**DATA MODELLING**

<http://www.1keydata.com/datawarehousing/concepts.html>

**Q1: What is Data Modeling?**

A data model documents and organizes data, how it is stored and accessed, and the relationships among different types of data. The model may be abstract or concrete.

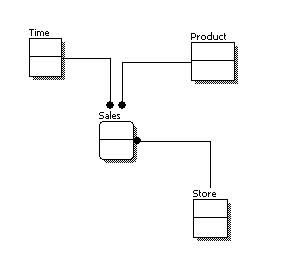
Defn: Data modeling is a [process](http://en.wikipedia.org/wiki/Software_development_process) used to define and analyze data [requirements](http://en.wikipedia.org/wiki/Requirement) needed to support the [business processes](http://en.wikipedia.org/wiki/Business_process) within the scope of corresponding information systems in organizations

**Q2: Explain different data models.**

**Dimensional Data Model**

**Conceptual Data Model**A conceptual data model identifies the highest-level relationships between the different entities. Features of conceptual data model include:

* Includes the important entities and the relationships among them.
* No attribute is specified.
* No primary key is specified.

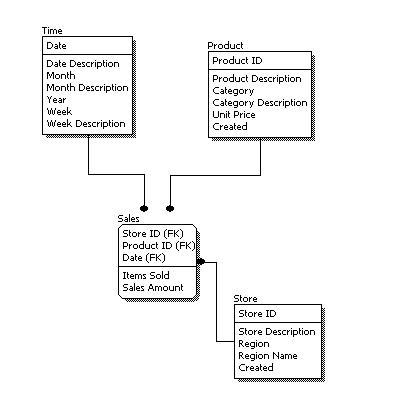


**Logical Data Model**A logical data model describes the data in as much detail as possible, without regard to how they will be physical implemented in the database. Features of a logical data model include:

* Includes all entities and relationships among them.
* All attributes for each entity are specified.
* The primary key for each entity is specified.
* Foreign keys (keys identifying the relationship between different entities) are specified.
* Normalization occurs at this level.

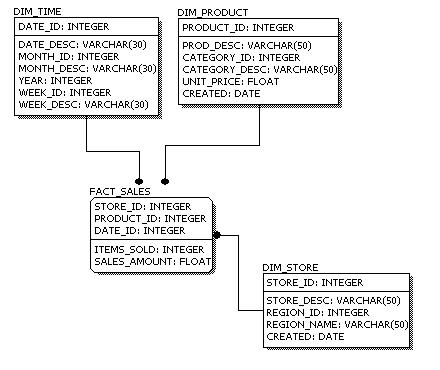
The steps for designing the logical data model are as follows:

1. Specify primary keys for all entities.
2. Find the relationships between different entities.
3. Find all attributes for each entity.
4. Resolve many-to-many relationships.
5. Normalization.

The figure below is an example of a logical data model.  


**Physical Data Model**Physical data model represents how the model will be built in the database. A physical database model shows all table structures, including column name, column data type, column constraints, primary key, foreign key, and relationships between tables. Features of a physical data model include:

* Specification all tables and columns.
* Foreign keys are used to identify relationships between tables.
* Denormalization may occur based on user requirements.
* Physical considerations may cause the physical data model to be quite different from the logical data model.
* Physical data model will be different for different RDBMS. For example, data type for a column may be different between MySQL and SQL Server.



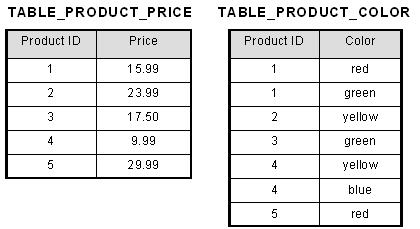
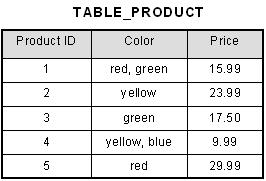
**Q10: Database Normalization**

