Advanced Database System

Content

Lecture 1 SQL

1. SQL History

    1. 70s **developed** by IBM (SEQUEL), 80s commercial, 86 ANSI, 87 ISO
    2. Structured Query Language

2. SQL Persistence

    1. 90s: Object-Oriented DBMS (OQL)
    2. 00s: XML (Xquery, Xpath, XSLT)
    3. 10s: NoSQL & MapReduce

3. Pros and Cons

    1. Pros:

        1. Declarative: You tell what you need without considering how to get it
        2. Implemented widely: Vsarying levels of efficiency and completeness (Most DBMS support SQL-92)
        3. Feature-rich: With many added features and extensible to other languages and data resources

    2. Cons:

        1. Constrained: Domain specific, only for data, cannot write a program (Not for Turing-test)

4. Terminology:

    1. Database: set of named relations
    2. Relation: a table
        1. Schema: structure of a relation

            $$\mathtt{Student}(\mathtt{sid}\colon \mathit{int}, \mathtt{name}\colon \mathit{text}, \mathtt{dept}\colon \mathit{text})$$

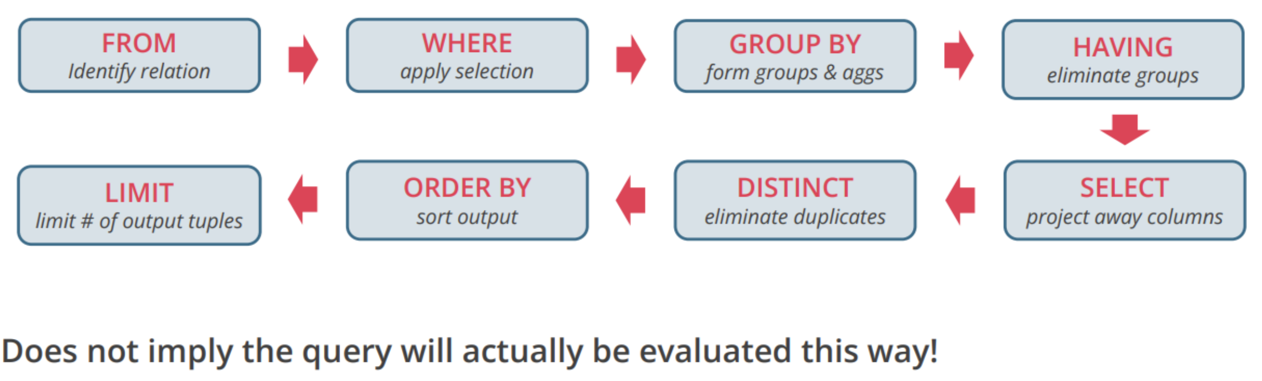
        2. Instance: Collection of data satisfying the schema (a multiset or bag of tuples)
    3. Tuple: a row or record
    4. Attribute: a field or column

5. SQL Language:

    1. DDL: data definition language, define and modify schema
    2. DML: data manipulation language, write queries intuitively
    3. DCL: Data Control Language, control access to data
    4. RDBMS: select and run algorithms for queries, different choices do not change result

6. Single-table queries

```
    SELECT [DISTINCT] <column expression list>
    FROM <single table>
    [WHERE <predicate>]
    [GROUP BY <column list> [HAVING <predicate>]]
    [ORDER BY <column list> [ASC|DESC]] [LIMIT <count> [offset]]
```



FROM Identify relation → WHERE apply selection → GROUP BY form groups & aggs → HAVING eliminate groups → SELECT project away columns → DISTINCT eliminate duplicates → ORDER BY sort output → LIMIT limit # of output tuples

