

## 海量数据处理中的云计算

# C8. HBase (二)

北京邮电大学信息与通信工程学院

2013年春季学期

# 本节目录

- 操作HBase数据方式
- 从RDMBS到HBase的表设计转变
- HBase数据操作基本功能
- HBase数据操作高级功能

# 本节目录

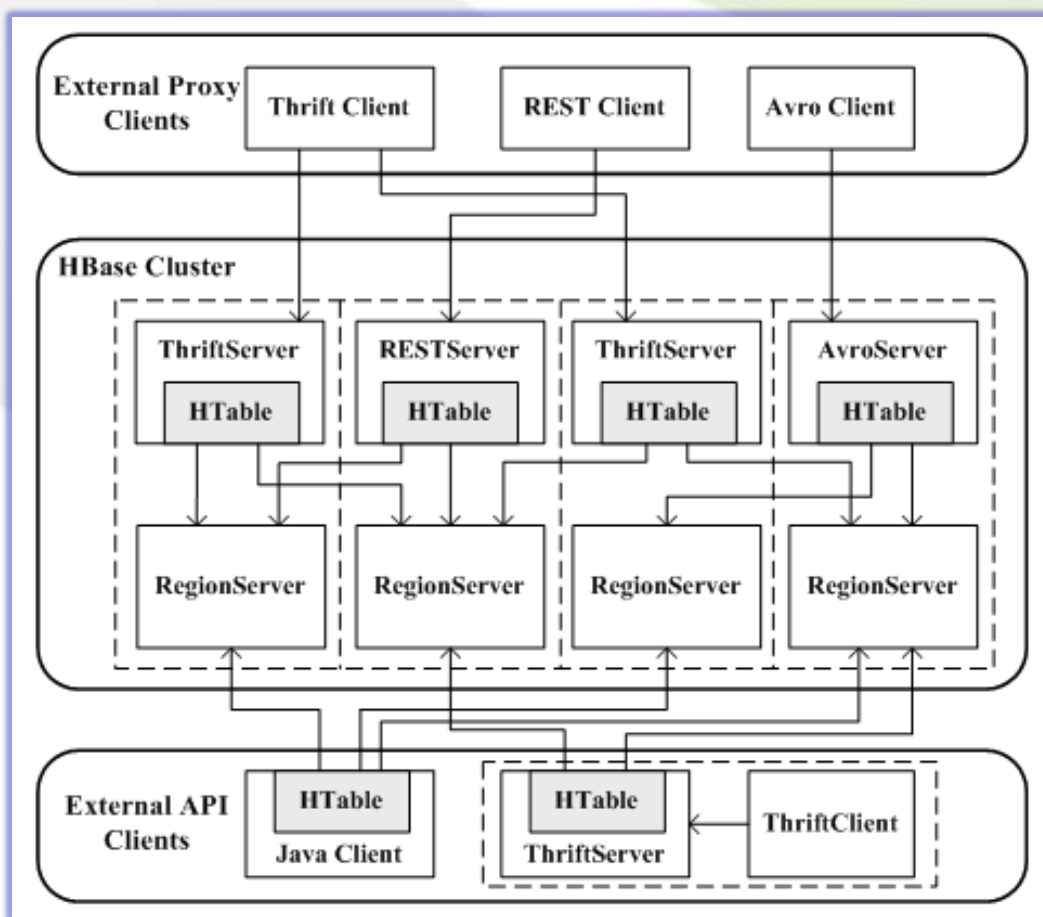
- 操作HBase数据方式
- 从RDMBS到HBase的表设计转变
- HBase数据操作基本功能
- HBase数据操作高级功能

# 操作HBase数据方式（1） - Shell

- HBase的Shell工具
  - 通过Shell可以连接到本地或远程的HBase服务器上操纵数据
  - Shell工具是基于JRuby实现
  - 命令行启动：`$ $HBASE_HOME/bin/hbase shell`
- 五类命令：
  - 表管理：创建、删除和修改表的相关操作指令
  - 数据管理：对表中的数据进行操作
  - 工具：管理和优化数据存储方式的功能
  - 复制：将数据备份到多个节点的相关操作指令
  - 其他：查看HBase集群状态和版本
- 示例：
  - `create 'table', 'col_f1', 'col_f2'`
  - `put 'table', 'row1', 'col_f1', 'value'`
  - `get 'table', 'row1', {COLUMN => 'col_f1'}`

# 操作HBase数据方式（2） - 非Java API

- 代理服务器封装Java API的能力进行封装
  - REST代理服务器：支持HTTP接口
  - Thrift代理服务器：支持RPC接口
  - Avro代理服务器：支持RPC接口



# 操作HBase数据方式（3） - Java API

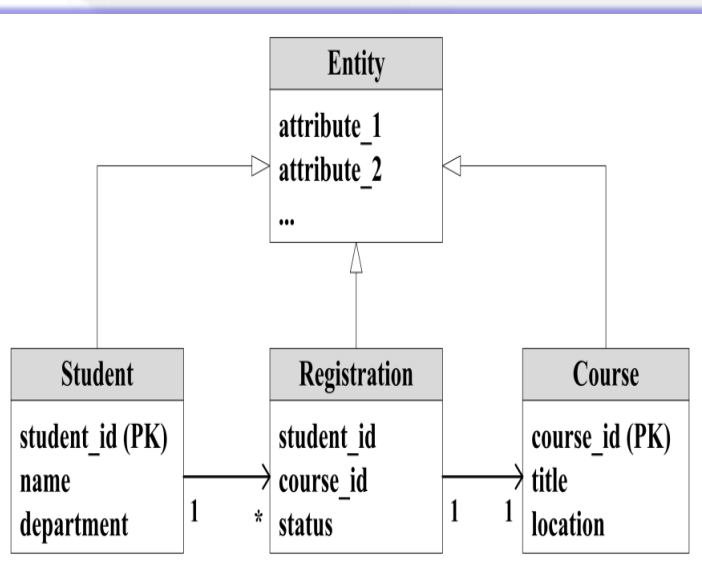
- HBaseConfiguration
  - 配置类
- HBaseAdmin
  - 对数据表结构进行操作的接口
- HTableDescriptor
  - 数据表相关的属性和操作接口
- HColumnDescriptor
  - 数据列相关的数据和操作接口
- HTable , Put/Get/Delete
  - 数据的插入、检索和删除操作

# 本节目录

- 操作HBase数据方式
- 从RDMBS到HBase的表设计转变
- HBase数据操作基本功能
- HBase数据操作高级功能

# RDBMS表设计

- RDBMS表的ER设计模型：以实体（Entity）、实体间关系（Relationship）、以及实体属性（Attribute）为核心的建模过程
- 数据库设计三大范式：
  - 属性具有原子性，不可再分解
  - 记录有惟一标识，即实体的惟一性
  - 任何字段不能由其他字段派生出来，要求字段没有冗余

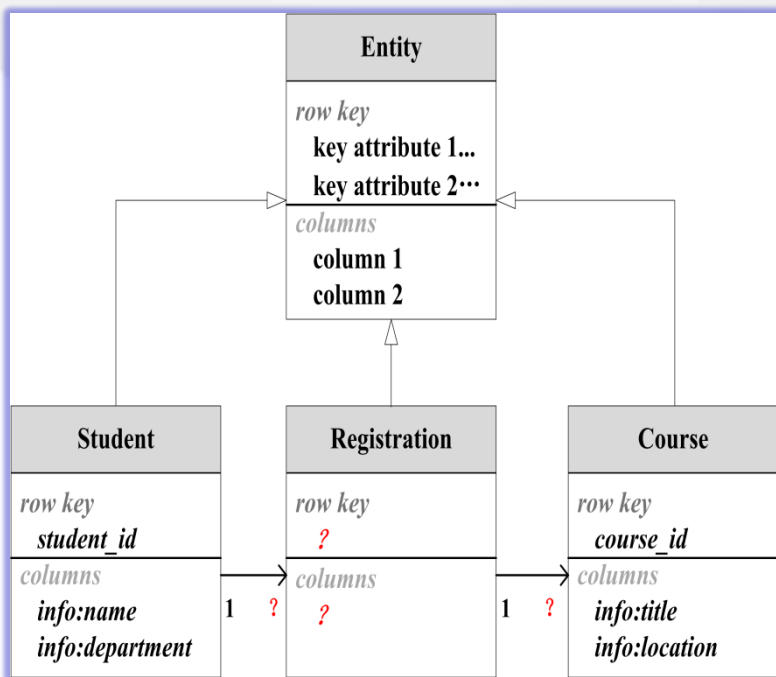


Student ( 强实体 )		
student_id	name	department
1	张三	计算机系
2	李四	计算机系
3	王五	经管系
Course ( 强实体 )		
course_id	title	location
1	高等数学	教1楼101
2	网络原理	教2楼202
3	计算机原理	教3楼303
Registration ( 弱实体 )		
student_id	course_id	status
1	2	ok
1	3	pending
2	1	ok
3	1	ok



# HBase表设计

- 行关键字取代了主键，属性由列关键字取代
- HBase表设计的三大原则：
  - Denormalization, Duplication, and Intelligent Keys ( DDI )
  - 属性原子性、实体ID唯一性
  - 字段冗余



Student		
行关键字	列族:info	
	限定词:name	限定词:department
1	张三	计算机系
2	李四	计算机系
3	王五	经管系
Course		
行关键字	列族:info	
	限定词:title	限定词:location
1	高等数学	教1楼101
2	网络原理	教2楼202
3	计算机原理	教3楼303
Registration		
行关键字	列族	
	限定词	
?	?	
...		
行关键字	列族	
	限定词	
...	...	

