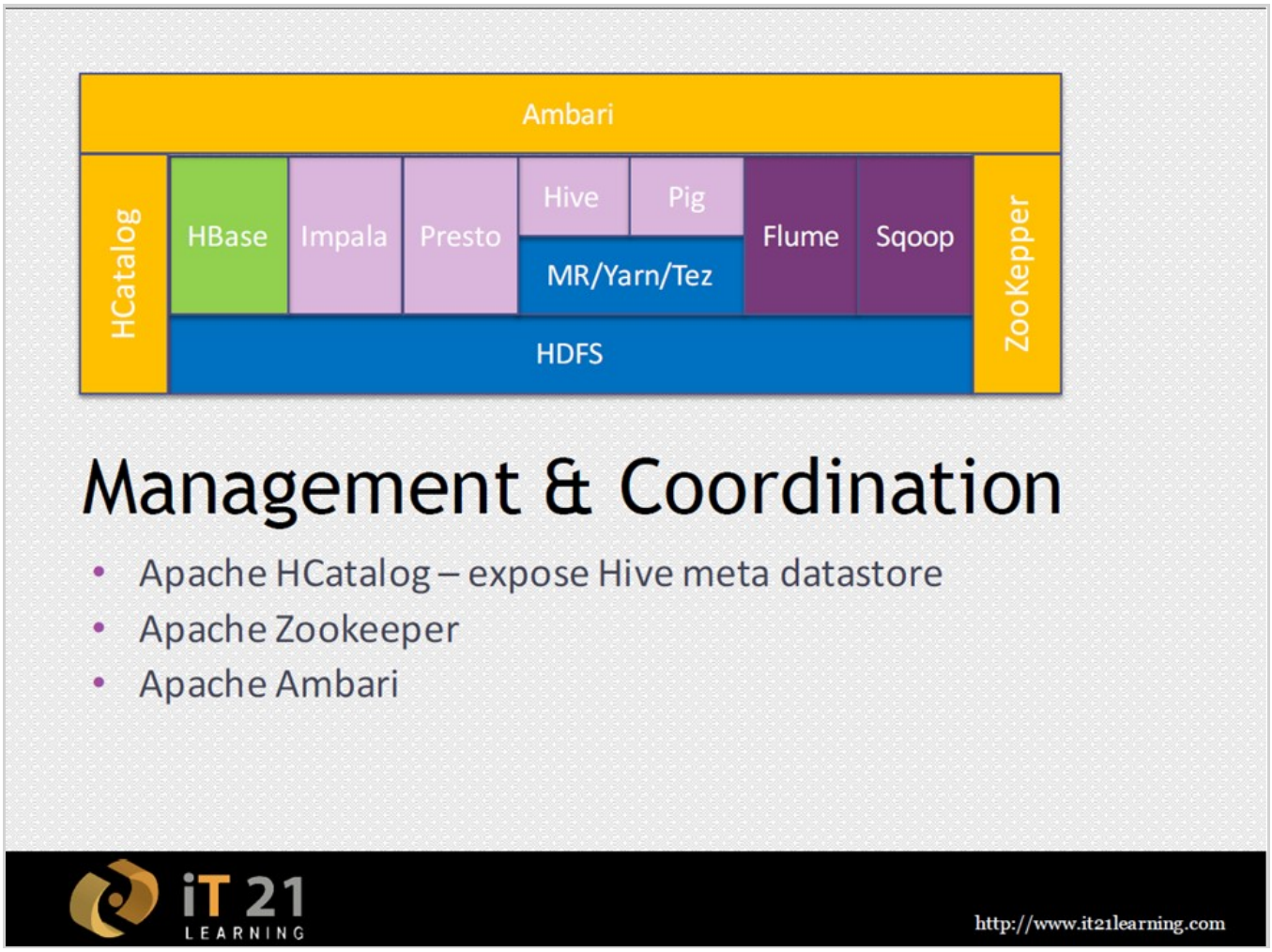
28


HBase
Impala
Presto
Hive
Pig
MR/Yarn/Tez
Flume
Sqoop
HDFS
Database
Apache HBase
iT 21 LEARNING
http://www.it21learning.com

29


iT 21 LEARNING
http://www.it21learning.com
Apache HBase
HBASE
An open source distributed NoSQL database on top of Hadoop
Versioned and key-value, column store
Real time read and write
High scalable, fault tolerant, and HA – ZooKeeper
Auto-sharding
Strong consistent
Out of box support for history date – designed row key & timestamp
Rest and Java API support (SQL support from Phoenix)
Rows are ordered and accessed by row key
All values stores as byte arrays
Row Key
Data
New_York
geo:{'country':'USA','state':'NY'}
demography:{'population':'8,175,133'@ts=2010,
'population':'8,244,910'@ts=2011}
Different data separated into CFs
Cell can have multiple versions

30


Ambari
HCatalog
HBase
Impala
Presto
Hive
Pig
MR/Yarn/Tez
Flume
Sqoop
ZooKepper
HDFS
Management & Coordination
• Apache HCatalog – expose Hive meta datastore
• Apache Zookeeper
• Apache Ambari
iT 21 LEARNING
http://www.it21learning.com

31

Apache ZooKeeper

- An open source, high-performance coordination service for distributed applications.
- Exposes common services in simple interface:
  - naming
  - configuration management
  - locks & synchronization
  - group services
- Build according to your requirement for specific needs.
- *It can be used in*
  - Locking - allow for serialized access to a shared resource
  - Synchronization – synchronizing shared resource
  - Configuration Management – centralize distributed system
  - Leader Election – dynamically leader selection

