جامعة المستقبل – كلية العلوم
قســم الانـظـمـة الـطبية الـذكـيـة
Intelligent Medical Systems Department

جامـــعـة المـــســتـقـبـل
AL MUSTAQBAL UNIVERSITY

Map-Reduce Techniques

Lecture (9)

Prof. Dr. Mehdi Ebady Manaa

Coping with Node Failure

Suppose the compute node at which a Map worker resides fails. This failure will be detected by the ***Master***, because it periodically pings the Worker processes. All the Map tasks that were assigned to this Worker will have to be redone, even if they had completed. The reason for redoing completed ***Map tasks is that their output destined for the Reduce tasks resides at that compute node, and is now unavailable to the Reduce tasks.*** The Master sets the status of each of these Map tasks to idle and will schedule them on a Worker when one becomes available. The Master must also inform each Reduce task that the location of its input from that Map task has changed.

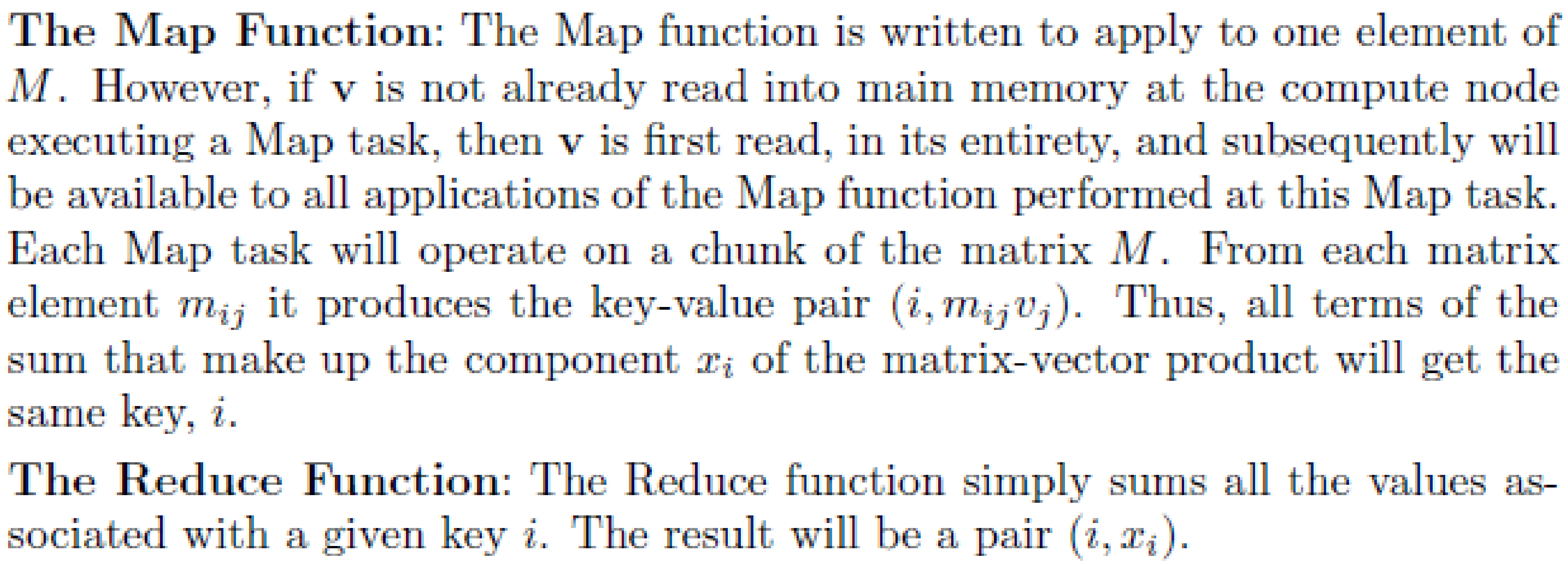
Algorithms using Map-Reduce

Suppose we have an n×n matrix M, whose element in row i and column j will be denoted m_{ij}. Suppose we also have a vector v of length n, whose jth element is v_j. Then the matrix-vector product is the vector x of length n, whose ith element x_i is given by

$$x_i = \sum_{j=1}^{n} m_{ij} v_j$$

The matrix M and the vector **v** each will be stored in a file of the DFS. We assume that the row-column coordinates of each matrix element will be discoverable, either from its position in the file, or because it is stored with explicit coordinates, as a triple (i, j, m_{ij}). We also assume the position of element v_j in the vector **v** will be discoverable in the analogous way.