# HBase表设计

	A	B	C	D	E
1					
2					
3					
4					
5					
6					
7					

# HBase表设计

	A	B	C	D	E
1					
2					
3	A3	B3	C3	D3	E3
4					
5					
6					
7					

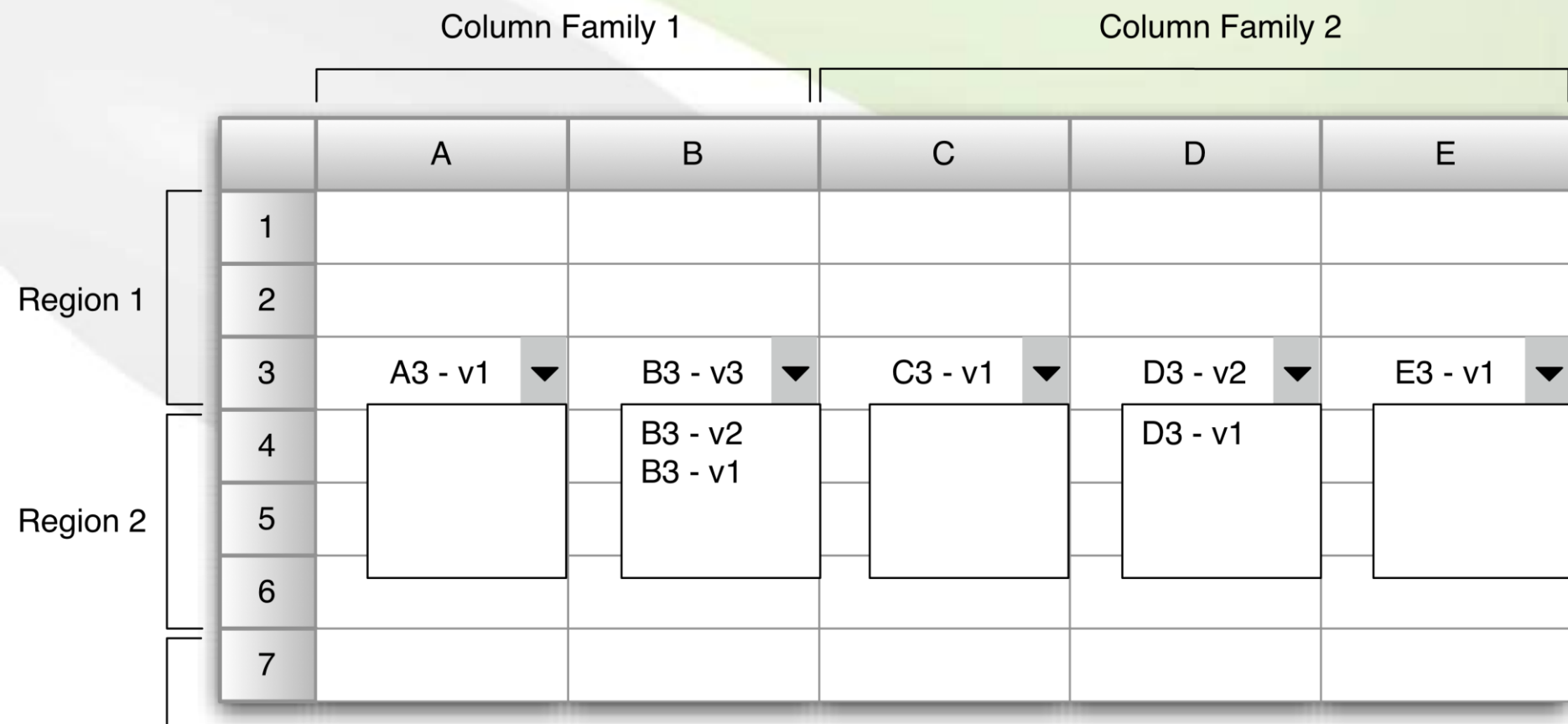
# HBase表设计

	A	B	C	D	E
1					
2					
3	A3 - v1	B3 - v3	C3 - v1	D3 - v2	E3 - v1
4					
5					
6					
7					













# HBase表设计

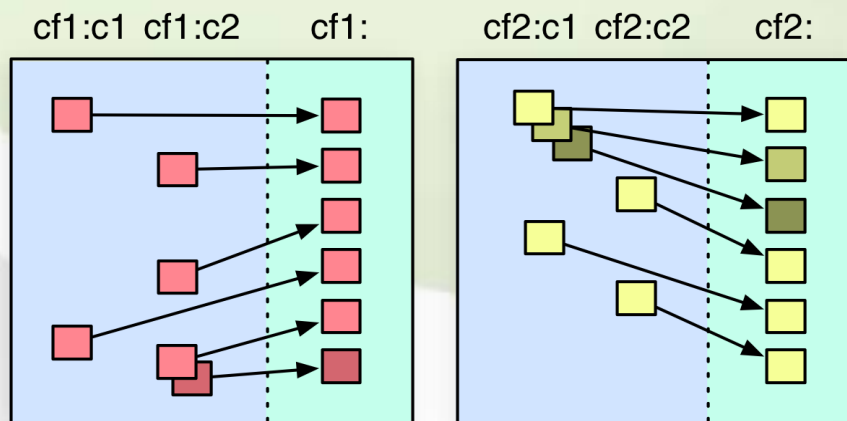
	A	B	C	D	E
1					
2					
3	A3 - v1 ▼	B3 - v3 ▼	C3 - v1 ▼	D3 - v2 ▼	E3 - v1 ▼
4		B3 - v2 B3 - v1		D3 - v1	
5					
6					
7					

# HBase表设计



# HBase表设计

	cf1:c1	cf1:c2	cf2:c1	cf2:c2
r1				
r2				
r3			  	
r4				
r5				
r6		 		



Store

r1 : cf1 : c1 : t1 : <value> \x00

r1 : cf1 : c1-<value> : t1 : \x00

r1-<value> : cf1 : c1 : t1 : \x00

= Same Storage Requirements



r1 : cf1 : c1 : t1 : <value>

r2 : cf1 : c2 : t1 : <value>

r4 : cf1 : c2 : t1 : <value>

r5 : cf1 : c1 : t1 : <value>

r6 : cf1 : c2 : t2 : <value>

r6 : cf1 : c2 : t1 : <value>

StoreFile "cf1/1234"

r3 : cf2 : c1 : t3 : <value>

r3 : cf2 : c1 : t2 : <value>

r3 : cf2 : c1 : t1 : <value>

r4 : cf2 : c2 : t1 : <value>

r5 : cf2 : c1 : t1 : <value>

r6 : cf2 : c2 : t1 : <value>

StoreFile "cf2/5678"

# 本节目录

- 操作HBase数据方式
- 从RDMBS到HBase的表设计转变
- **HBase数据操作基本功能**
- HBase数据操作高级功能



# HBase数据操作实例

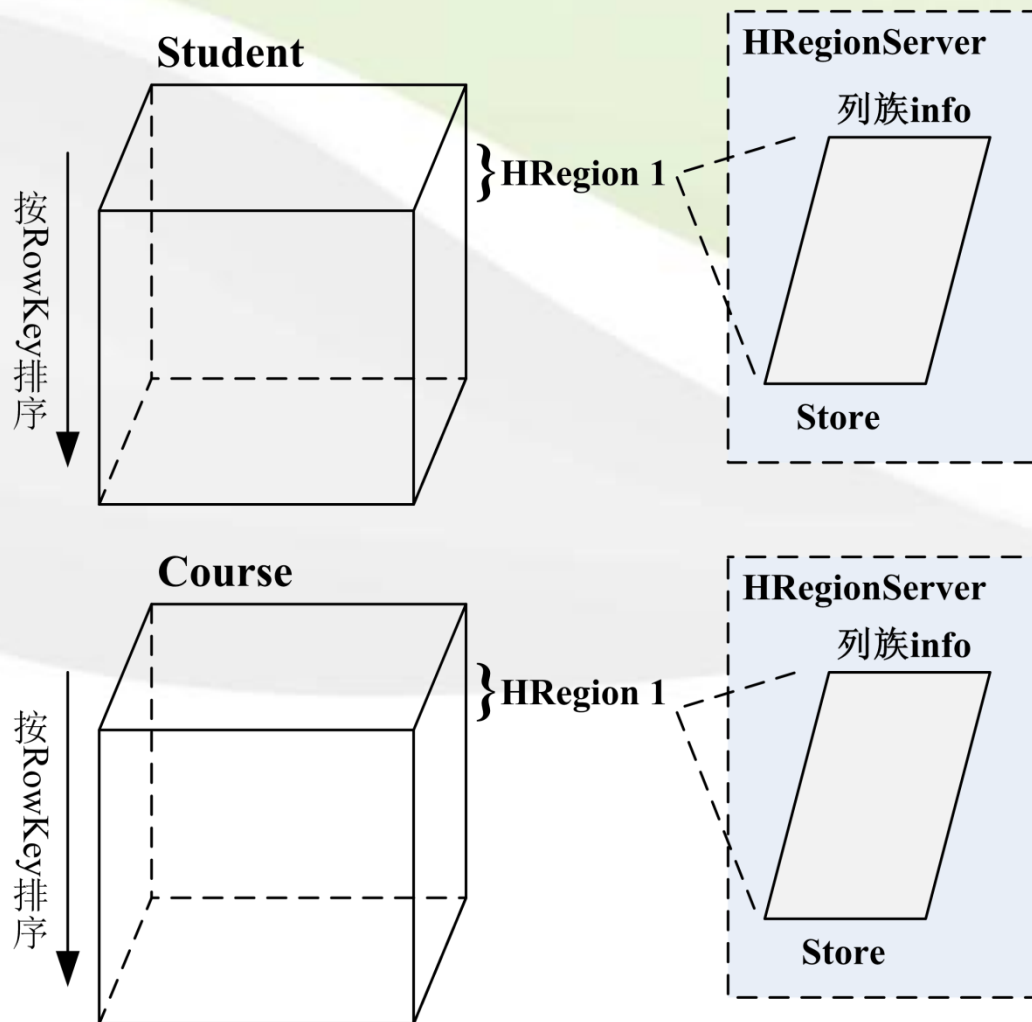
Student		
行关键字	列族:info	
	限定词:name	限定词:department
1	张三	计算机系
2	李四	计算机系
3	王五	经管系
Course		
行关键字	列族:info	
	限定词:title	限定词:location
1	高等数学	教1楼101
2	网络原理	教2楼202
3	计算机原理	教3楼303