32

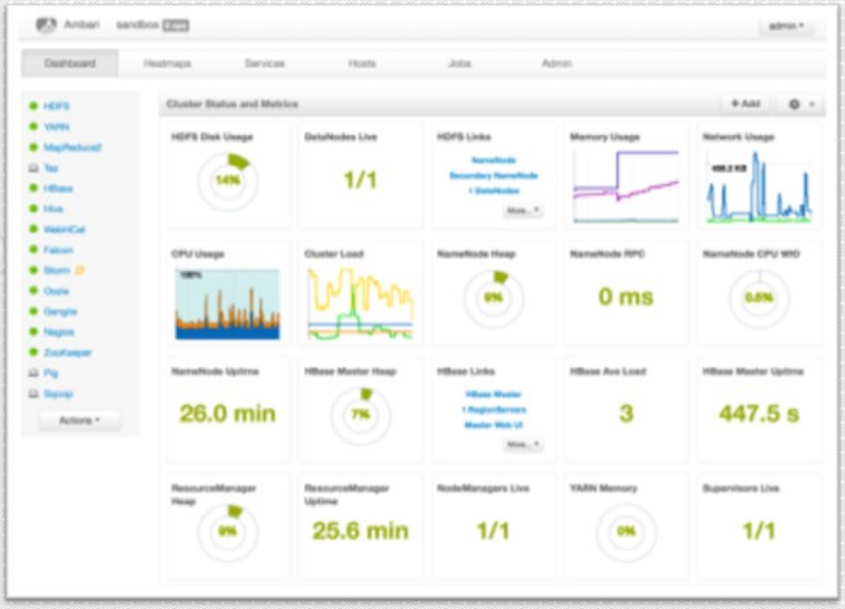
iT 21 LEARNING

http://www.it21learning.com

Apache Ambari

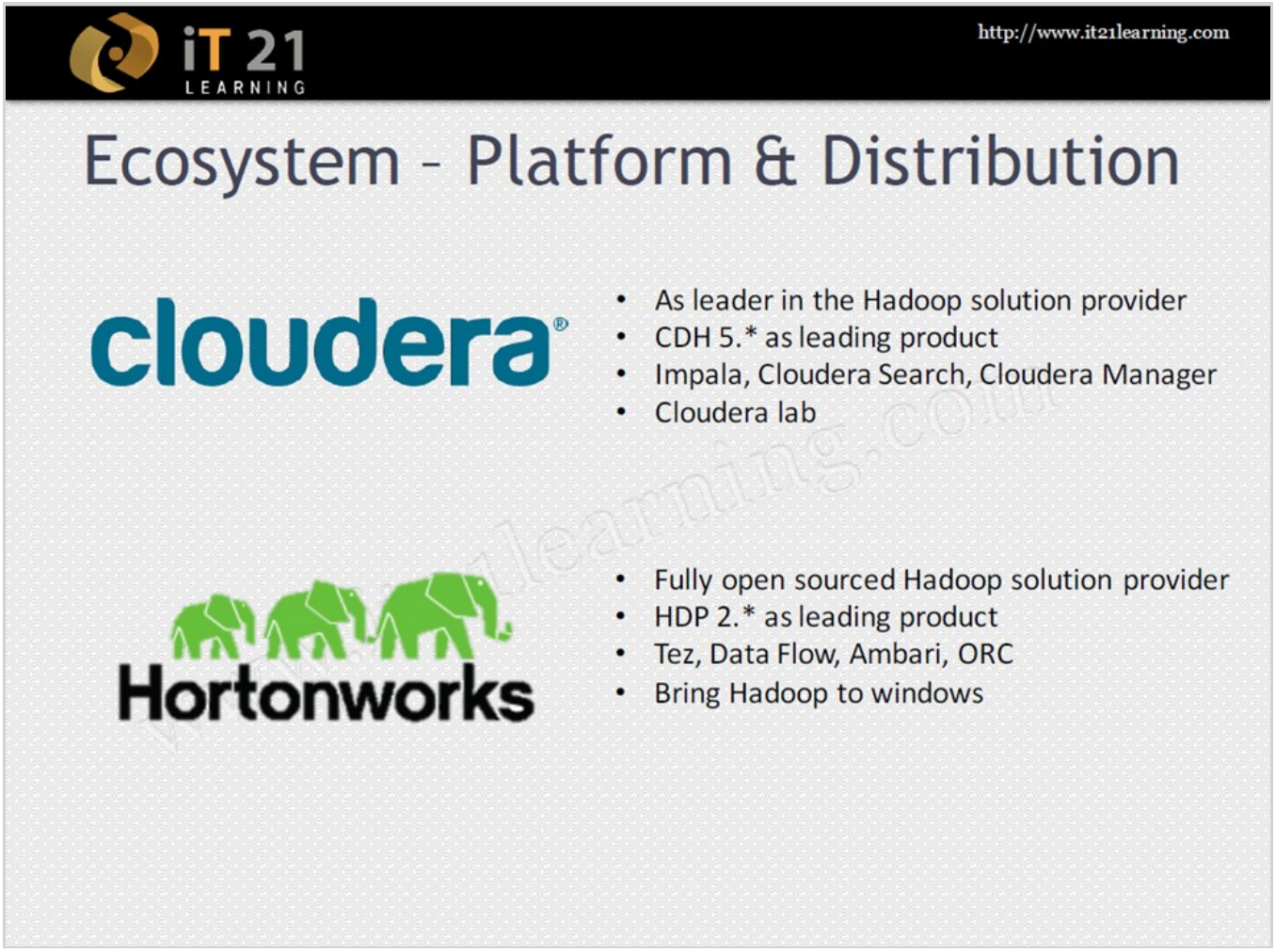
Apache Ambari is a platform to provision, manage and monitor Hadoop clusters – a free version of Cloudera Manager.

- Integrate with the Enterprise Robust API for integration with existing enterprise systems, such as Teradata Viewpoint and Microsoft Sys Center.
- Extend for the Ecosystem Provide extensible platform for Customers, Partners and the Community to, such as Stacks and Views.



33

Ecosystem - Platform & Distribution (1)

