石田

ABOUT THE BOOK

This book lives up to its title by describing the fundamental concepts behind database systems. Database Systems Concepts behind
database systems. Database Systems Concepts shows how to solve many of
the orohioms encryinglened in dealersies and all the scoling systems the problems encountered in designing and using a database system. Readers are introduced to the entity/relationship and relational models first, followed by the network and hierarchical models. Several chapters are devoted to the physical organization of databases, index techniques, and query processing, and the latter part of the book delves into advanced areas, including coverage of distributed databases, database security, and artificial intelligence.

$\frac{1}{2}$ is the fundamental concepts behind the fundamental concepts behind $\frac{1}{2}$

systems. Database Systems is Network in System Concepts shows the System Concepts shows the many at th University of Texas at Austin. Prior to joining the University of Texas faculty, Dr. Korth was a staff member of the IBM T.J. Watson Research Center in New York, where he was involved in the design and Implementation of a distributed office automation system. His writings on database systems have appeared in several ACM and IEEE publications.

Abraham Silberschatz is Prolessor of Computer Sciences at the University.
of Texas at Austin, where he specializes in the area of concurrent processing. His research interests include operating systems, database systems, distributed in interests include operating systems, database
systems, distributed systems, and programming languages. Dr. Sliberschatz F. K. Korth is Assistant Protection and Coauthor at the Protection of the Prot Concepts textbook. Concepts textbook. was a start member of the IBM μ

Silberts is Provinces at Trumputer Sciences at the University of University of the University of the area at t Interests include operating systems. database programing languages. 0r. Silberschalz is

McGraw-Hill Book Company Serving the Need for Knowledge 1221 Avenue of the Americas New York, N.Y. 10020

<u>Copy of the copy </u>

Computer textbook. Sciences. and coauthor oi the best—selling Operating System

<u>.</u>
Available available a

ISBN 0-07-044752-7

MERGET

HE NR'

ABRA

IBM Ex. 1010

DATABASE

E 2165

'ELATIONA

Ileana-Hut

 MamaNuYorlt.N.V.10020

ABOUT

database

Readers

totlowed

devoted

including

ABOUT

University

Center in NewImplementationdatabase

Abraham

Texas

processing.

Concepts

Korth

book

THE BOOK

introduced

THE AUTHORS

systems have appeare^d

header oi the First DavidBruton

at Austin. where

His

sissr

 \mathcal{C} the Need for Knowledge' 122i

systemsfidistnbo'teii'systems,

problems encountered

FST-IBM000842

McGraw-Hill Advanced Computer Science Series

<u> CONTRACTOR ACTIVITY SERVICE STATIONS AND</u>

Compute-s . Lindsay,

McCraw—Hill

Advanced

 Problem-SolvingLevin, and

Ń

繎

SYSTEMS .

ol the publisher.

Oeut-Dhi752-73

editor was Kaye Face; the production

Planning, Inc.

DODO

designed to the second

Design

FST-IBM000843

Architecture of I'ipelined

Buchanan. Rigenbaum, and

McGraw-Hill Advanced Computer Science Senes

Davis and Lenat: Knowledge-Based Systems in Artificial Intelligence

Rogge: The Architecture of Pipelined Computers

Lindsay, Bachanan, Feigenbaum, and Lederberg: Applications o

tent: Knowledge-Based Systems in Artificial Intelligence ¹¹

leder berg: Applications of Artificial Intelligence . In memory for \mathcal{A} Chemistry: The Dendral Project ... nilsson: Anilsson: A

DATABASE SYSTEM CONCEPTS

Harbison: HYDRA/Cmmp: An uperimental Computer System

uited
t of
writ!
... ni
ct
m uite
st c
wr uite
t e
ivi Copyright © 1986 by McGraw-Hill, Inc. All rights reserved. Printed in the United
States of America. Except as permitted under the United States Copyright Act of
States of America. Except as permitted under the United State States of America. Except as permitted trade the bring distributed in any form or
1976, no part of this publication may be reproduced or distributed in any form or —
—
—

' ISBN

no part at this publication may be reproduced or distributed \mathcal{L} or ' by

by Anne Canevari Green. Project supervision was done by Caliber

Communication and the moment

 $\frac{1}{2}$ written permission in the control of the c

by any sears of the publisher.

Francisco of 7890 DODO 89

ISBN 0-07-044752-7

The editor was Kaye Pace; the production supervisor was Joe Campanella; the

ends the concept of the production of the canonical by Anne Caneva 1986 by McGraw-Hill, Inc. All rights reserved. Printed in the United States reserved. Printed in the United St
Excel
1986 by McGraw-Hill, Inc. 2008 ot America. Exempt or permitted under the United Sutes Copyright Act of . 1976. In 1976 um

 \mathbf{R}

44 Entity-Relationship Model

network

between

the concep^t of relation.

relations represen^t par^t of ^athe tables that were used

the various concepts underlying

banking

- 2.7 Every weak entity set can be converted to a strong entity set by simply adding appropriate attributes. Why, then, do we have weak entities?
- 2.8 Suppose that you design an E-R diagram in which the same entity set appears several times. Why is this a bad practice that should be avoided whenever possible?
- 2.9 When designing an E-R diagram for a particular enterprise, there exists several alternative designs.
	- a. What criteria should you consider in deciding on the
- Every weak entity set converted to a strong entities? simply adding and appropriate choice? $\frac{1}{2}$ and $\frac{1}{2}$ b. Come up with several alternative E. Suppose that you design an E-R and the U-R and the
Suppose that you design an E-R and the U-R and the
 tavor of one of the alternatives.

