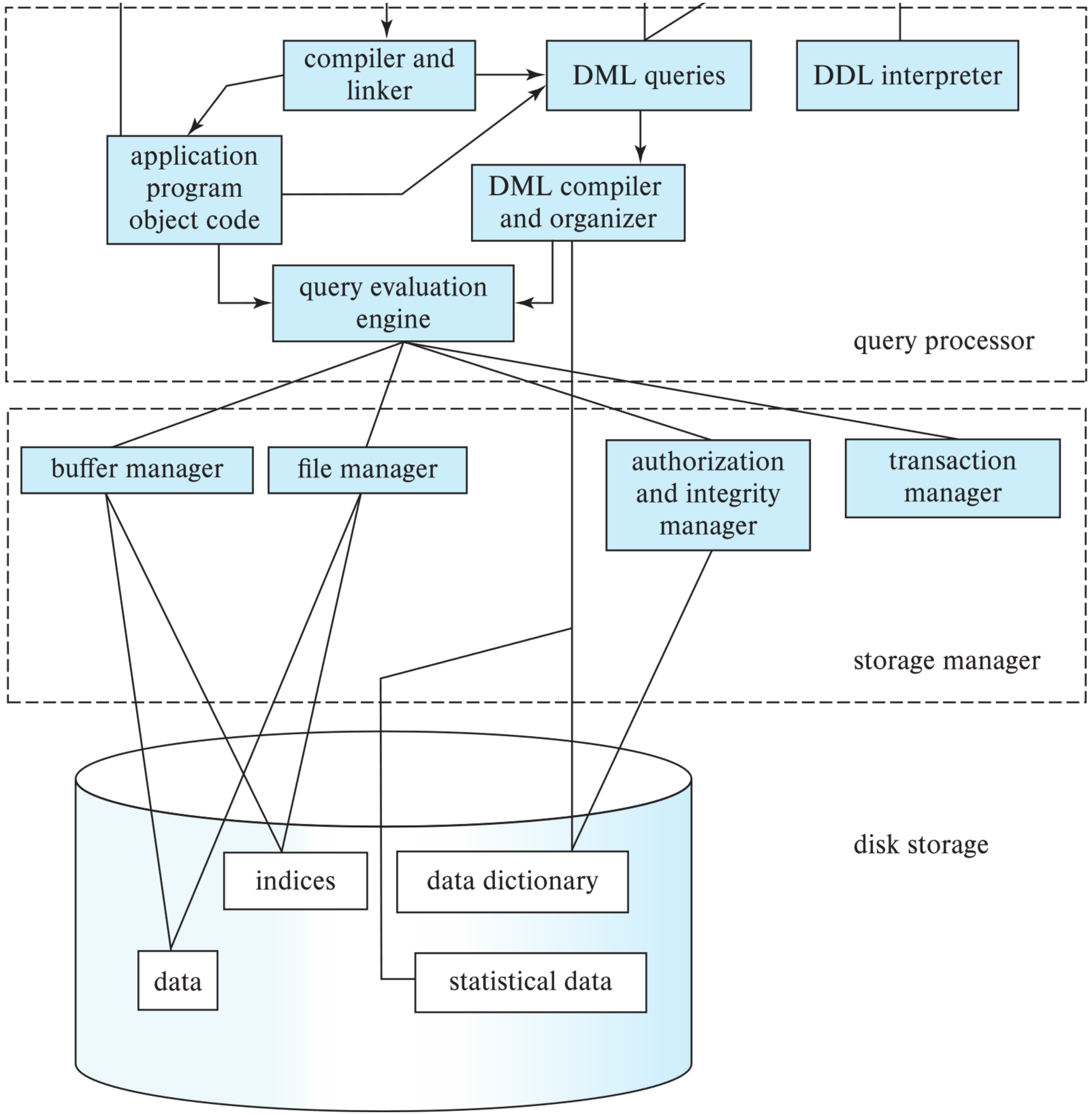