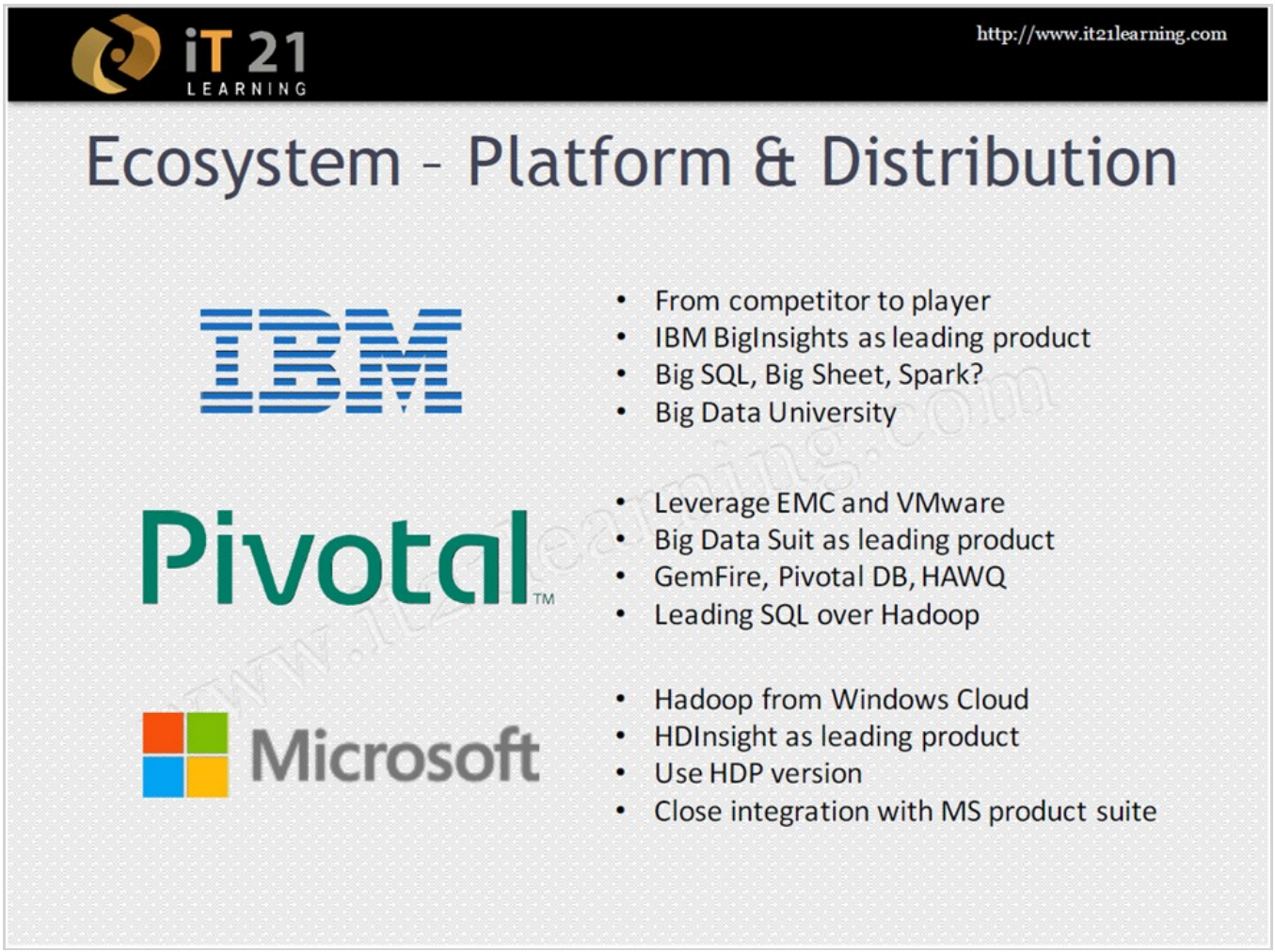


34

Ecosystem - Platform & Distribution (2)

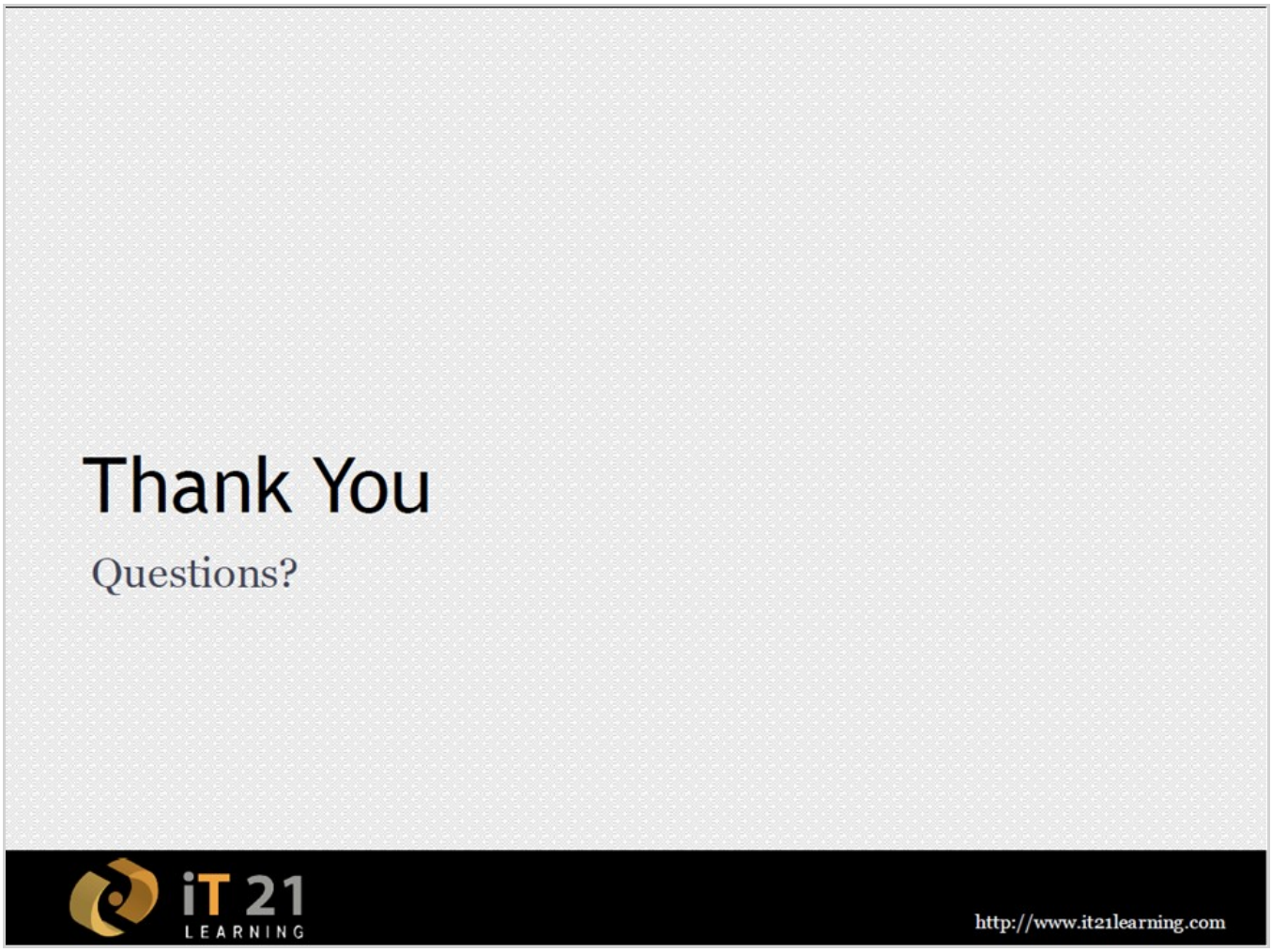
35

Ecosystem - Platform & Distribution (3)



