Queries in SQL – Product and Join

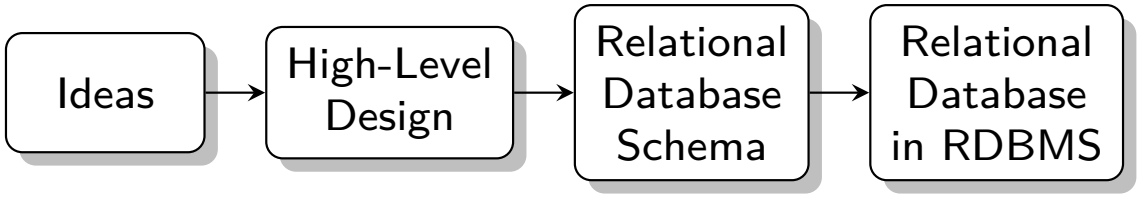
Hui Chen [a]

[a]CUNY Brooklyn College, Brooklyn, NY, USA

March 13, 2025

Outline

Overview


Ideas
High-Level Design
Relational Database Schema
Relational Database in RDBMS

Outline

Introduction to SQL

SQL (pronounced as "sequel") is the principal language used to describe and manipulate relational database, and has several aspects:

- Data definition language (DDL).
  - SQL includes commands to create database objects such as tables, indexes, and views, as well as commands to define access rights to those database objects.
  - Topics of this lecture: SQL commands to create database tables (relations)
- Data manipulation language (DML).
  - SQL includes commands to insert, update, delete, and retrieve data within the database tables.
- Transaction control language (TCL).
  - The DML commands in SQL are executed within the context of a transaction.
- Data control language (DCL).
  - Data control commands are used to control access to data objects.

Outline

Queries to SQL

A SQL can be understood as a relational algrebra query. We discussed

- Selection
- Projection

These queries involve only a single relation. How about the queries involving more than one relation?

- Products
- Joins

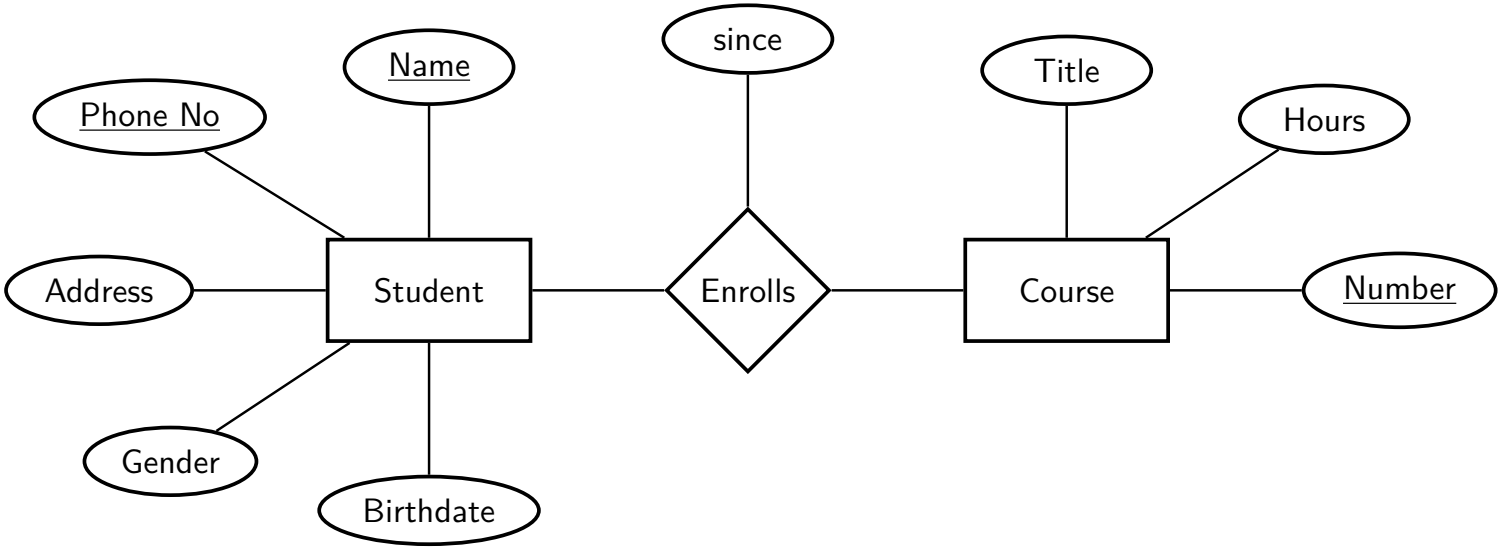
Products

$$R3 := R1 \times R2 \tag{1}$$

```
SELECT *
FROM R1, R2
```

Let's Consider Our Example on Students and Courses ...

Consider the following database model



whose relational database schemas are,

`Students(`<u>`name`</u>`:string, `<u>`phone`</u>`:string, address:string, gender:string, birthdate:date)`

`Courses(name:string, title:string, `<u>`number`</u>`, hours:integer)`

`Enrollment(`<u>`sname`</u>`:string, `<u>`sphone`</u>`:string, `<u>`cnumber`</u>`:string, since:datetime)`

Products in SQL: Example

$$Results := Students \times Enrollment \tag{2}$$

```
SELECT *
FROM Students, Enrollment;
```

more explicitly

```
SELECT *
FROM Students CROSS JOIN Enrollment;
```

θ-Join

$$R3 := R1 \bowtie_C R2 \tag{3}$$

```
SELECT *
FROM R1 JOIN R2
ON C
```

or more explicitly,

```
SELECT *
FROM R1 INNER JOIN R2
ON C
```

"`INNER JOIN`"? Is there an "`OUTER JOIN`"?

