L21: Joins 2

CS3200 Database design (sp18 s2)

https://course.ccs.neu.edu/cs3200sp18s2/

4/2/2018

Announcements!

- Please pick up your exam if you have not yet
- Changed class calendar
- Outline today
  - Joins
  - Relational algebra
- Next class
  - Query Optimizations

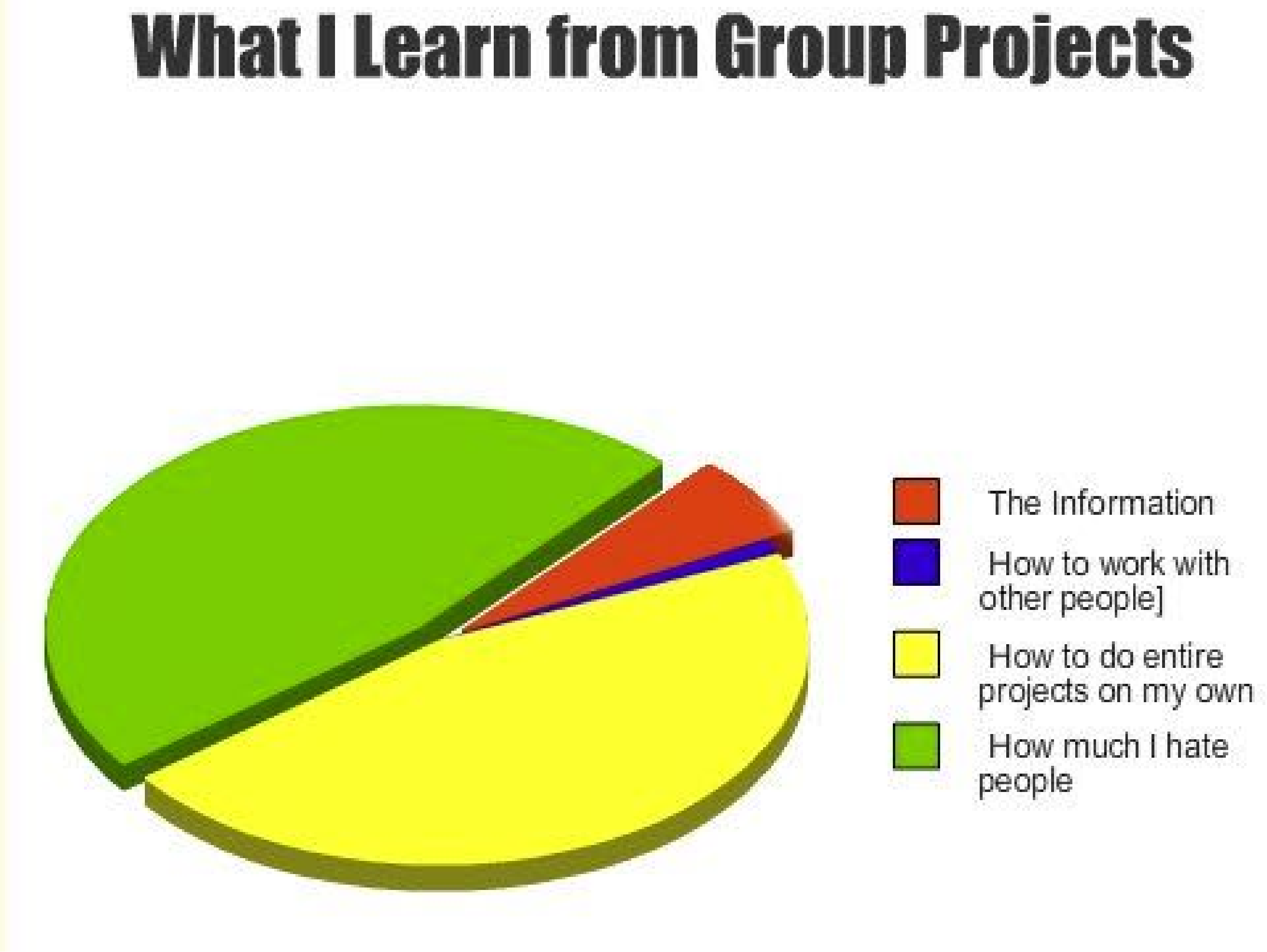
Query Processing and Database Internals				
17	M Mar 19	**Exam 2** I/O Cost Models & Merge Sort	GUW Ch 11.4	
18	R Mar 22	I/O Cost Models & External Sort	GUW Ch 11.4	Q8
19	M Mar 26	Indexing and B+ trees	GUW Ch 13.1-13.3	
20	R Mar 29	Joins 1	GUW Ch 15.9	
21	M Apr 2	Joins 2, Relational Algebra	GUW Ch 2 and 16.3	HW5
22	R Apr 5	Relational Algebra, Query Optimization, NoSQL	GUW Ch 5	P2 (R 4/5), Q9 (FR 4/6)
23	M Apr 9	NoSQL	GUW Ch 8 and 14	
NoSQL				
24	R Apr 12	NoSQL, Class Review and Course Evaluation		Q10 (optional)
	M Apr 16	No class: Patriot's day		HW6 (TU 4/17)
	R Apr 19	No class: Reading day		
	M Apr 23	**Exam 3** (1-3pm, location TBD)		

Extra contributions is now closed

A total of **14** vote(s) in **69** hours

7 (50% of users)	Great! Yes to optional homework. It may increase my contribution score and may help others too. I think PPTX is the right expressive format.
6 (43% of users)	Great! Yes to optional homework. It may increase my contribution score and may help others too. I think quizlet is the the better despite more restricted format.
1 (7% of users)	No! You are the instructor. Students should not be asked to create helpful examples for other students.

Group Projects: what is your experience?



Source: Found on the Web as variation of http://www.inquisitr.com/160288/graph-what-i-learned-from-group-projects/

Peer Evaluation Form

Motivation

Use this form to **evaluate the contributions of your group members** to the project. Failure to hand in this form will lead to the assumption that you did not contribute anything to your group project. Note that these evaluations are confidential and will **never be shown to your group members**. Please respond as honestly and professionally as possible!

Group contributions encompass issues such as: *1. Group Participation* (attends meetings regularly and is on time), *2. Time Management & Responsibility* (accepts fair share of work and reliably completes it by the required time), *3. Adaptability* (displays or tries to develop a wide range of skills, readily accepts changed approach or constructive criticism), *4. Creativity/Originality* (problem-solves when faced with impasses or challenges, originates new ideas, initiates team decisions), *5. Communication Skills*: (effective in discussions, good listener, capable presenter, proficient at diagramming, representing, and documenting work), *6. General Team Skills* (positive attitude, encourages and motivates team, supports team decisions, helps team reach consensus, helps resolve conflicts in the group).

The list above is just for guidance and we do not ask you to evaluate your group members in each of those aspects individually. Instead, you will give one single evaluation based on how much and **how effective you think that they have overall contributed** to the group's final products. Note that more effective individuals can contribute more to the output with less time committed. Life is not about time present, but ultimate impact.

Evaluation

Please allocate a **total of 100±1 points** among your team members, including yourself, with **higher points going to those members who contributed most**, e.g., (33/33/33/33) or (20/20/20/20/20) if all team members contributed equally, or 70/30 (40/40/20) if one member was substantially working less, or (50/25/25) if one member has contributed double than others, etc. Put yourself in the first slot, and if you feel that you have not participated as much as other people, rate yourself accordingly.

