DBMS

Last Minute Notes – DBMS

See Last Minute Notes on all subjects [here](http://quiz.geeksforgeeks.org/lmns/).

We will discuss the important key points useful for GATE exams in summarized form. For details you may refer [this](http://quiz.geeksforgeeks.org/gate-cs-notes/).

**Doubt what is er**

[**E-R Diagram**](https://www.geeksforgeeks.org/database-management-system-er-model/): The most common asked questions in ER diagram is minimum number of tables required for a given ER diagram. Generally, following criteria are used:

**forget**

|  |  |
| --- | --- |
| **Cardinality** | **Minimum No. of tables** |
| 1:1 cardinality with partial participation of both entities | 2 |
| 1:1 cardinality with total participation of atleast 1 entity | 1 |
| 1:n cardinality | 2 |
| m:n cardinality | 3 |

**Note:** This is a general observation. Special cases need to be taken care. We may need extra table if attribute of a relationship can’t be moved to any entity side.

**Cardinality** in context of **DBMS** refers to maximum number of relationship in which an entity can participate. ... For example, in a **database** table that stores bank account numbers, the “Account Number” column should have very high **cardinality** – by definition, every item of data in this column should be totally unique.Jul 29, 2015

[**Keys of a relation**](https://www.geeksforgeeks.org/dbms-keys-candidate-super-primary-alternate-and-foreign/): There are various types of keys in a relation which are:

* **Candidate Key:**The minimal set of attributes which can determine a tuple uniquely. There can be more than 1 candidate key of a relation and its proper subset can’t determine tuple uniquely and it can’t be NULL.
* **Super Key:**The set of attributes which can determine a tuple uniquely. A candidate key is always a super key but vice versa is not true.
* **Primary Key and Alternate Key:** Among various candidate keys, one key is taken primary key and others are alternate keys.
* **Foreign Key:** Foreign Key is a set of attributes in a table which is used to refer the primary key or alternative key of the same or other table.

[**Normal Forms**](https://www.geeksforgeeks.org/database-normalization-normal-forms/)

**Normalization** is the process of minimizing **redundancy** from a relation or set of relations. Redundancy in relation may cause insertion, deletion and updation anomalies. So, it helps to minimize the redundancy in relations. **Normal forms** are used to eliminate or reduce redundancy in database tables.

* **First Normal Form:**A relation is in first normal form if it does not contain any multi-valued or composite attribute.
* **Second Normal Form:**A relation is in second normal form if it does not contain any partial dependency. A dependency is called partial dependency if any proper subset of candidate key determines non-prime (which are not part of candidate key) attribute.
* **Third Normal Form:**A relation is in third normal form if it does not contain any transitive dependency. For a relation to be in Third Normal Form, either LHS of FD should be super key or RHS should be prime attribute.
* **Boyce-Codd Normal Form:**A relation is inBoyce-CoddNormal Form if LHS of every FD is super key. The relationship between Normal Forms can be represented as: **1NF⊃2NF ⊃3NF ⊃BCNF**

(\*\*)

**Denormalization** is a database optimization technique in which we add redundant data to one or more tables. This can help us avoid costly joins in a relational database. Note that denormalization does not mean not doing normalization. It is an optimization technique that is applied after doing normalization.

In a traditional normalized database, we store data in separate logical tables and attempt to minimize redundant data. We may strive to have only one copy of each piece of data in database.  
For example, in a normalized database, we might have a Courses table and a Teachers table.Each entry in Courses would store the teacherID for a Course but not the teacherName. When we need to retrieve a list of all Courses with the Teacher name, we would do a join between these two tables.  
In some ways, this is great; if a teacher changes is or her name, we only have to update the name in one place.  
The drawback is that if tables are large, we may spend an unnecessarily long time doing joins on tables.  
Denormalization, then, strikes a different compromise. Under denormalization, we decide that we’re okay with some redundancy and some extra effort to update the database in order to get the efficiency advantages of fewer joins.

