UNIT 12 RELATIONAL DATA BASE MANAGEMENT SYSTEM

Objective

After going through this unit, you should be able to:

- Identify the relational approach as one which organizes data in tables
- enumerate various options for field type specifications
- define the meaning of relational operations such as SELECT, PROJECT, JOIN etc.
- identify and enumerate the features suggested by E.F. Codd for determining how relational **a** DBMS product is.

Structure

12.1 Introduction

12.2 Relational data model

12.3 Relational operations

12.4 The12 commandments of CODD

12.5 Examples of relational data base

12.6 Summary

12.7 Self assessment exercises

12.8 Further Readings

12 .1 INTRODUCTION

The relational system is a major development in database management even though full-fledged relational DBMSs became available commercially only in the early 1980s, more than a decade after the network and the hierarchical systems' appeared.. The relational approach is substantially different from other database. Approaches in Urns of its logical structures and mode of I/O operations. In the relational approach, data are organized into tables called relations, each of which is implemented as a file. A row in a relation is called a topple in the relational. terminology, and it represents a record or an entity, each column in a relation represents a field or an attribute implemented as fields. For example, a CUSTOMER relation consists of a number of customer entities. The, characteristics of a customer entity are described by its attributes such as customer's number, customer's name and customer's address.

The following are some relational terms and their equivalent conventional names:

Relational terms	Conventional terms
Relation, base table	file
tuple	record, entity
column	field, attribute
foreign key	connection field

One of the characteristic of the relational approach is the simplicity of its logical representation. For the users, tables are much easier to understand than complex tree or network structures.

12.2 RELATIONAL DATA MODEL

Logically, a relational data model consists of a collection of tables, each of which represents a conceptual record type. Thus, the schema for a relational system includes data definitions for a number of base tables. A base table can be described with the SQL as follows:

CREATE TABLE base-table-name (field-named data-type(NOT NULL)

field –name 1 data-type (NOT NULL)

.field –name-n-type (NOT NULL)

The tables given at the end of this section illustrates the relational data model. The relational data model is composed of three relations, INVENTORY, CUSTOMER and TRANSACTION. A one-to-many relationship between INVENTORY and TRANSACTION is implicitly established by the presence of a common field (INV-NO) in both relations. Similarly, the one-to-many relationship between CUSTOMER and INVENTORY is implicitly described by the presence of the common field, CUST-ID.

The set of statements below shows the creation of the relations given above with the SQL CREATE TABLE command. The following are various options *for* a field type specification:

(1) CHAR:a fixed-length character string

(2) VARCHAR:a variable-length character string

(3) INTEGER:a full word binary integer

(4) FLOAT:a floating point number

(5) DECIMAL (m,n):a decimal number of m digits with n digits after the decimal point.

INVENTORY (INV-NO, INV-NAME, UNIT-PRICE)

CUSTOMER (CUST-ID, CUST-NAME)

TRANSACTION (TRANS-NO, INV-NO, CUST-ID-, QTY, DATE-OF-TRANS)

CREATE TABLE INVENTORY

(INV-NO)	CHAR(5)NOT NULL.
INV-NAME	VARCHAR (15)
UNIT-PRICE	FLOAT

CREATE TABLE CUSTOMER

(CUST-ID	CHAR(4)NOT NULL.
CUST-NAME	VARCHAR (20))

CREATE TABLE TRANSACTION

(TRANS-NO	CHAR(4)NOT NULL.
INV-NO	CHAR(5)NOT NULL.
CUST-ID	CHAR(4)NOT NULL.
QTY	INTEGER
DATE-OF-TRANS	CHAR(4))

The NOT NULL option is used to specify that the content of a data field should not be left undefined. The primary key is not explicitly declared, but the primary key and all foreign keys must be specified at NOT NULL.

