Algortihms and datastructures

Kristoffer Klokker

2022

Contents

1 Database introduction

Databases are a collection of data stored in a DBMS (database management system) which serves the purpose of:

- Create database and specifying their schemas (logical structure of the data)
- Query the data (questions about data or retrieving the data)
- Store large amount of data in long periods with easy access and modificatio of the data
- Durable and should be able to recover data in case of error or misuse
- Allow multiple user access at once

Today the norm in database systems are relation databasese which present the data as tables, and the underlying datastructure is not needed for use of the system.
In the case of multiple different database and systems which should be syncronised either a data warehouse is used where a periodically copy of the smaller databases is made. Another approach is a middleware which is a translation between two databases schemes.
A database has mainly two users, admin which can modify the schema using DDL command (data-definition language) which modify the schema by altering the metadata.
The other user being a normal user allowed to do DML command (data-manipulation language).
When a DML command is executed two subsystems are handling the command:

1.1 Query compiler

The compiler takes the query and creates a query plan (a sequence of actions) and passes it the the execution engine.
A request of data sends data data in tuples to the buffer manager, which is responsible for all data transaction between disk storage and memory
The compiler consist of

- Query parser - which builds a tree from the textual query
- Query preprocessor - Sematic check of query to ensure a valid query and transforms the query into algebraic operators

- Query optimizer - Transofrm the query to the best avaliable sequence of operation on the actual data based on metadata and schema structure

1.2 Transaction manager

The transaction manager is used to log for possible recovering and ensuring durablity
Also the transaction has a concurrency-controle manager to ensure a bundle of transaction is executed as they were one unit and locking data when used to ensure no data is wrongly overwritten.
The transaction also manages such that every execution is isolated in case of revertion.
The transaction followed the ACID test, where

- A - atomicity which ensures that in case of error a transaction is never half completed
- C - consistency in data and data constraints
- I - isolation ofe ach operation done in order in a transaction
- D - durability of data such it is never lost after a transaction

2 The relational model of data

A data model is used for describing data and conisist of:

- Structure of data - Referred to as physical data model, but is simply a high level data structure
- Operations on the data - A limited set of operations in DBS at hight level, which makes it more flexible for underlying improvements
- Constraints of data - Constraints on data to ensure data integrity

2.1 The semistructured-data model

The data is setup in a relation more like a tree rather than table.
Here XML is mostly used to represent datam by nested tags.

```
<Movies>
  <Movie title="Gone with the wind">
    <Year>1939</Year>
```

```
        <Length>231</Length>
        <Genre>drama</Genre>
    </Movie>
    <Movie title="Star Wars">
        <Year>1977</Year>
        <Length>124</Length>
        <Genre>sciFi</Genre>
    </Movie>
</Movies>
```

2.2 Document Type Definition model

This is model based on XML to standify a more text based model of a database. This model focus on all the data types in a database and how they are nested. An example is as follows:

```
<!DOCTYPE Stars [
    <!ELEMENT Stars (Star*)>
    <!ELEMENT Star (Name, Address+, Movie*)>
    <!ELEMENT Name (#PCDATA)>
    <!ELEMENT ADDRESS (Street , City)>
    <!ELEMENT Street (#PCDATA)>
    <!ELEMENT CITY (#PCDATA)>
    <!ELEMENT Movie EMPTY>
        <!ATLIST Title CDATA #REQUIRED, Genre CDATA #IMPLIED>
]>
```

Here $\#PCDATA$ is some data for an element and it can here be seen how nesting is allowed by defining star with address wich is also defined later. Here the database Stars also include any number of star as stated at first. The last movie is an alternative writing were instead of $< Movie > Hello < /Movie >$ the $EMPTY$ makes it be $< MovieTitle = "Hello", "Comedy" >$, $CDATA$ is simply characterData and $\#REQUIRED$ means it must be filled whereas $\#IMPLIED$ is optional.

2.3 The basics of relational database

Relation refers to the two dimensional table of data.With attributes being the coloumns and rows being a tuple. The tuple is then made of an relations where a relation with attributes are a schema.
A relation is defined by $Name(attribute : type, attribute2 : type)$ and a tuple is in the same order and valeis for the given attributes.

Relations comes in sets and not lists and therefore order is not important
A database may contain a key which is attribute(s) which define a unique relation, if no combination of attributes are unique a ID for the relation can be created.

