♦ 15 Database design interview Questions & Answers

Posted on August 19, 2014 by Arulkumaran Kumaraswamipillai — No Comments ↓

Q1. What is normalization? When to denormalize?

A1. **Normalization** is a design technique that is widely used as a guide in designing relational databases. Normalization is essentially a two step process that puts data into tabular form by removing repeating groups and then removes duplicated data from the relational tables.

Redundant data wastes disk space and creates maintenance problems. If data that exists in more than one place must be changed, the data must be changed in exactly the same way in all locations which is time consuming and prone to errors. A change to a customer address is much easier to do if that

data is stored only in the Customers table and nowhere else in the database.

Inconsistent dependency is a database design that makes certain assumptions about the location of data. For example, while it is intuitive for a user to look in the Customers table for the address of a particular customer, it may not make sense to look there for the salary of the employee who calls on that customer. The employee's salary is related to, or dependent on the employee and thus should be moved to the Employees table. Inconsistent dependencies can make data difficult to access because the path to find the data may not be logical, or may be missing or broken.

First Normal form	Second Normal form	Third Normal form
A database is said to be in First Normal Form when all entities have a unique identifier or key, and when every column in every table contains only a single value and doesn't contain a repeating group or composite field.	A database is in Second Normal Form when it is in First Normal Form plus every non-primary key column in the table must depend on the entire primary key, not just part of it, assuming that the primary key is made up of composite columns.	A database is in Third Normal Form when it is in Second Normal Form and each column that isn't part of the primary key doesn't depend on another column that isn't part of the primary key.

Q. When to denormalize? Normalize for accuracy and denormalize for performance.

Typically, transactional databases are highly normalized. This means that redundant data is eliminated and replaced with

Hadoop & BigData Int
Java Architecture Inte
Scala Interview Q&As
Spring, Hibernate, & M
Testing & Profiling/Sa
Other Interview Q&A t
Free Java Interviev

16 Technical Key Areas

open all | close all
Best Practice (6)
Coding (26)
Concurrency (6)
Design Concepts (7)
Design Patterns (11)
Exception Handling (3
Java Debugging (21)
Judging Experience In
Low Latency (7)
Memory Managemen
Performance (13)
QoS (8)
Scalability (4)
SDLC (6)
Security (13)
Transaction Managen

80+ step by step Java Tutorials

open all | close all
Setting up Tutorial (6)
Tutorial - Diagnosis (2
Akka Tutorial (9)
Core Java Tutorials (2
Hadoop & Spark Tuto

keys in a one-to-many relationship. Data that is highly normalized is constrained by the primary key/foreign key relationship, and thus has a high degree of data integrity. Denormalized data, on the other hand, creates redundancies; this means that it's possible for denormalized data to lose track of some of the relationships between atomic data items. However, since all the data for a query is (usually) stored in a single row in the table, it is much faster to retrieve.

Real life examples of denormalized tables

1) Datawarehouse make use of denormalized tables to produce complex multidimensional reports more effciently.
2) Staging tables in ETL (Extract Transform Load) operations are denormalized.

Q2. How do you implement one-to-one, one-to-many and many-to-many relationships while designing tables?
A2. **One-to-One** relationship can be implemented as a single table and rarely as two tables with primary and foreign key relationships.

One-to-Many relationships are implemented by splitting the data into two tables with primary key and foreign key relationships.

Many-to-Many relationships are implemented using join table with the keys from both the tables forming the composite primary key of the junction table.

Q3. What is an ER diagram?
A3. An ER diagram is also known as an Entity Relationship diagram, which is a graphical representation of entities and their relationships to each other, typically used in regard to the organization of data within databases.

Q4. How can you performance tune your database?
A4.

1) Denormalize your tables where appropriate.
2) Proper use of index columns: An index based on numeric fields is more efficient than an index based on character columns.
3) Reduce the number of columns that make up a composite key.
4) Proper partitioning of tablespaces and create a special tablespace for special data types like CLOB, BLOB etc.
5) Data access performance can be tuned by using stored procedures to crunch data in the database server to reduce the network overhead and also caching data within your application to reduce the number of accesses.

