GHICA VAN EMDE BOAS-LUBSEN

Feature Analysis of Business System 12

X-1998-01, received: August 1998

ILLC Scientific Publications
Series editor: Dick de Jongh

Technical Notes (X) Series

Institute for Logic, Language and Computation (ILLC)
University of Amsterdam
Plantage Muidergracht 24
NL-1018 TV Amsterdam
The Netherlands
e-mail: illc@wins.uva.nl
Feature Analysis of Business System 12

Ghica van Emde Boas - Lubsen

IBM, the Netherlands, PO Box 24, 1420 AA Uithoorn; emdeboasl.nl.ibm.com

Abstract. Business System 12 is the name of a Relational database system developed by IBM, in the Netherlands, in the early 1980-ies, building on the work of a group at Peterlee, UK. A first version of BS12 became operational in 1983. The system was never installed at a client; users accessed the system in a time shared environment. After IBM management had decided in the late 1980-ies no longer to support the system, it remained in use on behalf of a major application until 1996.

In the database literature extremely little has been written about this system. The present report is a facsimile version of an unpublished internal memorandum at IBM Uithoorn; it describes the functionality of the system in the style as advocated in the book by Schmidt and Brody, Relational Database Systems, Analysis and Comparison, published in 1983. BS12 has a number of features which are rather unique for the Relational Database Systems, at least for those available in the 1980-ies. Its properties made the system particularly suitable for performing experiments in the field of deductive databases. This facsimile edition primarily has the purpose of making this information bibliographically traceable.
FEATURE ANALYSIS OF BUSINESS SYSTEM 12

October 4th, 1984

H. van Emde Boas

INS-Development Centre,
IBM, Uithoorn.

It is possible that this material may contain reference to, or information about, IBM products (machines and programs), programming, or services that are not announced in your country. Such reference or information must not be construed to mean that IBM intends to announce such IBM products, programming, or services in your country.
CONTENTS

1.0 Introduction .................................................. 1
  1.1 Identification ................................................ 1
  1.2 Status .......................................................... 1
    1.2.1 System .................................................. 1
    1.2.2 Applications ........................................... 1
  1.3 System Background ............................................ 1
  1.4 Overall Philosophy ........................................... 1
  1.5 Essentially Relational Characteristics ...................... 2
  1.6 Interfaces ................................................... 2
  1.7 Documentation ................................................ 3
  1.8 General System Description .................................. 3

2.0 Database Constituents ........................................... 4
  2.1 General Description ......................................... 4
  2.2 Segments ...................................................... 4
    2.2.1 Segment Structure ...................................... 5
    2.2.2 Segment operations .................................... 5
    2.2.3 Segment Constraints ................................... 5
  2.3 Table .......................................................... 5
    2.3.1 Table Structure ......................................... 5
    2.3.2 Table Operations ....................................... 6
    2.3.3 Table Constraints ...................................... 6
    2.3.4 Additional Properties of Tables ....................... 7
  2.4 Views ........................................................ 7
    2.4.1 View Structure .......................................... 7
    2.4.2 View Operations ....................................... 7
    2.4.3 View Constraints ....................................... 8
    2.4.4 Additional Properties of Views ....................... 8
  2.5 Row ........................................................... 8
    2.5.1 Row Structure ........................................... 8
    2.5.2 Row Operations ......................................... 8
    2.5.3 Row Constraints ........................................ 8
    2.5.4 Additional Properties of Rows ....................... 9
  2.6 Column ........................................................ 9
    2.6.1 Column Structure ....................................... 9
    2.6.2 Column Operations ..................................... 9
    2.6.3 Column Constraints ................................... 9
  2.7 Domain ........................................................ 9
    2.7.1 Domain Structure ....................................... 9
    2.7.2 Domain Operations .................................... 10
    2.7.3 Domain Constraints ................................... 10
  2.8 Language tables ................................................ 10
    2.8.1 general function language ................................ 11
    2.8.2 Use of Business System 12 language tables .............. 12
  2.9 Additional Database Constituents .................................. 13

3.0 Functional Capabilities ....................................... 14
  3.1 Qualification ................................................ 14
    3.1.1 Restriction ............................................. 14
    3.1.2 Quantification ......................................... 14
    3.1.3 Set Operations ........................................ 15
    3.1.4 Joining ................................................ 15
    3.1.5 Nesting and Closure .................................... 16
    3.1.6 Additional Aspects of Qualification .................... 17
  3.2 Retrieval and Presentation ................................... 17
    3.2.1 Database Queries ....................................... 17
    3.2.2 Retrieval of information about Database Constituents ... 17
3.2.3 Retrieval of System Performance Data
3.2.4 Report Generation
3.2.5 Constraints and Limitations
3.3 Alteration
3.3.1 Insert Facilities
3.3.2 Delete Facilities
3.3.3 Modify Facilities
3.3.4 Commit and Undo Facilities
3.3.5 Additional Alteration Facilities
3.3.5.1 TRANSFER
3.3.5.2 PROCESS
3.4 Additional Functional Capabilities
3.4.1 Arithmetic and String Operations
3.4.2 Sorting
3.4.3 Library Functions
3.4.4 User Defined Functions
3.4.5 Transactions
3.4.6 Multi-tuple Alterations
3.4.7 Grouping
3.4.8 Exception Handling Mechanism

4.0 Definition, Generation, and Administration Facilities
4.1 Definition Facilities
4.1.1 Constituent of a Database Definition
4.1.2 Segment Definition
4.1.3 Table Definition
4.1.4 View Definition
4.1.5 Row Definition
4.1.6 Attribute Definition
4.1.7 Domain Definition
4.1.8 Definition of Additional Database Constituents
4.2 Generation Facilities
4.2.1 Constituent of a Database Generation
4.2.2 Generation of Database Constituents
4.3 Database Redefinition
4.3.1 Renaming Database Constituents
4.3.2 Redefining Database Constituents
4.4 Database Regeneration and Reorganization
4.4.1 System-Controlled
4.4.2 DBA-Controlled
4.5 Database Dictionary

5.0 Interfaces and DBMS Architecture
5.1 System Architecture
5.2 Interface Descriptions
5.2.1.1 Conversational Facilities of Business System 12 interface
5.2.1.2 AS interface
5.2.1.3 APL interface
5.2.1.4 VSPC PL/I interface

6.0 Operational aspects
6.1 Security
6.1.1 Access Control
6.1.1.1 data sharing
6.1.2 Capability
6.2 Physical Integrity
6.2.1 Concurrency Control
6.2.2 Crash Recovery
6.3 Operating Environment
6.3.1 Software Environment (Operating System)
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.3.2 Hardware Environment (CPU, Memory, Peripherals)</td>
<td>32</td>
</tr>
<tr>
<td>7.0 Essentially Relational Solutions for Generalized DBMS Problems</td>
<td>33</td>
</tr>
<tr>
<td>8.0 Database Applications Using the System</td>
<td>34</td>
</tr>
<tr>
<td>A.0 Feature Summary</td>
<td>35</td>
</tr>
<tr>
<td>A.1 Database constituents</td>
<td>35</td>
</tr>
<tr>
<td>A.2 Operations</td>
<td>36</td>
</tr>
<tr>
<td>A.3 Schema definitions</td>
<td>37</td>
</tr>
<tr>
<td>A.4 Additional facilities</td>
<td>37</td>
</tr>
<tr>
<td>A.5 Functional Classes</td>
<td>38</td>
</tr>
<tr>
<td>A.6 Interface Flavors</td>
<td>38</td>
</tr>
<tr>
<td>A.7 Operational Aspects</td>
<td>39</td>
</tr>
</tbody>
</table>
PREFACE

The format and content of this document are based on:

the Feature Catalogue of Relational Concepts, Languages and Systems
May 1980
Working Paper RTG-80-81
of the Relational Database Task Group
of ANSI/X3/SPARC - Database System Study Group

as published in:

Relational Database Systems, Analysis and Comparison
Edited by Joachim W. Schmidt, Michael L. Brodie
With a Foreword by E.F. Codd
Springer-Verlag.

