Database Administration Labs (DBALabs)

With the support of the EC LLP Transversal programme of the European Union

Disclaimers
This project has been funded with support from the European Commission. This publication [communication] reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Trademarks of products mentioned are trademarks of the product vendors.

Learning Objectives:
- understanding the basic concepts and technologies of the mainstream DBMS
- understanding basics of managing the mainstream DBMS instances

Prerequisites:
- We expect that the participant of the lab is familiar with SQL basics and
- knows how to use SQL commands with the tools of the selected DBMS

Contents

Introduction ......................................................................................................................... 2
Part I – Concepts .................................................................................................................. 4
  Database Administrator Roles ......................................................................................... 4
  Database Environment .................................................................................................... 5
  Database Server Instance .............................................................................................. 6
  Database data files, pages, and transaction log .......................................................... 8
  SQL standard, SQL implementations, Scripts and Tools ........................................... 10
  Metadata in System Tables and System Views ............................................................ 11
  Security .......................................................................................................................... 13
Part II – Database Administration ..................................................................................... 15
  Database Design ............................................................................................................ 15
  DBMS Selection and Licensing .................................................................................... 16
  Physical Design ............................................................................................................. 16
  Organizational Design .................................................................................................. 17
  Instance Installation ...................................................................................................... 17
  Database Implementation .............................................................................................. 17
  Index Design .................................................................................................................. 17
  Loading and Unloading Data ......................................................................................... 18
Databases are everywhere from enterprise applications on mainframe computers to mobile phones. Even if the DBMS products are becoming quite automatically managed systems, the current mainstream DBMS products still need to be managed by database professionals, especially since the up-to-date data is a most important asset of any company or organization.

For database management many database related professional roles can be identified, such as

- Data Administrator (DA)
- System Analyst, Data Analyst
- Database Designer
- Database Administrator (DBA)
- System Administrator (SA)
which however are often in a small company combined to tasks of a single database professional typically called as DBA.

This tutorial provides an introduction to database administration, the duties and tasks of a database administrator. We will focus on the management of multi-user online-transaction processing (OLTP) database servers. At the same time, this tutorial is a kind of umbrella tutorial, since for many of the topics we have more detailed tutorials to which we often refer for more details.

This tutorial is the first version of our vendor independent study material on database administration focusing on the “Big Three” DBMS products used by ICT industry - DB2, Oracle, and SQL Server. Future versions of this tutorial will be available at http://www.dbtechnet.org/labs/dba_lab/DBALabs.pdf

Note: In these tutorial versions of DBTechNet, with DB2 we mean DB2 for LUW (Linux, Unix or Windows) and not the mainframe editions of DB2, and all our DB2 examples have been tested using the free DB2 Express-C 9.7edition, which has proved to be an excellent tool for self-studying purposes. Of Oracle DBMS products, we use Oracle XE 10g, and of SQL Server product family we use SQL Server 2008 Express.

As the main sources for compiling this material we have used manuals of the “Big Three” DBMS products, and the Web articles by Craig S. Mullins and his book “Database Administration” [1], which is based on his experience at BMC and contacts with BMC’s customer base and the mainstream of DBMS products.

In this first version of the Database Administration tutorial, we try to provide at least an overview of the administration tasks, and in the future we try to provide more detailed material with examples.

The tutorial consists of three parts:
- Part I introducing concepts,
- Part II presenting the basic administration functions, and
- Part III just listing more advanced topics to be covered in future versions plus appendices.

The tutorial itself is generic in terms of the DBMS products, but in examples presented in appendices we use the free SQL Server 2008 Express. The reasons for this are: it is free, it has excellent GUI tools for educational purposes (although limited compared with the commercial editions of SQL Server), and it is not available on the free Linux platforms which we use in our free virtual laboratories. So the appendices at least literal provide a short introduction to the concepts and tools of SQL Server, in case a student does not have access to the actual software. Also to a new user the appendices will give a short “First Steps” introduction to SQL Server administration using snapshots of the tools which come with the commercial editions of the product, which is widely used in ICT industry.

The examples in the appendices may help an user of DB2 Express-C or Oracle XE in corresponding tasks providing vision or models in search of solutions using facilities of these free products, which are available in our virtual laboratories.
Part I - Concepts

Database Administrator Roles

Database administrator (DBA) is the job role of database professionals who plan, implement, control, and maintain one or more database servers, quite often of different database management system (DBMS) products and editions. Due to the variety of types of organizations, applications and DBMS product mixtures, the DBA roles may be of different types, such as

- System management-oriented, such as a SAP R/3 system administrator who is responsible for the whole system environment; is expert in knowing the used DBMS product and the application server(s); knows how to take technically care of the database; but who is not necessarily familiar with the logical structures or content in the database of all those over 15,000 different tables.
- DBMS product-oriented, typically a certified DBA of Oracle, DB2, or SQL Server.
- Architecture/Technology-oriented specialist DBA who manages an "orchestra" of database servers of different DBMS products; supports, for example, sales organization; and controls remotely database servers of customers.
- Application oriented DBA who knows thoroughly the logical structure of the database of an application; perhaps supports, for example, sales organization; and controls remotely database servers of customers.
- Management-oriented DBA role which also involves the administration of paperwork and records of various topics, including license management, and user roles in organization and mapping of user roles to database roles, access control, authorization, and security architecture.

These are covered in more details in Mullins' article "Types of DBAs" at http://www.craigsmullins.com/dbta_065.htm

Partly based on Mullins' book [1], the responsibilities of different phases in database application development life cycle between the roles of Data Administrator (DA), Data Analyst / Database Designer / Architect, Database Administrator (DBA) and System Administrator (SA) can be seen according to Table 1, where "X" stands for the primary responsibility and "(x)" stand for a "vice responsibility".

<table>
<thead>
<tr>
<th>Responsibility of by DA, Analyst, Designer/Architect</th>
<th>DBA</th>
<th>SA</th>
</tr>
</thead>
<tbody>
<tr>
<td>IT infrastructure</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data and Metadata Policy</td>
<td>X</td>
<td>(x)</td>
</tr>
<tr>
<td>Requirements Analysis</td>
<td>X</td>
<td>(x)</td>
</tr>
<tr>
<td>Data Analysis</td>
<td>X</td>
<td>(x)</td>
</tr>
<tr>
<td>Database Modeling, Design</td>
<td>(x)</td>
<td>X</td>
</tr>
<tr>
<td>Development</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Testing</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Implementation</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Maintenance and Tuning</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

Table 1  Comparison of responsibilities of the DA, DBA, and SA roles

Tasks of DBA roles can be performed as part-time or a full-time by one person or by a team.

Development-oriented and production-oriented DBAs need different skills, as Mullins points out in his article “DBAs Need Different Skills in Development and Production” at http://www.craigsmullins.com/dbta_023.htm.
and in the article at http://www.craigsmullins.com/dbta_085.htm he lists "The 17 Skills Required of a DBA" covering also the most important duties of a DBA. In summary we can agree that a DBA has to be "Jack-of-all-Trades" as Mullins says. The following "DBA rules of thumb" characterize the work of a DBA as listed by Mullins [1]

1. Document everything
2. Keep anything
3. Automate!
4. Share your knowledge
5. Analyze, simplify, and focus
6. Don't panic
7. Measure twice, make it right at first time
8. Understand your business, not just the technology
9. Don't become a hermit, communicate
10. Use all of the resources at your disposal
11. Keep your knowledge up-to-date.

The top priorities in the mind of a professional DBA are the reliability, security, integrity and recoverability of data in the database without compromises. Then comes availability, performance, and scalability in serving the applications and serving the developers and users.

**Database Environment**

Databases provide the reliable storage for the persistent data of applications. Concepts of DBMS system, database instance, and database have no globally accepted standard definition, but for this tutorial we adopt the definitions for these concepts from database administrator's (DBA) point of view to the mainstream DBMS products: DB2, SQL Server, and Oracle, the "Big Three".

Building a database environment starts by designing the application architecture, setting up the ICT infrastructure and selecting the proper DBMS product and set of tools, considering the applications, types of data, volumes, needed capacities and scalability, cost of ownership, editions of the DBMS, etc. In the selection of DBMS, some benchmarks and customer references might be consulted.

**Software**

For installing the DBMS software, it is important to read carefully the requirements of the DBMS version for the server computer, and to select proper place for the software, select proper components and tools, and select proper language to be used in the software dialogs.

**Instance and Databases**

After installing the DBMS software product on a server computer, one or more database server instances can be installed on the server. Usually the software installing routine prompts also for installing of the first instance. Figure 1 presents a generic overview of a database server instance. This is just an instance reference model, which we us to explain common features of the DBMS products even if the products use different terminologies and partly different database environment architectures, shortly explained in Figure 2.

For the database instance, an account in the server domain is entered for the privileges of the instance processes. Also time zone and localization including defaults for character sets and collations (sort orders
of characters) are asked. The instance configuration and registry of the database(s) created into the instance are stored in special control files. A database consists of tablespace files and transaction log files. Some DBMS products use special control files also for storing the database configuration.

