Работа с большими данными

Содержание курса

01 Введение. Распределенные вычисления. MapReduce.

02 HDFS. Apache Spark. RDD.

03 Базы данных. Spark SQL. Хранение больших данных.

04 Подробнее о модели вычислений Spark. Знакомство со Scala.

05 Алгоритмы машинного обучения на больших данных. spark.ml.

06 Рекомендательные системы. Виды. Их метрики.

07 Обработка потоковых данных. Structured streaming и интеграция с spark.ml.

08 Модели в продакшен. Управление кластером.

3. Базы данных. Spark SQL. Хранение больших данных.

План:

1. Базы данных. СУБД. Нормализация отношений.
2. SQL vs NoSQL.
3. Типы хранения данных в Hadoop. Parquet. ORC.
4. Поддерживаемые форматы. Structured vs unstructured data.
5. Типы join-ов. Схема. SparkSQL.

Базы данных.

Базы данных

1. *БД хранится и обрабатывается в вычислительной системе*.

2. *Данные в БД логически структурированы (систематизированы)* с целью обеспечения возможности их эффективного поиска и обработки в вычислительной системе.

3. *БД включает схему, или метаданные*, описывающие логическую структуру БД в формальном виде (в соответствии с некоторой метамоделью).

Мини-гайд по Базам данных

Системы управления базами данных

СУБД — это общий термин, относящийся ко всем видам абсолютно разных инструментов, от компьютерных программ до встроенных библиотек. Эти приложения управляют или помогают управлять наборами данных. Так как эти данные могут быть разного формата и размера, были созданы разные виды СУБД.

