

ADABAS D

CONCEPTS AND FACILITIES



Manual Order Number: ESD611-006ALL

This document is applicable to ADABAS D Version 6.1.1 PE and to all subsequent releases, unless otherwise indicated in new editions or technical newsletters.

Specifications contained herein are subject to change and these changes will be reported in subsequent revisions or editions.

Readers' comments are welcomed. Comments may be addressed to the Documentation Department at the address on the back cover.

© April 14, 2025, SOFTWARE AG, Germany & SOFTWARE AG of North America, Inc.

All rights reserved

Printed in the Federal Republic of Germany

The SOFTWARE AG documentation often refers to numerous hardware and software products by their trade names. In most, if not all cases, these designations are claimed as trademarks or registered trademarks by their respective companies.

1 Introduction

This document provides the reader with a comprehensive overview of the facilities and capabilities of ADABAS D. After reading it through, the reader will be in a position to assess the benefit of ADABAS for his Information Technology (IT) organization.

ADABAS offers the following advantages compared with other SQL systems:

- **Easier use**
- **Better performance**
- **Better cost/benefit ratio**

ADABAS makes 'State of the Art' database technology available to its users and redefines this technology in sub-areas.

ADABAS is designed as an open system. A variety of software products from other companies can use ADABAS as server database system. It is thus possible to integrate it simply into the existing IT environment. For example, the openness allows access from a Windows-based development tool to ADABAS.

There is an extensive catalog of cross-industry solutions which contains industry-neutral applications (e.g., bookkeeping) as well as a variety of industry-specific applications.

In this area, SOFTWARE AG cooperates intensively with software partners. These partners possess comprehensive know-how in specific application areas and offer excellent software solutions with ADABAS as a basis.

2ADABAS D

Simplicity

ADABAS is designed for simple operation. This starts with the configuration for which only very few, easily understandable parameters have to be determined. Database operation is supported in a uniform way by the menu-driven administration tool CONTROL. There are no tedious and erroneous size estimations for tables and indexes in ADABAS. Nor are there overflows of internal database areas resulting from false estimates. Only the total size of the database is specified. The individual tables and indexes increase and decrease dynamically without requiring administration tasks such as the definition of tablespaces or extents. For the user, this means: silent operation without permanent supervision. Even the balanced load distribution to the disk drives available to ADABAS is done implicitly, i.e., without analysis and administrative intervention.

High Availability

ADABAS D is designed for operation 7 days a week and 24 hours a day. This high degree of availability is possible because the backup of data sets, the administration of database objects and most changes to the configuration can be carried out while the database is operating and because ADABAS does not require any reorganization. Application programs that are running react immediately to a new index structure or to the withdrawal of privileges. It is not necessary to run the precompiler again or to reopen a session for this purpose.

Storage Space Optimization

The space requirement for stored data is reduced by dynamic storage space management as well as by data compression. The required logging space is reduced to a minimum compared to conventional page-oriented logging because ADABAS employs row-oriented and field-oriented logging. As a result, ADABAS requires considerably less disk space in comparison with other SQL systems.

Performance

The performance of an SQL system is largely determined by the secondary storage organization, the transaction management and the locking granularity, the optimizer, and the scalability of the system on multiprocessor configurations.

As a secondary storage organization, ADABAS uses B* trees which provide a balanced performance for read and write operations and do not degenerate in most application situations. In addition, the B* trees in ADABAS are used to provide the user with a clustering in primary key order which maintains this property even when updates are performed.

The transaction management is based on user locks being managed on row level as a rule. Group commits and a buffering of the log up to the end of the transaction minimize the I/O requirements for the logging. At the end of a transaction, there is no writing-through from the data buffer.

The ADABAS optimizer is cost-based and works with statistics on the size of the database objects and on value distributions. Whereas in other SQL systems the processing strategy is usually determined at prepare time, ADABAS determines the strategy at execute time on the basis of the current parameter assignment. To be able to do this, several alternative processing strategies are compiled at prepare time from which the most favorable is selected at execute time.

The scalability on multiprocessor systems means that ADABAS can convert a surplus of CPU performance into extra database performance. This is not self-evident because access to global data has to be synchronized, i.e., sequentialized, within the system. The multi-threaded/multi-server architecture of ADABAS allows the full exploitation of multiprocessor systems.

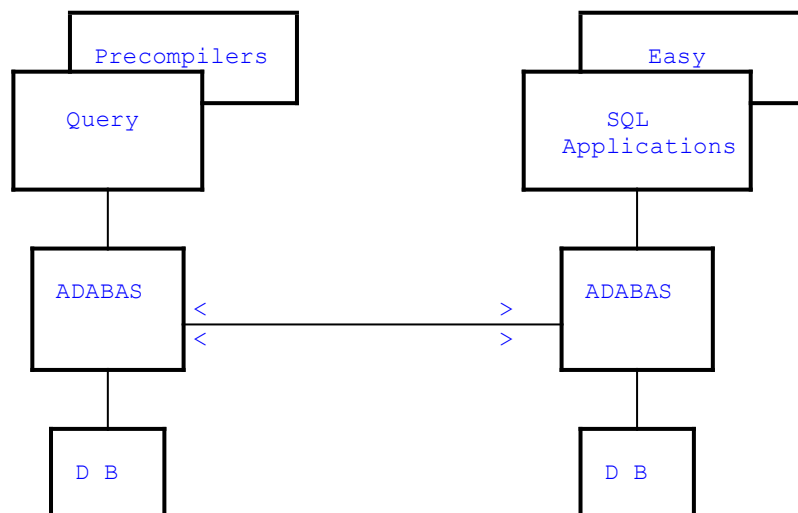
Fault Tolerance

ADABAS offers fault-tolerant concepts to prevent hardware breakdowns. MIRRORRED DEVSPACES can be set up to protect against disk errors. Table replications in a distributed configuration guarantee protection against computer breakdowns.

3Distribution

3.1 Logical View of the Distribution

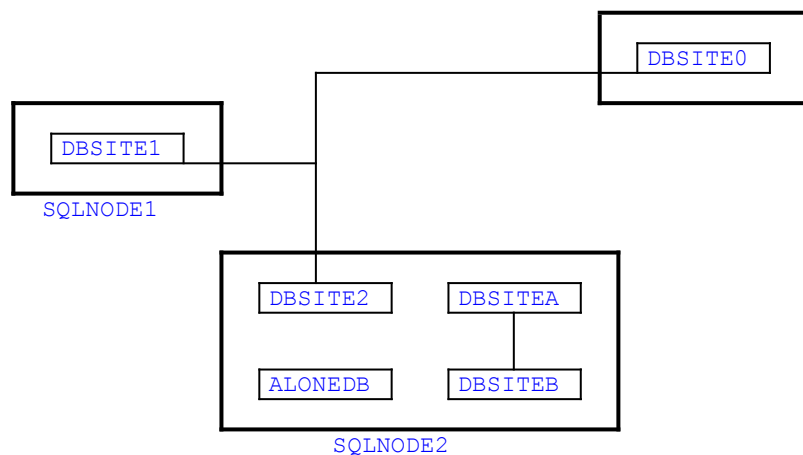
From the user's point of view, a distributed ADABAS behaves in exactly the same way as a central ADABAS. The user has the illusion that all the data available in the distributed database is located on the node with which he has opened his database session. SQL applications or ADABAS user tools which access or modify non-local data implicitly lead to distributed operations within the network without the user noticing it. To do this, of course, it is technically necessary that the ADABAS systems (SERVERDBs) communicate with the nodes involved in these operations.



3.2 Physical View of the Distribution

A local database is called a "SERVERDB". Its node in the network is called "SERVERNODE".

With the aid of a network, ADABAS allows a co-operation between distributed SERVERDBs. As a rule, there is one SERVERDB on each SERVERNODE. For various reasons, however, (test/production, definition of separately managed databases), several SERVERDBs can be installed on one SERVERNODE.



The following configuration table shows the structure of the databases depicted above.