36

Question - Thank you

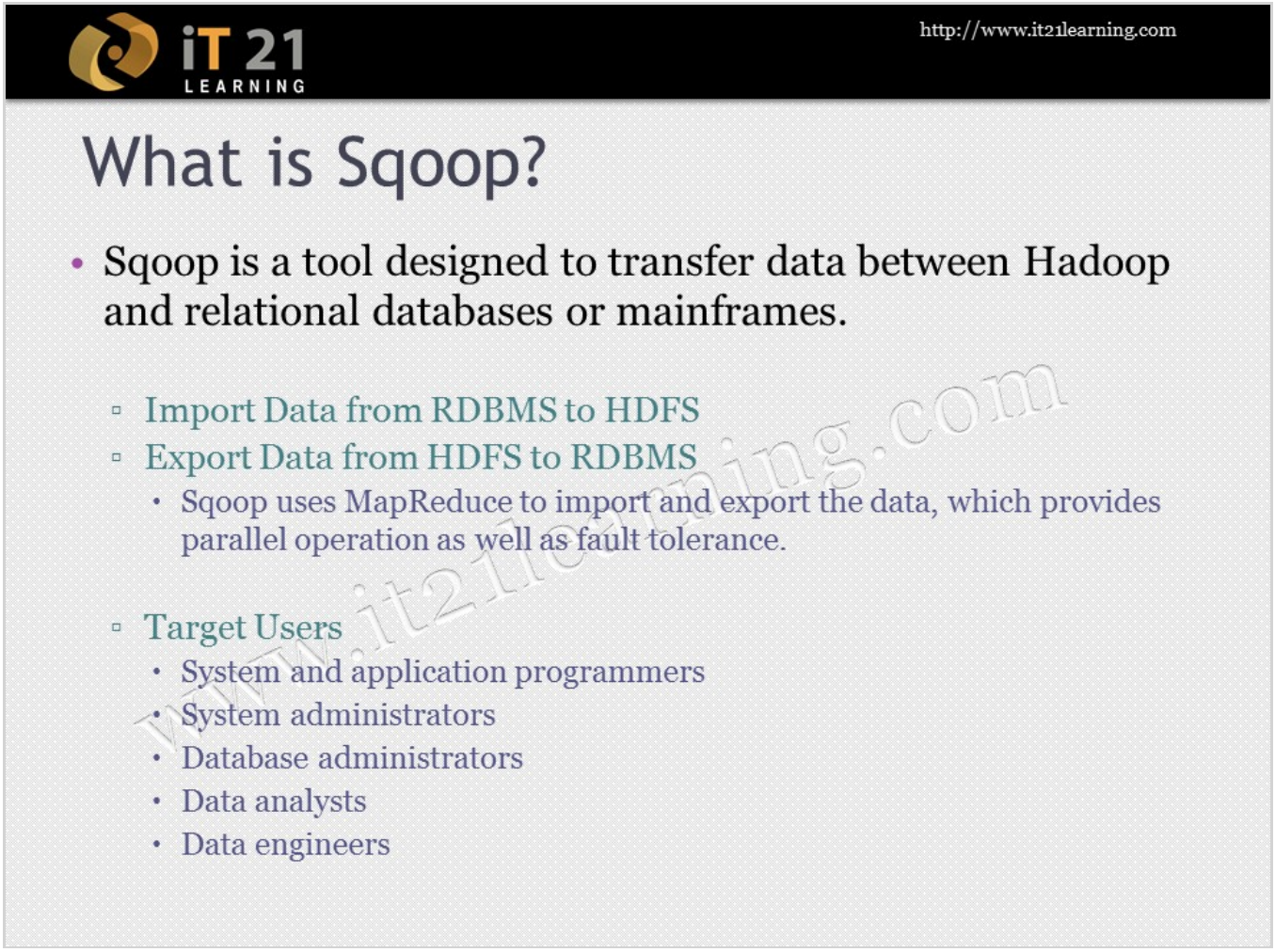


37

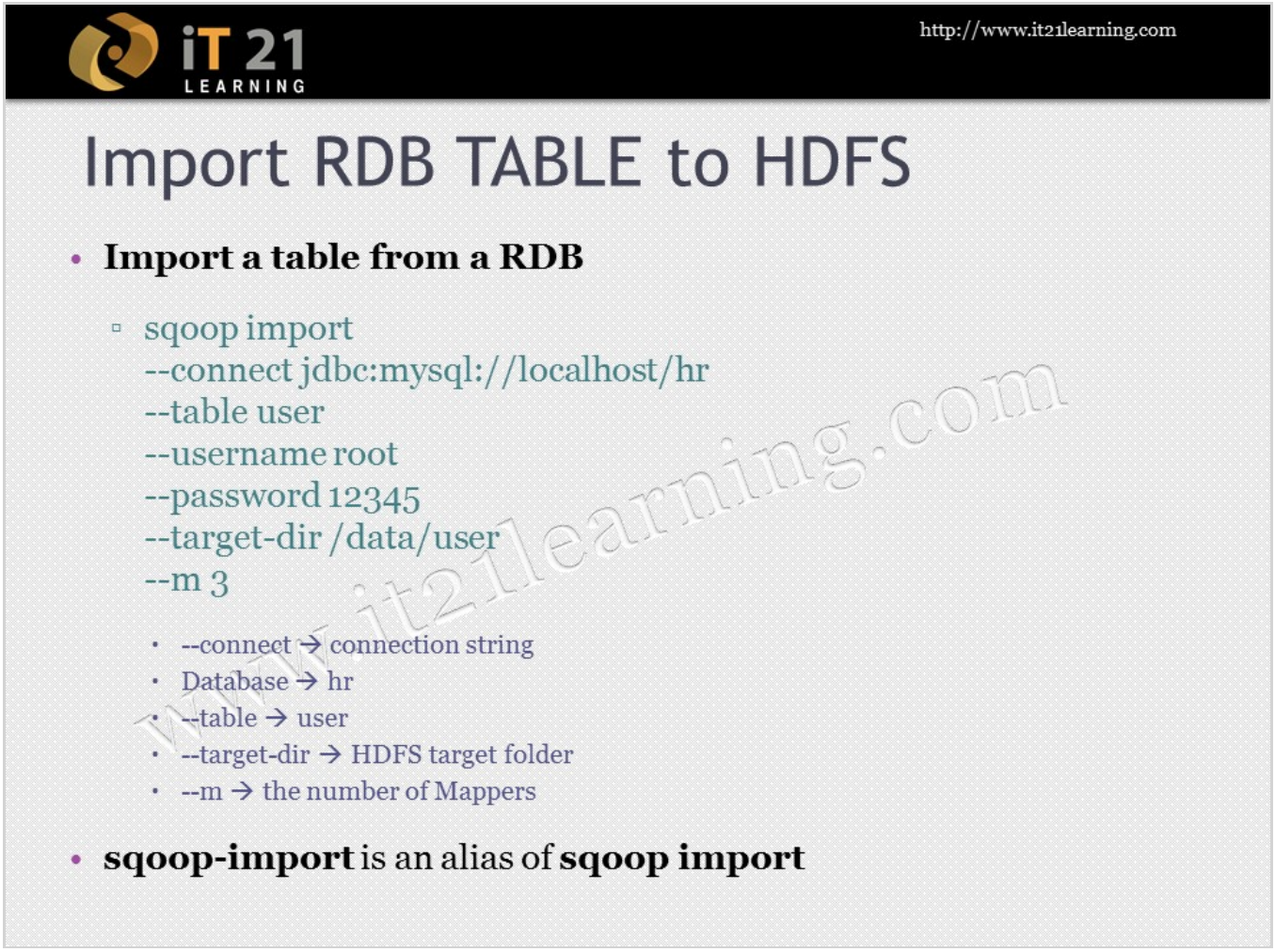
Appendix B - Hadoop Data Tool - Apache Sqoop


