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

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

A	B
7	0

$\pi_B(A)$

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

$A \bowtie B$

A	B	C
1	5	8
2	4	9

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

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

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

G	K
1	5
2	4

4 Transaction

A transaction is a collection of queries which is executed atomicly.
This is to prevent race conditions in case of read then write.
The transaction is also used such it can both end up commiting or rollback changes it has done.
The transaction is started by: START TRANSACTION . . IF... COMMIT ELSE ROLLBACK A transaction can also have different proberties. This includes it being able to read only or red write. Where read only makes it possible for more concurrency.
This is set by SET TRANSACTION READ ONLY OR READ WRITE in the other case, but this is default.
This is added like other attributes before the transaction starts.

For a transaction there are 3 different terms which is used for isolations levels:

- Dirty reads
- Nonrepeatable reads

- Phantoms

Dirty reads are a term for uncommitted data, which may be read. This may be a tradeoff which can be made for a faster operation.
Nonrepeatable reads refers to if a transaction does multiple reads, if allowed the reads may differ due to database manipulations or inserts.
Phantom tubles are tubles which are inserted while a transaction is ongoing.

There are 4 diffferent isolations levels:

- Read uncommited
- Read commited
- Repeatable read
- SSerializable

As default the transaction is set as Serializable, which does not read dirty, has repeatable read, and does not allow phantoms.
Repeatable read, does not allow dirty reads, has repeatable reads and does allow phantoms.
Read committed does not allow dirty reads, has nonrepeatable reads and allow phantoms.
Read uncommited allows dirty reads and phantoms and is nonrepeatable.
To set a transaction level it can be done like: SET TRANSACTION ISOLATION LEVEL REPEATABLE READ; DISCLAIMER: if dirty reads are allowed the default is read only and not read write.

5 Design theory for Relational Databases

5.1 Functional dependencies

A functional dependecy is defined as:

$$A_1, A_2, ..., A_n \rightarrow B_1, B_2, ..., B_n$$

Functional dependecies are constraints on a relation which states:
If two tuples agree on the A attributes of the dependency they must agree on the B attributes
The transitive rule states for the FDs $A \rightarrow B$ and $B \rightarrow C$ will $A \rightarrow C$ be true.
The splitting combining rule states for the FD $A \rightarrow BC$, it can be spliited

into $A \to B$ and $A \to C$, or combined the other way around.
The trivial FD for the attribute A is $A \to A$.
A basis of FD's are just the given FD's
The minimal basis of FD's are only singletons, and removing any FD will not be equivalent to the basis.

5.1.1 Keys

A key is a minimal set of attributes which will be unique for every set of attributes.

$$\{A_1, A_2, ..., A_n\}$$

Here the attributes are part of the key.
Superkey: A set of attributes of which one is a key.
Some uses the terminology where key is not minimal and candidate key is minimal.

5.1.2 Closures

A closure for the attribute A, is all attributes, which can be computed from which $\{A\}^+ \to \{A, B, C\}$, from the singletons $A \to B$ and $B \to C$
The closure is found by starting with the trivial, then FD's which include the derived attributes is inserted until no more informations from FD's can be added.
The closure of $\{superKey\}^+$ will result in all attributes.
The colsure of $\{key\}^+$ will also result in all attributes, but no attribute of the key can be removed.
Projecting FD's onto a new relation with a limited amount attributes, will only the FD's which involves the new relation hold.

6 Designing a Database

6.1 Anomalies

Anomailies are repeated information in multiple tuplees.
The can lead to lost of data, if repeated data turns out to be the last data when deleted or non updated values for all repeated data.
To prevent this decomposing relations can be done

This is where all the repeated data is in one relation, and the none repated attribute is in another relation with key also.