Note: The terminology used in this document differs from the terminology used in the feature catalogue and adheres to the one used in other Business System 12 documentation. The following translation list can be used:

| Database   | Segment |
| Relation   | Table   |
| Tuple      | Row     |
| Attribute  | Column  |
Most examples in this feature analysis are based on the database below:

**Table: S**

<table>
<thead>
<tr>
<th>S#</th>
<th>SNAME</th>
<th>STATUS</th>
<th>CITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>Smith</td>
<td>20</td>
<td>London</td>
</tr>
<tr>
<td>S2</td>
<td>Jones</td>
<td>10</td>
<td>Paris</td>
</tr>
<tr>
<td>S3</td>
<td>Blake</td>
<td>30</td>
<td>Paris</td>
</tr>
<tr>
<td>S4</td>
<td>Clark</td>
<td>20</td>
<td>London</td>
</tr>
<tr>
<td>S5</td>
<td>Adams</td>
<td>30</td>
<td>Athens</td>
</tr>
</tbody>
</table>

**Table: SPJ**

<table>
<thead>
<tr>
<th>S#</th>
<th>P#</th>
<th>J#</th>
<th>QTY</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>P1</td>
<td>J1</td>
<td>200</td>
</tr>
<tr>
<td>S1</td>
<td>P1</td>
<td>J4</td>
<td>700</td>
</tr>
<tr>
<td>S2</td>
<td>P3</td>
<td>J1</td>
<td>400</td>
</tr>
<tr>
<td>S2</td>
<td>P3</td>
<td>J2</td>
<td>200</td>
</tr>
<tr>
<td>S2</td>
<td>P3</td>
<td>J3</td>
<td>200</td>
</tr>
<tr>
<td>S2</td>
<td>P3</td>
<td>J4</td>
<td>500</td>
</tr>
<tr>
<td>S2</td>
<td>P3</td>
<td>J5</td>
<td>600</td>
</tr>
<tr>
<td>S2</td>
<td>P3</td>
<td>J6</td>
<td>400</td>
</tr>
<tr>
<td>S2</td>
<td>P3</td>
<td>J7</td>
<td>800</td>
</tr>
<tr>
<td>S2</td>
<td>P5</td>
<td>J2</td>
<td>100</td>
</tr>
<tr>
<td>S3</td>
<td>P3</td>
<td>J1</td>
<td>200</td>
</tr>
<tr>
<td>S3</td>
<td>P4</td>
<td>J2</td>
<td>500</td>
</tr>
<tr>
<td>S4</td>
<td>P6</td>
<td>J3</td>
<td>300</td>
</tr>
<tr>
<td>S4</td>
<td>P6</td>
<td>J7</td>
<td>300</td>
</tr>
<tr>
<td>S5</td>
<td>P2</td>
<td>J2</td>
<td>200</td>
</tr>
<tr>
<td>S5</td>
<td>P2</td>
<td>J4</td>
<td>100</td>
</tr>
<tr>
<td>S5</td>
<td>P5</td>
<td>J5</td>
<td>500</td>
</tr>
<tr>
<td>S5</td>
<td>P6</td>
<td>J7</td>
<td>100</td>
</tr>
<tr>
<td>S5</td>
<td>P6</td>
<td>J2</td>
<td>200</td>
</tr>
<tr>
<td>S5</td>
<td>P1</td>
<td>J4</td>
<td>1000</td>
</tr>
<tr>
<td>S5</td>
<td>P3</td>
<td>J4</td>
<td>1200</td>
</tr>
<tr>
<td>S5</td>
<td>P4</td>
<td>J4</td>
<td>800</td>
</tr>
<tr>
<td>S5</td>
<td>P5</td>
<td>J4</td>
<td>400</td>
</tr>
<tr>
<td>S5</td>
<td>P6</td>
<td>J4</td>
<td>500</td>
</tr>
</tbody>
</table>

**Table: P**

<table>
<thead>
<tr>
<th>P#</th>
<th>PNAME</th>
<th>COLOR</th>
<th>WEIGHT</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>Nut</td>
<td>Red</td>
<td>12</td>
</tr>
<tr>
<td>P2</td>
<td>Bolt</td>
<td>Green</td>
<td>17</td>
</tr>
<tr>
<td>P3</td>
<td>Screw</td>
<td>Blue</td>
<td>17</td>
</tr>
<tr>
<td>P4</td>
<td>Screw</td>
<td>Red</td>
<td>14</td>
</tr>
<tr>
<td>P5</td>
<td>Cam</td>
<td>Blue</td>
<td>12</td>
</tr>
<tr>
<td>P6</td>
<td>Cog</td>
<td>Red</td>
<td>19</td>
</tr>
</tbody>
</table>

**Table: J**

<table>
<thead>
<tr>
<th>J#</th>
<th>JNAME</th>
<th>CITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>J1</td>
<td>Sorter</td>
<td>Paris</td>
</tr>
<tr>
<td>J2</td>
<td>Punch</td>
<td>Rome</td>
</tr>
<tr>
<td>J3</td>
<td>Reader</td>
<td>Athens</td>
</tr>
<tr>
<td>J4</td>
<td>Console</td>
<td>Athens</td>
</tr>
<tr>
<td>J5</td>
<td>Collator</td>
<td>London</td>
</tr>
<tr>
<td>J6</td>
<td>Terminal</td>
<td>Oslo</td>
</tr>
<tr>
<td>J7</td>
<td>Tape</td>
<td>London</td>
</tr>
</tbody>
</table>

1.0 INTRODUCTION

1.1 Identification

Business System 12 is a business information service offered by IBM Information Network Services intended for time sharing use.

1.2 Status

1.2.1 System

The first release was offered as a service from the International Network in Zoetermeer, The Netherlands, starting October, 1983.

Since then Business System 12 has been installed in several other countries in Europe.

The current release (July 1984) is 1.1

Planned enhancements for release 2 are:

- Exploitation of S/370 Extended Architecture
- Database Structure improvements
- Operational improvements
- Intelligent Workstation
- MVS batch interface

1.2.2 Applications

Business System 12 is suited for applications in the Time-Sharing Service environment where the decisive factors to use Business System 12 may be among others:

- The availability of an international network.
- The requirement to develop and change applications quickly and simply.
- The requirement to access very large volumes of data.

1.3 System Background

Business System 12 was developed in the INS Development Center, Uithoorn, The Netherlands.

1.4 Overall Philosophy

Business System 12 was developed to address the need for an intelligent database management system in the Service environment.
All functions of Business System 12 are defined in ONE Application Program Interface (API for short) aimed at higher productivity of System Engineers and better customer self-sufficiency. This Application Program Interface is common to all categories of application programs.

A further key objective of Business System 12 is the data and application independency resulting from objects within Business System 12 (data, views, clists) being accessible between different users via different applications - provided the access authorization criteria are met.

It was also felt necessary to put great emphasis on security aspects:

- Users should only be able to see data which they own or which is explicitly shared with them and for which they know the access password.
- Users should be protected from loss of data or database integrity by system failure or user errors.
- It should be impossible to impact the processing of other users for instance by keeping locks on important system resources.

1.5 Essentially Relational Characteristics

Business System 12 satisfies all the requirements of a fully relational system, as specified by E.F. Codd (TODS Vol. 4, No. 4, Dec. 1979).

1.6 Interfaces

The architecture and design of Business System 12 allow different interfaces to be provided to different end users. Currently, there are four main ways to access Business System 12:

- Conversational Facilities of Business System 12
- Application System
- APL
- PL/I

No matter which user interface is used, basically each interface communicates with Business System 12 through the same API, in which all of the following items can be expressed:

- Database Schema Definition.
- Query language.
- Database altering.
- Constraint Definition.
- Database Generation and Regeneration.
- Database Schema Redefinition and Renaming.
- Data Entry.
- Security Definition, Monitoring and Control.
- Database Control (utilities). load, dump, backup, restore, recovery, monitoring, etc.
- Database Dictionary (database design, dictionary query, etc.)

Some functions the Application Programming Interface does not provide are:

Introduction
• Report generation.
  Limited facilities are available in the Conversational Facilities of Business System 12.
• Definition of Storage structure, Indexes and Access Paths:
  These are not part of the interface but are done implicitly by the system.

1.7 Documentation

The information given in this document is mainly based on internal project documents. The generally available documentation comprises:

3. Business System 12 Twelve Commands Card, GX11-6077

1.8 General System Description

Business System 12 is the total complex of a Data Base Management System and different user and application front ends.

Business System 12 provides Relational data base access and manipulation facilities, data integrity, data and application independence, together with powerful organizational and operational features like segmented database, and backup and recovery functions.

A unique feature of Business System 12 is its support of user defined functions, command lists, process definitions and views as database objects, which can be manipulated as ordinary tables and which can be defined in a structured language, allowing arguments, local variables etc. to be used.
2.0 DATABASE CONSTITUENTS

2.1 General Description

The constituents of Business System 12 are:

<table>
<thead>
<tr>
<th>BS12 TERM</th>
<th>FEATURE CATALOGUE TERM</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB (Database)</td>
<td>Segment</td>
</tr>
<tr>
<td>R (Relation)</td>
<td>Table</td>
</tr>
<tr>
<td>T (Tuple)</td>
<td>Row</td>
</tr>
<tr>
<td>A (Attribute)</td>
<td>Column</td>
</tr>
<tr>
<td>D (Domain)</td>
<td>Domain</td>
</tr>
</tbody>
</table>

A DB consists of R's of possibly different types. An R consists of a set of T's of identical type. A T is a set of A values. Each A is taken from a named domain D whose underlying type is character, name, numeric or timestamp.

The structure of a complete Business System 12 database is described by a set of system tables:

Table of Segments
This table has a row for each segment, describing its status etc. There is one Table of Segments in a system.

