Query Processing for Non-traditional Applications

CS848 Spring 2013

Cheriton School of CS

The ACME PAYROLL System: A Case Study

ACME Corporation wishes to develop the PAYROLL system to more carefully manage information about its personnel, their salaries, etc.

The ACME PAYROLL System: A Case Study

ACME Corporation wishes to develop the PAYROLL system to more carefully manage information about its personnel, their salaries, etc.

Infrastructure for PAYROLL :

- A computer with mass storage.
- Department APS (short for *applications*).
- Department DBA (short for *database admininstration*).

The ACME PAYROLL System: A Case Study

ACME Corporation wishes to develop the PAYROLL system to more carefully manage information about its personnel, their salaries, etc.

Infrastructure for PAYROLL :

- A computer with mass storage.
- Department APS (short for *applications*).
- Department DBA (short for *database admininstration*).

APS is responsible for PAYROLL.
DBA is responsible for the computing resources.

The ACME PAYROLL System: A Case Study

ACME Corporation wishes to develop the PAYROLL system to more carefully manage information about its personnel, their salaries, etc.

Infrastructure for PAYROLL :

- A computer with mass storage.
- Department APS (short for *applications*).
- Department DBA (short for *database admininstration*).

APS is responsible for PAYROLL.
DBA is responsible for the computing resources.

ACME will use a *relational DBMS* to implement PAYROLL.

The ACME PAYROLL System: A Case Study

ACME Corporation wishes to develop the PAYROLL system to more carefully manage information about its personnel, their salaries, etc.

Infrastructure for PAYROLL :

- A computer with mass storage.
- Department APS (short for *applications*).
- Department DBA (short for *database admininstration*).

APS is responsible for PAYROLL.
DBA is responsible for the computing resources.

ACME will use a *relational DBMS* to implement PAYROLL.

The ACME PAYROLL System: A Case Study

ACME Corporation wishes to develop the PAYROLL system to more carefully manage information about its personnel, their salaries, etc.

Infrastructure for PAYROLL :

- A computer with mass storage.
- Department APS (short for *applications*).
- Department DBA (short for *database admininstration*).

APS is responsible for PAYROLL.
DBA is responsible for the computing resources.

ACME will use a *relational DBMS* to implement PAYROLL.

ACME PAYROLL System: Information

There is a rough dichotomy of information in database systems into what are commonly termed *data* and *metadata*.

ACME PAYROLL System: Information

There is a rough dichotomy of information in database systems into what are commonly termed *data* and *metadata*.

Example of PAYROLL data important to APS.

1. Mary is an employee.
2. Mary's employee number is 3412.
3. Mary's salary is 72000.

ACME PAYROLL System: Information

There is a rough dichotomy of information in database systems into what are commonly termed *data* and *metadata*.

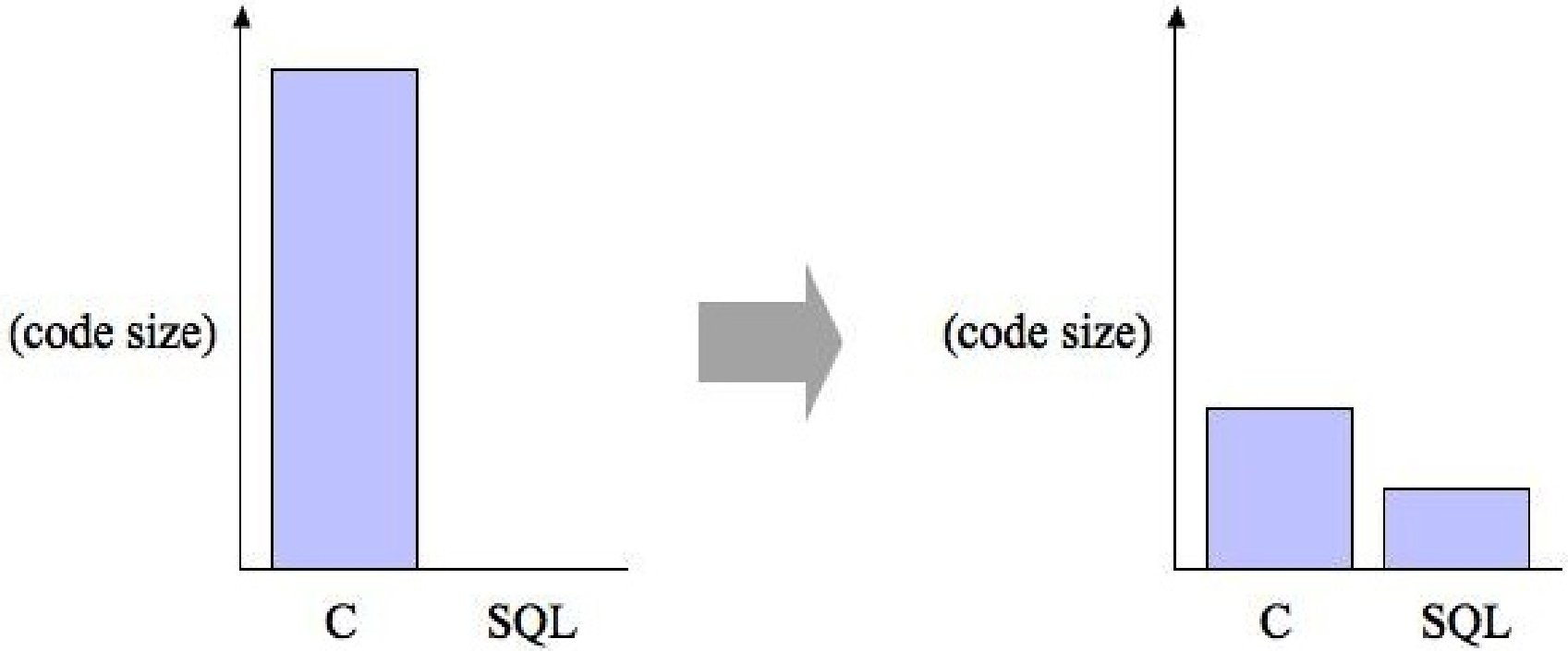
Example of PAYROLL data important to APS.

1. Mary is an employee.
2. Mary's employee number is 3412.
3. Mary's salary is 72000.

Example of PAYROLL metadata specified by APS.