SQL vs No-SQL

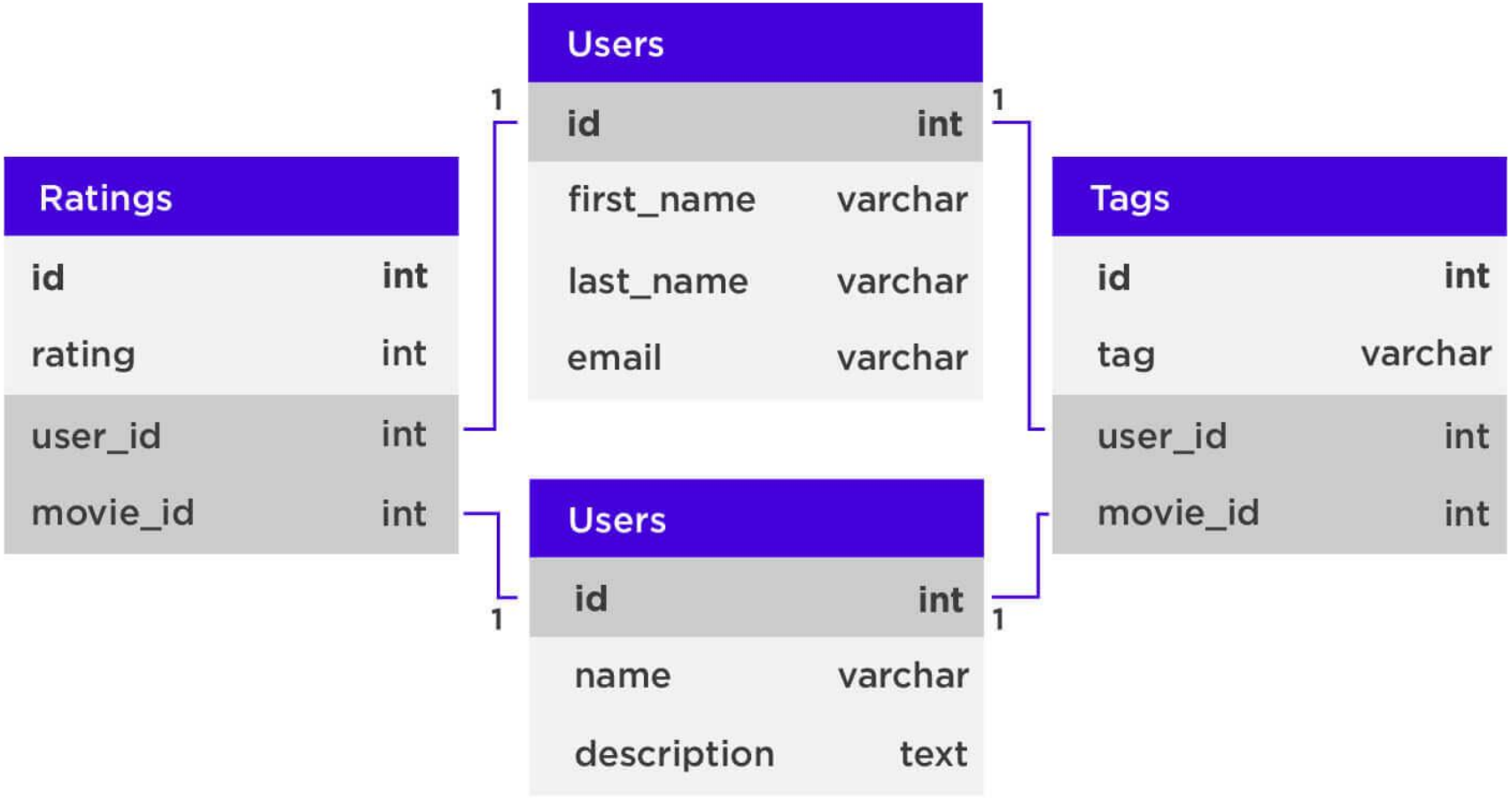
Реляционная модель данных

Реляционные БД

Реляционные СУБД берут своё название от модели БД с которой работают. На данный момент и, наверное, в ближайшем будущем эти СУБД будут наиболее популярным выбором для хранения данных.

Реляционные СУБД используют строго описанные структуры данных - схемы. Схема базы данных включает в себя описание содержания, структуры и ограничений целостности, т.е. она определяет таблицы, поля в каждой таблице, а также отношения между полями и таблицами.

Примеры: SQLite, MySQL, PostgreSQL


Users
id int
first_name varchar
last_name varchar
email varchar
Ratings
id int
rating int
user_id int
movie_id int
Tags
id int
tag varchar
user_id int
movie_id int
Users
id int
name varchar
description text
1
1
1
1

Нормализация отношений

Нормализация отношений

Нормализация отношений (таблиц) — Нормализация имеет своей целью избавиться от избыточности в отношениях и модифицировать их структуру таким образом, чтобы процесс работы с ними не был обременён различными посторонними сложностями.

Нормализация баз данных заключается в приведении структуры хранения данных к нормальным формам (NF). Всего таких форм существует 8, но часто достаточным является соблюдение первых трех.

Мини-гайд по нормальным формам

Идентификатор предмета	Наименование предмета	Материал
1	Стул	Металл
2	Стол	Массив дерева
3	Кровать	ЛДСП
4	Шкаф	Массив дерева
5	Комод	ЛДСП

Идентификатор предмета	Наименование предмета	Материал
1	Стул	Металл
2	Стол	Натуральное дерево
3	Кровать	ЛДСП
4	Шкаф	Массив дерева
5	Комод	ЛДСП

Идентификатор предмета	Наименование предмета	Материал
1	Стул	Металл
2	Стол	Натуральное дерево
3	Кровать	ЛДСП
4	Шкаф	Массив дерева
5	Комод	ЛДСП
6	Тумба	Дерево

Идентификатор предмета	Наименование предмета	Идентификатор материала
1	Стул	2
2	Стол	1
3	Кровать	3
4	Шкаф	1
5	Комод	3

Идентификатор материала	Материал
1	Массив дерева
2	Металл
3	ЛДСП

ACID

ACID

Это набор из четырех требований к транзакционной системе, обеспечивающих максимально надежную и предсказуемую работу. Не все базы данных полностью реализуют ACID.

Атомарность (atomicity)

Атомарность гарантирует, что каждая транзакция будет выполнена полностью или не будет выполнена совсем. Не допускаются промежуточные состояния.

Согласованность (consistency)

Согласованность — это требование, подразумевающее, что в результате работы транзакции данные будут допустимыми. Это вопрос не технологии, а бизнес-логики: например, если количество денег на счете не может быть отрицательным, логика транзакции должна проверять, не выйдет ли в результате отрицательных значений.

Изолированность (isolation)

Гарантия того, что параллельные транзакции не будут оказывать влияния на результат других транзакций. Мы разобрались с изоляцией выше.

Долговечность (durability)

Изменения, получившиеся в результате транзакции, должны оставаться сохраненными вне зависимости от каких-либо сбоев. Иначе говоря, если пользователь получил сигнал о завершении транзакции, он может быть уверен, что данные сохранены.