Table of User Profiles
This table contains a row for every user, representing a "profile" for this user, such as user id., type, maintenance status, contract, storage allocated etc. There is one Table of User Profiles in a system.

Table of Tables
This table contains a row for every table a user owns, specifying its name, contents and characteristics. There is one Table of Tables for every user.

Table of Columns
This table describes the columns of a data table. There is one row for every column and there exists one Table of Columns for every data table.

2.2 Segments

The Business System 12 Database comprises all the data, of any nature, that is under its control, belonging to all its users and to Business System 12 itself. It is a Segmented Database because it may be divided into a number of separate, discrete bundles called Segments.

When a user is installed on Business System 12, he is assigned to one of these segments. That means that all of the data owned by him will be stored in this segment. Other users may also be assigned to the same segment. That does not mean that they can automatically access each other's data, but just that their data is stored within the same physical datasets (shared access to data is under the control of its owner, and is independent of segments).
2.2.1 Segment Structure
Each segment comprises two distinct datasets:

The Permanent Store
The Permanent Store contains all the permanent data belonging to users on this segment - their stored tables and dictionaries - as well as control information, generated and maintained by the system, about the contents of the dataset.

The Scratchpad
It contains various things, generated either by Business System 12 or by users, which can be regarded as temporary or are recreatable. The data of a user in the scratchpad will be deleted at the end of every session of that user.

2.2.2 Segment operations
Segments can be created, deleted, backed-up, restored etc., by manipulating the Table of Segments and by the SEGMENT command intended for use by the System Administrator.

2.2.3 Segment Constraints
Special authorization is needed to manipulate segments.

2.3 Table

2.3.1 Table Structure
In Business System 12 a relation is known as a table consisting of rows and columns. Loosely speaking, a table can be compared with a file, a row is equivalent to a record in the file and a column is the set of values in a field for all records. The difference with conventional data processing is, that in Business System 12 only complete tables can be manipulated instead of single records.

Some characteristics of Business System 12 tables are:

- The names of tables and columns have the NAME datatype, which means that names are a normalized form of character strings, for example: "MYTAB 01" would be the same as "mytab 1".
- Table names are required to be unique within the set of tables owned by a user. This is enforced by that fact that table names are the key column of the Table of Tables.
- All rows in a table have the same type and use the same set of columns.
- Duplicate rows are not allowed. Every table must have a primary key, which can consist of one or more columns.
- Columns in a row can have different types.
- The ordering of columns is insignificant.

Database Constituents
• Alias names for tables can be defined by adding a row to the Table of Synonyms. (of which there is one per user).

2.3.2 Table Operations

Data can be manipulated using the twelve relational commands from which Business System 12 derives its name:

**SELECT** Selects a subset of the rows of a table satisfying a specified predicate.

**PRESENT** Removes or renames columns. (in many relational systems this operator is called **PROJECT**).

**CALCULATE** Generates new or adjusts existing columns.

**GENERATE** Generates a one-row view, in the same way as **CALCULATE**, but operating on constant values.

**SUMMARY** Performs vertical operations such as column totalling and subtotaling.

**JOIN** Combines the rows of the first participating table with rows from the second participating table that have matching values in their common columns (the columns which have the same name and the same basic data type in both tables).

**MERGE** Same as **JOIN**. However rows from the first table which cannot be matched are expanded using values from a third single-row table.

**QUAD** A quadratic **JOIN** on tables who have no common columns.

**DIFFERENCE** The rows in the first table with non-matching values.

**EXCLUSION** The rows in both tables with non-matching values.

**INTERSECTION** The rows with matching values.

**UNION** All the rows of two tables.

2.3.3 Table Constraints

• All Tables must have at least one key column, although for some table types a key column is implicitly defined. This guarantees uniqueness of rows.

• Tables must be in first normal form.

• Tables cannot have more than 300 columns.

• Inter-table constraints can be implemented by the user with a TRAP CLIST. Within this CLIST any sequence of Business System 12 commands can be used. The CLIST is invoked when a table is prepared for access.
2.3.4 Additional Properties of Tables

In Business System 12 not all tables are of the same type. In general three groups of tables are recognized with different content:

Data tables. These tables can take any shape and contain user data.

System tables. These tables contain system and data dictionary information. They have a fixed shape, which means that their Table of Columns is predefined. The system tables for which a user has access authorization can be manipulated by that user, such as his own Table of Tables or any Table of Columns of his own data tables.

Language tables. This is a special type of table which is used to contain function language statements, rather than true data. Language tables can be used for different purposes, depending on their content attribute. The most common are command lists and view definitions. Language tables and Function language are described in more detail in "Language tables" on page 10.

2.4 Views

2.4.1 View Structure

A VIEW defines the result of an arbitrarily complex relational expression.

View definitions can either exist as session view definitions or as stored views in the database.

A view can be defined (using DEFINE commands) as a sequence of relational commands. A view name can be used instead of a table name anywhere in a relational statement. The result of a view is '\$', the current table, at the end of a stored view definition. The name of a stored view must be unique across all tables the user owns.

The session view names are searched first when reference is made to a table name, therefore they override whatever else may be present in the database. The user can circumvent this by qualifying the table name with a user identification.

2.4.2 View Operations

A stored view can be created by creating a table with CONTENT(VIEW) and adding rows to it containing the text of its definition.

A session view definition can be made by simply entering:

\[
\text{DEFINE \ viewname = relational expression}
\]

Of course, this relational expression can contain any temporary or permanent view name.

Intermediate results are never executed when they are defined.
Commands are provided to KEEP temporary definitions as a stored view and to STORE the result of a view (the actual data) in the database.

As follows from the view definition, there is no distinction in Business System 12 between operations on base tables or views.

2.4.3 View Constraints

Not all views can be updated. In principle, all views can be updated for which no ambiguity exists which base table row must be updated.

2.4.4 Additional Properties of Views

Stored views may use Business System 12 Function Language, which means that for view definition evaluation IF-THEN-ELSE logic and local variables may be used. Also arguments may be passed to the view.

2.5 Row

2.5.1 Row Structure

A row is a set of column values.

The structure of the row is defined implicitly by the definition of the table itself. The Table of Columns which exists can be seen as the definition of the row structure.

2.5.2 Row Operations

Single rows cannot be retrieved in Business System 12, except with a suitable SELECT command. The result of a query is always a complete table.

Row updating, adding or deleting can be done with the three row manipulation commands:

ADD, CHANGE, DELETE

A more detailed description will be given in section 3.3.

Note: The only way to delete a table in Business System 12 is to delete its row in the Table of Tables.

2.5.3 Row Constraints

No duplicate rows are allowed in a table.

No ordering is maintained for rows in the database. Ordered output can be requested.

The only limit to the number of rows in a table is the amount of secondary storage available.

Database Constituents
2.5.4 Additional Properties of Rows

Rows can be addressed explicitly by specifying their key values in the row manipulation commands.

Rows are treated as elements of a set.

2.6 Column

2.6.1 Column Structure
A column is defined as a row in the Table of Columns for a table. A Table of Columns row contains among others, the following values:

- Name, Key-ness, Ordering, Domain name, Data type, Default value, Formatting information.

2.6.2 Column Operations
A table which is created but does not yet contain any rows can be restructured, which means that rows in the Table of Columns can be added, changed or deleted.

2.6.3 Column Constraints
When the table itself contains data, no adding or deleting of Table of Columns row is allowed anymore. Some changes can still be made.

A column cannot be renamed in a base table, in a view columns can be renamed through the RENAME parameter of the PRESENT command.

2.7 Domain

2.7.1 Domain Structure

In Business System 12 a domain is a set of user specified rules against which every potential update to a column of data will be checked before it is applied to the data.

This set of rules is contained in a row in the Table of Domains every user owns.

Every column has a domain associated with it, the domain name is specified as an operand to the column name during table create.

Each domain relates to a basic datatype which can be any of:

- CHARACTER, NUMERIC, NAME, BIT, TIMESTAMP

A user can create his own domains by adding rows to the Table of Domains. A set of standard domains is provided also in the Table of Domains.
belonging to the INSTALLation user id, which is included in the search order for every user.

There are no distinguished domain values such as NULL. DEFAULT values can be specified instead.

2.7.2 Domain Operations
The information which can be found in a row of the Table of Domains is:

Name
Datatype Any of the basic types.
Length max. length for character strings
Decimals max. nr of decimals for numeric data.
Formatting info The way data is presented to the user.

Check expression A logical expression which should evaluate to true or false when applied to a potential update. User defined functions can be used, in which any field in any table accessible to the user can be referenced.

Valid expression A relational expression resulting in a one-column table. If the new value is found as a value in that column, it is accepted.

There is some overlap between the information in the domain and in the Table of Columns row. When a table is created, this information is copied from the domain row into the Table of Columns row. Values changed afterwards in the Table of Columns row, will override those still in the domain.