Database server instance is activated automatically when the server is started, or it can be started manually by start up command. At activation phase, special server instance processes are started and caches including bufferpool(s) and log cache(s) are allocated in the main memory of the server computer. Listener process listens to a configured port of TCP/IP connection requests from application clients, and passes serving of the requests to one of the agent processes/threads. In case some client is local in the server computer, then shared memory protocol is typically used instead of TCP/IP. The special roles of Oracle’s instance processes are explained in Oracle manuals and many textbooks, for example in "Database Systems" by Connolly and Begg[2], whereas for DB2, only limited documentation of its processes are provided. In SQL Server, a single process takes care of most of the work accompanied with the SQL agent process for scheduled tasks and as listener of TCP/IP connections.

Sizes of the used pages, bufferpools, log buffers, and other caches can be configured for databases. The active instance can be configured to alert about problems using emails or messaging and write problem reports in errorlog files. Instance can also be configured to write reports to trace files of filtered events.

Figure 1. Database server instance (reference model)
Active instance has also **other caches** in main memory for data structures of current connections, transactions, locks, commands and execution plans of the commands, loaded procedures and triggers, etc.

Comparing instance architectures of the Big Three and our reference model in Figures 1 and 2, we can say that DB2 matches the model. Every instance has a unique name and unique port in scope of the TCP/IP of the server computer. In the DB2 vocabulary, a tablespace is called "table space" and bufferpool as "buffer pool". Table spaces may use different page sizes and therefore every database has buffer pools for all page sizes used. Every database has also log buffers of its own.

In Oracle an instance consists of one database, so these terms are often used like synonyms. All instances on a server computer have common listener which listens to the same TCP/IP port, and the instances are identified by their unique SID names on the "TCP/IP & port domain".

In Microsoft SQL Server (MSSS) one instance can be the default instance on the server computer, whereas additional instances need to have unique instance names. Database names must be unique in the scope of the instance. MSSS does not use configuration files, but in addition to application databases (user databases), every instance has the following four **system databases**: Master, Model, Msdb, and TempDB. Master takes care of configuration and controlling of the whole instance. No application tables or other schema objects should be created in the Master database! All databases in a MSSS instance share the same bufferpool. Tablespaces are called "filegroups" in the vocabulary of MSSS, but as default, only one filegroup and data file are created for a new database.

The instance can be started (processes instantiated) as service and stopped (shutdown). It can be configured to start automatically or manually. An active instance, database, or tablespace can be changed to read-only state or closed totally from users into a "quiescent" state.

Even if multiple instances can be implemented in the same server computer, performance can better managed if only one instance, and actually no other software applications, is installed on the same server computer. Development work in the same database as production work is not acceptable due to security reasons, and even development in a separate database in the same server may slowdown performance in the production database. Therefore, at least for business critical database applications, separate database environments on separate computers should be used for development and production.

Especially in large ERP applications for quality assurance (QA) reasons, even more database environments are needed for a single application as described in Figure 3

- one or more development and test environments,
- QA environment used for system and stress tests of every change before moving it to production environment. QA environment might be used also for user training purposes
- and the actual production environment, which might also consist of a **cluster** of database servers for high-availability, reliability, and scalability.
Database data files, pages, and transaction log

A database is a collection of object structures (tablespaces, tables, indexes, etc) and data which are managed as a "consistent whole". The contents of the database are stored on one or more data files on discs and these files are managed as file groups called tablespaces. A table or index on a table is created in a tablespace. This means that the pages of the created object will be stored in file pages of some files of that tablespace. The data files of a database are identified internally by unique file numbers in the

Figure 3. From test to production (see Mullins [1])

Figure 4. Database Server Instance
database, and managed as sequence of pages (also called blocks) having page size of 4, 8, 16, .. KB and identified accordingly by page numbers in the file.

**Transaction log files** are sequential files which store records of before and after images of all inserted, updated or deleted rows in all transactions, and records of commits and rollbacks, so that when a transaction ends and the commit/rollback record is written on disc in transaction log, we can trust that in case of logical or physical failure the database can be recovered up to the consistent state of that last committed transaction. Transaction log files are chained circularly, and when a transaction log file gets full, writing continues to the next file in the chain, and contents of the filled log file need to be archived so that the file space can be reused when the previous file in the chain gets filled again.

![Figure 5. Typical page format](image)

The typical **page format** of a table or index is presented in Figure 5. Every page has the following 3 parts:
- page header containing various control data used by the DBMS,
- data area for variable length records for storing rows or index entries of the object structure
- slot index (slot directory) containing offset addresses of the records on the data area.

Typically a page is used for rows of the same table only, and the table is indicated in a field of the page header. Rows are stored on the data area records, which contain also some control information about the row, such as column lengths and offsets on the record. Row addressing is based on the indirect address **RID** (also called as ROWID, or tuple id TID) which is built from file number, page number, and slot number. In case a modified row has grown in size so that it no longer fits in the original page, the RID address remains the same but the record is split into parts which are stored on pages where there is enough room to accommodate, and the parts are chained together to preserve its original content and its sequence.
The pages of a table (or an index) are stored in double chained lists in the set of files of the tablespace into which the table (or index) was created. Read and write operations between the data files and the bufferpool occur page at a time, but for a faster sequential access in case of pre-fetching, a set of 8 or more pages in the chain are stored adjacent to each other, and the set is called as an extent. Also, for fast finding of pages having room for new records, the database maintains chains of free list pages.

For more detailed information of the page header fields and the record structures, we refer to textbooks and DBMS product manuals.

Figure 6  Relationships of storage concepts

Figure 6 presents a simplified map of storage concepts. With objects in the figure we mean tables and indexes, whereas Oracle calls these segments. Typically an extent is used for a single object only, but SQL Server starts object storages sharing the first extents with multiple objects. A table page is typically used for records of a single table, but in case of Oracle’s cluster segment, pages can be shared between multiple tables, parent and its child tables. In this paper, we are mainly interested in database pages of tables and indexes in general.

For performance reasons, the needed index and data pages are first fetched into page frames in a bufferpool, allocated in the main memory of the server computer. The fetched pages remain in the buffer pool as long as there is room available, and so if a page is needed again, no disc I/O is spent for retrieving the page. This is the main performance benefit and the key for scalability of multi-user databases.

**SQL standard, SQL implementations, Scripts and Tools**

ISO SQL standard versions before 1992 were compromises between DBMS vendors and left many issues to be implementation-specific. This continues still even if the standard in many issues goes further than the current implementations. The standard does not even define such concepts as database instance, user, database, tablespace, index, locking, transaction log, and many functionalities needed in database management, such as instance start-up or close, create database, database backup and restore, etc. These, however, appear with quite similar syntaxes in almost every SQL implementation.

SQL implementations can also be used as scripts, which make it possible to automate database maintenance functions. Extending the scripts with parameters and control structures has led to wizard scripts, and various maintenance tools, such as import, export, load, unload, tracing, etc. Typical
maintenance and development tools include SQL editors, backup/restore, export/import, scheduling maintenance routines, trace and monitor utilities. These are often collected into all-in-one toolboxes, called “enterprise manager” providing graphical user interface, such as Control Center and Data Studio of DB2, Enterprise Manager of Oracle, and Management Studio of SQL Server. These tools with graphical user interface release DBA from remembering the many options of the maintenance commands, which these tools themself use “behind the scene”\(^1\). Recently the trend has been to move these tools into browser-based tools.

Also, many third-party companies have specialized in the area of database administration tools, providing tools which can be used maintaining multiple database products with generic interfaces, thus helping DBAs who need to maintain different types of database servers.

**Metadata in System Tables and System Views**

**System Tables and Views**

Metadata i.e. data about data means all the data that database instance processes need to know about the databases, physical and logical structures, roles, users, system and schema privileges, tables, columns, constraints, views, stored routines, triggers, etc. in the databases. The metadata is organized as system tables, which are also called system catalog, or data dictionary for every database. So no database is empty, since every database contains system tables, which contain at least the information of the structures of those system tables.

Usually the contents of system tables are made available to applications only thru various system views, and DBA and some other privileged user roles can access most of the system views, whereas other users have only limited access to the system views.

The implementations of the system tables in DBMS products are very different, and ISO SQL standard does not actually define what system tables should contain. The standard specifies a definition for system tables as tables in DEFINITION_SCHEMA, but it is provided only as an example describing how to define the set of standardized views to system tables defined as views in INFORMATION_SCHEMA, which should be available for applications [3].

Beside metadata of the database structures, also dynamic cache views (called dynamic management views in SQL Server 2008) to current contents in the caches, such as sessions, transactions, locks and lock waits are available to privileged users. In our tutorial on SQL Concurrency Technologies, we demonstrate the use of these views studying behavior of locks as concurrency control mechanisms.