Q. Can you give some database performance tuning tips based on your experience?

1. Materialized views are one of the important SQL tuning tools in Oracle. Instead of the entire company accessing a single database server, user load can be distributed across multiple database servers with the help of materialized views in Oracle. Through the use of multi tier materialized views, you can create materialized views based on other materialized views, which enables you to distribute user load to an even greater extent because clients can access materialized view sites instead of master sites. To decrease the amount of data that is replicated, a materialized view can be a subset of a master table or master materialized view.

a) Materialized views are schema objects that can be used to summarize, precompute, replicate, and distribute data.
b) It allows you to pre-join complex views and pre-compute summaries for super-fast response times. Unlike an ordinary view, which does not take up any storage space or contain any data, a materialized view provides indirect access to table data by storing the results of a query in a separate schema object.
c) You can define a materialized view on a base table, partitioned table or view and you can define indexes on a materialized view.
d) A materialized view can be stored in the same database as

its base tables (improves query performance through query rewrite) or in a different database.

2. As a rule of thumb, every table should have a clustered index. Generally, but not always, the clustered index should be on a column that increases in one direction (i.e. monotonic) such as an identity column, or some other column where the value is increasing and is unique. In many cases, the primary key is the ideal column for a clustered index. Create an index on any column that is a foreign key. If you know it will be unique, set the flag to force the index to be unique.

3. Avoid temp tables as much as you can, but if you need a temp table, create it explicitly using Create Table #temp.

4. Only return the columns and the rows you need.

5. Avoid full table scan where possible. The full table scan can be caused by

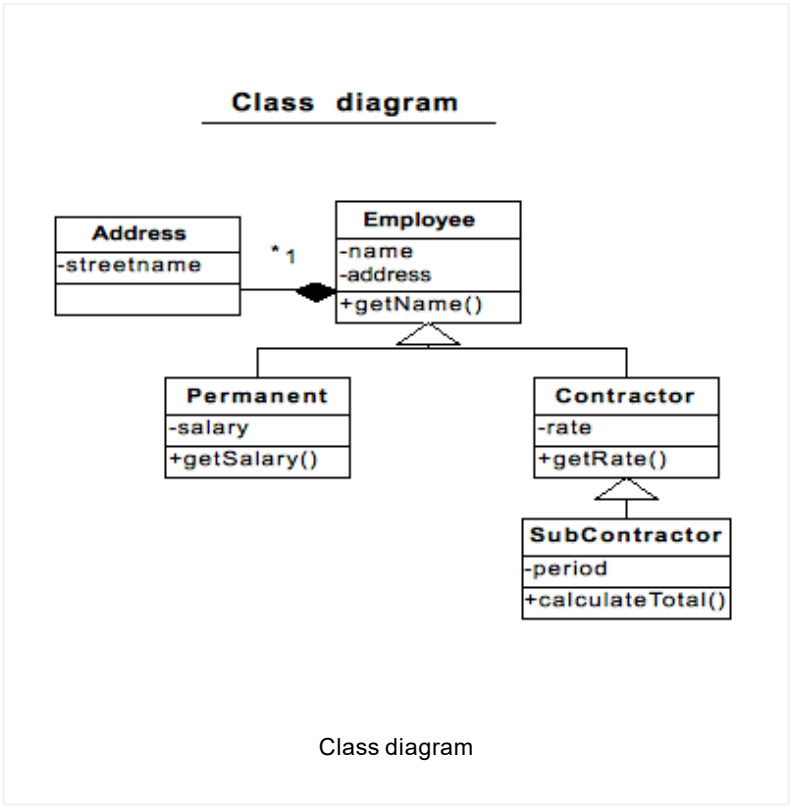
a) No WHERE condition.
b) No index on any type of field in the WHERE clause.
c) NOT IN predicate that is easier to write (replace NOT IN with a left outer join).
d) WHERE clauses like column_name is not null, condition 1 or condition 2, column_name between … and …, not equality comparisons
e) Use of SQL “LIKE clause” operator to find a string within a large table column (e.g. VARCHAR(2000), CLOB, BLOB). DISTINCT, ANY, and ALL.