Hadoop Data Tool - Sqoop
Wayne Shen
iT 21
LEARNING
http://www.it21learning.com

1


iT 21 LEARNING
http://www.it21learning.com
What is Sqoop?
• Sqoop is a tool designed to transfer data between Hadoop and relational databases or mainframes.
▫ Import Data from RDBMS to HDFS
▫ Export Data from HDFS to RDBMS
• Sqoop uses MapReduce to import and export the data, which provides parallel operation as well as fault tolerance.
▫ Target Users
• System and application programmers
• System administrators
• Database administrators
• Data analysts
• Data engineers

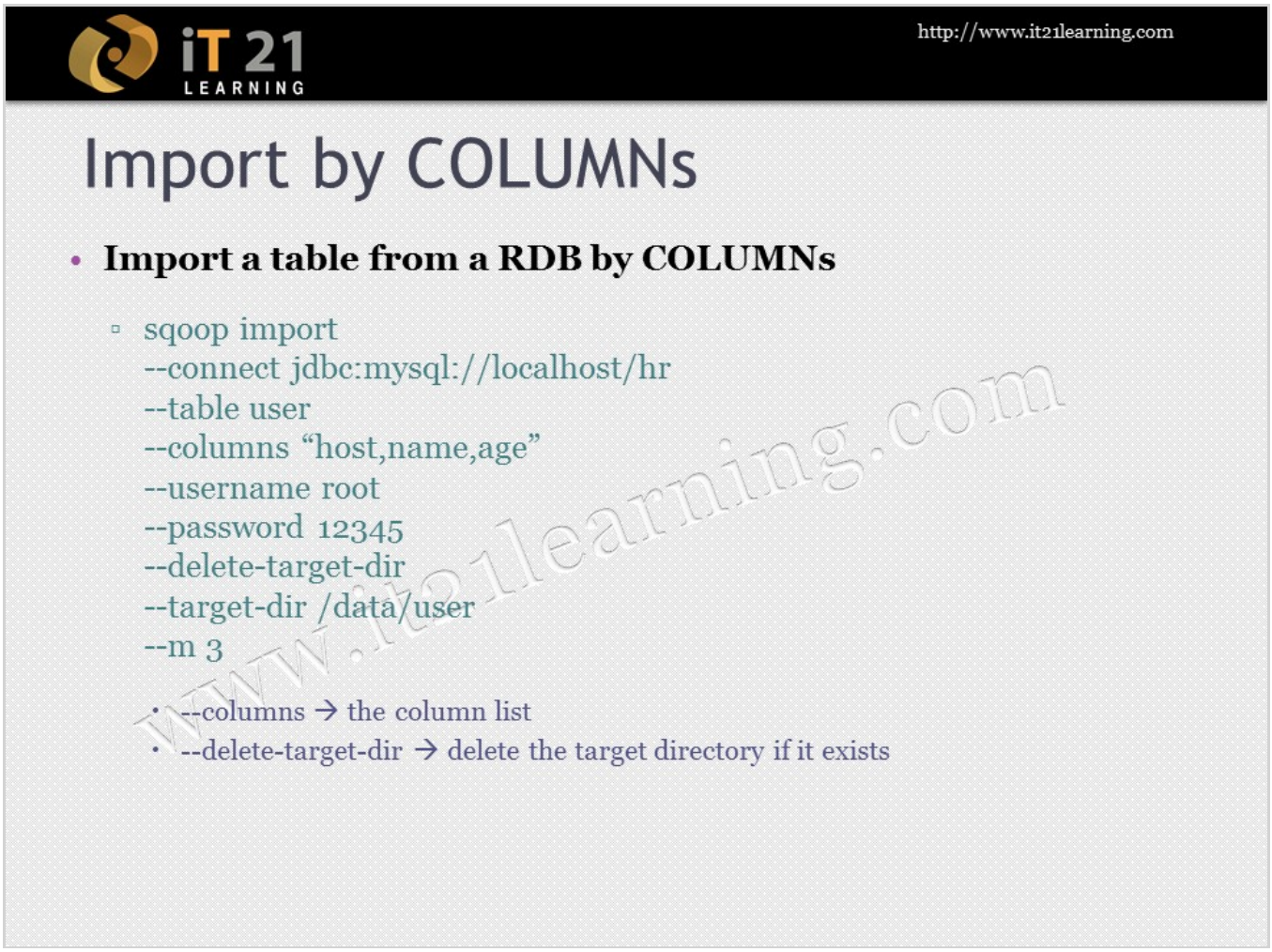
2


iT 21 LEARNING
http://www.it21learning.com
Import RDB TABLE to HDFS
Import a table from a RDB
sqoop import
--connect jdbc:mysql://localhost/hr
--table user
--username root
--password 12345
--target-dir /data/user
--m 3
--connect → connection string
Database → hr
--table → user
--target-dir → HDFS target folder
--m → the number of Mappers
sqoop-import is an alias of sqoop import
www.it21learning.com