Is the process of organizing the [fields](http://en.wikipedia.org/wiki/Field_(computer_science)) and [tables](http://en.wikipedia.org/wiki/Table_(database)) of a [relational database](http://en.wikipedia.org/wiki/Relational_database) to minimize [redundancy](http://en.wikipedia.org/wiki/Data_redundancy) and dependency. Normalization usually involves dividing large tables into smaller (and less redundant) tables and defining relationships between them. The objective is to isolate data so that additions, deletions, and modifications of a field can be made in just one table and then propagated through the rest of the database using the defined relationships.

|  |  |
| --- | --- |
| Level | Rule |
| [First normal form (1NF)](http://agiledata.org/essays/dataNormalization.html#1NF) | An entity type is in 1NF when it contains no repeating groups of data. i.e. No repeating columns |
| [Second normal form (2NF)](http://agiledata.org/essays/dataNormalization.html#2NF) | An entity type is in 2NF when it is in 1NF and when all of its non-key attributes are fully dependent on its [primary key](http://www.agiledata.org/essays/keys.html). |
| [Third normal form (3NF)](http://agiledata.org/essays/dataNormalization.html#3NF) | An entity type is in 3NF when it is in 2NF and when all of its attributes are directly dependent on the [primary key](http://www.agiledata.org/essays/keys.html). |

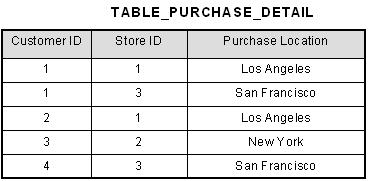
**First NF**:

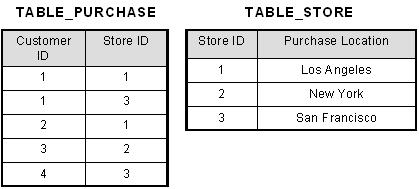
* There are no repeating or duplicate fields
* Each cell contains only a single value
* Each record in Unique identified by Promary Key



**Second NF**:

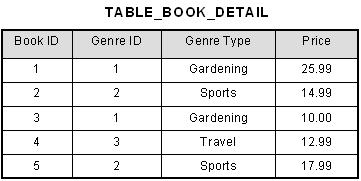
* It is in first normal form
* All non-key attributes are fully functional dependent on the primary key

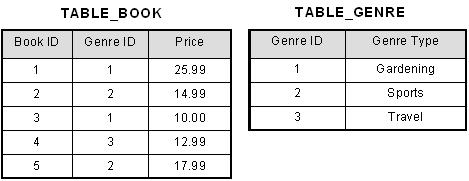




**Third NF**:

* It is in second normal form
* There is no transitive functional dependency





**Common SQL Queries**

**Question 1: SQL Query to find second highest salary of Employee**

Answer : There are many ways to find second highest salary of Employee in SQL, you can either use SQL Join or Subquery to solve this problem. Here is SQL query using Subquery :

select MAX(Salary) from Employee WHERE Salary NOT IN (select MAX(Salary) from Employee );

**Question 2: SQL Query to find Max Salary from each department.**

Answer : You can find maximum salary for each department by grouping all records by DeptId and then using MAX() function to calculate maximum salary in each group or each department.

SELECT DeptID, MAX(Salary) FROM Employee  GROUP BY DeptID.

This questions become more interesting if Interviewer will ask you to print department name instead of department id, in that case you need to join Employee table with Department using foreign key DeptID, make sure you do LEFT or RIGHT OUTER JOIN to include departments without any employee as well.  Here is the query

SELECT DeptName, MAX(Salary) FROM Employee e RIGHT JOIN Department d ON e.DeptId = d.DeptID GROUP BY DeptName;

In this query we have use RIGHT OUTER JOIN because we need name of department from Department table which is on right side of JOIN clause, even if there is no reference of dept\_id on Employee table.