It is also worth noting that this capability may not suit best for too frequent activities as in online transaction processing (i.e. OLTP) environments. In other databases equivalent functionalities can be achieved through triggers on base tables that would update/insert aggregate or dimension tables and queries can be executed against these aggregated or dimension tables as oppose to the base tables.

Q5. How will you map objects to a relational database? How will you map class inheritance to relational data model?

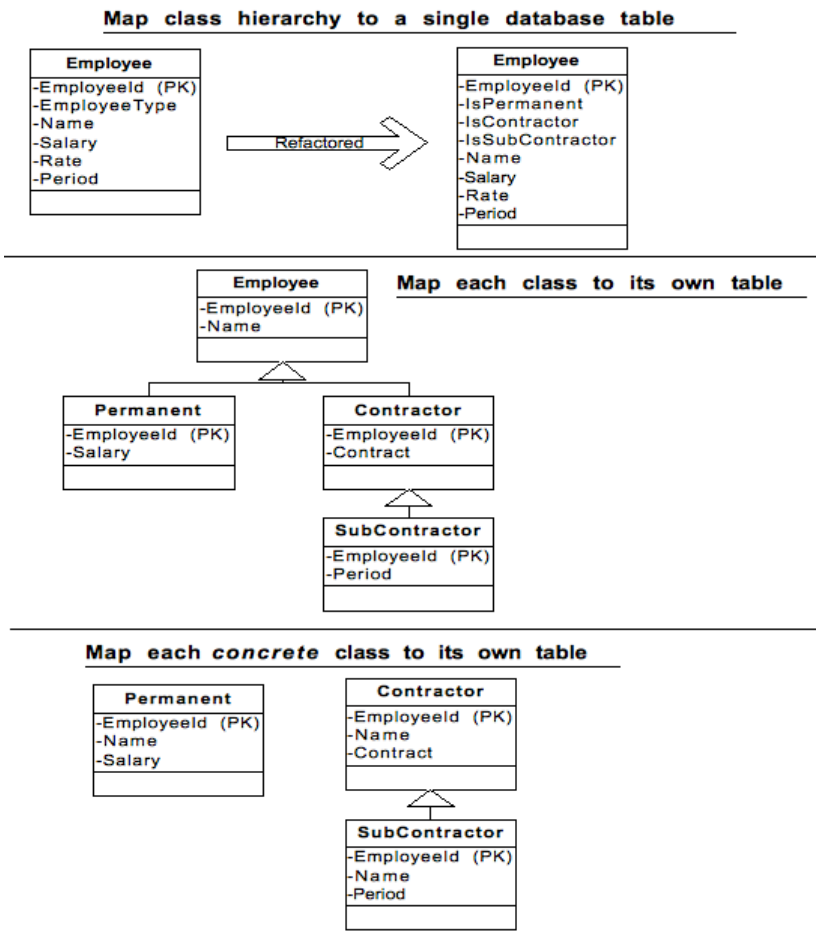
A5. Due to impedance mismatch between object and relational technology you need to understand the process of mapping classes (objects) and their relationships to tables and relationships between them in a database. Classes represent both behavior and data whereas relational database tables just implement data. Database schemas have keys (primary keys to uniquely identify rows and foreign keys to maintain relationships between rows) whereas object schema does not have keys and instead use references to implement relationships to other objects. Let us look at some basic points on mapping

Class diagram in OO world:



Class diagram

Relational mapping



Object to Relational Mapping (ORM)

1) Classes map to tables in a way but not always directly.
2) An attribute of a class can be mapped to zero or more columns in a database. Not all attributes need to be persistent.
3) Some attributes of an object are objects itself. For example an Employee object has an Address object as an attribute. This is basically an association relationship between two objects (i.e. Employee and Address).
4) In its simple form an attribute maps to a single column whereas each has same type (i.e. attribute is a string and column is a char, or both are dates etc). When you implement mapping with different types (attribute is a currency and column is a float) then you will need to be able to convert them back and forth.

