LAB 5 Notes

The Relational Algebra

- Any questions on the project (Discuss)
- In the previous lab we discussed Postgres. Today we will talk al little emore about SQL and if time return to the postgres manual

Ch.1: Overview of Database Systems Ch.2: Introduction to Database Design Ch.3: The Relational Model Ch.4: Relational Algebra Ch.5: SQL Ch.8: Storage and Indexing	Ch.9: Storing Data: Disks and Files Ch.10: Tree-Structured Indexing Ch.11: Hash-Based Indexing Ch.12: Overview of Query Evaluation Ch.13: External Sorting Ch.14: Evaluation of Relational Operators Ch.15: A Typical Relational Query Optimizer Ch.16: Overview of Transaction Management

Outline

1) SQL used in may contexts

2) A Glace at SQL Operators. I will try to emphasize on the most important aspects and then jump into examples

3) Examples on SQL.

4) If time permits we will look at the project manual.

SQL (Structured Query Language)

- Widely used relational database language
- Current ANSI/ISO standard is SQL:1999 but SQL:92 is most widely used
- SQL – Query Language **but has several other aspects**
  1) **DDL** (Definition Language) Create/delete/Alter tables & Views. Creating indexes/ deleting indexes
  2) **DML** (Manipulation Language) Insert/Delete/ Update Rows
  3) **Triggers** SQL:99 supports triggers which are actions

     Triggers are not constrains

     CREATE TABLE products (
     product_no integer,
     name text,
     price numeric **CHECK (price > 0)**

     CREATE TRIGGER if_dist_exists
     BEFORE INSERT OR UPDATE ON products FOR EACH ROW
     EXECUTE PROCEDURE sendemail2managers ('did', 'distributors', 'did');
  4) **Embedded and Dynamic SQL (will be covered as part of the project)**

     Allows SQL code to be executed from a host language such as C or Java.

5) **Security. (chapter 21)**
GRANT SELECT ON products to Cashiers;

6) **Advanced Features.**
SQL:99 supports advanced features like text and XML data management
GRANT SELECT ON products to Cashiers;

A) SQL BASIC QUERY BLOCK

```
SELECT [DISTINCT] select-list
FROM from-list
WHERE qualification;
```

Sailors(<u>sid</u>, name, rating, age)

1	Chris	20
2	Chris	35
3	Chris	20
4	John	15

SELECT DISTINCT name, age
FROM Sailors;
↙ selects all the distinct pairs
i.e. chris, 20
chris, 35

Relational Algebra => p name, age (Sailors)

In fact :
SELECT name, age
FROM Sailors;
IS NOT EQUIVALENT TO SOME IMPLEMENTATIONS OF SQL.
IT IS EQUIVALENT To the ANSI/SQL query because ansi sql works with sets

ANSI/SQL
Chris, 20
Chris, 35

Access/SQL
Chris, 20
Chris, 35
Chris, 20

- FACT with regards to power: **Relational Algebra < SQL92 < SQL99**
- Some queries are more expressive in Relational Algebra. **(e.g. division)** so it is suggested to use both.

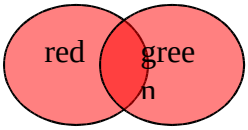
Examples: (Begins and starts with B and has at least three characters)
SELECT *
FROM Sailors
WHERE name LIKE 'B_%B'

B) #1 Set Manipulation constructs: **SQL UNION, INTERSECT AND EXCEPT**

+ Set Manipulation constructs extend the basic query form
+ Union compatible

(SELECT [DISTINCT] select-list-X
FROM from-list
WHERE qualification)
UNION/INTERSECT/EXCEPT (MINUS)
(SELECT [DISTINCT] select-list-X
FROM from-list
WHERE qualification)

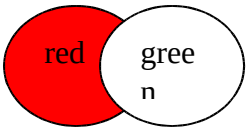
Sailors who reserved Red or green boat



SELECT *
FROM SailorsReserveBoats
WHERE color=red
UNION
SELECT *
FROM SailorsReserveBoats
WHERE color=green

SELECT *
FROM SailorsReserveBoats
WHERE color=red OR color=green;

Sailors who reserved Red but not green boat



SELECT *
FROM SailorsReserveBoats
WHERE color=red
EXCEPT
SELECT *
FROM SailorsReserveBoats
WHERE color=green

C) #2 Set Manipulation constructs: **Correlated Nested and nested IN, EXIST**

(SELECT [DISTINCT] select-list
FROM from-list
WHERE **attribute** [NOT] IN
(SELECT **attribute**
FROM from-list

Union compatible

WHERE condition)

Example:
NOT CORELLATED IN (work well by optimizer)

: Select sailors who reserved boat 103

```
SELECT *
FROM EMPLOYEE
WHERE sid IN
        (SELECT R.sid
         FROM RESERVES R)
```

```
SELECT *
FROM EMPLOYEE E, RESERVES R
WHERE E.sid = R.sid AND
        R.bid=103;
```

CORELLATED EXISTS (ARE NOT optimized adequately)

Allows us to check whether a set is empty or not.
e.g. usually helpful in correlated queries.

```
(SELECT [DISTINCT] select-list
FROM from-list
WHERE EXISTS
        (SELECT attribute
         FROM from-list
         WHERE condition)
```

e.g. select the employees with the highest salary

```
SELECT *
FROM EMPLOYEE E1
WHERE EXISTS (SELECT MAX(E2.salary)
                FROM EMPLOYEE E2
                WHERE E2.id = E.id)
```

```
SELECT *
FROM EMPLOYEE
WHERE E.salary=
        (SELECT MAX(salary)
         FROM EMPLOYEE);
```

D) AGGREGATE OPERATORS

SELECT [COUNT, SUM, AVG, MAX, MIN(attribute)]
FROM from-list
WHERE ~~COUNT(X)~~