No-SQL

NoSQL обычно означает не только SQL: то есть базовая база данных имеет свойства, отличные от свойств обычных и традиционных систем баз данных. Таким образом, нет четкого различия, которое квалифицирует базу данных как NoSQL, кроме того факта, что они не обеспечивают характеристики соответствия ACID.
NoSQL-базы оптимизированы для приложений, которые должны быстро, с низкой временной задержкой (low latency) обрабатывать большой объем данных с разной структурой. Таким образом, нереляционные хранилища непосредственно ориентированы на Big Data.

Всего есть 4 основных типа:

1. Хранилище типа ключ-значение
2. Документо-ориентированное хранилище
3. Колоночное хранилище
4. Графовое хранилище

Хранилище ключ-значение

Основано на большой хеш-таблице, содержащей ключи и значения. Применяются для хранения изображений, создания специализированных файловых систем, в качестве кэшей для объектов, а также в масштабируемых Big Data системах, включая игровые и рекламные приложения, а также проекты интернета вещей

Примеры: Riak, Amazon DynamoDB, Redis;

Документо-ориентированное хранилище

Хранит документы, состоящие из тегированных элементов. Данные, представлены парами ключ-значение затем сжимаются в виде полуструктурированного документа из тегированных элементов, в виде похожим например на JSON. Хорошо подходит для каталогов, пользовательских профилей, где каждый документ уникален и изменяется со временем.

Пример: CouchDB, MongoDB;

Колоночное хранилище

Колоночные базы так и называются потому, что хранят данные не в строках а в колонках. Каждая колонка — это отдельная таблица из одной колонки, которая хранит только свои значения. Наличие временных меток (timestamp) позволяет использовать такие СУБД для организации счётчиков, регистрации и обработки событий, связанных со временем: системы биржевой аналитики, IoT/IIoT-приложения, систему управления содержимым и т.д.

Примеры: HBase, Cassandra;

Графовое хранилище

Графовые базы данных предназначены для хранения взаимосвязей и навигации в них. Она использует узлы и рёбра для отображения и хранения данных. Такие СУБД используются в задачах, ориентированных на связи: социальные сети, выявление мошенничества, маршруты общественного транспорта, дорожные карты, сетевые топологии.

Пример: Neo4J.

Форматы хранения данных в Hadoop

Row-Oriented vs Column-Oriented

Row-oriented: rows stored sequentially in a file

Key	Fname	Lname	State	Zip	Phone	Age	Sales
1	Bugs	Bunny	NY	11217	(123) 938-3235	34	100
2	Yosemite	Sam	CA	95389	(234) 375-6572	52	500
3	Daffy	Duck	NY	10013	(345) 227-1810	35	200
4	Elmer	Fudd	CA	04578	(456) 882-7323	43	10
5	Witch	Hazel	CA	01970	(567) 744-0991	57	250

Column-oriented: each column is stored in a separate file
Each column for a given row is at the same offset.