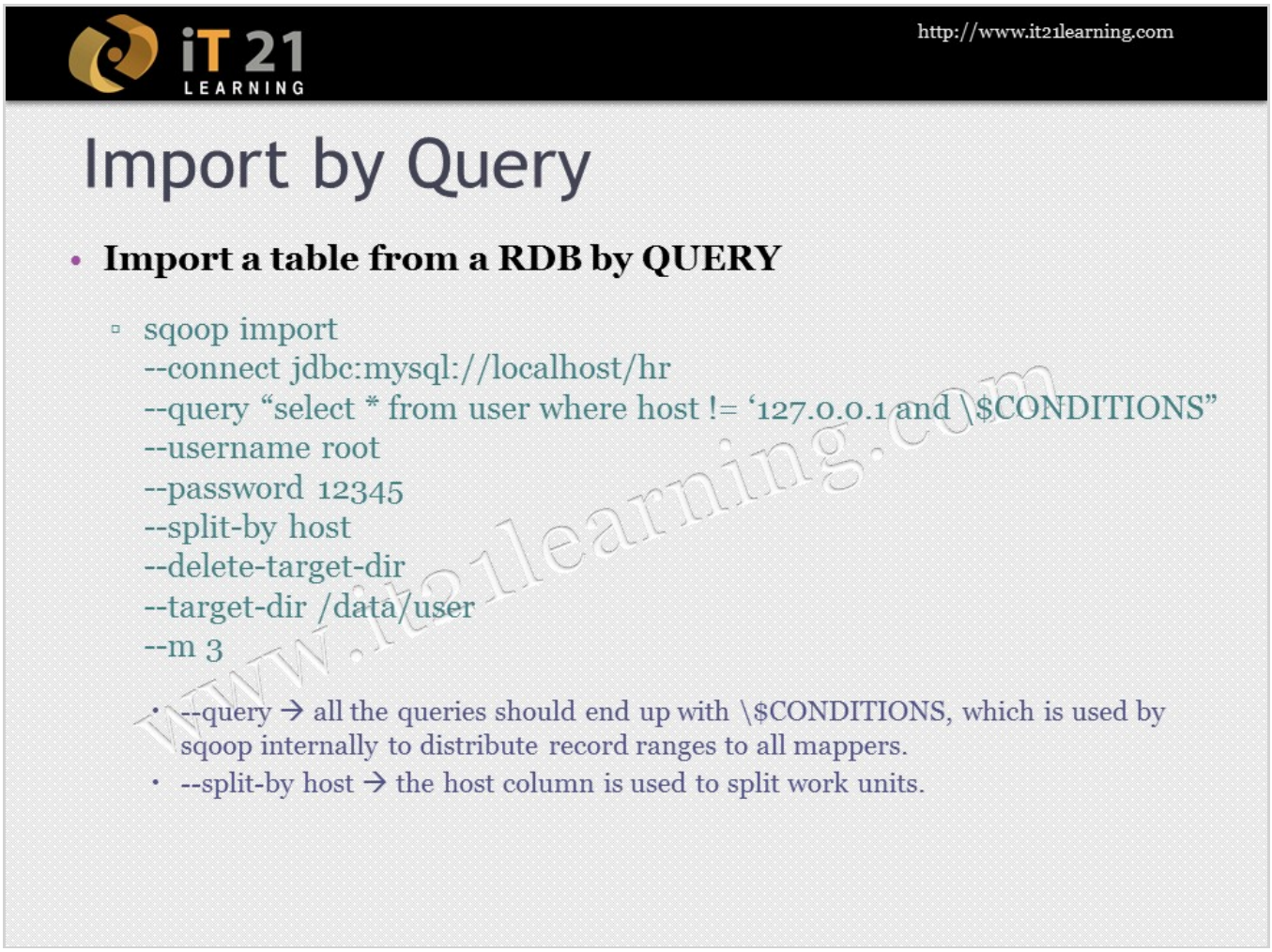
3


iT 21 LEARNING
http://www.it21learning.com
Import by WHERE
Import a table by filtering with WHERE
sqoop import
--connect jdbc:mysql://localhost/hr
--table orders
--where "order_date > '2015-10-10'"
--username root
--password 12345
--delete-target-dir
--target-dir /data/orders
--m 3
WHERE for filtering records
www.it21learning.com

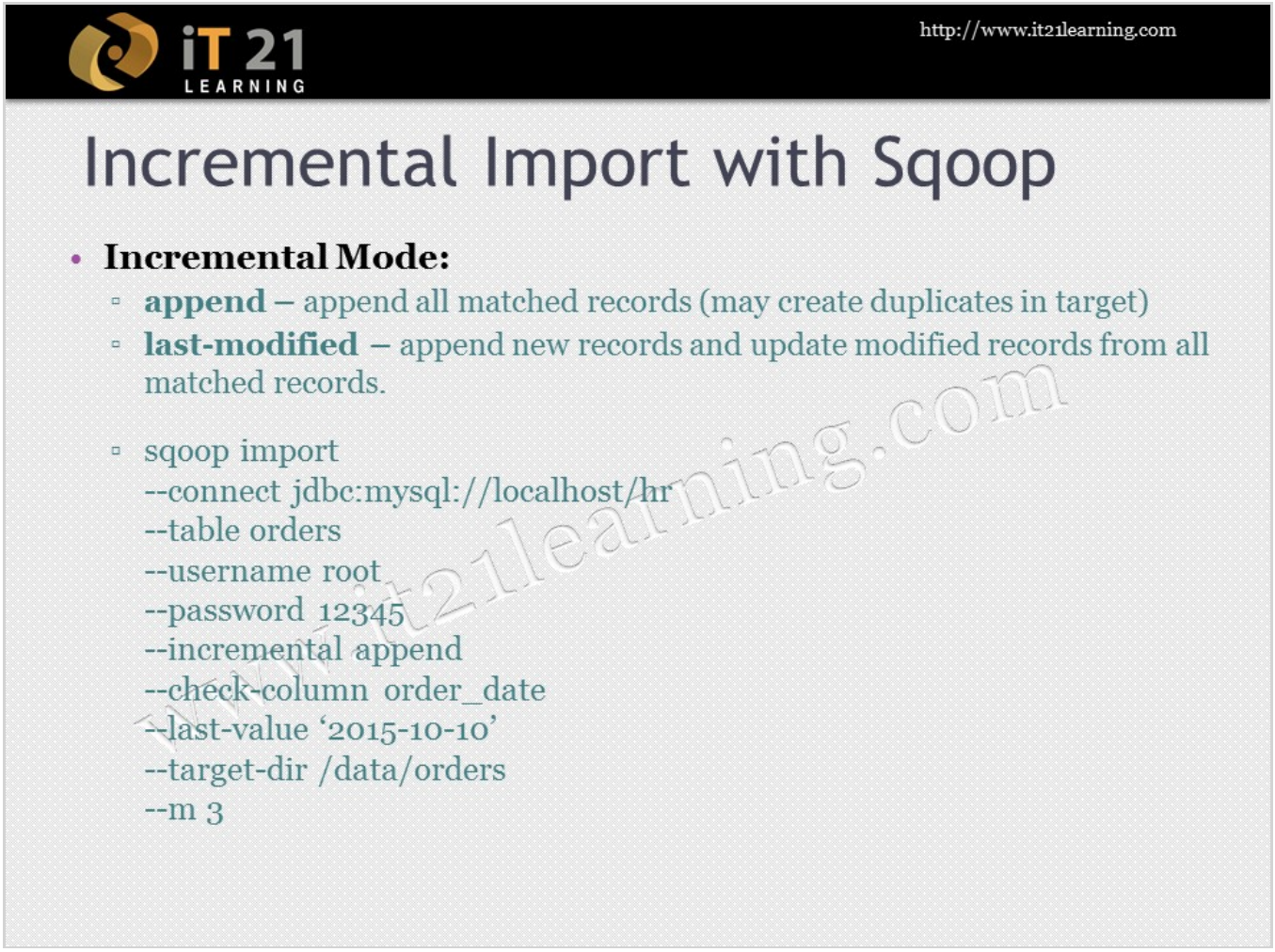
4


iT 21 LEARNING
http://www.it21learning.com
Import by COLUMNs
Import a table from a RDB by COLUMNs
sqoop import
--connect jdbc:mysql://localhost/hr
--table user
--columns "host,name,age"
--username root
--password 12345
--delete-target-dir
--target-dir /data/user
--m 3
--columns → the column list
--delete-target-dir → delete the target directory if it exists
www.it21learning.com