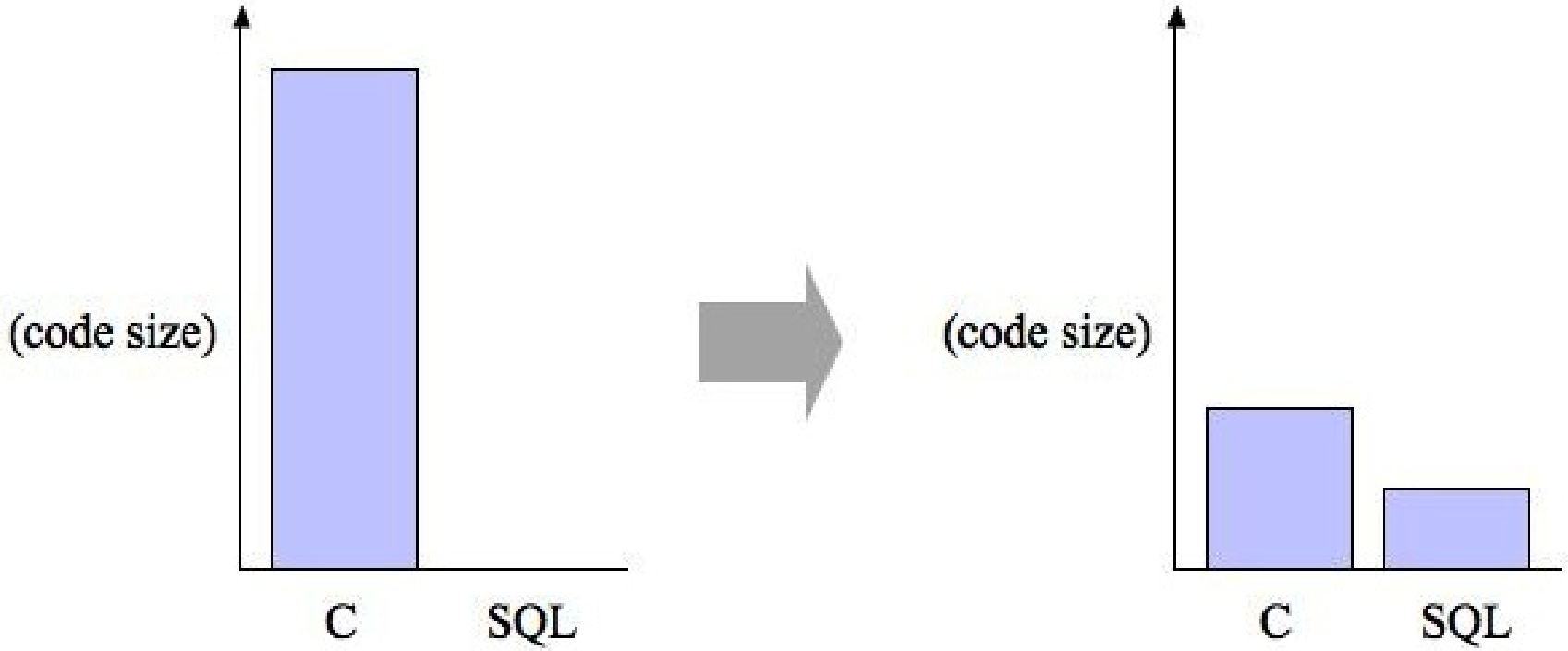
4. There is a kind of entity called an `employee`.
5. There are attributes called `enumber`, `name` and `salary`.
6. Each employee entity has attributes `enumber`, `name` and `salary`.
7. Employees are identified by their `enumber`.

ACME PAYROLL System: Motivation



Reduces time needed by APS to create PAYROLL.