[**Relational Algebra**](https://www.geeksforgeeks.org/database-management-system-relational-algebra/): Procedural language with basic and extended operators.

|  |  |
| --- | --- |
| [**Basic Operator**](https://www.geeksforgeeks.org/basic-operators-in-relational-algebra-2/) | **Semantic** |
| **σ(Selection)** | Select rows based on given condition |
| **∏(Projection)** | Project some columns |
| **X (Cross Product)** | Cross product of relations, returns **m\*n** rows where m and n are number of rows in R1 and R2 respectively. |
| **U (Union)** | Return those tuples which are either in R1 or in R2. Max no. of rows returned **= m+n**andMin no. of rows returned = **max(m,n)** |
| **−(Minus)** | R1-R2 returns those tuples which are in R1 but not in R2. Max no. of rows returned =**m**and Min no. of rows returned = **m-n** |
| **ρ(Rename)** | Renaming a relation to other relation. |

|  |  |
| --- | --- |
| [**Extended Operator**](https://www.geeksforgeeks.org/extended-operators-in-relational-algebra/) | **Semantic** |
| **∩ (Intersection)** | Returns those tuples which are in both R1 and R2. Max no. of rows returned = min(m,n) and Min no. of rows returned = 0 |
| **⋈c(Conditional Join)** | Selection from two or more tables based on some condition (Cross product followed by selection) |
| **⋈(Equi Join)** | It is a special case of conditional join when only equality condition is applied between attributes. |
| **⋈(Natural Join)** | In natural join, equality condition on common attributes hold and duplicate attributes are removed by default. **Note:** Natural Join is equivalent to cross product if two relations have no attribute in common and natural join of a relation R with itself will return R only. |
| **⟕(Left Outer Join)** | When applying join on two relations R and S, some tuples of R or S does not appear in result set which does not satisfy the join conditions. But Left Outer Joins gives all tuples of R in the result set. The tuples of R which do not satisfy join condition will have values as NULL for attributes of S. |
| **⟖(Right Outer Join)** | When applying join on two relations R and S, some tuples of R or S does not appear in result set which does not satisfy the join conditions. But Right Outer Joins gives all tuples of S in the result set. The tuples of S which do not satisfy join condition will have values as NULL for attributes of R. |
| **⟗(Full Outer Join)** | When applying join on two relations R and S, some tuples of R or S does not appear in result set which does not satisfy the join conditions. But Full Outer Joins gives all tuples of S and all tuples of R in the result set. The tuples of S which do not satisfy join condition will have values as NULL for attributes of R and vice versa. |
| **/(Division Operator)** | Division operator A/B will return those tuples in A which is associated with every tuple of B.**Note:**Attributes of B should be proper subset of attributes of A. The attributes in A/B will be Attributes of A- Attribute of B. |

[**SQL**](https://www.geeksforgeeks.org/sql-tutorial/): As opposed to Relational Algebra, SQL is a non-procedural language.

|  |  |
| --- | --- |
| **Operator** | **Meaning** |
| [**Select**](https://www.geeksforgeeks.org/sql-select-query/) | Selects columns from a relation or set of relations.**Note:**As opposed to Relational Algebra, it may give duplicate tuples for repeated value of an attribute. |
| [**From**](https://www.geeksforgeeks.org/sql-sub-queries-clause/) | **From** is used to give input as relation or set of relations from which data needs to be selected. |
| [**where**](https://www.geeksforgeeks.org/sql-where-clause/) | **Where**is used to give condition to be used to filter tuples |
| [**EXISTS**](https://www.geeksforgeeks.org/sql-exists/) | **EXISTS**is used to check whether the result of a correlated nested query is empty (contains no tuples) or not. |
| [**Group By**](https://www.geeksforgeeks.org/sql-group-by/) | **Group By**is used to group the tuples based on some attribute or set of attributes like counting the no. of students group by department. |
| [**Order By**](https://www.geeksforgeeks.org/sql-order-by/) | **Order By**is used to sort the fetched data in either ascending or descending according to one or more columns. |
| [**Aggregate functions**](https://www.geeksforgeeks.org/database-management-system-aggregate-functions/) | Find the aggregated value of an attribute. Used mostly with group by. e.g.; count, sum, min max.**select count(\*) from student group by dept\_idNote:** we can select only those columns which are part of group by. |
| [**Nested Queries**](https://www.geeksforgeeks.org/nested-queries-sql/) | When one query is a part of other query. Solving nested queries questions can be learnt in**http://quiz.geeksforgeeks.org/nested-queries-sql/** |