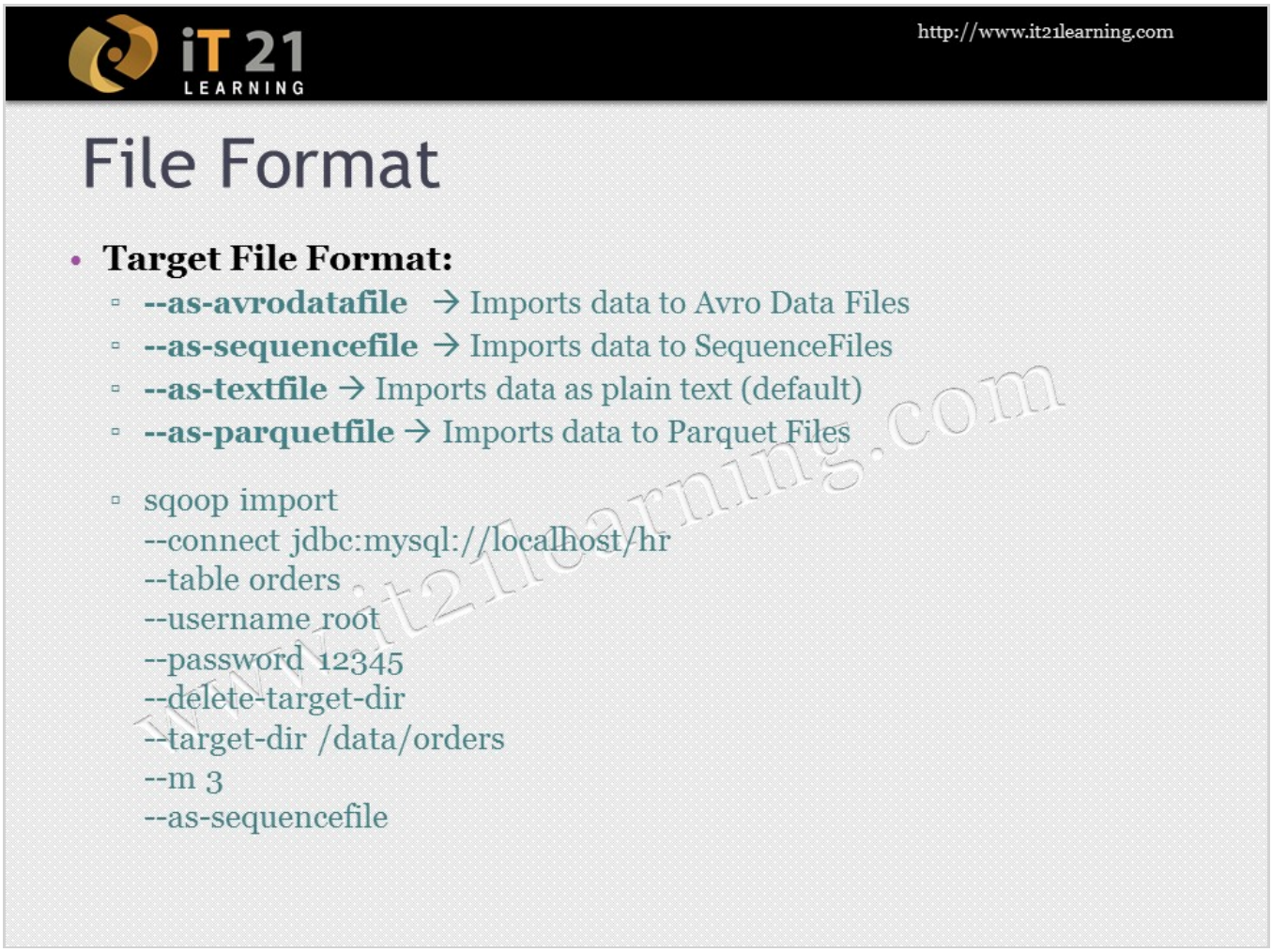
5


iT 21 LEARNING
http://www.it21learning.com
Import by Query
Import a table from a RDB by QUERY
sqoop import
--connect jdbc:mysql://localhost/hr
--query "select * from user where host != '127.0.0.1 and \$CONDITIONS"
--username root
--password 12345
--split-by host
--delete-target-dir
--target-dir /data/user
--m 3
--query → all the queries should end up with \$CONDITIONS, which is used by sqoop internally to distribute record ranges to all mappers.
--split-by host → the host column is used to split work units.

6


iT 21 LEARNING
http://www.it21learning.com
Incremental Import with Sqoop
Incremental Mode:
append – append all matched records (may create duplicates in target)
last-modified – append new records and update modified records from all matched records.
sqoop import
--connect jdbc:mysql://localhost/hr
--table orders
--username root
--password 12345
--incremental append
--check-column order_date
--last-value '2015-10-10'
--target-dir /data/orders
--m 3
www.it21learning.com