In SQL/DS or DB2, new fields may be added to an existing base table with an ALTER TABLE command. In SYSTEM R, however, an EXPAND TABLE command is used instead. For example:

SQL/DS or DB2 ALTER TABLE CUSTOMER ADD CUST-ADDR

CHAR (20)

SYSTEM R;EXPAND TABLE CUSTOMER ADD CUST-ADDR

CHAR (20)

A new field, CUST-ADDR, with a data type CHAR(20) is added to the CUSTOMER base table. These commands allow users to expand a base table even after it is loaded with data.

Once a table is defined with a CREATE TABLE command, a new empty base table is created, and the table may be loaded immediately with an interactive SQL INSERT

Command, For example, the following statement inserts a new record in the CUSTOMER base table:

INSERT INTO CUSTOMER VALUES ('C3', 'JANE')

In SQL/DS, a system supplied utility called Data Base Services (DBS) can be used to initially load or Odd rows to tables from a sequential file. In DB2, the LOAD utility is used to load base tables with sequential files.

A base table may be deleted with a DROP TABLE statement as follows:

DROP TABLE base table name.

INVENTORY

INV-NO	INV-NAME	UNIT-PRICE
11	CHAIR	75.00
12	TABLE	259.15
13	DESK	399.00

CUSTOMER

CUST-ID	CUST-NAME
C1	DAVE
C2	EDDIE
C3	JANE

TRANSACTION

TRANS-NO	INV-NO	CUST-ID	QTY	DATE-OF-TRANS(DD/MM)
T1	11	C1	12	05/06
T2	13	C1	2	12/08
13	12	C2	1	25/07
14	13	C3	2	21/10

Sample data for the relations: INVENTORY, CUSTOMER and TRANSACTION.

12.3 RELATIONAL OPERATIONS

The following operations can be performed on the tables illustrated in the previous section.

1) **SELECT operation**

The SELECT operation is used to select rows from a table or, to select records from a file, in the conventional terminology,

2) PROJECT operation

A PROJECT operation is used to select desired columns (or vertical subsets) from a source relation. Duplicated tuples will be automatically eliminated from the resulting relation.

3) **JOIN operation**

The JOIN operation combines two tables horizontally over common values in a specified field of each relation. The two fields to be compared must have a common domain.

The JOIN operation is accomplished by comparing each record in the first table with every record in the second table for a possible match in the specified field.

The two records are joined and all the records thus combined from the new relation. The JOIN operation terminates when the last record in the first table has been compared with all records in the second.

4) **DIVISION operation**

The DIVISION operation selects rows from a table based on a range of values specified in another table. To perform such an operation, we first sort the first relation on the

ascending order of say CUST-ID value will be selected from the first table if its associated INV-NO values specified in the second.

Not all commercial DBMSs can perform all the functions described above.

Set Operations

Three traditional set operators of relational algebra are UNION, DIFFERENCE, and INTERSECTION. These three operators can be applied only to source relations that have compatible layouts. In other words, operand relations must be of the same degree (same number of fields), and the corresponding columns of each relation must have the same domain. Using the two such source relations each of the three operations can be performed as follows:

(1) **UNION Operation**

The UNION operation merges records in two tables.

(2) **DIFFERENCE Operation**

The DIFFERENCE operation is similar to subtraction. It removes from a relation those

Records that appear in another relation.

(3) **INTERSECTION Operation**

The intersection of two relations produces a new relation which consists only of records which belong to both source relations.

12.4 THE 12 COMMANDMENTS OF CODD

12 COMMANDMENTS FOR DETERMINING HOW RELATIONAL A DBMS PRODUCT IS;

Information Rule: Tables format

Rule 1: All information in a relational data base is represented explicitly at the logical level and in exactly one way - by values in R-tables.

Guaranteed Access Rule: via combination of table name/primary key, column name Rule 2: Each and every datum (atomic value) in a relational data base is guaranteed to be logically accessible by resorting to a combination of R-table name, primary key value and column name.

Systematic treatment of null values

Rule 3: Indicator (distinct from the empty character string or a string of blank characters, and distinct from zero or any other number) are supported in fully relational DBMS for representing at the logical level the fact that the information is missing (applicable and inapplicable information) in a systematic way - independent of data type. Besides the logical representation, the DBMS must support manipulative functions for these indicators and these must also be independent of the data type of the. missing information.