Configuration table:

Database	Type	SERVERDB	SERVERNODE
1	distributed	DBSITE0 DBSITE1 DBSITE2	SQLNODE0 SQLNODE1 SQLNODE2
2	distributed	DBSITEA DBSITEB	SQLNODE2 SQLNODE2
3	stand alone	ALONEDB	SQLNODE2

3.3 Client-Server Operation and Distributed Databases

Client-server operation is very different from the operation of distributed databases. The following overview should make this clear:

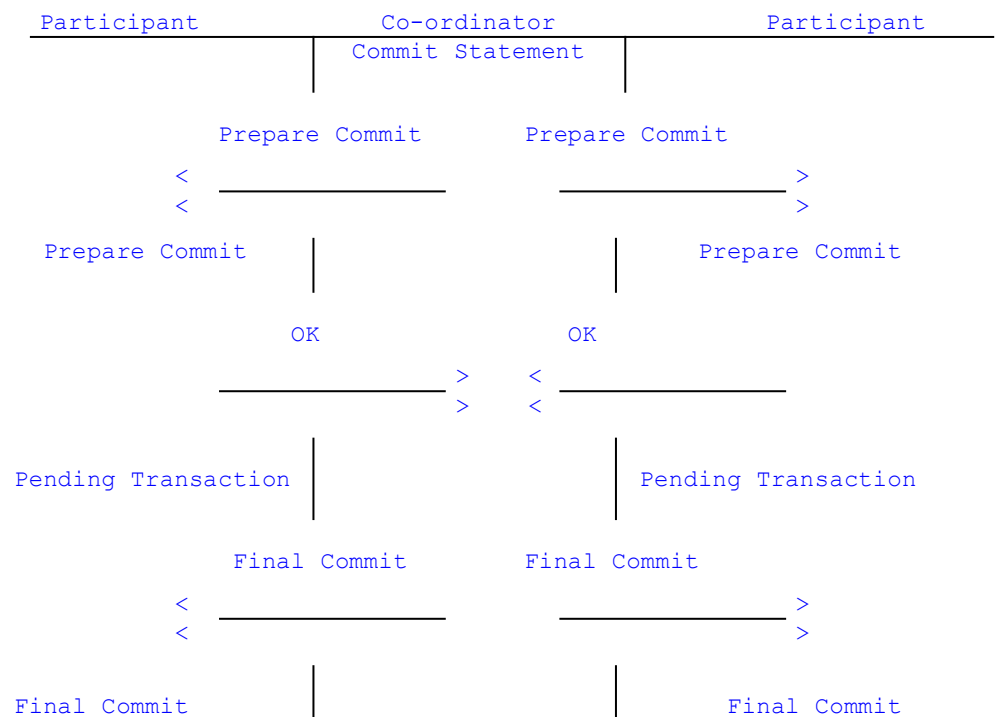
Property	Client-Server	Distribution
Access to several SERVERDBs from an application program	YES S	YES
Access is visible in the application logic	YES	NO
Application program knows the storage location (SERVERDB) of the data object	YES S	NO
One command can operate on data in several SERVERDBs	NO	YES
One command can modify data in several SERVERDBs	NO O	YES
Two-phase commit protocol	NO	YES
Global catalog of all database objects in all SERVERDBs	NO	YES S

On the basis of these properties, it is apparent that a distributed database system makes the user completely independent of the physical location of the database objects in the network and has the same properties as a logically centralized system. Client-server configurations are useful to separate an application server from a database server and make it independent of the load. For multi-server operations,

however, client-server configurations are not a good idea because the user generally is not able to cope with them. Here it is better to exploit the capabilities of a distributed database system because otherwise, the consistency problems of distributed data have to be solved on the applications level.

3.4 Two-Phase Commit Protocol

The two-phase commit protocol serves to manage distributed transactions among several SERVERDBs. The application session is linked with a SERVERDB which is called the "co-ordinator" and manages all transactions of this session. Other SERVERDBs involved in the distributed transaction are called "participants".



Most importantly the two-phase commit protocol drastically shortens the time required to solve problems by tedious communication between the co-ordinator and/or the participants. The "commit" of the application is dealt with by the co-ordinator in two phases. Up to the "prepare commit" of the co-ordinator, the participants can roll back the transaction independently because it cannot have been completed on any of the other SERVERDBs. When a SERVERDB responds to the

"prepare commit" with "OK", this transaction is put into an undecided state (pending transaction) from which it can no longer be released by this participant alone. If the co-ordinator receives an "OK" response from all participants, it sends the "final commit" to all participants in the next phase. The transaction is thus completed successfully. If one of the participants has problems and reports "NOT OK", the co-ordinator will send a rollback statement to all participants thus aborting the transaction. If the co-ordinator or some of the participants stop functioning, extensive protocols are necessary to decide on the result of a pending transaction. These protocols are executed automatically when a SERVERDB is restarted in a distributed configuration.

In the entire network there is no central co-ordinator. This role is assumed by the local SERVERDB with which the application program holds its database session.

3.5 Synchronous Replication

ADABAS supports replications of a table on all SERVERDBs in a distributed database. This can be useful in the case of static data because network communication is reduced in this way. Another reason for keeping replications is to secure against breakdowns and to improve availability. Replications are synchronously maintained by ADABAS; i.e., they are identical in the entire network at commit time.

In contrast to other database systems, ADABAS allows replications of tables to be modified even if not all SERVERDBs are available. This is achieved by means of a majority concept which accepts modifications to replicated data as long as the majority of SERVERDBs are available. In this way, the consistency of replications can be guaranteed even in the case of network partitions because the smaller, separated sub-network may not modify these replications.

With these properties of ADABAS, the consistency of replications is no longer a problem that has to be dealt with by application programming or database operating.

3.6 Asynchronous Replication

Not every application environment needs replicated data to be synchronized at commit time. In many cases larger update intervals can be tolerated, e.g., when a frozen snapshot is required for management reporting or decision support systems.

For this reason ADABAS supports "snapshots". Snapshots, like views, describe portions of the database but, unlike views, they are materialized and typically kept on another SERVERDB. In the simplest case a snapshot comprises a part of a table. Like views, snapshots can also be defined via joins of several tables.

To update the snapshot contents, there is a REFRESH command which must be issued explicitly (e.g., from an application program). This has the advantage that consistency level and authorization can be chosen when a snapshot is created and a detailed error handling can be ensured. Most of the operating systems (e.g., UNIX) allow such REFRESH programs to be executed automatically in certain intervals (e.g., daily). When data of simple snapshots (i.e., those built from one table) is updated, only the modifications are transferred. For complex snapshots, the data is transferred completely.

Snapshots are read-only and cannot be modified. Their update (REFRESH) is not subject to the logging. After a database recovery, a new REFRESH is therefore required to produce the current snapshot state.

3.7 The Twelve Rules for Distributed Databases from Date

C. J. Date formulated twelve rules or criteria as a catalog of requirements for a distributed database system. The starting point is the requirement: "To the user, a distributed system should look exactly like a non-distributed system". ADABAS satisfies this requirement.

Rules:

1. LOCAL AUTONOMY

All SERVERDBs of ADABAS are locally autonomous; i.e., their local database and operating functions are independent of the state of the other SERVERDBs. This holds also for backup and recovery processes. Local actions of a SERVERDB are independent of the availability of another SERVERDB.

2. NO RELIANCE ON A CENTRAL SITE

There is no "master" SERVERDB which has global knowledge and thus represents a "single point of failure". In ADABAS, all SERVERDBs have equal status and each has a copy of the distributed global catalog.

3. CONTINUOUS PROCESSING

This rule aims at ensuring that it is not necessary, e.g., to shut down the entire network to extend the distributed database by adding a new node. ADABAS ensures this.

4. LOCATION INDEPENDENCE

With ADABAS, users do not have to know which SERVERDB keeps a certain database object. The navigation necessary to access non-local data is implicitly carried out by ADABAS without the user noticing it.

5. FRAGMENTATION INDEPENDENCE

ADABAS provides the possibility of fragmenting tables vertically. For this purpose partial tables are formed which can be recombined into a full table by means of updatable join views without the user having to know the different storage locations. The possibility of horizontal fragmentation of a table will be provided in future versions of ADABAS.

6. REPLICATION INDEPENDENCE

ADABAS provides the possibility of having replications kept by the system on all SERVERDBs of the distributed database without the user having to ensure the consistency of these replications.

7. DISTRIBUTED QUERY PROCESSING

For queries to ADABAS, the selected data is found without specifying its storage location and is joined together. The ADABAS optimizer minimizes these data transfers by selecting a processing sequence for complex statements that reduces the size of the intermediate results to be transferred to a minimum.