6.2 Boyce-Codd Normal Form

To decompose the relation Boyce-Codd Normal form (BCNF) is used.
This is a rule which states, that for all FD's in the relations the left sides should be part of the super key.
So the the relation $R(A, B, C)$ with the FD's $A \to B$ and $A \to C$, will A be the super key.
This relation will therefore be decomposible into two relations $R(A, B)$ and $R(A, C)$.
It can here be noted that, transitive relations may be used.
Example the relation $R(A, B, C, D)$ with the FD's $A \to B$, $B \to C$ and $B \to D$.
It can here be seen A is the key due to being the only attribute from which all other attributes can be found.
Therefore the relation can be described in two relations $R(A, B)$,$R(B, C, D)$.
Here BCNF will hold.
Performing a decompose of R using $X \to Y$, is done by having $R_1 = X^+$ and $R_2 = R - (X^+ - X)$
The third normal form is a modification of BCNF where the FD $X \to A$ is only a violation if X is not a superkey and A is prime (member of a key).
3NF can be smart due to it will always preserve FD's, which BCNF does not gurantee.

6.2.1 Decompositiong with 3NF

First all FD's are splitted. Then the FD's are simplified or removed if repeated.
Then the candidate key is all, keys on the right of the FD's clousing sets.
The relations will then be the simplified / minmal basis of FD's.

6.3 Valid decompose

A measure if the decomposition is correct, is if when the natural join is performed on the given decomposed relations, we shall get the original relation back with the same number of tuples and correct tuples.
The case method, consist of writing a tablau, with a row representing a decomposed relation. Then the row is filled with know values and unknown

valeus are subscribted with the row number.
Then from the FD's the rows should be able to be combined by finding equal variables from which the subscribted variables are removed, ending up with a row with no subscribted values.

7 High-level DB models

The most used model is the ER diagram, which describes schemas and their relation to other schemas.
The ER model consist of:

- Rectangles - The sets / schemas
- Ovals - The attributes
- Diamonds - The relation between two schemas

A tuple in the model is called a relationship.
A relation may have mannnny sets attached.
Edges may also have labels, to indicate if two schemas are related in more ways.
A relation may also have attributes, in cases where it makes the most sense. This can also be replaced by a relation more if, many attributes may accour.
Some models like UML does not allow more than a binary relation. To combat this a connecting schema, from which relations can go out to each schema.

7.1 Multiplicity of ER models

ER diagrams can also show restrictions, this include the number of relations a schema can have.
The standard is many-many, this means a relation may involve as many as it wants from both sides.
Then there is many-one which dictates that, a relation may only be related to one relation. This is indicated by an arrow which points towards the set if which only one can exist.
There are also one-one which dictates only one relation can relate to another relation. This is indicated by an arrow at both ends of an edge.

7.2 Subclasses in ER models

A subclass use a isa object to show the subclass relation.
The object is in form of a triangle, where the parent is connected at the top and the subclass at the bottom.
The subclass will then inhert all attributes from the parent and can have new attributes or relations.
A isa object is always one-one, but the arrows are not needed.

7.3 Designing a model

To create a good model, the model should be simple and not include redundant infomration.
Be of course as faithfull to what it tries to model.
In some cases, a relation may not be the answer. If the the set E:

- Has only arrows entering it.
- The only key in E is all attributes.
- No relationship involves E more than once

If all these cases are met, the relation and the set should be removed, and the attributes of E and the relation should be in the related schema.

7.4 Keys in ER models

In ER models keys work like in DBMS.
Here the constraints are that a set must have a key, otherwise it is not a set, which a relation can form.
There may be more than one key, but a primary key must be picked.
With isa's the enitity must have a key which is inherted from the parent.
The key is shown by underlining the key attribute(s).
A weak key is a key from another set.
Weak keys can be valid, in a many-one binary relation which must exist.
In this scenario a key from set F can be used as a weak key for set E.
This is illustrated by the a double rectangle on the set with the weak link and a double border on the relations diamond which provides the weak key.

7.5 Constraints

A model may also include constraints in the relations.
By using rounded arrows, not only is it many-one relation, it states that at least one set must exists in the relation.
In the case of a restriction on the many-many relation, number critera (Ex, < 10) is written at the intersect of edge and set and states the minimum.

7.6 Converting from ER diagram to DB model

To convert it is pretty straight up. The given sets and relations are converted into schemas, with the given attributes.
It may be needed to combine some schemas, this is common if a schema mostly just contains keys, and a many-one is in place.
In this case a big schema can be made, where the relations attributes and the schema with mostly key attributes are added.
When working with weak keys, the key attribute simply is the weak schema, the relation schema and of course the original schema.