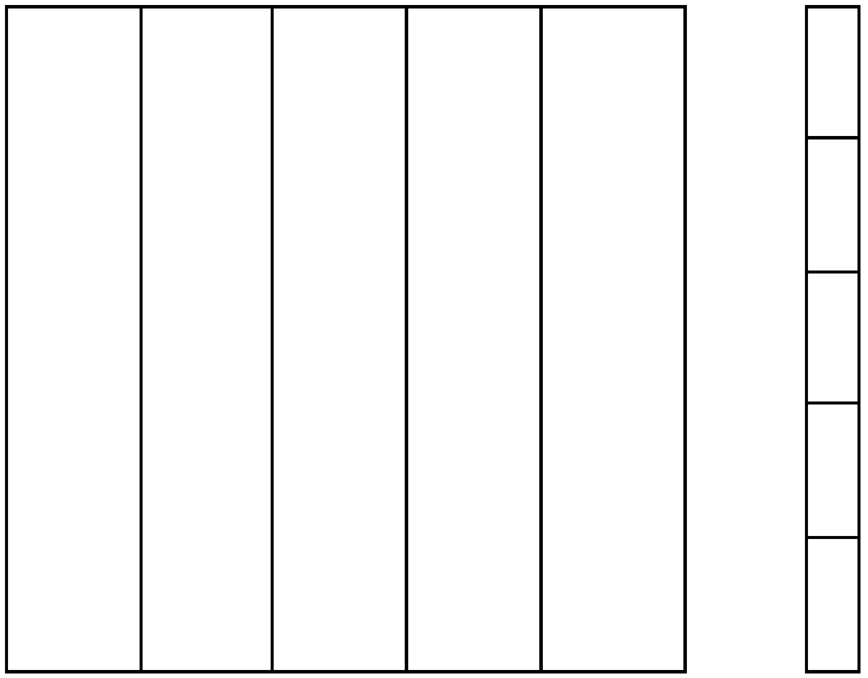
Map-Reduce Algorithm



The Map Function: The Map function is written to apply to one element of M. However, if $\mathbf{v}$ is not already read into main memory at the compute node executing a Map task, then $\mathbf{v}$ is first read, in its entirety, and subsequently will be available to all applications of the Map function performed at this Map task. Each Map task will operate on a chunk of the matrix M. From each matrix element m_{ij} it produces the key-value pair $(i, m_{ij}v_j)$. Thus, all terms of the sum that make up the component x_i of the matrix-vector product will get the same key, i.

The Reduce Function: The Reduce function simply sums all the values associated with a given key i. The result will be a pair (i, x_i).