Your Group #: ____ Focus of your Group Project: ____

	Name	Points
Yourself:		
Member 2:		
Member 3:		
Member 4:		
(Member 5):		
	Total (you will need to do the math):	**100**

Explanations

In the box below, you can optionally assess your team member's individual contributions with descriptive comments. For example, did any team member(s) make less than average contributions? Were there any special circumstances that resulted in the less than average contribution (e.g., extended sickness)? Did any team member(s) make outstanding contributions? Was there any team member you helped you understand topics you previously did not understand? Was there a change in your team members (e.g., dropped or added)? Or you feel that your point allocation needs further explanations? Please describe briefly giving concrete examples.

Naming Convention

Assuming your name is "Paul Kalkbrenner" and your group number is "3", then rename this Word document as "**G03_Kalkbrenner.pdf**" (note the two digits for group number). Submit the file on Blackboard by the specified deadline.

BNLJ: Some quick facts.

- We use M buffer pages as:
  - 1 page for S
  - 1 page for output
  - M-2 Pages for R

$$\mathrm{P}(R) + \frac{P(R)}{M-2} P(S) + \text{OUT}$$

- If P(R) <= M-2
  - then we do one pass over S, and we run in time P(R) + P(S) + OUT.
  - Note: This is optimal for our cost model!
  - Thus, if min {P(R), P(S)} <= M-2 we should always use BNLJ
    - ~~We use this at the end of hash join. We define end condition, one of the buckets is smaller than M-2!~~

Smarter than Cross-Products: From Quadratic to Nearly Linear

- All joins that compute the ***full cross-product*** have some **quadratic** term
  - For example we saw:

NLJ $P(R) + T(R)P(S) + OUT$

BNLJ $P(R) + \frac{P(R)}{M-2} P(S) + OUT$

- Next we'll see some (nearly) linear joins:
  - ~ O(P(R) + P(S) + OUT), where again OUT could be quadratic but is usually better

We get this gain by ***taking advantage of structure***- moving to equality constraints ("equijoin") only!

Index Nested Loop Join (INLJ)

```
Compute R ⋈ S on A:
  Given index idx on S.A:
    for r in R:
      s in idx(r[A]):
        yield r,s
```

Cost:

P(R) + T(R)**L* + OUT

where ***L*** is the IO cost to access all the distinct values in the index; assuming these fit on one page, $L \sim 3$ is good est.

→ We can use an **index** (e.g. B+ Tree) to ***avoid doing the full cross-product!***

Better Join Algorithms

- 2. Sort-Merge Join (SMJ)

- ~~3. Hash Join (HJ)~~

- ~~Comparison: SMJ vs. HJ~~

2. Sort-Merge Join (SMJ)

What we will learn next

- Sort-Merge Join
- “Backup” & Total Cost
- Optimizations

Sort Merge Join (SMJ): Basic Procedure

- To compute $\mathrm{R} \bowtie S\ on\ A$:

- Sort R, S on A using **external merge sort**

- **Scan** sorted files and "merge"

- [May need to "backup"- see next subsection]

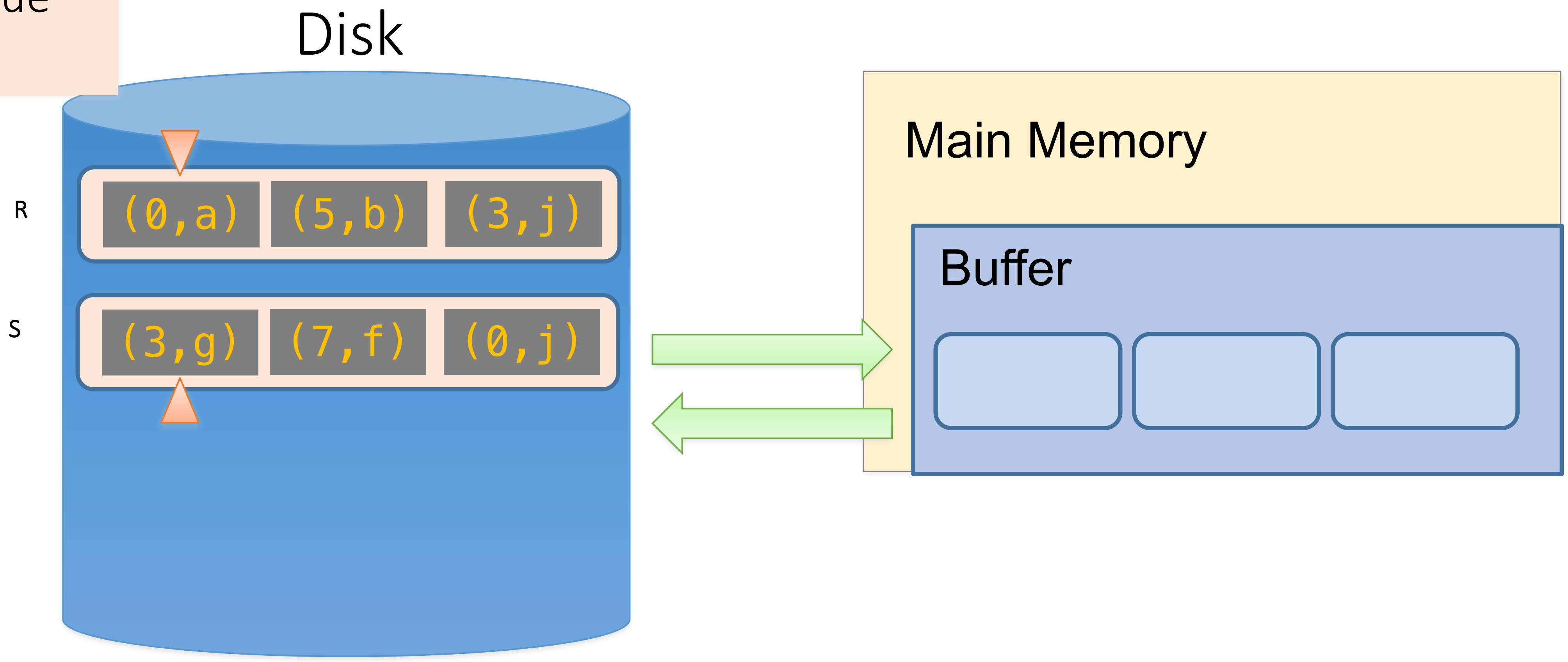
Note that we are only considering equality join conditions here

Note that if R, S are already sorted on A, SMJ will be awesome!