- 创建表
- 插入数据
- 检索数据
- 删除数据
- 批量组合操作

# 创建表

```
1: Configuration conf = HBaseConfiguration.create(); // 初始化配置
   // 创建HBaseAdmin对象
2: HBaseAdmin admin = new HBaseAdmin(conf);
   // 设置表信息
3: HTableDescriptor tableDesc = new HTableDescriptor(toBytes ("student"));
   // 设置列族信息
4: HColumnDescriptor colDesc = new HColumnDescriptor(toBytes("info"));
5: tableDesc.addFamily(colDesc); // 将列族信息加入到表信息中
6: admin.createTable(tableDesc); // 创建表
   // 验证表是否创建成功
7: Boolean isAvailable = admin.isTableAvailable(toBytes("student"));
8: System.out.println("Table student available: " + isAvailable);
```

# 创建表后



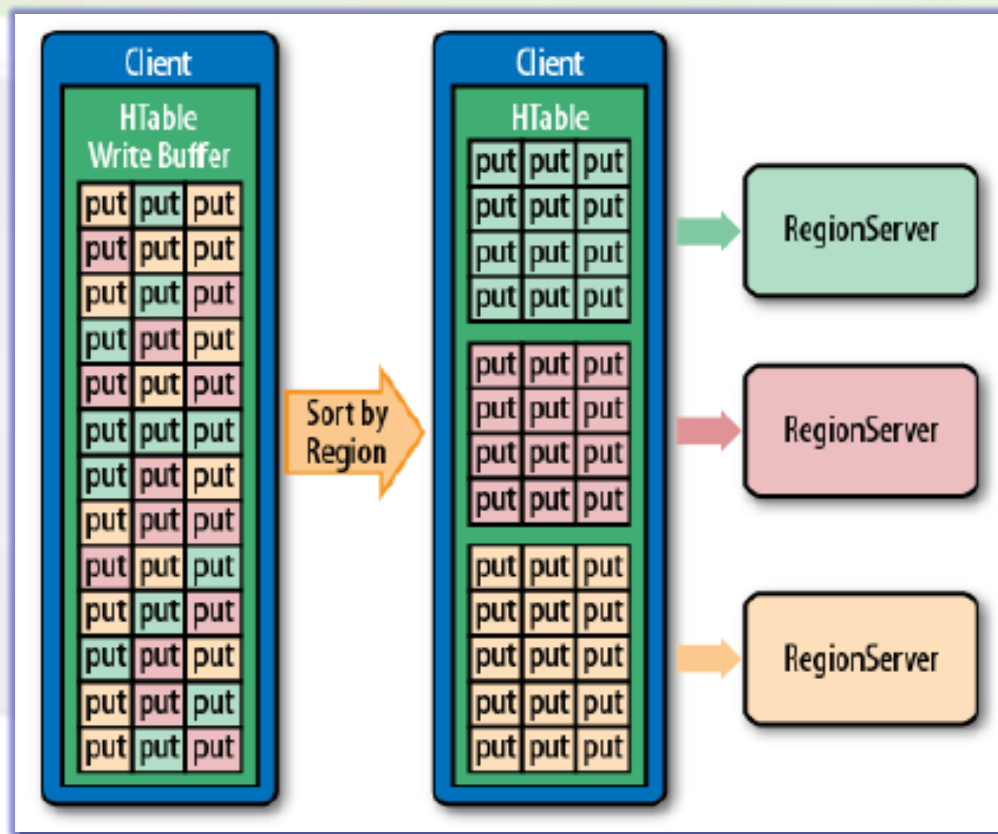
# 插入单行数据

```
1:  HTable table = new HTable(conf, "student");
2:  for each line in StudentFile
    // 提取数据
3:  Bytes rowkey = getStudentID(line)
4:  Bytes name = getStudentName(line)
5:  Bytes department = getStudentDepartment(line)
6:  Put put = new Put(rowkey); // 指定行索引
    // name和department列
7:  put.add(toBytes("info"), toBytes("name"), name);
8:  put.add(toBytes("info"), toBytes("department"), department);
9:  table.put(put); // 插入操作
10: end for
```

# 插入多行数据

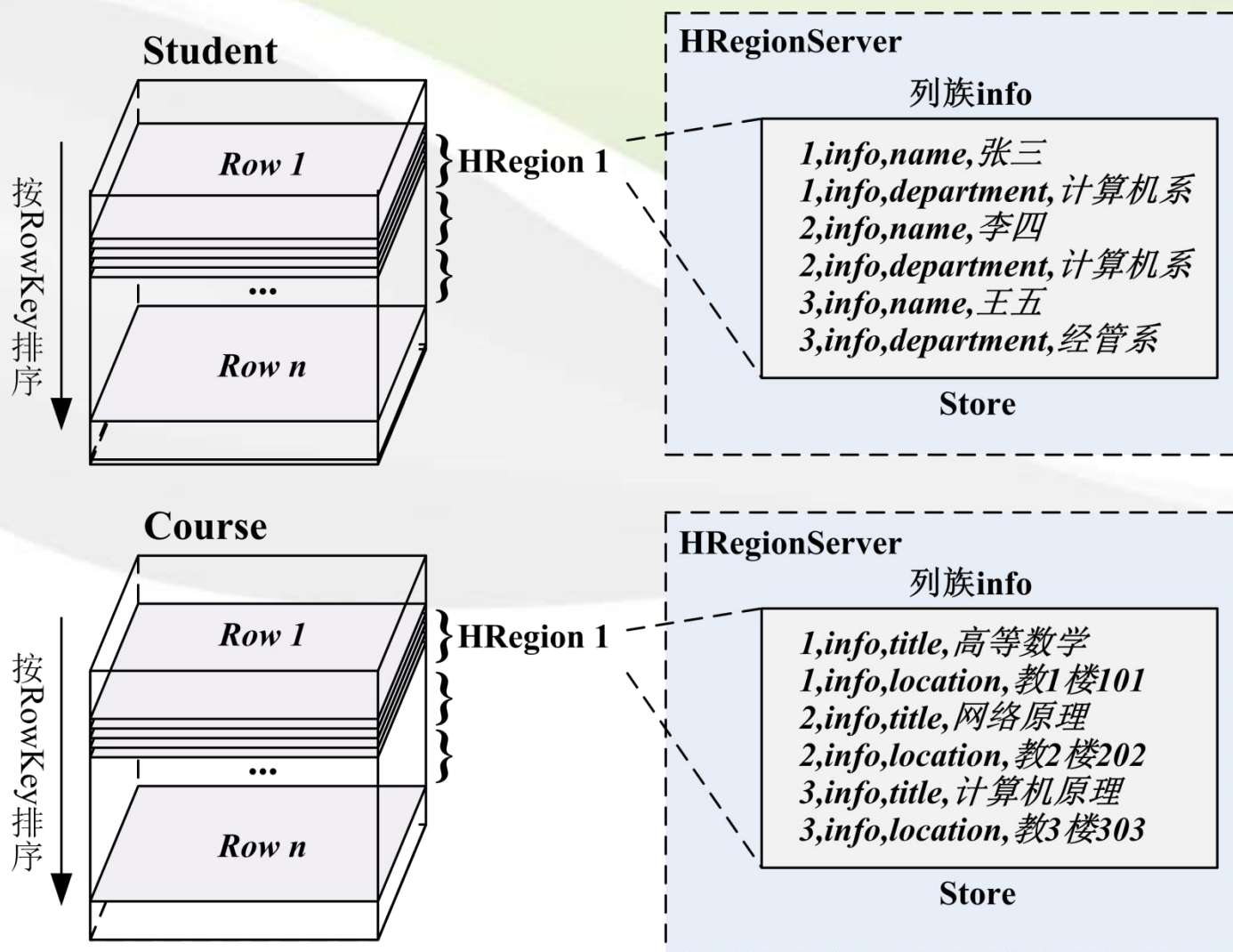
```
1:  HTable table = new HTable(conf, "student");
2:  List<Put> puts = new ArrayList<Put> ();
3:  for each line in StudentFile
4:    for i=0 to i=putsCount // 批量插入数量
5:      Bytes rowkey = getStudentID(line)
6:      Bytes name = getStudentName(line)
7:      Bytes department = getStudentDepartment(line)
8:      Put put = new Put(rowkey); // 指定行索引
9:      put.add(toBytes("info"), toBytes("name"), name);
10:     put.add(toBytes("info"), toBytes("department"), department);
11:     puts[i] = put
12:   end for
13:   table.put(puts); // 批量插入操作
14: end for
```