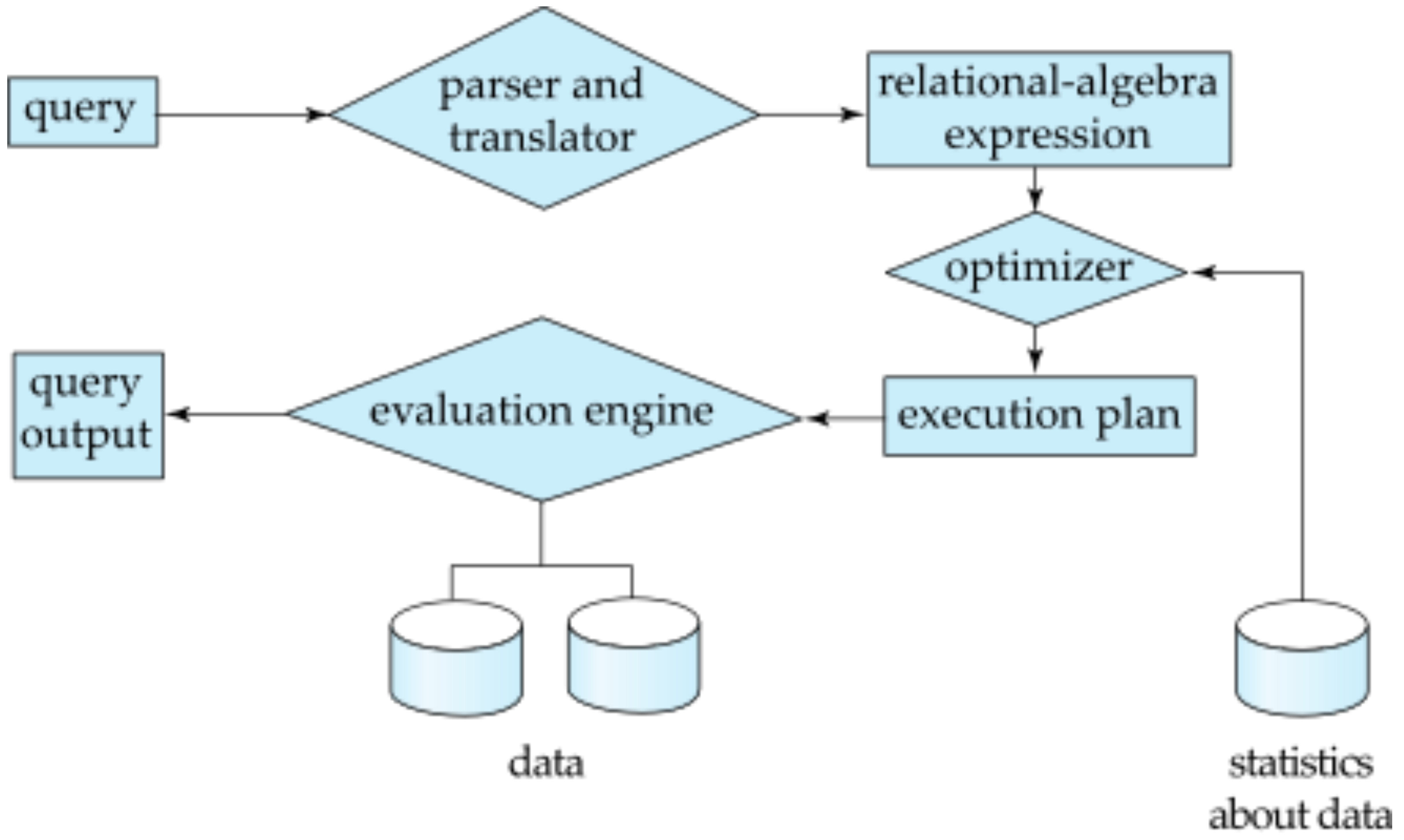
Relational Algebra
(and other stuff)