SMJ Example: $R \bowtie S$ *on A* with 3 page buffer

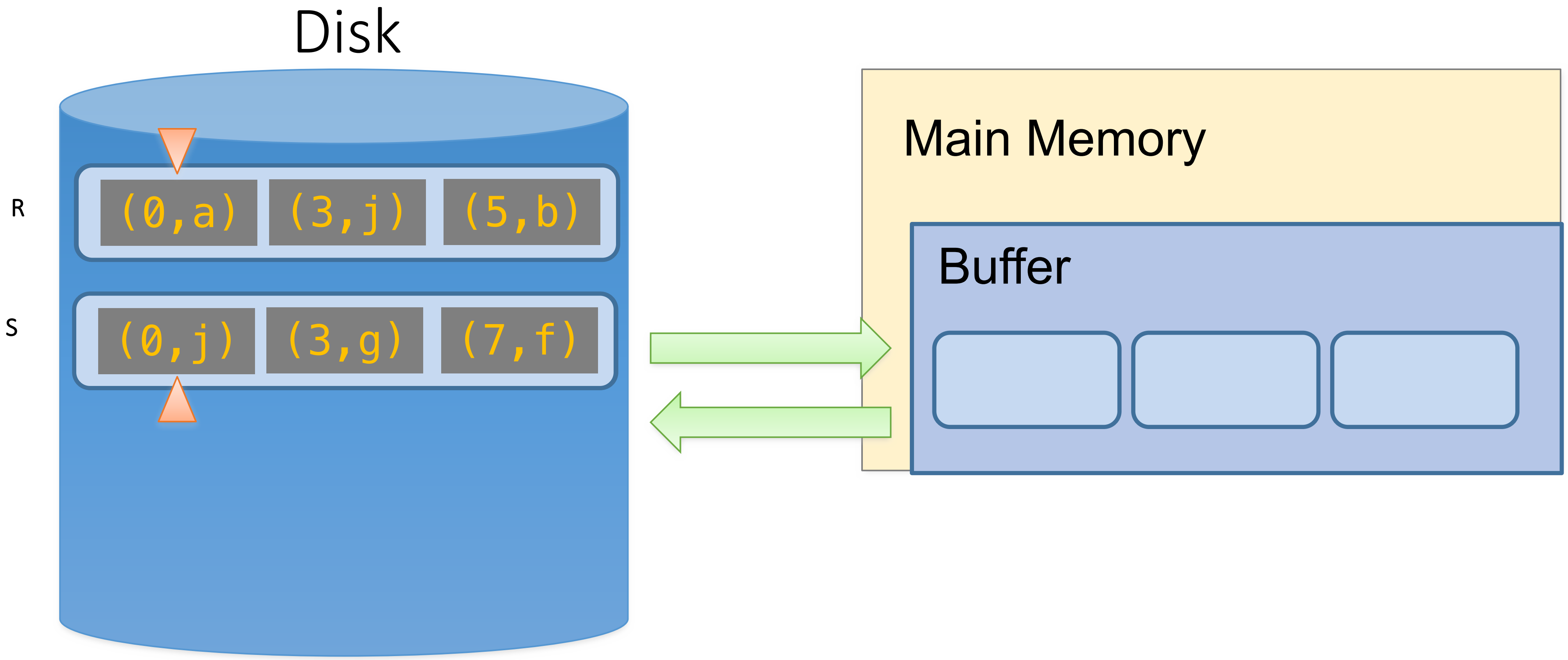
- For simplicity: Let each page be one tuple, and let the first value be A

We show the file HEAD, which is the next value to be read!



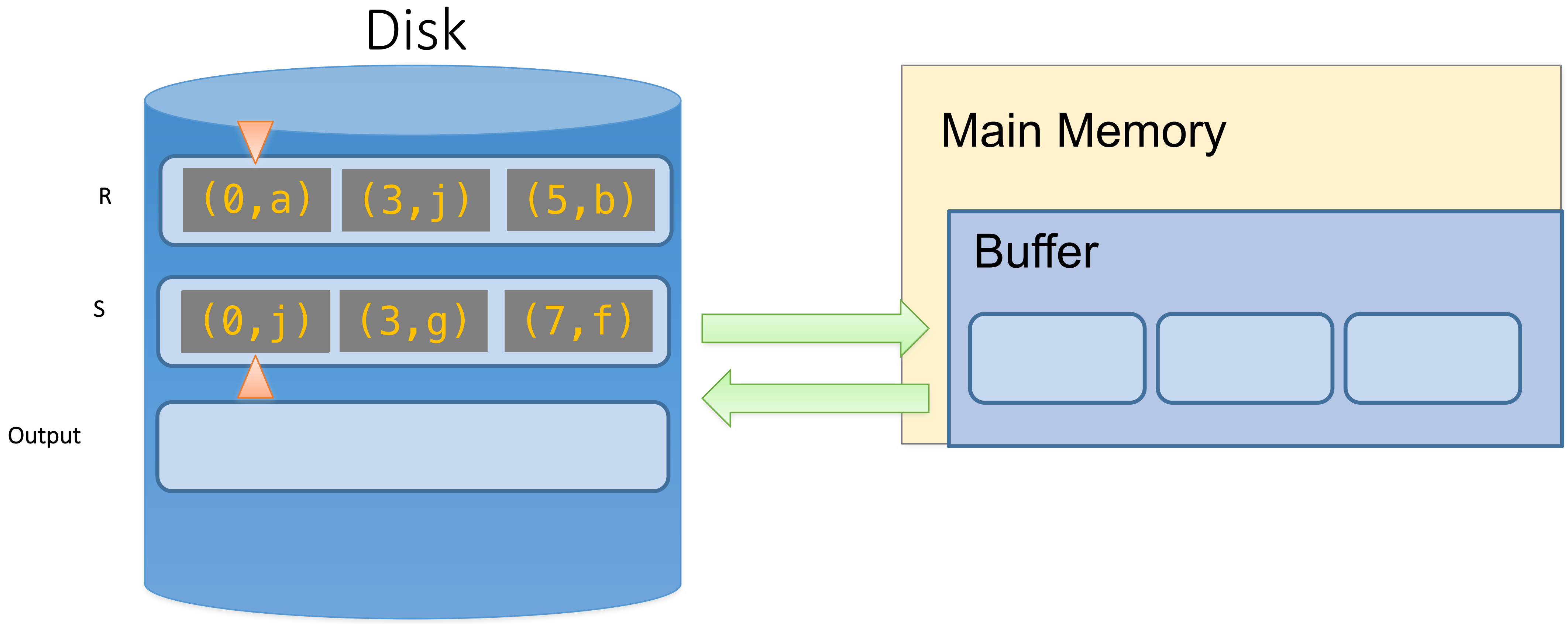
SMJ Example: $R \bowtie S$ on A with 3 page buffer

- 1. Sort the relations R, S on the join key (first value)