Key
1
2
3
4
5

Fname
Bugs
Yosemite
Daffy
Elmer
Witch

Lname
Bunny
Sam
Duck
Fudd
Hazel

State
NY
CA
NY
CA
CA

Zip
11217
95389
10013
04578
01970

Phone
(123) 938-3235
(234) 375-6572
(345) 227-1810
(456) 882-7323
(567) 744-0991

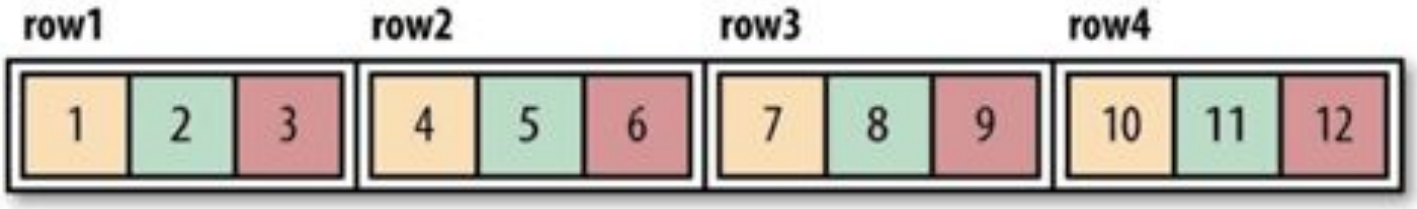
Age
34
52
35
43
57

Sales
100
500
200
10
250

Гайд по форматам хранения данных

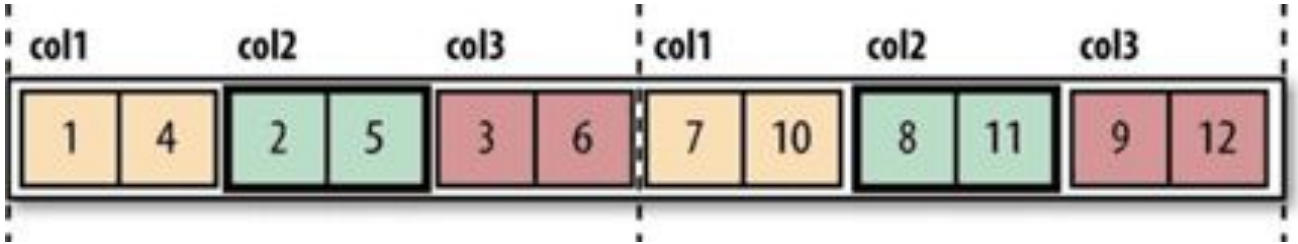
Линейные форматы

В линейных форматах (AVRO, Sequence) строки данных одного типа хранятся вместе, образуя непрерывное хранилище. Даже если необходимо получить лишь некоторые значения из строки, все равно вся строка будет считана с диска в память. Линейный способ хранения данных обусловливает пониженную скорость операций чтения и выполнении избирательных запросов, а также больший расход дискового пространства.



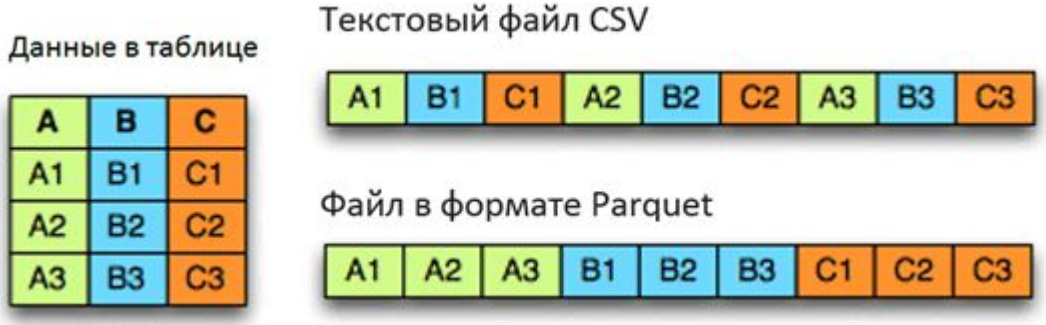
Колоночные форматы

В колоночно-ориентированных форматах (Parquet, RCFile, ORCFile) файл разрезается на несколько столбцов данных, которые хранятся вместе, но могут быть обработаны независимо друг от друга. Такой метод хранения информации позволяет пропускать ненужные столбцы при чтении данных, что существенно ускоряет чтение данных и отлично подходит в случае, когда необходим небольшой объем строк или выполняются избирательные запросы, как, например, в СУБД Apache Hive.