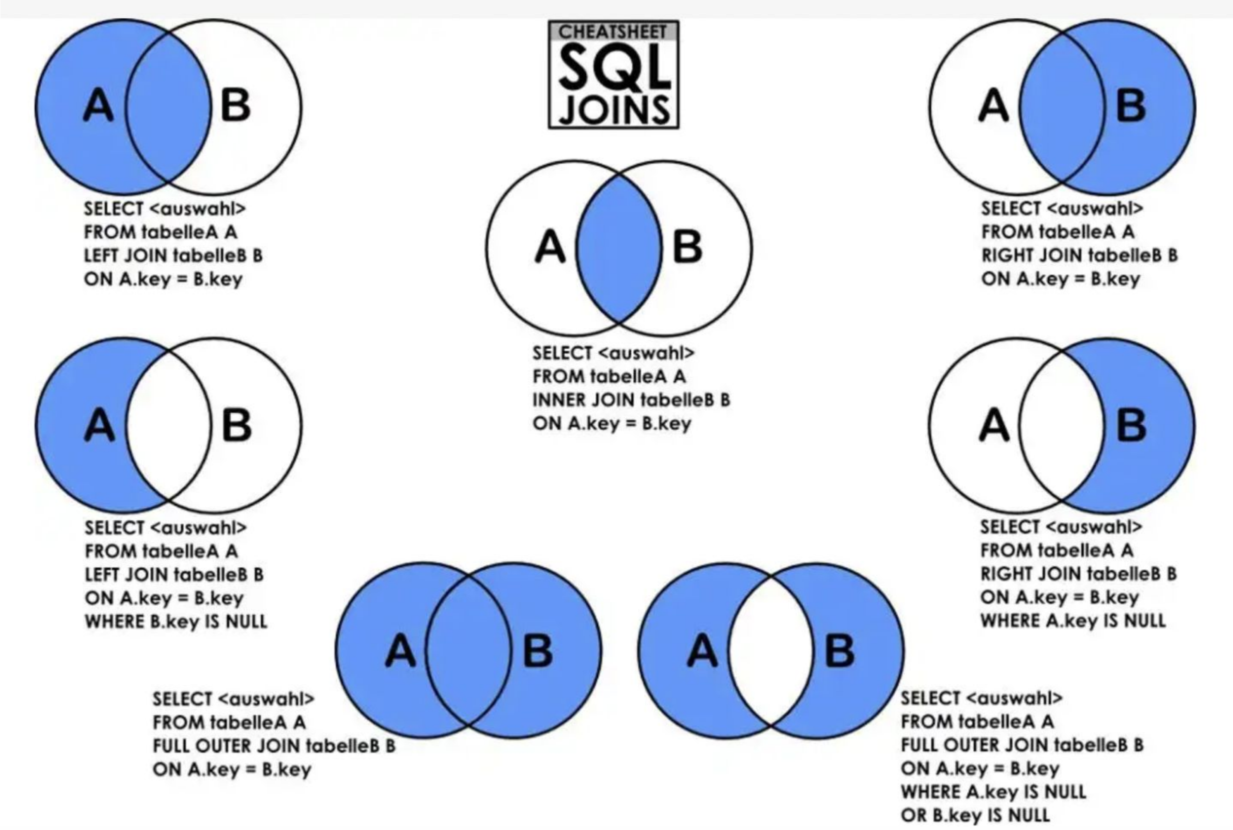
Does not imply the query will actually be evaluated this way!

1. ORDER BY: order by one or more columns, default ASC. `ORDER BY grade DESC, sid ASC`. ***Otherwise the output is non-deterministic, depend on the alog for query processing***
2. LIMIT: limit the number of tuples in output relation。 Can set offset to skip first records. `LIMIT 3 OFFSET 1`
3. AGGREGATES: use functions to return a summary from a group, including AVG, COUNT, SUM, MIN, MAX. `SELECT AVG(age) as avg_age. COUNT(sid) as cnt FROM Student`
4. GROUP BY: patition table into groups with the same GROUP BY column values (can be a list of columns) `GROUP BY dept, name`
5. HAVING: similar to where clause, but for filtering a group. `GROUP BY dept HAVING AVG(age) > 21`

7. Multiple-Table queries

```
    SELECT [DISTINCT] <column expression list>
    FROM <table1 [AS t1], table2 [AS t2], ....>
    [WHERE <predicate>]
    [GROUP BY <column list> [HAVING <predicate>]]
    [ORDER BY <column list> [ASC|DESC]] [LIMIT <count> [offset]]
```

1. JOIN QUERY:



```
SELECT <column list>
FROM <table>
[INNER | NATURAL | {LEFT | RIGHT | FULL} OUTER] JOIN
ON <qualification list>
where ...
```

1. LEFT/RIGHT (OUTER) JOIN: Return all matched rows and preserve all unmatched rows from the table on the left/right of the join clause, use NULLs in the fields of non-matching tuples.
2. FULL OUTER JOIN: Return all matched rows and preserve all unmatched rows with NULLs in unmatched fields.
3. INNER JOIN: Return all matched rows.
4. NATURAL JOIN: Return rows with same field type and name in two tables. (inner join on columns with same name and type). ***The number of tables with the same name should not >1***