**Question 11 : Write SQL Query to find duplicate rows in a database? and then write SQL query to delete them?**Answer : You can use following query to select distinct records :

SELECT \* FROM emp a WHERE rowid = (SELECT MAX(rowid) FROM EMP b WHERE a.empno=b.empno)

To Delete:

DELETE FROM emp a WHERE rowid != (SELECT MAX(rowid) FROM emp b WHERE a.empno=b.empno);

**N maximum salary:**

SELECT MAX(EmpSalary)

FROM Salary

WHERE EmpSalary IN(SELECT TOP N EmpSalary FROM Salary ORDER BY EmpSalary ASC)

**N minimum salary:**

SELECT MIN(EmpSalary)

FROM Salary

WHERE EmpSalary IN(SELECT TOP N EmpSalary FROM Salary ORDER BY EmpSalary DESC)

SELECT MIN(EmpSalary) from (

SELECT EmpSalary from Employee ORDER BY EmpSalary DESC LIMIT 3

);

Select EmpName,salary from

(

select EmpName,salary ,Row\_Number() over(order by salary desc) as rowid

from EmpTbl)

as a where rowid=3

Read more: <http://java67.blogspot.com/2013/04/10-frequently-asked-sql-query-interview-questions-answers-database.html#ixzz3j3tRkteQ>

**Delete duplicate values from single column table.**

CREATE TABLE #t1

(

id int

);

INSERT INTO #t1

(

id

)

VALUES

(1),

(2),

(3),

(3),

(3),

(4)

with CTE as (

select rowid,id,row\_number() over (partition by id order by rowid) as newrowid from (

select row\_number() over (order by id) as rowid, id from #t1) a

)

select rowid,id,newrowid from CTE a where newrowid >1

**Alternate :**

WITH CTE AS (

SELECT ID,ROW\_NUMBER() OVER (ORDER BY ID) ROWNUM FROM #T1

)

SELECT ID FROM CTE A

WHERE ROWNUM != (SELECT MAX(ROWNUM) FROM CTE B WHERE A.ID = B.ID)

**Alternate:**

delete a1 from (

select row\_number() over(partition by id order by rownum) ord,id,rownum from (

select row\_number() over(order by id) as rownum,id from #t1) a

) a1 where a1.ord>1

**Cricket Schedule combination:**

DROP TABLE #T1

CREATE TABLE #t1

(

id varchar(10)

);

INSERT INTO #t1

(

id

)

VALUES

('c1'),

('c2'),

('c3'),

('c5'),

('c4')

SELECT \* FROM #T1 A,#T1 B

WHERE A.ID <>B.ID AND A.ID>B.ID ORDER BY 1,2;

**Generating rownum without rank:**

**Name is unique field**

SELECT name, sal, (SELECT COUNT(\*) FROM EMPLOYEE i WHERE o.name >= i.name) row\_num

FROM EMPLOYEE o

order by row\_num

**Correlated Subquery to generate running total :**

select Qty,

(select sum(Qty) from t where Qty<=t1.Qty)

from t t1 order by Qty

**Find consequitve values same**

DROP TABLE #cricket

CREATE TABLE #cricket

(overnum NUMERIC(4) NOT NULL,

ballno VARCHAR(10),

run int

)

INSERT INTO #cricket VALUES

(1, 1, 0),

(1, 2, 1),

(1, 3, 1),

(1, 4, 0),

(1, 5, 0),

(1, 6, 1),

(2, 1, 0),

(2, 2, 1),

(2, 3, 1),

(2, 4, 1),

(2, 5, 0),

(2, 6, 1)

with CTE as (

select overnum,ballno,RUN from #cricket

)

SELECT A.OVERNUM FROM #cricket A JOIN #cricket B ON A.overnum = B.overnum AND A.ballno = B.ballno+1

WHERE A.run =0 AND B.RUN=0

**Generate rownumber without rownum**

SELECT name, sal, (SELECT COUNT(\*) FROM EMPLOYEE i WHERE o.name >= i.name) row\_num

FROM EMPLOYEE o

order by row\_num

DROP TABLE #EMP

CREATE TABLE #EMP

(EMPNO NUMERIC(4) NOT NULL,

ENAME VARCHAR(10),

JOB VARCHAR(9),

MGR NUMERIC(4),

HIREDATE DATETIME,

SAL NUMERIC(7, 2),

COMM NUMERIC(7, 2),

DEPTNO NUMERIC(2))