2.7.3 Domain Constraints
Domains can be added, changed or deleted as the user requires, however Business System 12 will not keep track where a domain is used and if data stays valid when a domain is changed. To assist in validating the data under such circumstances, the VALIDATE command is provided.

Domain checks are only performed when adding or changing rows, not when rows are deleted.

2.8 Language tables
Language tables are a special form of table in which the data is meaningful to Business System 12 or an application. They are distinguished from data tables by their possession of a 'CONTENT' of CLIST, VIEW, etc, i.e. anything except DATA.

Language tables whose contents are understood by Business System 12 contain two columns, the line number and a statement. The statement is a character string which represents an Business System 12 function language statement, as described below. The column names provided by Business System 12 are LINE and TEXT.
The order of rows in a language table has meaning, and the line number indicates that order.

Note that a language table is a table, and can be used in relational expressions by reference to its name, just like any other table. The exception to this is a View. When the name of a View is used, the relational expressions which it contains are executed, and the result is used, rather than the statements in the View itself. The result of the View is the last explicit or implicit assignment to asterisk (*) in the table. The DEFINITION function can be used to see these statements.

2.8.1 general function language.

Business System 12 has a general function language which can be used in several of the forms of function. Its statements and syntax are described below.

Function Arguments
A function can be invoked with an argument list.

Local Variables
If a name appears on the left of an assignment, it is a local variable, and any further references to that name can be made.

System Variables
These variables are available to provide system information. The system variables are:
ARGS, USEDBY, SESSID, MATCH, OLDVALUE, NEWVALUE, COMMAND, SEVERITY, REASON, WORST, SYSTEMLEVEL, FIRST.

The reader is referred to the Business System 12 reference guide for a precise description of their meaning.

System Functions
As for system variables, these functions can be used in function language statements to provide system information. They are:

    NOW

    OWNEROF(table-name-expr)

    FOUND(table-name-expr)

Expressions
The full Business System 12 scalar expressions are available, including all the functions. Other functions may be invoked, as is appropriate in the context. Recursive execution is allowed. They can operate on any argument, named data item, constant, or system variable which is available.

A single data item with no operations involved is considered to be a trivial form of expression.

function language statements

- LET
  Used to assign the result of an expression to a variable.
- IF
IF condition
THEN
t-statements
ELSE
e-statements
END IF

The condition is a Boolean expression which can be true or false. It may be comprised of logical expressions and bit variables.

If the condition is true, the t-statements are executed, otherwise the e-statements are executed.

• LOOP

LOOP ![name] [,WHILE condition] [,UNTIL condition]
LEAVE
REPEAT
END LOOP

This causes a group of statements to be executed repeatedly.

• Using RETURN, execution of the function is terminated, and the value of the return expression is passed back to the invoker.

• COMMAND
It is allowed to specify the word COMMAND in front of any API command used in a function.

• PASS
is used to preserve the values of local variables from one execution of a function to the next. PASS is only allowed within a PROCESS function.

• CALL is used to pass control to function of the same type. Arguments can be passed.

2.8.2 Use of Business System 12 language tables

• CLIST
Contains a set of Business System 12 commands that can be invoked via the RUN command.

• VIEW
A sequence of Business System 12 relational expressions which produce a view.

• PROCESS
A special command list used with the PROCESS command. The process function is invoked for every row of a table that is specified as an operand in the command.

• FUNCTION User defined functions can include the general function language statements. RETURN is used to provide the result of the function.
2.9 Additional Database Constituents

Other database constituents comprise:

**Snapshot** can be made with the Business System 12 STORE command.

**Trigger**
With a TRAP CLIST access to tables can be monitored.

**Transaction**
Units of work can be delimited and concurrency control is provided.

**Authority control**
Authority to access (part of) tables can be granted and revoked.

**Relational dictionary**
All system information is available as system tables.
3.0 FUNCTIONAL CAPABILITIES

3.1 Qualification

The mechanism to select items from the database is implemented in Business System 12 through the twelve relational commands.

3.1.1 Restriction

The two commands with which subsets of a table can be found are: 

SELECT and PRESENT (the usual PROJECT).

<table>
<thead>
<tr>
<th>PRESENT</th>
<th>SELECT</th>
</tr>
</thead>
<tbody>
<tr>
<td>(tabexpr, collist)</td>
<td>(tabexpr, logexpr)</td>
</tr>
</tbody>
</table>

Figure 1. SELECT end PRESENT: A pictorial representation.

The logical expression in SELECT can be any expression which evaluates to true or false and can include user defined functions. See section 3.4 for a more detailed description of expressions.

Compare operations can only be performed on values of the same basic datatype. Columns do not necessarily have the same domain.

3.1.2 Quantification

Existential quantification is provided by the

EMPTY('...')

function. Its value is '1'B if the table identified by '...' contains no rows, else its value is '0'B.

Further quantification can be achieved with the SUMMARY command. This command is described in more detail in section 3.4.7. The for quantification relevant summarization functions are ALL and ANY. Absolute quantification (there is at least ...) can easily be coded.
3.1.3 Set Operations

The Business System 12 set operators are

\[ \text{UNION, INTERSECTION, DIFFERENCE, EXCLUSION.} \]

![Diagram of set operations](image)

Figure 2. The SET operators: A pictorial representation.

3.1.4 Joining

Three join operators are provided in Business System 12:

\[ \text{JOIN, MERGE, QUAD.} \]

\text{JOIN} gives a natural join over the common columns of two (or more) tables. Any two tables or views can be joined, provided they have at least one common column.

\text{MERGE} is a restricted version of an outer join, (all the rows of the first table are kept, even if they do not match), the restriction being that the common columns must be key in the second table.

Functional Capabilities
Note: To ensure joining on the proper columns, the RENAME facility of the PRESENT command can be used.

QUAD is the cartesian product of two tables which have no columns in common.

![Diagram of JOIN and MERGE operations]

**Figure 3. The JOIN operators: A pictorial representation.**

3.1.5 Nesting and Closure

All twelve relational commands use table-expressions as operands. This means that any view name or relational expression can be used there.

Nesting can go to any depth, virtual storage permitting.

The Business System 12 commands are relationally complete.
3.1.6 Additional Aspects of Qualification

Relational \textit{DIVIDE} is provided as a system supplied stored view.

3.2 \textit{Retrieval and Presentation}

3.2.1 Database Queries

The result of a query in Business System 12 will always be a table, whether a view result or a base table.

The contents of this table can be transferred over the interface with a series of commands:

\begin{verbatim}
START TRANSFER table-expression [,options ]
GET
END TRANSFER
\end{verbatim}

At the external level, most users will not see these commands, for example in Conversational Facilities of Business System 12 you can simply say:

\begin{verbatim}
DISPLAY table-expression
\end{verbatim}

3.2.2 Retrieval of information about Database Constituents

Since all information about database constituents is stored in tables, the so called \textit{SYSTEM TABLES}, they can be retrieved, qualified and altered in exactly the same way as any other table. For some alterations, authorization is restricted to privileged users.

3.2.3 Retrieval of System Performance Data

Information about secondary storage usage and Processing Unit usage are stored in system accounting tables. The user can query the rows containing his own information.

A user can see his own table directory (the Table of Tables), the installation directory and per user, a directory of the tables which are shared with him (the Table of Shared Access).

Access paths are invisible to the user, as are the hash indexes.

3.2.4 Report Generation

Apart from \textit{SUMMARY} which can perform grouping and (sub-)totalling, no Report Generation facilities are provided in Business System 12 itself. It was considered that the external interfaces should provide these. Conversational Facilities of Business System 12 has limited, AS more extensive reporting facilities available.
3.2.5 Constraints and Limitations

START/END/CANCEL TRANSFER cannot be used within a CLIST.

3.3 Alteration

Alteration of tables can be achieved in Business System 12 with the ADD, DELETE, CHANGE and TRANSFER commands.

All tables for which a user has update authority, (including data tables, system tables, language tables) can be altered by that user.

As mentioned before, altering rows of a system table has side effects, for example adding a row to the Table of Tables creates a new table.

3.3.1 Insert Facilities

With the ADD command a row can be added to a table.

Syntax:

ADD table-expression, key-column(value), ... < , column-name = expression, ... >

Example:

ADD S,S#(S6),SNAME='Jansen',STATUS=40,CITY='Amsterdam'

Column values not provided are given default values as found in the Table of Columns.

An ADD operation will be rejected if the table already has a row with identical values in the key columns. The same is true if domain constraints are violated.

3.3.2 Delete Facilities

With the DELETE command a row can be deleted from a table.

Syntax:

DELETE table-expression, key-column(value)

Example:

DELETE S,S#(S4)

Note: The only way to delete a table is by removing its row from the Table of Tables.

To delete the parts table, we could write:

DELETE TAB(OWN),TABLE('P'N)

With the CLEAR command multiple rows can be deleted from a table.

