Why would you ever need to compile database code?

Once you have the DBMS, you'd expect it to come in an executable form only, so how and why would you ever recompile it? The answer to this question is linked to other terms such as 'open source', modular, embedded etc. See below.

Stored Procedures

How are stored procedures controlled/executed?

Oppertunity to optimize, as code is repeated/reexecuted?

1.Some db operation triggers the procedure eg:SQL

2.Call by name eg:DBcleanup

3.timing eg.start on, interval

Question: how are modular design, compilers and open source related?

Modular or layered program refers to the design approach for the software itself. Modular allows a 'plug n' play' approach to the software subsystems of the program. So, in an Open source DBMS, you might be **able to reprogram** the Index subsystem, or the optimiser, or the recovery system etc to suit your requirements; or alternatively you may wish to **remove sections to minimise DBMS code** (see Embedded later). If you edit a C++ program, you will need to recompile it and then link it to the other modules of the DBMS. **Modular software design enables customisation, and tailored DBMS installations.**

Explain the features of MySQL DBMS listed.

Written in C and C++. Tested with a broad range of different compilers.

The MySQL Server design is multi-layered with independent modules.

Fully multi-threaded using kernel threads.

Multi-threading relates to **CPU scheduling**. A thread is a management unit of work for running on the CPU. Let's say you have 3 processes running on the computer: an email client, an internet browser and a DBMS. All three have equal importance so they are all scheduled for CPU time equally

The server is available as a separate program for use in a client/server networked environment. It is also available as a library that can be embedded (linked) into standalone applications.

Discuss SQL as an issue for a Database Administrator (DBA) using the following guidelines

Dialects What are they, why do they exist and why is it important that the DBA manage them?

SQL versions: what are versions and why is it important for the DBA to manage them?

Describe the data dictionary/data catalog in a DBMS

The data dictionary is not one thing. It is a collection (set) of database tables that store all the data required by the DBMS to manage the database. In some DBMS this is called the Information_schema. There are tables for Users, Connections, Transactions, Databases, Tables, Indexes, etc. This is called **meta data** i.e. it is data that describes other data. **The data dictionary in RDBMS can be queried using standard SQL**.

The dictionary stores all the meta data for system config and performance settings etc.

Data Dictionary/ Data Catalog: Explain why accessing the Dictionary/catalog is a potential critical point for system efficiency/performance.

Every database program that references a table name requires work in the back ground to check the existence of that table name, and then check the attribute names and data types referenced by the user program e.g Insert into Student (Sno, Sname, DOB) values ('S22', 'J.Murphy', '1/1/76')

Accessing the Dictionary/catalogue is a critical point for system efficiency because different components of the DBMS must access it to perform their function. Explain what aspects of the data dictionary are required when processing the following query?

Select Name, Student_No
From Student
Where Dept = 'Computing'

Indexing and Index Management

- **Explain why you would not recommend creating indexes on every attribute in a table?**
- **Explain why a Query processor/optimiser may not opt to use an index on a column even though it is available? Mention the role of Statistics**

Threads

- **Explain the term multi-threading**
- **Explain how multi-threading is of significance to DBMS architecture e.g. load balancing, availability.**

Recovery

Understanding asynchronous (independent) buffer\cache management and block I/O is essential to understanding recovery procedures in transaction based DBMS.

Asynchronous I/O
Block I/O
Transactions

A transaction is a **logical unit of work**. A transaction is not necessarily just a single database operation. It may be a sequence of several operations that transforms the database from one consistent (correct!) state to another. Database operations usually consist of operations on records in the database e.g. Insert, delete, update, select/retrieve etc. It is possible that **intermediary points during a transaction leave the database in an inconsistent state but the end state must be correct**. This property of transactions has critical implications for recovery i.e. returning the database to a correct/consistent point (note use interchangeably from now on)

Properties of a transaction : the ACID properties.

Atomicity : all or nothing. A group or sequence taken as 1 unit.

Consistency : a transaction transforms the database from one consistent state to another, intermediate inconsistency allowed.

Isolation : transaction activities are concealed from other transactions running concurrently until that transaction commits i.e. transactions are independent. This is handled by the concurrency control programs of the DBMS.

Durability : once committed modifications made by a transaction survive even if there is a subsequent system crash i.e. assume that non volatile storage is secure.

Why is block I/O used and why is it asynchronous?

I/O efficiency. The bottle neck is usually the Disk (2nd storage).

You have 3 Disk delays:

- Seek time is the highest, then
- Disk Rotation, with
- Transmission as the shortest delay.

As the Database Administrator (DBA) you are responsible for the location and size of the on-line log. In your opinion what are the issues that influence your decisions in these matters?

Checkpoints

A checkpoint is a point in recent time where we guarantee that any operation that is in the log is also written out to the physical database. Recovery can safely start from this point in the log/time to recovery the state of the DB to the time just before failure (or as close as possible).