**Database Schemas and Schema Objects**

Every object in a database belongs to one and only one database schema. A schema can be considered as a kind of logical database in a physical database. For example the system tables belong to some system schemas. According to ISO SQL-86, user tables and related objects belong to some schema owned by some authorization ID i.e. some user registered in the database, although the standard does not define how users are introduced to the database, but leaves this as implementation-specific. According to the schema model of SQL-86, all objects which the user creates in the database belong to that unique schema of the creator, and if some other user is granted some privilege to access that object, the grantee need to qualify

\(^1\) In some tasks these tools may fail, and DBA need to be able to use the SQL dialect of the DBMS, as we have shown in our Backup and Recovery tutorial.
the object using the schema name as the namespace of that object. This is still the default schema model in Oracle.

According to Melton [3] the schema model of ISO SQL-92 was changed so that a user could own multiple schemas, and ISO SQL:1999 defined that a schema could be owned by a role.

In DBMS products, a DBA can create objects in a schema, which is not actually owned by any user. In DB2 a user can change the current default schema in SQL session.

Starting from version 2005 of SQL Server, the user and schema concepts are separated. Access to schemas can be granted to users. Ownership of a schema can be granted to multiple users, and a user can have ownership to multiple schemas, and schema ownerships can be changed.

According to ISO SQL standard, schemas belong to catalogs and a schema object can be qualified as follows

```
[[<catalog name>].]<schema name>.<object name>
```

Catalog containers have been implemented in some DBMS products, but not in the Big Three. However, remote databases can be linked accessible in Oracle, DB2, and SQL Server, and the reference name assigned to the remote database can act as the catalog qualifier. In SQL Server, a user may have access to other databases in the local instance and those database names can be used as the catalog qualifiers of schema objects in those other databases.

Figure 7 presents a simplified overview of relationships of schema objects where database and index are not concepts of ISO SQL. In SQL Server, index names need to be unique in the scope of the table, whereas in Oracle and DB2 indexes are schema objects i.e. index names need to be unique in a schema.
Note: In DB2 stored procedure names can be overloaded, i.e. several procedures can have the same name, and the procedure matching the procedure call will be solved according to the matching number of parameters. Entering also the clause SPECIFIC <unique name> in CREATE command, it is possible to assign an extra unique name for the procedure, which name can be used on dropping the right procedure.

Security

Authentication and authorization

The typical user authentication solutions used in DBMS product include mainly the following

- SQL authentication, in which a virtual username is registered in the instance and protected by a password, so that whoever knows the username and password can log in into the databases in the instance. In general, this is not considered as secure as the following 2 solutions.
- Operating System (OS) authentication, in which case the selected users or user groups in OS domain are granted connect access to the instance, and DBMS trusts the authentication service of the OS domain (from local or other computer in the domain).
- Kerberos authentication, in which connect access to the instance / database is granted to the selected users known by and authenticated by the Kerberos server in use.

Of these solutions, in the Big Three products, the following authentications have been implemented:

- In Oracle, all three authentication solutions are supported.
- In DB2, only OS authentication and Kerberos authentication are supported on instance and database-level. For more details we refer to Chong [4].
- In SQL Server, only SQL and OS authentications are supported, and the default is OS authentication. Authentication grants access as login user only to the instance, and accessing selected databases in the instance need to be granted separately, as shown in Appendix 1.

ISO SQL standard and all DBMS products support Discretionary Access Control as authorization mechanism where authorization is the policy defining what the user can do in the database, what privileges a user has on the instance and database-level, and what access privileges a user has to certain schema objects.

Privileges on instance level are management privileges of instance level roles usually assigned to OS administrators. The user who first installs the instance typically has all privileges for the instance and database in the instance. In Oracle, this user can log in to instance as SYS user in SYSDBA role. In DB2, the corresponding authority is SYSADM. In SQL Server, the authority is SYSADMIN server role, but if the instance is configured to use mixed authentication, which includes SQL authentication, then the special virtual login sa also has the SYSADMIN server role.

For other instance and database-level administrative roles and authorities, we refer to administration manuals of the products.
Typical instance and database-level authority a user needs is CONNECT access, which automatically grants the user all the privileges which have been granted to PUBLIC role in the database. For privilege to create tables and other schema objects in the owned schema, the user needs database-level authority, which in Oracle is RESOURCE, in DB2 CREATETAB, and in SQL Server the database-level role DB_DDLADMIN.

The user who has created a table owns the table and has exclusively all privileges to that object. The user may separately GRANT to other users SELECT, INSERT, UPDATE, DELETE or REFERENCES to that table depending on their access needs. We assume that the details of these are well-known to the reader. For more details, we refer SQL textbooks and the product manuals.

Oracle and DB2 support also Mandatory Access Control which means that data in database is classified for secrecy and also users can be classified and cannot see the data which is classified on higher security level than the user's own classification. This access control goes beyond ISO SQL standard and is mainly needed in military and national security applications, and these "paranoid truth" applications are totally out of the scope of this tutorial.

More typical secrecy arrangements include keeping the database server computer behind firewalls and applying encryption in network traffic, backups, and occasionally in storing database data on discs.

The use of SQL views is the well-known security method which should already be familiar to all readers.

Instance roles

We have already mentioned some instance-level administration roles for all the Big Three. On general level, the trend is that administrative people should not be able to see the data contents in the database even if they may maintain databases and structures in the database and manage privileges of applications and end users. Typical administrative role is some operator role, the duty of which is to keep the databases available for applications and make sure that backups are done reliably.

Database roles

All the Big Three have some built-in database-level authorities/roles, which usually provide some privileges for some administration tasks.

Granting the separate access privileges separately to all individual users can be huge work for the DBAs, for example, granting SELECT, INSERT and UPDATE privileges on 100 tables to 100 users would require 30 000 GRANT commands. This is solved by creating application roles based on needs duties of the application user groups, granting the needed privileges to these roles and granting the roles to proper users or groups of users.

A user may be granted multiple roles. In Oracle, a user can select which of the granted roles are active at a time in the SQL session.
Part II – Database Administration

Database Design

Database designing starts with requirements analysis, data analysis, conceptual modeling, and modeling of the logical database solution including data types and primary key planning. The logical database is normalized usually up to the third normal form (3NF). Based on the logical data model the physical model will be designed and finally implemented into the database instance of the selected DBMS product.

For these database design and implementation steps different types of modeling tools can be used as presented in Figure 8.

The primary responsibility of Database Designing work belongs to data analysts, data designers, and data architects, but especially in small companies the role of DBA and all these other professional roles may be taken care by same person.

The training material on the subject area of database design has been covered in DBTech Database Modeling and Design workshop.
DBA acts usually only as technical consultant in following steps of database design
- Conceptual design review
- Logical design review
- Normalization / denormalization review

whereas DBA has the primary responsibility in following tasks
- DBMS selection and licensing
- Physical design including capacity planning
- Organizational design
- Instance installation
- Database implementation
- Index design
- Loading and unloading data
- Database connectivity for applications
- Consulting application developers, SQL review of application codes
- Production test and database rollout to production
- Monitoring, tracing and Performance Tuning of production
- Backup and recovery
- Disaster recovery planning
- Data reorganizing management
- DBMS version upgrades and database migrations

**DBMS Selection and Licensing**

Based on the planned / available application and hardware architecture, licensing terms, available benchmark comparisons and cost, the DBMS product edition is selected. For more details on DBMS selection, we refer to chapter 10.7 in textbook “Database Systems” by Connolly and Begg [2]. In practice a major DBMS product which is already known and in use in the organization, is the winner.

Licensing policies and prices of DBMS products may depend on the limit of concurrent users, limit of number of processors, operating system, etc. In our tutorials and workshops we tend to use only the free Express editions of the Big Three, so the price topics out of the scope of our tutorial, although managing the licensing and cost topics are important in a DBA’s work.

**Physical Design**

In Part 4 of their book “Database Systems” Connolly and Begg [2] present a detailed methodology starting from conceptual database design, to logical database design, physical database design, and to monitoring and tuning of the operational database instance.

In physical database design, the logical database plan is mapped to physical design, preferably using some modeling tool. The planned data types of logical database design will be mapped to data types supported in the selected DBMS product. UDTs help in keeping data types consistent. Procedures and functions are tested. DBA designs and tests triggers for business rules.
Possible partitioning of large tables is planned.

Table and index allocation on tablespaces is planned, and tablespace allocation on dedicated discs is planned. Also allocation of the special tablespace of system tables, tablespaces for temporary spaces (TempDB database in SQL Server), large tables and corresponding indexes, large objects including XML are planned, preferably on separate discs. Allocation of transaction log files on separate fast disks is also planned.

For the final implementation of the database the SQL DDL commands will be generated based on the designed physical model as described in Figure 1.

**Organizational Design**

In this phase, a DBA plans with SA the allocation of servers, and the integration with other applications, maintenance windows of the database instances including backup strategy, collection of statistics, monitoring disc space and free space in tablespace files, alerts, designing database roles and security solutions including firewalls.

Planning backup strategy includes scheduling of full and partial backup windows, scheduling transaction log backups and archiving transaction log history.

**Instance Installation**

In this phase, the DBMS software is installed in the server, including all available fixpacks for the DBMS version.