 $\frac{1}{2}$ bad practice that should be a bad problem possible?

for a particular enterprise, there exists a particular enterprise, there exists a particular enterprise, there exists a particular exists a particular enterprise, there exists a particular exists a particular enterprise, t

 \mathbb{R} and \mathbb{R} are contributed by \mathbb{R} . In the Wang \mathbb{R}

anglegation were introduced introd by Smith and Smith [1977]. benzerini and Santucd [1983] have usedthese concepts in defining cardinality constraints in the E-RF α

discussions are offered by Tsicluitzis and Lochovsky [1982]

concepts of generalization, specialization, and

A BARAGAAN RADA

several times. When α is the second times. When α

=

 \rightarrow

 $\overline{}$

2.10

set

a.What

b.Come

Bibliographic

based

data

 $\mathbb{P}_\mathcal{F}$

 These 1981], andEggplage

 \blacksquare t

by Chen [1983].

entity

Scheuermann in der Scheuermann in

proposed.

The

Basic

and

When designing and the control of the cont

The entity relationship data model was introduced by Chon design are offered by Chen [1977], Sakai [1980], and Ng [1981]. Modeline techniques based on the E-R approach are covered by Schiffner and $\frac{1}{2}$ $\frac{1}{1}$ to database $\frac{1}{1}$ of user re Scheuermann [1979], Lusk et al. [1980], Casanova [1984], and Wang [1984].

 $\begin{bmatrix} \mathbf{r} \\ \mathbf{r} \end{bmatrix}$ proposed. These include CABLE (Shoshani 1978), GERM (Benneworth et Notes Various data manipulation languages for the E-R model have been al. 1981), and GORDAS [ElMasri and Wiederhold 1983]. A graphical query a piùcan query
I Mendelzon

relationship data model was introduced by Smith and Smith [1977]. Lenzerini and Santucci [1983] I
and used these concepts in defining cardinality constraints in the E-P model used these concepts in defining cardinality constraints in the E-R model.
Basic textbook discussions are effected by Tai basic in the E-R model. egation were

Basic textbook discussions are offered by Tsichritzis and Lochovsky 1982 and by Chen $[1983]$.

model have been

'u'l'j

|
| 1::0
| 1:1
| 1:1 ec,
"F" From a historical perspective, the relational data model is relatively new. The first database systems were based on either the hierarchical model (see 3 than is the relational model.

 ,ugh tables are a simple, intuitiv concept of a relation.

In the years following the introduction of the relational model, a substantial theory has developed for relational databases. This theory assists in the design of relational databases and in the efficient processing of user requests for information from the database. We shall study this theory in Chapter 6, after we have introduced all the major data models.

$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ denoting the definitional Database as a collection of the database as a collection of the database of the database as a collection of the database of the database of the database of the dat

 enterprise. They differ slightly fromin Chapter ² in. order to simplify our

 $\frac{d\mathbf{y}}{d\mathbf{y}}$

this chapter, we shall be using ^a number of different relations to

A relational database consists of a collection of tables, each of which is assigned a unique name. Each table has a structure similar to that
presented in Chapter 2, where we represented E-R databases by tables. A row in a table represents a relationship among a set of values. Since a table is a collection of such relationships, there is a close correspondence between the concept of table and the mathematical concept of relation, from For information from the database informational data model takes its naturely interest in the data model takes its naturely interest in the matter of relation. introduce the concept of relation.
In this chapter, we shall be using a number of different relations to

illustrate the various concepts underlying the relational data model. These relations represent part of a banking enterprise. They differ slightly from the tables that were used in Chapter 2 in order to simplify our a unique presentation. We shall discuss appropriate rel detail in Chapter 6.

detail in Chapter 6.
Consider the *deposit* table of Figure 3.1. It name, account-number, customer-name, balance. For each attribute, there is a set of permitted values, called the domain of that attribute. For the

 \mathbf{L}

branch—name, for example, the domain

(1) where Ul is a branch where Ul is a branch where U

attribute

balances.As

possible

account-numbers,

(that is, ¹¹³

Mathematicians

 ^I refer tot[branch-name]t[branch-name|of ^t on

 mathematicians relythe attribute whose domain \mathbf{z}

attribute whose domain

 Theai t

essentially

Chapter 3

the

 \rightarrow

(branch-name

Customer-scheme

way

 that the attribute customer-name appears incoincidence. Rather, the use of common c

Deposit-scheme to

Structure of Relational Databases 47

Figure 3.1 The deposit relation.