Database Architecture (Centralized/Shared-Memory)



Query Processing

1. Parsing and translation
2. Optimization
3. Evaluation



Relational Model

- All the data is stored in various tables.
  - row are related (hence the name relational)
  - Columns of tables == attributes
- Rows in tables are (usually) identified by a unique **Key**
  - keys may be one or more columns
  - there may be one than one key
- There are one or more tables in a database
- Tables my be linked (by keys)

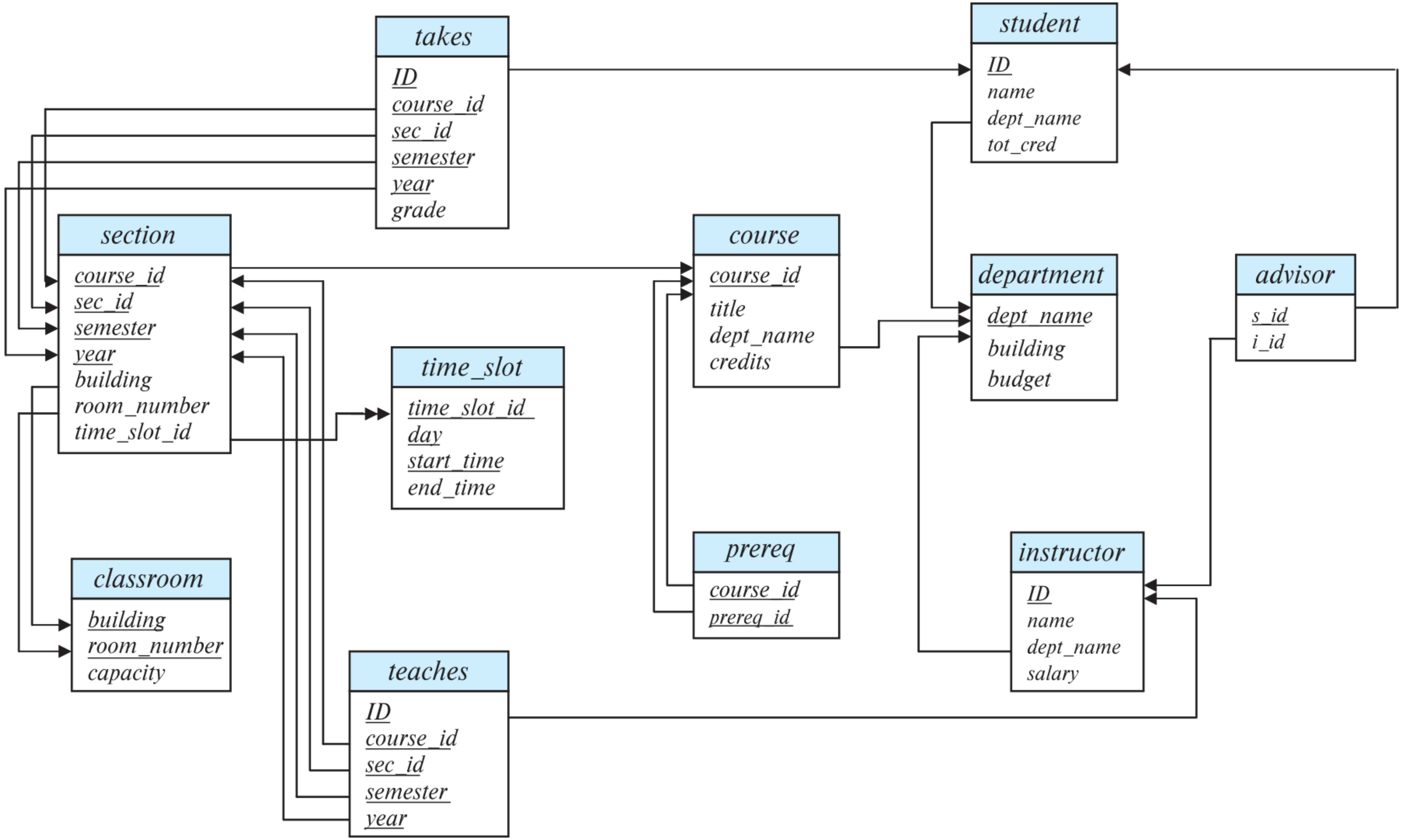
ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

(a) The *instructor* table

Data Manipulation Language (DML)

- Language for accessing and updating the data organized by the appropriate data model
  - DML also known as query language
- There are basically two types of data-manipulation language
  - **Procedural DML** -- require a user to specify what data are needed and how to get those data.
  - **Declarative DML** -- require a user to specify what data are needed without specifying how to get those data.
    - SQL is a declarative DML
    - MongoDB also uses a declarative DML (but it looks fairly procedural)

Schema Diagram for University Database



Relational Algebra

- A procedural language consisting of a set of operations that take one or two relations as input and produce a new relation as their result.
- Six basic operators
  - select: σ
  - project: $\prod$
  - ~~union: $\cup$~~
  - ~~set difference: –~~
  - Cartesian product: x
  - ~~rename: ρ~~

Select Operation

- The **select** operation selects tuples that satisfy a given predicate.
- Notation: $\sigma_{p}(r)$
- *p* is called the **selection predicate**
- Example: select those tuples of the *instructor* relation where the instructor is in the "Physics" department.
  - Query

$$\sigma_{dept\_name=\text{“Physics”}}(instructor)$$

  - Result

ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
33456	Gold	Physics	87000

Select Operation (Cont.)