Vector Problem

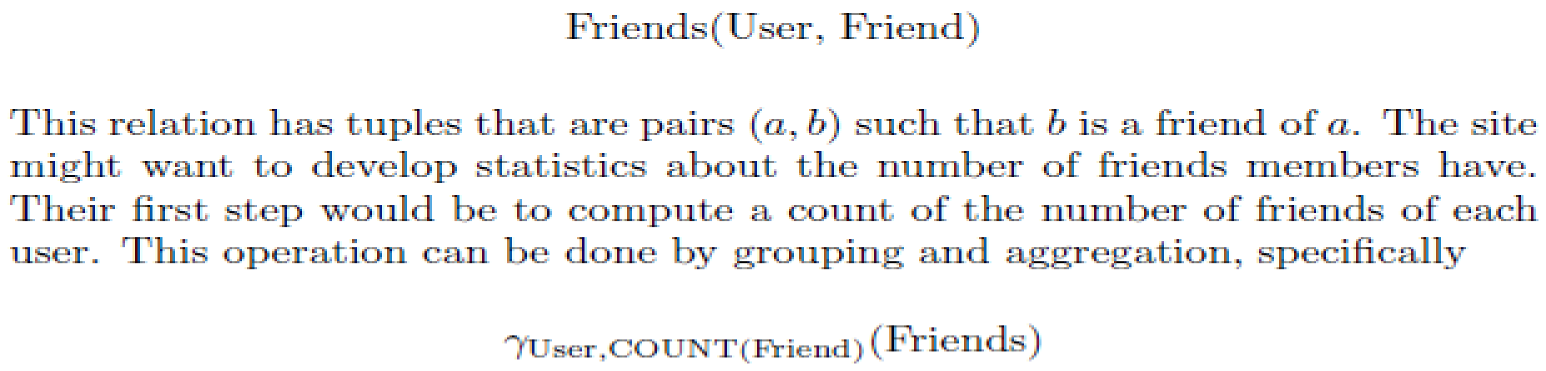


Matrix M Vector $\mathbf{v}$

Relational Algebra Operations

1. **Selection**: Apply a condition C to each tuple in the relation and produce as output only those tuples that satisfy C. The result of this selection is denoted $\sigma_C(R)$.
2. **Projection**: For some subset S of the attributes of the relation, produce from each tuple only the components for the attributes in S. The result of this projection is denoted $\pi_S(R)$.
3. **Union**, Intersection, and Difference: These well-known set operations apply to the sets of tuples in two relations that have the same schema. There are also bag (multiset) versions of the operations in SQL, with somewhat unintuitive definitions, but we shall not go into the bag versions of these operations here.
4. Natural Join: Given two relations, compare each pair of tuples, one from each relation. The natural join of relations R and S is denoted $R \bowtie S$.
5. Grouping and Aggregation:4 Given a relation R, partition its tuples according to their values in one set of attributes G, called the grouping attributes. Then, for each group, aggregate the values in certain other attributes. The normally permitted aggregations are SUM, COUNT, AVG, MIN, and MAX, with the obvious meanings.

Example of Algebra Relation

- From web site relation, find the paths of length 2
- **Please refer to example 2.4**
- **Example 2**
- 

Friends(User, Friend)

This relation has tuples that are pairs (a, b) such that b is a friend of a. The site might want to develop statistics about the number of friends members have. Their first step would be to compute a count of the number of friends of each user. This operation can be done by grouping and aggregation, specifically

$$\gamma_{\text{User},\text{COUNT}(\text{Friend})}(\text{Friends})$$

Selection Algorithm

Selections really do not need the full power of MapReduce. They can be done most conveniently in the map portion alone, although they could also be done in the reduce portion alone. Here is a MapReduce implementation of selection $\sigma_C(R)$.

The Map Function: For each tuple t in R, test if it satisfies C. If so, produce the key-value pair (t, t). That is, both the key and value are t.

The Reduce Function: The Reduce function is the identity. It simply passes each key-value pair to the output.