# 启用客户端写缓存



- `setWriteBufferSize(long writeBufferSize)`
- `setAutoFlush(false)`
- `flushCommits()`

# 插入数据后

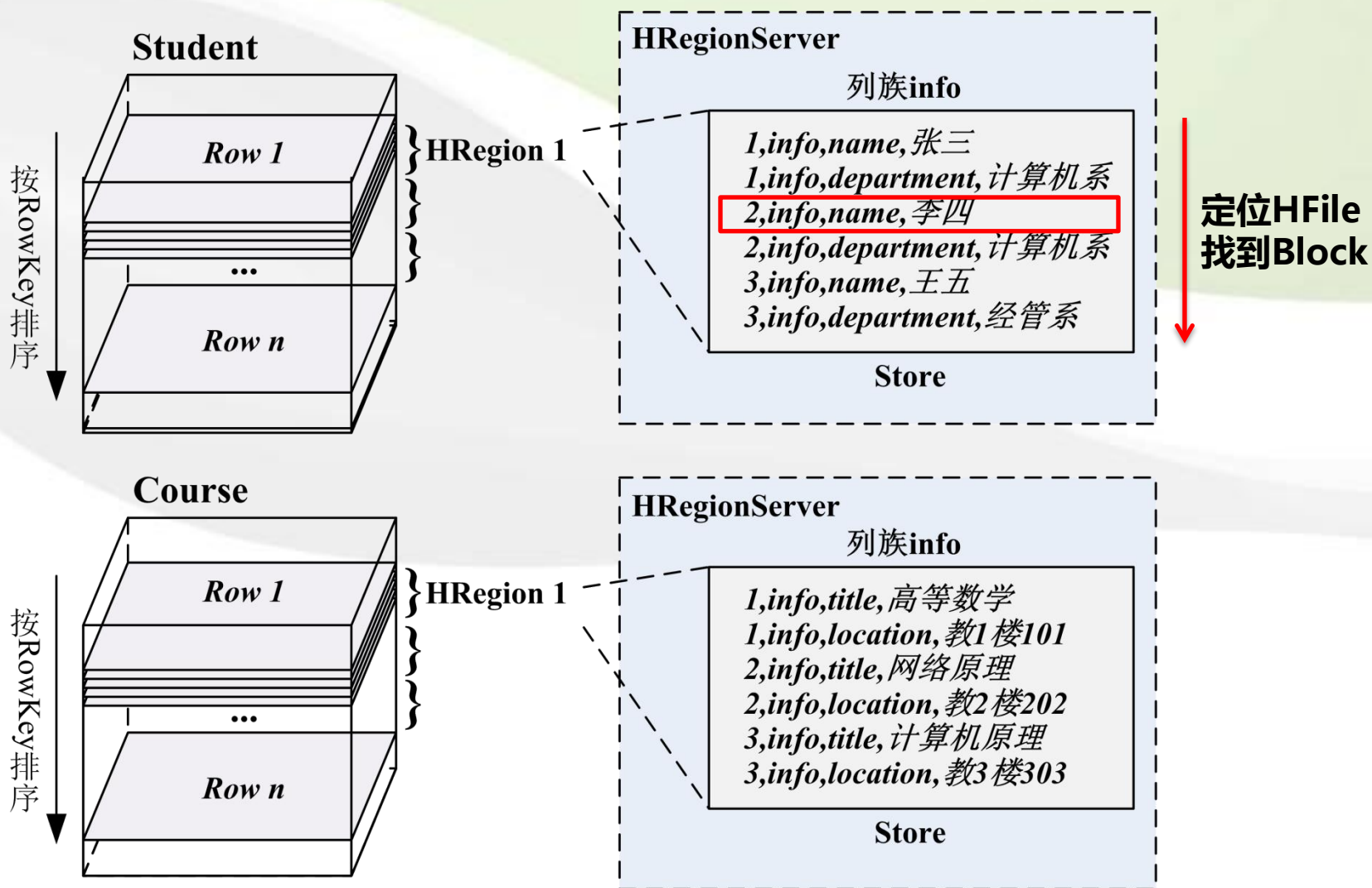


# 检索单行数据

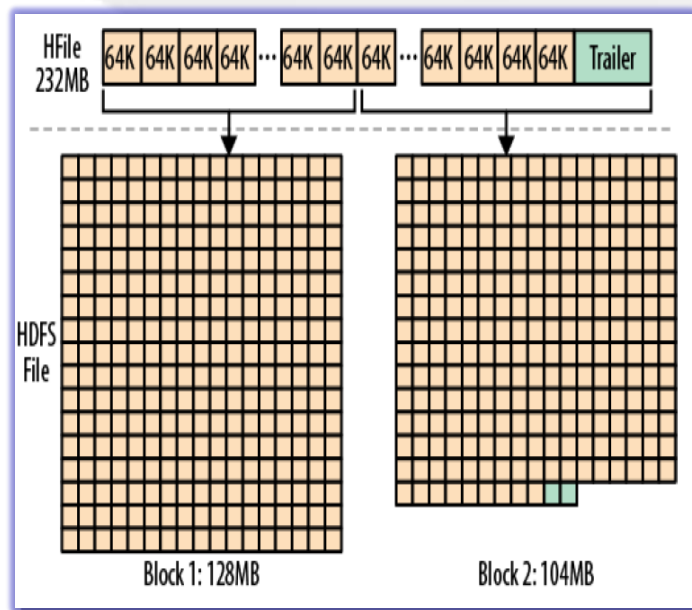
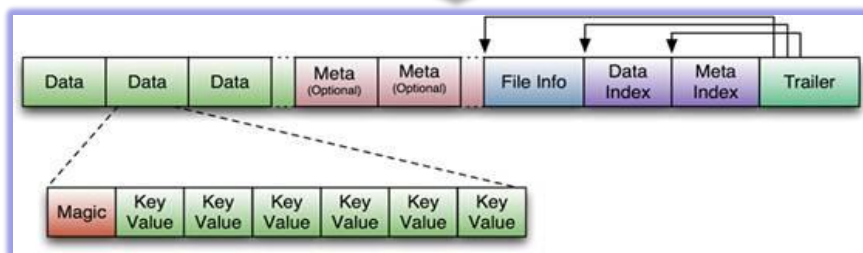
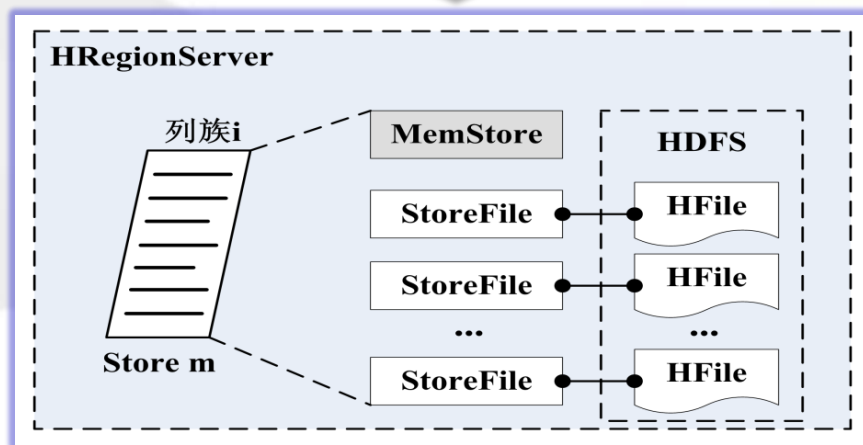
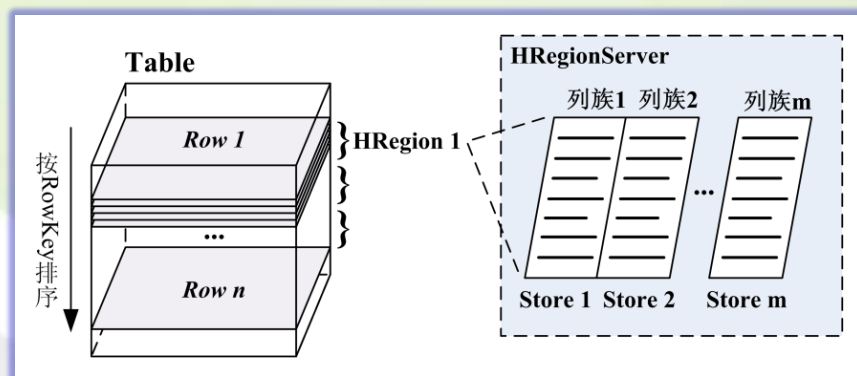
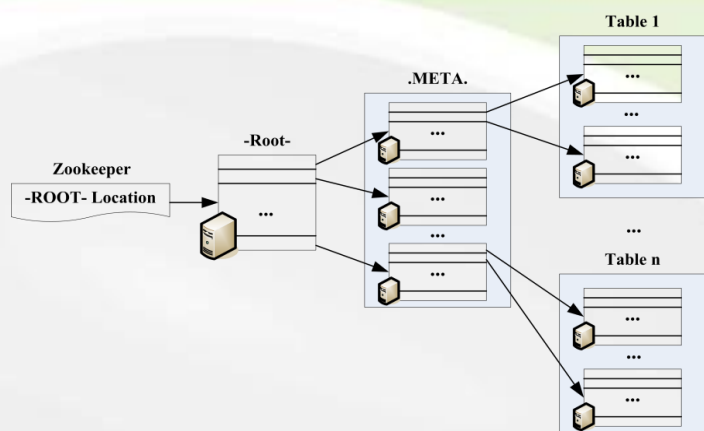
```
1:  HTable table = new HTable(conf, "student");  
    // 设置要读取的行索引  
2:  rowkey = toBytes(2)  
3:  Get get = new Get(rowkey);  
    // 设置要读取的列关键字  
4:  get.addColumn(toBytes("info"), toBytes("name"));  
    // 读取数据  
5:  Result result = table.get(get);  
6:  byte[] value = result.getValue(toBytes("info"), toBytes("name"));  
7:  System.out.println("Value: " + toString(value));
```