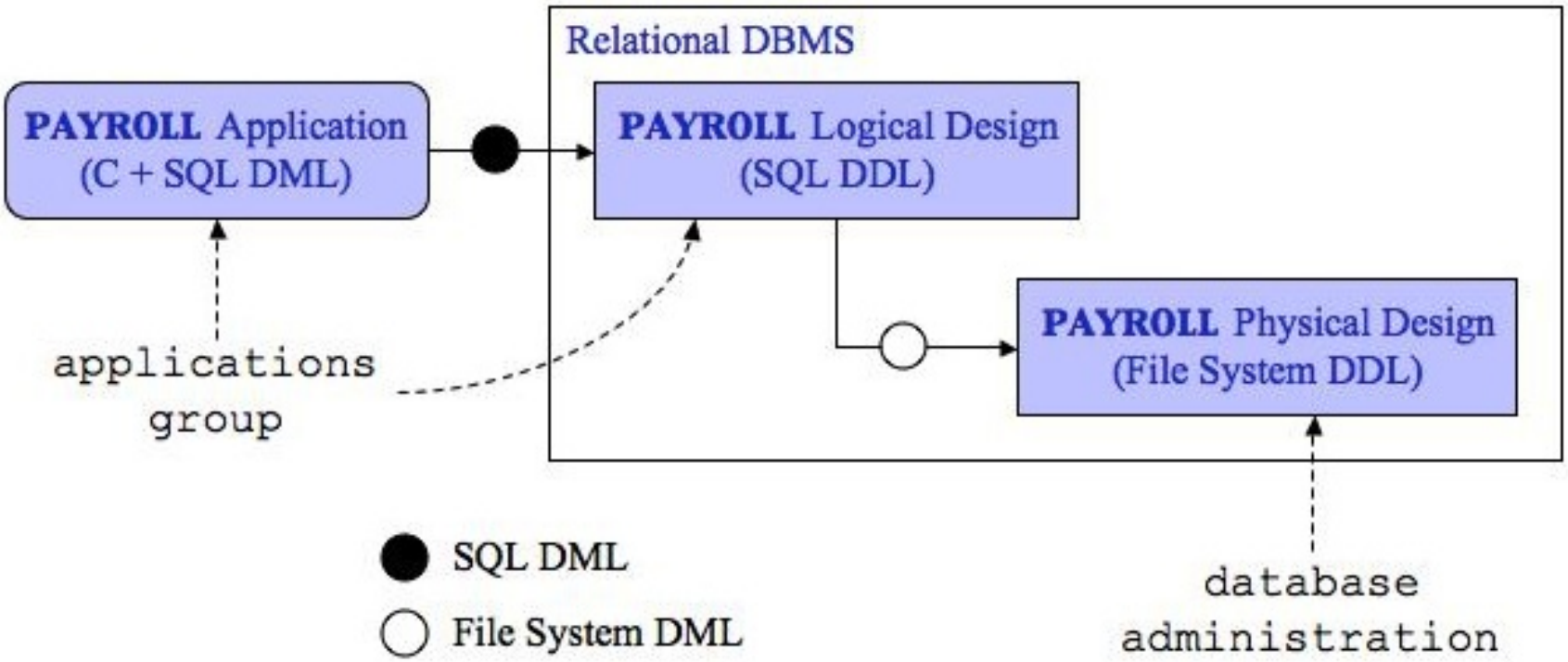
ACME PAYROLL System: Motivation



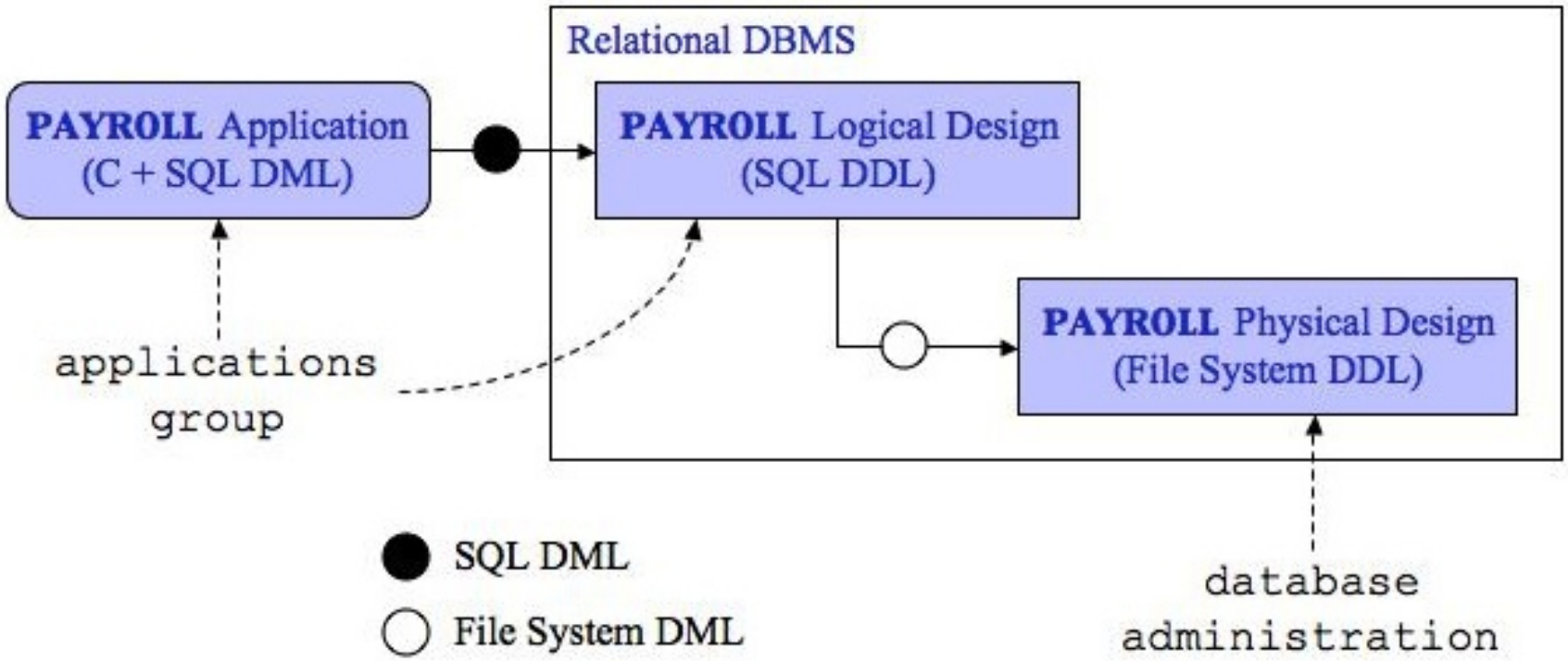
Reduces time needed by APS to create PAYROLL.

Reduces time needed for new APS personnel to learn about `payroll`.

ACME PAYROLL System: Architecture

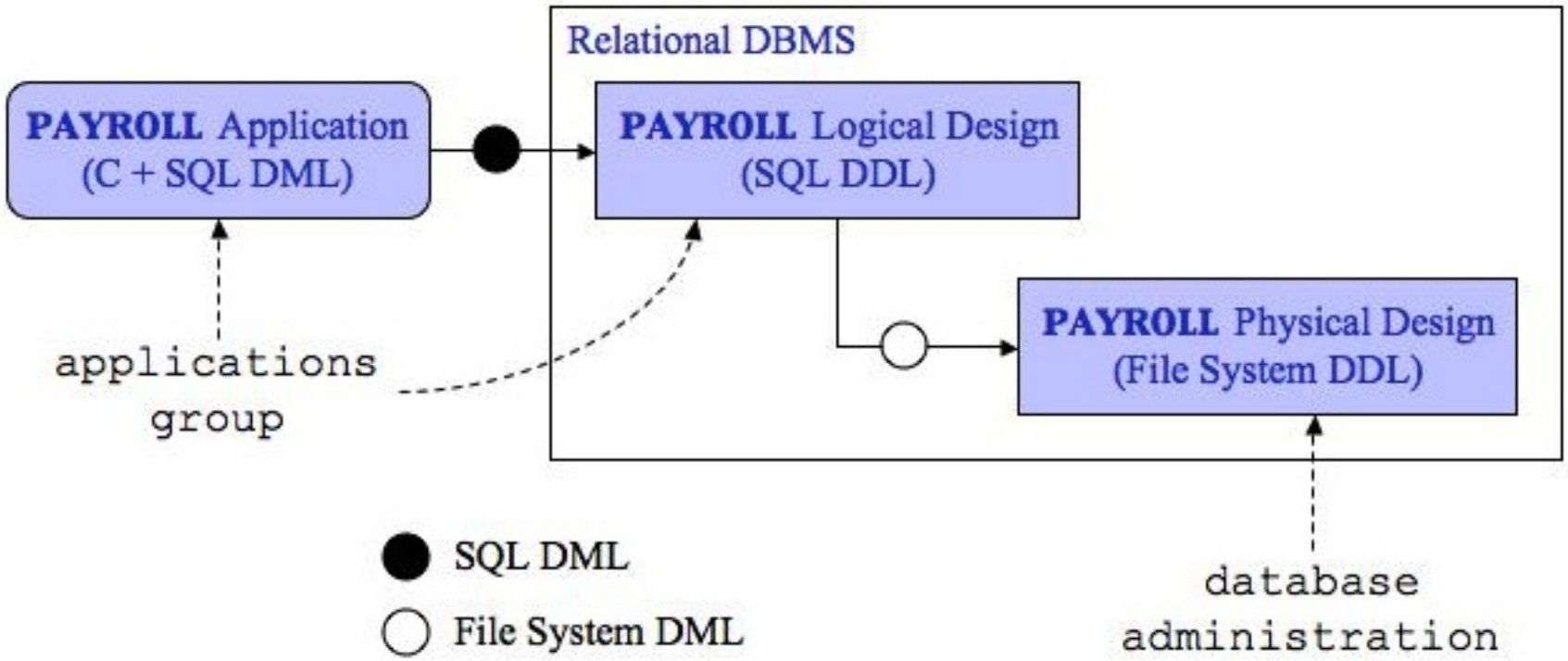


ACME PAYROLL System: Architecture



Metadata is expressed in a *data definition language* (DDL).

ACME PAYROLL System: Architecture



Metadata is expressed in a *data definition language* (DDL).

Code/specifications that access and update data is expressed in a *data manipulation language* (DML).

ACME PAYROLL System: Physical Design

A *physical design* for PAYROLL selected by DBA.

8. There is a file of records called `emp-file`.
9. There are record fields `emp-num`, `emp-name` and `emp-salary`.
10. Each `emp-file` record has the fields `emp-num`, `emp-name` and `emp-salary`.
11. File `emp-file` is organized as a B-tree data structure that supports an `emp-lookup` operation, given a value for attribute `enumber`.
12. Records in file `emp-file` correspond one-to-one to `employee` entities.
13. Record fields in file `emp-file` encode the corresponding attribute values for `employee` entities, for example, `emp-num` encodes an `enumber`.

ACME PAYROLL System: Queries and Query Plans

A PAYROLL *user query* specified by APS.

14. Find the `salary` for any `employee` whose `enumber` is given by a parameter *p*.

ACME PAYROLL System: Queries and Query Plans

A PAYROLL *user query* specified by APS.

14. Find the `salary` for any `employee` whose `enumber` is given by a parameter *p*.

A *query plan* selected by a *query compiler*.