Syntax:

CLEAR table-expression

Functional Capabilities
For example to remove all screws from the parts table:

    CLEAR SELECT(P,PNAME='Srew')

Or to delete all tables whose name starts with Q:

    CLEAR SELECT(TAB(OWN),SUBSTR(TABLE,1,1)='Q')

Domain constraints are not considered for delete operations. Authorization can explicitly in- or exclude deletes.

3.3.3 Modify Facilities

Any non-key column field in any table can be modified with the CHANGE command, authorization permitting.

Syntax:

    CHANGE  table-expression, key-column(value), ... <, column-name = expression, ... >

Example:

    CHANGE S,S#(S4),CITY='Rotterdam'

Column values not provided are not changed.

With the UPDATE command multiple rows can be updated in a table.

Syntax:

    UPDATE  table-expression-1 <, USING (table-expression-2)>
       , column-name = expression, ... <, options>

For example add 20% to the weight of all screws from the parts table:

    UPDATE SELECT(P,PNAME='Srew'),WEIGHT=WEIGHT*1.20

The USING keyword indicates that information in a second table should be used.

3.3.4 Commit and Undo Facilities

A sequence of tentative alterations can be delimited by

START/END/CANCEL UNIT. These so called 'units of work' can be nested, which means that alterations in an inner unit can be undone separately. Data is only committed to the database when the outermost unit is ended.

3.3.5 Additional Alteration Facilities

3.3.5.1 TRANSFER

An alternative way to alter blocks of data also the

START/END/CANCEL TRANSFER can be used. The syntax of these commands and the allowed options are extensive and rather complex. They are meant to be used by special applications or interfaces.

Functional Capabilities
3.3.5.2 PROCESS

A user defined function can be applied to every row of a base table or view result. This user function, which is stored in a language table of type PROCESS, can update fields in the current row, issue any database command and pass arguments to the next invocation of the process function for the next row.

3.4 Additional Functional Capabilities

3.4.1 Arithmetic and String Operations

The following arithmetic operators are supported:

**  Exponentiation
/  Division
*  Multiplication
+  Addition
-  Subtraction
%  Percent

The following comparison operators are supported:

=  Equal
>  Greater than
<  Less than
!= or <> or ><  Not equal
=> or <= or ==  Not greater than
<= or >= or =>  Not less than

The result of a comparison operation is a bit value.

The following logical operators are supported:

&  AND
&&  EXCLUSIVE OR
|  OR

The following monadic operators are supported:

~  Not
+  Plus
-  Minus

The concatenation operator is supported:

||

The operator is used to concatenate, or combine, two character data items to form a single character expression.

The assignment operator can be expressed by:

=

Functional Capabilities
and is pronounced 'gets' or 'receives'. It is used to assign the result of the expression on the right of the operator to the variable on the left.

The range operator can be expressed by:

: Range A : B indicates any value between and including A and B. The Range operator is always used together with a comparison operator.

3.4.2 Sorting

Sorting is done implicitly when necessary by invoking the IBM program product OS-SORT.

Ordering of rows can be requested by specifying ORDER in the Table of Columns of a data table or by using the ORDER() parameter in the START TRANSFER command.

3.4.3 Library Functions

More complex arithmetic operations, all string operations, and all user-defined operations, are represented as functions. A function reference has the form

function-name [( argument [, argument ...] )]

and can be used wherever a constant or variable can be used. The function operates on the arguments to produce a result, and this value is used in the expression. The arguments can be constants, variables, or expressions.

These functions operate on values to produce a result which is a scalar.

For a detailed description the reader is referred to the Business System 12 reference guide. An overview of the available functions is given here.

functions which perform calculations:

MODULO ( argument1, argument2 )
REMAINDER ( argument1, argument2 )
TRUNCATE ( argument1 [,position] )
ROUND ( argument1 [,position] )

mathematical functions:

SQR ( argument )

string analysis functions:

SUBSTRING ( string, start-position [,length] )
INDEX ( string, search-string [,count] )
VERIFY ( string, verification-string )
LENGTH( string )

conversion functions:

Functional Capabilities
DATE ( argument [,format] )
TIME ( argument [,format] )
NAME ( character | name )
VALUE ( argument [,format] )
CHARACTER ( argument [,format] )
BIT ( character )

other functions:

IF( condition, true-expression ,false-expression )
EMPTY ( character-expression )
CTTV ( character-expression , type )
CTTV stands for: Convert Table To Value.
CWINI ( expression, type )
CWINI stands for: Column Whose Name Is.
COLNAME ( expression )
ROWCOUNT ( name-expression | character-expression )

functions which operate on groups:

These functions can only be used in the SUMMARY operation.

TOTAL ( arith-expr)
COUNT
MAX ( arith-expr)
MIN ( arith-expr)
ALL ( bit-expr )
ANY ( bit-expr )

3.4.4 User Defined Functions

A user defined function is any language table with content FUNCTION. It therefore consists of a sequence of function language statements. A RETURN statement defines its result. It is not allowed to use an API command.

Expressions in a function can refer to columns in the table on which it operates, system variables, and constants. They can use any scalar operator available in Business System 12, and any Business System 12 or user-defined function.

To take a trivial example, consider the PAY RISE function. Where 'A AND C' is the name of a column of the table EMPLOYEES, the value of which is passed to the function PAY RISE, which could be used as follows

CALCULATE(EMPLOYEES, SALARY = SALARY + PAY RISE(A AND C) )

PAY RISE could be defined by the following statements

!RISE = (2*(5-1))*SALARY/100 IF JOB TITLE = 'MANAGER' THEN !RISE = !RISE *2 RETURN !RISE

3.4.5 Transactions

A user can delimit units of work by using

START UNIT
END/CANCEL UNIT

Functional Capabilities
constructs.

Units can be nested.

Implied units of work are single commands and TRANSFER blocks.

3.4.6 Multi-tuple Alterations

Multi-tuple alterations can be performed with UPDATE and PROCESS.

3.4.7 Grouping

Grouping can be performed with the SUMMARY command. The grouping library functions are described in section 3.4.3.

    SUMMARY (table-expr., GROUP(column-spec.),
               column=grouping function,... )

3.4.8 Exception Handling Mechanism
4.0 DEFINITION, GENERATION, AND ADMINISTRATION FACILITIES

4.1 Definition Facilities

4.1.1 Constituents of a Database Definition

4.1.2 Segment Definition

Is performed by adding a row to the Table of Segments.

There is no limit to the number of its constituents such as tables, apart from the availability of secondary storage.

4.1.3 Table Definition

Is performed by adding a row to the Table of Tables.

Alternatively, the CREATE command can be used. For example, the command to create table S of the parts database:

```
CREATE S,KEY(S#(CHAR)), COLUMNS(SNAME(CHAR),
   STATUS(NUMERIC),CITY(CHAR))
```

Some limitations are:

- The maximum number of columns in a table is: 300
- The maximum number of key columns in a table is: 10
- The maximum number of order columns in a table is: 10

4.1.4 View Definition

Views can be defined in two ways:
Either with the DEFINE command, which provides a temporary definition for the duration of the session, or via adding rows to a table with content VIEW

For example a simple view providing all suppliers in London, could be defined as:

```
DEFINE LONDON SUPP = SELECT(S,CITY='London')
```

or as:

```
CREATE LONDON SUPP,CONTENT(VIEW)
ADD DEF(LONDON SUPP),LINE=10,TEXT='* = SELECT(S,CITY="London")'
```

A more general view could be made in this way:

```
CREATE CITY SUPP,CONTENT(VIEW)
ADD DEF(CITY SUPP),LINE=10,TEXT='* = SELECT(S,CITY=!1)'
```

The result of this view could then be seen, using the Conversational Facilities of Business System 12:

```
DISPLAY CITY SUPP('London')
```
4.1.5 Row Definition
Is implicit in table create.
A limitation is:
- The maximum number of characters in a row is: 8196

4.1.6 Attribute Definition
Is performed by adding a row to the Table of Columns. See section 2.6.
Limitations are:
- Column names should be unique within a table.
- The maximum number of characters in a column is: 1000

4.1.7 Domain Definition
Is performed by adding a row to the Table of Domains. See section 2.7.

4.1.8 Definition of Additional Database Constituents
The definition of user defined FUNCTIONs, CLISTs and PROCESSes can be done in the same way as a permanent VIEW is defined, by creating a table with the proper content and then adding rows to it.

4.2 Generation Facilities
The generation facilities are part of the Application Programming Interface and most of its facilities are already described. Some additional comments are made below.

4.2.1 Constituents of a Database Generation
As described, new segments can be added to the system by adding a new row to the Table of Segments.

4.2.2 Generation of Database Constituents
In addition to the Application Programming Interface facilities to populate a segment there is an IMPORT available to load VSPO files into Business System 12 tables.

Copying of relations can be done by using the STORE command.