Dynamic on-line catalogue in relational form--so users can interrogate it

Rule 4: The data base description is represented at the logical level just like ordinary data, so that authorized users can apply the same relational language to its interrogation as they apply to the regular data.

Comprehensive data sub-language rule--one language for every thing, data/view . Definition, data manipulation integrity constraints etc.

Rule 5: A relational DBMS (no matter how many languages and what modes of terminal use it supports - for example, the fill in the blanks mode) must support at least one language (1) whose statements are expressible per some well defined syntax as character strings; and (2) which is comprehensive in supporting all of the following terms:

1. Data definition
2. View definition

3. Data manipulation (interactive and by program)

4. Integrity constraints

5. Authorization

6. Transaction boundaries (begin, commit and roll-back)

View updating rule--all views updatable

Rule 6: The DBMS includes an algorithm for determining (at view definition time) whether that view is tuple-insert able and tuple-delete able; and whether each of its columns is updatable. It records the result of this investigation in the catalogue.

High level INSERT/UPDATE/DELETE One command updates many records.

Rule 7: The capability of handling a base relation or a derived relation as a single operand applies not only to the retrieval of data but also the insertion, update and deletion of data.

Physical data independence--users not affected by changes to storage representation to access method

Rule 8: Application programs and terminal activities remain logically unimpaired whenever any changes are made in either storage representation or access methods.

Logical data independence–users not affected by change to the base tables that preserve information

Rule 9: Application programs and terminal activities remain logically unimpaired when information-preserving changes of any kind that theoretically permit unimpairment are made to the base tables.

Integrity independence–integrity constraints defined in the catalogue, not programs. Include

-- Entity integrity -- no prime key rule

-- Referential integrity -- a prime key exists for each non-nul foreign key

Rule 10: Integrity constraints specific to a particular relational data base must be definable in the relational data sub-language and storable in the catalogue (not in the application programs).

Distribution independence–users not affected by distribution and re-distribution of data. All data appears local to the site

Rule 11: A relational DBMS has distribution independence.

Non-Subversion rule-- no low level interfaces to bypass integrity rules

Rule 12: If a relational system has a low-level (single-record-at-a-time) language, that low level cannot be used to subvert or bypass the integrity rules and constraints expressed in the higher-level relational language (multiple-records-at-a-time).

12.5 EXAMPLES OF RELATIONAL DATA BASE

Mention of some Relational Databases has been made earlier. A number of products are now available in the market which fulfils broadly requirements of a relational database enumerated in the earlier sub-section 12.4.

The number of products available worldwide may be very large, and a list of almost 50 commercial well-known products is given in alphabetically order as follows:

ACCENT R	MILLDATA
ADABAS/NATURAL	MIMER
AMBASE	MISTRESS
ARCHON: QDMS	OMNIBASE
AUTOPRO	ORACLE
BASIS-DM	PEDMS
CLIO	QDMS
CORTEX	RAMIS II
CUPID	RAPPORT
DATA BOSS/4	RDB
DATA BOSS/32	RELIANT
DM	REVEAL-DBMS
DATACOM/DB	RTFILE
DB2	SEED-DBMS
FOCUS	SEQUITUR
GEM	SIMBAD
GUVNOR	SIR
IMPRS	SYBASE
INFO	SYSTEM 1032
INFOCEN	SYSTEM 2000
INFORM	TOTAL
INFORMIX	ULTRA
INGRES	UNIFY
INTAC	USER II
MAPS/DB	X-AMPLE

When an organisation decides to go in for a relational database it adopts certain criteria, which depend on the specific need of the organisation. However, a fairly reasonable check-list to pick out the relevant criteria for short-listings relational database would be as follows:

Short-List Criteria

1.	Portability:	Must be easily transportable from machine to machine regardless of type. Must be able to run under UNIX. Software produced independent of hardware manufacturer preferable.
2.	Programmability:	Must allow for the production of maintainable application code and macros.Must have host language interface for commonly used DP languages, i.e. COBOL
3.	Query Language:	Must provide a query language processor.
4.	Report Writer:	Must provide a report writing facility.
5.	Data Dictionary:	An integrated data dictionary must be Provided as an integral of the system.