# 检索数据



# 检索数据过程



# 检索多行数据

```
1:  HTable table = new HTable(conf, "student");
2:  List<Put> gets = new ArrayList<Get> ();
3:  for i=0 to i=getsCount // 批量读取数量
4:    gets[i] = new Get(toBytes(i));
5:  end for
6:  results = table.get(gets); // 批量读取操作
7:  for each result in results
8:    String row = toString(result.getRow());
9:    byte[] value = null;
10:   if result.containsColumn(toBytes("info"),toBytes("name"))
11:     value = result.getValue(toBytes("info"), toBytes("name"));
12:   end if
13:   System.out.println("Row" + row + "with Value: " + toString(value));
14: end for
```

# 删除多行数据

```
1: HTable table = new HTable(conf, "student");
2: List<Put> deletes = new ArrayList<Delete> ();
3: for i=0 to i=deletesCount // 批量读取数量
4:   Delete delete = new Delete(toBytes(i));
5:   delete.setTimestamp(timestamp)
4:   deletes[i] = delete
5: end for
6: table.delete(deletes); // 批量删除操作
7: table.close()
```

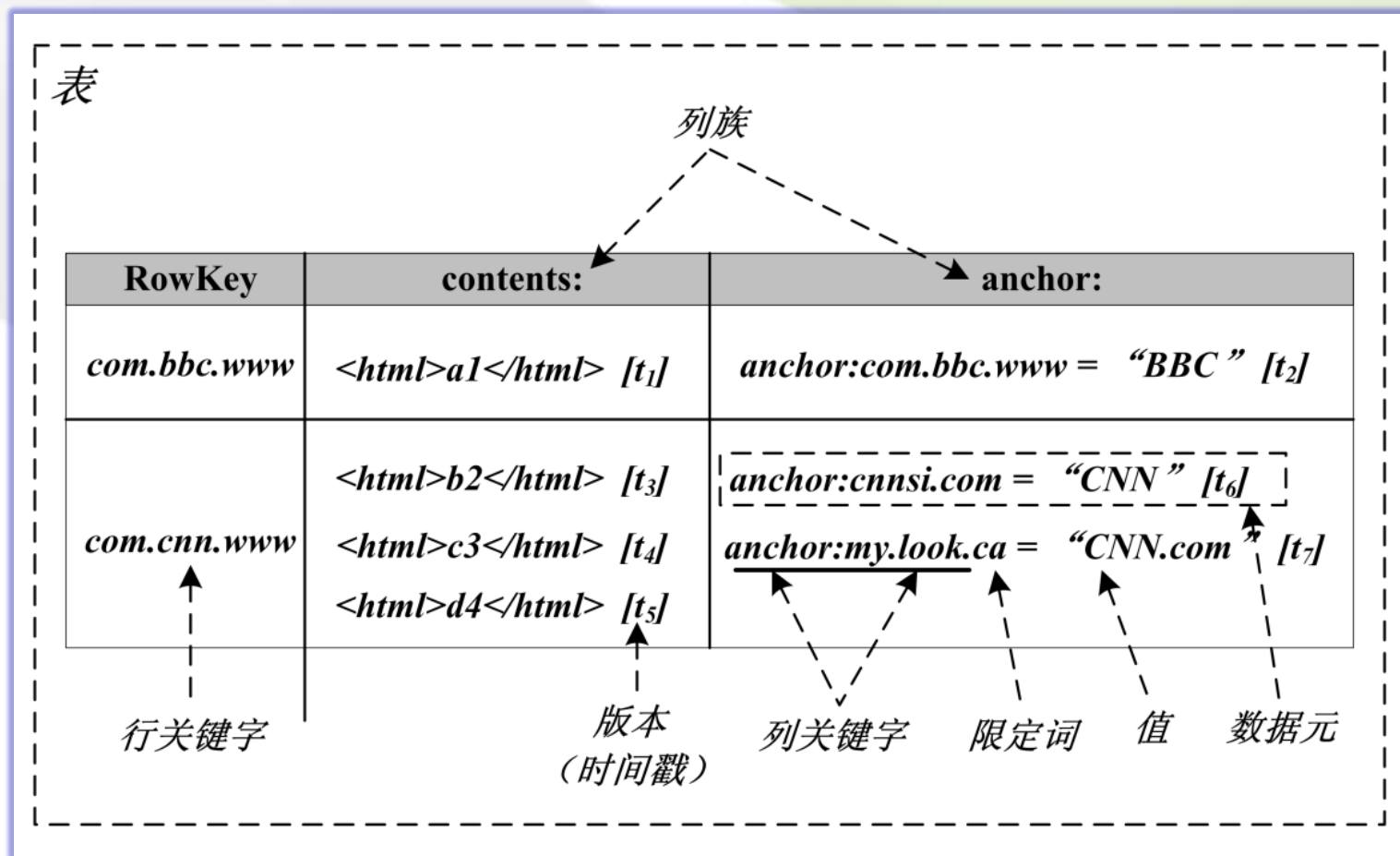
# 批量组合操作

```
1: HTable table = new HTable(conf, "student");
2: List<Row> batch = new ArrayList<Row>();
3: Put put = new Put(toBytes(rowkey)); // put操作
4: put.add(toBytes("info"), toBytes("name"), name);
5: put.add(toBytes("info"), toBytes("department"), department);
6: batch.add(put);
7: Get get = new Get(rowkey); // get操作
8: get.addColumn(toBytes("info"), toBytes("name"));
9: batch.add(get);
10: Delete delete = new Delete(toBytes(i)); // delete操作
11: delete.setTimestamp(timestamp)
12: batch.add(delete);
13: Object[] results = new Object[batch.size()];
12: table.batch(batch, results); // 批量组合操作
```

# 更新操作哪去了？

- 思考

- HDFS的特性
- HBase的表结构



# 本节目录

- 操作HBase数据方式
- 从RDMBS到HBase的表设计转变
- HBase数据操作基本功能
- **HBase数据操作高级功能**

# 高级功能（1） - 联合查询

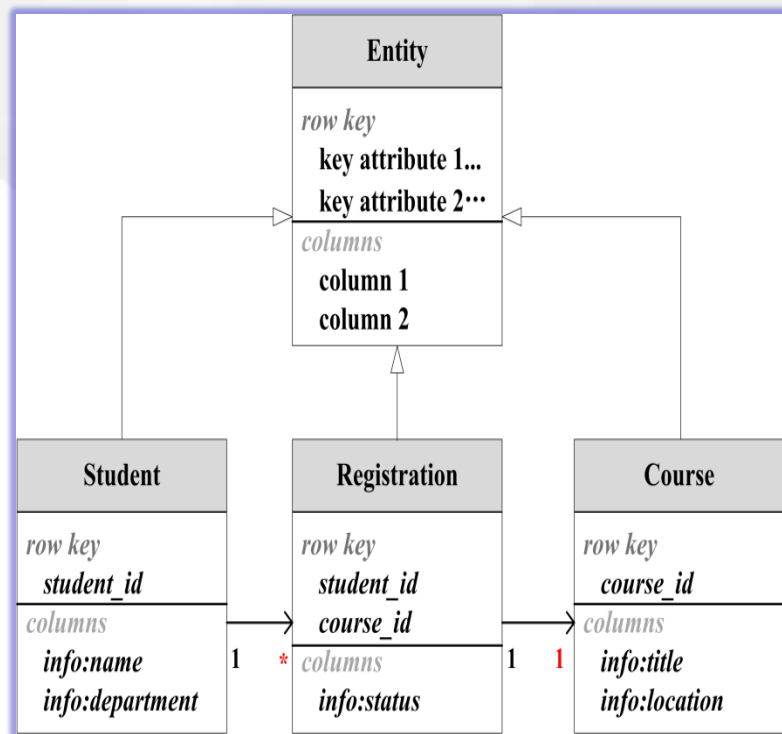
- RDBMS联合查询：张三选修成功的课程名称和地点
  - SELECT @sid=student\_id FROM Student WHERE name="张三"
  - SELECT title, location FROM Course c JOIN Registration r ON (c.course\_id=r.course\_id) WHERE r.student\_id=@sid and r.status="ok"
- 结果：*网络原理*                      *教2楼202*