4.3 Database Redefinition

4.3.1 Renaming Database Constituents
No renaming of tables is possible other than by providing an alias in the Table of Synonyms.
4.3.2 Redefining Database Constituents

Columns can be deleted and added to a table when the table is still empty.

A change to the definition of a column can always be made, except that its basic datatype cannot be changed.

Domains can be redefined as required by changing its row in the Table of Domains.

4.4 Database Regeneration and Reorganization

Tables can be created and deleted at any time.

Reorganization of the database itself is necessary at periodic intervals to make the best use of disk space available.

4.4.1 System-Controlled

The system takes no initiative in reorganization activities.

4.4.2 DBA-Controlled

The database administrator provides a backup / reorganize plan. Further description is given in "Crash Recovery" on page 31.

4.5 Database Dictionary

The database dictionary is the set of system tables which describe all system information. They are described in earlier sections.
5.0 INTERFACES AND DBMS ARCHITECTURE

5.1 System Architecture

VSPC

<table>
<thead>
<tr>
<th>AS</th>
<th>APL</th>
<th>CBS12</th>
<th>PL/I</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Control Interface (SSM)

BS12 main

- Interpreter
- Compiler
- Optimizer

Data Management System
- Storage strategy/backup
- Disk via VSAM

Other batch (auxiliary processors)

MVS

Figure 4. Business System 12: Structure overview.

In essence the Business System 12 architecture can be condensed into the following list of items:

- the Business System 12 data base management system operates as an auxiliary processor of MVS/VSPC in release 1. More interfaces will be available in release 2.

- the relational data base is accessible from several (VSPC, later other) foreground processors.

- all communication between the different processors of VSPC is done via SSM (Shared Storage Manager).

- background (other auxiliary) processors can also communicate with the Business System 12 DBMS via SSM.

- the design is such that other modes of communication can be integrated at a later release.

- the Business System 12 data base management system is split into 3 fairly distinct components:
  - control and services
  - interpreter, compiler and optimizer
- data management, storage and access strategy, backup and VSAM access to disk.

The part of Business System 12 which is running the Auxiliary Processor is independent of VSPC. It only uses SSM to communicate with VSPC via defined protocols. Any system using SSM and following the Business System 12 rules is able to communicate with Business System 12.

The interpreter/compiler/optimizer processes the commands received at the Application Program Interface, and internally uses the services of the database manager when these commands require operations on stored data. These operations include creation, modification, and retrieval of stored data, and many 'housekeeping' operations concerned with backup, recovery and reorganization.

The Database Management System is that component of Business System 12 which looks after the stored data.

Below there are the low level access methods.

5.2 Interface Descriptions

In essence there is only one interface to Business System 12, The Application Programming Interface or API for short.

The user cannot access Business System 12 directly but will do so via one of its interfaces, currently:

- Conversational Facilities of Business System 12
- PL/I
- APL
- AS

As described, all user interfaces come together at the Application Program Interface. The reader should bear in mind that there are two levels of user interface:

1. end-user interfaces aimed directly at the user at a terminal, e.g. AS and the Conversational Facilities of Business System 12 Interface,

2. and those which operate at the Application Program Interface level, e.g. PL/I.

5.2.1.1 Conversational Facilities of Business System 12 interface

The Conversational Facilities of Business System 12 Interface is currently a VSPC foreground processor, which, amongst other functions, simply passes the Application Program Interface instructions and the generated responses between the user and Business System 12. Another feature is a flexible data entry and display facility, in order to allow command and table handling, easy manipulation of data for screen and typewriter terminals.

The Conversational Facilities of Business System 12 Interface is beneficial to the following categories of use:
Ad-hoc data manipulation and query.

Data preparation and debugging tool for VSPC application development.

The Conversational Facilities of Business System 12 Interface provides the end-user with the functions of Business System 12 with emphasis on:

- Simplicity: Small number of commands for data- and result- creation and manipulation
- Flexibility: user setup of screen layout and dynamic function assignment of PFkeys.

5.2.1.2 AS interface

This interface has a number of major objectives. These are:

- to provide a primary application interface by which Business System 12 is presented to the end user.
- to present the benefits of Business System 12 in an easy to understand and end-user oriented form.
- to provide a major functional enhancement to AS which will not be restricted to current host system limitations.
- to extend the data independence and data integrity features of AS by taking advantage of the benefits of Business System 12 data base management.

5.2.1.3 APL interface

The APL user communicates with Business System 12 via the shared storage management facilities of VSPC using APL shared variables.

An APL workspace contains functions to simplify communication with Business System 12 for the user.

The APL Interface combines the full power of data analysis in APL with the full power of Business System 12 data management using data from the user's Business System 12 data base.

5.2.1.4 VSPC PL/I interface

The routines to communicate via SSM are already available to the VSPC PL/I programmer. To communicate with Business System 12, the user must be aware of the communication protocols. Rather than impose such extra effort on the PL/I programmer, a set of PL/I subroutines is available which a VSPC PL/I programmer can use to achieve the CALLs to Business System 12 via SSM.
6.0 OPERATIONAL ASPECTS

6.1 Security

This section describes the security features provided by Business System 12.

6.1.1 Access Control

Before a user can access Business System 12, he must have access to the supporting host system. In release 1 this will be VSPC, an IBM service which gives access to a large computer on a time-sharing basis.

Next, a user can enter (via his interface, e.g. Conversational Facilities of Business System 12) the API LOGON command. Together with this command the user should supply his user-id and password. If either of these is not correct, the LOGON is rejected. Else a physical connection with the terminal is established.

6.1.1.1 data sharing

An Business System 12 user who owns data may allow other users selective access to it, using the SHARE command. He may choose which users may access what parts of his data, and he may selectively restrict the mode of access. The information about data sharing is strictly under control of its owner and stored in his Table of Shared Access.

The RETRACT command allows an object (as defined under the SHARE command) to be retracted.

Another way to restrict access to a table is by using table passwords. Any access to that table should then be accompanied by the proper password.

Finally, a TRAP CLIST can be specified for a table. This CLIST is executed at every access to a table. It can issue any Business System 12 command, look at the userid invoking it, restrict access to certain time intervals etc.

6.1.2 Capability

The authorities a user can grant on a table via the SHARE command are:
READ, UPDATE, ADD, DELETE, CHANGE, EXECUTE (a CLIST), BYPASS (validation).

When a user shares a view, the sharer cannot see how the view is defined, unless the definition is also explicitly shared. Execute authority allows a user to execute a language table, such as a CLIST, table without necessarily being able to see its contents.

An Business System 12 user has not only access to his own tables. He has also full access to tables which belong to the 'INSTALL user-id' and to tables which belong to the 'SYSTEM user-id'.
6.2 Physical Integrity

6.2.1 Concurrency Control

In Business System 12 many users can operate simultaneously. In general they will not be aware of each others presence. By suitably delimiting units of work (transactions) with \textit{START/END UNIT} commands, a user is able to keep his view of the database constant during his transaction. Single commands and CLISTS are implicit transactions.

A lock manager who locks complete tables for either shared or exclusive access ensures maximum integrity.

6.2.2 Crash Recovery

Transaction backout is fully supported by Business System 12. In case of a system failure, all non-committed transactions at the time of failure will be rolled back when the system restarts.

Roll-forward is not supported. As a consequence, a disk crash of the database will require a restore to the status of the last backup.

Business System 12 provides the following backup and recovery functions for its data bases:

- Backup of complete segments
- On-line restoration of complete segments
- On-line restoration of a complete user
- On-line restoration of a specific table of a user
- Limited reorganization of segments

Segment 1 of the Business System 12 data base plays a special role for the backup and recovery of the data base. It is reserved for the Data Maintenance Administrator, the Backup and Recover APs, and the tables related to the backup and recovery of the Business System 12 data base. These tables are referred to as data maintenance tables.

The Data Maintenance Administrator is a special Business System 12 user that is responsible for the planning of the regular backups and reorganizations of the Business System 12 data base and the restoration of parts of the data base as required by unusual circumstances.

Business System 12 allows multiple backup, reorganization, or restoration operations to proceed in parallel. For this purpose, the concept of Backup and Recover APs is introduced. Backup and Recover APs are Business System 12 users with special authorities that perform the actual backup, reorganization, or restoration of the individual parts of the Business System 12 data base. It is not possible to backup or reorganize an individual user or even a specific table. Also, differential backup of segments is not provided. That means that always all tables of a segment are backed up regardless of whether or not they were modified since the previous backup for the segment.

There are two sets of data maintenance tables. The tables which the Data Maintenance Administrator must provide in order to describe which
objects of the data base are to be backed up and reorganized and in order to supply the JCL for the Recover APs, and the tables that are created by the backup, reorganization, or restoration process (providing status and history information).

6.3 Operating Environment

6.3.1 Software Environment (Operating System)

Business System 12 runs with the MVS operating system.