In creating the database instance, instance name, TCP/IP address and port, other network protocols, proper timezone, country setting, localization i.e. culture dependent formats for time, money, decimal point, character set and collations are defined.

**Database Implementation**

The physical database design is written/generated into form of CREATE commands for the database, tablespaces and files according to the disc allocation plan, CREATE commands for schemas and schema objects, ALTER commands for constraints, etc, and the commands are executed. Users/groups are logins and roles are created. Privileges are granted for roles

Configuration parameters for the database are defined, such as, page size, size of bufferpool and other caches. Also a checkpoint interval and some other configuration parameters are selected, although current trend is to allow DBMS tune many parameters automatically.

**Index Design**

Indexes are critical for performance. They are used by a database for the following purposes:

- Apply predicates to provide rapid look up of the location of data in a database, reducing the number of rows navigated
- To avoid sorts for DISTINCT, GROUP BY, ORDER BY and UNION clauses
- To induce order for joins
To provide index-only access, which avoids the cost of accessing data pages
To enforce uniqueness in a relational database

However, indexes need additional hardware resources:
They add extra CPU and I/O cost to UPDATE, INSERT, DELETE, and LOAD operations
They increase preparation time because they provide more choices for the optimizer
They can use a significant amount of disk storage.

These topics have been covered by Gulutzan and Pelzer [5], Ramakrishnan and Gehrke [7], and for an in depth coverage we refer to the book of Lahdenmäki and Leach [6].
We have covered these topics in the DBTech Index Design Lab tutorial at http://www.dbtechnet.org/labs/idp_lab/IDPLabs.pdf

**Loading and Unloading Data**

Data from earlier databases need to be first exported to external files using UNLOAD or EXPORT utilities. Then data from these files and perhaps some other sources is loaded to the tables of the database using the LOAD or IMPORT utility of the DBMS. For the mapping of the fields in the files to corresponding rows need to be defined. Details of using LOAD and IMPORT utilities are covered in special product manuals of this topic. Also Gulutzan and Pelzer [5] have covered these topics.

Loading XML documents and other large objects will need extra planning, perhaps programming.

**Database Connectivity for Application Development**

Databases are needed for applications. A DBA needs to implement, advice and control the infra for data access technologies used by applications. This means managing the listener processes at server side for serving the database connection requests of clients, taking care of the firewall at server and providing passthrough to trusted applications at trusted client sites. It also means arranging proper network format and protocol services both at server-side and client-side, possibly with encrypted traffic and user authentication services, using, for example, Kerberos service. At client side workstations or application servers, proper database drivers (ODBC or JDBC drivers) and datasource definitions need to be configured for accessing the databases.

Note: In JDBC connections the secure connections need Type 2 drivers, whereas in educational environments we usually prefer using Type 4 drivers without extra security services, just for keeping it simple.

**Consulting Application Developers**

Duties of DBA include also consulting the application developers on the connectivity topics. Database provides the reliable storage for persistent data of applications. The reliable service provided by the DBMS has to be met by applications using well-formed SQL transactions. Accessing multi-user database, the transactions need to request for proper concurrency services from the DBMS. We cover the concurrency topics in our Concurrency tutorial series starting with
“SQL Concurrency Technologies” and “RVV Paper” tutorials. A DBA needs to be the database professional who understands these services, consults the developers on these topics, helps in previewing the well-formness of transaction and used SQL commands.

Developers should test the correctness and performance of the SQL commands before applying these in the application programs. Correctness should be tested in test databases using covering content, and performance should be tested with test data volumes corresponding the volumes in production databases. Performance should be verified using execution plan analysis, which is available nowadays in every DBMS product, usually both as estimate and as by-result of actual execution. Some DBMS products allow simulation of large tables just by changing the volume statistics to estimate future performance in the production environment. DBA is the expert to advice in use these to developers.

**Production test and database rollout to production**

A DBA is the responsible person for the infra and database when a developer team runs system test of the application, and when actual data is loaded to the database and condition of the database is verified ready for starting the production use.

**Monitoring and Performance Tuning**

**Avoiding bottlenecks**

Typical performance bottlenecks can be caused by

- inefficient disc use, too much I/Os
- inefficient memory use
- inefficient configuration parameters
- inefficient database design
- inefficient data access

Performance tuning should start from trying to avoid these bottlenecks by

- selecting proper hardware configuration
- assigning reasonable DBMS configuration parameter values
- training and QA review of logical and physical database design, and index design
- training and QA review of programming of transactions and SQL commands

which means avoiding

a) Inefficient CPU use by
   - using dedicated database server computer, avoiding competition with other services
   - having enough processors

b) Inefficient disc use, too much I/Os and I/O waits by
   - well planned file allocations to multiple dedicated discs avoiding I/O channel contention, especially using dedicated fast discs (future flash discs) for transaction log, and temporary tablespace files
   - systematic index design, trying to avoiding sorts in processing of DISTINCT, GROUP BY, ORDER BY, UNION clauses
- bypassing file system buffers

- installing enough main memory to avoid swapping
- reserving large enough bufferpools, log buffers, and other caches

d) Inefficient configuration parameters by

- making use of self-tuning wizards, for example Self-Tuning MemoryManager (STMM) of DB2 [10]

e) Inefficient data access by

- scheduling regular statistics collection to provide optimizer relevant information for access plan optimizations
- avoiding unnecessary “normalized collation” usage of Unicode data [10]
- consulting developers on concurrency management and SQL tuning

Note: Our virtual computer labs do not provide us with relevant hardware for performance tuning, since we usually have the whole virtual computer on a single disc, and every disc of a virtual computer is just a file in the host computer. Usually we share a single CPU with the host computer’s OpSys and applications, and we have only limited memory available.

Monitoring and tracing

Mullins defines database performance as follows

“database performance can be defined as the optimization of resource use to increase throughput and minimize contention, enabling the largest possible workload to be processed." (http://www.craigmullins.com/dbta_111.htm )

From point of view of an application the key performance indicators (KPI) of database server are the transaction throughput and the response time experienced by users.

**Throughput** in a database server is typically measured by transactions per second (TPS). Transaction Processing Performance Council uses this indicator in their special benchmark applications (see [http://www.tpc.org](http://www.tpc.org) for comparing DBMS implementations in special environments, but since actual transactions vary by numbers and types of used SQL commands, depending on the application, the TPS measurements are mainly useful as measurements of the server when following the workload at peak hours. Since only successful transactions should be counted, the TPS should be measured based on commits, and eliminating user “thinking time”.

Another key performance indicator of database applications is the average **response time** experienced by users, and which therefore can be measured in the client application as average time from the moment when user gives control to the application up to the moment when control returns to the user. The response time should not exceed 2-3 seconds.

As these figures depend on the application, some **baseline measurement** should be measured, so that increased values of these measurements should be taken as indications of a need for further investigations.

Using monitoring tools we usually see the current resource usages, whereas tracing tools provide measurement reports from longer periods of time.
By proper monitoring of the production environment, the DBA is kept informed on how the hardware, resource allocations, and the configuration efforts are succeeding. Monitoring and tracing itself may spend resources, so the monitoring/tracing strategy needs to be planned carefully. Monitoring and tracing can be done on operating system, DBMS, and application levels.

Operating systems provide tools for monitoring the use of system resources, such as, workload of processors, memory usage and swapping, disc I/O frequencies, network activity, etc. For example, on Windows platforms we can use Performance Monitor of the administrative tools, and on Linux platforms the System Monitor. Also, system level errors and warnings are automatically traced in some operating system traces, such as, Event views on Windows platforms.

DBMS products include monitoring tools of their own, but, for example, on Windows platforms SQL Server and DB2 integrate with Performance Monitor so that when the instance is running, sets of their performance counters can be monitored in real time or logged into Data Collector sets. In Figure 9, the user is selecting Buffer Cache Hit Ratio counter of SQL Server Express 2008 to be monitored.

Bufferpool hit ratio is one of the most important KPIs in databases telling how many row requests have been served directly from the bufferpool, thus avoiding disc I/Os. According to Rees [10], “hit ratios of 80-85 or better for data and 90-95% or better for indexes are generally considered good for an OLTP environment”.

Figure 9. SQL Server counters available in Windows Performance Monitor.

Using the tools of DBMS, or performance tools of some third party, or just using dynamic administrative views in SQL scripts, DBA can control the production environment, for example, for
long lock waits to find problematic transactions which might need tuning. In worst cases, the DBA may need to kill a blocking process to allow other production to proceed.

With DBA tools like Oracle Enterprise Manager, the DBA may drill down to see the execution plan of the SQL commands using most of the resources.

"nail reports"

Using tracing tools like SQL Profiler of SQL Server, the DBA can filter from the trace history SQL commands with peak resource usage, and help developers to tune the SQL commands, or tune the database by building new indexes to sort out the problem if the command is used frequently. Need and suggestions for new indexes might be solved using Index Tuning wizards.