- **As the DBA, setting the frequency of checkpoints is an important issue. Explain.**
- **Describe two mechanisms by which the frequency of the checkpoints might be controlled.**

1.Transaction Consistent Checkpoints(TCC). Before a TCC can start, the DB must be in a 'no activity' state, so existing transactions must be allowed to complete, and then the checkpoint can begin. No new transaction is allowed to start from the Checkpoint start time until it is finished. Note no recovery work is required back in time beyond the Checkpoint time). Why is that? (in class discussion). This is not good for busy on-line applications or for long transactions.

TCC is basic and has problems:

a. how long before existing transactions finish and

b. forced wait on new trans that would like to start

2. Action consistent checkpoints ACC improve on TCC by allowing existing transactions to complete the current step/operation before checkpoint starts BUT not finish completely. Once the active transactions finish their next operation the CKP can begin to write the modified pages to the disk. No activity is allowed during the actual checkpoint procedure. ACC prevent new transactions starting from the time the checkpoint is instigated. This limits the REDO to the CKP time however you may need to UNDO past the CKP time back to the start of the DB therefore UNDO data must be retained for those transactions.

3. Fuzzy checkpoints: See Appendix for details (check in class if details are needed for exam)

Fuzzy checkpoints concentrate on the 2nd principle of efficient checkpointing. They are programmed to manage the number of pages so that only some are written (e.g. by tracking time and only writing a page if it has been in RAM for some time)**.** As not all pages are written out, **it means that there needs to be more work if failure occurs figuring out (by processing the log) to find what was written and what was not, but in the normal case that is infrequent. On the positive side, you gain by saving time for every checkpoint.**

Undo/Redo:

Recovery Procedure (Undo/Redo) (general recovery procedure; buffers independent, ACC)

Using the above example: at restart time the following recovery procedure is used.

1. retrieve the last checkpoint record

2. create two lists called UNDO and REDO. Set UNDO equal to the list of all transactions given in the checkpoint record.

3. search through the log from the checkpoint time forward.

4. If a BEGIN TRANSACTION log entry is found for transaction T, add T to the UNDO list.

5. If a COMMIT entry is found for transaction T, move T from UNDO to REDO.

Note : ROLLBACK and implications.

6. When the end of the log is reached both list identify the transactions that require partial changes to be undone and unpropagated modifications for committed transactions (i.e. not logged) to be redone.

In the above example T3 and T5 must be undone. T2 and T4 redone.

The system now works backward through the log, undoing the transactions in the UNDO list i.e. replace existing with previous values: note insert and delete, until the start of transaction marker is found. The system then works forward again, redoing the transactions in the REDO list (need only redo from checkpoint time forward). Note possible confusion with backward/forward recovery systems described earlier.

Alternatives to Undo/Redo

There are other recovery schemes if the buffer manager program works differently. If it is guaranteed that all writes held until commit: therefore we know that modified data is held in RAM/buffers and not written to permanent storage so we can assume if RAM fails then effectively those changes have been discarded and so we have no Undo work to perform. This scheme is called **No undo/Redo**.

If writes are flushed directly we get **undo/no redo**. In other words we are sure that any modified page has been written to Disk so it's safe hence no need to Redo

What are the basic operations in recovery using Backups?

- Backup/dump and
- Restore utilities.

Concurrency

a) **Versioning? What is it, why use it, how might it be implemented?**

Multi-Version Concurrency Control (MVCC)

In certain application such as Web browsing, where there are many transactions(sessions) reading, but few that update, and even less few that commit an update. A problem with 'locks' is that the writers effectively block readers. If this delay cost is unacceptable, an alternative concurrency control mechanism is to use 'versioning' instead of locks. Essentially, each transaction operates on its own private copy (version) of the shared data resource (be it a row, table or entire document).

The DBMS **no** longer needs to manage slow locks and queues for access; it now needs to manage the versions of the data, particularly when any modification commits.

Note, any updates that are abandoned (i.e. not committed) can be just deleted.

A DBMS can control what version is correct in time, by using timestamps & transaction identifiers (i.e. locks and lock conflicts are not used for correctness)

Explain the difference between lock promotion and lock escalation.

Locks are data structures that indicate which part of the system (row, page, table, tablespace, database, ...) is protected in some way. DB2 remembers this information in memory. Blocking the whole database all the time when an application accesses a small part of it doesn't make much sense. So if you have an application that deals with a lot of data, it potentially has to have many locks. Those locks accumulate and need space in memory.

So when the list is almost full DB2 does a "lock escalation". That means, many row-level locks held by one transaction on a table are escalated to a single table-level lock, for example. Instead of having 1000s or millions of locks (per row), it is better to have lock on that table , there by relieving the pressure on the lock list.

Lock promotion is the action of exchanging one lock on a resource for a more restrictive lock on the same resource, held by the same application process.

Explain why a request for a lock promotion or escalation must be granted or else explain in what conditions such a request can be denied?

Data Access Protocol(DAP): There are two versions of the DAP, one basic or weak, and the second strong.

State the basic DAP.

> **Database audit and protection (DAP)** tools provide comprehensive security for relational database management systems (RDBMSs). DAP tools have their roots in the basic monitoring capabilities of database activity monitoring (DAM) tools. In response to expanded client requirements, vendors have added additional capabilities, such as data discovery and classification, threat and vulnerability management, application-level analysis, intrusion prevention and activity blocking, and identity and access management analysis.