8. DISTRIBUTED TRANSACTION MANAGEMENT

For reasons to do with consistency of transactions, ADABAS uses the two-phase commit protocol described above for distributed transactions.

9. HARDWARE INDEPENDENCE

ADABAS is available on the most important hardware platforms.

10. OPERATING SYSTEM INDEPENDENCE

ADABAS is available on the most important operating system platforms. A distributed database can be implemented on various operating systems.

11. NETWORK INDEPENDENCE

ADABAS supports a connectivity via TCP/IP (sockets). Further protocols (ISO, LU 6.2, XTI, IPX/SPX, etc.) are supported in conjunction with the SOFTWARE AG product ENTIRE NET-WORK.

12. DBMS INDEPENDENCE

The aim is that a distributed SQL system consists of nodes in which different SQL systems are employed. At present, however, this aim cannot be achieved because it requires a standardization unifying the various manufacturers. A start has been made with DRDA from IBM and ODBC from Microsoft as well as the work of the SQL Access Group. But it will still be several years before distributed heterogeneous SQL systems can operated sensibly.

3.8 Location-independent Access

With ADABAS, users do not have to know which SERVERDB keeps a certain database object. The navigation necessary to access non-local data is implicitly carried out by ADABAS without the user noticing it.

This distinguishes ADABAS considerably from other SQL systems and their distribution facilities. In ADABAS, the usual two-level naming

`<owner>.<table name>`

is also sufficient in a distributed configuration because the system knows where this table is located and where the catalog entries for this owner are. In other SQL systems, database objects in a distributed configuration have to be given three-level names

`<serverdb>.<owner>.<table name>.`

This means nothing less than that the problem of accessing distributed database objects is referred back to application programming and is not made available as a system facility. Even if in other SQL systems this three-level naming is hidden by SYNONYM or ALIAS objects, manual catalog modifications are necessary in all SERVERDBs when shifting a table from one SERVERDB to another, whereas in ADABAS no administrative work is required.

Three-level naming thus means that knowledge about the distribution of the database objects is kept only in the heads of the database administration but not in the system. Genuine distributed database systems with a global catalog require only two-level naming, the same as central systems require.

It is only two-level naming which means that there is a single-database distribution. Three-level naming implies a multi-database distribution, which means a loose combination of the databases where the problems of catalog management are left to the user.

3.9 Gateways

A distributed ADABAS configuration allows other database systems to be integrated. The gateway facility of ADABAS hides the existing differences and makes a foreign database system look like an ADABAS SERVERDB to the rest of the network.

A gateway is offered for DB2 (MVS version). In this gateway the connectivity to the mainframe is based on TCP/IP. (For some UNIX systems it is also possible to use a connectivity that is based on LU 6.2.) This allows decentralized UNIX or NT systems to read- and write-access DB2 data as well as ADABAS data within one transaction.

Another gateway supported between ADABAS D and ADABAS C (MVS version) allows ADABAS D and ADABAS C data to be read- and write-accessed within one transaction. In this case, even the more complex record structures of ADABAS C (periodic groups and multiple fields) are made accessible through SQL.

Using these gateways, ADABAS allows great flexibility in extending the existing IT infrastructure.

3.10Connectivity

ADABAS has a built-in connectivity for client/server and server/server configurations based on TCP/IP (sockets).

No add-on products or components (except the TCP/IP socket protocol stack) must be acquired, installed, and maintained.

4SQLMODE

In spite of SQL standardization by SQL-89 and SQL-92, there are still considerable differences in the extent of the SQL language between the SQL systems available on the market. For the portability of an application program, essential parts (e.g., the most important return messages to be dealt with by the application program) have not yet been sufficiently taken into account by the standardizations currently available.

To provide more options to our users, ADABAS supports the most important SQL dialects by selecting an SQLMODE:

- **ADABAS**
- **ANSI**
- **ORACLE**
- **DB2**

The SQLMODE determines the extent of the SQL language, the structure and significance of SQLCA and SQLDA, data types supported, as well as the numbering of the most important return messages. The aim is to be able to take over existing application systems which were written for another SQL system without modifying them by selecting the suitable SQLMODE. Or, as in the case of SQLMODE DB2, to be able to use an ADABAS platform to develop DB2 applications. The SQLMODE can be set for each session.

ADABAS

This is the most attractive SQLMODE because it is the most powerful. Facilities exceeding the standard and performance spectrum of other SQL systems are only available in this SQLMODE. For the user, this means maximum productivity combined with optimal performance.

ANSI

For users who want to achieve the highest possible portability of their applications, the restriction to the entry level of SQL-92 can be selected with this SQLMODE. ADABAS will then only accept SQL statements which conform to the ANSI standard. The fulfillment of the ANSI standard by ADABAS is assured by the United States National Institute of Standards and Technology (NIST) test suite against which ADABAS D was certified in October 1994.

This option is particularly interesting for software companies which have to make their products available on various relational database systems and for whom the standardized facilities suffice.

ORACLE

With this SQLMODE, ADABAS understands ORACLE-specific SQL extensions; e.g., for built-in functions or dynamic SQL. The point of reference for ORACLE compatibility is ORACLE7.

This option is interesting for software companies that have developed a program on the basis of ORACLE and want to make it available on ADABAS. The porting of the application in this case requires only that the application sources be precompiled.

For users who want to introduce a two-company policy for database suppliers in their organization, this is an interesting perspective.

DB2

SQLMODE DB2 provides the compatibility with DB2 V 3.1.

This is particularly interesting for users who have made a strategic decision in favor of DB2 (e.g., as data warehouse) and who pursue a consistent database strategy in their organization in the area of decentralization.

With this SQLMODE, a DB2-compatible database system that is ideally suited to decentralized systems in the area of administration can be implemented also on PC and UNIX platforms.

5Platforms

ADABAS D is available for the following platforms:

- **HP-UX**
- **SUN Solaris**
- **IBM AIX**
- **DEC OSF/1**
- **SNI Sinix**

- **Windows NT**
- **Windows 3.1 (as clients only)**
- **OS/2**

The support of further platforms for specific projects is possible.

6SQL Extensions

6.1Data Types

ADABAS D supports the following data types:

CHAR (N) N ≤ 254

For every SERVERDB, a default code, ASCII or EBCDIC, is specified which does not have to agree with the machine code. This default can then be overridden in each table column, as required. It is thus possible, e.g., to store all data on an ASCII computer in an EBCDIC coding. This may be desirable to avoid differing sorting sequences in a client-server configuration for CHAR values with upper and lower cases.

For transparent storage, there is, apart from ASCII and EBCDIC, the CHAR variant BYTE. In client-server configurations, contents from CHAR (N) BYTE columns are not converted implicitly into the code of the client.

VARCHAR (N) N ≤ 254

The internal storage of CHAR values is performed up to a length of 30 in a fixed format. Larger CHAR values are stored implicitly internally in a representation of varying length. By explicitly using the data type VARCHAR, even shorter CHAR values can be represented internally using varying length.

BOOLEAN

Often only the states "available/not available" or "true/not true" have to be distinguished for attributes. This can be done by CHAR(1) columns which have been set to appropriate values. But the embedding in a programming language such as C/C++ or COBOL is simplified by providing a boolean data type. It is easier to formulate WHERE qualifications with Boolean columns.

DATE

Here, an internal format is stored as YYYYMMDD. The external representation can be configured according to the usual conventions in Europe, North America, or in the Pacific region.

TIME

TIME values are kept internally in the form HHHHMMSS. As in the case of DATE values, it is possible to configure for different external representation. DATE and TIME values are special CHAR values. This means that, apart from the date and time functions, all CHAR functions can be applied to them.

TIMESTAMP

TIMESTAMP values are kept internally in the form
YYYYMMDDHHMMSSmmmmμμμ.

They are a combination of a DATE and TIME value extended by the specification of milliseconds and microseconds. In addition to their usage as a timestamp, it is also convenient to use them for time arithmetics, because then overflows in the day's portion need not be handled by the user. TIMESTAMP values are special CHAR values. This means that, apart from the date and time functions, all CHAR functions can be applied to them.

FIXED (N,M)

For numeric values, a decimal fixed point representation with a maximum of 18 digits is provided.

FLOAT (N)

Apart from the fixed point representation by *FIXED*, a decimal floating point representation is available with a maximum precision of 18 places and a value range of 10 .