Q6. How do you map inheritance class structure to relational data model?

A6. Relational databases do not support inheritance. Class inheritance can be mapped to relational tables as follows:

Map class hierarchy to single database table (aka union mapping): The whole class hierarchy can be stored in a single table by adding an additional column named “EmployeeType”. The column “EmployeeType” will hold the values “Permanent”, “Contract” and “SubContract”. New employee types can be added as required. Although this approach is straightforward it tends to break when you have combinations like an employee is of type both “Contractor” and “SubContractor”. So when you have combinations, you can use refactored table by replacing type code column “EmployeeType” with boolean values such as isPermanent, isContractor and isSubContractor.

Map each class to its own table (aka vertical mapping): You create one table per class (even those that are abstract). The data for a permanent employee is stored in two tables (Employee and Permanent), therefore to retrieve this data you need to join these two tables. To support additional employee type say a Contractor, add a new table.

Map each concrete class to its own table (aka horizontal mapping): You create one table per concrete class. There are tables corresponding to each class like Permanent, Contractor and SubContractor. So join is not required. To support additional employee type, add a new table.

So which approach to use? No approach is ideal for all situations. Each approach has its own pros & cons.

Map class hierarchy to single database table:
Advantages are: no table joins are necessary to query objects in the same hierarchy and adding a new class to the hierarchy has very little overhead. **Disadvantages are:** Database constraints have to be relaxed to accommodate all attributes in the class hierarchy and also it is not easy to identify from the table schema which attributes belong to which class.

Map each class to its own table: **Advantages are:** Table schemas are separated cleanly and database constraints can be applied. **Disadvantages are:** Suffers from performance problems. If you need to query all employees then all 4 tables (i.e. Employee, Permanent, Contractor & SubContractor) need to be queried.

Map each concrete class to its own table: Advantage is: simplest approach. Disadvantage is: duplicated base class columns in each subclass table making adding an attribute to the baseclass more difficult.

Finally, No approach is ideal for all situations. The most efficient way is to “map class hierarchy to single database table” (i.e. union mapping). For dealing with complex legacy data “use map each class to its own table” (i.e. vertical mapping) which gives you more flexibility but this flexibility comes at a price of performance. The simplest way to map is to use “map each concrete class to its own table” (i.e. horizontal mapping) but this simplicity comes at a price of creating a very unnatural object model.

Q7. What is a view? Why will you use a view?
A7. View is a pre-compiled SQL query, which is used to select data from one or more tables. A view is like a table but it doesn’t physically take any space (i.e. **non-materialized views**). Views are used for

1) Providing inherent security by exposing only the data that is needed to be shown to the end user.
2) Enabling re-use of SQL statements.
3) Allows changes to the underlying tables to be hidden from clients, aiding maintenance of the database schema (i.e. encapsulation).

Views with multiple joins and filters can dramatically degrade performance because **non-materialized** views contain no data and any retrieval needs to be processed. The solution for this is to use **materialized views** or create **de-normalized tables to store data**. This technique is quite handy in overnight batch processes where a large chunk of data needs

to be processed. Normalized data can be read and inserted into some temporary de-normalized table and processed with efficiency.

Q8. What's the difference between a primary key and a unique key?
A8. Both primary key and unique key enforce uniqueness of the column on which they are defined. But by default **primary key creates a clustered index on the column**, whereas **unique creates a non-clustered index by default**. Another major difference is that, primary key doesn't allow NULLs, but unique key allows one NULL only.

Q. What is the best practice relating to primary key generation?

1) A best practice in database design is to use an internally generated primary key. The database management system can normally generate a unique identifier that has no meaning outside of the database system. For example "Sequences" in Oracle and "Identity" columns in Sybase.