attribute *branch-name*, for example, the domain would be the set of all notation $\frac{1}{2}$, $\frac{1}{2}$ branch names. Let D_1 denote the the set of all $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 &$ s in domain $\begin{array}{c} \bullet \\ \bullet \end{array}$ s a customer name (that is, v_3 is in domain D_3), and v_4 is a balance (that is, v_4 is in domain D_d). In general, deposit will contain only a subset of the set of all $\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$ possible rows. Therefore deposit is a subset of:

\times D_i \mathbb{R} and let \mathbb{R} be a set and let \mathbb{R} set and let \mathbb{R} set and let \mathbb{R} set of all \mathbb{R} set of all \mathbb{R} set and let

In general, a table of n columns must be a subs

appears first in the list of domains, ²

relations, we shall use the mathematical terms relation and tuple

 $\frac{1}{\sqrt{2}}$ the deposit relation of $\frac{1}{\sqrt{2}}$ such the tuples. Let the tuples $\frac{1}{\sqrt{2}}$

 \mathbf{x} , \mathbf{D}_i is an account number (that is, $\sum_{i=1}^k D_i$

 is in domain D3), and ^v' is ^a balance (that is, ²¹⁴ is in Mathematicians define a *relation* to be a subset of a cartesian product of
a list of domains. This corresponds almost exactly with our definition of
table. The only difference is that we have assigned names to attributes, **EXECUTE:** Mathematicians define a relation to be a subset $ger 1 to$ f domains, $\begin{array}{lll} \text{d} & \text{d} & \text{d} & \text{d} \\ \text{d} & \text{d} & \text{d} & \text{d} \\ \text{d} & \text{d} & \text{d} & \text{d} \end{array}$ **Example Example Example Because** tables are essentially relations, we shall use the mathematical terms relation and tuple in place of the terms table and row.

In the deposit relation of Figure 3.1, there are seven tuples. Let the tuple variable t refer to the first tuple of the relation. We use the notation *t*[branch-name] to denote the value of t on the branch-name attribute. Thus, $t[branch\text{-}name] = \text{``Downtown''}.$ Similarly, $t[account\text{-}number]$ denote \mathbf{v} value of \mathbf{t} on the *eccount-number* attribute, etc. \sqrt{r} write $\frac{d\mathbf{r}}{dt}$ difference is that we have assigned names to attribute state \mathcal{A}

place of the terms table and radiation of the terms table and raw. This is not a contribute to the terms of the

t[1] to denote the value of tuple t on the first attribute (branch-name), t[2] for account-number, etc. Since a relation is a set of tuples, we use the mathematical notation of $t \in r$ to denote that tuple t is in relation r .

When we talk about a database, we must differentiate between the database schema, that is, the logical design of the database, and a database instance, which is the data in the database at a given instant in time.

The concept of a relation scheme corresponds to the programming $\frac{3}{2}$ ianguage notion of type demitton. A variation relation.

It is convenient to give a name to a relation scheme, just as we give account-number, etc. Since the ames to type definitions in programming convention of using lowercase names for relations and names beginning with an uppercase letter for relation schemes. Following this notation, we the set of all $\frac{d}{dt}$ use *Deposit-scheme* to denote the relation scheme. data in

Deposit-scheme = (branch-name, account-numbe

 \mathbb{R} In general, a relation scheme is a list of attribu is, v_4 is in $\begin{bmatrix} 1 \end{bmatrix}$ domains. We denote the fact that *deposit* is a relation.

deposit (Deposit-scheme) $\frac{1}{2}$ in the definition of $\frac{1}{2}$ in programming the adopt the adopt the adopt the adopt the adopt the adopt the set of $\frac{1}{2}$

We shall not, in general, be concerned about the precise definition of the domain of each attribute until we discuss file systems in Chapter 7. However, when we do wish to define our domains, we use the notation

(branch-name: string, account-number: integer, customer-name: string, balance: integer)

general, a relation scheme is a relation scheme is a list of attribute the fact that deposits is a relation of ϵ scheme for that relat

Customer-scheme = (customer-name, street, customer-city)

Note that the attribute customer-name appears in both relation schemes. This is not a coincidence. Rather, the use of common attributes in relation schemes is one way of relating tuples of distinct relations. For example, customer-namesuppose we wish to find the cities where depositors of the Perryridge branch live. We would look first at the *deposit* relation to find all depositors of the Perryridge branch. Then, for each such customer, we look in the denotes the the customer relation to find the city he or she live
e may write

Chapter 3

⁼

 \rightarrow

. I

have

Downtown

E 1

are not using

Customer-scheme

Borrow-scheme

superkey

For the purpose of this chapter, we assume that the relation schemes for our banking enterprise are as follows:

Branch-scheme = $(b$ ranch-name, assets, branch-city) $Customer\text{-}scheme = (customer\text{-}name, street, customer\text{-}city)$ Deposit-scheme = (branch-name, account-number, customer-name, balance) Borrow-scheme = (branch-name, loan-number, customer-name, amount)

xiels and a structure of Sayethey (*makiname* is the hotel and the structure of Sayethey (*makiname*) and (*bernch*-