8. Nested queries

```
 SELECT S.name FROM Student S
 Where S.sid IN (
     SELECT E.sid FROM Enrolled E
     WHERE E.cid = 'INF-11199'
 )

 SELECT S.name FROM Student S
 Where EXISTS (
     SELECT E.sid FROM Enrolled E
     WHERE E.cid = 'INF-11199'
         AND S.sid = E.sid
```

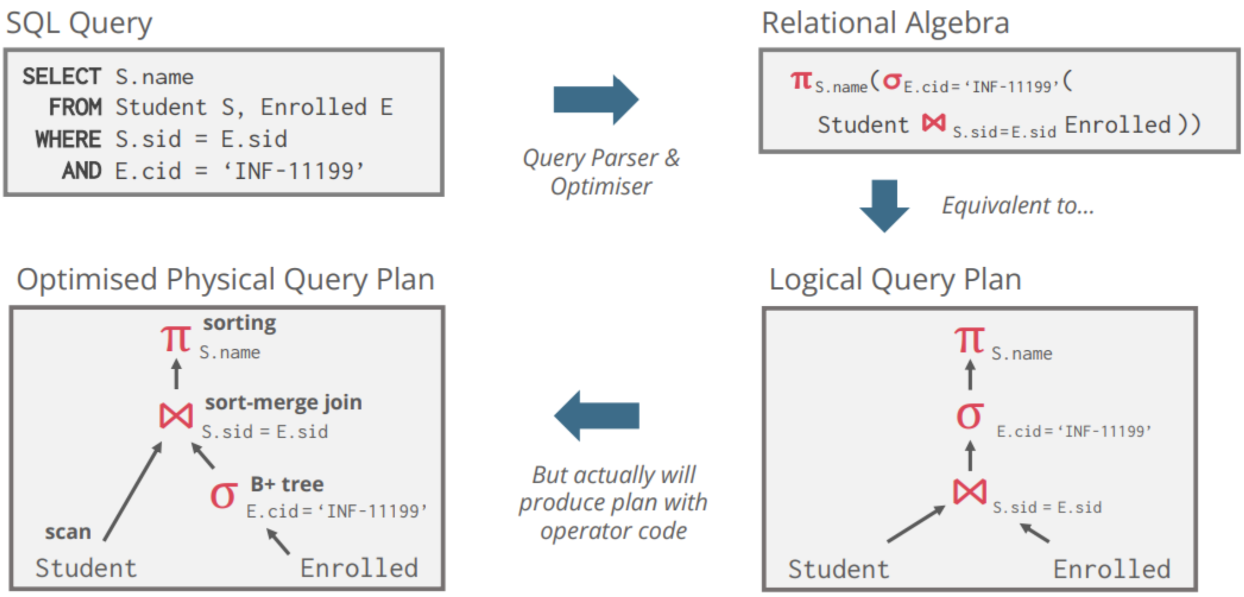
```
        )

```

They are equivelent.

9. Setcomparison operators:
    1. IN, NOT IN, EXISTS, NOT EXISTS, op ALL, op ANY (op means =, <>, >, >=, <, <=)
    2. ALL: must satisfy expression for all rows in subquery
    3. ANY: must satisfy expression for at least one row in subquery
    4. IN: equal to '=ANY()'
    5. NOT IN: equal to '!= ALL()'
    6. EXISTS: at least one row returned

Lecture 2 Relational Algebra

1. Query execution overview



SQL Query

```
SELECT S.name
  FROM Student S, Enrolled E
 WHERE S.sid = E.sid
   AND E.cid = 'INF-11199'
```

Query Parser & Optimiser

Relational Algebra

$\pi_{S.name}(\sigma_{E.cid='INF-11199'}(\text{Student} \bowtie_{S.sid=E.sid} \text{Enrolled}))$

Equivalent to...