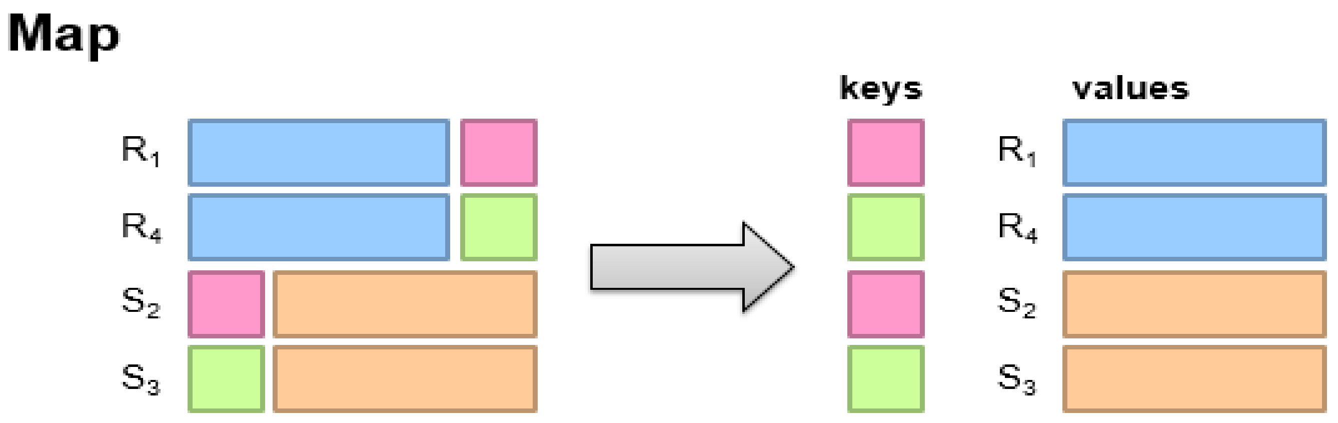
Projection in MapReduce

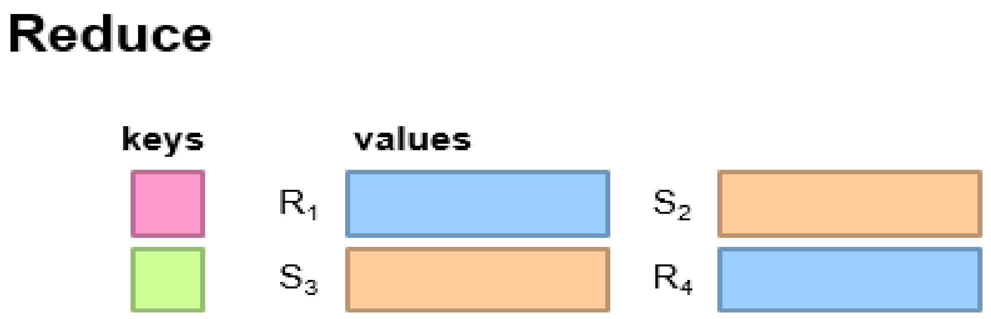
Projection is performed similarly to selection, because projection may cause the same tuple to appear several times, the Reduce function must eliminate duplicates. We may compute $\pi_S(R)$ as follows.

The Map Function: For each tuple t in R, construct a tuple t' by eliminating from t those components whose attributes are not in S. Output the key-value pair (t', t').

The Reduce Function: For each key t' produced by any of the Map tasks, there will be one or more key-value pairs (t', t'). The Reduce function turns $(t', [t', t', \ldots, t'])$ into (t', t'), so it produces exactly one pair (t', t') for this key t'.