The notion of a structure of Sayethey (*bench-mane*) and (*bernch-mane*) and (*bernch-mane*) and (*bernc* **purpose of the Chapter 2, is applicable also to the relation** Branch-scheme, {branch-name} and {branch-name, branch-city} are both superkeys. {branch-name, branch-city} is not a candidate key because ${branch-name} \subseteq {branch-name}$, branch-city) and ${branch-name}$ itself is a superkey. (branch-name), however, is a candidate key, which for our purpose will also serve as a primary key. The attribute branch-city is not a superkey since two branches in the same city may have different names (and different asset figures). The primary key for the customer-scheme is customer-name. We are not using the social-security number, as was done in \int Chapter 2, in order to have smalle example of a bank database. We expect that in a real world database the social-security attribute would serve as a primary

 {branch-name}Let R be a relation scheme. If we say that ${\bf j}$ is not ${\bf R}$, we are restricting consideration to rela (a) distinct tuples have the same values on all ${t_1}$ and ${t_2}$ are in r and ${t_1 \neq t_2}$, then ${t_1[K] \neq t_2[K]}$.

serve as ^a primary key. The attribute branch-city is not ^a

Smith 2000 and 2000

 Jackson \mathbf{F} and \mathbf{F} and \mathbf{F} and \mathbf{F} round the state of Hill Turner and the control of the control Williams and the contract of t Town Adams and the second contract of the second contract of the second contract of the second contract of the Johnson

Hayes

4

entity—relationship

 $\overline{}$ account $\overline{}$

 \mathbf{C} for each

values for street and

7L
2

when

 andinformation

the values for the street and

shall assume the relation

tuples, we must use null values. Thus, in the α

Customer-scheme and one on Deposit—scheme, we can represen^t

becomes available. In Chapter 6, we shall study criteria to help us

provided

incomplete

customers

decide

w

"は数字" 小手をおいます

Harrison (1986) (1986) (1986) Figure 3.2 The customer relation. ry and the state of the state of
The state of the st Pittsfield Turner

of the entity-relationship model, we would say that the attribute $name$ represents the same er have already

It would appear that, for our banking example, we could have just one relation scheme rather than several. That is, it may be easier for a user to think in terms of one relation scheme rather than several. Suppose we
used only one relation for our example, with scheme used only one relation for or applicable

$Account-info-scheme = (branch-name, account-number, customer-name,$,.-.-.'r-name,

Observe that if a customer has several accounts, we must list her or his **Franch-name** [*loan-number* | customer address once for each account. That is, we must repeat certain information appear that, for our banking example, in our banking example, the several times. This repetition is wasteful and was avoided by our use of $\frac{3}{2}$ [Downtown 17] Jone
two relations. If a customer has one or more accounts two relations. If a customer has one or more accounts, but has not the provided an address, we cannot construct a tuple on Account-info-scheme, $\frac{d}{dx}$ since the values for the street and customer-city are not known. To repres the values for *street* and *customer-city* must be null. By using two relations, ne on Customer-scheme and one on Deposit-scheme, we can represent $\frac{3}{12}$ North To customers whose address is unknown, without using null values. We $\frac{1}{100}$ is that if a contract the must be information becomes available. In Chapter 6, we shall study criteria to help \mathbb{R} us decide when one set of relation schemes is better than another. For the pay of the state of the pay of the state of the pay of the state of t now, we shall assume the relation schemes are given. an address, we cannot construct ^a tuple on Account-info-sdreme, simply use a tuple on Deposit-scheme to represent the information about the Perryridge

address is unknown, with the information of the information about the information about the information about the information of the information about the information about the information about the information of the info

sun-tam

FST-IBM000846

Section 3.2

"branch-name

client relation

o'custarrrer'rmrnc

"employee-name" for all tuples of all tu

i.

t. It seems reduced to the seems reduced

relation

3.2 Formal Query Languages

A query language is a language in which a user requests information from the database. These languages are typically higher-level languages than standard programming languages. Query languages can be categorized as being either procedural or nonprocedural. In a procedural language, the user instructs the system to perform a sequence of operations on the database to compute the desired result. In a nonprocedural language, the user
describes the information desired without giving a specific procedure for
obtaining that information.
 $\sigma_{amount} > 1200$ (borrow) describes the information desired without giving a specific procedure

that includes elements of both the procedural and the nonprocedural chapter. First, we look at two "pure" languages: one procedural and one database. These "pure" languages lac commercial languages, but they illustrate the fundamental techniques for extracting data from the database. approaches. We shall study several commercial languages later in this

等
おいつのは、
このことには、The Relational Algebra

standard

instructsthe control of the control

 $-\alpha$ \mathbb{R}^n

Most

approaches.

commercial

fundamental fundamental dengan ke

 \mathbf{z}

chapter.

includes in the second second in the second second in the second in the second in the second in the second in
in the second in the secon

nonprocedural.