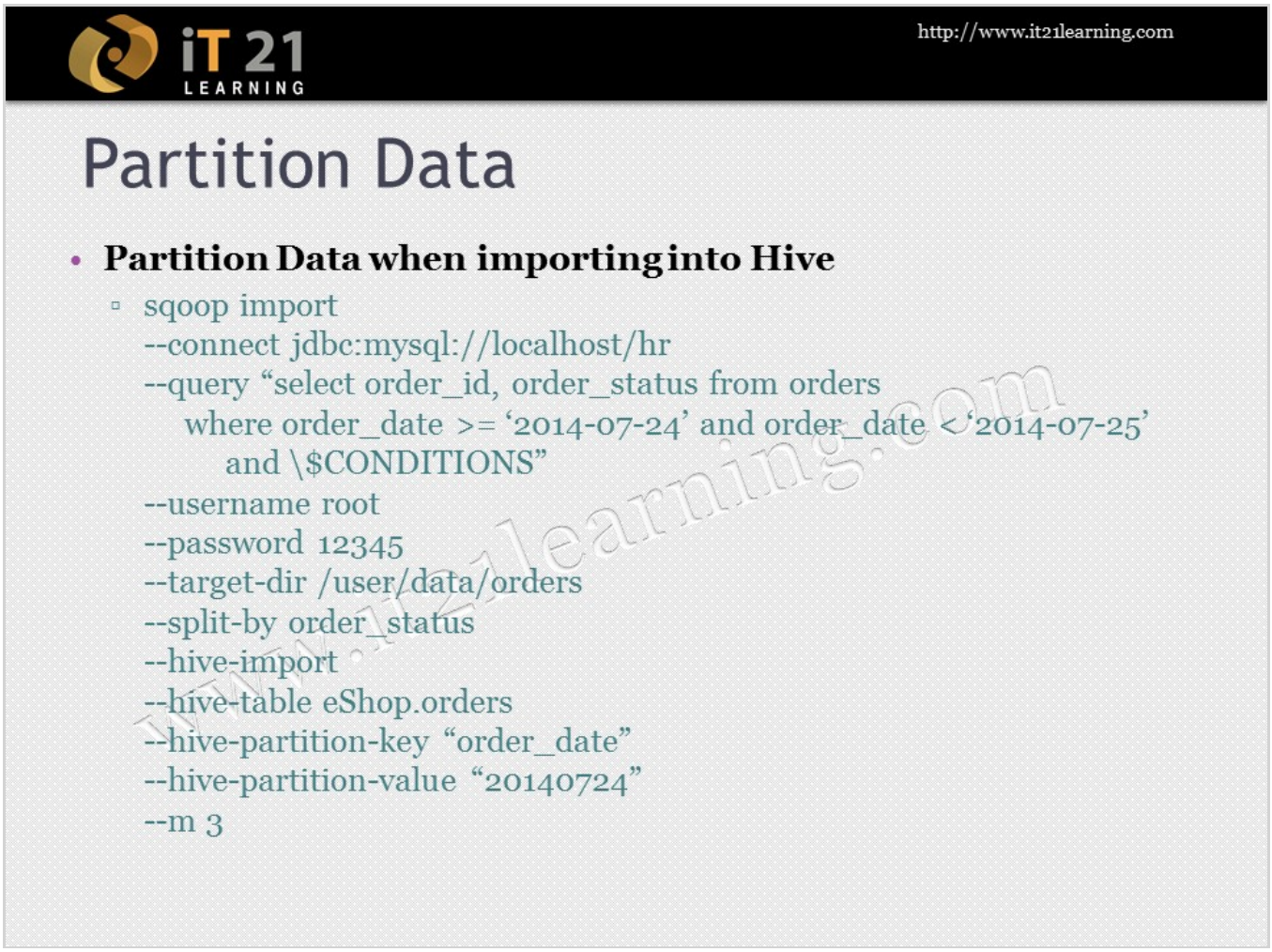
7


iT 21 LEARNING
http://www.it21learning.com
File Format
Target File Format:
--as-avrodatafile → Imports data to Avro Data Files
--as-sequencefile → Imports data to SequenceFiles
--as-textfile → Imports data as plain text (default)
--as-parquetfile → Imports data to Parquet Files
sqoop import
--connect jdbc:mysql://localhost/hr
--table orders
--username root
--password 12345
--delete-target-dir
--target-dir /data/orders
--m 3
--as-sequencefile
www.it21learning.com

8

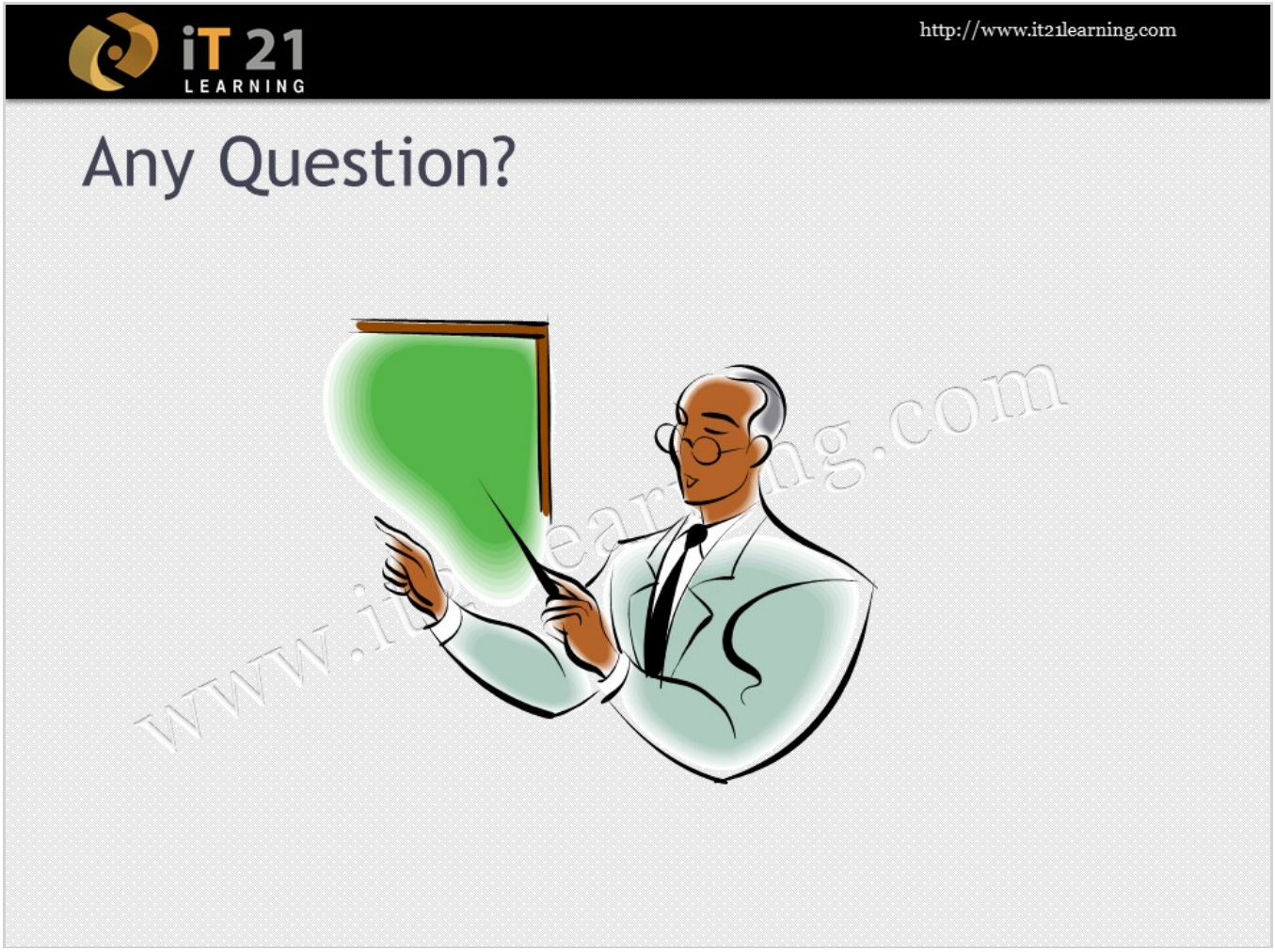

iT 21 LEARNING
http://www.it21learning.com
Import into Hive
• sqoop import
--connect jdbc:mysql://localhost/hr
--table orders
--username root
--password 12345
--hive-import
--create-hive-table
--hive-table eShop.orders
--hive-overwrite
--m 3
--as-parquetfile
• --hive-import → import into Hive
• --create-hive-table → create a new hive table. If target table exists, error out;
• --hive-overwrite → overwrite existing target data
www.it21learning.com

9


iT 21 LEARNING
http://www.it21learning.com
Partition Data
Partition Data when importing into Hive
sqoop import
--connect jdbc:mysql://localhost/hr
--query "select order_id, order_status from orders
where order_date >= '2014-07-24' and order_date < '2014-07-25'
and \$CONDITIONS"
--username root
--password 12345
--target-dir /user/data/orders
--split-by order_status
--hive-import
--hive-table eShop.orders
--hive-partition-key "order_date"
--hive-partition-value "20140724"
--m 3

10

Any Question?



11