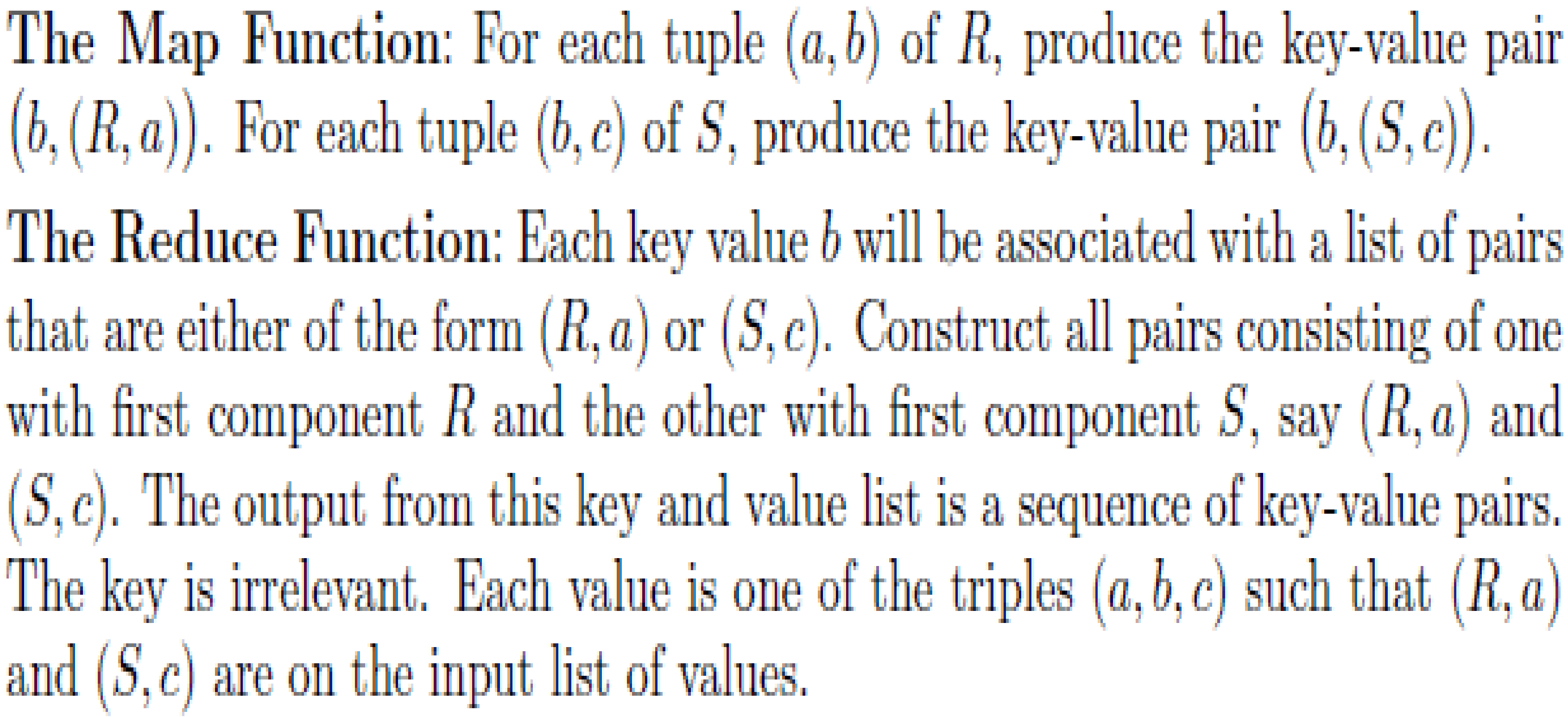
Natural Join in Map-Reduce





Note: no guarantee if R is going to come first or S

Natural Join of Map-Reduce



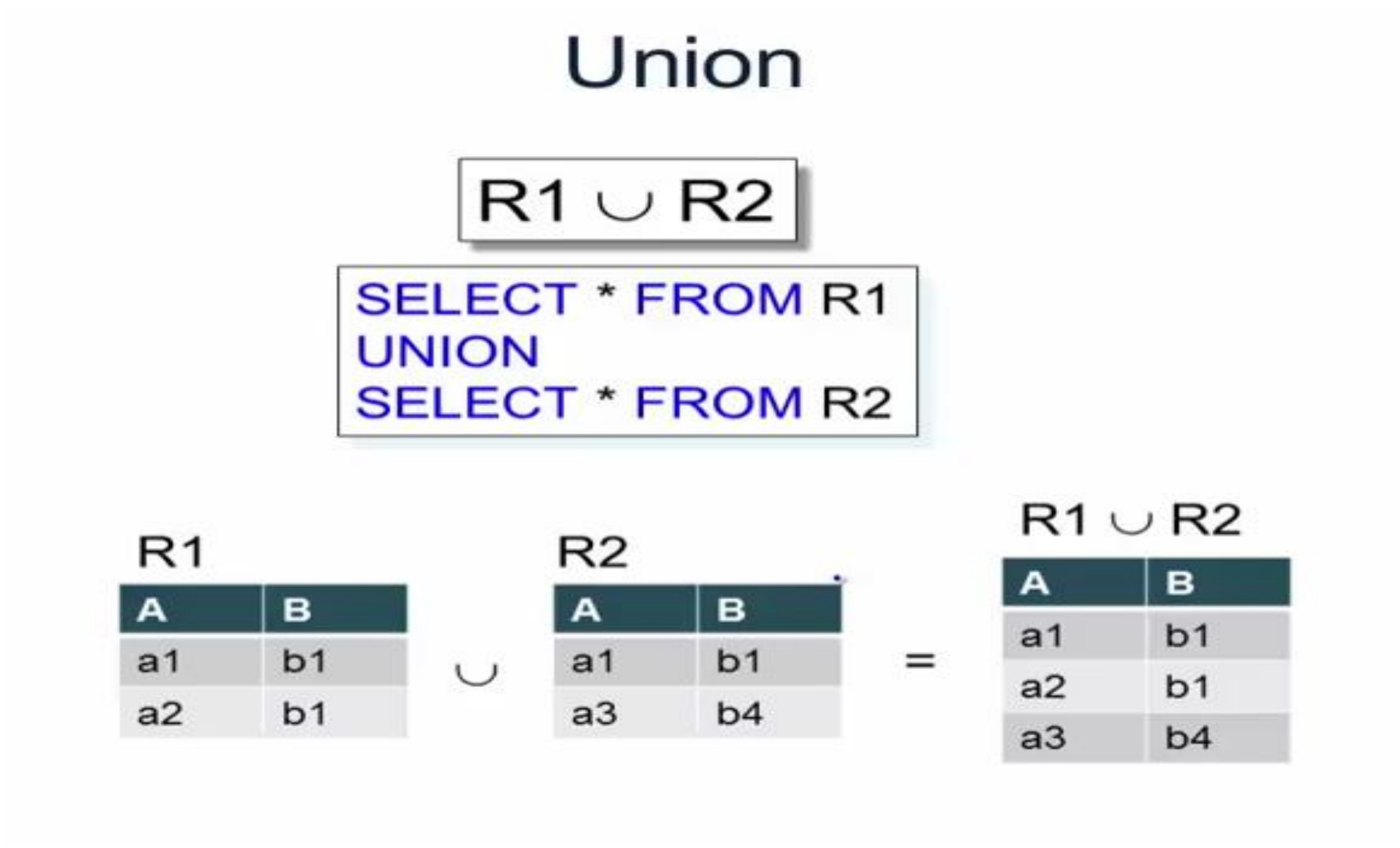
The Map Function: For each tuple (a, b) of R, produce the key-value pair $(b, (R, a))$. For each tuple (b, c) of S, produce the key-value pair $(b, (S, c))$.