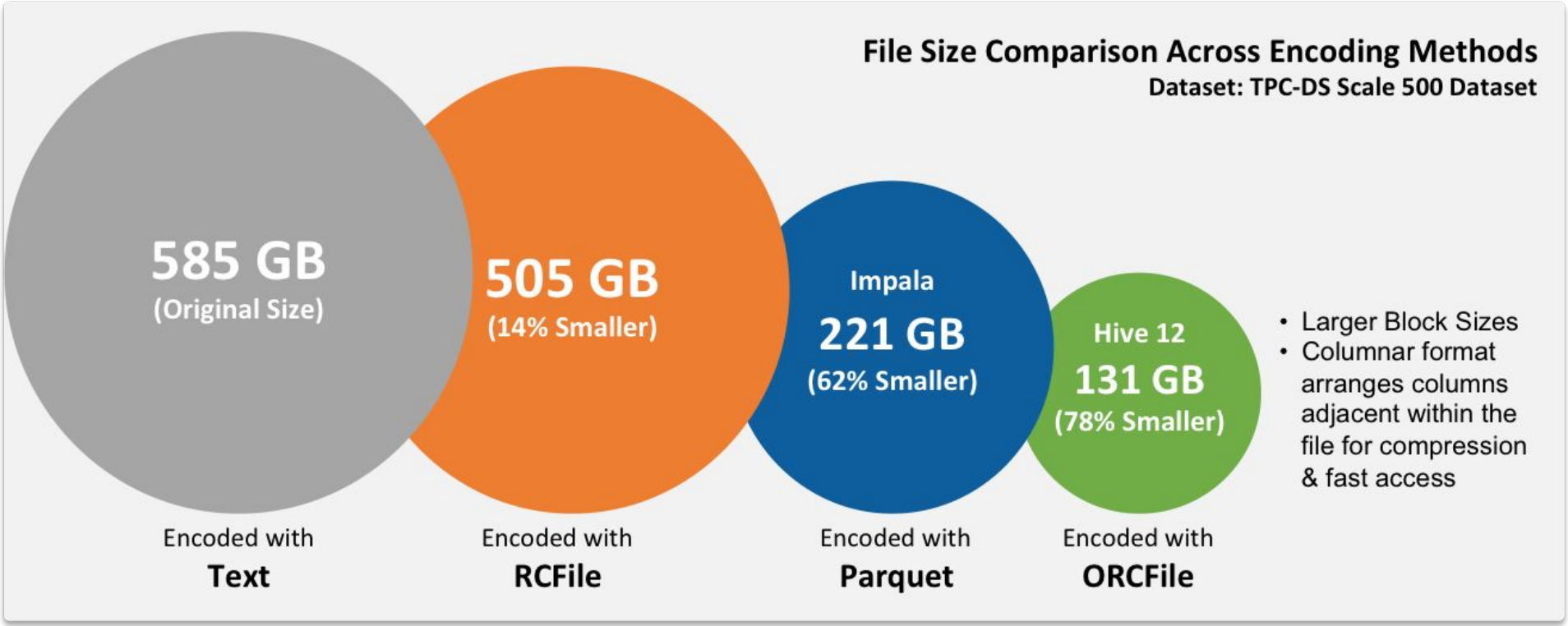
Parquet

Parquet - это столбчатый формат хранения данных. Это помогает повысить производительность, иногда значительно, разрешая хранение и доступ к данным для каждого столбца. Интуитивно понятно, что если бы вы работали с файлом размером 1 ГБ со 100 столбцами и 1 миллионом строк и хотели запрашивать данные только из одного из 100 столбцов, возможность доступа только к отдельному столбцу была бы более эффективной, чем доступ ко всему файл.



ORC

ORC расшифровывается как Optimized Row-Columnar. В некотором смысле это дополнительный уровень оптимизации по сравнению с чистыми столбцовыми форматами, такими как Parquet. ORCFiles хранит данные не только по столбцам, но и по строкам, также известным как полосы. Таким образом, файл с данными в табличном формате можно разделить на несколько более мелких полос, каждая из которых состоит из подмножества строк из исходного файла. При таком разделении данных, если пользовательской задаче требуется доступ только к небольшому фрагменту данных, процесс может опросить конкретную полосу, содержащую данные.


File Size Comparison Across Encoding Methods
Dataset: TPC-DS Scale 500 Dataset
585 GB
(Original Size)
505 GB
(14% Smaller)
Impala
221 GB
(62% Smaller)
Hive 12
131 GB
(78% Smaller)
• Larger Block Sizes
• Columnar format arranges columns adjacent within the file for compression & fast access
Encoded with
Text
Encoded with
RCFile
Encoded with
Parquet
Encoded with
ORCFile

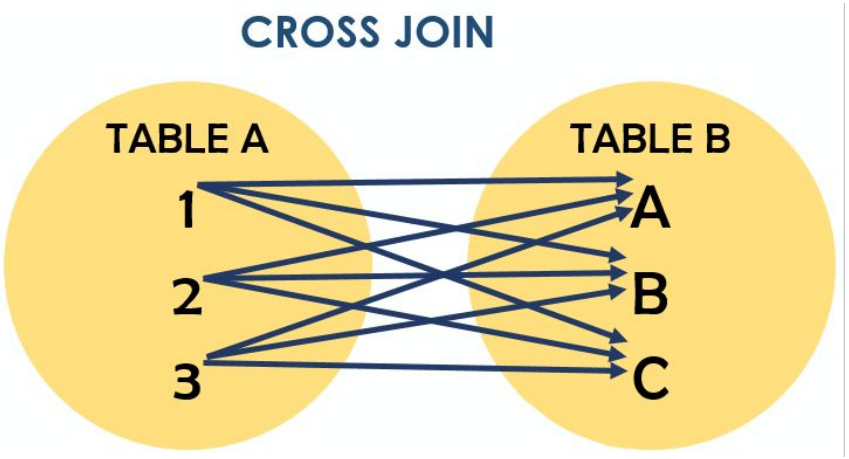
Типы Join-ов

Перекрестное объединение (CROSS JOIN)