Natural Join

$$R3 := R1 \bowtie R2 \tag{4}$$

```
SELECT *
FROM R1 NATURAL JOIN R2
```

θ-Join: Example

$$Results := Students \bowtie_{Students.name=Enrollments.sname AND Students.phone=Enrollment.sphone} Enrollments \quad (5)$$

```
SELECT *
FROM Students INNER JOIN Enrollment
ON Students.phone = Enrollment.sphone
   AND
   Students.name = Enrollment.sname;
```

What question (in English) does this query answer?

Natural-Join: Example

$$Results := Students \bowtie Enrollments \quad (6)$$

```
SELECT *
FROM Students NATURAL JOIN Enrollment
```

For the schemas we have, is it meaningful? What does it really do?

Outline

Combining Projection with Join

$$R3 := \pi_L(R1 \bowtie_C R2) \tag{7}$$

```
SELECT L
FROM R1 JOIN R2
ON C
```

or more explicitly

```
SELECT L
FROM R1 INNER JOIN R2
ON C
```

Combining Projection with Natural Join

$$R3 := \pi_L(R_1 \ltimes R_2) \tag{8}$$

```
SELECT L
FROM R1 NATURAL JOIN R2
```

Projection and θ-Join: Example

$$Results := \pi_{Enrollment.cidnum}(Students \bowtie_{Students.name=Enrollments.snameANDStudents.phone=Enrollment.sphone} Enrollments) \quad (9)$$

```
SELECT Enrollment.cidnum
FROM Students INNER JOIN Enrollment
ON Students.phone = Enrollment.sphone
   AND
   Students.name = Enrollment.sname;
```

What question (in English) does this query answer?

Union, Intersection, and Difference

- (Union) For $\cup$, use `UNION` or `UNION DISTINCT` for set union, use `UNION ALL` for bag union
- (Intersection) For $\cap$, use `INTERSECT` or `INTERSECT DISTINCT` for set intersection, use `INTERSECT ALL` for bag intersection
- (Difference) for $-$, use `EXCEPT` or `EXCEPT DISTINCT` for set difference, use `EXCEPT ALL` for bag difference. Some DBMS also support `MINUS`.

Union, Intersection, and Difference: Examples

Given relation (Enrollment(sname, sphone, cnumber, since)), using Union, Intersection, Difference to answer the following questions:

- Who are the students who take either 3810 or 3171 or both?
- Who are the students who take both 3810 and 3171?
- Who are the students who take 3810 but not 3171?
- Who are the students who take 3171 but not 3810?
- Who are the students who take either 3810 or 3171 but not both?

Summary (1 of 3)

Relations in SQL are bags.

Relational Algebra	SQL Implementation
$\sigma_C(R)$	`SELECT * FROM R WHERE C`
$\pi_L(R)$	`SELECT L FROM R`
$R_1 \times R_2$	`SELECT * FROM R1 CROSS JOIN R2`
$\rho R_1(R_2)$	`ALTER TABLE R1 RENAME TO R2`

Summary (2 of 3)

Relational Algebra	SQL Implementation
$R_1 \cap R_2$	`SELECT * FROM R1 INTERSECT DISTINCT SELECT * FROM R2`
$R_1 \cup R_2$	`SELECT * FROM R1 UNION DISTINCT SELECT * FROM R2`
$R_1 - R_2$	`SELECT * FROM R1 EXCEPT DISTINCT SELECT * FROM R2`

versus

Relational Algebra	SQL Implementation
$R_1 \cap R_2$	`SELECT * FROM R1 INTERSECT ALL SELECT * FROM R2`
$R_1 \cup R_2$	`SELECT * FROM R1 UNION ALL SELECT * FROM R2`
$R_1 - R_2$	`SELECT * FROM R1 EXCEPT ALL SELECT * FROM R2`

Summary (3 of 3)

Inner vs. outer joins. Some DBMS's do not support full outer joins, but you should note $R_1 \rtimes\ltimes_C R_2 = R_1 \rtimes_C R_2 \cup R_1 \ltimes\rtimes_C R_2$

Relational Algebra	SQL Implementation
$R_1 \ltimes_C R_2$	`SELECT * FROM R1 INNER JOIN R2 ON C`
	`SELECT * FROM R1 JOIN R2 ON C`
$R_1 \ltimes R_2$	`SELECT * FROM R1 NATURAL JOIN R2`
$R_1 \rtimes\ltimes_C R_2$	`SELECT * FROM R1 OUTER JOIN R2 ON C`
	`SELECT * FROM R1 FULL OUTER JOIN R2 ON C`
	`SELECT * FROM R1 FULL JOIN R2 ON C`
$R_1 \rtimes_C R_2$	`SELECT * FROM R1 LEFT OUTER JOIN R2 ON C`
	`SELECT * FROM R1 LEFT JOIN R2 ON C`
$R_1 \ltimes\rtimes_C R_2$	`SELECT * FROM R1 RIGHT OUTER JOIN R2 ON C`
	`SELECT * FROM R1 RIGHT JOIN R2 ON C`

Outline

Assignment

Let's work on an assignment using paper and pencil/pen ...