Tuning

Execution plans of problematic SQL commands reveal if optimizer was not able to properly use indexes. The reason for optimizer failure might be caused by missing statistics, missing indexes, nonmatching order of key columns in the indexes, nonmatching data types, comparison of expressions, or problematic form of the SQL command. Sometimes use of full table scan can be more efficient than a heavy use of row fetches from an index scan.

Some DBMS products, such as Oracle and SQL Server, provide optimizer hints to be included in the SQL commands for affecting the decisions of the optimizer to produce a better plan.

For workload evaluation, the DBA needs to measure a baseline workload trace against which new time-to-time workload traces will be compared regularly.

Changing configuration parameters of an instance or a database should be done with extra care. In this case, remember to keep record of configuration changes, and change only one parameter at a time and observe the effects.

Backup and Recovery

A system backup of the server computer or even backup of all database files is not enough for maintaining durability of the database contents up to the last committed transaction in case of hardware failure or system failures. Database backups need always to be carried out using special database backup routines, in well planned and scheduled automatic backup jobs of the database, and automatic archiving of the transaction log history in some external server, from which the transaction history can be retrieved in case the database need to be recovered.

Backups and database recovery are perhaps the most important duties of a DBA. For more details of these topics we refer to special backup manuals of the products, and to the DBTech Backup and Recovery Lab tutorial at http://www.dbtechnet.org/labs/ccr_lab/RCLabs.pdf. Overview of Oracle's backup and recovery, including point-in-time recovery and the new flashback technology, is presented in "Database Systems" by Connolly and Begg [2].
Disaster Recovery Planning

In case of database inconsistency or when the whole database is lost, the database has to be recovered from backups and the whole transaction history since the full backup. Acceptable time window for the recovery depends on how critical the database is for the daily business in the company. Considering the acceptable time, there is no room for experimenting, but the recovery routines need to be planned and tested beforehand, since everything need to work right on the first try.

Data Reorganizing Management

Collecting statistics may reveal fragmentation of data and index pages, or decreased control of a clustering index in finding proper pages for rows. Also, appearing performance problems may indicate disorganization of data. These problems are solved by the reorganization of tables and indexes. The database management tools have become quite powerful and, in many cases, can solve some reorganization automatically, even online. However, due to performance and possible concurrency problems, it is better to use a maintenance time window, closing the database from users for the time of reorganizing.

Reorganization of a table requires a complex series of steps to accomplish, of which we present a generic scenario in Listing 1.

1. Close use of the database from applications
2. Backup the database, so that in case of problems it is always possible to restore the database into the original state
3. Disable transaction logging
4. Export the data from the table to be reorganized, sorting according to the possible cluster key of clustered or clustering index
5. Disable foreign key constraints of possible child tables
6. Drop indexes of the table
7. Truncate contents of the table to be reorganized, or drop the table and the related database objects (views and triggers) and re-create the table (including possible clustered index) and the related database objects
8. Import or load the data back to the table
9. Re-create indexes of the table
10. Enable foreign key constraints of possible child tables
11. Backup the database
12. Collect statistics of the table and its indexes
13. Enable transaction logging
14. Allow use of the database to applications

Listing 1. A generic table reorganization scenario

If no changes are needed in the table structure and there are no child tables, then truncate can be used, otherwise delete and re-create of the object(s) is the safer way.
Some structure changes can be done by ALTER command, but depending on the DBMS, there are limitations on what can be done using ALTER. Database management tools may have wizards which can do even more complicated changes. The most complicated table structure changes may need extra programming work. In Appendix 2 we verify how our generic reorganization scenario can be applied using tools of SQL Server 2008.

As much as possible of DBA's regular work should be automated. This is often done by writing
shell scripts combined from operating system commands, and commands and utilities of the DBMS product.

Modern products may provide also wizards for creating automated maintenance jobs to be run on some regular basis, reorganizing tables and indexes, collecting statistics, backing up databases and transaction logs, running ETL data transfers to data warehouses, etc.

As an example of maintenance automation, above is a report of a maintenance plan steps generated by a wizard of SQL Server Management Studio

**DBMS Version Upgrades and Database Migrations**

Every now and then DBMS vendors publish updates to the DBMS software, and it is the responsibility of the DBA to test and decide when and which updates are installed into the production environment.

Sometimes, perhaps after every 1-5 years, the vendors introduce new versions of DBMS products. These usually contain major changes, and compatibility with the current database instance and databases need to be tested in QA environment before making the version upgrade into the production environment.

Sometimes the new versions require changes in system tables, or even to database structures. In that case, the vendor may deliver migration test utilities, which check and advice if the migration of the database contents to the format supported by the new version is possible, or if a transformation procedure is to be used.

**Reverse Engineering**

So called Lower CASE database modeling tools, see Figure 8 above, allow also reverse engineering in database administration. This means that we can apply the tool to present graphical presentation of the database structures, usually ER-model of a selected sample of the base tables or all base tables, and their relationships. This helps to get current view of the structures and in updating documentation of the database structures. These tools make it also possible to generate series of SQL DDL commands, which can be used to copy the structures to some other database. An example of reverse engineering using SQL Server tools is presented in Appendix 3.

Reverse engineering topics are discussed in the DBTech Reverse Engineering Lab, held in 2009 at ATHEI Thessaloniki, and using DB2 and SQL Server on Windows and the free Oracle Data Modeler 1.5 as the Lower CASE tool. Material of theory and tasks are available from following links

Part III – Advanced Administration Topics

**Distribution**
This subject area includes Replicated and Distributed Databases, Cluster solutions, Partitioning, and use of Stand-by servers. Replication subject area has been covered in DBTech Replication workshop materials.

**Application Server Management**
As an example JBoss has been used in DBTech ORM workshop

**Data Warehousing Management**
On management and tools of Data Warehouses we refer to Chapter 19 in the book of Mullins and textbook of Connolly and Begg
This subject area has been covered in DBTech Data Warehousing workshop.

**XML and Large Objects**
XML has been covered in DBTech XML and Databases VLW and tutorial at http://www.dbtechnet.org/labs/xml_lab/XMLandDatabasesTutorial.pdf

**Technology Trends**
Business Intelligence
Application Servers
Cloud Computing
Mobile Computing, this subject area has been covered in DBTech Android workshop.
Review Questions and Exercises

Q1 Explain the concept of Database Server Instance. What are the main differences between the DB2, Oracle, SQL Server, and MySQL instances in terms of software architecture?

Q2 Why transaction logs are the most important files of a database?

Q3 Explain main usages of indexes.

Q4 Explain the difference between concepts of clustering index and clustered index.

Q5 Explain what fragmentation means, how it can be detected, and what can be done to cure the situation.

Q6 Explain the purpose of Bufferpool (Data Cache) and checkpoint operations.

Q7 Explain purpose of schema objects.

Q8 Explain purpose of authentication and authorization.

Q9 Explain purpose of system tables.

Q10 Explain purpose of concurrency control.

Q11 Explain purpose of database backups and transaction log archiving.

Q12 Explain purpose of tablespaces.

Q13 Explain the concept and purpose of row ID (RID, ROWID, TID).

Q14 Explain purpose of collecting database statistics.

Q15 Why we might need reorganization of a table?

Q16 Explain steps in a table reorganization.

Note to instructors: Mullins [1] provides many review questions at the end of every chapter.
**Hands-on Labs to know your DBMS**

As major hands-on labs we suggest that you verify the operations presented in appendices using SQL Server, or find out how to apply corresponding tasks using DB2 Express-C or Oracle XE using our free downloadable virtual labs, for example from [http://www.dbtechnet.org/download/DebianDBVM05.zip](http://www.dbtechnet.org/download/DebianDBVM05.zip)

DBA is the expert in technical questions concerning the used DBMS products. An easy reading for DBA and application developers, is the book “SQL Performance Tuning”, by Gulutzan and Pelzer [5] reporting of supported SQL features in the “Big Eight” DBMS products in 2002, starting from Installation Parameters including COLLATE options, and best practices of coding in SQL language in these products, explaining in generic terms concepts like storage solutions, and problems like fragmentation, etc.

We extend the tests of JOINs and Referential Integrity (RI) Constraints in a set of scripts in file [http://www.dbtechnet.org/labs/dba_lab/JOINs_and_RULEs.txt](http://www.dbtechnet.org/labs/dba_lab/JOINs_and_RULEs.txt) for possible comparisons of the supported features by yourself. You may get surprised to see that, for example, all RI rules of the ISO SQL standard are supported in some of the Big Three DBMS products, but none of them will support them all.

DBA is the guarantee expert on instance wide concurrency problems of applications, as consultant and as the solver of acute problems. We explain the typical concurrency problems and concurrency control mechanisms of the DBMS products in our tutorial on SQL Concurrency Technologies and its accompanying hands-on labs.

Index Design is one of the major performance tuning options and expertise of a DBA. Basics on these topics are found in our Index Design Tutorial and its accompanying hands-on labs.