**Doubt in the question**

[**Conflict serializable and Conflict Equivalent**](https://www.geeksforgeeks.org/conflict-serializability/): A schedule is conflict serializable if it is conflict equivalent to a serial schedule.

**Checking for Conflict Serializability**

To check whether a schedule is conflict serializable or not, find all **conflicting operations pair**s of a schedule and draw precedence graph ( For all conflicting operation pair, an edge from Ti to Tj if one operation of conflicting pair is from Ti and other from Tj and operation of Ti occurs before Tj in schedule). If graph does not contain cycle, the schedule is conflict serializable else it is not conflict serializable.

Schedules are said to be conflict equivalent if 1 schedule can be converted into another by swapping non conflicting operations.

**Conflict Serializable:** A schedule is called conflict serializable if it can be transformed into a serial schedule by **swapping non-conflicting operations**.

**Conflicting operations:** Two operations are said to be conflicting if all conditions satisfy:

* They belong to different transactions
* They operate on the same data item
* At Least one of them is a write operation

Example: –

* **Conflicting** operations pair (R1(A), W2(A)) because they belong to two different transactions on same data item A and one of them is write operation.
* Similarly, (W1(A), W2(A)) and (W1(A), R2(A)) pairs are also **conflicting**.
* On the other hand, (R1(A), W2(B)) pair is **non-conflicting** because they operate on different data item.
* Similarly, ((W1(A), W2(B)) pair is **non-conflicting.**

**Note:** Two phase locking protocol produce conflict serializable schedule but may suffer from deadlock. On the other hand, Time-Stamp based protocols are free from deadlock yet produce conflict serializable schedule.

[**View Serializable and View Equivalence**](https://www.geeksforgeeks.org/dbms-how-to-test-two-schedule-are-view-equal-or-not-2/) : Two schedules S1 and S2 are said to be view-equivalent if all conditions are satisfied for all objects:

* If the transaction Ti in S1 reads an initial value for object X, in S2 also, Ti must read the initial value of X.
* If the transaction Ti in S1 reads the value written by transaction Tj in S1 for object X, same should be done in S2.
* If the transaction Ti in S1 is the final transaction to write the value for an object X, in S2 also, Ti must write the final value of X.

A schedule is view serializable if it is view equivalent to any serial schedule.

**Irrecoverable Schedules:**For a transaction pair < Ti, Tj >, if Tj is reading the value updated by Ti and Tj is committed before commit of Ti, the schedule will be irrecoverable.

**Recoverable Schedules:**For a transaction pair < Ti, Tj >, if Tj is reading the value updated by Ti and Tj is committed after commit of Ti, the schedule will be recoverable.

**Cascadeless Recoverable Schedules:**For a transaction pair < Ti, Tj >, if value updated by Ti is read by Tj only after commit of Ti, the schedule will be cascadeless recoverable.

**Strict Recoverable:** For a transaction pair < Ti, Tj >, if value updated by Ti is read or written by Tj only after commit of Ti, the schedule will be strict recoverable. The relationship between them can be represented as:

**Strict ⊂ Cascadeless Recoverable ⊂ recoverable ⊂ all schedules**

**// Doubt**

**(\*\*)**

**[File structures](https://www.geeksforgeeks.org/indexing-in-databases-set-1/" \t "_blank)**

**Primary Index :**: A primary index is an ordered file, records of fixed length with two fields. First field is same as primary key as data file and second field is a pointer to data block, where the key is available.