6.3.2 Hardware Environment (CPU, Memory, Peripherals)

The hardware environment is an IBM-INS Compute Center.
7.0 ESSENTIALLY RELATIONAL SOLUTIONS FOR GENERALIZED DBMS PROBLEMS

The **claimed** advantages of relational systems which are incorporated in Business System 12 are:

- **Simplicity.**
  The only visible objects in the system are *tables* with *rows* and *columns*.
  The twelve relational commands cover the relational algebra and provide a powerful query language.

- **Uniformity.**
  Closure is provided through the relational commands. There is no other way to interact with the database than via relational queries.

- **Data independence.**
  Access Paths are not visible to the user, neither are indexes. Ordering is only achieved through explicit request.

- **Permits optimization.**
  Since no navigational details can be specified or seen by the user, the system decides how to perform data access and tries to do so in the best way. This optimization is done by transforming the access tree of a relational query into its optimal form. For example, **SELECT**s are performed first to limit the number of **rows** flowing through the tree; **Indexes** are built to optimize **JOIN** performance; **Information** is preserved to be able to do keyed access instead of repeated sequential scan if possible; etc.

- **Basis for high level interfaces.** The only interface to Business System 12 is the Application Programming Interface, which is a high level interface.

- **Multiple views of data.**
  The user can define views on his data any way he desires.

- **Security.**
  Tables are only accessible via relational queries and those are always subject to authorization control.

- **Basis for database semantics.** Domain definitions, **CLISTS**, user **VIEWs** and **FUNCTIONs** can be used to impose some form of semantics on the user data.
8.0 DATABASE APPLICATIONS USING THE SYSTEM

A general application offered by the IBM International Network Services is the XSHARE database. It is essentially an extensive portfolio application.

Other applications are the property of the customers who developed them.
A.0 FEATURE SUMMARY

The tables given here are identical to the ones in chapter 4 of the Relational Databases analysis.

We hope that our interpretation of the features of Business System 12 would coincide more or less with that of the authors of the book. In that case Business System 12 can be compared to the features of the systems described there.

The terminology used is again that of the book. Where necessary explanations are provided.

A.1 Database constituents

<table>
<thead>
<tr>
<th></th>
<th>Business System 12</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Database</td>
<td>yes</td>
</tr>
<tr>
<td>2. Relation</td>
<td>yes</td>
</tr>
<tr>
<td>3. View</td>
<td>yes</td>
</tr>
<tr>
<td>4. Snapshot</td>
<td>yes</td>
</tr>
<tr>
<td>5. Tuple</td>
<td>yes</td>
</tr>
<tr>
<td>6. Attribute</td>
<td>yes</td>
</tr>
<tr>
<td>7. Domain (user defined)</td>
<td>yes</td>
</tr>
<tr>
<td>8. Other</td>
<td>transaction, trigger</td>
</tr>
<tr>
<td></td>
<td>hash index, clists</td>
</tr>
<tr>
<td></td>
<td>functions, dictionary,</td>
</tr>
<tr>
<td></td>
<td>scratchpad</td>
</tr>
</tbody>
</table>
### A.2 Operations

<table>
<thead>
<tr>
<th>Feature</th>
<th>Business System 12</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Retrieval tuple handling</td>
<td>Duplicates always removed</td>
</tr>
<tr>
<td>2. Nesting</td>
<td>Fully</td>
</tr>
<tr>
<td>3. Closure</td>
<td>fully closed</td>
</tr>
<tr>
<td>4. Set membership operator</td>
<td>yes</td>
</tr>
<tr>
<td>5. Set operators (union, intersection, difference, equality)</td>
<td>yes</td>
</tr>
<tr>
<td>6. Group by operator</td>
<td>yes</td>
</tr>
<tr>
<td>7. Arithmetic operators</td>
<td>yes</td>
</tr>
<tr>
<td>8. String operators</td>
<td>yes</td>
</tr>
<tr>
<td>9. Transaction operators</td>
<td>yes</td>
</tr>
<tr>
<td>10. Existential quantifier</td>
<td>implied</td>
</tr>
<tr>
<td>11. Universal quantifier</td>
<td>implied</td>
</tr>
<tr>
<td>12. Boolean operators</td>
<td>yes</td>
</tr>
<tr>
<td>13. Query language environment</td>
<td>stand alone and host</td>
</tr>
<tr>
<td>14. Self-join capability</td>
<td>full</td>
</tr>
<tr>
<td>15. Equi-join capability</td>
<td>full</td>
</tr>
<tr>
<td>16. Natural join capability</td>
<td>full</td>
</tr>
<tr>
<td>17. Projection capability</td>
<td>Explicit</td>
</tr>
<tr>
<td>18. Sort capability</td>
<td>Explicit</td>
</tr>
<tr>
<td>19. Insert</td>
<td>Explicit</td>
</tr>
<tr>
<td>20. Delete</td>
<td>Explicit</td>
</tr>
<tr>
<td>21. Modify</td>
<td>Explicit</td>
</tr>
<tr>
<td>22. Update over more than one relation</td>
<td>yes</td>
</tr>
<tr>
<td>23. Commit / undo</td>
<td>Explicit</td>
</tr>
<tr>
<td>24. Triggers</td>
<td>yes</td>
</tr>
<tr>
<td>25. Aggregate functions</td>
<td>yes</td>
</tr>
<tr>
<td>26. User defined functions</td>
<td>yes</td>
</tr>
<tr>
<td>27. Library functions</td>
<td>yes</td>
</tr>
<tr>
<td>28. User defined error conditions (exit capabilities)</td>
<td>yes</td>
</tr>
<tr>
<td>29. Format of schema information</td>
<td>Relational</td>
</tr>
<tr>
<td>30. System performance data available</td>
<td>no</td>
</tr>
<tr>
<td>31. Report generator</td>
<td>Restricted</td>
</tr>
<tr>
<td>32. Relationally complete</td>
<td>yes</td>
</tr>
</tbody>
</table>
### A.3 Schema definitions

<table>
<thead>
<tr>
<th>Feature</th>
<th>Business System 12</th>
</tr>
</thead>
<tbody>
<tr>
<td>Define database</td>
<td>yes</td>
</tr>
<tr>
<td>Generate database</td>
<td>yes</td>
</tr>
<tr>
<td>Fast load / unload</td>
<td>yes</td>
</tr>
<tr>
<td>Destroy database</td>
<td>yes</td>
</tr>
<tr>
<td>Define relation</td>
<td>yes</td>
</tr>
<tr>
<td>Destroy relation</td>
<td>yes</td>
</tr>
<tr>
<td>Reorganize relation</td>
<td>no</td>
</tr>
<tr>
<td>Define snapshot</td>
<td>yes</td>
</tr>
<tr>
<td>Define view</td>
<td>yes</td>
</tr>
<tr>
<td>Drop attribute</td>
<td>no</td>
</tr>
<tr>
<td>Add attribute</td>
<td>no</td>
</tr>
<tr>
<td>Rename attribute</td>
<td>no</td>
</tr>
<tr>
<td>Define tuple</td>
<td>no</td>
</tr>
<tr>
<td>Other</td>
<td></td>
</tr>
</tbody>
</table>

### A.4 Additional facilities

<table>
<thead>
<tr>
<th>Feature</th>
<th>Business System 12</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attribute synonyms</td>
<td>no</td>
</tr>
<tr>
<td>Relation synonyms</td>
<td>yes</td>
</tr>
<tr>
<td>Duplicate tuples</td>
<td>no</td>
</tr>
<tr>
<td>Can view be defined over more than one relation</td>
<td>yes</td>
</tr>
<tr>
<td>Can view be defined from other views</td>
<td>yes</td>
</tr>
<tr>
<td>User/DBA can define mapping to update view explicitly</td>
<td>yes</td>
</tr>
<tr>
<td>Selection criterion</td>
<td>yes</td>
</tr>
<tr>
<td>Aggregate function</td>
<td>yes</td>
</tr>
<tr>
<td>Primary keys required</td>
<td>yes</td>
</tr>
<tr>
<td>Key can be modified</td>
<td>no</td>
</tr>
<tr>
<td>Key can be declared unique</td>
<td>always</td>
</tr>
<tr>
<td>Concatenated primary key</td>
<td>yes</td>
</tr>
<tr>
<td>Views inherit keys</td>
<td>yes</td>
</tr>
<tr>
<td>Null valued keys allowed</td>
<td>no</td>
</tr>
</tbody>
</table>
### A.5 Functional Classes