Backup and Recovery are the most important duties of a DBA, and basics of these are explained in our Backup and Recovery tutorial, with its accompanying hands-on labs.
References and Links


Web sites of interest
Oracle products, manuals, tutorials, etc at http://otn.oracle.com
DB2 LUW v9.7 Information Center at http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/index.jsp
(Note: other versions and releases may be available by replacing the part “v9r7”)
DB2 on Campus at http://www.db2oncampusblog.com/

Appendix 1  First Steps in SQL Server Administration

SQL Server is widely used in companies. Since it runs only on Windows platforms, we cannot make it available in our free virtual computer labs. Also, the free SQL Server Express edition lacks many useful tools and features of the commercial editions. To help the readers who use the Express edition in starting its use, and to give some overview of the tools of the commercial editions, we have compiled this introduction using the developer edition of SQL Server.

A user who belongs to Administrative users can install SQL Server instance. We skip details of installing the software and the first instance, since there are many articles in Web on these. In the following we give short overview how to start the instance, create a database, define a SQL authenticated user who can create tables.

SQL Server startup

As the user who installed the software and the instance, we start the instance using SQL Server Configuration Manager, selecting the SQL Server instance and click "Start" on the alternate mouse button pop-up menu (which we just for short call as AMB menu).

For application connectivity we enable Shared Memory, Named Pipes and TCP/IP protocols for the server.
and same protocols for native clients accessing the server.

![Figure 1.3](image)

Local connections will use Shared Memory, but remote clients and JDBC connections will need TCP/IP.

**Using SQL Server Management Studio**

We start then SQL Server Management Studio (SSMS) and on "Connect to Server" form, log in using Windows Authentication to the default instance in the "(local)" computer

![Figure 1.4](image)

**Defining SQL authentication and sa user**

As default SQL Server uses only OS authentication called Windows authentication, but to keep tests simple, we show how to configure it to use also SQL authentication, which is not so safe in production use.
In the Object Explorer Selecting the instance using the AMB menu, we select "Properties"

![Object Explorer](image)

Figure 1.5

and from Server Properties form we select "Security" page, and we define the mixed "SQL Server and Windows Authentication mode", and press OK.
At this phase we need to stop and restart the instance from the Object Explorer menu using AMB menu for the instance.
After the instance is up and running again, we select instance -> Security -> Logins -> "sa" login, and from the AMB menu "Properties" proceeding to "Login Properties - sa" form, where we define password to sa login user, who is the virtual system administrator, and press OK.

Creating a new database

We could create new database from our current connection, but in the next we test also connection as sa login user. From Object Explorer "Connect" menu we select "Database Engine..." proceeding into new "Connect to Server" login form by which we now log in using SQL Server Authentication / sa user to the instance.
In this new connection we select from Object explorer "Databases" and from the AMB menu we select "New Database..." and fill "TEST" as name of the new database.
and pressing OK we return to SSMS / Object Explorer and see there our new database.

We will now create a new user login "user1"

and to keep it simple we enter "user1" as the new login name, define some password, unselect "Enforce password policy", and select "TEST" as the default database as follows.
We proceed to select "User Mapping" and selecting TEST database for user1 we select db_ddladmin as the database role in TEST for user1 as follows.
Figure 1.10

Pressing OK we will see that user1 is now listed as a user in Security / Users list for the database TEST.

Selecting "Schemas" of database TEST and "New Schema..." in the AMB menu we switch to "Schema - New" form and enter "schema1" as name of the new schema and "user1" as the schema owner.

Figure 1.11
and we still need to select Databases -> TEST -> Security -> Users -> user1
and "Properties" from the AMB menu. Entering "schema1" as the default schema for user1 we press OK.

![Database User - user1]

Figure 1.12

We can now disconnect sa's connection from the Object Explorer and start a new Connect for login user "user1".
From the Object Explorer menu we select the database which we want to access using Transact-SQL commands in a "SQL Query" pane of SSMS, and from AMB menu select "New Query"
In the appearing SQLQuery pane, we will enter SQL commands creating a small test table, insert there a row, and read contents of the table into grid in Results pane by pressing Execute on the tool menu bar.

![Image of SQL Server Management Studio](image)

Figure 1.15

Then refreshing Tables for TEST database in Object Explorer, we can open available the information of the Table T which appears as schema1.T. When opening the Columns of the table, we see the information of the column structures.

In SSMS we could also open multiple SQLQuery panes, each using connection of its own, and experiment with concurrency topics, and watch SQL Server locks in other panes, as we do in the DBTech SQL Concurrency Technologies tutorial.

**SQL Server fixed and built-in roles**

For login users following Fixed Server Roles can be granted on instance level:
but these should be granted only to DBA and some for operator. Other users or user groups only login is granted by adding the login definition in instance / Security / Users, which gives them the PUBLIC role on instance-level.

<table>
<thead>
<tr>
<th>Server-level role</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysadmin</td>
<td>Members of the sysadmin fixed server role can perform any activity in the server.</td>
</tr>
<tr>
<td>serveradmin</td>
<td>Members of the serveradmin fixed server role can change server-wide configuration options and shut down the server.</td>
</tr>
<tr>
<td>securityadmin</td>
<td>Members of the securityadmin fixed server role manage logins and their properties. They can GRANT, DENY, and REVOKE server-level permissions. They can also GRANT, DENY, and REVOKE database-level permissions. Additionally, they can reset passwords for SQL Server logins.</td>
</tr>
<tr>
<td>processadmin</td>
<td>Members of the processadmin fixed server role can end processes that are running in an instance of SQL Server.</td>
</tr>
<tr>
<td>setupadmin</td>
<td>Members of the setupadmin fixed server role can add and remove linked servers.</td>
</tr>
<tr>
<td>bulkadmin</td>
<td>Members of the bulkadmin fixed server role can run the BULK INSERT statement.</td>
</tr>
<tr>
<td>diskadmin</td>
<td>The diskadmin fixed server role is used for managing disk files.</td>
</tr>
<tr>
<td>dbcreator</td>
<td>Members of the dbcreator fixed server role can create, alter, drop, and restore any database.</td>
</tr>
<tr>
<td>public</td>
<td>Every SQL Server login belongs to the public server role. When a server principal has not been granted or denied specific permissions on a securable object, the user inherits the permissions granted to public on that object. Only assign public permissions on any object when you want the object to be available to all users.</td>
</tr>
</tbody>
</table>

Figure 1.16 (source: BooksOnline)

Figure 1.17 (source: SQL Server BooksOnline)

Login users can be mapped to database users in databases with same name or other username, and they automatically get all privileges of the PUBLIC role in those databases.

On database-level the following built-in roles can be granted to database user, and of these the db_ddladmin role gives possibility to enter DDL commands creating, for example, tables
Monitoring and Tracing

To demonstrate use of SQL Server tracing by Profiler tool we show trace of following batch of commands entered in SSMS SQLQuery pane:

```sql
INSERT INTO T VALUES (2, 'some text');
UPDATE T SET `s` = 'new text' WHERE id = 1;
SELECT * FROM T;
```

Figure 1.19

To demonstrate what kind query statistics we get from a simple SELECT command in SSMS, we enter a simple SELECT command as follows:
Figure 1.20

User1 does not see very much from the Object Explorer / Management menu, so we open a new connection to sa user and can access various Activity Monitor counters as follows
and first we get the following Activity Monitor Overview

from which we can drill-down to counters
Figure 1.23

From the SSMS Object Explorer / Management we can start studying SQL Server event Logs as follows. Then on problems we can proceed to operating system Event views.

Figure 1.24
Appendix 2  Table Reorganization Example in SQL Server 2008

The de facto database management tool in SQL Server 2008 is the SQL Server Management Studio (SSMS). Most of reorganizing tasks can be done smoothly using wizards of SSMS, but sometimes we may get error message telling that the task is not possible. To give the idea of possible steps in reorganizing a table using Transact-SQL commands and some SQL Server utilities, we consider reorganizing the parent table of the following pair of tables.

```sql
CREATE TABLE Parent ( 
    pid    INT NOT NULL CONSTRAINT PK_Parent PRIMARY KEY, 
    pv     INT, 
    ptx    CHAR(4), 
    filler CHAR(3000) CONSTRAINT DF_Parent_filler DEFAULT 'FF' 
); 
CREATE TABLE Child ( 
    cid    INT NOT NULL CONSTRAINT PK_Child PRIMARY KEY, 
    cv     INT, 
    fk     INT, 
    CONSTRAINT FK_Child_Parent FOREIGN KEY (fk) REFERENCES Parent (pid) 
); 
```

Like all mainstream DBMS products, SQL Server creates automatically UNIQUE Index for the primary key having the name of the explicit constraint name of the primary key, but as default this index is clustered index, which means that rows of the table build the leaf level of the index, so in fact the table lays in the index. As part of our example, we want change the clustered index to be based on column pv instead of column pid so that the rows are stored in the order of column pv.

Let us consider how the steps of our generic Reorganization plan of a table can be applied in this case using SQL Server 2008:

1. Close use of the database from applications - Change the Options / State / Restrict Access to “SINGLE-USER” for the database Properties in SSMS, see Figure 2.1
2. Backup the database, so that in case of problems it is always possible to restore the database into the original state
3. Disable transaction logging - This is not supported in SQL Server, but we can set the recovery mode in Figure B1 options to “simple”
4. Export the data from the table to be reorganized, sorting according to the possible cluster key of clustered or clustering index
5. Disable foreign key constraints of possible child tables
6. Drop indexes of the table
7. Truncate contents of the table to be reorganized, or drop the table and the related database objects (views and triggers) and re-create the table (including possible clustered index) and the related database objects.