15. Invoke operation `emp-lookup(`*p*`)`, just once, on file `emp-file`. If an `emp-file` record is found, then extract and return the value of field `emp-salary`.

ACME PAYROLL System: Queries and Query Plans

A PAYROLL *user query* specified by APS.

14. Find the `salary` for any `employee` whose `enumber` is given by a parameter *p*.

A *query plan* selected by a *query compiler*.

15. Invoke operation `emp-lookup(`*p*`)`, just once, on file `emp-file`. If an `emp-file` record is found, then extract and return the value of field `emp-salary`.

The query is *physically data independent*.

ACME PAYROLL System: Queries and Query Plans

A PAYROLL *user query* specified by APS.

14. Find the `salary` for any `employee` whose `enumber` is given by a parameter *p*.

A *query plan* selected by a *query compiler*.

15. Invoke operation `emp-lookup(`*p*`)`, just once, on file `emp-file`. If an `emp-file` record is found, then extract and return the value of field `emp-salary`.

The query is *physically data independent*.

The plan is *physically data dependent*.

ACME PAYROLL System: Queries and Query Plans

A PAYROLL *user query* specified by APS.

14. Find the `salary` for any `employee` whose `enumber` is given by a parameter *p*.

A *query plan* selected by a *query compiler*.

15. Invoke operation `emp-lookup(`*p*`)`, just once, on file `emp-file`. If an `emp-file` record is found, then extract and return the value of field `emp-salary`.

The query is *physically data independent*.

The plan is *physically data dependent*.

The plan is *executable*.

The ACME LINUX-INFO System: A Case Study

ACME Corporation wishes to develop the LINUX-INFO system to monitor the operating systems deployed in their organization.

The ACME LINUX-INFO System: A Case Study

ACME Corporation wishes to develop the LINUX-INFO system to monitor the operating systems deployed in their organization.

Infrastructure for LINUX-INFO :

- Computer(s) running the Linux OS.
- Department APS (short for *applications*).

The ACME LINUX-INFO System: A Case Study

ACME Corporation wishes to develop the LINUX-INFO system to monitor the operating systems deployed in their organization.

Infrastructure for LINUX-INFO :

- Computer(s) running the Linux OS.
- Department APS (short for *applications*).

APS is responsible for LINUX-INFO.

The ACME LINUX-INFO System: A Case Study

ACME Corporation wishes to develop the LINUX-INFO system to monitor the operating systems deployed in their organization.

Infrastructure for LINUX-INFO :

- Computer(s) running the Linux OS.
- Department APS (short for *applications*).

APS is responsible for LINUX-INFO.

ACME will use a *relational technology* to implement LINUX-INFO.

ACME LINUX-INFO System: Data and Metadata

Example of LINUX-INFO data important to APS.

1. process `gcc` is running
2. `gcc`'s process number is 1234.
3. the user running `gcc` is 145.
4. `gcc` uses file "foo.c"

ACME LINUX-INFO System: Data and Metadata

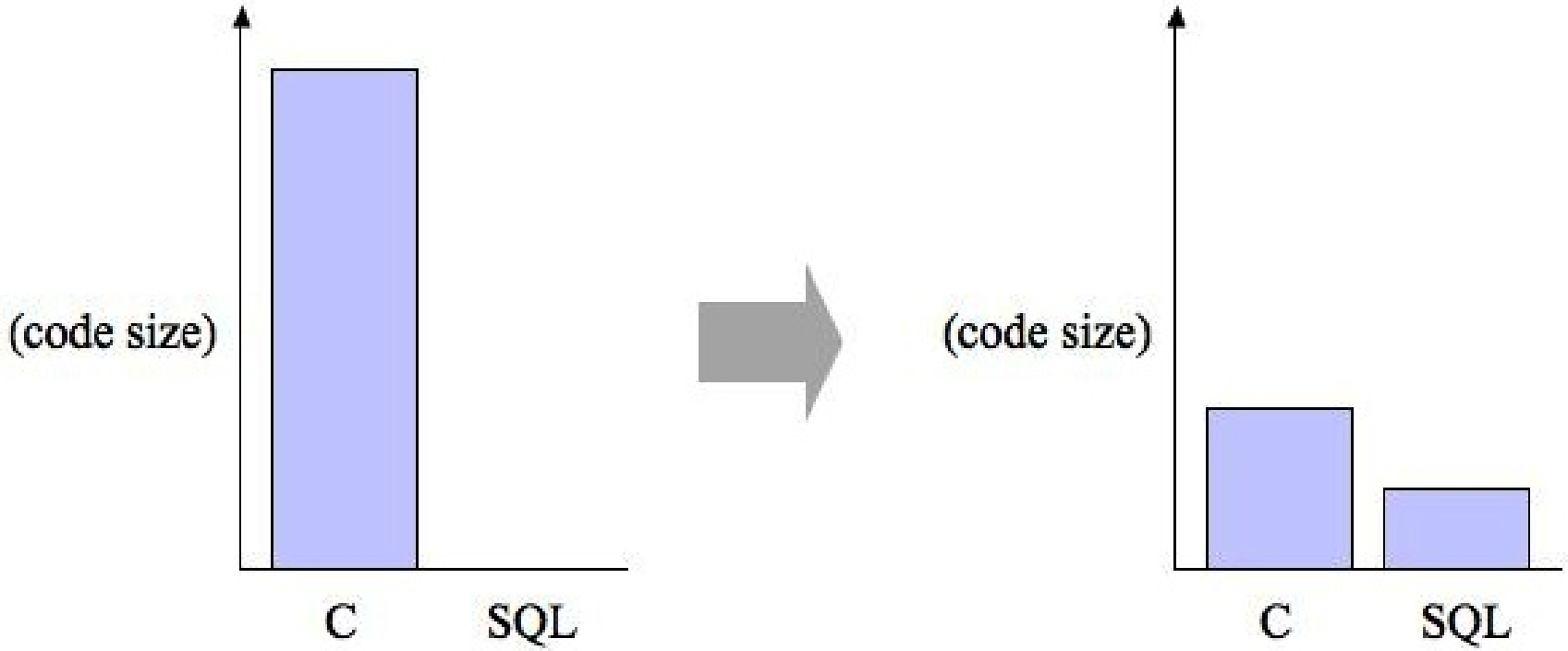
Example of LINUX-INFO data important to APS.

1. process `gcc` is running
2. `gcc`'s process number is 1234.
3. the user running `gcc` is 145.
4. `gcc` uses file "foo.c"

Example of LINUX-INFO metadata specified by APS.

4. There entities called `process` and `file`.
5. There are attributes called `pno`, `pname`, `uname`, and `fname`.
6. Each process entity has attributes `pno`, `pname` and `uname`.
7. Each file entity has attribute `fname`.
8. Processes are identified by their `pno`.
9. Files are identified by their `fname`.
10. There is a relationship `uses` between processes and files.

ACME LINUX-INFO System: Motivation



Reduces time needed by APS to create LINUX-INFO. Reduces time needed for new APS personnel to learn about LINUX.

The LINUX System: Physical Design

A *physical design* for LINUX (selected by Linus Torvalds).