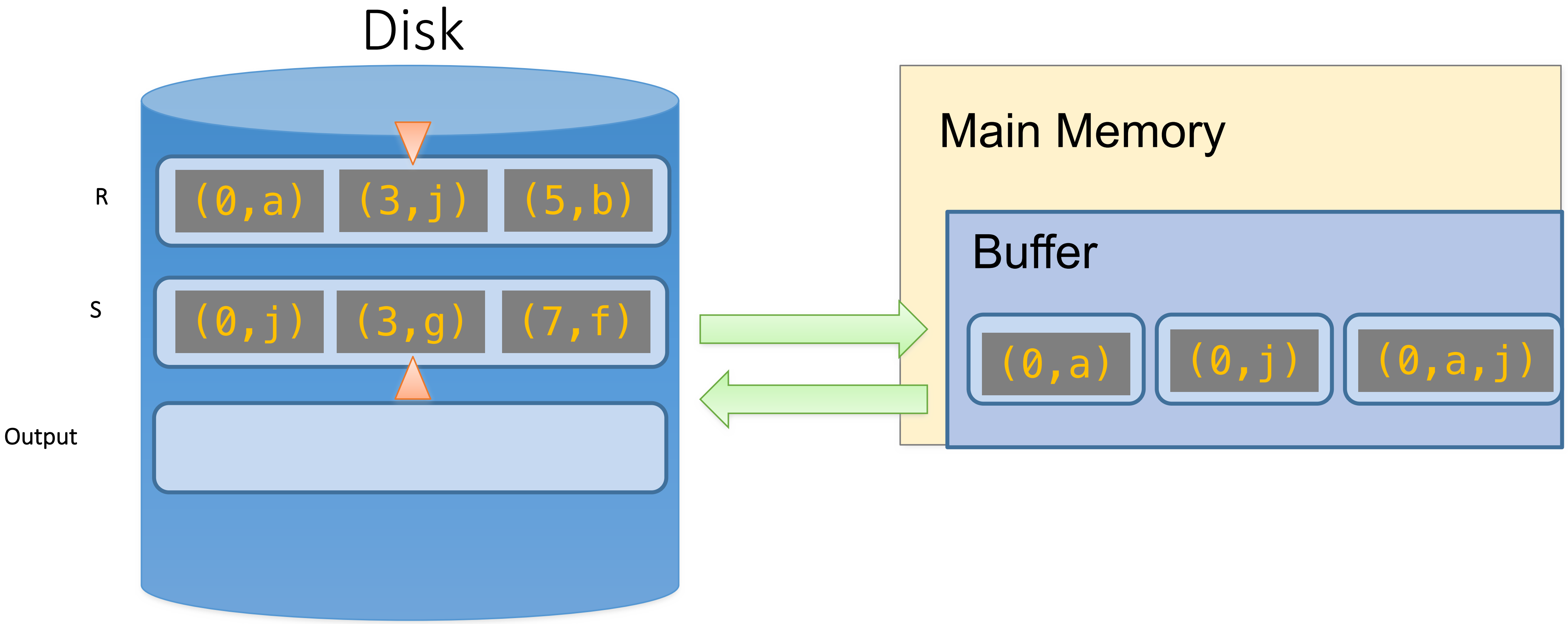
SMJ Example: $R \bowtie S$ *on* A with 3 page buffer

- 2. Scan and "merge" on join key!



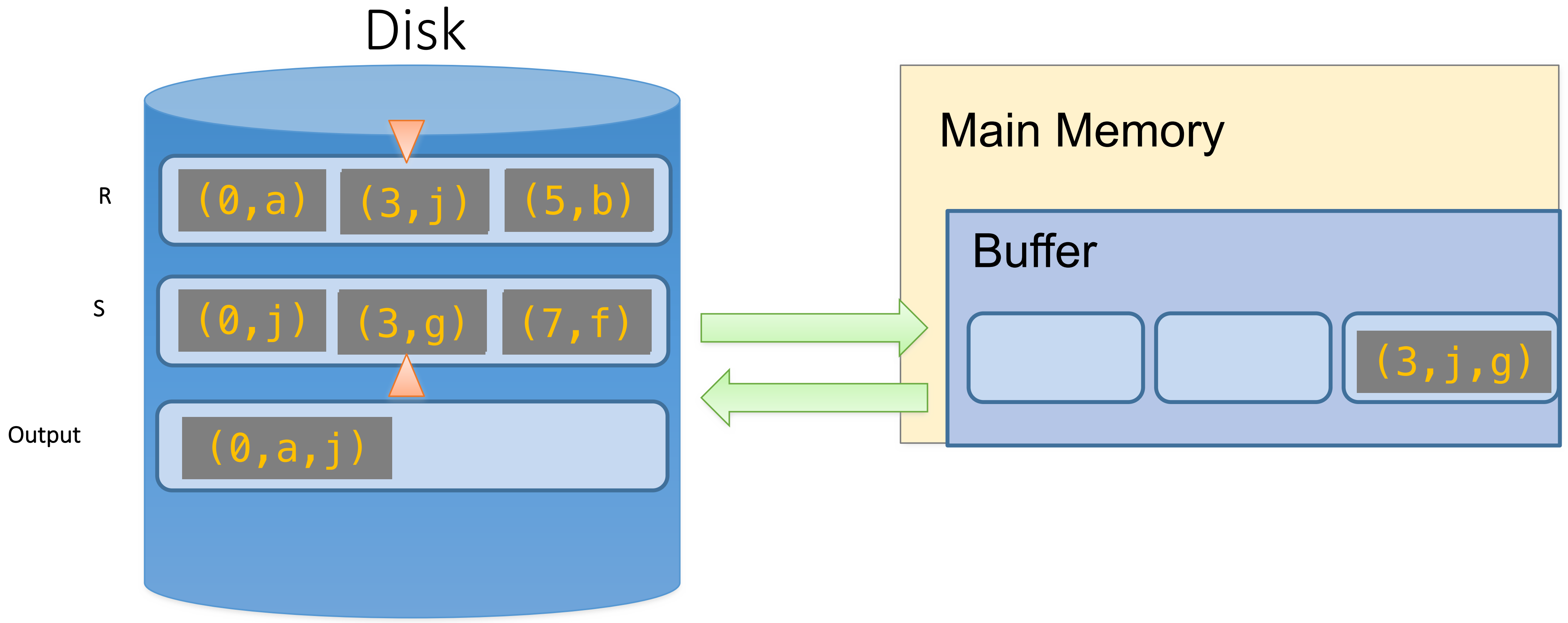
SMJ Example: $R \bowtie S$ *on A* with 3 page buffer

- 2. Scan and "merge" on join key!



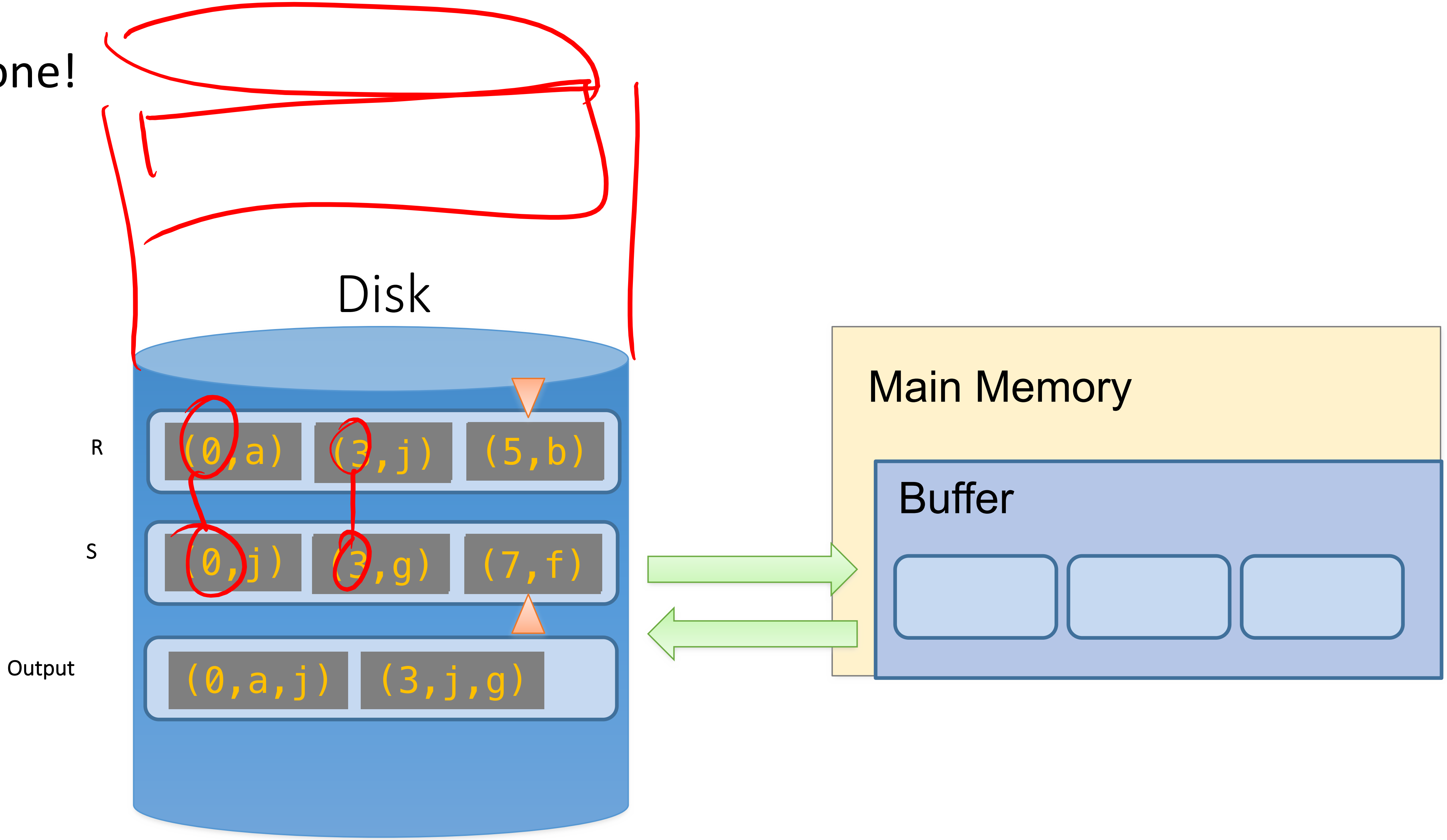
SMJ Example: $\mathrm{R} \bowtie S\ on\ A$ with 3 page buffer