LONG

For storing non-formatted data (BLOB), ADABAS provides the data type *LONG* which can accept data of up to 2.1 GB per column. As with *CHAR* columns, the variants *ASCII*, *EBCDIC*, and *BYTE* are supported. ADABAS is thus in a position to manage extensive text, image, and voice information in an appropriate way.

The additional data types occurring in other *SQLMODEs* are understood by ADABAS and mapped onto those described above.

6.2LONG Columns

To support application programming with non-formatted data sets (texts, graphics, voice, images), ADABAS provides the data type LONG.

LONG columns are specified in the usual way when defining a table (CREATE TABLE). They are completely subject to the transaction concept, the authorization, and the dynamic memory management of ADABAS. For each table, several LONG columns can be created.

Within the SQL statements INSERT, UPDATE, DELETE, and SELECT or FETCH, LONG columns can only be operated as containers. This means that in the application program, a host variable that is sufficiently long must be defined in which the entire content of the LONG column can be read in or out.

LONG columns can grow dynamically and be created, updated, and deleted during database operation. Saving and restoring LONG columns is done with the usual ADABAS backup and recovery utilities.

6.3 Data Integrity

The integrity of data is supported in ADABAS by declarations using powerful DDL statements and options.

Integrity rules for tables:

PRIMARY KEY	is the primary key of a table. ADABAS ensures the uniqueness of the primary key.
UNIQUE	is a set of columns whose values uniquely identify a row in a table.
NOT NULL	is a mandatory column. ADABAS ensures that this column is always set to values.
DEFAULT	assigns a default value specific to a column.
CHECK	The CHECK definition checks the row of the table affected by INSERT and UPDATE. In a CHECK rule, a WHERE condition can be formulated for this row which has to be satisfied.
FOREIGN KEY	Referential integrity constraints between tables are expressed by specifying the foreign key (inclusive ON DELETE).

In addition to value lists or value ranges, the CHECK rule allows complex conditions to be formulated which can also refer to other columns in the same row.

Referential integrity in ADABAS is also supported in a distributed configuration when the tables involved are located on different SERVERDBs. Hardly any other SQL system offers such a feature.

To be able to formulate extensive business data models with ADABAS, domain definitions are supported as well as table definitions. Domains are user-defined data types which can be used in CREATE TABLE statements instead of the predefined data types (CHAR, FIXED, FLOAT, ...). By using domains, integrity rules can be defined on the level of data elements and their use in the definition of tables helps in modelling the data uniformly and consistently.

Example:

```
CREATE DOMAIN itemno CHAR(12) CHECK LIKE '????.???.'

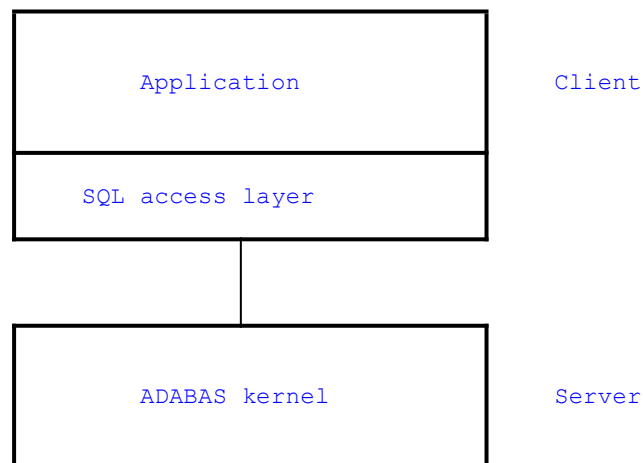
CREATE TABLE item(
    itno          itemno,
    itname        CHAR(30) NOT NULL,
    entrydate     DATE NOT NULL,
    purchaseprice FIXED(6,2) NOT NULL
                CHECK purchaseprice > 0,
    sellprice     FIXED(8,2) NOT NULL
                CHECK sellprice > purchaseprice,
    PRIMARY KEY (itno))
```

Data integrity in ADABAS can be ensured in most cases by means of declarative integrity rules. Only in exceptional cases is it necessary to formulate procedural triggers to monitor highly complex integrity conditions. In general, the use of declarative rules has advantages over a procedural formulation via triggers because ADABAS ensures, when an integrity rule is modified, that the existing data set corresponds to the new integrity rule. Such a check cannot be carried out for procedurally formulated rules.

6.4DB Procedures and Triggers

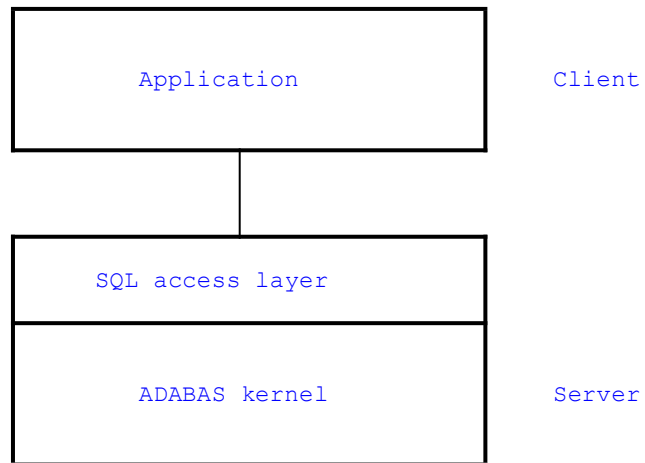
In an ADABAS application with a modular structure, the SQL statements are typically not distributed over the entire application but are concentrated in a single access layer. This access layer has a procedural interface with the rest of the application at which the typical operations for application objects are made available.

For example, insert, update, delete, and select operations are realized on the application object CUSTOMER where the rest of the application does not need to know the number of tables that represent this application object and which integrity checks are performed when modifying customer information.



In client-server configurations, there is an interaction between client and server when executing any SQL statement in the access layer.

The number of these interactions can be drastically reduced when the SQL access layer is no longer run in the client but in the server. ADABAS provides a language for this purpose which allows an SQL access layer to be formulated on the server side.



This has three important advantages:

- The number of interactions between client and server is reduced by several factors. Client-server communication is only required for each operation on the object layer, no longer for each SQL statement. This enhances the performance of client-server configurations considerably.
- The second advantage has to do with software engineering. The SQL access layer contains the procedurally formulated integrity and business rules. The concentration of these rules on the server side and their elimination from applications means that these rules can be updated centrally so that they become immediately effective in all applications. In this way, these integrity and decision rules also become part of the database catalog.
- An SQL access layer relocated to the server side also allows for creating client-specific database functionality and is thus an important customizing tool.

All suppliers of SQL DBMS are moving technically in the direction of supporting DB procedures and triggers. But with respect to the individual offerings a very close look must be taken at the procedural completeness of the DB procedure language (goal: full programming language) and of the testability of new DB procedures (goal: test environment independent of the DB kernel, preferably 4GL level). In ADABAS, as an additional feature, the code of a DB procedure may run optionally on the client or on the server. This simplifies the development and testability decisively, and characterizes the DB procedure basically as centralized application code.

With the exception of the data type LONG, all data types supported by ADABAS can be used in DB procedures for input/output parameters. It is also possible to pass the name of a cursor (i.e., of a result table) as input and output parameters. This allows a result to be selected in the form of a table within a DB procedure and it allows the result rows to be retrieved by a sequence of FETCH statements outside the DB procedure.

The author or owner of a DB procedure can authorize other ADABAS users to employ this procedure (EXECUTE privilege). This leads to an implicit privilege extension of the other users; i.e., they can execute operations on database objects with a DB procedure for which they are not privileged. In this way, it can be ensured that the manipulation of certain database objects is only possible with DB procedures and no longer with the SQL language.

Triggers are special DB procedures. They possess the same procedural power but are not called explicitly; they run implicitly as a consequence of an INSERT, UPDATE or DELETE operation in a table. This provides a general user-exit mechanism which allows the user to formulate further arbitrary actions on a database in addition to the normal statement semantics. Among other things, triggers permit the formulation of complex integrity rules, the checking of complicated access protection conditions, and the execution of implicit database modifications.

In ADABAS, triggers are generally performed after the execution of the DML statement. Within the trigger, access is given to the before and after image of the table row by qualifying OLD or NEW for the corresponding column. This is an

essential prerequisite for the creation of consistency or integrity rules which concern modifications of such a column.