8. There are process records called `task-struct`.
9. Each `task-struct` record has record fields `pid`, `uid`, `comm`, and `file-struct`.
10. All `task-struct`s is organized as a tree data structure.
11. The `task-struct` records correspond one-to-one to `process` entities.
12. Record fields in `task-struct` encode the corresponding attribute values for `process` entities, for example, `pid` encodes an `pno`, etc.
13. Similarly, `fs`s correspond appropriately to (open) `file` entities.
14. `file-struct` field of `task-struct` is an array of `fd`s; an entry in this array indicates that the `process` corresponding to this `task-struct` is using the `file` represented by the `fd` record in the array.

ACME LINUX-INFO System: Queries and Query Plans

A LINUX-INFO *user query* specified by APS.

14. Find the `files` used by `process` invoked by user 145.

ACME LINUX-INFO System: Queries and Query Plans

A LINUX-INFO *user query* specified by APS.

14. Find the `files` used by `process` invoked by user 145.

A *query plan* selected by a *query compiler*.

15. Scan tree of `task-struct`s, for each check if its `uid` attribute is 145 and, if so scan the `file-struct array` in the `task-struct` and print out the names of files described by non-NULL file descriptors (`fd`).

ACME LINUX-INFO System: Queries and Query Plans

A LINUX-INFO *user query* specified by APS.

14. Find the `files` used by `process` invoked by user 145.

A *query plan* selected by a *query compiler*.

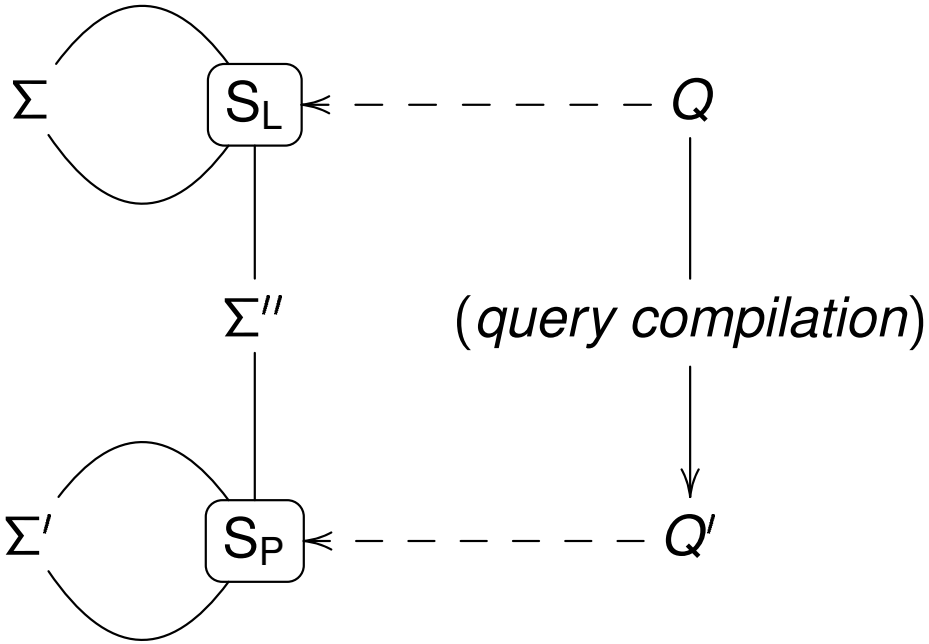
15. Scan tree of `task-struct`s, for each check if its `uid` attribute is 145 and, if so scan the `file-struct array` in the `task-struct` and print out the names of files described by non-NULL file descriptors (`fd`).

Question:

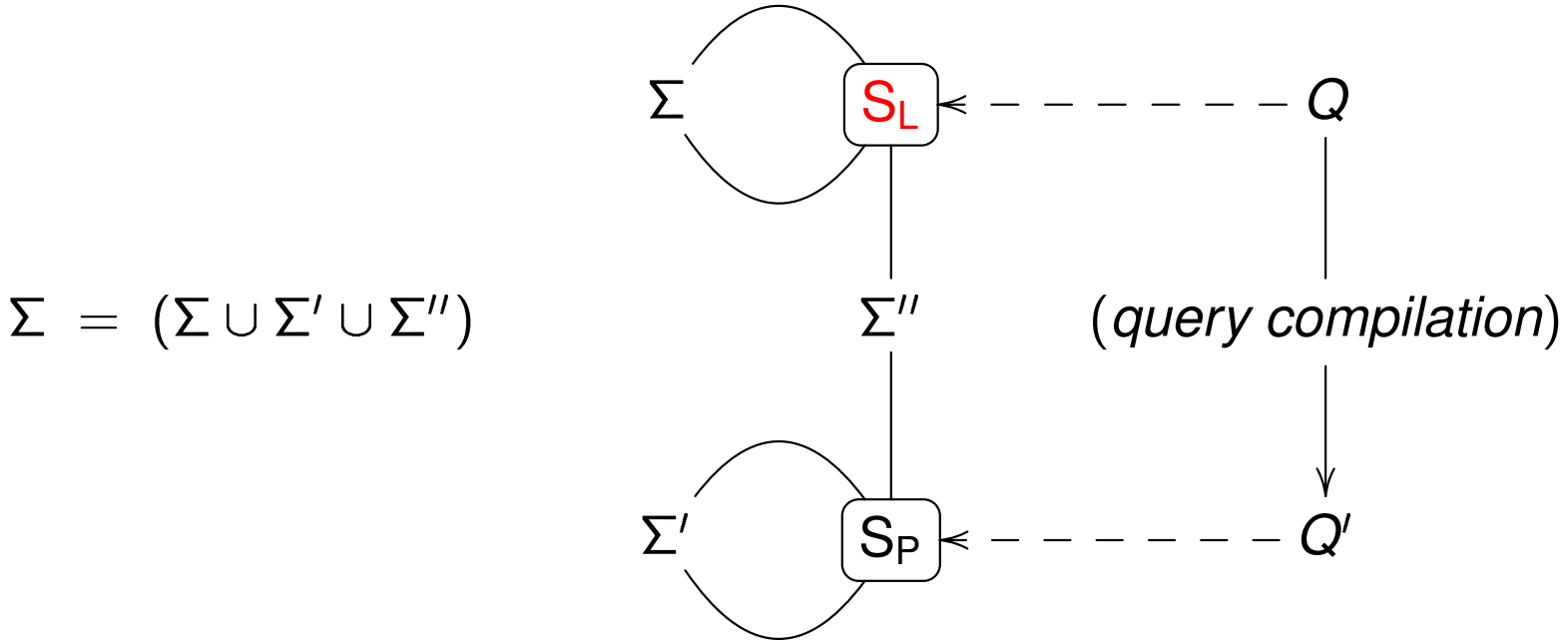
Does the *physical design* allow APS to list all files known to the Linux system?