7.6.1 isa objects

To convert isa object there are multiple ways.
The most strightforward way is creating everything as schemas.
Then the keys will connect each schema to the other to obtain all information.
The object oriented approach, makes every possible schema combination according to the isa object.
The null approach creates a schema with all attributes, from every set in the isa. Then null values are made when the object is parent etc.
The null approach makes queries the most simple and fast, but uses more unused space unlike the object orented approach which has only one tuble for every entry with no nulls, but many schemas.

8 Stored Procedures

Stored procedures are function in query language.
They are defined by:
CREATE PROCEDURE ¡name¿ (¡parameters¿) . . . In case of a return is is: CREATE PROCEDURE ¡name¿ (¡params¿) RETURN ¡type¿ The body of the procedure can be SQL commands and other forms of command listed below.

8.1 Variables

Variables are done by: DECLARE ¡name¿ ¡type¿; SET ¡name¿ = ¡expression¿;

8.2 Procedure calls

CALL ¡name¿(¡params¿);

8.3 If statements

If statements are build up in the following form: IF ¡condition¿ THEN ... ELSEIF ¡condition¿ THEN ... ELSE ... END IF; Where conditions are done through sql queries like NOT EXISTS and COUNT.

8.4 Cursor

A cursor is a query which points towardsa a DB and can be used to fetch tubles.
DECLARE ¡name¿ CURSOR FOR SELECT ¡attributes¿ FROM ¡DB¿ WHERE ...

8.5 Loop

The loop in itself is really simple consisting of: ¡label¿: LOOP ... END LOOP; As seen there is no guard so therefore an if condition is done itself to leave the loop.
The most common use case is looping through a DB, to create a leave if a variable is used for when a cursor hits the end: DECLARE $\mathrm{Not}_{F}oundCONDITIONFORSQLSTAT$
Where 02000 is a SQL state representing null in a select.
So a loop which exit when a cursor hits the end is: DECLARE age INTEGER; ¡label¿: LOOP FETCH FROM ¡cursorVariable¿ INTO age; IF $\mathrm{Not}_{F}oundTHENLEAVE < label > ENDIF; ...ENDLOOP;$

8.6 For loops

For loops are used for cursors on tables. For loops can here help by declaring the cursor and the assigned attribute variables itself.
FOR ¡loop name¿ AS ¡cursor name¿ CURSOR FOR ¡query¿ DO ... END FOR; Here in the query the attributes listed after the select are the variable names used in the do.

8.7 Exception handlers

In case of an exception in a given block of code, an exception handler can be used.
A code block starts with BEGIN the code and then END.
A handler cosist of DECLARE <where to go> HANDLER FOR <condition list> <statement> The where to go can be 3 options which determine the outcome after executing the statement:
CONTINUE - continues execution after the line which gave exception
EXIT - exits the code block
UNDO - exits the code block and revert changes from code block in DB
An example for a handler can be: DECLARE Too_Many CONDITION FOR SQLTATE '21000'; BEGIN DELCARE EXIT HANDLER FOR Too_Many RETURN NULL; ... END;

9 Call-level Interface

When working with a DB, a high level language is used to connect and communicate.
A CLI here provides an envirement for the DB, a connection to the DB, and a way to create statements.

9.1 JDBC

JDBC is a Java CLI, to create a connection the following can be done:

```
import java.sql.*;

Class.forName("org.postgresql.Driver"); //For postsql envirement
String url = "jdbc:postgresql://" + host + "[:" + port + "]/" + databa

Connection myCon = DriverManager.getConnection(URL, username, password
Statement stat = myCon.createStatement("SELECT...");
ResultSet res = stat.executeQuery();
Statement stat2 = myCon.createStatement();
stat2.executeUpdate("INSERT...");
while(res.next()) {
   int resInt = rest.getInt(1); //if the tuble is (name, age, sex) it
   String resString = rest.getString(0);
}
```

DISCLAIMER:
Don't reuse statements due to closing.
Use myCon.setAutoCommit(false); such myCon.commit() and myCon.rollBack() can be used.