- We allow comparisons using

  $=, \neq, >, \geq. <. \leq$

  in the selection predicate.
- We can combine several predicates into a larger predicate by using the connectives:

  $\wedge$ (**and**), $\vee$ (**or**), $\neg$ (**not**)
- Example: Find the instructors in Physics with a salary greater $90,000, we write:

  $$\sigma_{\textit{dept\_name}=\text{“Physics”} \wedge \textit{salary} > 90{,}000}\ (\textit{instructor})$$

- The select predicate may include comparisons between two attributes.
  - Example, find all departments whose name is the same as their building name:
  - $\sigma_{\textit{dept\_name}=\textit{building}}\ (\textit{department})$

Project Operation

- A unary operation that returns its argument relation, with certain attributes left out.
- Notation:

$$\prod_{A_1, A_2, A_3 \ldots. A_k} (r)$$

where A_1, A_2, …, A_k are attribute names and r is a relation name.

- The result is defined as the relation of k columns obtained by erasing the columns that are not listed
- Duplicate rows removed from result, since relations are sets

Project Operation Example

- Example: eliminate the *dept_name* attribute of *instructor*
- Query*:*

  $\prod_{ID,\ name,\ salary}$ (*instructor*)

- Result:

ID	*name*	*salary*
10101	Srinivasan	65000
12121	Wu	90000
15151	Mozart	40000
22222	Einstein	95000
32343	El Said	60000
33456	Gold	87000
45565	Katz	75000
58583	Califieri	62000
76543	Singh	80000
76766	Crick	72000
83821	Brandt	92000
98345	Kim	80000

Composition of Relational Operations

- The result of a relational-algebra operation is relation and therefore of relational-algebra operations can be composed together into a **relational-algebra expression**.
- Consider the query -- Find the names of all instructors in the Physics department.

$$\prod_{name}(\sigma_{dept\_name = \text{“Physics”}}(instructor))$$

- Instead of giving the name of a relation as the argument of the projection operation, we give an expression that evaluates to a relation.

Cartesian-Product Operation

- The Cartesian-product operation (denoted by X) allows us to combine information from any two relations.
- Example: the Cartesian product of the relations *instructor* and t*eaches* is written as:

    instructor X *teaches*

- We construct a tuple of the result out of each possible pair of tuples: one from the *instructor* relation and one from the *teaches* relation (see next slide)
- Since the instructor *ID* appears in both relations we distinguish between these attribute by attaching to the attribute the name of the relation from which the attribute originally came.
    - *instructor.ID*
    - *teaches.ID*

The *instructor* x *teaches* table

instructor.ID	*name*	*dept_name*	*salary*	*teaches.ID*	*course_id*	*sec_id*	*semester*	*year*
10101	Srinivasan	Comp. Sci.	65000	10101	CS-101	1	Fall	2017
10101	Srinivasan	Comp. Sci.	65000	10101	CS-315	1	Spring	2018
10101	Srinivasan	Comp. Sci.	65000	10101	CS-347	1	Fall	2017
10101	Srinivasan	Comp. Sci.	65000	12121	FIN-201	1	Spring	2018
10101	Srinivasan	Comp. Sci.	65000	15151	MU-199	1	Spring	2018
10101	Srinivasan	Comp. Sci.	65000	22222	PHY-101	1	Fall	2017
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...
12121	Wu	Finance	90000	10101	CS-101	1	Fall	2017
12121	Wu	Finance	90000	10101	CS-315	1	Spring	2018
12121	Wu	Finance	90000	10101	CS-347	1	Fall	2017
12121	Wu	Finance	90000	12121	FIN-201	1	Spring	2018
12121	Wu	Finance	90000	15151	MU-199	1	Spring	2018
12121	Wu	Finance	90000	22222	PHY-101	1	Fall	2017
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...
15151	Mozart	Music	40000	10101	CS-101	1	Fall	2017
15151	Mozart	Music	40000	10101	CS-315	1	Spring	2018
15151	Mozart	Music	40000	10101	CS-347	1	Fall	2017
15151	Mozart	Music	40000	12121	FIN-201	1	Spring	2018
15151	Mozart	Music	40000	15151	MU-199	1	Spring	2018
15151	Mozart	Music	40000	22222	PHY-101	1	Fall	2017
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...
22222	Einstein	Physics	95000	10101	CS-101	1	Fall	2017
22222	Einstein	Physics	95000	10101	CS-315	1	Spring	2018
22222	Einstein	Physics	95000	10101	CS-347	1	Fall	2017
22222	Einstein	Physics	95000	12121	FIN-201	1	Spring	2018
22222	Einstein	Physics	95000	15151	MU-199	1	Spring	2018
22222	Einstein	Physics	95000	22222	PHY-101	1	Fall	2017
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...