The average number of block accesses using index = **log2 Bi + 1**, where Bi = number of index blocks.  
   
**Clustering Index :**Clustering index is created on data file whose records are physically ordered on a non-key field (called Clustering field).  
   
**Secondary Index :**Secondary index provides secondary means of accessing a file for which primary access already exists.

Number of index entries = Number of records

[**B Trees**](https://www.geeksforgeeks.org/b-tree-set-1-introduction-2/)  
At every level , we have Key and Data Pointer and data pointer points to either block or record.

[**Properties of B-Trees :**](https://www.geeksforgeeks.org/b-tree-set-1-introduction-2/)  
Root of B-tree can have children between **2** and **P**, where P is Order of tree.

**Order of tree** – Maximum number of children a node can have.

Internal node can have children between **⌈ P/2 ⌉** and **P**  
Internal node can have keys between **⌈ P/2 ⌉ – 1** and **P-1**

[**B+ Trees**](https://www.geeksforgeeks.org/database-file-indexing-b-tree-introduction/)  
In B+ trees structure of leaf and non-leaf are different, so their order is. Order of non-leaf will be higher as compared to leaf nodes.

Searching time will be less in B+ tress, since it doesn’t have record pointers in non-leaf because of which depth will decrease.

This article has been contributed by Sonal Tuteja.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

Commonly asked Dbms questions | Set1

**What are advantages of DBMS over traditional file based systems?**  
**Ans:** Database management systems were developed to handle the following difficulties of typical Fille-processing systems supported by conventional operating systems.  
1. Data redundancy and inconsistency  
2. Difficulty in accessing data  
3. Data isolation – multiple files and formats  
4. Integrity problems  
5. Atomicity of updates  
6. Concurrent access by multiple users  
7. Security problems

(\*\*)

**What are the differences between DDL, DML and DCL in SQL?  
Ans:** Following are some details of three.  
***DDL***stands for Data Definition Language. SQL queries like CREATE, ALTER, DROP and RENAME,truncate come under this.(create,alter,drop,rename)  
***DML***stands for Data Manipulation Language. SQL queries like SELECT, INSERT and UPDATE,DDL come under this. (SIU)  
***DCL*** stands for Data Control Language. SQL queries like GRANT and REVOKE come under this.

**Remember(\*\*)**

**What is the difference between having and where clause?**  
**Ans:** HAVING is used to specify a condition for a group or an aggregate function used in select statement. The WHERE clause selects before grouping. The HAVING clause selects rows after grouping. Unlike HAVING clause, the WHERE clause cannot contain aggregate functions. (See [this](http://newtonapples.com/difference-clause-clause/)for examples).

**What is Join?**  
**Ans:** An SQL Join is used to combine data from two or more tables, based on a common field between them. For example, consider the following two tables.

**Table –** Student Table

|  |  |  |
| --- | --- | --- |
| ***ENROLLNO*** | ***STUDENTNAME*** | ***ADDRESS*** |
| 1000 | geek1 | geeksquiz1 |
| 1001 | geek2 | geeksquiz2 |
| 1002 | geek3 | geeksquiz3 |

**Table –** StudentCourse Table

|  |  |
| --- | --- |
| ***COURSEID*** | ***ENROLLNO*** |
| 1 | 1000 |
| 2 | 1000 |
| 3 | 1000 |
| 1 | 1002 |
| 2 | 1003 |

Following is join query that shows names of students enrolled in different courseIDs.

SELECT StudentCourse.CourseID, Student.StudentName

FROM StudentCourse

INNER JOIN Student

ON StudentCourse.EnrollNo = Student.EnrollNo

ORDER BY StudentCourse.CourseID;

**What is a view in SQL? How to create one**  
**Ans:** A [view](http://en.wikipedia.org/wiki/View_(SQL))is a virtual table based on the result-set of an SQL statement. We can create using create view syntax.