Logical Query Plan

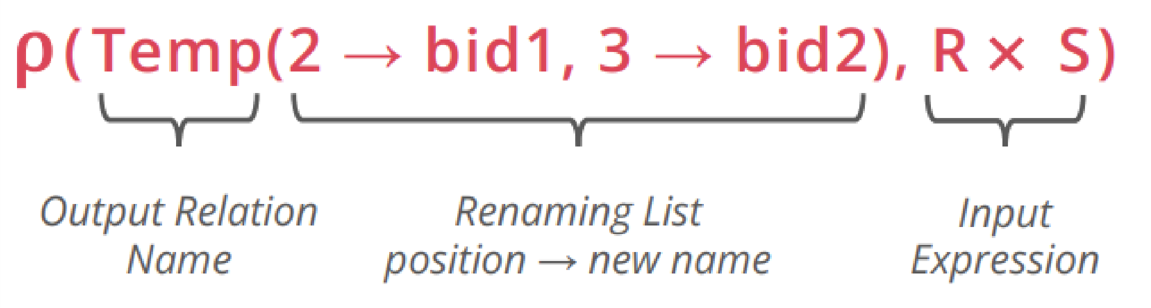
But actually will produce plan with operator code

Optimised Physical Query Plan

    1. SQL query is the declarative description of computation (about what you want)
    2. RA is the operational description of computation (for system execution)

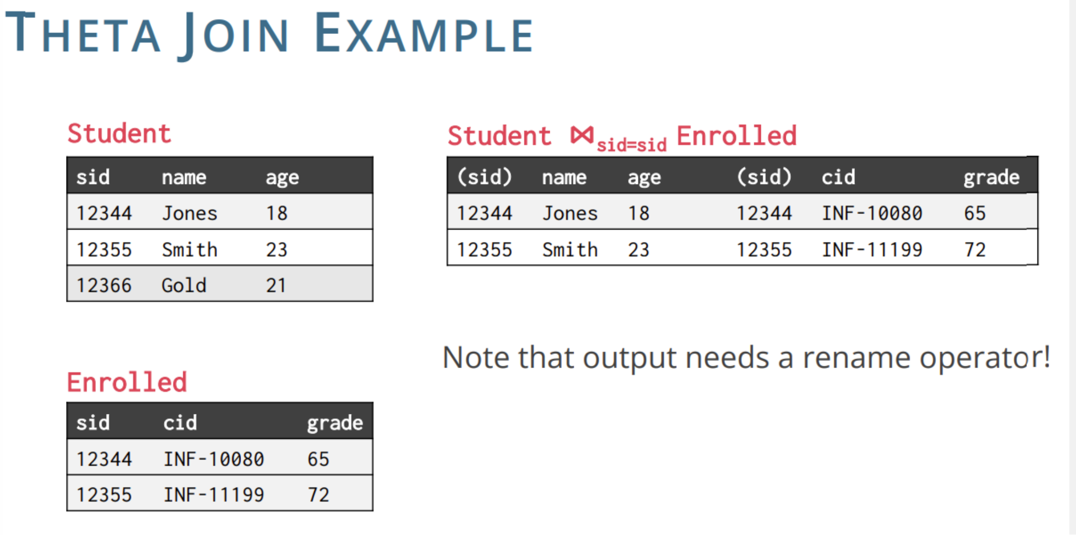
2. Relational query language
    1. Relational Calculus (basis for SQL): Based on first order logic and describe the result of computation. Tuple Relational Calculus (TRC): **{S | S ∈ Student ∃E ∈ Enrolled (S.sid = E.sid ∩ E.cid = 'INF-11999')}**
    2. Relational Algebra: Operational description of transformations, algebra on sets
3. Codd's theorem:
    1. Established equivalence in expressivity between **Relational Calculus and Relational Algebra
    2. Connect declarative representation of queries with operational description. It is constructive and we can compile SQL into relational algebra
4. Relational algebra
    1. features:
        1. Closed: result is also a relation instance, which enables rich composition
        2. Typed: input schema determines output schema, which enables statical check whether queries are legal

2. Operators:
   1. σ : Selection, responde to *WHERE* clause. Select a subset of rows that satisfy a selection predicate. $\boldsymbol{\sigma}_{\boldsymbol{age=21} \wedge \boldsymbol{dept='CS'}}\boldsymbol{(Student)}$
   2. π : Projection, respond to *SELECT* clause. Select a subset of columns. $\boldsymbol{\pi}_{\boldsymbol{age}}\boldsymbol{(Student)}$. Note that set semantic remove duplicates.
   3. ρ : Renaming, rename relations and their attributes. Positional arguments are ed in relational algebra.



$$\rho(\underbrace{\text{Temp}}_{\text{Output Relation Name}}(\underbrace{2 \rightarrow \text{bid1}, 3 \rightarrow \text{bid2}}_{\text{Renaming List position} \rightarrow \text{new name}}), \underbrace{R \times S}_{\text{Input Expression}})$$