Explain using an example how this basic DAP is weak in terms of concurrency? That is, devise concurrent transactions to show that database integrity can be compromised if the weak version of the DAP is used.

Explain how the strong version of DAP prevents the problem(s) outlined in part b above.

"Locks should be always maintained until end of transaction". Discuss your opinion of this statement.

What is lock granularity? Why is it of significance in concurrency control?

- **Describe why a request for promotion can be denied.**

Security

Write a detailed note on Security Standards. You should include reference to
Why are standards used?

May be Industry specified e.g. an ISO standard. Alternatively, may be National e.g. US Department of Defence, NSA etc. Standards and Metrics are valuable for appraisal of security systems, independent assurance etc

US Dept of Defence National Security Standards.

The general security classification scheme used defines four security classes(D,C,B,A). D provides minimal protection,

- C covers Discretionary protection where access control is subject to the discretion of some user e.g. the data owner or DBA. This just covers user access
- Class B covers Mandatory control where categories and classification levels are used. Each data object is given a classification level, and each user is assigned a clearance level. In this type of system rules are set in advance for what type of user can access what categories of data. A user can only do what the rules state, there is no facility for an ordinary user to control access.

Discretionary + Mandatory
Explain why mandatory is considered stronger than discretionary.

Mandatory control is stronger than Discretionary as Mandatory not only covers basic access controls of 'who does what', but in addition it involves Information classification (data): different types of data: general, private, secret, admin.

Discuss Authentication of users for Access Control (types and effectiveness)

Types of access control

- Column Name dependent : vertical subset
- Row Content dependent : horizontal subset

} these can be implemented by using either SQL Views or by Grant

- Access action Types
- Context Dependent Control

} Can only be implemented by SQL Grant

- History-Dependent

} neither of these are implemented by Grant/View

Proliferation of rights

In a decentralised system one of the ways that control can be administered is by Ownership. The owner is usually the person/group who creates the data. The owner may have all or a default subset access controls on the object. Ownership is very hard to control particularly if the owner of data can pass on access rights to others. This issue is known as the **proliferation of rights control**. In general, we need to **control the delegation of rights, so** how do we record or restrict to whom that user delegates?

If you have a certain right, can you give it to someone else? This is simple to implement, just set a 'copy privilege' flag. However, what if we want to restrict delegation to a subset of users?

The system could also control **proliferation of rights** by only allowing a rule to be delegated a certain number of times. Note another alternative would be to enforce that once a user, who has been allocated delegate rights on an object, decides to pass on those rights then that user loses delegation power.

Write a detailed note on risk assessment in database security

Risk Assessment = Likelihood x Severity:

0 – 4 Trivial Impact. No further improvements necessary provided control measures are maintained.

5 – 8 Impact. Risk is tolerable when control measures identified are implemented. Review on a six-month basis.

9 – 12 High Impact. Further Risk reduction measures should be considered. Implement measures within 1 month.

16 – 25 Intolerable Impact. Immediate control measures are required. Implement business protection measures immediately.

SQL Grant

Describe the SQL Grant statement

This form of the GRANT statement grants privileges on tables or views. This form of the GRANT statement grants privileges on tables or views. The privileges held by the authorization ID of the statement must include at least one of the following:

- For each table or view identified in the statement:
    - Every privilege specified in the statement
    - The system authority of *OBJMGT on the table or view
    - The system authority *EXECUTE on the library containing the table or view
- Administrative authority

If WITH GRANT OPTION is specified, the privileges held by the authorization ID of the statement must include at least one of the following:

- Ownership of the table
- Administrative authority

Explain how SQL Grant can be used to implement different types of security access control.

Grants or revokes types of access on a table or view to a user. When a table or view is created the system gives full grant options to the user that created the object. The user is given the option to grant other users selective access to the object through the GRANT and REVOKE syntax.

```
GRANT SELECT ON TABLE MyTable TO toby
```

SQL Injection

SQL injection is a code injection technique, used to attack data driven applications, in which malicious SQL statements are inserted into an entry field for execution (e.g. to dump the database contents to the attacker). SQL injection must exploit a security vulnerability in an application's software, for example, when user input is either incorrectly filtered for string literal escape characters embedded in SQL statements or user input is not strongly typed and unexpectedly executed. SQL injection is mostly known as an attack vector for websites but can be used to attack any type of SQL database.

How are SQL injection attacks prevented?

Permissions Settings: Limiting the permissions on the database logon used by the web application to only what is needed may help reduce the effectiveness of any SQL injection attacks that exploit any bugs in the web application.

Pattern Check: Integer, float or Boolean parameters can be checked if their value is valid representation for the given type. Strings that must follow some strict pattern.

Parameterized Statements: With most development platforms, parameterized statements that work with parameters can be used (sometimes called placeholders or bind variables) instead of embedding user input in the statement. A placeholder can only store a value of the given type and not an arbitrary SQL fragment. Hence the SQL injection would simply be treated as a strange (and probably invalid) parameter value.

In many cases, the SQL statement is fixed, and each parameter is a scalar, not a table. The user input is then assigned (bound) to a parameter