In case of mass INSERT, UPDATE, or DELETE statements, one SQL statement modifies more than one table row. When doing so, it must be decided whether a trigger has to be activated for each table row concerned or only once for the SQL statement. ADABAS supports both options; statement-oriented triggers are formulated by a specific trigger locking.

Apart from a development environment and the testability of DB procedures and triggers, the SQL extensions described in the following in the form of temporary tables and subtransactions are required for their practical application.

6.5 Temporary Tables

ADABAS permits the user to define temporary tables which exist up to the end of the session. Temporary tables are implicitly deleted at the end of a database session.

Example :

```
CREATE TABLE TEMP.delivery
(seq_no          FIXED(5),
customer        CHAR(20),
product         CHAR(20),
address         CHAR(30),
PRIMARY KEY(seq_no)) IGNORE ROLLBACK
```

Temporary tables are particularly suitable as auxiliary tables which are only required for the duration of an application execution. These tables have no overhead for catalog administration because they are user-specific. They can be operated optionally with or without logging (IGNORE ROLLBACK).

Especially in DB procedures, temporary tables can be used to create large intermediate results which can be accessed via SELECT and FETCH.

6.6 Subtransactions

The usual transaction concept (COMMIT, ROLLBACK) ensures that related modifications of several data objects are performed completely or not at all.

When DB procedures are called from an application, the user would like to have the same behavior as for SQL statements. If it is successful, the DB procedure should perform all the database actions. If it is not successful, it should not leave any effect in the database. This can only be realized with the help of subtransactions. A DB procedure must not contain any COMMIT or ROLLBACK because otherwise it would intervene unexpectedly in the transaction control of the application program. To be able to reset the effects of a DB procedure (and only these effects) if there is an error, the DB procedure must be formulated as a subtransaction. For this purpose, ADABAS provides the commands SUBTRANS BEGIN, SUBTRANS END and SUBTRANS ROLLBACK. If it is successful, a DB procedure is concluded with SUBTRANS END. If error situations arise which require a resetting of the DB procedure's effects, this can be achieved by means of SUBTRANS ROLLBACK without this influencing the top-level transaction control of the application program. Since DB procedures can be called in a nested way, the nesting of subtransactions within subtransactions is also supported.

Apart from the use of subtransactions for programming DB procedures and triggers, they are also necessary to program library functions containing SQL statements with a proper error handling; that is, without influencing the transaction control of the main program which invoked the library function.

6.7 Array Statements

Apart from DB procedures and triggers, ADABAS also supports arrays as host variables to enhance performance in client-server configurations in the statements INSERT, UPDATE, DELETE, SELECT, and FETCH. Thus it is possible to process not only one row of a table but a large number of rows with one SQL statement. This reduces the interaction between client and server.

In the case of FETCH statements, the attempt is always made to transfer more than one row of the table into the application program. No special programming is required.

6.8 Isolation Levels

ADABAS provides various locking levels with respect to read consistency, specifically for each session: so-called isolation levels.

ADABAS	ANSI	Lock Mode
0	0	read uncommitted
1, 10	1	read committed
15	-	consistent select
2, 20	2	repeatable read
3, 30	3	serializable

Isolation level 0 makes so-called "dirty reads" possible, i.e., the access to database objects that are just being modified by another user in an uncommitted transaction. Since no share locks are set implicitly, isolation level 0 allows the highest degree of parallel operation in OLTP operation. In order to compensate for the possible effects of a "dirty read", a check read should be done or the optimistic locks described below should be used.

For the isolation level 1 or 10, ADABAS implicitly sets a share lock on every table row currently being read, thus preventing access to modifications of transactions that have not yet been completed. This share lock is kept until the next read command for the same table is issued; i.e., for each table a share lock is set to one row at the most.

With respect to the handling of locks during the processing of mass commands, isolation level 15 is an extension of the isolation level 1 or 10. The tables addressed are locked in share mode for the duration of command processing. In this way, the tables involved cannot be modified while the commands are being processed.

For the isolation level 2 or 20, mass operations are first secured by a table share lock as in isolation level 15. In addition, all the rows read are locked in share mode. These locks are only released at the end of the transaction.

By implicitly setting table share locks on all tables processed up to the end of the transaction, the isolation level 3 or 30 ensures that database users only see those modifications they themselves have made.

By means of options in a SELECT statement, it is possible to work in a specific isolation level but to achieve a higher or lower read consistency with respect to multiuser effects.

In all isolation levels, updates of table rows lead to exclusive locks for these rows up to the end of the transaction.

6.9 Optimistic Locking

For simplified programming of OLTP applications, ADABAS provides optimistic locks. They make it possible to write an application for operation in isolation level 0 without the usual repetitive check reading being necessary.

For this purpose, an optimistic lock is demanded when accessing a database object. When updating the same database object, the application program receives a message if, in the meantime, other users have made modifications to this database object. Otherwise, the update is executed without the application having to verify the database object with a check reading.

Optimistic locks do not preclude share locks or exclusive locks.

6.10 Transaction Chaining

The opposing requirements of consistency and maximum concurrency must be harmonized for the processing of master-detail structures in OLTP applications.

If, for example, all items ordered by a particular customer are locked during the processing of an order operation, all the other order operations will be blocked. If the customer row is released after each order item, it can happen that the input of order items that are still open is blocked by the simultaneous processing of other orders made by the same customer.

For this reason, ADABAS provides a transaction chaining via the KEEP LOCK option in the COMMIT statement. Such a transaction chaining allows a user to perform a COMMIT at the end of a transaction and to extend simultaneously some of the locks (typically those on the master row) to the next transaction.

6.11 Updatable Join Views

In SQL systems, a view is a table that exists only virtually and whose contents are defined by a SELECT statement referring to permanent tables (base tables) or other views.

If at least two base tables are used in the FROM clause of the SELECT statement of a view definition, it is called a join view.

Whereas SQL systems usually do not permit modifying operations (INSERT, UPDATE, DELETE) in join views, these can be done with ADABAS. In this way, the view concept is extended considerably and enables, for example, attribute migration between tables without this having effects on existing application programs. Even complex application objects consisting of several tables can be made available with updatable join views.

The following constructs are not permitted in updatable join views:

- Subquery
- GROUP BY or HAVING
- DISTINCT
- UNION, INTERSECT, or EXCEPT
- Outer Join

The view must be defined WITH CHECK OPTION and contain the primary key columns of all tables involved. For master-detail (1:N) structures, a FOREIGN KEY must be defined between the tables involved.

A join view which satisfies these rules is updatable. In a distributed configuration, the tables involved in a join view can be stored in different SERVERDBs.

6.12 Outer Join

With the usual join of two tables, the result table contains only those rows for which the join condition is satisfied. The outer join also permits rows to be included in the result table for which there are no corresponding rows in the second table. These kinds of rows are supplemented in the outer join by NULL values in the columns which, properly speaking, should come from the other table. When joining two tables, the NULL value can be added for the "left" or the "right" table or also for both.

To be able to execute outer joins over more than two tables with a clearly defined semantics, ADABAS offers the possibility of formulating a further SELECT instead of a table name in the FROM part of the SELECT statement. In this way, in the case of outer joins, the processing sequence is laid down implicitly.

6.13 Scrollable Cursors

In cursor processing, ADABAS supports more than the usual FETCH logic which runs sequentially through the SELECT result once. In addition to FETCH NEXT as default, FETCH PREV, FETCH FIRST, FETCH LAST, FETCH POS, and FETCH SAME are also supported. It is thus possible to run through the same SELECT result several times and, above all, to formulate easily and directly a backward scrolling in results. Anyone who has tried to program this backward scrolling with the usual means available in other SQL systems will be thankful for this facility.

There are positioned accesses not only within SELECT results. Also when accessing individual rows of a table with a primary key definition, ADABAS supports the SELECT variants SELECT NEXT, SELECT PREV, SELECT FIRST, SELECT LAST, and SELECT DIRECT. The specification of direction refers to the primary key order. In this way, application programming based on the primary key order is supported more directly and more efficiently than would be possible using the usual SQL constructs (ORDER BY).

6.14 Built-in Functions

ADABAS provides an extensive range of built-in functions, most of which can be employed in the SELECT statements for qualification or data formatting.

