**10/19/2021**

**Introduction to Relational Calculus**

by K. Yue

E.g.

CREATE TABLE actor (

 actor\_id SMALLINT UNSIGNED NOT NULL AUTO\_INCREMENT,

 first\_name VARCHAR(45) NOT NULL,

 last\_name VARCHAR(45) NOT NULL,

 last\_update TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

 PRIMARY KEY (actor\_id),

 KEY idx\_actor\_last\_name (last\_name)

)ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE [TEMPORARY] TABLE [IF NOT EXISTS] *tbl\_name*

 (*create\_definition*,...)

 [*table\_options*]

 [*partition\_options*]

BNF:

[]: optional

|: or

*tbl\_name: italic: non-terminal*

CREATE: capital letter: terminal

 [*table\_options*]: ENGINE=InnoDB DEFAULT CHARSET=utf8

**DML**

* Make sure you are familiar with the core DML statements.
* Basically declarative.
* CRUD: create, read, update, delete
* Basic update commands: one table at the time.
	+ INSERT: C
	+ UPDATE: U
	+ DELETE: D
* Retrieval commands:
	+ SELECT: R
	+ Various JOIN: JOIN, LEFT JOIN, RIGHT JOIN, etc. can be used within the select statement.

SELECT:

SELECT

 [ALL | DISTINCT | DISTINCTROW ]

 [HIGH\_PRIORITY]

 [STRAIGHT\_JOIN]

 [SQL\_SMALL\_RESULT] [SQL\_BIG\_RESULT] [SQL\_BUFFER\_RESULT]

 [SQL\_NO\_CACHE] [SQL\_CALC\_FOUND\_ROWS]

 *select\_expr* [, *select\_expr*] ...

 [*into\_option*]

 [FROM *table\_references*

 [PARTITION *partition\_list*]]

 [WHERE *where\_condition*]

 [GROUP BY {*col\_name* | *expr* | *position*}, ... [WITH ROLLUP]]

 [HAVING *where\_condition*]

 [WINDOW *window\_name* AS (*window\_spec*)

 [, *window\_name* AS (*window\_spec*)] ...]

 [ORDER BY {*col\_name* | *expr* | *position*}

 [ASC | DESC], ... [WITH ROLLUP]]

 [LIMIT {[*offset*,] *row\_count* | *row\_count* OFFSET *offset*}]

 [*into\_option*]

 [FOR {UPDATE | SHARE}

 [OF *tbl\_name* [, *tbl\_name*] ...]

 [NOWAIT | SKIP LOCKED]

 | LOCK IN SHARE MODE]

 [*into\_option*]

*into\_option*: {

 INTO OUTFILE '*file\_name*'

 [CHARACTER SET *charset\_name*]

 *export\_options*

 | INTO DUMPFILE '*file\_name*'

 | INTO *var\_name* [, *var\_name*] ...

}

SELECT DISTINCT <<result\_columns>> -- [3] construct result columns: RA: project: vertical subseting
FROM <<source\_tables>> -- [1] conceptually join to form a large table:
-- joins
WHERE <<conditions\_for\_inclusion>> -- [2] Filter rows: RA: select: horizontal subsetting

1. <<source\_tables>>: the source tables to gather the result data
2. <<conditions\_for\_inclusion>>: the conditions to be satisfied for results to be included and the conditions the tables should be connected together.
3. <<result\_columns>>: the result columns or expressions desired to be displayed.

SELECT DISTINCT <<result\_columns>> -- [5] construct result columns
FROM <<source\_tables>> -- [1] conceptually join to form a large table
WHERE <<conditions\_for\_inclusion>> -- [2] Filter rows -> raw rows
GROUP BY <<GROUP BY columns>> -- [3] form groups: each row now is a group
 [HAVING where\_condition] -- [4] filter the groups (new rows).

Spring 2020:

(6) List all customer names who have rented 40 or more times in the following format.

+---------------+------------------+
| customer      | Number of rental |
+---------------+------------------+
| ELEANOR HUNT  |               46 |
| KARL SEAL     |               45 |
| MARCIA DEAN   |               42 |
| CLARA SHAW    |               42 |
| TAMMY SANDERS |               41 |
| SUE PETERS    |               40 |
| WESLEY BULL   |               40 |
+---------------+------------------+
7 rows in set (0.02 sec)

-- Q6. List all customer names who have rented 40 or more times

-- in the following format.

SELECT DISTINCT CONCAT(c.first\_name, ' ', c.last\_name) AS customer,

 COUNT(r.rental\_id) AS `Number of rental`

FROM customer c INNER JOIN rental r ON (c.customer\_id = r.customer\_id)

GROUP BY customer

HAVING `Number of rental` >= 40

ORDER BY `Number of rental` DESC;

(7) For each film, lists the id, film title, the number of copies in the inventory, the number of times the film has been rented. Only list the film rented for 32 or more times.

+---------+---------------------+------------+------------+
| film\_id | title               | num\_copies | num\_rented |
+---------+---------------------+------------+------------+
|     103 | BUCKET BROTHERHOOD  |          8 |         34 |
|     738 | ROCKETEER MOTHER    |          8 |         33 |
|     382 | GRIT CLOCKWORK      |          8 |         32 |
|     767 | SCALAWAG DUCK       |          8 |         32 |
|     489 | JUGGLER HARDLY      |          8 |         32 |
|     730 | RIDGEMONT SUBMARINE |          8 |         32 |
|     331 | FORWARD TEMPLE      |          8 |         32 |
+---------+---------------------+------------+------------+
7 rows in set (0.06 sec)

-- Q7. For each film, lists the id, film title, the number of copies in the

-- inventory, the number of times the film has been rented. Only list the film

-- rented for 32 or more times.

SELECT f.film\_id, f.title,

 COUNT(DISTINCT i.inventory\_id) AS num\_copies,

 COUNT(DISTINCT r.rental\_id) AS num\_rented

FROM film f LEFT JOIN inventory i ON (f.film\_id = i.film\_id)

 LEFT JOIN rental r ON (i.inventory\_Id = r.inventory\_id)

GROUP By f.film\_id, f.title

HAVING num\_rented >= 32

ORDER BY num\_rented DESC;