Physical Design and Query Compilation: Overview

$$\Sigma = (\Sigma \cup \Sigma' \cup \Sigma'')$$

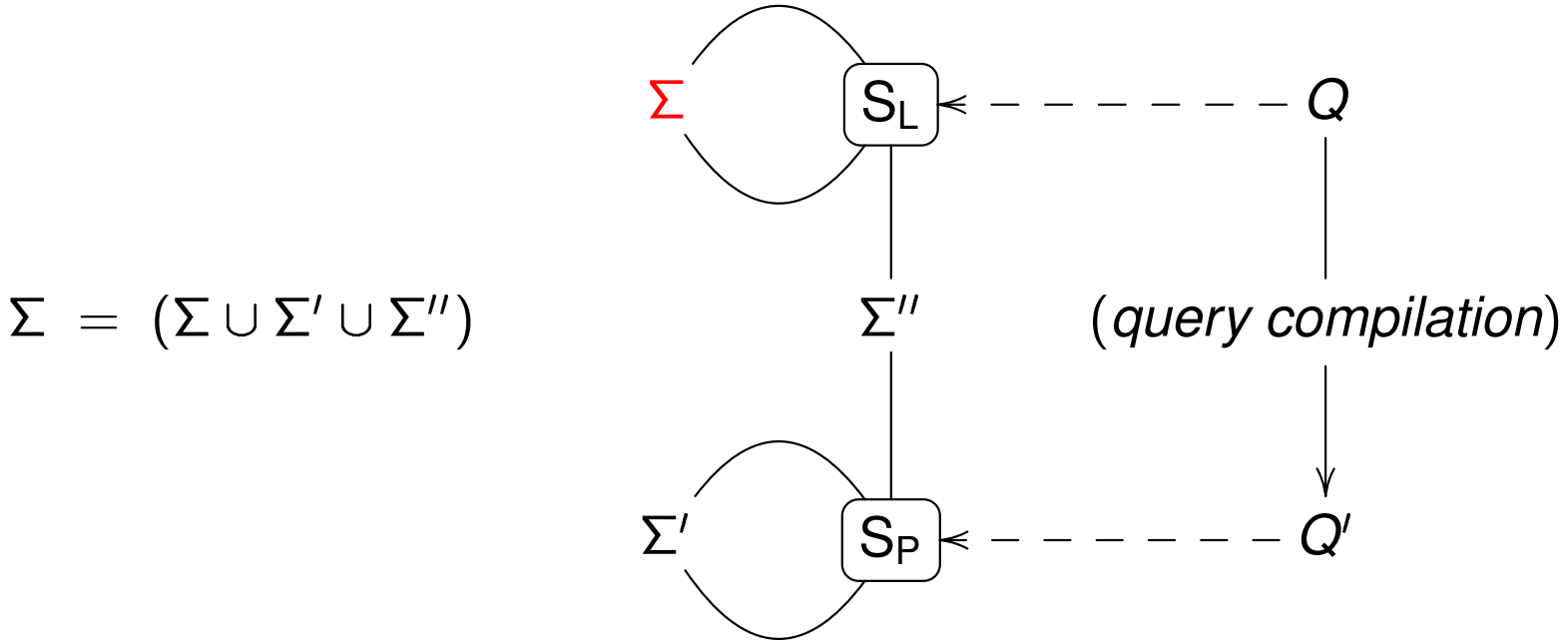


Physical Design and Query Compilation: Overview



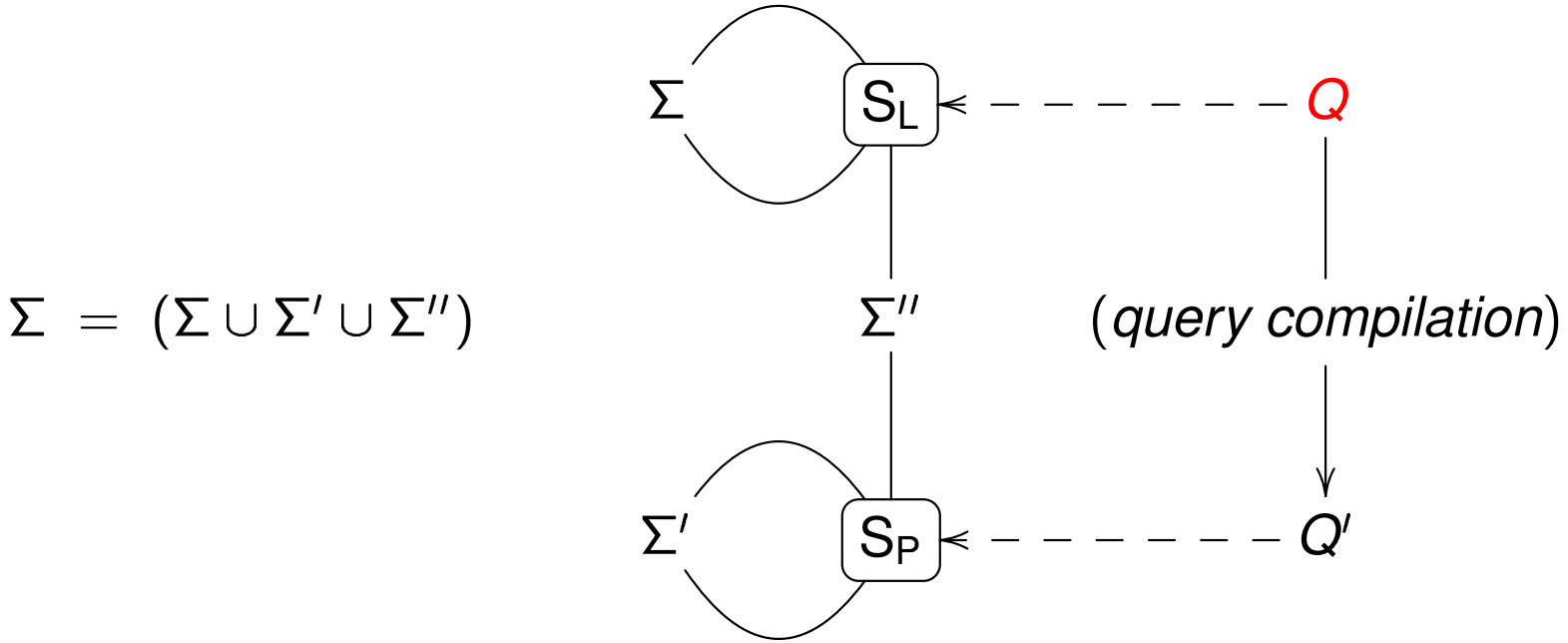
4. There is a kind of entity called an `employee`.
5. There are attributes called `enumber`, `name` and `salary`.