The Reduce Function: Each key value b will be associated with a list of pairs that are either of the form (R, a) or (S, c). Construct all pairs consisting of one with first component R and the other with first component S, say (R, a) and (S, c). The output from this key and value list is a sequence of key-value pairs. The key is irrelevant. Each value is one of the triples (a, b, c) such that (R, a) and (S, c) are on the input list of values.

Union, Intersection and Difference by Map-reduce

Please provide your answer

Union



Union

$R1 \cup R2$

```
SELECT * FROM R1
UNION
SELECT * FROM R2
```

R1

A	B
a1	b1
a2	b1

$\cup$

R2

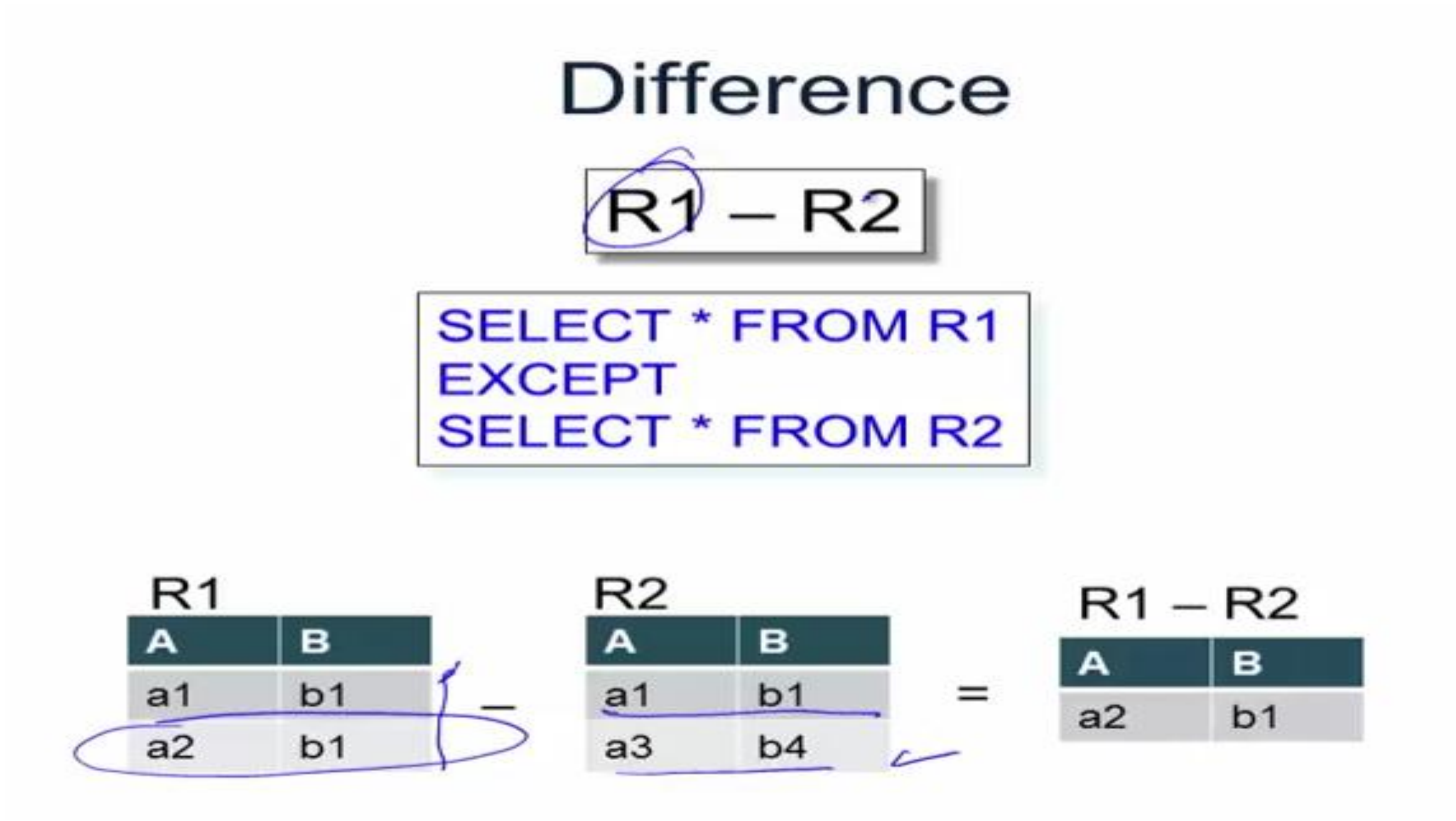
A	B
a1	b1
a3	b4

=

$R1 \cup R2$

A	B
a1	b1
a2	b1
a3	b4

Difference



Difference

$\boxed{R1 - R2}$

```
SELECT * FROM R1
EXCEPT
SELECT * FROM R2
```

R1

A	B
a1	b1
a2	b1

–

R2

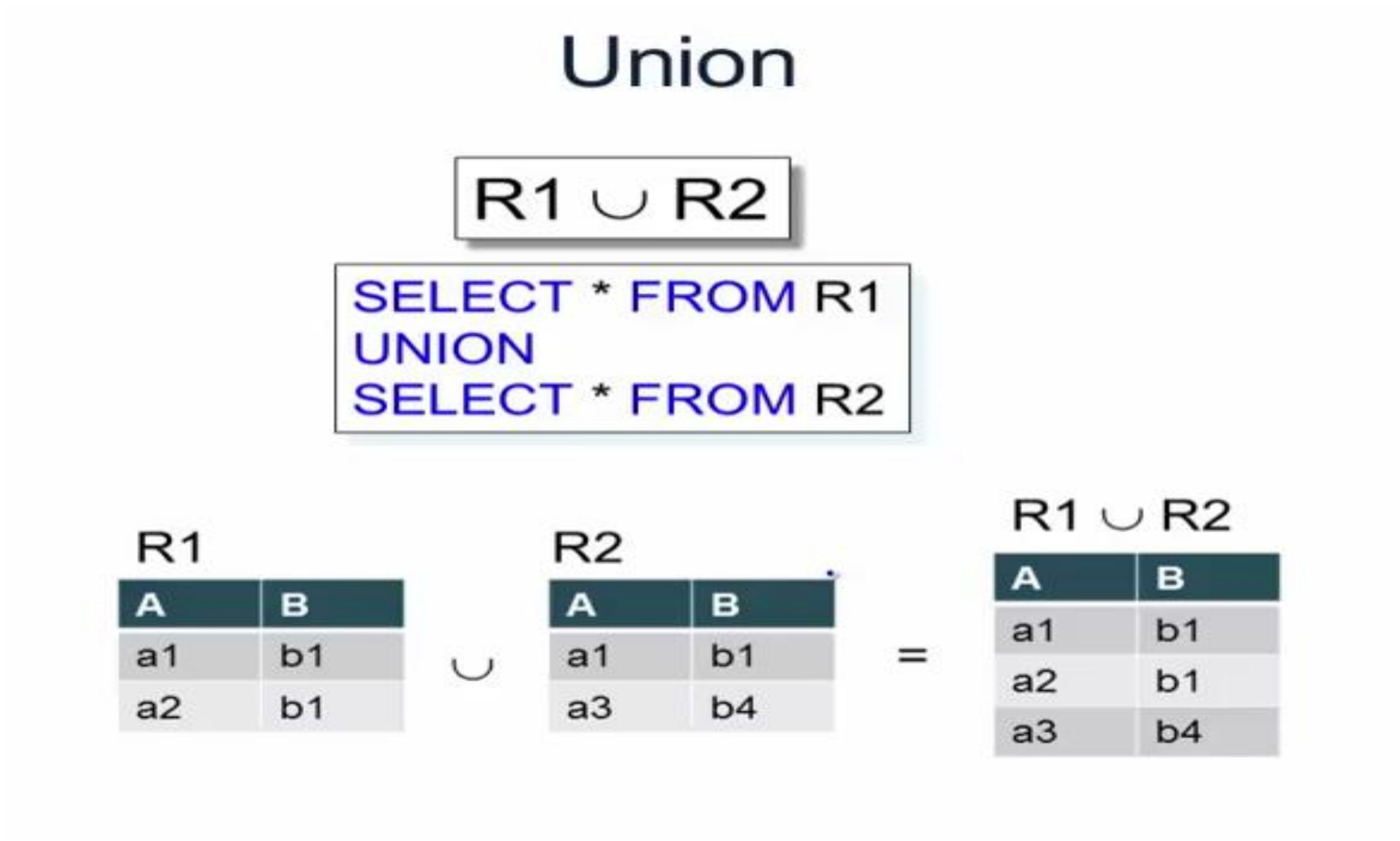