**CREATE VIEW view\_name AS**

**SELECT column\_name(s)**

**FROM table\_name**

**WHERE condition**

**What is a Trigger?**  
**Ans:**A [Trigger](http://en.wikipedia.org/wiki/Database_trigger) is a code that associated with insert, update or delete operations. The code is executed automatically whenever the associated query is executed on a table. Triggers can be useful to maintain integrity in database.

**What is a stored procedure?**  
**Ans:** A [stored procedure](http://en.wikipedia.org/wiki/Stored_procedure) is like a function that contains a set of operations compiled together. It contains a set of operations that are commonly used in an application to do some common database tasks.

**What is the difference between Trigger and Stored Procedure?**  
**Ans:** Unlike Stored Procedures, Triggers cannot be called directly. They can only be associated with queries.

**What is a transaction? What are ACID properties?**

**Ans:** A [Database Transaction](http://en.wikipedia.org/wiki/Database_transaction) is a set of database operations that must be treated as whole, means either all operations are executed or none of them.  
An example can be bank transaction from one account to another account. Either both debit and credit operations must be executed or none of them.  
[ACID](http://en.wikipedia.org/wiki/ACID)(Atomicity, Consistency, Isolation, Durability) is a set of properties that guarantee that database transactions are processed reliably.

**Consistency- any transaction from one valid state to another**

**remember**

**What are indexes?**  
**Ans:** A [database index](http://en.wikipedia.org/wiki/Database_index) is a data structure that improves the speed of data retrieval operations on a database table at the cost of additional writes and the use of more storage space to maintain the extra copy of data.  
Data can be stored only in one order on disk. To support faster access according to different values, faster search like binary search for different values is desired, For this purpose, indexes are created on tables. These indexes need extra space on disk, but they allow faster search according to different frequently searched values.

**Index to address**

**So direct access**

A **Natural Join** is where 2 tables are **joined** on the basis of all common columns. A **Inner Join** is where 2 tables are **joined**on the basis of common columns mentioned **in the** ON clause. ... **Inner join and natural join** are almost same but there is a slight **difference between** them.

# SQL query to get highest salary?

Below is simple query to find the employee whose salary is highest.

SELECT name, MAX(salary) as salary FROM employee

Remember

We can nest the above query to find the second largest salary.

SELECT name, MAX(salary) AS salary

FROM employee

WHERE salary < (SELECT MAX(salary)

FROM employee);

**How to find the third largest salary?**  
Simple, we can do one more nesting.

SELECT name, MAX(salary) AS salary

FROM employee

WHERE salary < (SELECT MAX(salary)

FROM employee

WHERE salary < (SELECT MAX(salary)

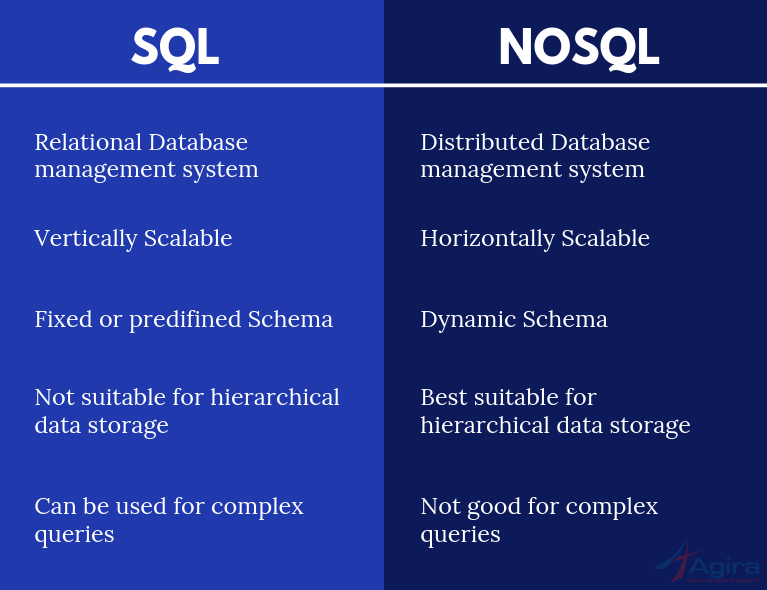
FROM employee) );