6.	Security/integrity:	Security features such as automatic rollback recovery, as well as data integrity features, such as password protection and record locking, must be in evidence
7.	Database:	Full support for relational database design Techniques must be in evidence.
8.	Screen generators:	A facility for producing data Entry/retrieval/update screens must be available.
9.	User-base/Stability:	A good user-base in the country and a good Stable market position must be evident.
10.	PC Version:	A version of the product must be available For PC use, and must be compatible with the full version.
11.	Other:	Must show no bias towards a specific Discipline e.g. accounting, statistics etc.

When such criteria are applied the actual product which is most appropriate for an organisation may be any of these, but the most important market players who have acquired reasonable significant share of the market players are as follows: (arranged in alphabetical order) Informix, Ingress, Oracle and Unify.

Of these Oracle and Ingress are almost in neck and neck competition and each subsequent release try to uncalculated features which have been found to be appreciated by the competing product. The information is based on data sheets of vendors and is meant to identify points of comparison, rather than conveying whether a product is superior.

The final choice of the most appropriate package would need a detailed consideration of technical and administrative criteria as indicated in Unit 14.

FEATURE	INFORMIX	UNIFY
1. Program Development	Need not have to be a programmer to develop programs.	Has to be more of 'C' programmer to develop applications.
2. Implementation	Full ANSI SQL compatible with	Limited SQL Implementation with
	- Data definitions	- No data definition statement
	- Data control	- No data control statement
	- Integrated SQL Dictionary	- No callable data dictionaries
3. Embedded SQL Products	Available - ESQL /C - ESQL /COBOL	No Embedded SQL Products No dynamic capabilities
	- ESQL /ADA	Must use 'C' function library
4. Integration with Office automation products	Full Integration between SQL and "SMART WARE" office automation tools namely	No Integration with automation products.
	- Spread Sheet with business graphics - Word Processor - Database Manager - Communications - Application generator	
5. Support of Networks	AT&T STARLAN, TCP/1P DEC NET, MS-NET Like NOVELL, 3COM etc.	DOS NETWORKS Like NOVELL etc. are not available. Limited networking capabilities under Unix environment.
6. Maintenance	Reconfiguring of database is very simple. Automatic reconfiguration takes placae while changing tables and adding Indexes.	Have to reconfigure entire database to change tables add Indexes, change pointers etc.
	The DBCHEK facility helps to retrieve the corrupted table information	Have to rebuilt entire database if single table gets corrupted
	There is no concept of pointers	Pointers used for explicit relationships (pre-joins) becomes corrupted and has be rebuilt (REPOINTS)

SL NO.	FUNCTION	INFORMIX	ORACLE
1.	Features of 4GL debugger along with SQL based tools.	Full 4GL Interpreter/ complex applications would require 3GL interface.	Limited 4GL functions
2.	Windowing capability	Displays 2 lines	Triggers several pages
3.	Expanded memory	Can run without expanded memory Runs on 640K DOS	Can not run without extended memory.
4.	DOS NETWORK	NOVELL, 3 COM	No DOS Network etc. products
5.	Spread Sheet features	Full fledged Spread Sheet interface with Business Graphics and macro are available.	ORACLE-CALC has no macros and limited financial functions + expensive.
6.	Portability	Same versions run on various machines.	Several different incompatible versions exist on various machines.
7.	Currency data types	YES	No money data type very difficult to format money in forms, reports.
8.	Recovery process	Turbo module is available which is fault tolerant and automatic recovery module.	Entire database reside in a single file. If any parts gets corrupted, no way to recover except to back up.
9.	Embedded Languages	Informix provide an embedded approach to using SQL from C. The SQL statements are embedded into the source code (This source code is pre-processed by a utility provided with the data manager and compiled and linked as normal Ccode)	Oracle provides a call level approach to embedding SQL in C Code (SQL statements are passed to Oracle sub-routines as strings of character. Therefore, there is no pre-processor step however). The code is readable since the queries are burried in functions call argument lists.
10.	Exporting data files.	Easy to export data files.	Oracle's ODL facility does not provide a means of determining the actual space being consumed by an individual table.