Student		
student_id	name	department
1	张三	计算机系
2	李四	建筑系
3	王五	经管系
Course		
course_id	title	location
1	高等数学	教1楼101
2	网络原理	教2楼202
3	计算机原理	教3楼303
Registration		
student_id	course_id	status
1	2	ok
1	3	pending
2	1	ok
3	1	ok



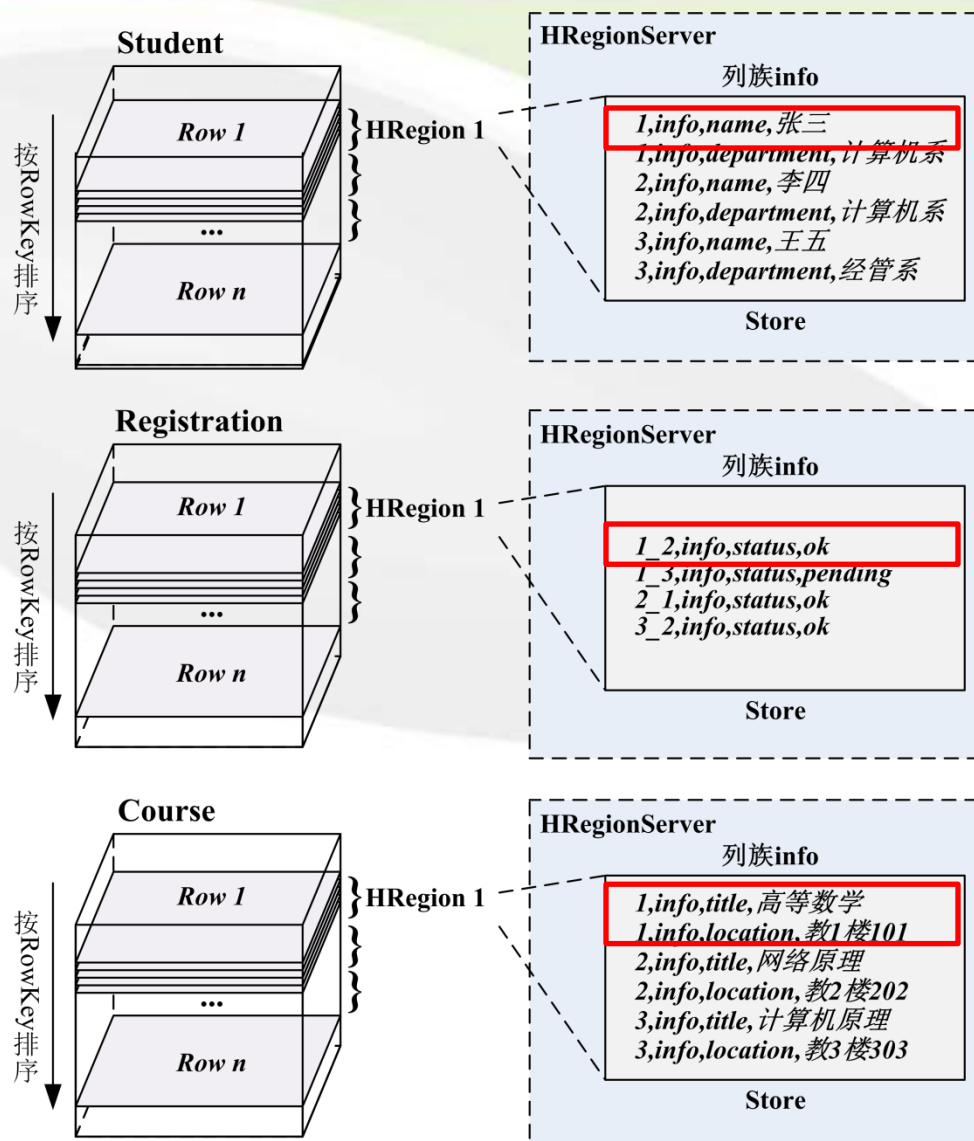
# 高级功能（1） - 联合查询的HBase朴素实现

## ● 表设计



Student		
行关键字	列族:info	
	限定词:name	限定词:department
1	张三	计算机系
2	李四	计算机系
3	王五	经管系
Course		
行关键字	列族:info	
	限定词:title	限定词:location
1	高等数学	教1楼101
2	网络原理	教2楼202
3	计算机原理	教3楼303
Registration		
行关键字	列族:info	
	限定词:status	
1_2	ok	
1_3	pending	
2_1	ok	
3_1	ok	

# 高级功能（1） - 联合查询的HBase朴素实现



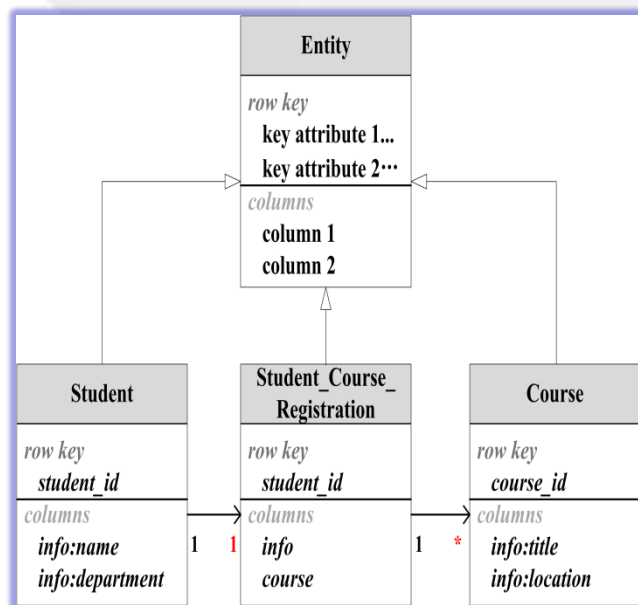
```
1: table(conf, "Student");
2: sid = tabel.get("张三")

3: table(conf, "Registration");
4: cid =table.filter(sid, "ok"),

5: table(conf, "Course");
6: table.get(cid)
```

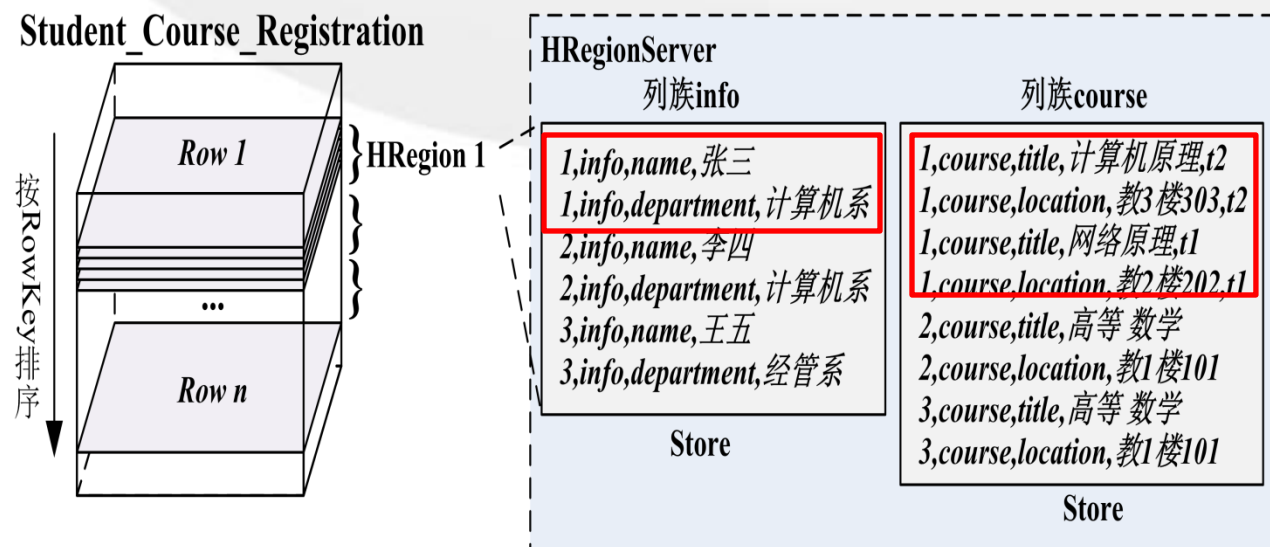
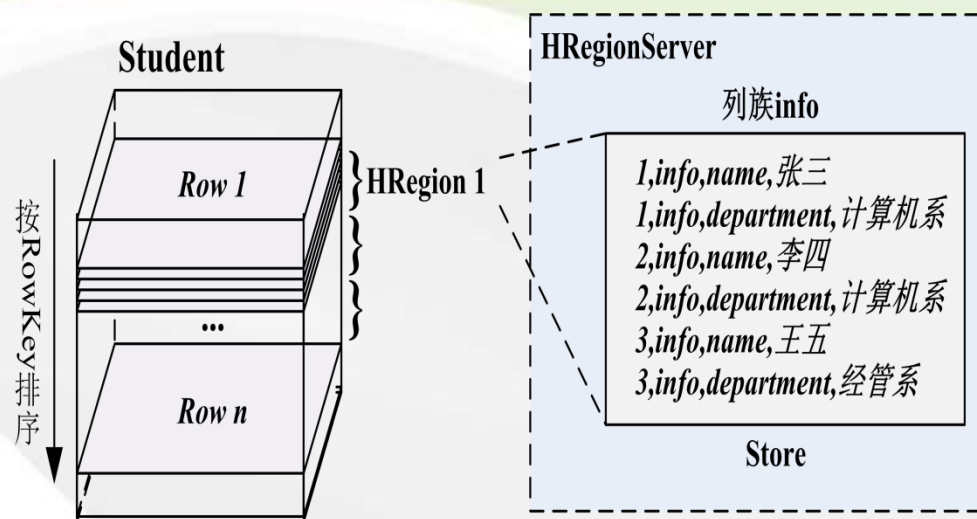
# 高级功能（1） - 联合查询的去范式化实现