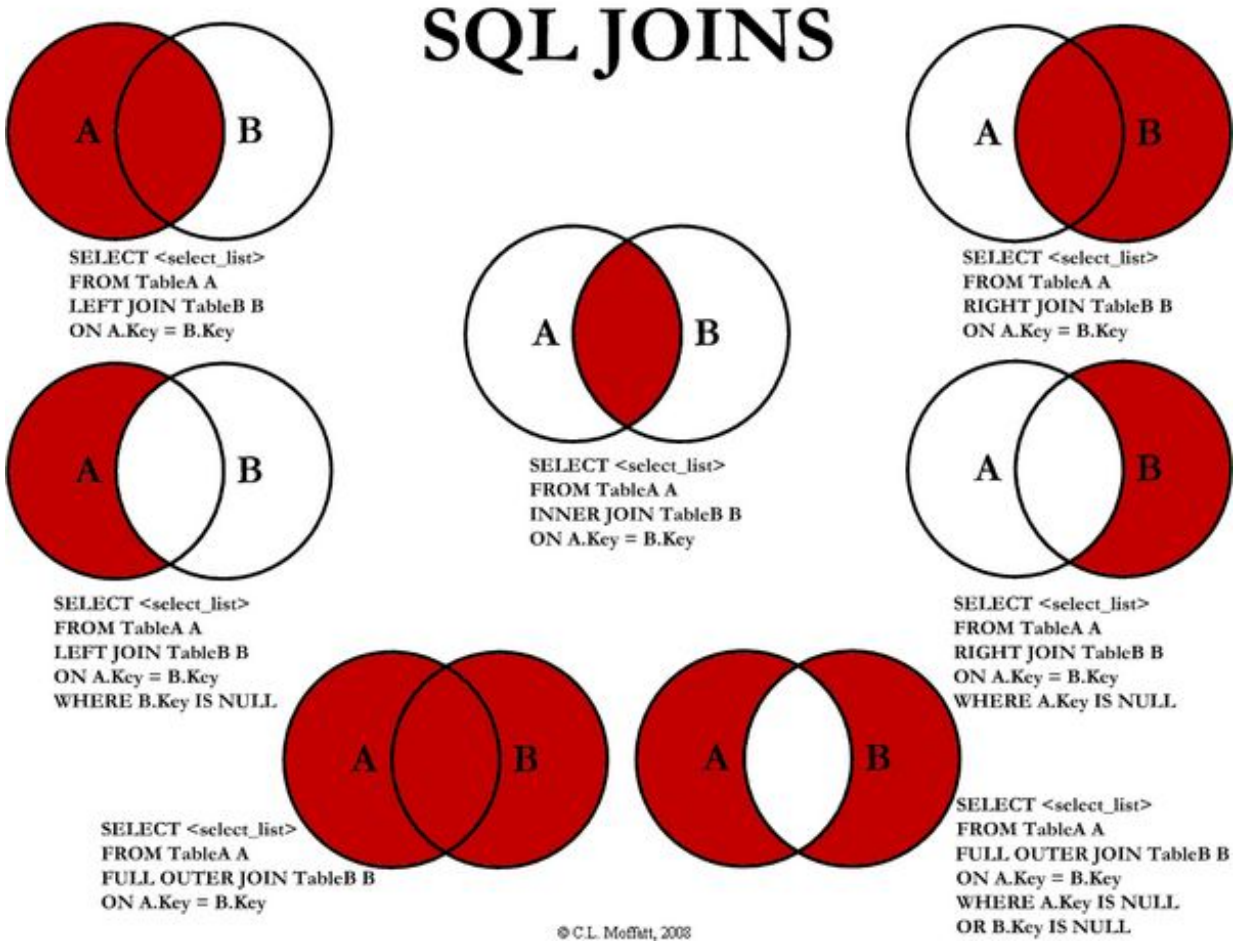
Оператор *перекрёстного соединения*, или *декартова произведения* CROSS JOIN соединяет две таблицы. Порядок таблиц для оператора неважен, поскольку оператор является коммутативным.

Заголовок таблицы-результата является объединением (конкатенацией) заголовков соединяемых таблиц.