 \mathbb{R}^n

We

data from

produce a control of the second s

other These

Fundamentalselect and

on

lowercase Greek

subscript to

 \blacksquare

relations andselect operation

^a new

 \mathbb{F}_2 . We are the set of \mathbb{F}_2

The relational algebra is a procedural query language. There are five commental operations in the relational algebra systems of the system and super a system of $\frac{1}{2}$ select, project, cartesian-product, union, and set-difference. All of these shall studyoperations produce a new relation as their result.

In addition to the five fundamental operations, we shall introduce several other operations, namely, set intersection, theta join, natural join, and
division. These operations, will be defined in terms of the fundamental division. These operations will be defined in terms of the fundamental operations.

Fundamental operations

letter sigma (a) to denote selection. The predicate

(borrow). Figure

0'. The argumen^t relation is ^given in parentheses

relational algebra is a project operations are called a
contract of the property operation in the other three relations. perate on one relation. The other three re relations and are, therefore, called binary operations. s, since they

The select operation selects tuples that satisfy a given predicate. We use The *select* operation selects tuples that satisf

the lowercase Greek letter sigma (or the five

annexes as a subscript for π . The argument solary appears as a subscript to σ . The argument relation is given in parenthe

Figure 3.4 Result of $\sigma_{branch\cdot name} =$ $\frac{1}{2}$ settlement predicate. We use that satisfy a given predicate.

And the company's second the company's and the company's company's company's company's company's company's company's

following the σ . Thus, to select those tuples of the borrow relation where the branch is "Perryridge," we write

$\sigma_{branch\text{-}name} = \text{``Pervridge''}(borrow)$

If the borrow relation is as shown in Figure 3.3, then the relation that results from the above query is as shown in Figure 3.4. We may find all

the oriental the oriental, we allow comparisons using selection predicate. Furthermore, several predicates may be combined into a larger predicate using the connectives and (\wedge) and or (\vee) . Thus, to find those tuples pertaining Perryridge branch, we write relation Is as shown in Figure 3.3, then the relation that

 $\sigma_{branch-name}$ = "Perryridge" \land amount > 1200 (borrow)

The selection predicate may include comparisons between two attributes. To illustrate this, we

$Client-scheme = (customer-name, employee-name)$

predice the indicating that the employee is the "personal be combined into the connection of the connection of relation *client* (Client-scheme) is shown in Figure 3.5. We may find all those customers who have the same name as their personal banker by writing

$\sigma_{\text{customer-name}} = \text{employee-name}$ (client)

If the client relation is as given in Figure 3.5, the answer is the relation shown in Figure 3.6.
In the above example, we obtained a relation (Figure 3.6) on $(custom$

Client-schemename, employee-name) in which tuples t. It seems redundant to list the person's name twice. We would

Figure 3.5 The client relation.

list the person's name twice. We would .

one attribute relation

t

subscript we wantto

revisit the

that instead

from

domains). From

that results from

we want to be

 \mathbf{a}

et. any si

resultSuppose asa
2

> \mathbf{L} us

The

information

operations

 \mathbf{N} .

H.

dties in

relation

 relation^r ⁼

. figure

 $\overline{}$

Section 3.2

prefer a one attribute relation on (customer-name) which lists all those who have the same name as their personal banker. The project operation allows us to produce this relation. The project operation is a unary operation that copies its argument relation, with certain columns left out. Since a relation is a set, any duplicate rows are eliminated. Projection is denoted by the Greek letter pi (II). We list those attributes that we wish to appear in the result as a subscript to Π . The argument relation follows Π in parentheses.

Suppose we want a relation showing customers and the branches from which they borrow, but do not care about the amount of the loan, nor the loan number. We may write $\frac{3}{5}$ ioan number. We may write

$\Pi _{branch\text{-}name\text{-}}$ customer-name (borrow)

name as their personal banker." We write

Notice that instead of giving the name of a relation as the argument of the projection operation, we give an expression that evaluates to a relation.

The operations we have discussed up to this point allow us to extract time. We have information from only one relation at at time. We have not yet been able to the prince of the second information from several relations. One operation that allows us and the second of the second of the second of the second operation is a binary operation. We shall use infix notation for binary $\begin{bmatrix} \text{Hayes} \\ \text{Hayes} \end{bmatrix}$ [Ones $\begin{bmatrix} \text{Genn} \\ \text{Genn} \end{bmatrix}$ Sand Hill $\begin{bmatrix} \text{Vayes} \\ \text{Sayes} \end{bmatrix}$ $\frac{1}{2}$: coperations and, thus, write the cartesian product of relations r_1 and r_2 as $\frac{1}{2}$ lines and r_3 lines in $\frac{1}{2}$ lines $r_1 \times r_2$. We saw the definition of cartesian product earlier in this chapter (recall that a relation is defined to be a subset of a cartesian product of a \int ohnson \int ohnson \int ohnson \int Smith \int North R $\sum_{k=1}^{\infty}$ intuition about the definition of the relational algebra operation x .
 $\sum_{k=1}^{\infty}$ and $\sum_{k=1}^{\infty}$ is a relation of choosing the attribute names for the $\sum_{k=1}^{\infty}$ i indexpanded to a relation. P \mathbb{R}^n relation that results from a cartesi

Suppose we want to find all clients of bank employee Johnson, as well as the cities in which the clients live. We need the information in both the dient relation and the customer relation in order to do so. Figure 3.7 shows the relation $r = client \times customer$. The relation scheme for r is

 $\sum_{i=1}^{55}$ choosing the attribute 1.6 Result of $\sigma_{\text{out-term error}}$ σ_{out} $\sigma_{\text{out-term error}}$ (client). a Cartesian Pigure 3.6 Result of $\sigma_{\text{customer-name}} = \epsilon$ mployee-name (client).