8. Import or load the data back to the table.

9. Re-create indexes of the table.

10. Enable foreign key constraints of possible child tables.

11. Backup the database.


13. Enable transaction logging – In SQL Server the recovery model should be “full” in production use.

14. Allow use of the database to applications – Change the Options / State / Restrict Access back to “MULTI-USER”.

In the following we will experiment with steps 4-12 in a small database in which we have created the tables Parent and Child:

For the beginning we need some test data in our database, so we insert following rows to get some disorder:

```
INSERT INTO Parent (pid, pv) VALUES (21, 0);
INSERT INTO Child (cid, cv, fk) VALUES (1, 0, 21);
```
and using the following Transact-SQL procedure we will generate an additional sample of test data

```sql
CREATE PROCEDURE ParentGen
AS
DECLARE @looper INTEGER,
        @looper2 INTEGER,
        @rand FLOAT,
        @intRand INTEGER,
        @txtRand CHAR(4);
BEGIN
    SET @looper = 1;
    WHILE (@looper < 10) BEGIN
        SET @rand = RAND() ;
        SET @intRand = CAST((1000 * @rand) AS INTEGER) ;
        SET @txtRand = CAST(@intRand AS CHAR(4)) ;
        INSERT INTO Parent (pid,pv,ptx,filler) VALUES
            (@looper, @looper, @txtRand, @txtRand);
        SET @looper = @looper + 1;
    END;
    SET @looper2 = @looper;
    SET @looper = 1;
    WHILE (@looper2 < 15) BEGIN
        SET @rand = RAND() ;
        SET @intRand = CAST((1000 * @rand) AS INTEGER) ;
        SET @txtRand = CAST(@intRand AS CHAR(4)) ;
        INSERT INTO Parent (pid,pv,ptx,filler) VALUES
            (@looper2, @looper, @txtRand, @txtRand);
        SET @looper = @looper + 1;
        SET @looper2 = @looper2 + 1;
    END;
    INSERT INTO Parent (pid,pv,ptx,filler) VALUES
        (@looper2, @looper, @txtRand, @txtRand) ;
    INSERT INTO Parent (pid,pv,ptx,filler) VALUES
        (@looper2+1, @looper, @txtRand, @txtRand) ;
END;
GO
SET NOCOUNT ON;
EXEC ParentGen;
GO
```

So now we have the following test data in our Parent table:

```sql
SELECT * FROM Parent ;
```

<table>
<thead>
<tr>
<th>pid</th>
<th>pv</th>
<th>ptx</th>
<th>filler</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>636</td>
<td>636</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>628</td>
<td>628</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>607</td>
<td>607</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>431</td>
<td>431</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>23</td>
<td>23</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>69</td>
<td>69</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>358</td>
<td>358</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>377</td>
<td>377</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>750</td>
<td>750</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>693</td>
<td>693</td>
</tr>
<tr>
<td>11</td>
<td>2</td>
<td>563</td>
<td>563</td>
</tr>
</tbody>
</table>
In the following we create a couple indexes, a view, and triggers as a sample of related objects for the Parent table:

```sql
CREATE INDEX ix_pv ON Parent (pv);
CREATE INDEX ix_ptx ON Parent (ptx);
GO
CREATE VIEW VParent (id, v, tx) AS
    SELECT pid, pv, ptx FROM Parent;
GO
```

The following dummy triggers just test the firing of these triggers using the special message print command of SQL Server:

```sql
CREATE TRIGGER trgParentUpdate ON Parent
AFTER UPDATE
AS
PRINT 'UPDATE trigger of table Parent was fired';
GO
CREATE TRIGGER trgParentInsert ON Parent
AFTER INSERT
AS
PRINT 'INSERT trigger of table Parent was fired';
GO
CREATE TRIGGER trgParentDelete ON Parent
AFTER DELETE
AS
PRINT 'DELETE trigger of table Parent was fired';
GO
```

Note: In SQL Server we can only create AFTER-command or INSTEAD OF triggers, and no row-level triggers as we can do in DB2 and Oracle.

To test the view and the triggers, we can enter the following batch of commands

```sql
INSERT INTO VParent (id, v) VALUES (22, 0);
UPDATE VParent SET v = v+1 WHERE id = 22;
DELETE VParent WHERE id = 22;
GO
```

and we will get the following results

```
INSERT trigger of table Parent was fired
UPDATE trigger of table Parent was fired
DELETE trigger of table Parent was fired
```
Latest at this step we should take a full backup of the database using the backup wizard of SSMS.

Figure 2.2 Starting Database Backup by SSMS
Note: After backup of any application database, we should also backup the system databases Master and Msdb of our SQL Server instance, as explained in our Backup and Recovery tutorial.

In case we need to recover this start state of the database, we can restore the database content using RESTORE overwriting the current content, whatever, selecting from the SSMS Object Explorer pane the <database> - Tasks – Restore – Options

and press OK to restore the old database contents.
Figure 2.5

We will now proceed with step 4 exporting the contents of Parent table using the **BCP utility**, documented in SQL Server Books Online, in a Command Prompt window, first generating format file of data to be copied as SQL Server **native data types** using `"-n"` option in the BCP commands.

![Command Prompt](image)

```
C:\>rem create format file
C:\>BCP REORG.dbo.Parent format nul -T -n -f C:\temp\Parent.fnt -S\(local\)\SQLEXPRESS
C:\>rem export data in native format
C:\>BCP REORG.dbo.Parent out C:\temp\Parent.dat -S\(local\)\SQLEXPRESS -T -n
Starting copy...
17 rows copied.
Network packet size <bytes>: 4096
Clock Time <sec.> Total : 1 Average : 17000.00 rows per sec.
C:\>type C:\temp\Parent.fnt
```

Figure 2.6
At steps 5-6 we will disable or drop all related objects, i.e. indexes, triggers and foreign keys of the referencing child tables, and re-create the table with its new clustering index:

```
DROP INDEX Parent.ix_pv ;
DROP INDEX Parent.ix_ptx ;
DISABLE TRIGGER ALL ON Parent ;
```

A referencing foreign key is dropped, and rows of Parent table deleted by commands

```
ALTER TABLE Child DROP CONSTRAINT FK_Child_Parent ;
DELETE FROM Parent ;
ALTER TABLE Parent DROP
    CONSTRAINT PK_Parent ;
ALTER TABLE Parent ADD
    CONSTRAINT PK_Parent PRIMARY KEY NONCLUSTERED (pid) ;
```

but more change options we have, if we drop the foreign key and the Parent table as follows

```
ALTER TABLE Child DROP CONSTRAINT FK_Child_Parent ;
DROP TABLE Parent ;
```

In SQL Server, this drops also related views and triggers.

We can now create the Parent table in a new form, including some data type changes

```
CREATE TABLE Parent ( pid  INT NOT NULL
    CONSTRAINT PK_Parent PRIMARY KEY NONCLUSTERED ,
  ptx  CHAR(4),
  pv   SMALLINT,
newcol VARCHAR(20),
filler CHAR(3200)
    CONSTRAINT DF_Parent_filler DEFAULT 'FF') ;
```

We can now create the new clustered index to column pv as follows

```
CREATE CLUSTERED INDEX cix_pv ON Parent (pv) ;
```

so that when we load the data back into the table, the table pages are already in the index.

The step 7 will be implemented as ORDER clause of BULK INSERT command in step 8. Since we have changed the order of the columns and added a new column, the fields in the exported bulk data file do not match with the new table structure, so before the BULK INSERT we need to create first a mapping view\(^2\), compatible with the data file, as follows

\(^2\) BULK INSERT cannot use virtual view, such as common table expression, so we need to create a mapping view, which we can drop after the BULK INSERT has been done.
CREATE VIEW BulkMapping AS
  SELECT pid, pv, ptx, filler FROM Parent;
GO

and then we can finally load the old contents by

BULK INSERT REORG.dbo.BulkMapping
  FROM 'C:\temp\Parent.dat'
  WITH (DATAFILETYPE='native',
        FORMATFILE='C:\temp\Parent.fmt',
        ORDER(pid ASC));

and drop the mapping view

DROP VIEW BulkMapping;

We can now verify the results using following query

SELECT * FROM Parent;