- 2. Scan and “merge” on join key!



SMJ Example: $R \bowtie S$ *on A* with 3 page buffer

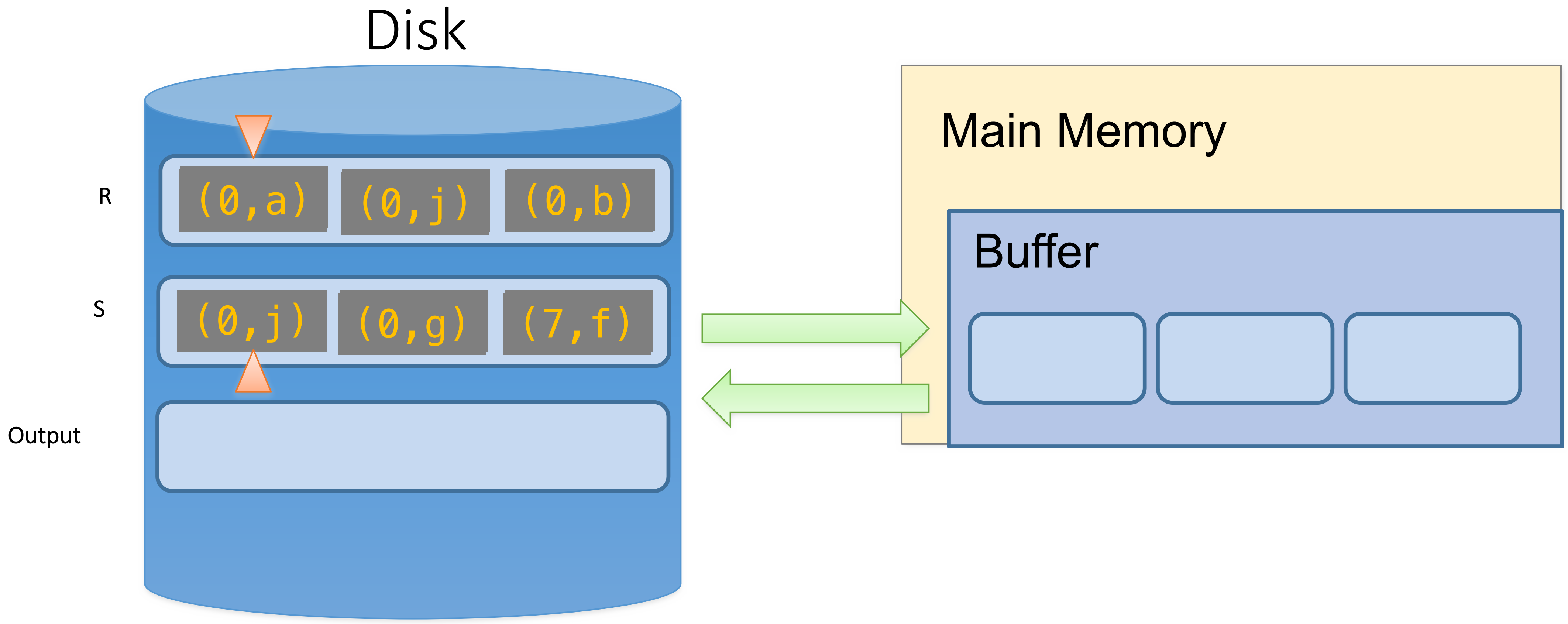
- 2. Done!



What happens with duplicate join keys?

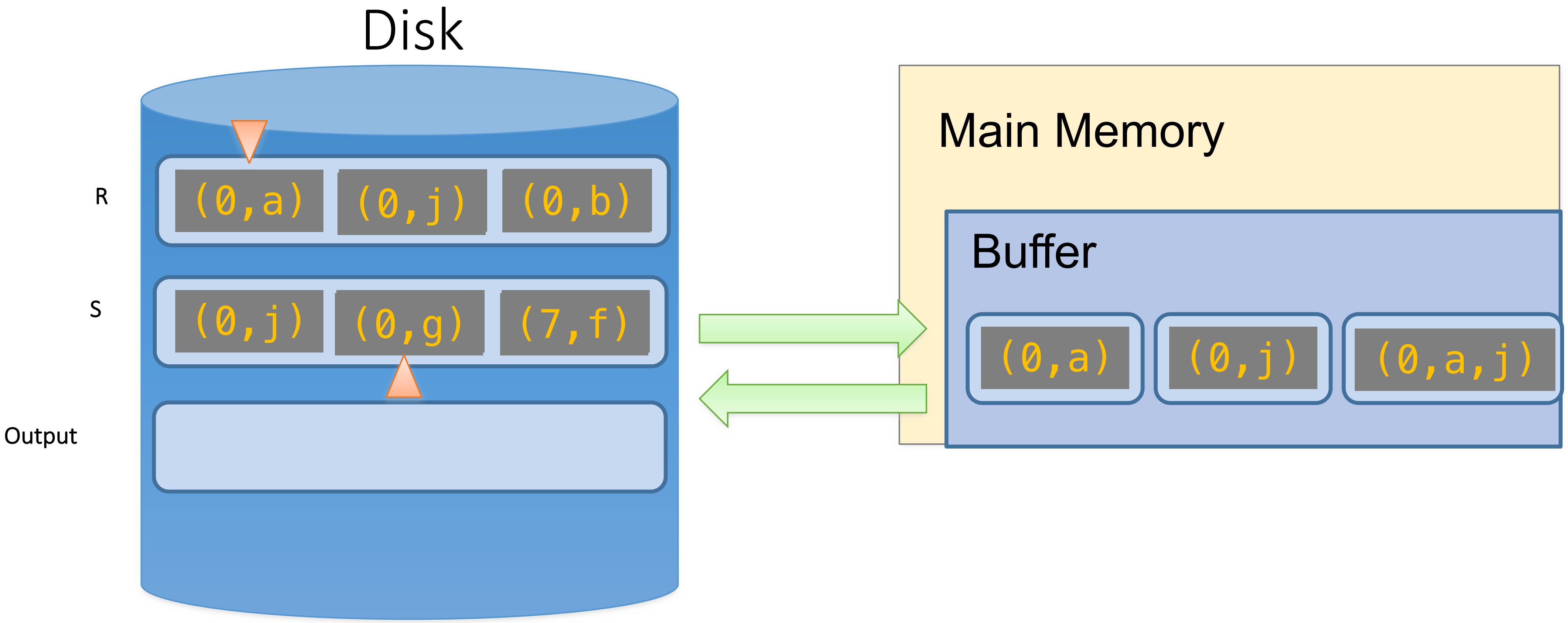
Multiple tuples with Same Join Key: “Backup”