A	B
a1	b1
a3	b4

=

R1 – R2

A	B
a2	b1

Union



Union

$R1 \cup R2$

```
SELECT * FROM R1
UNION
SELECT * FROM R2
```

R1

A	B
a1	b1
a2	b1

$\cup$

R2

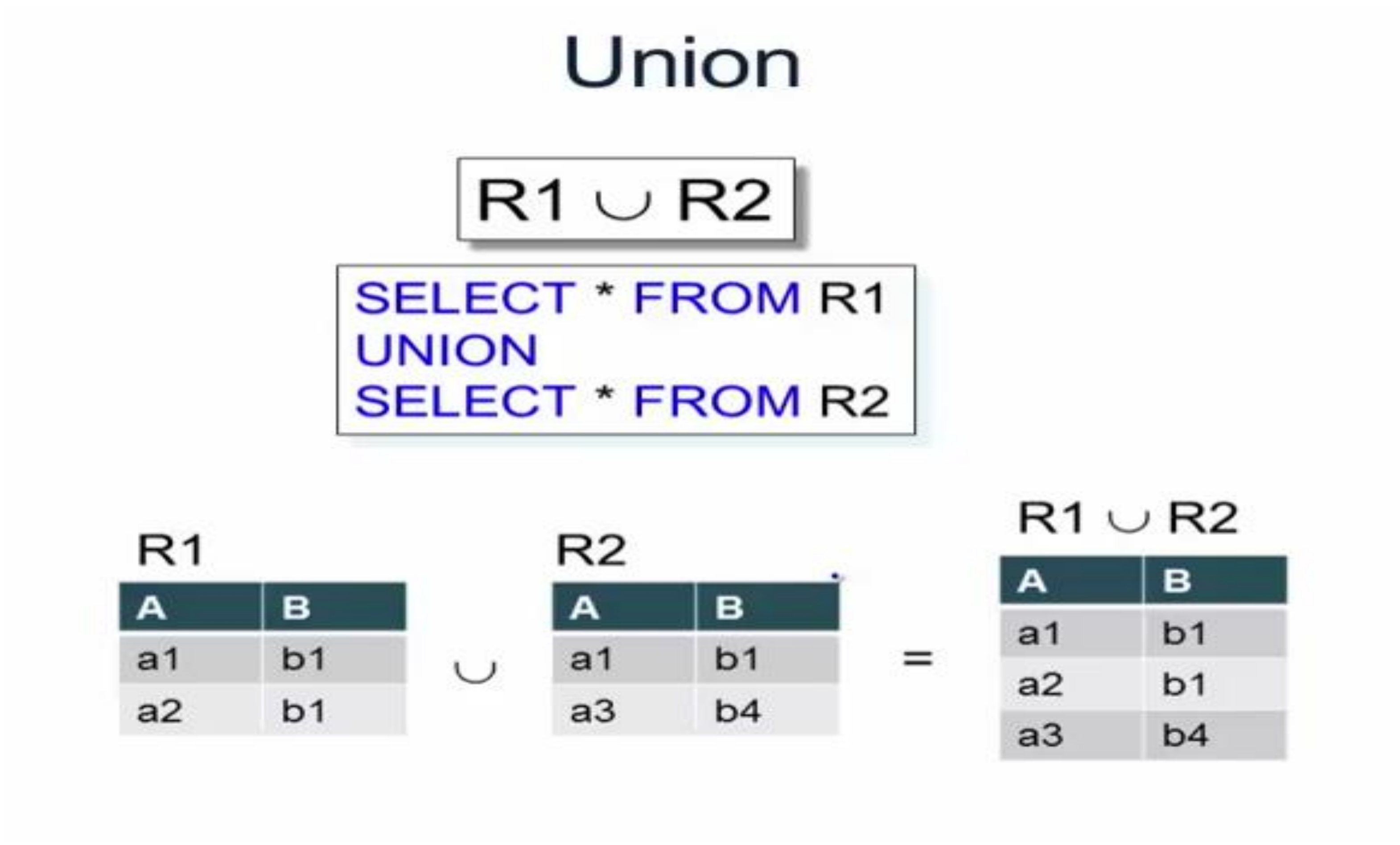
A	B
a1	b1
a3	b4

=

$R1 \cup R2$

A	B
a1	b1
a2	b1
a3	b4

Intersection



Union

$R1 \cup R2$

```
SELECT * FROM R1
UNION
SELECT * FROM R2
```

R1

A	B
a1	b1
a2	b1

$\cup$

R2

A	B
a1	b1
a3	b4

=

$R1 \cup R2$

A	B
a1	b1
a2	b1
a3	b4

THANK YOU