Spark SQL

Why SQL

- Spark default RDDs —> `(Key, Value)`
- What if not `(Key, Value)`

Why SQL

```
{'name': 'Amy', age: 18,  hobby: 'drinking'}

{'name': 'Greg', age: 60,  hobby: 'fishing'}

{'name': 'Susan', age: 30}
```

Older than 18, With hobbies

```
rdd.filter(lambda d: d['age'] > 18) \
    .filter(lambda d: 'hobby' in d.keys()) \
    .map(lambda d: d['name'])
```

```
{'name': 'Amy', age: 18,  hobby: 'drinking'}
{'name': 'Greg', age: 60,  hobby: 'fishing'}
{'name': 'Susan', age: 30}
```

This is simpler

```
SELECT name

WHERE age > 18

AND hobby IS NOT NULL
```

```
rdd = sc.parallelize(['{"name":"Amy","age":18,"hobby":"drinking"}',
                      '{"name":"Greg","age":60,"hobby":"fishing"}',
                      '{"name":"Susan","age":30}'])

json_rdd = sqlContxt.jsonRDD(rdd)

json_rdd.registerTempTable('jrdd')
rdd2 = sqlContxt.sql('SELECT * FROM jrdd WHERE age > 18 AND hobby IS NOT NULL')

"""Row(age=60, hobby=u'fishing', name=u'Greg')"""
```

SQLContext

- A subcontext of `SparkContext`
- `sqlContext = SQLContext(sc)`

Create a jsonRDD

- **Read in from a file**

```
jrdd = sqlContext.jsonFile(filepath)
```

- Convert from a regular RDD

```
jrdd = sqlContext.jsonRDD(rdd)
```

Querying

- **Create a table from json_rdd**

```
jrdd.registerTempTable('table_name')
```

- **Query from table define above**

```
jrdd2 = jrdd.sql('SELECT * FROM table_name')
```

Inside a jsonRDD

```
Row(age=18, hobby=u'drinking', name=u’Amy')
```

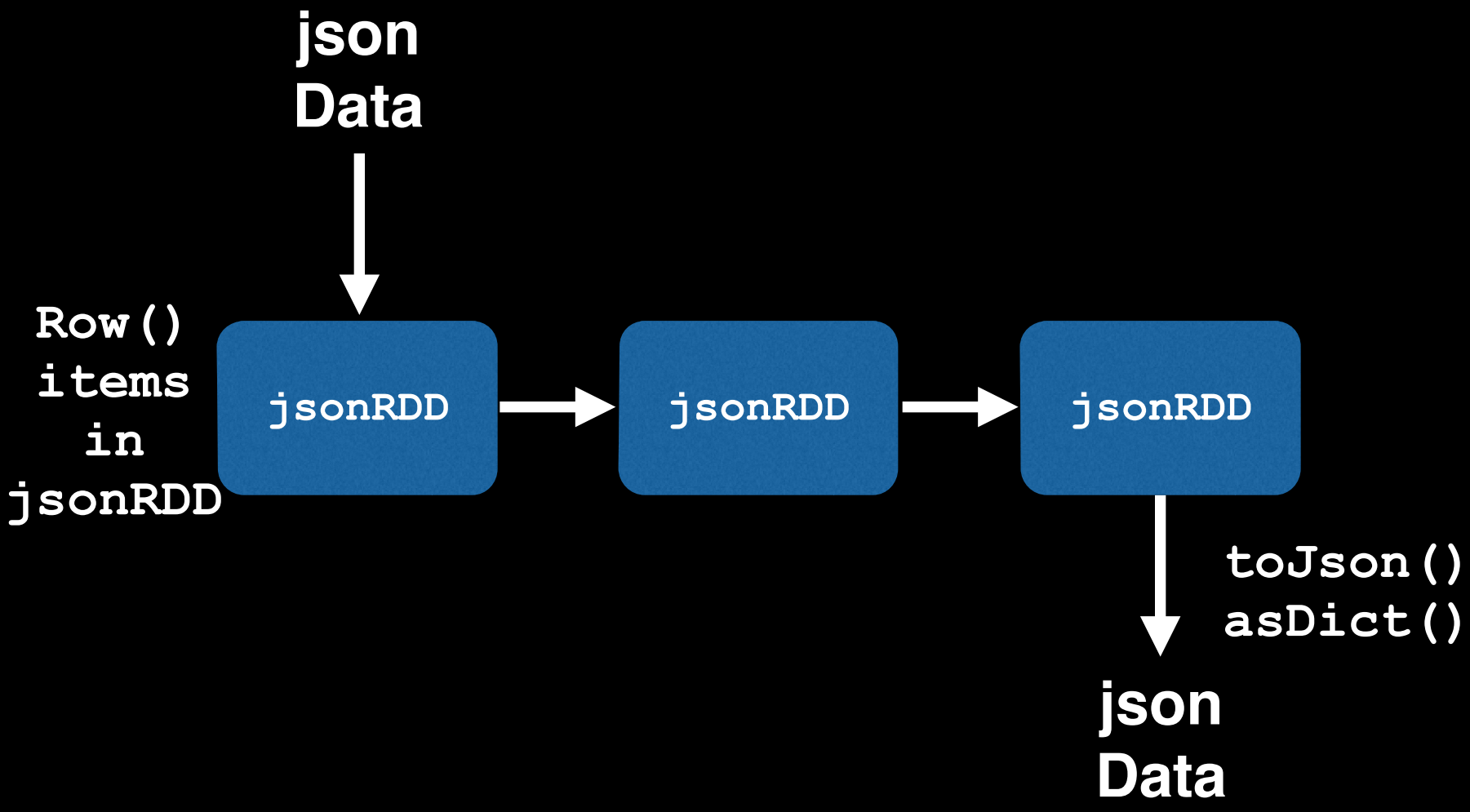
Convert back to json

- **Map `asDict()` to jsonRDD**

```
jrdd.map(lambda row: row.asDict())
```

- **Call `toJson()` on jsonRDD**

```
jrdd.toJson()
```


json
Data
Row()
items
in
jsonRDD
jsonRDD
jsonRDD
jsonRDD
toJson()
asDict()
json
Data

SQL Rules

- Syntax is same as SQL
- Subqueries are allowed

Schema Inference

- Infer schema from the first row
- Important that 1st row has no missing data
- Can also manually define fields

https://spark.apache.org/docs/1.2.0/sql-programming-guide.html

All the other Spark concepts apply to SQL Spark