11. Table size	With Informix one can look at file size and get an accurate size for the tables.	Oracle does not provide a means of determining the actual space being consumed by an individual table.
12. Database size	No limit on database size.	pre-allocation of space is required. One has to use a trial and error approach to determine the amount of space to use.
	informix dynamically size the files associated with the database.	There is no way to shrink an oversized database, other than to export the entire database and to re-create space.
13. Possibility of obtaining run-time versions	Runtime and development versions are separately available.	No run-time versions available.
14. Report generation features	Superior reporting statements are available in this product.	All required functions of report writer are not available.
15. Market spread (UNIX)	Large number of installations since it is designed to operate under UNIX.	Not much installations under UNIX.
16. Graphic capability	Interface is available thru' 'SMARTWARE' Spread Sheet.	Not available in UNIX environment.
17. Availability in UNIX V.3	Runs under UNIX V.3	UNIX V.3 is not available as of today.
18. Security features	Database, user, table page, record and field level security features supported.	Page level security is not available.
19. Development tools	All development tools like data base creation, report writer, query, Form generator, menu driven table creation are	Weak in report writer only SQL is available for table creation.
20. User friendliness/ learn time	Menu driven non-procedural and has English language like commands.	Not very user-friendly. Non uniform function key assignment. Forms handling is cumbersome.
21. Integrated package	products available as one package. It is easier to upgrade.	Not available as on integrated packages which could result into confusion during integration.

22. Single screen multi-table access	Allows such an access.	Only base table is accessible from one form, the others can be accessed through an interface (Triggers).
23. Documentation quality of manuals	More examples and better structure of manuals.	Fewer examples.
24. Product reliability	More user experience in UNIX world.	Fewer installations in UNIX environment.

12.6 SUMMARY

In relational database systems, data are organised into tables, or relations. Data relationships between relations are implicitly established via foreign keys. Each relation is implemented as a separate file called a base table or a stored table, and indexes can be created for random access of records in a table via either primary or secondary keys. A data sub-model is the relational term for an external schema. It consists of one or more views which derive data from one or more tables.

The two relational sublanguages called relational algebra and relational calculus suggested by Codd have triggered much research and development in the relational data manipulation languages. One of the unique features of the relational DML is its ability to manipulate relations in their entirely. In other words, a whole table may be retrieved as a result of a single SQL statement.

Some basic operations of the relational algebra described are SELECT, PROJECT, JOIN, DIVISION, UNION, DIFFERENCE, and INTERSECT. On the other hand, the relational calculus is a nonprocedural language with which the user specified fields to be retrieved and a predicate to indicate the selection criteria. Thus, the user can specify what is needed without having to code the detailed procedural steps to achieve it.

12.7 SELF ASSESSMENT EXERCISE

1. With reference to your own organisation, further refine the short-listing criterion given in section 12.5 to 5 most important ones.

2. Constant dummy data for the 3 tables show in Section 12.2 and see for yourself how you could use the INSERT statement to add one record to each of the tables.

3. Elaborate role 8 of codds commandments to illustrates its managerial importance and implications.

12.8 FURTHER READINGS

1. Atre S. *Database Structural Techniques for Design, Performance & Management, John Wiley & Sons, 1980*

2. Date C.J. *An Introduction to Database Systems,* Addison-Wesley,1981

3. Hawry Stkiewycz *LT* *Database Analysis and Design, SRA, 1984*

4. Weiderhold, G, *Database Design* 2nd Edition, Mc Grave Hill, 1983

5. Ven Halle Fleming, *Handbook of Relational Database Design,* Wesley, 1990