Basic arithmetical operations

`+, -, *, /, DIV, MOD`

Operations on sets of values

`COUNT, MAX, MIN, SUM, AVG, STDDEV, VARIANCE`

Arithmetical functions

`TRUNC, ROUND, FIXED, CEIL, FLOOR, SIGN, ABS,
POWER, SQRT, LENGTH, INDEX, COS, SIN, TAN, COT,
COSH, SINH, TANH, ACOS, ASIN, ATAN, ATAN2,
RADIANS, DEGREES, EXP, LN, LOG, PI`

Functions on character strings

`SUBSTR, || (concatenation), & (concatenation),
LFILL, RFILL, TRIM, LTRIM, RTRIM, EXPAND, UPPER,
LOWER, LPAD, RPAD, MAPCHAR, INITCAP, REPLACE,
TRANSLATE, ALPHA, ASCII, EBCDIC, SOUNDEX`

Date functions

`ADDDATE, SUBDATE, DATEDIFF, DAYOFWEEK, DAYOFMONTH,
WEEKOFYEAR, DAYOFYEAR, MAKEDATE, YEAR, MONTH, DAY,
TIMESTAMP, DAYNAME, MONTHNAME`

Time functions

`ADDTIME, SUBTIME, TIMEDIFF, MAKETIME, HOUR,
MINUTE, SECOND, TIME, MICROSECOND`

Generic functions

VALUE, GREATEST, DECODE, LEAST

Conversion functions

NUM, CHR, HEX, CHAR

With these built-in functions, extensive and complex calculations or evaluations can be performed very easily on the ADABAS database. This saves a major part of conventional application programming.

6.15DB Functions

ADABAS allows a user to extend the effects of the SQL statements with user-specific DB functions. In addition to the above-mentioned built-in functions, it is possible to define own user-specific functions for use in SELECT statements or WHERE qualifications. This allows more application logic to be shifted to the database server.

6.16 Special Characters

As a European SQL system, ADABAS puts special emphasis on supporting the various special characters in the individual European alphabets (e.g., the German umlauts).

These special characters present two sorts of problems: their representation on the terminal and their internal sorting.

For presenting these special characters, conversions are necessary because not all hardware manufacturers use the ISO-ASCII character set. ADABAS therefore provides configurable conversion tables (TERMCHAR SET) which can be activated specifically for each session depending on the type of terminal. The correct input and output and the correct internal storage can thus be achieved even in a heterogeneous hardware, terminal, and PC environment. This is an essential requirement, especially when operating distributed configurations.

To be able to sort these special characters according to the user's conventions and not according to the often arbitrary internal codes, ADABAS provides configurable sorting tables. In these, it can be specified, for example, that for the German "ü" a sorting according to "ue" is desired. With the built-in function MAPCHAR, a virtual column can be generated from the stored data which sorts the data set in the desired manner.

6.17 Authorization

ADABAS provides the user with a comprehensive authorization concept that supports four functional user classes and column-oriented access rights. It is thus possible to generate an individual partial view on the data set for each user and to protect the data from unauthorized access and modifications.

ADABAS distinguishes between four classes of user:

- **SYSDBA**
- **DBA**
- **RESOURCE**
- **STANDARD**

In addition to the rights of a DBA, the SYSDBA has the right to create users with the status DBA on his SERVERDB.

Users with DBA status can create users and usergroups with RESOURCE and STANDARD status, create private data and pass on privileges to other users. A DBA also has the right to create table replications on all SERVERDBs in a distributed configuration and to define snapshots. The user status DBA includes all rights which a user with the RESOURCE status has.

RESOURCE users can define their own tables, views, and synonyms and pass on privileges for these objects.

STANDARD users may define views and synonyms but otherwise may only execute operations on data for which they have been privileged.

Several RESOURCE or STANDARD users can be grouped together by their DBA into a user class. This makes the administration of privileges easier because all members of a usergroup obtain the same rights with respect to SQL authorization.

Privileges are granted to users or usergroups with the GRANT statement and are withdrawn with REVOKE. Privileges relate to tables, views, columns, and DB procedures. With views, it is also possible to formulate privileges which depend on the database contents (value-dependent privileges).

The following privileges relating to database objects can be granted:

SELECT (column list)
INSERT
DELETE
UPDATE (column list)
SELUPD (column list)
INDEX
ALTER
REFERENCES
EXECUTE

The privileges SELECT, INSERT, DELETE, UPDATE, and SELUPD relate to the corresponding SQL statements. INDEX allows the use of CREATE/DROP INDEX; ALTER the use of ALTER TABLE, and REFERENCES the reference to a table in the REFERENCES clause of a table definition. With EXECUTE, the right to call a DB procedure can be passed on to other users. ADABAS supports the WITH GRANT option with which the recipient of a privilege is able to pass on this privilege.

DBAs can restrict the use of resources by database users for whom they are responsible. With PERMLIMIT and TEMPLIMIT, the allocatable disk space can be restricted. COSTWARNING and COSTLIMIT prevent expensive SQL queries. With the CONTROL function ACCOUNTING, the use of resources per user can be recorded and accounted precisely.

6.18Catalog Access

ADABAS provides extensive possibilities for obtaining information about the static and dynamic aspects of the database objects.

This is done using system tables which are provided in the form of views and which build the SQL catalog. These system tables can be accessed via SELECT statements. Descriptions of all currently defined database objects and information about their space requirements can be retrieved. Further system tables are offered, in addition, containing data obtained by monitoring the current database operation.

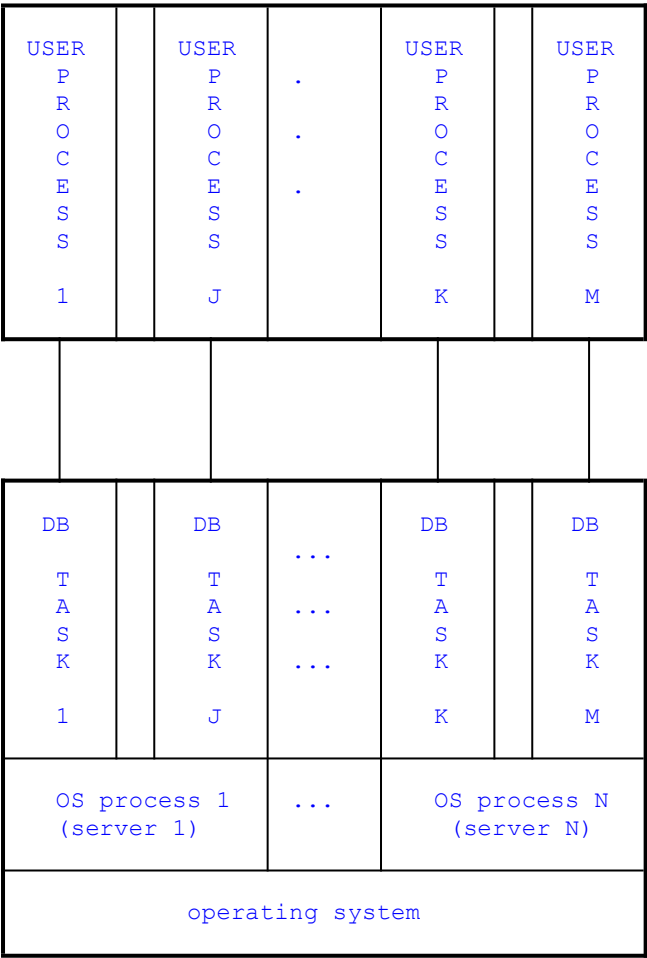
Special end user tools and development environments adapt themselves to the current database structure by accessing the SQL catalog and by using dynamic SQL statements. Their support therefore requires differently structured SQL catalogs for the different SQLMODEs. In SQLMODE ORACLE, for example, there are system table views for ORACLE7.

7 Operating System Embedding

The ADABAS server consists of the database kernel, the tool and administration components, as well as the operating system embedding. By means of this system embedding, the differences between the operating systems are neutralized so that ADABAS can be run in principle on a variety of operating systems with very different properties. The main platforms for ADABAS D are open systems, i.e., UNIX, Windows NT, and OS/2 operating systems.

The system embedding makes sure that ADABAS is compiled from the same source code on all the platforms that are supported, thus displaying an identical behavior.

The ADABAS operating system embedding puts a multi-threaded/multi-server process structure into effect:



When opening a database session, every application process (user process) is assigned to a fixed database task (DB task). As a rule, to save resources, this DB task is realized not by an operating system process but by an internal tasking within an operating system process.