- 数据库设计三大范式
  - 属性具有原子性，不可再分解
  - 记录有惟一标识，即实体的惟一性
  - 任何字段不能由其他字段派生出来，要求字段没有冗余
- 去范式化实现联合查询：空间换时间



Student_Course_Regsitration					
行关键字	列族:info		列族course		
	限定词:name	限定词:department	限定词:title	限定词:location	限定词:status
1	张三	计算机系	计算机原理 [t2] 网络原理 [t1]	教3楼303 [t2] 教2楼202 [t1]	pending [t2] ok [t1]
2	李四	计算机系	高等数学	教1楼101	ok
3	王五	经管系	高等数学	教1楼101	ok

# 高级功能（1） - 联合查询的去范式化实现



```
1: table(conf,
"Student_Course_Registration");
2: Get get = new Get("张三");
3: get.setMaxVersions()
4: result=table.get(get)
5: filter("course:status"="OK")
```

# 高级功能（2） - 非主键条件查询

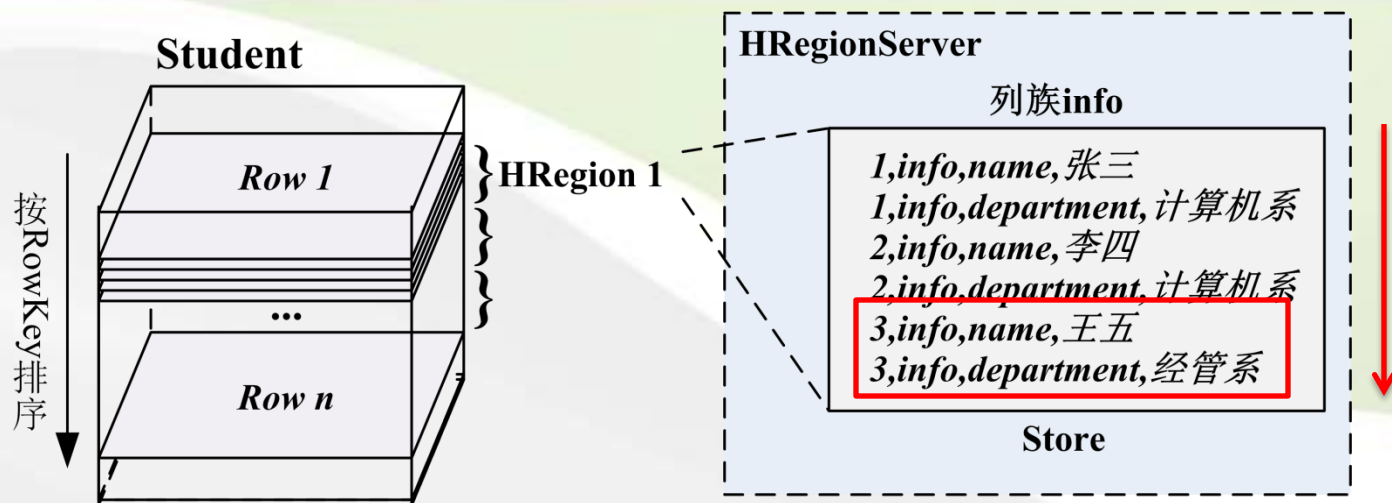
- RDBMS非主键查询：王五的所在系
  - SELECT department FROM Student WHERE name="王五"
- 结果：经管系

Student		
student_id	name	department
1	张三	计算机系
2	李四	建筑系
3	王五	经管系

- HBase如何根据非rowkey的值进行条件查询？

Student		
行关键字	列族:info	
	限定词:name	限定词:department
1	张三	计算机系
2	李四	计算机系
3	王五	经管系

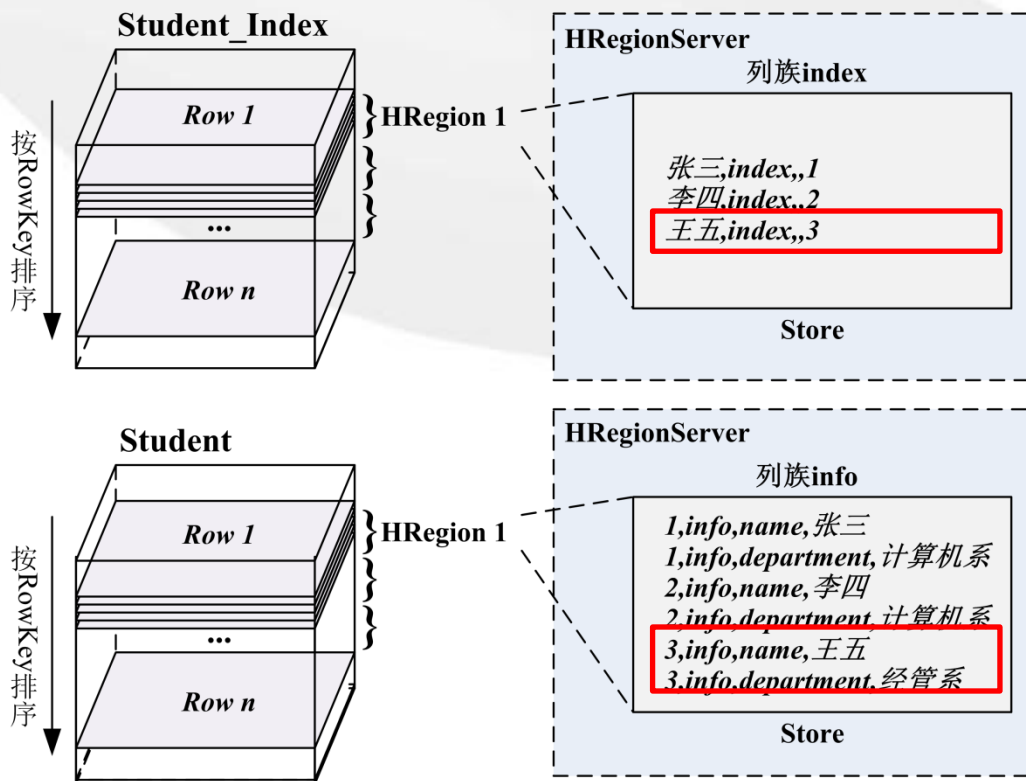
# 高级功能（2） - 非rowkey条件普通查询



```
1: HTable table = new HTable(conf, "student");
2: Scan scan = new Scan();
3: scan.addColumn(toBytes("info"), toBytes("name"));
4: scan.setStartRow(toBytes(1)); // start key is inclusive
5: scan.setStopRow(toBytes(4)); // stop key is exclusive
6: ResultScanner results = table.getScanner(scan);
7: for each result in results
8:   byte[] value = result.getValue(toBytes("info"), toBytes("name"));
9:   if value=="王五"
10:    break
11:   end if
12: end for
```