Тело результата логически формируется следующим образом. Каждая строка одной таблицы соединяется с каждой строкой второй таблицы, давая тем самым в результате все возможные сочетания строк двух таблиц.



Joins

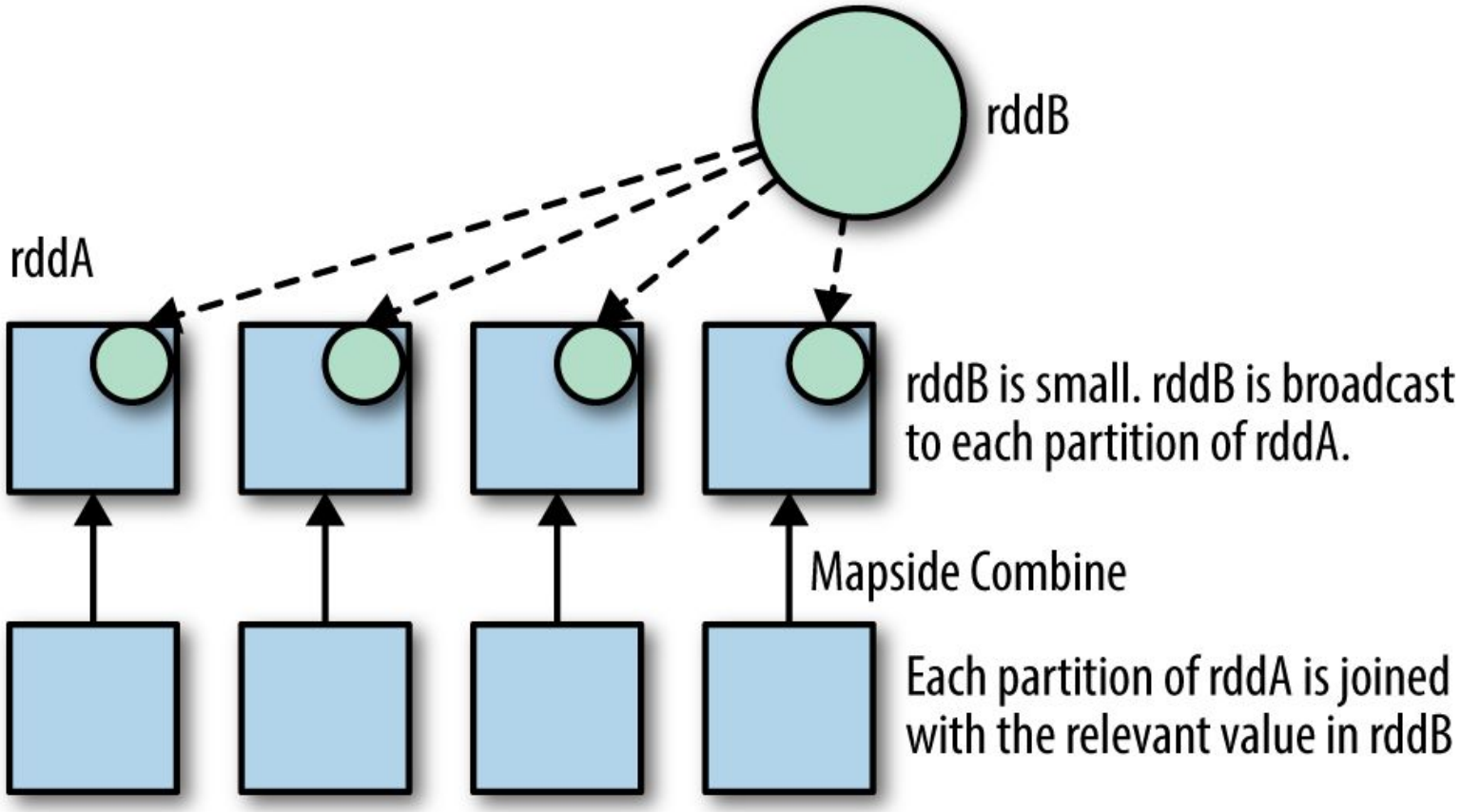

SQL JOINS
A
B
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
© C.L. Moffatt, 2008