Join Operation

- The Cartesian-Product

  instructor X *teaches*

  associates every tuple of instructor with every tuple of teaches.
  - Most of the resulting rows have information about instructors who did NOT teach a particular course.
- To get only those tuples of "*instructor* X *teaches* " that pertain to instructors and the courses that they taught, we write:

  $\sigma_{instructor.id = teaches.id} (instructor \times teaches))$

  - We get only those tuples of "*instructor* X *teaches"* that pertain to instructors and the courses that they taught.
- The result of this expression, shown in the next slide

Join Operation (Cont.)

- The table corresponding to:

  $\sigma_{\,instructor.id\ =\ teaches.id}\,(instructor\ \text{x}\ teaches))$

instructor.ID	*name*	*dept_name*	*salary*	*teaches.ID*	*course_id*	*sec_id*	*semester*	*year*
10101	Srinivasan	Comp. Sci.	65000	10101	CS-101	1	Fall	2017
10101	Srinivasan	Comp. Sci.	65000	10101	CS-315	1	Spring	2018
10101	Srinivasan	Comp. Sci.	65000	10101	CS-347	1	Fall	2017
12121	Wu	Finance	90000	12121	FIN-201	1	Spring	2018
15151	Mozart	Music	40000	15151	MU-199	1	Spring	2018
22222	Einstein	Physics	95000	22222	PHY-101	1	Fall	2017
32343	El Said	History	60000	32343	HIS-351	1	Spring	2018
45565	Katz	Comp. Sci.	75000	45565	CS-101	1	Spring	2018
45565	Katz	Comp. Sci.	75000	45565	CS-319	1	Spring	2018
76766	Crick	Biology	72000	76766	BIO-101	1	Summer	2017
76766	Crick	Biology	72000	76766	BIO-301	1	Summer	2018
83821	Brandt	Comp. Sci.	92000	83821	CS-190	1	Spring	2017
83821	Brandt	Comp. Sci.	92000	83821	CS-190	2	Spring	2017
83821	Brandt	Comp. Sci.	92000	83821	CS-319	2	Spring	2018
98345	Kim	Elec. Eng.	80000	98345	EE-181	1	Spring	2017

PostgreSQL

starting

- UNIX> psql
  - enter the postgres interpreter logging in using your UNIX id and connected to a database whose name is your UNIX id
    - prompt looks like: gtowell=# where gtowell the the name of the DB you are connected to
  - QUIT: exit; or ctrl-d or \q
- UNIX> psql -h 127.0.0.1 -U xxx_123 -d rocket
  - prompts for a password
    - Enter posgres interpreter logged in as user xxx_123 and connected to the rocket database
      - Entry into postgres that is password based   (will be useful)

PostgreSQL
non-SQL commands

In other DBs (MySQL) many of these commands are in SQL syntax

- \? — show all non-SQL commands
- \q — quit
- \l — list all databases
- \c xxx — connect to a database
- \dt — list tables in database
- \d xxx — describe the table xxx
- \H — generate HTML table

- There are a lot more (use \? to see them all)

PostgreSQL

init file

- /home/USER/.psqlrc

```
\timing
\pset pager off
```

- My .psqlrc file
  - in general, an Unix, a file of the form .*rc in your home directory is a configuration file
    - turn on timing so I see how long each command takes
    - turn off paging (all output to screen immediately — UNIX cat vs UNIX less)

    - see https://www.digitalocean.com/community/tutorials/how-to-customize-the-postgresql-prompt-with-psqlrc-on-ubuntu-14-04

“schema”

- Textbook — schema is the set of tables in a database

- PostgreSQL — a grouping of tables in a database.
  - every database has a default schema named public
    - Idea: permissions can be set on a per schema basis.   So, allow different access to different parts of DB
  - databases may have multiple schema
    - \dn -- shows the current schema

.sql files

kind of like a file containing a program

- comments are line preceded by "—" two dashes

  -- this is a comment

- may contain postgres specific commands
  - \c rocket
- to print a non-query

  gtowell=# \echo 'hello'
  hello

- to use psql < aaa.sql > aaa.out
- to also show commands as they execute

  psql --echo-queries

SQL <--> relational Algebra

<table>
<tr><td>SELECT column [,column]*</td><td>projection --- $\prod_{column\ [,column]^*}$</td></tr>
<tr><td>FROM table [, table]*</td><td>cartesian product -- σ (table1 x table2))</td></tr>
<tr><td>JOIN tableX ON booleanExpression [AND|OR boolleanExpression]*</td><td>cartesian product -- $\sigma_{table1.columnX=table2.columnY}$ (table1 x table2))</td></tr>
<tr><td>WHERE booleanExpression [AND|OR boolleanExpression]*</td><td>selection -- $\sigma_{boolean\ [[\&,\wedge]\ boolean]}$(table)</td></tr>
</table>