INSERT INTO #EMP VALUES

(7369, 'SMITH', 'CLERK', 7902, '17-DEC-1980', 800, NULL, 20),

(7499, 'ALLEN', 'SALESMAN', 7698, '20-FEB-1981', 1600, 300, 30),

(7521, 'WARD', 'SALESMAN', 7698, '22-FEB-1981', 1250, 500, 30),

(7566, 'JONES', 'MANAGER', 7839, '2-APR-1981', 2975, NULL, 20),

(7654, 'MARTIN', 'SALESMAN', 7698, '28-SEP-1981', 1250, 1400, 30),

(7698, 'BLAKE', 'MANAGER', 7839, '1-MAY-1981', 2850, NULL, 30),

(7782, 'CLARK', 'MANAGER', 7839, '9-JUN-1981', 2450, NULL, 10),

(7788, 'SCOTT', 'ANALYST', 7566, '09-DEC-1982', 3000, NULL, 20),

(7839, 'KING', 'PRESIDENT', NULL, '17-NOV-1981', 5000, NULL, 10),

(7844, 'TURNER', 'SALESMAN', 7698, '8-SEP-1981', 1500, 0, 30),

(7876, 'ADAMS', 'CLERK', 7788, '12-JAN-1983', 1100, NULL, 20),

(7900, 'JAMES', 'CLERK', 7698, '3-DEC-1981', 950, NULL, 30),

(7902, 'FORD', 'ANALYST', 7566, '3-DEC-1981', 3000, NULL, 20),

(7934, 'MILLER', 'CLERK', 7782, '23-JAN-1982', 1300, NULL, 10)

DROP TABLE #DEPT

CREATE TABLE #DEPT

(DEPTNO NUMERIC(2),

DNAME VARCHAR(14),

LOC VARCHAR(13) )

INSERT INTO #DEPT VALUES (10, 'ACCOUNTING', 'NEW YORK')

INSERT INTO #DEPT VALUES (20, 'RESEARCH', 'DALLAS')

INSERT INTO #DEPT VALUES (30, 'SALES', 'CHICAGO')

INSERT INTO #DEPT VALUES (40, 'OPERATIONS', 'BOSTON')

--Pull nth highest without subquery and top word

SELECT EMPNO,ename,sal FROM #EMP ORDER BY SAL DESC OFFSET 2 ROWS fetch next 1 rows only

--Max salary in each dept

SELECT DEPTNO,MAX(SAL) FROM #EMP GROUP BY DEPTNO

--2nd highest salary employees

SELECT \* FROM #EMP A JOIN (SELECT MAX(SAL) AS SAL FROM #EMP WHERE SAL NOT IN (SELECT MAX(SAL) FROM #EMP )) B ON A.SAL = B.SAL

--2nd highest salary employees from each dept

SELECT \* FROM #EMP A JOIN(

SELECT DEPTNO,SAL FROM (SELECT DEPTNO,SAL,DENSE\_RANK() OVER (PARTITION BY DEPTNO ORDER BY SAL DESC) AS ROWNUM FROM #EMP ) A WHERE ROWNUM = 2

) B ON A.SAL = B.SAL

--Nth highest salary employees from each dept with rank function

SELECT DEPTNO,SAL FROM (SELECT DEPTNO,SAL,DENSE\_RANK() OVER (PARTITION BY DEPTNO ORDER BY SAL DESC) AS ROWNUM FROM #EMP ) A WHERE ROWNUM = 2 --N

--Nth highest salary employees from each dept correlated query

select \* from #emp a where

3 = (select count(1) from #emp b where b.sal> a.sal)

--Difference of salary between employee and ranked employe below him

WITH CTE AS (

SELECT SAL,DENSE\_RANK() OVER (ORDER BY SAL DESC) AS ROWNUM FROM #EMP

)

SELECT A.SAL,B.SAL,B.SAL - A.SAL FROM CTE A JOIN CTE B ON A.ROWNUM = B.ROWNUM+1

--Fetch alternate record

SELECT \* FROM (SELECT \*,ROW\_NUMBER() OVER (ORDER BY SAL DESC) AS ROWNUM FROM #EMP) A WHERE ROWNUM %2 <> 0

--list dept only if has 4 or less than 4 employees

SELECT DEPTNO,COUNT(DISTINCT EMPNO) FROM #EMP A GROUP BY DEPTNO HAVING COUNT(DISTINCT EMPNO) <=4

--Count employee dept wise

SELECT COUNT(EMPNO),DEPTNO FROM #EMP GROUP BY DEPTNO

--Select all the employee group by deptno and sal in descending order.