2) It is bad practice to use timestamps as a primary key or using it as part of your composite primary key because you can get a primary key collision when two concurrent users access the table within milliseconds.

3) For better performance minimize use of composite keys or use fewer columns in your composite keys.

4) Where possible avoid using columns with business meaning as your primary key. For example Avoid using taxfilenumber, zipcode etc as your primary key because more than one town may have the same zipcode, and taxfilenumber is private and should be encrypted and stored, and some people may not have a taxfile number, you may want to reuse the same taxfilenumber after an individual's death, an individual may have more than one taxfilenumber, etc.

5) Remember to choose carefully, as it is difficult to change the primary key in a production table.

Q9. What are constraints? Explain different types of constraints?
A9. Constraints enable the RDBMS (Relational DataBase Management Systems) to enforce the integrity of the database automatically, without needing you to create triggers, rule or defaults.

Types of constraints: NOT NULL, CHECK, UNIQUE, PRIMARY KEY, FOREIGN KEY, etc.

Q. What are the best practices relating to constraints?

1) Always define referential constraints to improve referential integrity of your data. For example, A "BankDetail" table can have BSB number and accountnumber as part of **unique key constraint** to prevent duplicate account details, while having a generated unique identifier as the primary key.

2) Perform all your referential integrity checks and data validations using constraints (foreign key and constraints) instead of triggers, as constraints are faster. Limit the use of triggers only for auditing, custom tasks and validations that can not be performed using constraints. Constraints save you time as well as you don't have to write code for these validations, allowing the RDBMS to do all the work for you.

Q10. What is an index? What are the types of indexes? How many clustered indexes can be created on a table? What are the advantages and disadvantages of creating separate index on each column of a table?
A10. The books you read have indexes, which help you to go to a specific key word faster. The database indexes are similar.

Indexes are of two types. **Clustered indexes** and **non-clustered indexes**. When you create a clustered index on a table, all the rows in the table are stored in the order of the clustered index key. So, there can be only one clustered

index per table. Non-clustered indexes have their own storage separate from the table data storage. The row located could be the RowID or the clustered index key, depending upon the absence or presence of clustered index on the table.

If you create an index on each column of a table, it improves the query (i.e. SELECT) performance, as the query optimizer can choose from all the existing indexes to come up with an efficient execution plan. At the same time, data modification operations such as INSERT, UPDATE, and DELETE will become slower, as every time data changes in the table, all the indexes need to be updated. Another disadvantage is that, indexes need disk space, the more indexes you have, more disk space is used. Now a days disk spaces are very cheap, hence does not matter much.

Q11. What is a database trigger?
A11. A trigger is a fragment of code that you tell to run before or after a table is modified.

There are typically three triggering EVENTS that cause trigger to 'fire':

INSERT event (as a new record is being inserted into the database).
UPDATE event (as a record is being changed).
DELETE event (as a record is being deleted).

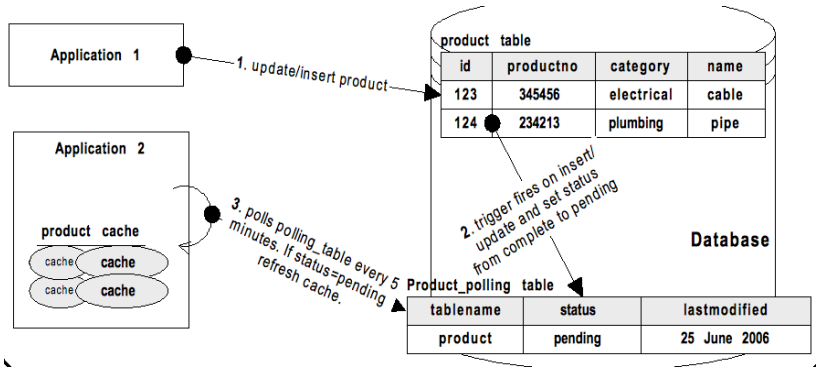
Triggers can restrict access to specific data, perform logging, or audit access to data.