**select min(salary)from(select distinct salary from emp order by salary desc)where rownum<=2;**

**Commonly asked interview question set 2** **left**

**Difference between sql and no sql**

Key **Differences between SQL and NoSQL**. **SQL**Database is a Relational Database and a structured one whereas **NoSQL** is a Non-relational database likely to be more document and distributed than structured. ... **SQL** databases are vertically scalable whereas **NoSQL** databases are horizontally scalable.



# SQL Server: Foreign Keys with cascade delete

This SQL Server tutorial explains how to use **Foreign Keys with cascade delete** in SQL Server with syntax and examples.

## What is a foreign key with Cascade DELETE in SQL Server?

A foreign key with cascade delete means that if a record in the parent table is deleted, then the corresponding records in the child table will automatically be deleted. This is called a cascade delete in SQL Server.

A foreign key with cascade delete can be created using either a CREATE TABLE statement or an ALTER TABLE statement.

### **Syntax**

The syntax for creating a foreign key with cascade delete using a CREATE TABLE statement in SQL Server (Transact-SQL) is:

CREATE TABLE child\_table

(

column1 datatype [ NULL | NOT NULL ],

column2 datatype [ NULL | NOT NULL ],

...

CONSTRAINT fk\_name

FOREIGN KEY (child\_col1, child\_col2, ... child\_col\_n)

REFERENCES parent\_table (parent\_col1, parent\_col2, ... parent\_col\_n)

ON DELETE CASCADE

[ ON UPDATE { NO ACTION | CASCADE | SET NULL | SET DEFAULT } ]

);

**Order of execution**(FW GHSOL)

**FROM** & **JOIN**s determine & filter rows  
**WHERE** more filters on the rows  
**GROUP BY** combines those rows into groups  
**HAVING** filters groups

**Select**

**ORDER BY** arranges the remaining rows/groups  
**LIMIT** filters on the remaining rows/groups

# How to print duplicate rows in a table?

SELECT name, section FROM tbl

GROUP BY name, section

HAVING COUNT(\*) > 1

SELECT rowid,rownum fROM EMP

ROWID ROWNUM

----- ----------------------

AAAR4AAAFAAGzg7AAA, 1

AAAR4AAAFAAGzg7AAB, 2

AAAR4AAAFAAGzg7AAC, 3

AAAR4AAAFAAGzg7AAD, 4

AAAR4AAAFAAGzg7AAE, 5

**24.How Do you find all Employees with its managers?(Consider there is manager id also in Employee table)**

**Answer:**

Select e.employee\_name,m.employee name from Employee e, Manager m where e.Employee\_id=m.Manager\_id;

**GRANT AND REVOKE**

| **BASIS FOR COMPARISON** | **GRANT** | **REVOKE** |
| --- | --- | --- |
| Basic | Grant command is used for giving the privileges to the users. | Revoke command is used for taking away the privileges from the users. |
| When the control is decentralized | Granting is simpler. | Revoking is quite complex to perform. |
| Syntax | grant <privilege record> on <relation title or view title> to <user/role record>; | revoke <privilege list> on <relation name or view name> from <user/role list>; |

#### **SYNTAX of GRANT Command:**

grant <privilege record>  
on <relation title or view title>  
to <user/role record>;

**GRANT** SELECT, INSERT, UPDATE **ON** FACULTY, DEPARTMENT **TO** A1, A2;

### **Definition of Revoke**

The **REVOKE** command in SQL is defined to take away the granted privileges (authorizations) from the user of the database. The one who has the authority to withdraw the privileges is the database administrator.

#### **SYNTAX of REVOKE Command:**

revoke <privilege list>  
on <relation name or view name>  
from <user/role list>;

**REVOKE** INSERT, UPDATE **ON** FACULTY, DEPARTMENT **FROM** A1, A2;