Physical Design and Query Compilation: Overview



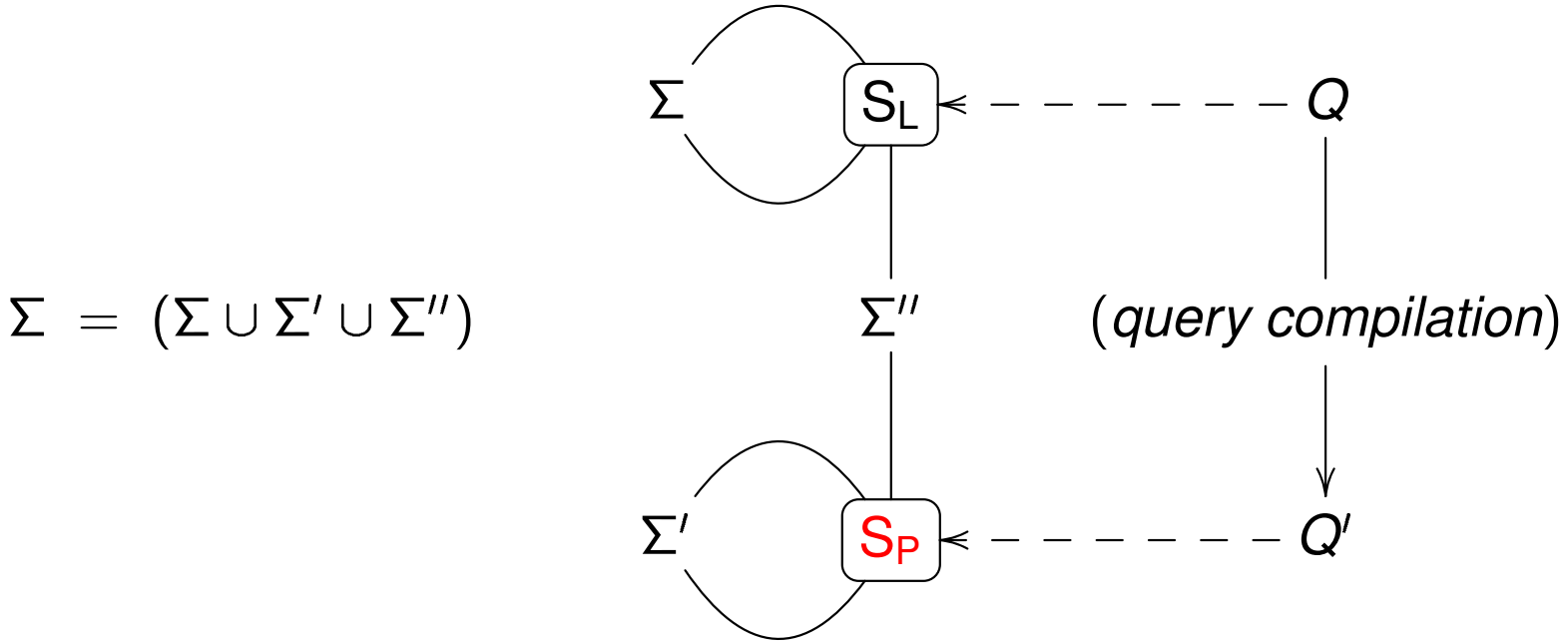
6. Each employee entity has attributes `enumber`, `name` and `salary`.
7. Employees are identified by their `enumber`.

Physical Design and Query Compilation: Overview



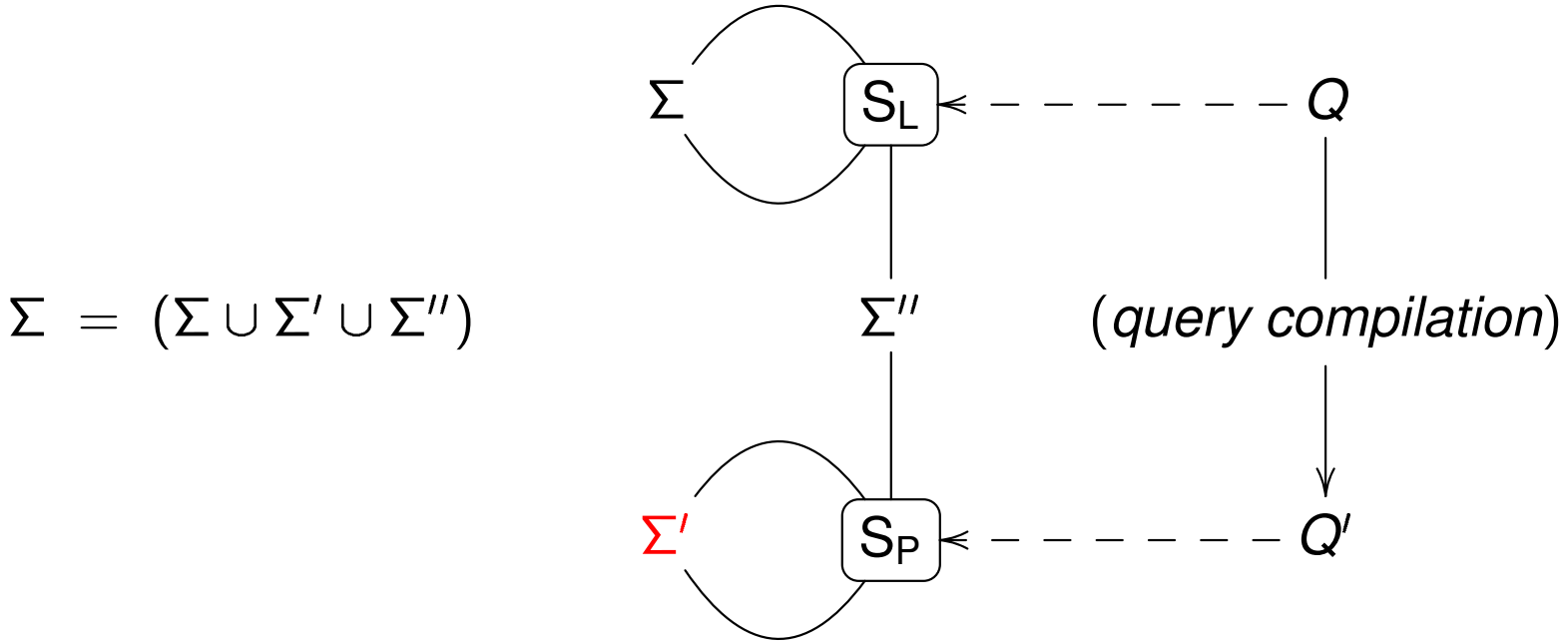
13. Find the `salary` for any `employee` whose `enumber` is given by a parameter p.