   4. ∪ : Union, concatenate two relations. They must have same number of fields and fields in the corresponding positions have same type (compatible). *UNION* eliminates duplicates and *UNION All* keeps duplicates.
   5. − : Set difference, R - S means removing tuples in both of them from R. R and S should be compatible. Respond to *EXCEPT* and *EXCEPT ALL* in SQL.
   6. × : Cross product, R × pairs each row of R with each row of S. |R| * |S| rows in the result. If two attributes with same name in the result, leave them unnamed and identify them by position.
   7. ∩ : Intersection, return the same tuples they share. The input relations should be compatible. respond to *INTERSECT* in SQL.
   8. $\Join$: Join
      1. Hierarchy:
         Theta Join ($\Join$$_{\theta}$): join on loogical expression θ
         - Equi-Join: theta join with theta being a conjunction of equalities
           - Natural Join ($\Join$): equi-join on all matching column names
      2. Theta-Join example



THETA JOIN EXAMPLE

Student

sid	name	age
12344	Jones	18
12355	Smith	23
12366	Gold	21

Enrolled

sid	cid	grade
12344	INF-10080	65
12355	INF-11199	72

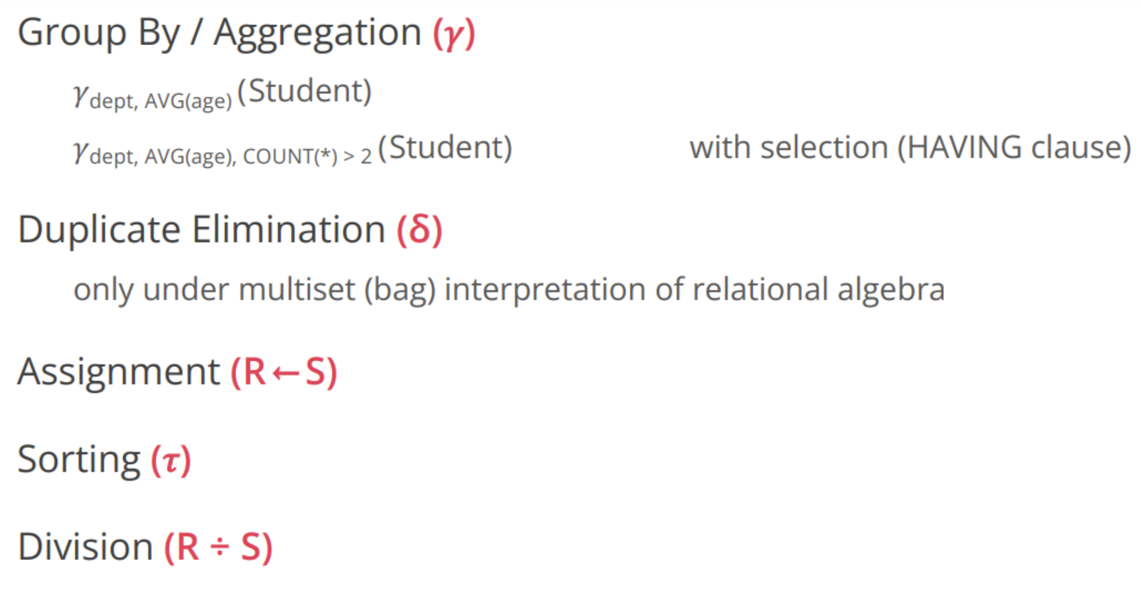
Student $\bowtie_{sid=sid}$ Enrolled

(sid)	name	age	(sid)	cid	grade
12344	Jones	18	12344	INF-10080	65
12355	Smith	23	12355	INF-11199	72

Note that output needs a rename operator!

      3. Natural Join: R $\Join$ S = $\pi_{\text{unique field}}\sigma_{\text{eq.matching field}}(R \times S)$. Compute R × S, select rows where fields appearing in both relaitons have equal values and project onto the set of all unique fields.

9. Extra operators:



Group By / Aggregation (γ)

$\gamma_{\text{dept, AVG(age)}}$(Student)

$\gamma_{\text{dept, AVG(age), COUNT(*)} > 2}$(Student) with selection (HAVING clause)

Duplicate Elimination (δ)

only under multiset (bag) interpretation of relational algebra

Assignment ($R \leftarrow S$)

Sorting (τ)

Division ($R \div S$)

5. Relational algebra and sets
    1. Pure relational algebra has set semantics: no duplicate tuple in a relation instance, but can also be defined over bags (multiset)
    2. SQL has multiset semantics