- 1. Start with sorted relations, and begin scan / merge…



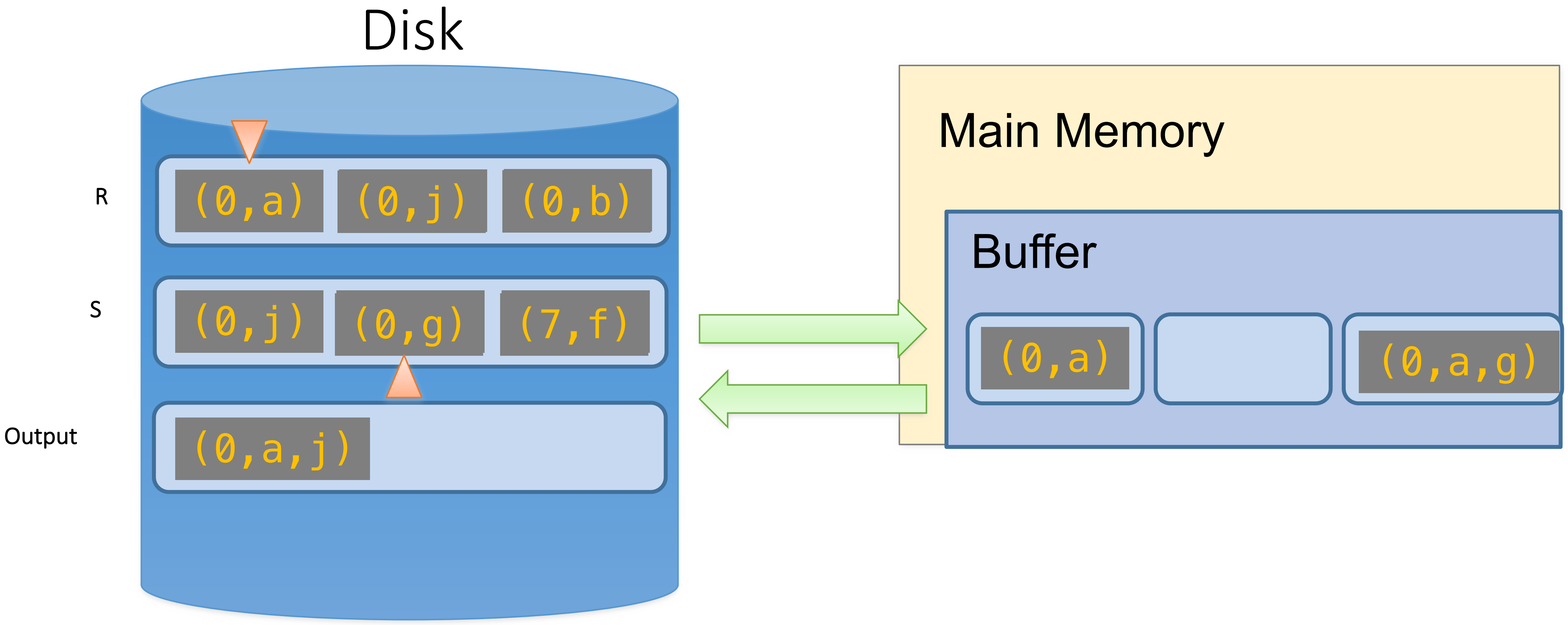
Multiple tuples with Same Join Key: “Backup”

- 1. Start with sorted relations, and begin scan / merge…



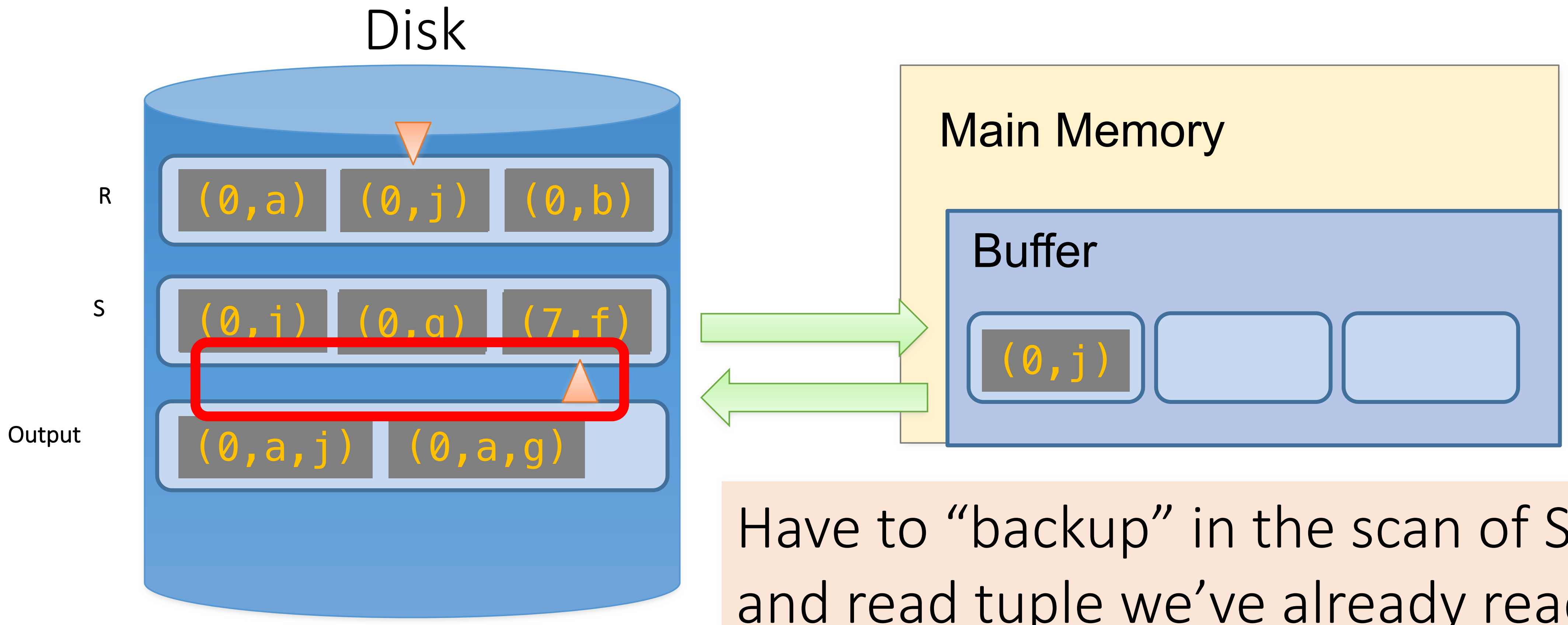
Multiple tuples with Same Join Key: “Backup”

- 1. Start with sorted relations, and begin scan / merge…



Multiple tuples with Same Join Key: "Backup"

- 1. Start with sorted relations, and begin scan / merge…



Have to "backup" in the scan of S and read tuple we've already read!