<table>
<thead>
<tr>
<th>pid</th>
<th>ptx</th>
<th>pv</th>
<th>newcol</th>
<th>filler</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>NULL</td>
<td>0</td>
<td>NULL</td>
<td>FF</td>
</tr>
<tr>
<td>10</td>
<td>693</td>
<td>1</td>
<td>NULL</td>
<td>693</td>
</tr>
<tr>
<td>1</td>
<td>636</td>
<td>1</td>
<td>NULL</td>
<td>636</td>
</tr>
<tr>
<td>2</td>
<td>628</td>
<td>2</td>
<td>NULL</td>
<td>628</td>
</tr>
<tr>
<td>11</td>
<td>563</td>
<td>2</td>
<td>NULL</td>
<td>563</td>
</tr>
<tr>
<td>12</td>
<td>282</td>
<td>3</td>
<td>NULL</td>
<td>282</td>
</tr>
<tr>
<td>3</td>
<td>607</td>
<td>3</td>
<td>NULL</td>
<td>607</td>
</tr>
<tr>
<td>4</td>
<td>431</td>
<td>4</td>
<td>NULL</td>
<td>431</td>
</tr>
<tr>
<td>13</td>
<td>782</td>
<td>4</td>
<td>NULL</td>
<td>782</td>
</tr>
<tr>
<td>14</td>
<td>631</td>
<td>5</td>
<td>NULL</td>
<td>631</td>
</tr>
<tr>
<td>5</td>
<td>23</td>
<td>5</td>
<td>NULL</td>
<td>23</td>
</tr>
<tr>
<td>6</td>
<td>69</td>
<td>6</td>
<td>NULL</td>
<td>69</td>
</tr>
<tr>
<td>15</td>
<td>631</td>
<td>6</td>
<td>NULL</td>
<td>631</td>
</tr>
<tr>
<td>16</td>
<td>631</td>
<td>6</td>
<td>NULL</td>
<td>631</td>
</tr>
<tr>
<td>7</td>
<td>358</td>
<td>7</td>
<td>NULL</td>
<td>358</td>
</tr>
<tr>
<td>8</td>
<td>377</td>
<td>8</td>
<td>NULL</td>
<td>377</td>
</tr>
<tr>
<td>9</td>
<td>750</td>
<td>9</td>
<td>NULL</td>
<td>750</td>
</tr>
</tbody>
</table>

(17 row(s) affected)

Part of re-creating the table related objects of step 6 is implemented in step 9 which in case of SQL Server re-creates the missing indexes, enables or re-creates the foreign key references from the child tables as follows:

CREATE INDEX ix_ptx ON Parent (ptx);
ALTER TABLE Child ADD CONSTRAINT FK_Child_Parent
  FOREIGN KEY (fk) REFERENCES Parent (pid);
ALTER TABLE Child CHECK CONSTRAINT FK_Child_Parent; -- enabling
CREATE INDEX ix_fk ON Child (fk);

In case we dropped the Parent table, the triggers were lost, and we need to create them again, but if we only deleted rows from the table, then we can enable them by following
ENABLE TRIGGER ALL ON Parent;

After updating statistics as follows

UPDATE STATISTICS Parent WITH ALL;
UPDATE STATISTICS Child WITH ALL;

the systems has following information and statistics available for the optimizer

and selecting Statistics – cix_pv - Properties – Details we will see following index statistics
Finally we need to set recovery model of database to "full" and we need to take a new full backup of the database and the system databases Master and Msdb, and after that allow use to applications by setting the database state / remote access to "MULTI-USER".

The collected statistics provide information for the SQL optimizer of the DBMS product for generating optimal execution plans for SQL requests of the applications. Explain plan functionalities of DBMS products make it possible DBA and application developers to see the generated execution plans of requested SQL commands for evaluate if the optimizer has selected proper indexes and access paths for the execution plan. The following example presents in graphical form the execution plan of SQL Server for a simple query of ours. The tree of the

```
SELECT pid, cid, cv
FROM Parent P JOIN Child C
ON P.pid=C fk ;
```

Figure 2.9. An Execution Plan generated by SQL Server
execution plan is evaluated from right to left. By moving cursor to some node in the tree we will get a pop-up tooltip on yellow background presenting the operation and statistics of that particular node.

**Appendix 3  SSMS as a Lower CASE Tool**

SQL Server Management Studio (SSMS) combines the facilities of an administration console, a SQL editor for interactive SQL of an application developer, and a Lower CASE tool of database designer and DBA for implementation of the database design and for reverse engineering the design from the existing database. However, the free SSMS Express lacks some functionalities of the commercial edition.

**Implementing a new table**

Start selecting the Tables node of your database in Object Explorer using the alternate mouse button (AMB) and select “New Table…”

![Figure 3.1](image)

and you get to a design pane presenting a form for entering column definitions as shown in Figure 3.2.

Let us continue defining the following columns

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Allow Nulls</th>
</tr>
</thead>
<tbody>
<tr>
<td>cid2</td>
<td>int</td>
<td></td>
</tr>
<tr>
<td>cv2</td>
<td>int</td>
<td>√</td>
</tr>
<tr>
<td>fk2</td>
<td>int</td>
<td></td>
</tr>
</tbody>
</table>

and press Close button in the right-hand side upper corner of the form for finishing the column definitions, then accepting saving of these changes in the database, and replacing name “Child2” instead of the suggested name “Table_1”.


Now open the Tables node in Object Explorer and with AMB select the new table “dbo.Child2”. Proceed selecting “Design…” from the pop-up menu presented in Figure 3.3.
Figure 3.3 Pop-up menu of table access activities

On the presented columns design form select column definition for cid2 using AMB and from the pop-up menu presented in Figure 3.4 select “Set Primary Key” for cid2.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Allow Nulls</th>
</tr>
</thead>
<tbody>
<tr>
<td>cid2</td>
<td>int</td>
<td></td>
</tr>
<tr>
<td>cu2</td>
<td>int</td>
<td></td>
</tr>
<tr>
<td>fi2</td>
<td>int</td>
<td></td>
</tr>
</tbody>
</table>

Figure 3.4 Pop-up menu for defining column constraints, etc
Reverse engineering a part of the database

SSMS can present selected parts of an existing database in diagram form, which can be used as documentation or editing, for example, the relationships between the tables. We start our reverse engineering of our sample database selecting Database Diagrams node with AMB and “New Database Diagram” from the pop-up menu presented in Figure 3.5.

![Figure 3.5](image)

Select all 3 tables from the Add Table form and press Add button.

![Figure 3.5 Add Table form](image)

You can arrange the tables on the diagram form as you like, and with AMB you get facilities presented in Figure 3.6.
You can proceed to define the missing foreign key of table Child2 referring to the Parent table as presented in Figure 3.7.
Generating Transact-SQL DDL commands

Reverse engineering of structures of an existing database can be started selecting “Tasks” from the AMB-menu of the database, and then “Generate Scripts...” as presented in Figure 3.8, and choosing the objects to be generated.

![Figure 3.8 Start generating DDL commands](image)

![Figure 3.9 Choose Objects for the DDL to be generated](image)
Figure 3.10 Scripting Options for the DDL to be generated

Figure 3.11 Saving and Publishing the generated DDL

And finally we can access the generated Transact-SQL DDL commands as presented in Figure 3.12. To be able to show the whole script in snapshot, we have deleted settings of ANSI_NULLS and QUOTED_IDENTIFIER preceding the view and child tables.
Figure 3.12  The generated DDL commands for objects of our sample database

Note that a general practice is to generate foreign key definitions as ALTER TABLE commands after all CREATE TABLE commands have been generated. There is a good reason to work like this, but what is the reason?
Index

Activity Monitor, 41
ALTER, 23
AMB menu, 29
archiving, 22
authorization, 13
Backup, 48
baseline, 20
baseline workload trace, 22
BCP utility, 50
blocking process, 21
bufferpool, 6
BULK INSERT, 51
catalog, 12
cluster, 7
cluster segment, 10
clustered index, 51
database, 6
database backup, 22
Database Diagram, 58
database environment, 5
database server instance, 5
database user, 39
DBMS version, 17
disaster recovery, 22
discretionary access control, 13
dynamic cache views, 11
event log, 43
execution plan, 54
EXPORT, 18
extent, 10
filegroup, 7
firewall, 18
fixpack, 17
foreign key, 51
fragmentation, 23
GRANT, 14
hit ratio, 21
IMPORT, 18
index allocation, 17
indexes, 17
instance configuration, 6
instance processes, 6
Kerberos authentication, 13
listener, 6
LOAD, 18
log buffer, 6
login name, 33
login user, 39
mandatory access control, 14
Master, 7
migration, 24
operator, 14
optimizer hints, 22
OS authentication, 13
page, 9
partitioning, 17
performance, 20
pre-fetching, 10
privileges, 13
procedure, 46
Profiler, 40
PUBLIC, 14, 39
query statistics, 40
quiescent state, 7
read-only state, 7
record, 9
recovery, 22
recovery model, 54
reorganization, 23
response time, 20
role, 14
row address, 9
ROWID, 9
sa, 13
schema, 11
segment, 10
SQL authentication, 13
SQL session, 12, 14
SSMS, 30
statistics, 53
system databases, 7
TempDB, 7, 17
throughput, 20
tooltip, 55
top priorities, 5
TPS, 20
transaction log, 9

transaction log files, 17
transaction logging, 23, 44
trigger, 47
Unicode, 20
UNLOAD, 18
views, 14
virtual computer, 20