<table>
<thead>
<tr>
<th>Business System 12</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Database schema definition</td>
</tr>
<tr>
<td>2. Database retrieval</td>
</tr>
<tr>
<td>Set-oriented</td>
</tr>
<tr>
<td>Record-at-a-time</td>
</tr>
<tr>
<td>Yes</td>
</tr>
<tr>
<td>Yes</td>
</tr>
<tr>
<td>3a) Database insert</td>
</tr>
<tr>
<td>Set-oriented</td>
</tr>
<tr>
<td>Record-at-a-time</td>
</tr>
<tr>
<td>Yes</td>
</tr>
<tr>
<td>3b) Database modify</td>
</tr>
<tr>
<td>Set-oriented</td>
</tr>
<tr>
<td>Record-at-a-time</td>
</tr>
<tr>
<td>Yes</td>
</tr>
<tr>
<td>3c) Database delete</td>
</tr>
<tr>
<td>Set-oriented</td>
</tr>
<tr>
<td>Record-at-a-time</td>
</tr>
<tr>
<td>Yes</td>
</tr>
<tr>
<td>Yes</td>
</tr>
<tr>
<td>4. Integrity constraints</td>
</tr>
<tr>
<td>Yes</td>
</tr>
<tr>
<td>Yes</td>
</tr>
<tr>
<td>5. Database generation, regeneration</td>
</tr>
<tr>
<td>Yes</td>
</tr>
<tr>
<td>Yes</td>
</tr>
<tr>
<td>6. Database schema redefinition and renaming</td>
</tr>
<tr>
<td>Yes</td>
</tr>
<tr>
<td>Yes</td>
</tr>
<tr>
<td>7. Report generation</td>
</tr>
<tr>
<td>Yes</td>
</tr>
<tr>
<td>Yes</td>
</tr>
<tr>
<td>8. Special data entry</td>
</tr>
<tr>
<td>Yes</td>
</tr>
<tr>
<td>Yes</td>
</tr>
<tr>
<td>9. Security, monitoring</td>
</tr>
<tr>
<td>Yes</td>
</tr>
<tr>
<td>Yes</td>
</tr>
<tr>
<td>10. Load/dump/recovery</td>
</tr>
<tr>
<td>Yes</td>
</tr>
<tr>
<td>Yes</td>
</tr>
<tr>
<td>11. Definition of access paths</td>
</tr>
<tr>
<td>Yes</td>
</tr>
<tr>
<td>Yes</td>
</tr>
<tr>
<td>12. Database dictionary</td>
</tr>
<tr>
<td>Yes</td>
</tr>
<tr>
<td>Yes</td>
</tr>
</tbody>
</table>

### A.6 Interface Flavors

<table>
<thead>
<tr>
<th>Host programming language</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interactive end user</td>
</tr>
<tr>
<td>Processor calls</td>
</tr>
<tr>
<td>New language special interface</td>
</tr>
<tr>
<td>BS12</td>
</tr>
<tr>
<td>Yes</td>
</tr>
<tr>
<td>Yes</td>
</tr>
</tbody>
</table>

Feature Summary 38
### A.7 Operational Aspects

<table>
<thead>
<tr>
<th>Feature</th>
<th>Business System 12</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Security (user related)</td>
<td></td>
</tr>
<tr>
<td>a. User id from O.S.</td>
<td>yes</td>
</tr>
<tr>
<td>b. Passwords</td>
<td>yes</td>
</tr>
<tr>
<td>c. File access</td>
<td></td>
</tr>
<tr>
<td>2. Security (data related)</td>
<td></td>
</tr>
<tr>
<td>a. Item of protection</td>
<td></td>
</tr>
<tr>
<td>b. Based on data value</td>
<td>relation/view clist</td>
</tr>
<tr>
<td>3. Concurrency control</td>
<td></td>
</tr>
<tr>
<td>a. Multiple user update</td>
<td>yes</td>
</tr>
<tr>
<td>b. Multiple commands per transaction</td>
<td>yes</td>
</tr>
<tr>
<td>c. Degrees of consistency</td>
<td>3</td>
</tr>
<tr>
<td>d. Explicit locks</td>
<td>no</td>
</tr>
<tr>
<td>4. Crash recovery</td>
<td></td>
</tr>
<tr>
<td>a. DB backup/restore</td>
<td>yes</td>
</tr>
<tr>
<td>b. Logging</td>
<td>yes, for undoing</td>
</tr>
<tr>
<td>c. Recovery from system crashes</td>
<td>no, for disk crash yes</td>
</tr>
</tbody>
</table>
ILLC Scientific Publications

Citing for Reports and Dissertations: Series-Year-Number, with CT = Computation and Complexity Theory; LP = Logic, Philosophy and Linguistics; ML = Mathematical Logic and Foundations; X = Technical Notes; MoL = Master of Logic Thesis; DS = Dissertations.

All previous ILLC-publications are available from the ILLC bureau. For prepublications before 1994, contact the bureau.

CT-1996-01 Peter van Emde Boas The Convenience of Tilings
CT-1996-02 A.S. Troelstra From Constructivism to Computer Science
CT-1997-01 Carl H. Smith, Rūsiņš Freivalds Catagory, Measure, Inductive Inference: A Triviality Theorem and its Applications
CT-1997-02 Peter van Emde Boas Resistance is Futile; Formal Linguistic Observations on Design Patterns
CT-1997-03 Harry Buhrman, Dieter van Melkebeek Complete Sets under Non-Adaptive Reductions are Scarce
CT-1997-04 Andrei Muchnik, Andrei Romanov, Alexander Shen, Nikolai Vereshchagin Upper Semi-Lattice of Binary Strings with the Relation "z is simple conditional to y"
CT-1998-01 Hans de Nivelle Resolution Decides the Guarded Fragment
CT-1998-02 Renata Wassermann On Structured Belief Bases - Preliminary Report
CT-1998-03 Johan van Benthem Temporal Patterns and Modal Structure
CT-1998-04 Ghica van Emde Boas-Lubsen, Peter van Emde Boas Compiling Horn-Clause Rules in IBM’s Business System 12 - an Early Experiment in Declarativeness
LP-1996-01 Renate Bartsch Understanding Understanding
LP-1996-02 David Beaver Presupposition
LP-1996-03 Theo M.V. Janssen Compositionality
LP-1996-04 Reinhard Muskens, Johan van Benthem, Albert Visser Dynamics
LP-1996-05 Dick de Jongh, Makoto Kanazawa Anglin’s Theorem for Indexed Families of R.E. Sets and Applications
LP-1996-06 François Lepage, Serge Lapierre The Functional Completeness of 4-value Monotonic Prothetics
LP-1996-07 Frans Voorbraak Probabilistic Belief Expansion and Conditioning
LP-1996-08 John Case The Power of Vacillation in Language Learning
LP-1996-09 Jaap van der Does, Willem Groeneveld, Frank Veltman An Update on Might
LP-1996-10 Jelle Gerbrandy, Willem Groeneveld Reasoning about Information Change
LP-1996-11 Renate Bartsch Propositional Attitudes in Dynamic Conceptual Semantics
LP-1996-12 Paul Dekker Reference and Representation
LP-1996-13 Rens Bod, Remko Scha Data-Oriented Language Processing: An Overview
LP-1996-14 Michiel van Lambalgen, Jaap van der Does A Logic of Vision: Preliminaries (preliminary to LP-1997-07: updated version on author’s homepage)
LP-1997-01 Johan van Benthem Dynamic Bits and Pieces
LP-1997-02 Paul Dekker On Denoting Descriptions
LP-1997-03 Paul Dekker On First Order Information Exchange
LP-1997-04 Jelle Gerbrandy Dynamic Epistemic Logic
LP-1997-05 Jelle Gerbrandy Bisimulation and Bounded Bisimulation
LP-1997-06 Jan van Eijck Typed Logic With States
LP-1997-07 Michiel van Lambalgen, Jaap van der Does A Logic of Vision (expansion of LP-1996-14)
LP-1997-08 Johan van Benthem Wider Still and Wider... Resetting the Bounds of Logic
LP-1997-09 Frans Voorbraak A Nonmonotonic Observation Logic
LP-1997-10 Jan van Eijck Dynamic Reasoning Without Variables
LP-1998-01 Hans Rott, Maurice Pagliuca Severe Withdrawal (and Recovery)
LP-1998-02 Jaap van der Does, Helen de Hoop Type-shifting and Scrambled Definites
DS-1996-03 Martin H. van den Berg *Some Aspects of the Internal Structure of Discourse: the Dynamics of Nominal Anaphora*

DS-1996-04 Jeroen Bruggeman *Formalising Organizational Ecology*

DS-1997-01 Ronald Cramer *Modular Design of Secure yet Practical Cryptographic Protocols*

DS-1997-02 Nataša Rakić *Common Sense Time and Special Relativity*

DS-1997-03 Arthur Nieuwendijk *On Logic. Inquiries into the Justification of Deduction*

DS-1997-04 Atocha Aliseda-LLera *Seeking Explanations: Abduction in Logic, Philosophy of Science and Artificial Intelligence*

DS-1997-05 Harry Stein *The Fiber and the Fabric: An Inquiry into Wittgenstein’s Views on Rule-Following and Linguistic Normativity*


DS-1998-01 Sebastiaan A. Terwijn *Computability and Measure*

DS-1998-02 Sjoerd D. Zwart *Approach to the Truth: Verisimilitude and Truthlikeness*

DS-1998-03 Peter Grunwald *not yet available*