On computers with one CPU only, it is convenient to operate as many DB tasks as possible (e.g., 200-300) by an internal tasking in an operating system process (server process). In multi-processor machines, there should be at least one operating system process (server process) with internal DB tasks per CPU so that the computer's capabilities can be fully exploited.

For the optimum use of various hardware configurations, ADABAS offers therefore a process structure in which the number of operating system processes and the degree of internal tasking can be adjusted. As limiting cases, either all DB tasks can be located in a single operating system process, or each user process is assigned to exactly one operating system process. All the mixed forms between these extremes can also be configured.

In line with the ADABAS goal of providing an easy-to-operate SQL system, an optimal process and task configuration which depends on the number of CPUs available is implicitly determined during installation. Only in exceptional cases will it be necessary to alter this initial setting.

Literature

Concepts and Facilities

Training Programs

Concepts and Facilities

8ADABAS D and Its Tools

CONTROL	TOOLS	PRECOMPILERS								
Installation	<table><tr><td rowspan="3">QUERY</td><td>SQL</td></tr><tr><td>Form</td></tr><tr><td>Report</td></tr></table>	QUERY	SQL	Form	Report	PRECOMPILER FOR				
QUERY	SQL									
	Form									
	Report									
Configuration		- C/C++								
Statistics		- COBOL								
Monitoring	<table><tr><td rowspan="3">EASY</td><td>QbE</td></tr><tr><td>Form</td></tr><tr><td>Report</td></tr></table>	EASY	QbE	Form	Report	CALL Interface (ODBC)				
EASY	QbE									
	Form									
	Report									
Restart										
Shutdown	<table><tr><td rowspan="5">LOAD</td><td>Export</td></tr><tr><td>Import</td></tr><tr><td>Migration</td></tr><tr><td>Cross Load</td></tr><tr><td></td></tr></table>	LOAD	Export	Import	Migration	Cross Load		<table><tr><td>Form</td></tr><tr><td>Report</td></tr></table>	Form	Report
LOAD	Export									
	Import									
	Migration									
	Cross Load									
Form										
Report										
Save/Restore										
AutoSave	<table><tr><td rowspan="5">ADABASIC</td><td>DB Procedures</td></tr><tr><td>Triggers</td></tr><tr><td>DB Functions</td></tr><tr><td>SQL</td></tr><tr><td></td></tr></table>	ADABASIC	DB Procedures	Triggers	DB Functions	SQL				
ADABASIC	DB Procedures									
	Triggers									
	DB Functions									
	SQL									
Weekly Schedule										
Backup Media	<table><tr><td rowspan="7">OfficePlus</td><td>Word</td></tr><tr><td>Excel</td></tr><tr><td>ACCESS</td></tr><tr><td>VisualBasic</td></tr><tr><td>Add-ons</td></tr><tr><td>ODBC</td></tr><tr><td></td></tr></table>	OfficePlus	Word	Excel	ACCESS	VisualBasic	Add-ons	ODBC		
OfficePlus	Word									
	Excel									
	ACCESS									
	VisualBasic									
	Add-ons									
	ODBC									
	<table><tr><td rowspan="2">DOMAIN</td><td>DBA Tool</td></tr><tr><td>Dictionary</td></tr></table>	DOMAIN	DBA Tool	Dictionary						
DOMAIN	DBA Tool									
	Dictionary									

ADABAS Kernel

ADABAS consists of the following components:

- **Database kernel**
- **Operating tool CONTROL**
- **Loading tool LOAD**
- **Administration tool DOMAIN**
- **End user tool QUERY**
- **End user tool EASY**
- **End user tool OfficePlus**
- **Programming tool ADABASIC**
- **Precompilers for C/C++ and COBOL**
- **CALL Interface (ODBC)**
- **ODBC driver (Windows)**

The facilities of these tools is described briefly in the following.

The approved ADABAS tools QUERY, EASY, and LOAD are available both as alpha clients and as Windows clients. Thus the user can choose whether to use them in UNIX or Windows environments.

9TOOLS

9.1QUERY

QUERY provides an interactive SQL interface as well as a report generator (REPORT).

SQL statements are entered directly on the screen by means of QUERY. The result is shown as a table in a standard format. It can be scrolled through on the screen and output on a printer or in a file.

With the report generator REPORT, result tables can be put into the desired output format step by step. After each individual REPORT command, the report will be displayed in the current format.

QUERY offers application programmers the opportunity of interactively testing the SQL statements that they want to use in programs.

Frequently recurring queries are stored as parameterized SQL statements together with a sequence of REPORT commands. When called via an input form or a command name, they provide a completely prepared report as the result.

Call privileges can be granted to other users for such stored commands. Via the LIST function, QUERY provides an individual menu of all stored commands which a user can currently call.

By virtue of the stored commands, QUERY becomes an attractive tool also at the departmental level because knowledge of SQL is no longer required. In addition, it is possible to call stored commands directly from the operating system level and thus to execute commands with long runtimes in batch operation.

QUERY provides a high level of support: The executed commands for each session are held in a command history and can be called again at any time. The language for the prompts and the most important characteristics of the standard format can be set individually.

REPORT displays the prepared report on the screen, writes it into a file, or outputs it on a printer. When printing, various page formats can be defined.

For master-detail structures, it is possible to display them simply in a report and to page back and forth on the master level as well as on the detail level.

Calculation fields are supported in reports: e.g., to accumulate results on the group level. Cross tables can be built and evaluated in subsequent queries.

Stored QUERY commands and REPORT can also be called from application programs. It is therefore not necessary to invent a vertical and horizontal scrolling within result tables for every application. The results of the calculation fields defined in a report can be transferred to the application program after a REPORT call.

9.2EASY

EASY is an interactive user interface of the database for end users and for sporadic users.

It enables requests to individual tables, the maintenance and administration of tables, and the creation of new (typically private) tables in the database. The individual access rights are respected when doing so.

EASY possesses an extremely simple user guidance and can be operated without any SQL knowledge. All functions are selected via menus and function keys. The language for the guidance texts in the forms can be selected by the user.

In EASY, requests are made according to the QUERY-BY-EXAMPLE principle: The user selects the table to which the request refers from a menu. He then sketches the desired result in a form that EASY creates dynamically for the table. Each user can define the operator symbols he wishes to use to formulate requests.

The result of a request is displayed either in file card format or in tabular format. The user can switch from one format to the other at any time and determine the current position with the cursor.

New entries can be inserted into the table in both formats. The displayed entries can be modified by overwriting or deleted by pressing a key.

In tabular format, the REPORT functions can be employed for preparing the table for printing. They are made available in EASY not via commands but via a convenient menu control. The display of master-detail structures with the associated scrolling feature is also supported in EASY.

Apart from forms in standard format, user-defined forms can be created with EASY.

Frequently used requests with subsequent REPORT formatting can be stored as stored requests which can be re-executed at any time.

9.3LOAD

LOAD supports the loading and extracting of data sets and catalog contents. In addition, it is an important instrument for installing ADABAS and can be sensibly employed for the distribution of SQL applications.

- Loading tables from external files (DATALOAD)
- Update loading individual table columns (DATAUPDATE)
- Providing database extracts in external files for further processing (DATAEXTRACT)
- Migration of databases and tables to other computers (TABLEEXTRACT)
(TABLELOAD)
- Migration of the database catalog to other computers (CATALOGEXTRACT)
(CATALOGLOAD)
- Execution of batch files with SQL and LOAD statements and data flow control (command file)

LOAD processes its own statements and SQL statements. These statements can be recorded as a command sequence in a command file and executed in batch mode. Flow control statements and reactions to return codes are possible in a command file.

LOAD can also be operated interactively. Statements and test data can be input via an editor. Syntax errors are displayed and can be corrected immediately. Command files can be developed and tested statement by statement.

The consistency of the database defined in the schema is ensured also during LOAD runs: Rows which would impair this consistency are rejected, collected in a protocol file, and provided with error comments. Apart from a row-wise loading

(DATALOAD), a page-oriented loading (FASTLOAD) is also supported which runs with a considerably enhanced performance. Here, too, the defined integrity conditions are checked.

LOAD provides a high level of support when converting input values from the external data format into the data type which was determined in the table definition. Various external data formats can also be selected when making the data available.

During the LOAD runs, the tables involved remain available for multi-user operations. The transaction behavior of LOAD can be set.