2.4 SQL language

SQL is the language used to create queries. SQL has tree kinds of relations, stored called tables (relations), views (relation which are not stored but used for computation), temporary tables (tables constructed by SQL temporary)
The data types avaliable by SQL are:

- $CHAR(n)$ - Character string of fixed length n
- BIT - Logical value with possible values being TRUE, FALSE, UNKNOWN
- INT - Number can also be $SHORTINT$ for small number
- $FLOAT$ - Higher precision numbers here $DOUBLE$ can also be used for more precision
- $DECIMAL(n, d)$ - Numbers of length n and the decinam placed at d
- $DATE$ and $TIME$ - both essentially being strings with a strict format

The basic commands for modifying tables are:

- DROP TABLE R; which removes the table R with all its entries
- ALTER TABLE R ADD a type; Adds attribute a as a *type* to table R
- ALTER TALBE R DROP a; Removes the attribute a form table R

SQL also has $DEFAULT$ which can be added after any attribute after type and describes the default value if non is given.

2.4.1 Keys

A PRIMARY KEY is used for securing no dublicates and only allows non null values in the key attribute.
UNIQUE allows null as a value in its attribute, but dublicates is still nto allowed.

When creating a table the key can be choosen by after an attribute after its type $PRIMARY\ KEY$ or $UNIQUE$ is inserted or at the end of the table definition $PRIMARY\ KEY\ (a)$ can be inserted where a are the attributes. Again Unique can also be used like this.

3 Algebraic Query Language

The algebraic query langauge is the operation behind the SQL real language. This is not a programming language, but the simplicity makes in easier to optimize and faster.
All the operations works on boths sets and bags(the allowance of mutliple accourences of tuples)
The following table is in precedence order from first at top and last on bottom.

Name	Symbol	Effect	*
Selection	$\sigma_C(R)$	Select tuples from R which meets the condition C	
Projection	$\pi_{A1,A2}(R)$	All tuples from R but only attributes $A1$ and $A2$	
Rename	$\rho_{S(A1,A2)}(R)$	All tuples in R but rename attributes to A1 and A2 Alternative $R2_{A1,A2} := R$	
Cartesian	$R \times S$	Every possible combination of tuples from R and S	2
Nat. join	$R \bowtie S$	Cartesian product but only where overlapping attributes are equal	2
Theta join	$R \bowtie_C S$	Every cartesian product of R and S which meet the C condition	2
Difference	$R - S$	Tuples which in R which is not in S	1
Union	$R \cup S$	All tuples from R and S	1
Intersect	$R \cap S$	Tuples which are in both R and S	1

1. Same attributes (and same type) in same order
2. In case of attributes with same name they will be "renamed" to 'relationName.Attribute'

It can here be seen that there is multiple combinations which are equal. Here the highest priority readability due to the query compiler rewriting it anyway. When writing linear notation be used as such:

$$R(a, y, l) := \sigma_{age<18}(People)$$

From which R now can be usedas a variable.
Often when combining operationsa tree is used where root is the final product and branches are the first operations done.

3.1 Examples

A

A	B
1	5
2	4

B

B	C
5	8
4	9

C

A	B
1	5
7	0

$A \cup C$

A	B
1	5
2	4
1	5
7	0

$A \cap C$

A	B
1	5

$A - C$

A	B
2	4

$\pi_B(A)$

B
5
4

$A \times B$

A	A.B	B.B	C
1	5	5	8
1	5	4	9
2	4	5	8
2	4	4	9

$\sigma_{B<4}(C)$

A	B
7	0

$A \bowtie_{A.B=B.B} C$

A	A.B	B.B	C
1	5	5	8
2	4	4	9

$A \bowtie B$

A	B	C
1	5	8
2	4	9

$\rho_{G,K}(A)$

G	K
1	5
2	4

4 Queries in SQL

4.1 Select

The most simple query is the SELECT FROM WHERE.
This a simple select query which selects attributes from a database where a conditions is met.
The conditions can use the 6 comparisions operators: $=, <>, <, >, <=, >=$ where $<>$ is not equal.
Operations may also be done on values for int $+, -, *, /, \%$ and for string $||$ which adds two string together.
When comparing string the $<$ operators are defined upon lexicongraphic order.
For string the LIKE can be used to match a pattern. A pattern uses % and _ where % is any number of wildcard charachters and _ is a single charachter.
Strings are defined by ' therefore when working with them inside a string two ' is used like ''
Logic operators avaliable is AND, OR and NOT
Ex. SELECT title, genre FROM Books WHERE author = Kristoffer Klokker

AND year = 2022

4.2 Rename

In case of a rename of attributes the keywords AS is used, here it is also possible to modify values.
Ex. SELECT title + ' and the stone' AS name from Books
COnstants are also avaliable if wanted.
Ex. SELECT title —— ' and the stone' AS name, 'forever' AS readTime from Books

4.3 Date and time

Dates are defined as: DATE 'year-month-day' Ex: DATE '2022-12-04'
Times are defined as: TIME 'hour:minut:second' Ex: TIME '19:42:12.55'
In case of timezones TIME 'hour:minut:second-hour:minut' this will be the amount of hours and minuts behind GMT. The minus could also be a plus.
Timestamps are used to combine date and time.
Timestamps are defined as 'TIMESTAMP 'year-month-day hour:minut:second'
These values can of course be compared both with equal and all the < as expected.