# 高级功能（2） - 基于索引表的二级索引

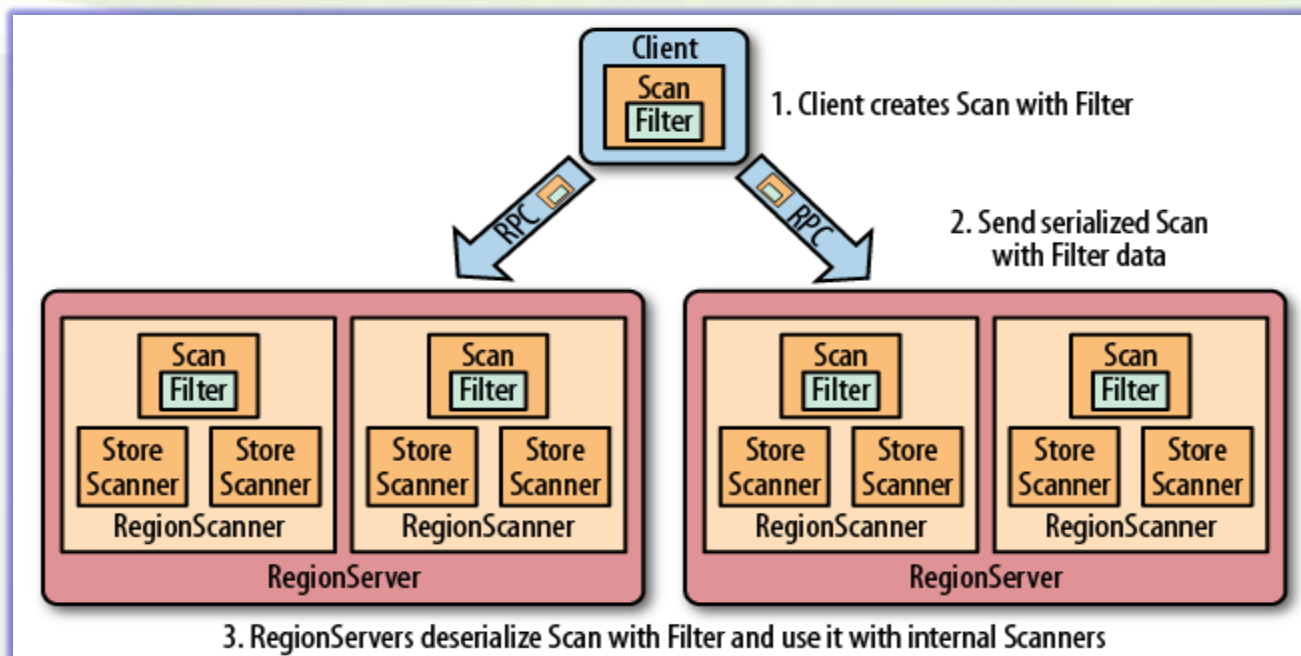
Student_Index	
行关键字	列族:index
	限定词:
张三	1
李四	2
王五	3



```
1: HTable table = new HTable(conf,
    "student_index");
2: rowkey = Bytes.toBytes("王五")
3: Get get = new Get(rowkey);
4: get.addColumn(toBytes("index"));
5: Result result = table.get(get);
6: byte[] id = result.getValue(toBytes("index"));
7: HTable table = new HTable(conf, "student");
8: Get get = new Get(id);
9: get.addColumn(toBytes("info"),
    toBytes("department"));
10: Result result = table.get(get);
11: byte[] department = result.getValue(
    toBytes("index"), toBytes("department"));
12: System.out.println("Department: " +
    toString(department));
```



# 高级功能（3） - 查询过滤



```
1: HTable table = new HTable(conf, "student");
2: Scan scan = new Scan();
3: scan.addColumn(Bytes.toBytes("info"), Bytes.toBytes("name"));
4: scan.setFilter(filter)
5: ResultScanner results = table.getScanner(scan);
6: for each result in results
7:   byte[] value = result.getValue(toBytes("info"), toBytes("name"));
8: end for
```



# 高级功能（3） - 过滤条件

- Comparison Filters
  - RowFilter
  - FamilyFilter
  - QualifierFilter
  - ValueFilter
  - DependentColumnFilter
- Dedicated Filters
  - SingleColumnValueFilter
  - SingleColumnValueExcludeFilter
  - PrefixFilter
  - PageFilter
  - KeyOnlyFilter
  - FirstKeyOnlyFilter
  - TimestampsFilter
  - RandomRowFilter
- Decorating Filters
  - SkipFilter
  - WhileMatchFilters

LESS	小于
LESS_OR_EQUAL	小于等于
EQUAL	等于
NOT_EQUAL	不等于
GREATER_OR_EQUAL	大于等于
GREATER	大于

# 高级功能（3） - 过滤条件示例

- 查询王五的所在系
  - SELECT department FROM Student  
WHERE name="王五"
- 结果：经管系
- Scan代码：

Student		
rowkey	info:name	info:department
1	张三	计算机系
2	李四	建筑系
3	王五	经管系

```
1: HTable table = new HTable(conf, "student");
2: Scan scan = new Scan();
3: scan.addColumn(Bytes.toBytes("info"));
4: SingleColumnValueFilter filter = new SingleColumnValueFilter(
    toBytes("info"), toBytes("name"), CompareOp.EQUAL, toBytes("王五"));
5: scan.setFilter(filter)
6: ResultScanner results = table.getScanner(scan);
7: for each result in results
8:   byte[] department = result.getValue(toBytes("info"), toBytes("department"));
9:   System.out.println("Department: " + toString(department));
10: end for
```

# 高级功能（4） - HBase与MapReduce集成

- HBase提供了与Hadoop包中Mapper和Reducer基础类相近的类

HBase类	功能	MapReduce类
org.apache.hadoop.hbase.mapreduce.TableMapper	mapper的基类	org.apache.hadoop.mapreduce.Mapper
org.apache.hadoop.hbase.mapreduce.TableReducer	reducer的基类	org.apache.hadoop.mapreduce.Reducer
org.apache.hadoop.hbase.mapreduce.TableInputFormat	数据输入类	org.apache.hadoop.mapreduce.InputFormat
org.apache.hadoop.hbase.mapreduce.TableOutputFormat	数据输出类	org.apache.hadoop.mapreduce.OutputFormat

- 还提供了一个TableMapReduceUtil类简化操作

# 高级功能（4） - HBase与MapReduce集成

- 原始表

Access_Log				
行关键字	列族:conent		列族:user	
	限定词:host	限定词:tag	限定词:id	限定词:name
1	cn.sina.news	news	1234	张三
10	cn.weibo	weibo	1234	张三
100	cn.sina.news	news	9988	李四
1000	cn.weibo	weibo	9977	王五

- 相同兴趣用户分组

User_Group	
行关键字	列族:user
	限定词:id
news	1234, 9988
weibo	1234, 9977

- Map : Scan      Reduce : Put

# 高级功能（4） - HBase与MapReduce集成

## ● Mapper代码

```
1: public static class Mapper extends TableMapper
2:     <ImmutableBytesWritable, ImmutableBytesWritable> {
3:     public void map(ImmutableBytesWritable row, Result values, Context context) {
4:         ImmutableBytesWritable id = null;
5:         ImmutableBytesWritable tag = null;
6:         for (KeyValue kv : values.list()) {
7:             if ("user".equals(Bytes.toString(kv.getFamily())))
8:                 && "id".equals(Bytes.toString(kv.getQualifier())) {
9:                 id = new ImmutableBytesWritable(kv.getValue());
10:            }
11:             if ("content".equals(Bytes.toString(kv.getFamily())))
12:                 && "tag".equals(Bytes.toString(kv.getQualifier())) {
13:                 tag = new ImmutableBytesWritable(kv.getValue());
14:            }
15:        }
16:        context.write(tag, id);
17:    }
18: }
```

# 高级功能（4） - HBase与MapReduce集成

## ● Reducer代码

```
1: public static class Reducer extends TableReducer
2:     <ImmutableBytesWritable, ImmutableBytesWritable, ImmutableBytesWritable> {
3:     public void reduce(ImmutableBytesWritable key, Iterable values, Context context) {
4:         String user_group = "";
5:         for(ImmutableBytesWritable val : values) {
6:             user_group += (user_group.length() > 0 ? "," : "") + Bytes.toString(val.get());
7:         }
8:         Put put = new Put(key.get());
9:         put.add(Bytes.toBytes("user"), Bytes.toBytes("id"), Bytes.toBytes(user_group));
10:        context.write(key, put);
11:    }
12: }
```

# 高级功能（4） - HBase与MapReduce集成

## ● Main代码

```
1: public static void main(String[] args) throws Exception {  
2:     Configuration conf = new Configuration();  
3:     conf = HBaseConfiguration.create(conf);  
4:     Job job = new Job(conf, "FindGroup_Example");  
5:     job.setJarByClass(FindGroup.class);  
6:     Scan scan = new Scan();  
7:     scan.addColumn(Bytes.toBytes("user"), Bytes.toBytes("id"));  
8:     scan.addColumn(Bytes.toBytes("content"), Bytes.toBytes("tag"));  
9:     TableMapReduceUtil.initTableMapperJob("Access_Log", scan, FindGroup.Mapper.class,  
10:      ImmutableBytesWritable.class, ImmutableBytesWritable.class, job);  
11:     TableMapReduceUtil.initTableReducerJob("User_group", FindGroup.Reducer.class, job);  
12:     System.exit(job.waitForCompletion(true) ? 0 : 1);  
13: }
```

# 作业

- 本节问题
  - 将access.log ( QQ群共享 ) 文件导入到HBase中 , 并用Shell进行一些简单的查询操作
- 要求 :
  - 将导入结果和Shell查询过程 , 截图发送到 [liujun@bupt.edu.cn](mailto:liujun@bupt.edu.cn)
- 下节课程预告 :
  - HBase应用及性能优化实例