Q. How can you keep track of all your database changes?

If you want to keep track of all changes to a particular record, such as who modified the record, what kind of modification took place, and when the record modification occurred then you can use triggers because you can capture every action that occurred on a particular table. For example, an INSERT trigger would fire when a particular database table has a record inserted.

Q. How will you communicate between two applications sharing the same database to update one of the applications' object cache?

As shown in the diagram below,



triggers in action

1. When application 1 updates/inserts a product in the "product" table,
2. A trigger is fired to modify the status of the "product_polling" table to "pending" from "complete" for updates, and for inserts creates a new record with a pending status.
3. Application 2 polls the "product_polling" table say every 5 minutes. If the status="pending" then application 2 reads the updated data from the product table and refreshes the cache. If the status is "complete" then the application 2 retries after 5 minutes.

Q12. When to not use a trigger, and when is it appropriate to use a trigger?
A12. The database triggers need to be used very judiciously as they are executed every time an event like insert, update or delete occur.

When to not use a trigger? Don't use a trigger where

1) database constraints like unique constraint, not null, primary key, check constraints, etc can be used to check for

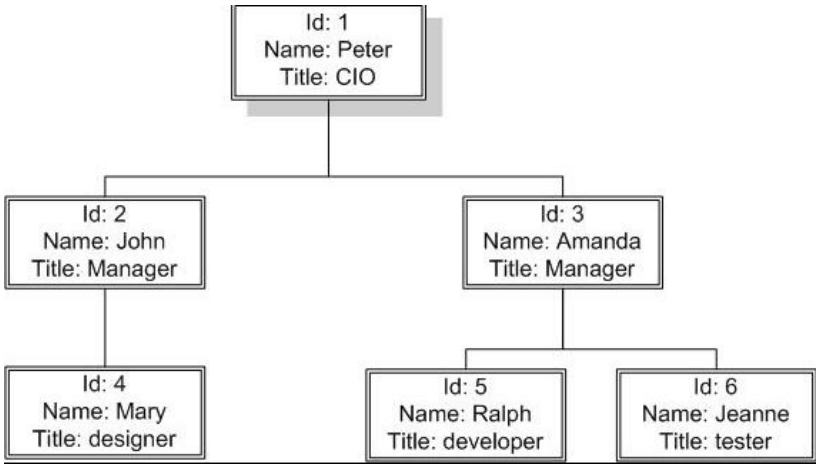
data validity.
2) triggers are recursive.

Where to use a trigger?

1) Maintaining complex integrity constraints (i.e. referential integrity) or business rules where other types of constraints cannot be used. Because triggers are executed as part of the SQL statement within its containing transaction causing the row change event, and the trigger code has direct access to the changed row, you could in theory use them to correct or reject invalid data.
2) Auditing information in a table by recording the changes. Some tables are required to be audited as part of the non-functional requirement for changes.
3) Automatically signaling other programs that action needs to take place when changes are made to a table.
4) Collecting and maintaining aggregate or statistical data.

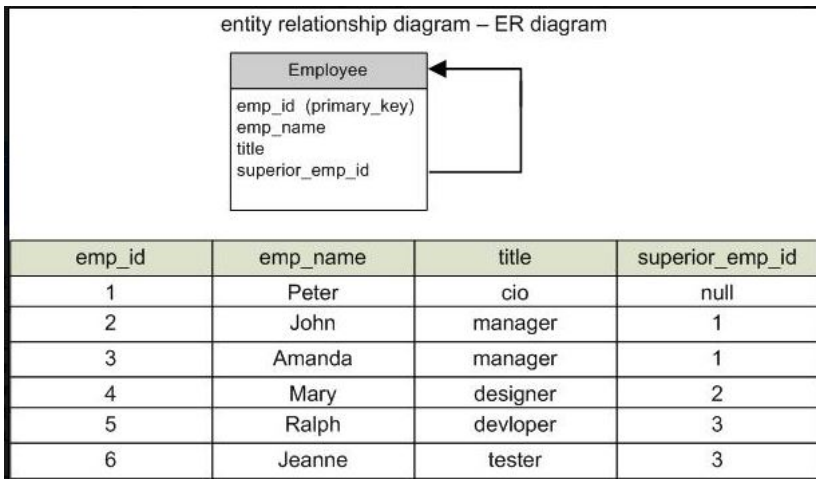
Q13. How would you go about copying bulk data in and out of a database?
A13. The process is known as bulk copy, and the tools used for this are database specific. For example, in Sybase and SQLServer use a utility called “bcp”, which allows you to export bulk data into comma delimited files, and then import the data in csv or any other delimited formats back into different database or table. In Oracle database, you achieve this via the SQLLoader. The DB2 database has IMPORT and LOAD command to achieve the same.