Physical Design and Query Compilation: Overview



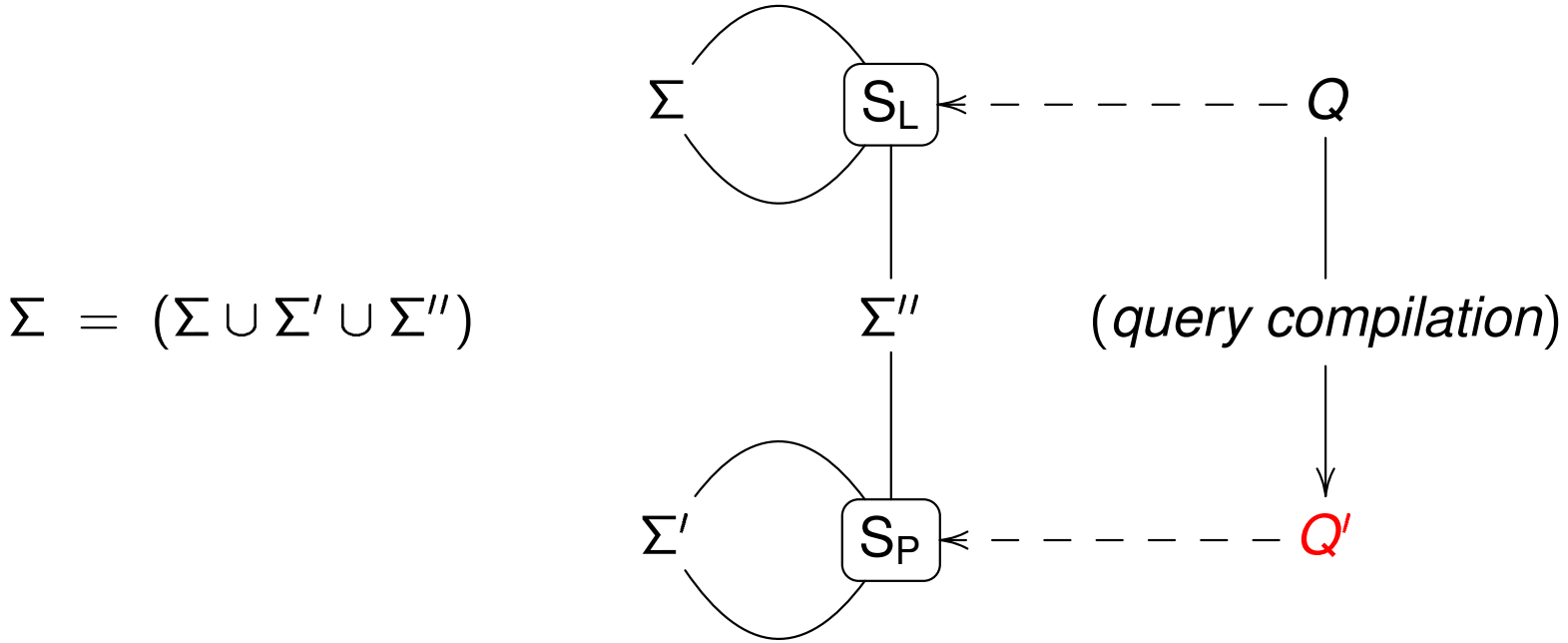
8. There is a file of records called `emp-file`.
9. There are record fields `emp-num`, `emp-name` and `emp-salary`.

Physical Design and Query Compilation: Overview



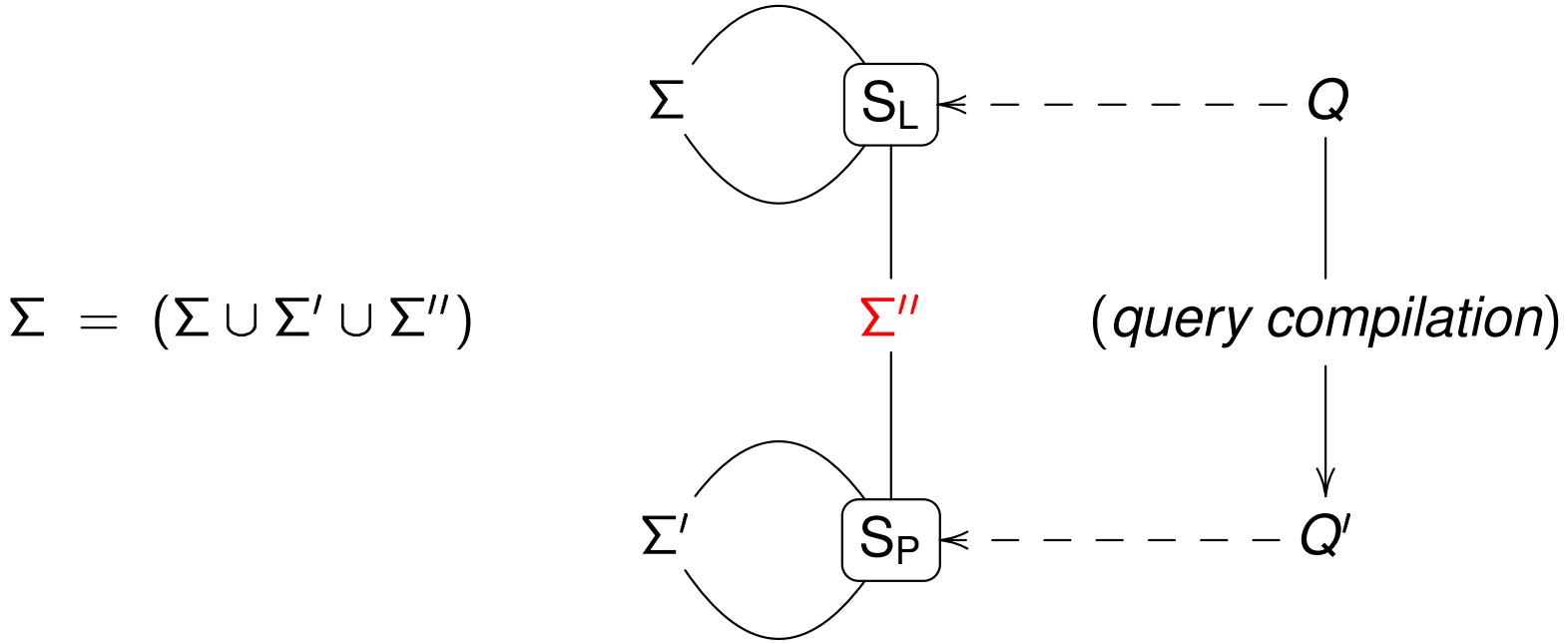
10. Each `emp-file` record has the fields `emp-num`, `emp-name` and `emp-salary`.
11. File `emp-file` is organized as a B-tree data structure that supports an `emp-lookup` operation, given a value for attribute `enumber`.

Physical Design and Query Compilation: Overview



15. Invoke operation `emp-lookup(`p`)`, just once, on file `emp-file`. If an `emp-file` record is found, then extract and return the value of field `emp-salary`.

Physical Design and Query Compilation: Overview



12. Records in file `emp-file` correspond one-to-one with `employee` entities.
13. Record fields in file `emp-file` encode the corresponding attribute values for `employee` entities, for example, `emp-num` encodes an `enumber`.

Food for Thought

Standard (relational) Physical Design

`CREATE TABLE foo` DDL command causes

1. a *logical symbol* `foo` to be created;
2. a (disk-based) *file* of (appropriate) records to be created; and
3. a link between these two objects to be *recorded* (where?)

Food for Thought

Standard (relational) Physical Design

`CREATE TABLE foo` DDL command causes

1. a *logical symbol* `foo` to be created;
2. a (disk-based) *file* of (appropriate) records to be created; and
3. a link between these two objects to be *recorded* (where?)

- Multiple indices for the same table (index only plans)
- Clustering and ordering
- Horizontal and vertical partitioning
- Data replication
- Delegation to other database engines
- Materialized views and cached query results

Food for Thought

Standard (relational) Physical Design

`CREATE TABLE foo` DDL command causes

1. a *logical symbol* `foo` to be created;
2. a (disk-based) *file* of (appropriate) records to be created; and
3. a link between these two objects to be *recorded* (where?)

- Multiple indices for the same table (index only plans)
- Clustering and ordering
- Horizontal and vertical partitioning
- Data replication
- Delegation to other database engines
- Materialized views and cached query results

Assignment for Discussion in next Lecture:

1. How are the above options *recorded* in a RDBMs? (speculation is ok)
2. Is there a uniform and compact way to describe *all* of the above options?