Shared Variables

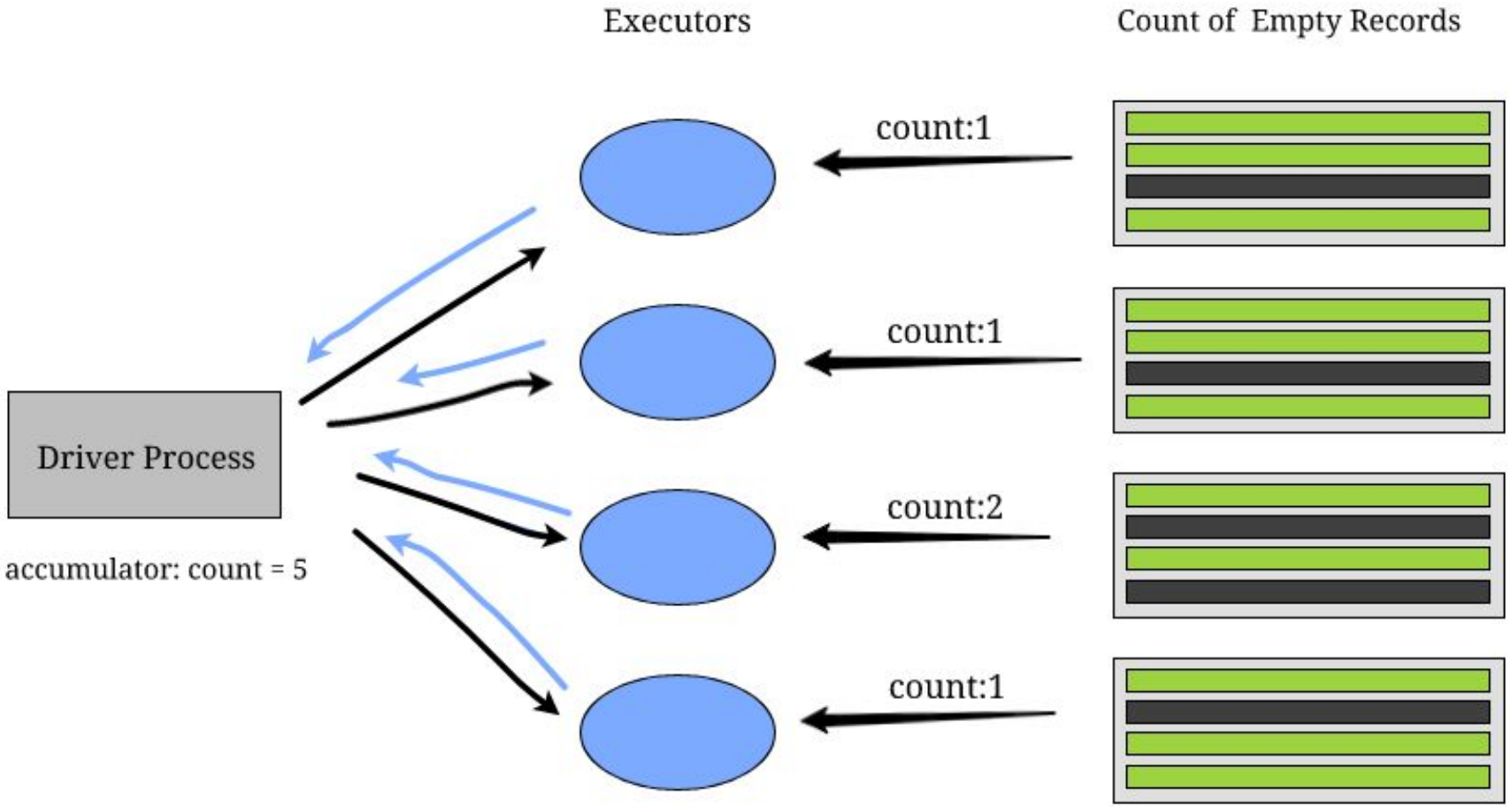
Shared Variables

Обычно, когда функция, переданная в операцию Spark (например, map или reduce), выполняется на удаленном узле кластера, она работает с отдельными копиями всех переменных, используемых в функции. Эти переменные копируются на каждую машину, и никакие обновления переменных на удаленной машине не передаются обратно программе драйвера. Поддержка общих переменных для чтения и записи в задачах была бы неэффективной. Однако Spark предоставляет два ограниченных типа общих переменных для двух распространенных моделей использования: broadcast и accumulators.

Broadcast


rddB
rddA
rddB is small. rddB is broadcast
to each partition of rddA.
Mapside Combine
Each partition of rddA is joined
with the relevant value in rddB

Accumulators


Executors
Count of Empty Records
count:1
count:1
count:2
count:1
Driver Process
accumulator: count = 5