Q14. How will you represent a hierarchical structure shown below in a relational database? or How will you store a tree data structure into DB tables?



SQL hierarchical data

A14. The hierarchical data is an example of the composite design pattern. The entity relationship diagrams (aka **ER diagram**) are used to represent logical and physical relationships between the database tables. The diagram below shows how the table can be designed to store tree data by maintaining the adjacency information via superior_emp_id.



emp_id	emp_name	title	superior_emp_id
1	Peter	cio	null
2	John	manager	1
3	Amanda	manager	1
4	Mary	designer	2
5	Ralph	devloper	3
6	Jeanne	tester	3

ER Diagram for Hierarchical data

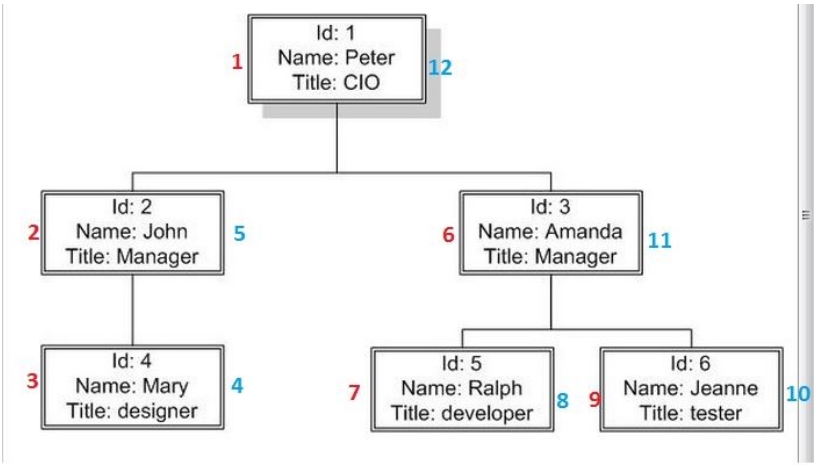
as you can see the “superior_emp_id” is a foreign key that points to the emp_id in the same table. So, Peter has null as he has no superiors. John and Amanda points to Peter who is their manager or superior and so on.

The above table can be created using SQL DDL (Data Definition Language) as shown below.

```
1 CREATE TABLE employee (
2 emp_id          NUMBER (4) CONSTRAINT emp_pk PRIM
3 emp_name        VARCHAR2 (40) NOT NULL,
4 title           VARCHAR2 (40),
5 dept_id         NUMBER (2) NOT NULL,
6 superior_emp_id NUMBER (4) CONSTRAINT emp_fk REFE
7 CONSTRAINT emp_pk
8 PRIMARY KEY NONCLUSTERED (emp_id))
9
```

Q15. Is there any other way to to store tree structure in a relational database?

A15. Yes, it can be done using the “modified preorder tree traversal” as described below.



Pre-order traversal

As shown in the diagram above, each node is marked with a left and right numbers using a modified preorder traversal as shown above. This can be represented in a database table as shown below.

emp_id	emp_name	title	left_val	right_val
1	Peter	cio	1	12
2	John	manager	2	5
3	Amanda	manager	6	11
4	Mary	designer	3	4
5	Ralph	developer	7	8
6	Jeanne	tester	9	10

As you can see the numbers indicate the relationship between each node. All left values greater than 6 and right values less than 11 are descendants of 6-11 (i.e Id: 3 Amanda). Now if you want to extract out the 2-6 sub-tree for Amanda you can write the SQL as follows

```
1 SELECT * FROM employee WHERE left_val BETWEEN 6 a
2
```

Which will return Amanda, Ralph, and Jeanne.

