# Intro to Database

Danny Tan

10/30/19

# Example

curve

| currency | name | date | df | df_tenor |
|---|---|---|---|---|
| USD | libor3 | 11/2/2012 | 0.95 | 100 |
| USD | OIS | 11/2/2012 | 0.99 | 100 |
| AUD | aud3 | 11/2/2012 | 0.98 | 100 |
| AUD | aud6 | 11/2/2012 | 0.98 | 100 |

# Rule of Precedence

All comparison operators → AND → OR

## Example:

```
SELECT *
FROM table_A
WHERE currency=  ‘AUD'
OR currency =  ‘USD’
AND name =  ‘OIS’
```

Condition1: currency =  ‘USD  ‘ AND name =  ‘OIS’
Condition2: currency =  ‘AUD '

# Example

| currency | name | date | df | df_tenor |
|---|---|---|---|---|
| USD | OIS | 11/2/2012 | 0.99 | 100 |
| AUD | aud3 | 11/2/2012 | 0.98 | 100 |
| AUD | aud6 | 11/2/2012 | 0.98 | 100 |

# Rule of Precedence 2

All comparison operators → AND → OR

## Example:

SELECT *

FROM SV

WHERE (currency = AUD'

OR currency = USD ‘)

AND name = ‘OIS’

Condition1: currency = AUD ‘ OR currency = USD ‘

Condition2: name= ‘OIS ‘

# Example

| currency | name | date | df | df_tenor |
|---|---|---|---|---|
| USD | OIS | 11/2/2012 | 0.99 | 100 |

# Group Functions

- SUM
- AVG
- MAX, MIN
- COUNT
- STDEV, VAR: standard deviation, variance

Functions may be different in different type of database

# Using group functions

SELECT [*column*,] *group_function(column)*
FROM *table*
[WHERE *condition*]
GROUP BY *column*
[ORDER BY *column* [ASC|DESC]]

- All columns in the SELECT clause must appear in the GROUP BY clause
- The GROUP BY column does not have to be in the SELECT list

# Example

| currency | name | date | df |
|---|---|---|---|
| USD | libor3 | 11/2/2013 | 0.95 |
| USD | OIS | 11/2/2013 | 0.96 |
| USD | OIS | 11/2/2013 | 0.97 |
| AUD | libor1 | 11/2/2013 | 0.99 |

Select AVG(df) as p from curve

where currency = ‘USD’

group by name

# Example

| p |
| --- |
| 0.95 |
| 0.965 |

# The HAVING clause

SELECT [*column*,] *group_function(column)*
FROM *table*
[WHERE *condition*]
GROUP BY *column*
[HAVING *group condition*]
[ORDER BY *column* [ASC|DESC]]

- Group conditions can only be restricted by the HAVING clause
- WHERE clause is used to pre-exclude rows before dividing them into groups
- Cannot use column alias in GROUP BY, HAVING

# Example

| currency | name | date | df |
|---|---|---|---|
| USD | libor3 | 11/2/2013 | 0.95 |
| USD | OIS | 11/2/2013 | 0.96 |
| USD | OIS | 11/2/2013 | 0.97 |
| AUD | libor1 | 11/2/2013 | 0.99 |

Select AVG(df) as p from curve

where currency = ‘USD’

group by name

Having name = ‘OIS’

# Example

| p |
| --- |
| 0.965 |

# Relational Algebra

- Algebra of sets concerned with operations over relations
- theoretical foundation for relational databases
- Basic guideline of query languages such as SQL.

# Relational Algebra In SQL

- Union
- Intersection
- set difference
- cartesian product

# Cartesian Product

- Basically, multiple table query

SELECT *table1.column, table2.column*
FROM *table1 [alias], table2 [alias]*

# Example

Table E

| E_ID | ename | dept |
|---|---|---|
| 1 | Ron | A |
| 2 | Alex | C |
| 3 | Mary | A |

Table D

| D_ID | dname |
|---|---|
| A | Marketing |
| B | Sales |
| C | Legal |

**Select * from E,D**

# Example

## Output

| E_ID | ename | dept | D_ID | dname |
|---|---|---|---|---|
| 1 | Ron | A | A | Marketing |
| 1 | Ron | A | B | Sales |
| 1 | Ron | A | C | Legal |
| 2 | Alex | C | A | Marketing |
| 2 | Alex | C | B | Sales |
| 2 | Alex | C | C | Legal |
| 3 | Mary | A | A | Marketing |
| 3 | Mary | A | B | Sales |
| 3 | Mary | A | C | Legal |

- Usually not useful in practice.
- Huge result.

# Multiple table query

SELECT *table1.column, table2.column*
FROM *table1 [alias], table2 [alias]*
WHERE *table1.column1 = table2.column2*

- Write join condition in the WHERE clause
- Simplify queries by using table aliases (if a table alias is used for a particular table name in the FROM clause, then that table alias must be substituted for all the table name throughout the SQL statement)

# Example

```
select * from E, D
where dept = D_id
```

or use alias

```
Select * from E a, D b where
a.Dept= b.D_id
```

# Example

| E_ID | ename | dept | D_ID | dname |
|---|---|---|---|---|
| 1 | Bill | A | A | Marketing |
| 2 | Sarah | C | C | Legal |
| 3 | John | A | A | Marketing |

# DB Designs

- Divides your information into subject-based tables to reduce redundant data.
- Accommodates your data processing and reporting needs.

General Process

- Find and organize the information required
- Divide the information into tables
- Turn information items into columns
- Specify primary keys
- Set up the table relationships
- Refine the design
- Apply the normalization rules

# DB Designs

GOAL:

Design a DB that calculate Option Prices

- Determine the information required
- What kind of options prices?
- What are the market data?
- Where to store result?

# Example

- Suppose its just simple textbook European Option Pricing

we need

$s, \sigma$,t, K, r,

**s will be from current stock table**

$\sigma$ will be from historical stock volatility

t is user input given

K is user input given

r is user input given

# Example

- One table that holds for stock information
- One table holds user input
- One table holds for calculation result

Stock Table: Date,cusips,price

UsersInput Table:

Date,option_id, time_to_expire, rate_uused

Output Table:

Date, option_id, option_prices