Backup

- At best, no backup → scan takes **P(R) + P(S)** reads
  - For ex: if no duplicate values in join attribute

- At worst (e.g. full backup each time), scan could take **P(R) * P(S)** reads!
  - For ex: if all duplicate values in join attribute, i.e. all tuples in R and S have the same value for the join attribute
  - Roughly: For each page of R, we'll have to back up and read each page of S…

- Often not that bad however, plus we can:
  - Leave more data in buffer (for larger buffers)
  - Can "zig-zag" (see animation)

SMJ: Total cost

- Cost of SMJ is cost of sorting R and S…

- Plus the cost of scanning: ~P(R)+P(S)
  - Because of backup: in worst case P(R)*P(S); but this would be very unlikely

- Plus the cost of writing out: ~P(R)+P(S) but in worst case T(R)*T(S)

External merge: slides p 26
External merge sort: slides p 43

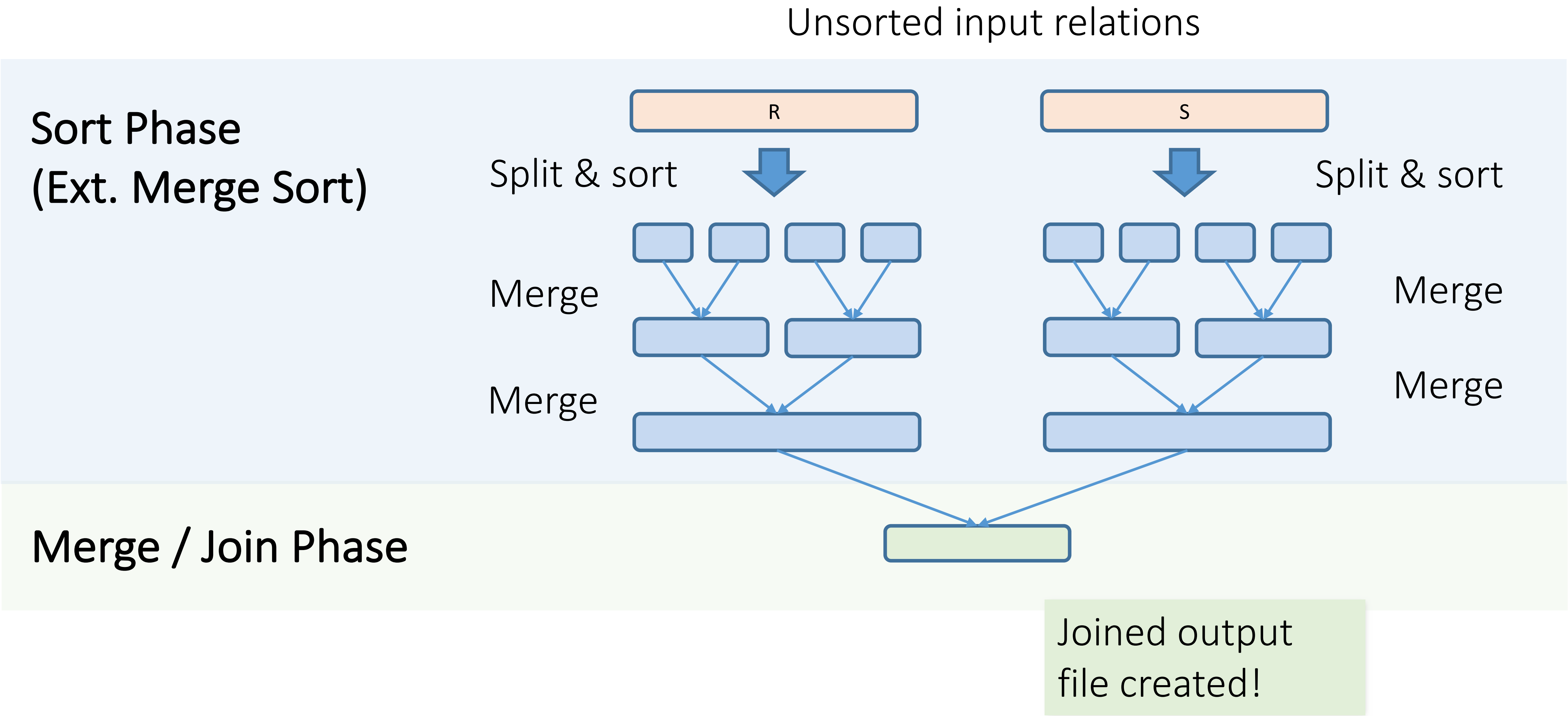
~ Sort(P(R)) + Sort(P(S)) + P(R) + P(S) + OUT

Recall: Sort(N) $\approx 2N\left(\left\lceil \log_{M-1} \frac{\boldsymbol{N}}{\boldsymbol{2M}} \right\rceil + 1\right)$

Note: this is using repacking, where we estimate that we can create initial runs of length ~2M

SMJ Illustrated

Given ***M*** buffer pages



SMJ vs. BNLJ: Comparison

- If we have M=100 buffer pages, P(R) = 1000 pages and P(S) = 500 pages:
- Cost for SMJ:
  - Sort:
  - Merge:
  - Sum:
- What is BNLJ?

SMJ vs. BNLJ: Comparison

- If we have M=100 buffer pages, P(R) = 1000 pages and P(S) = 500 pages:
- Cost for SMJ:
  - Sort: Sort both in two passes: 2 * 2 * 1000 + 2 * 2 * 500 = 6,000 IOs
  - Merge: Merge phase 1000 + 500 = 1,500 IOs
  - Sum: 7,500 IOs + OUT
- What is BNLJ?
  - $500 + 1000 * \left\lceil \frac{500}{98} \right\rceil$ = 5,500 IOs + OUT
- But, if we have M=35 buffer pages?
  - Sort Merge has same behavior (still 2 passes)
  - BNLJ? 15,500 IOs + OUT!

SMJ is ~ linear vs. BNLJ is quadratic…
But it’s all about the memory.

Takeaway points from SMJ

- If input already sorted on join key, skip the sorts.
  - SMJ is basically linear.
  - Nasty but unlikely case: Many duplicate join keys.

- SMJ needs to sort <u>both</u> relations
  - If max { P(R), P(S) } $< M^2$ then cost is 3(P(R)+P(S)) + OUT

L21: The Relational MOdel

CS3200 Database design (sp18 s2)

https://course.ccs.neu.edu/cs3200sp18s2/

4/2/2018

Our next focus

- The Relational Model
- Relational Algebra
- Relational Algebra Pt. II [Optional: may skip]

1. The Relational Model & Relational Algebra

What you will learn about in this section

- The Relational Model
- Relational Algebra: Basic Operators
- Execution

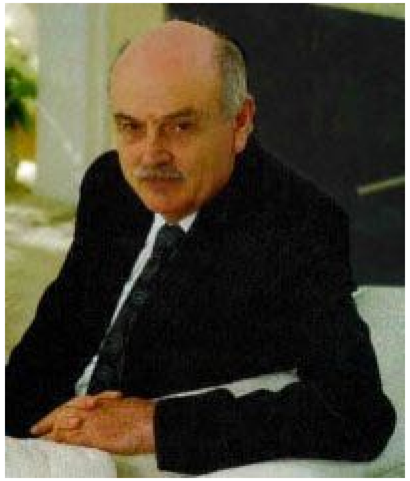
Motivation

The Relational model is **precise**, **implementable**, and we can operate on it (query/update, etc.)