If you want to get ancestors to a given node say 7-8 Ralph, you can write the SQL as follows

```
1 SELECT * FROM employee WHERE left_val < 7 and rig
2
```

Which will return: Peter and Amanda.

If you want to find out the **number of descendants for a node**, all you need is the left_val and right_val of the node for which you want to find the descendants count. The formula is

No. of descendants = (right_val – left_val -1) /2

So, for 6 -11 Amanda, (11 – 6 – 1) /2 = 2 descendants
for 1-12 Peter, (12 – 1 -1) / 2 = 5 descendants.
for 3-4 Mary, (4 -3 – 1) / 2 = 0, means it is a child and has no descendants.

Popular Posts

♦ 11 Spring boot interview questions & answers

828 views

♦ Q11-Q23: Top 50+ Core on Java OOP Interview Questions & Answers

768 views

18 Java scenarios based interview Questions and Answers

400 views

001A: ♦ 7+ Java integration styles & patterns interview questions & answers

389 views

01b: ♦ 13 Spring basics Q8 – Q13 interview questions & answers

296 views

♦ 7 Java debugging interview questions & answers

293 views

01: ♦ 15 Ice breaker questions asked 90% of the time in Java job interviews with hints

286 views

♦ 10 ERD (Entity-Relationship Diagrams) Interview Questions and Answers

280 views

♦ Q24-Q36: Top 50+ Core on Java classes, interfaces and generics interview questions & answers

240 views

001B: ♦ Java architecture & design concepts interview questions & answers

202 views

Bio | Latest Posts

Arulkumaran Kumaraswamipillai

Mechanical Eng to freelance Java developer in 3 yrs. Contracting since 2003, and attended 150+ Java job interviews, and often got 4 - 7 job offers to choose from. It pays to prepare. So, published Java interview Q&A books via Amazon.com in 2005, and sold 35,000+ copies. Books are outdated and replaced with this subscription based site.**945+** paid members. join my LinkedIn Group. **Reviews**

About Arulkumaran Kumaraswamipillai

Mechanical Eng to freelance Java developer in 3 yrs. Contracting since 2003, and attended 150+ Java job

interviews, and often got 4 - 7 job offers to choose from. It pays to prepare. So, published Java interview Q&A books via Amazon.com in 2005, and sold 35,000+ copies. Books are outdated and replaced with this subscription based site.**945+** paid members. join my LinkedIn Group. **Reviews**

‹ 01: ♦ ♥ 17 Java overview interview questions and answers

♦ 14+ SQL interview Questions & Answers ›

Posted in member-paid, SQL

Tags: Java/JEE FAQs, JEE FAQs, Novice FAQs

Leave a Reply

Logged in as geethika. Log out?

Comment

Post Comment

Empowers you to open more doors, and fast-track

Technical Know Hows

☀ Java generics in no time ☀ Top 6 tips to transforming your thinking from OOP to FP ☀ How does a HashMap internally work? What is a hashing function?
☀ 10+ Java String class interview Q&As ☀ Java auto un/boxing benefits & caveats ☀ Top 11 slacknesses that can come back and bite you as an experienced Java developer or architect

Non-Technical Know Hows

☀ 6 Aspects that can motivate you to fast-track your career & go places ☀ Are you reinventing yourself as a Java developer? ☀ 8 tips to safeguard your Java career against offshoring ☀ My top 5 career mistakes

Prepare to succeed

☀ Turn readers of your Java CV go from “Blah blah” to “Wow”? ☀ How to prepare for Java job interviews? ☀ 16 Technical Key Areas ☀ How to choose from multiple Java job offers?

Select Category ▼

© Disclaimer

 ↑