 \mathcal{N}^* , and the set of \mathcal{N} , and \mathcal{N} , \mathcal{N} , \mathcal{N} , and \mathcal{N} , and \mathcal{N} , \mathcal{N} , \mathcal{N}

which the clients live. We need the information in both the j fin j is $1'$ fin j if j is $1'$

and.the customer relation in order to do so. Figure 3.7 shows Johnson Iohnson Arlmag ['an like ⁱ ' ^t the

Johnson

lohnson and the same of the sa

client ^x customer. The relation scheme for ^I ¹⁵ Johnson Glenn Sand Hill

Figure 3.7 Result of client \times customer.

Green Walnut Stamford 'j ⁱ §=

Formal Query Languages 53

'

(clientrustomtrmnme,

customerslrcet,

name of the relation

That is, we have

appear in the control of th

are natu

Returning

 3.7. Assume

Now

 \overline{a}

relation,n t for some tuples ^t in

in

relation

containsrum

pertaining

see

cartesian al II-lea e al I

of, the

 $\overline{}$

Pertain to Johnson.

ge^t only

address

 $\frac{1}{2}$

of client, we know

 that client.customer-name"clientmstomer-mm:

embar 1980.
Personality of the state of the

Johnson

JohnsonJohnson

Johnson

Johnson

ncilentxustamer-name.("clientxustamer-Mme(adienl.employee-name

 =- "Johnson" (clientmstnmer)»

Section 3.2

(client.customer-name, client.employee-name, customer.customer-name, customer.street, customer.customer-city)

That is, we simply list all the attributes of both relations, and attach the name of the relation from which the attribute originally came. We need to attach the relation name to distinguish client.customer-name from customer.customer-name.

 \mathbb{R}^n tuples appear in r ? As you may have suspected, we construct a tuple from the customer relation. Thus r is a large relation, as can be seen from Figure 3.7.

Assume we have n_1 tuples in client and n_2 tuples in customer. Then there are n_1n_2 ways of choosing a pair of tuples: one tuple from each relation, so there are n_1n_2 tuples in r. In particular, note that it may be the from which the attribute original came.
 $\frac{d\mathbf{r}}{dt}$ case for some tuples t in r that i[client.customer-name] $rac{2\pi}{160}$ name.

In general, if we have relations $r_1(R_1)$ and $r_2(R_2)$, then $r_1 \times r_2$ is a relation whose scheme is the concatenation of R_1 and R_2 . Relation R contains all tuples t for which there is a tuplet relation that $t(n)$ which $t[R_1] = t_1[R_1]$ and $t[R_2] = t_2[R_2]$.
Returning to the query "Find all clients of Johnson and the city in "
"
.

which they live," we consider the relation $r = c$

 $\sigma_{client.employee-name}$ = "1 ohnson " (clie .t'h-'grurflll

then the result relation is as shown in Figure $\frac{1}{2}$ column may contain customers of employees other don't see why, look at the definition of cartesian product again). Note that $\frac{d}{dx}$ the *client customer-name* column contains only only the cartesian product operation associates every tuple of customer with every tuple of client, we know that some tuple in client \times customer has the $\frac{1}{200}$ address of the employee's customer. This occur .»a...

 σ client customer-name = customer.customer-name

 $\frac{1}{\sqrt{2\pi}} \left(\frac{\sigma_{\text{client number}}}{\sigma_{\text{client number}}}\right)^{1/2}$ $\sum_{n=1}^{\infty}$ to examployee-name = "Johnson".

 $\frac{1}{200}$ we get only those tuples of client \times customer than σ ⁻⁻⁻⁻⁻⁻) \rightarrow cartesian product a cartesian product and

 client.custonrer'narne column**• Pertain to Johnson.**

the cases where it has \mathbf{r}

("client" and the matter that was a state of the state of collection of collection

Have the street and city of the customer of Johnson.

p product of customer with every tuple of customer of $\frac{1}{2}$ **customer of** $\frac{1}{2}$ **customer** $\frac{1}{2}$

Curry Prigure 3.8 Result of $\sigma_{client.employee-name}$:

Hayes Main Harrison

Finally, since we want only customer-name **Adams Spring Pittsfield Johnson Spring Pittsfield Johnson Spring Pittsfield Johnson Spring Pittsfield Johnson** Johnson Alma Palo Alto

 $\Pi_{client.customer-name,\ customer.customer-city}$

 $\sigma_{client_customer-name} = c$ usto σ

(o client employee-name = "Johnson"

 $R = \frac{1}{2}$ action $\frac{1}{2}$ action $\frac{1}{2}$ (client $\frac{1}{2}$ customer).

advertising department: "Find all customers

branch:

loan at the Perryridge branch:

Section 3.2

but do

all relational algebra expressions.