Database maps internally into this *procedural language.*

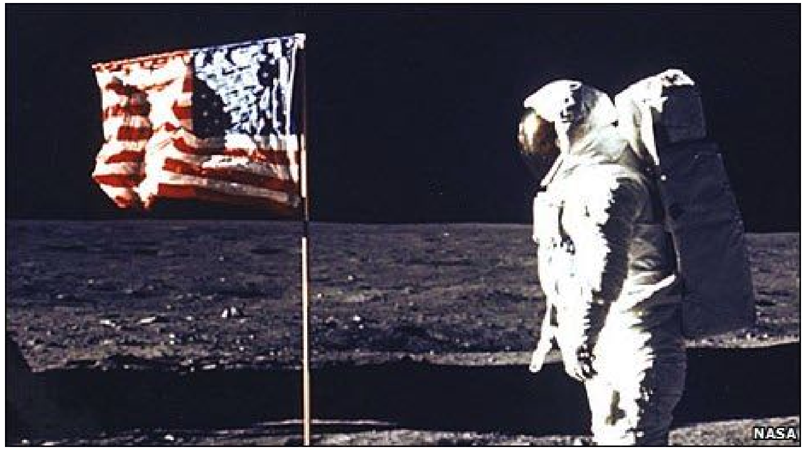
A Little History

- Relational model due to Edgar "Ted" Codd, a mathematician at IBM in 1970
  - A Relational Model of Data for Large Shared Data Banks". Communications of the ACM 13 (6): 377–387



Won Turing award 1981

- IBM didn't want to use relational model (take money from IMS)
  - Apparently used in the moon landing…



The Relational Model: Schemata

- Relational Schema:

```
Students(sid: string, name: string, gpa: float)
```

Relation name

String, float, int, etc. are the **domains** of the attributes

Attributes

The Relational Model: Data

An **attribute** (or **column**) is a typed data entry present in each tuple in the relation

Student

sid	name	gpa
001	Bob	3.2
002	Joe	2.8
003	Mary	3.8
004	Alice	3.5

The number of attributes is the **arity** of the relation

The Relational Model: Data

Student

sid	name	gpa
001	Bob	3.2
002	Joe	2.8
003	Mary	3.8
004	Alice	3.5

The number of tuples is the **cardinality** of the relation

A **tuple** or **row** (or *record*) is a single entry in the table having the attributes specified by the schema

The Relational Model: Data

Student

sid	name	gpa
001	Bob	3.2
002	Joe	2.8
003	Mary	3.8
004	Alice	3.5

Recall: In practice DBMSs relax the set requirement, and use multisets (or bags).

A **relational instance** is a ***set*** of tuples all conforming to the same *schema*

To Reiterate

- A <u>relational schema</u> describes the data that is contained in a <u>relational instance</u>

Let $R(f_1:Dom_1,\ldots,f_m:Dom_m)$ be a ***<u>relational schema</u>*** then, an ***<u>instance</u>*** of R is a subset of $Dom_1 \times Dom_2 \times \ldots \times Dom_n$

In this way, a ***<u>relational schema</u>*** R is a **total function from attribute *names* to types**

One More Time

- A relational schema describes the data that is contained in a relational instance

A relation R of arity *t* is a function:
$R : Dom_1 \times \ldots \times Dom_t \rightarrow \{0,1\}$

I.e. returns whether or not a tuple of matching types is a member of it

Then, the schema is simply the *signature* of the function

Note here that order matters, attribute name doesn't…
We'll (mostly) work with the other model (last slide) in which **attribute name matters, order doesn't!**

A relational database

- A <u>relational database schema</u> is a set of relational schemata, one for each relation

- A <u>relational database instance</u> is a set of relational instances, one for each relation

<u>Two conventions</u>:
1. We call relational database instances as simply ***databases***
2. We assume all instances are valid, i.e., satisfy the <u>*domain constraints*</u>

A Course Management System (CMS)

- Relation DB Schema
  - Students(sid: string, name: string, gpa: float)
  - Courses(cid: string, cname: string, credits: int)
  - Enrolled(sid: string, cid: string, grade: string)

Note that the schemas impose effective domain / type constraints, i.e. Gpa can't be "Apple"

Relation Instances

Sid	Name	Gpa
101	Bob	3.2
123	Mary	3.8

Students

cid	cname	credits
564	564-2	4
308	417	2

Courses

sid	cid	Grade
123	564	A

Enrolled

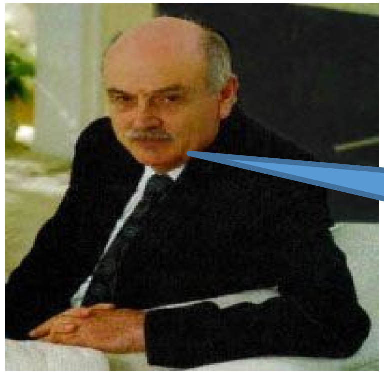
2nd Part of the Model: Querying

```
SELECT S.name
FROM Students S
WHERE S.gpa > 3.5;
```

We don't tell the system *how* or *where* to get the data- **just what we want, i.e., Querying is *declarative***

"Find names of all students with GPA > 3.5"

To make this happen, we need to translate the *declarative* query into a series of operators… we'll see this next!



Actually, I showed how to do this translation for a much richer language!

Virtues of the model

- Physical independence (logical too), Declarative

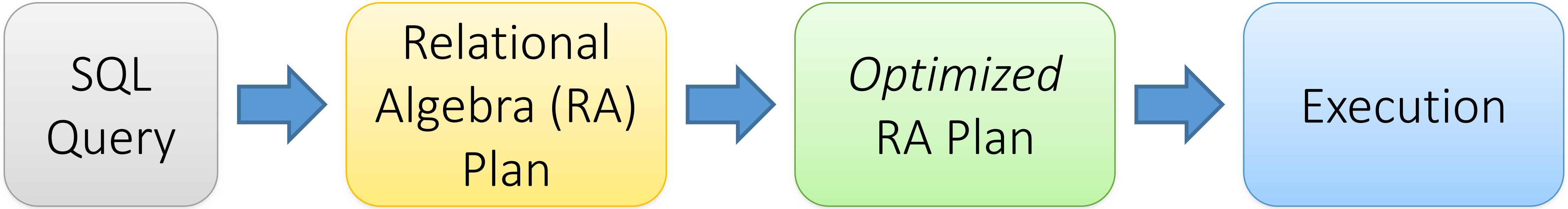
- Simple, elegant clean: Everything is a relation

- Why did it take multiple years?
  - Doubted it could be done efficiently.

2. Relational Algebra

RDBMS Architecture

- How does a SQL engine work ?



Declarative query (from user)

Translate to relational algebra expresson

Find logically equivalent- but more efficient- RA expression

Execute each operator of the optimized plan!