4.4 Working with null

A sql entry may be null due to: not knowing the value, no value really match the attribute or the value is withheld.
This can have consequence in the query condition.
When using any math operations on null it becomes null.
When comparing anything to null it becomes UNKNOWN.
To check if a value is null the IS NULL is used.
In the opearation 'UNKNOWN AND x' the statement is false when x is false and UNKNOWN when x is true.
In the operation 'UNKNOWN OR x' the statement is true when x is true and UNKNOWN when x us false.
When negating UNKNOWN the value is UNKNOWN.
UNKNOWN values will not interpretated as true and queries where the condition result in UNKNOWN will not show up.

4.5 Ordering

To order the output of a qeruy the ORDER BY ¡attributes¿ is used.
After the attributes list DESC can be added for a descending list whereas ASC is not needed due to ascending being the default.
Multiple attributes are used in case of a the first attribute being equal then the second attribute is used for ordering.

4.6 Multiple tables

When working with multiple tables the FROM can be filled up with multiple tables.
Ex. SELECT title, age FROM Books CoolPeople WHERE author = name
In case of the table having the same name or the same table being used an alias can be created by a space after the table followed by the wanted name.
Ex. SELECT Book1.name, Book2.name FROM Books Book1, Books Book2 WHERE Book1.title = Book2.title AND Book1.name ¡¿ Book2.name

4.7 Union, intersect and except

The relational algebra operation can be used on two SELECT statement to get the expected output.
Ex. SELECT name FROM Books INTERSECT SELECT name FROM CoolPeople
For union and except intersect is replaced with UNION or EXCEPT.
These operations will eliminate dublicates to prevent this the ALL keyword can be used. Ex. SELECT name FROM Books INTERSECT ALL SELECT name FROM CoolPeople

4.8 Join

The simplest join is the CROSS join also known as cartesian product Ex. Books CROSS JOIN CoolPeople
Theta joing the cross product but with a condition is done like cross but with ON and then condition Ex. Books JOIN CoolPeople ON author = name
Natural join where attributes are matched is done by taking the two tables with NATURAL JOIN between Ex. CoolPeople NATRUAL JOIN SmartPeople
Outher join is like natural join except instead of throwing away non matching

entries they are filled with null Ex. CoolPople NATURAL FULL OUTER JOIN SmartPeople.
This will include all entries from SmartPeople and CoolPeople but if an entry only exists in one the attributes from the other table is fileld with null.
Variant of LEFT and RIGHT also where left only includes all entries from the left table and only matching from the right Ex. CoolPeople NATURAL LEFT OUTER JOIN SmartPeople.

4.9 Subqueries

Subqueries are used for getting a value from a table which then is most often used in a comapre statement or as a table the selection is taken from.
If a subquery only return one value it is called scalar and can be used in an condition.
Ex SELECT title FROM Books Where author = (SELECT name FROM CoolPeople WHERE key = 3)
In case a subquery return multiple result the following operators can be used:

- EXISTS R - return true if R is not empty
- s IN R - return true if s is an entry in R
- s (>) ALL R - returns true if the math operation here ¿ is true for all entried in R
- s (>) ANY R - returns true if the math operation here ¿ is true for any entru in R

All the operators can ofcourse be negated by puttin NOT infront.

4.10 Sets

To get a set in a query the DISTINCT is used after the SELECT Ex. SELECT DISTINCT names FROM...

4.11 Aggregation operations

This is operations which is done on the while selection.
These operations include: SUM, AVG, MIN, MAX and COUNT.
Ex. SELECT AVG(age) FROM CoolPeople
In case of wanting only sets DISTINCT is used as follow SELECT SUM(DISTINCT

age) FROM CoolPeople.
GROUP BY is often here used to get the aggregation to a given group. Ex. SELECT name, SUM(age) FROM CoolPeople GROUP BY name.
This will give the age sum for each name in cool people.
NULL is ignored when calcilation an aggregation but a group can be in NULL.
THE GROUP BY can also works as a condition with the keyword HAVING. Ex. SELECT name, SUM(age) FROM CoolPeople GROUP BY name HAVING MIN(age) $>$ 18
This will return the names and the sum of ages with the same name but only people over 18 count.