2. The domains of the *i*th attribute of r and the *i*th attribute of s must be the same.

The set-difference operator, denoted by $-$, allows us to find tuples that are in one relation, but not in another. The expression $r - s$ results in a
relation containing but not in another. The expression $r - s$ results in a relation containing those tuples in'_r but not in s.

We can find all customers of the Perryridge branch who have an account there but do not have a loan there by writing:

 $\Pi_{\text{customer-name}}$ ($\sigma_{\text{branch-name}} = \text{``Peryridge''}$ (deposit)) $\Pi_{\text{customer-name}}$ ($\sigma_{\text{branch-name}} = \text{``Peryridge''}$ (deposit))

the intervalsation in the intervalsation of the ith attribute 3.3) and the 2. metasta in the ith attribute of r and the ith attribute of the ith attribute of the ith attribute $\frac{1}{2}$. The result relation for this query

 σ . So the expression the advertising department needs in our example is σ is to **find the relational algebra** σ

 $\Pi_{\text{customer-name}}$ ($\sigma_{\text{branch-name}}$ = "Penyndge" (lower-value) and σ_{error}) are five operators we have just seen allow us to The five operators we have just seen allow us to give a complete definition of an expression in the relational algebra. A basic expression in the relational algebra consists of either one of the following:

 $^{\bullet}$ A relation in the database.

customer-name and two depositors. This is due to the fact that Hayes is

both a borrower and a depositor of the Perryridge branch. Since relations

are sets, duplicate values are eliminated.

A general expression in the re in either or both of the two relations. This is accomplished by the two relations. This is accomplished by the
This is accomplished by The result relation of the result relation of the result relation for the result relat
 CUSSETVE that, in our example, we took the union of two sets, both of subexpressions. Let E_1 and E_2 be relational algebra expressions. Then, unions are taken between compatible relation of the relation of the relati

-
- relation. The former is a relation of four attributes and the latter of three. $\bullet E_1 E_2$ relation. The former is a relation of four attributes and the latter of three.

Furthermore, consider a union of a set of customer names and a set of
	-

 \overline{a}

- the database.
	- $\Pi_S(E_1)$, where S is a list consisting of some of the attributes appearing in E_1

are all relational algebra expressions.

FST-IBM000850

 ^aunion

customer-name α four attributes and the latter of of a set of customer names and a set of α set of $\$ $\frac{1}{\sqrt{2}}$ contracts the contract of $\frac{1}{\sqrt{2}}$ contracts of a union would not most situation would not most situations. Therefore, I 2012 is situated with the situation of \mathbf{V}

Contains of the contact of

accomplished by the binary operation union, denoted, as in set theory, by U. So the expression the advertising department needs in our example is

query, we need the information in the borrow relation (Figure 3.3) and the deposit relation (Figure 3.1). We know how to find all customers with a

 $\Pi_{\text{customer-number}}(\sigma_{\text{branch-name}} = \text{``Perrying''}(\text{borrow}))$

We know also how to find all customers with an account at the Perryridge

The result relation for this query appears in Figure 3.9. Notice that

make sense to take the union of the borrow relation and the customer

for a union operation $r \cup s$ to be legal, we require that two conditions hold:

there are three tuples in the result even though the Perryridge branch has

 $\left| \right|$

 $\mathbf{S} = \mathbf{S}$, the same arity. That is, they must instead instead inserting appearing at the attributes appearing appe

 $U \prod_{\text{customer-name}} (\sigma_{\text{branch-name}} = \text{``Perrying'''} (\text{deposit'})$

Chapter 3

^x custom» Observe that this particular theta join forces equality on those attributes that

algebra. Let us

nborrawuslmnrr—mme.

in

the

selection

mrtesian

We have now seen the five fundamental operations of the relational algebra: σ , Π , x , U , $-$. These five operators are sufficient to express any relational algebra query. However, if we restrict ourselves to just the five fundamental operators, some common queries are lengthy to express. Therefore, we define additional operators. These new operators do not add

any power to the algebra, but they do simplify common queries.
For each new operator we define, we give an equivalent expression
using only the five fundamental operators.
The first additional relational algebra operation

seen the five fundamental operations of the relations of the relation

 $\Pi_{\text{customer-range}}$ ($\sigma_{\text{branch-range}}$ = "Perr queries are lengthy to express. Therefore. define additional operators. These new operators do not add wanna.