SELECT SUM(SAL),DEPTNO FROM #EMP GROUP BY DEPTNO ORDER BY SUM(SAL) DESC

--running total sal of all employees

WITH CTE AS (

SELECT SAL,DENSE\_RANK() OVER (ORDER BY SAL DESC) AS ROWNUM FROM #EMP

)

SELECT (select sum(sal) from CTE b where a.ROWNUM<=b.ROWNUM),\* FROM CTE a

--running total sal of all employees by dept

WITH CTE AS (

SELECT deptno,SAL,DENSE\_RANK() OVER (partition by deptno ORDER BY SAL) AS ROWNUM FROM #EMP

)

SELECT (select sum(sal) from CTE b where a.ROWNUM<=b.ROWNUM and a.deptno = b.deptno group by deptno ),\* FROM CTE a

order by deptno,ROWNUM desc

--list the employees who are not managers

--An in statement will be parsed identically to field=val1 or field=val2 or field=val3. Putting a null in there will boil down to field=null which won't work.

select \* from #emp where empno not in (select distinct mgr from #emp where mgr is not null);

--manager with only one reportee

select mgr,count(empno) from #emp where mgr is not null group by mgr having count(empno) = 1

--Most hiring month

select top 1 month(HIREDATE),count(empno) from #emp where mgr is not null group by month(HIREDATE) order by count(empno) desc

--Name of manager with most reportees

select top 1 mgr,count(empno) from #emp where mgr is not null group by mgr order by count(empno) desc

--list managers who joined after the reportees

select a.empno as mgrno,a.hiredate as mgrhiredate,a.job,b.empno ,b.hiredate ,b.JOB from #EMP a right join #EMP b on a.empno = b.MGR

where a.hiredate>b.hiredate

--department with most managers and how many?

select deptno,count(distinct mgr) from #emp group by deptno;

--Print Employee with their level

with CTE as

(

select empno,mgr,ENAME,1 as level from #emp

where empno = '7839'

union all

select child.empno,child.mgr,child.ENAME,level + 1 from #emp child

join CTE parent

on child.mgr = parent.empno

)

select \*

from CTE

--hierarchial query:

drop table tbl;

create TABLE tbl(

id INT

, name VARCHAR(20)

, parent\_id INT

) ;

INSERT INTO tbl( id, name, parent\_id ) VALUES (1, 'Europe', NULL) ;

INSERT INTO tbl( id, name, parent\_id ) VALUES (2, 'Asia', NULL) ;

INSERT INTO tbl( id, name, parent\_id ) VALUES (3, 'Germany', 1) ;

INSERT INTO tbl( id, name, parent\_id ) VALUES (4, 'UK', 1) ;

INSERT INTO tbl( id, name, parent\_id ) VALUES (5, 'China', 2) ;

INSERT INTO tbl( id, name, parent\_id ) VALUES (6, 'India', 2) ;

INSERT INTO tbl( id, name, parent\_id ) VALUES (7, 'Scotland', 4) ;

INSERT INTO tbl( id, name, parent\_id ) VALUES (8, 'Edinburgh', 7) ;

INSERT INTO tbl( id, name, parent\_id ) VALUES (9, 'Leith', 8) ;

WITH CTE1 (id,parent\_id,name) AS

(

select id,parent\_id,name as path from tbl where parent\_id is null

union all

select t1.id,t1.parent\_id,t2.name||'-'||t1.name as path from tbl t1,CTE1 t2 where t2.id = t1.parent\_id

)

select id,name from cte1;

--Recursive CTE

WITH CTE1 (id) AS

(

select 1

union all

select id+1 from cte1 where id <5

)

select id from cte1;