The integration of ADABAS into the existing IT environment of a company and the automation of administrative measures when operating database applications are made considerably easier with LOAD. For example, LOAD supports the import of ORACLE databases on the basis of the EXPORT format of ORACLE.

9.4OfficePlus

In client-server configurations ADABAS supports Windows clients and makes the access to the ADABAS data available to them.

This is mainly achieved by the ODBC driver for ADABAS which makes SQL access possible from the usual Windows applications and Windows development environments.

For Microsoft Office users, there is in addition the ADABAS tool OfficePlus which improves the SQL integration of the Office tools such as Word, Excel, ACCESS, and Visual Basic.

A user can, for example, link a Word mail merge document or an Excel spreadsheet directly to a SELECT statement and transfer the current SQL data simply by clicking a button.

For Microsoft ACCESS, a more convenient way of selecting SQL tables to be processed is provided, as well as the ability to issue DDL statements for ADABAS.

Visual Basic is extended by default forms for tables and master-detail structures and an SQL control to ensure convenient SQL programming above the ODBC-API level.

9.5 DOMAIN

DOMAIN is the Windows-based DBA tool of ADABAS which provides information about the static and dynamic properties of the defined database objects. It offers all ADABAS DDL facilities for the creation of new database objects and the maintenance of the existing database objects in a menu-driven way. Essentially, the following database objects can be administered with DOMAIN (create, update, show, drop, comment):

- Tables
- Views
- Snapshots
- Synonyms
- Domains
- Indexes
- Triggers
- DB procedures
- DB functions
- Users
- Privileges

Access to DOMAIN information is also valuable for application programmers because they can inform themselves very easily about the structure and properties of the database objects they work on.

As for all database objects, usage information is implicitly maintained by ADABAS; DOMAIN represents the data dictionary associated with ADABAS. For example, it can be determined very easily which application programs use a certain table or column. This usage information is of greatest advantage for the maintenance of the database objects because the implications of modifications can only be assessed on the basis of this information.

9.6CONTROL

CONTROL is the convenient database operation tool of ADABAS which can basically be used to carry out the following tasks:

- Initialization and configuration of a SERVERDB
- Restart/Shutdown
- Backup and recovery functions
- Operation monitoring and control
- Performance monitoring and control

The installation and configuration of a SERVERDB requires only a few parameters. Apart from the definition of a few special users, basically only the disk areas allocated to the SERVERDB have to be specified.

For backing up the database contents, ADABAS provides the option of making a backup copy of the entire database while the database is in operation. This forms the basis for any necessary recovery operations if media failures occur.

The changes in the database contents since the last complete backup can be recorded in one or several log backups. These log backups can also be done in parallel to database operation. ADABAS supports log segmentation and the backup of completed log segments independently of database operation.

As an alternative to log backups, ADABAS also offers an incremental database backup which covers only the pages modified since the last complete or incremental backup. This is of interest when the database modifications affect only parts of the database.

In addition to the automatic backup of completed log segments to tape, CONTROL supports the definition of weekly schedules which ensure that the whole database or log is backed up in regular intervals. For this purpose, the number of desired backup generations must be determined and the number of required backup media must be identified and administered.

For large databases, CONTROL provides parallel backup and recovery by allowing simultaneous writing to or reading from several tape devices. The recovery time for large databases then depends only on the capacity of the largest disk and the number of used tape devices, and no longer on the total size of the database.

CONTROL facilitates performance and operation monitoring by automatically displaying information about the utilization levels of database and log, the number of database sessions, the hit rate in the data cache, and the collisions in the lock management.

9.7 ADABASIC

ADABASIC is the procedural SQL extension of ADABAS for the creation of DB procedures, triggers, and DB functions.

The ADABASIC language corresponds to Visual Basic. Using the ADABASIC workbench, individual basic subprograms (subs) can be created as DB procedures, triggers, or DB functions out of the Visual Basic development environment. These subprograms will then no longer be executed on the client but on the server.

Using the Visual Basic development environment considerably facilitates the creation, test, and especially the debugging of DB procedures, triggers, and DB functions. Simply by clicking a button, a Visual Basic subprogram can be shifted from the client to the server, and vice versa, for execution. This allows the user to test these sensitive subprograms on the client until they can be considered sufficiently stable to run without problems on the server side.

As a rule, DB procedures operate with the privileges of their authors, not those of the callers. Thus users of DB procedures may be allowed to do more than they would be able to do on the SQL level. Certain tables may be made accessible only via DB procedures and no longer on the SQL level (data encapsulation). Only DB procedures containing dynamic SQL which are marked accordingly run with the privileges of the particular caller.

9.8PRECOMPILERS

The SQL standard defines the embedding of SQL statements in a programming language such as C/C++ or COBOL. This is also the only interface at which the different SQL systems offer a certain degree of portability.

For this purpose, variables for database communication must be defined in the application program (DECLARE SECTION). The ADABAS precompiler also allows this DECLARE SECTION to be created implicitly. Apart from the elementary data types usual in the DECLARE SECTION, ADABAS also supports the use of records and arrays, thus simplifying application programming considerably.

The ADABAS precompilers provide the option of executing dynamically generated SQL statements and make a macro mechanism available in addition.

The report generator REPORT and the stored commands in QUERY can also be called from programs.

Programming is supported by convenient error handling routines and by extensive trace functions of the SQL statements. The tuning of SQL applications is made easier by an SQL profiling that identifies SQL statements which are frequently executed or require long runtimes.

Precompilers exist for the programming languages C/C++ and COBOL.

Apart from these precompilers, ADABAS operation from other programming languages is supported by a CALL Interface. The definition of the ADABAS CALL Interface corresponds to the ODBC standard.

For Windows development environments (e.g., Visual Basic, PowerBuilder, or SQLWindows), an ODBC driver for ADABAS is provided.

10 Documentation

Concepts and Facilities

Questions and Answers

Tutorial

Installation under UNIX

Installation under Windows NT

Installation under Windows

Installation under OS/2

User Manual for UNIX

User Manual for Windows NT

User Manual Windows Clients

User Manual for OS/2

QUERY

EASY

LOAD

DOMAIN

CONTROL

OfficePlus

ADABASIC

C/C++ Precompiler

COBOL Precompiler

CALL Interface

ADABAS Reference Manual

ADABAS Reference Manual for ANSI

ADABAS Reference Manual for DB2

ADABAS Reference Manual for ORACLE

Messages and Codes

11 Training Programs

Introduction to Relational Database Systems and SQL	(1 day)
ADABAS D - Introduction	(2 days)
ADABAS D - Database Administration (UNIX)	(3 days)
ADABAS D - Database Administration for R/3	(3 days)
ADABAS D - Database Design	(3 days)
ADABAS D - Programming	(3 days)

Table of Contents

1	Introduction.....
2	ADABAS D.....
3	Distribution.....
3.1	Logical View of the Distribution.....
3.2	Physical View of the Distribution.....
3.3	Client-Server Operation and Distributed Databases.....
3.4	Two-Phase Commit Protocol.....
3.5	Synchronous Replication.....
3.6	Asynchronous Replication.....
3.7	The Twelve Rules for Distributed Databases from Date.....
3.8	Location-independent Access.....
3.9	Gateways.....
3.10	Connectivity.....
4	SQLMODE.....
5	Platforms.....
6	SQL Extensions.....
6.1	Data Types.....
6.2	LONG Columns.....
6.3	Data Integrity.....
6.4	DB Procedures and Triggers.....
6.5	Temporary Tables.....
6.6	Subtransactions.....
6.7	Array Statements.....
6.8	Isolation Levels.....
6.9	Optimistic Locking.....
6.10	Transaction Chaining.....
6.11	Updatable Join Views.....
6.12	Outer Join.....

6.13	Scrollable Cursors.....
6.14	Built-in Functions.....
6.15	DB Functions.....
6.16	Special Characters.....
6.17	Authorization.....
6.18	Catalog Access.....
7	Operating System Embedding.....
8	ADABAS D and Its Tools.....
9	TOOLS.....
9.1	QUERY.....
9.2	EASY.....
9.3	LOAD.....
9.4	OfficePlus.....
9.5	DOMAIN.....
9.6	CONTROL.....
9.7	ADABASIC.....
9.8	PRECOMPILERS.....
10	Documentation.....
11	Training Programs.....