The result relation for this query appears in Figure 3.11. .-

 $\frac{1000}{1000}$ Note, however, that we do not include set intersection as a only the fundamental operation. We do not include set intersection a
selection of fundamental operation. We do not do so because we see that $f(x)$ relational algebra expression using set intersection by replacing the relational algebra expression using set intersection by replacing the $\left| \right|$ is the suppose to find a suppose when $\frac{1}{2}$ intersection operation with a pair of set different intersection operation with a pair of set difference operations as follows:
Greek. Using set in the Perryridge branch. Using set in the Perryridge branch. Using set in the Perryridge bra
Greek. Using set in the Perryridge \mathbb{R} operatorforms

Thus, set intersection does not add any power to the relational algebra. It
is simply more convenient to write r O s, than $r = (r - \epsilon)$ is simply more convenient to write $r \cap s$ than $r - (r - s)$.
The next operations we add to the algebra are used to simplify many

queries that require a cartesian product. Typically, a query that involves a
cartesian product includes a selection operation on the result of the cartesian product includes a selection opera operation. We $\frac{1}{2}$ cartesian product. Consider the query "Find all customers who have a loan relation at the Perryridge branch and the cities in which they live." We first form $\frac{320}{27}$ at the Perryriage branch and the cities in which the cartesian product of the *borrow* and *custom* those tuples that pertain to Perryridge and pertain to only one customer-
name. Thus we write

 $\frac{d}{d\theta}$ does not a not a subsequent to the relation of $\frac{1}{d\theta}$ is more convenient to write I n $\frac{1}{d\theta}$. ^a predicate 0,

 $\frac{d\mathbf{y}}{dt}$ is the next operation simply when \mathbf{y} \mathbf{t} are used to simplify many \mathbf{t} matrices.

the contract of the contract of

Additional

 $\frac{d\mathbf{r}}{d\mathbf{r}}$

 $-\epsilon_{\rm eff}$.

result relation

relational algebra

fundamental

Cartesian

 Cartesianname. tuples Thus

however, that

Perryridge branch

wethatwrite

"humanism-name.

All ALL The Anti-Anti-Anti-

fundamental sebagai pertama kemasa kemasa pertama kemasa kemasa pertama kemasa kemasa kemasa kemasa kemasa kem
Kemasa kemasa kemas

Theintersection

algebra

we

o, n, x, U, -. These

 $P = borrow. branch-name = "Pergrid"$ \land borrow.customer-name = customer.customer-name Section 3.2

customer-name **Hayes**

Figure 3.11 Customers with an account and a loan at the Perryridge branch.

For each new operator we define, we give an equivalent expression

In the theta join is a binary operation that allows us to combine the

section and cartesian product into one operation. The theta join is

intersection (

 $\Pi_{\text{borrow.customer-name, customer.customer-city}}$ (borrow \bowtie_{Θ} customer)

In this example, Θ is the predicate:

is the "formal symbol and the subscript 8 (the subscript 8 (the subscript 8 (the selection predicate).

The subscript 8 (the subscript 8 (the subscript 8 (the theory is replaced by the selection predicate). \wedge borrow.customer-name = customer.customer-name

 \mathbf{I} the predicate \mathbf{I} **reduce our relational algebra expression for the superior of the "Find all customers"** \mathbf{G} and \mathbf{a} and \math

$r \Join_{\theta} s = \sigma_{\theta} (r \times s)$

The natural-join operation n is a furthe borrow.borrow.borrow.borrow.com
Perry respectively.

 $\Pi_{\emph{borrow.customer-name, customer-cuty}}$

bommustomn—nnme ⁼ atstamer.cusiom-nnme customer)

both relation schemes. This sort of predicate occurs

 $(borrow \Join borrow.customer-name = customer.customer-name \textit{ Customer})$

Observe that this particular theta join forces equality on those attributes that appear in both relation schemes. This sort of predicate occurs frequently in practice. Indeed, if we are printing out pairs of (customer-name, customer-city), we would normally want the city to be the city in which customer lives, and not some arbitr all customers have not some arbitral and the some at the some arbitral dependence of q

Pittstield in der Statten kas – Alto<mark>stilia</mark>andi

Johnson

Although

 Well I

Since

customer-name. The customer and the custom

union di Ja

customers and the second s written in 1994

attributes

e consideration in the set of the

 \mathbf{H}

3.12

Customer-scheme)

the schemes for borrow

ncusttmrer-rmme.

those attributes in the second contributes in the second contributes in the second contributes in the second c

that approximate \mathbf{z}

are now in the set

relation schemes ^R

consider the

here, about

two

with

("customer-city

 Thatis, the

branch.

"customer-name

("branch-name ⁼

"Pcn'yridge" ("W9")

that we could

specify

Formal Query Languages -61

Although the definition of natural-join is a bit complicated, it is applied easily. We can use the natural join to write the query "Find all customers having a loan at some branch and their cities" as follows:

 $\Pi_{\text{customer-name}, \text{ customer-city}}$ (borrow \bowtie customer)

Since the schemes for borrow and customer (that is, Borrow-scheme and Customer-scheme) have the attribute customer-name in common, the natural-
3. Solution considers only pairs of tuples that have the same value on
3. Solution-name attribute-name only when necessary to a single tuple on
2. C $\frac{R}{2}$ section considers only pairs of tuples that have the same value ':the union of the two schemes (that is, branch-name, loan-number, customer-
name, amount, street, customer-city). After performing the projection, we state the natural join is central to much of relational database theory t_{max} and t_{max} a bit complicated, it is a bit complished easily.
The extinct of the definition is a bit complicated easily. The extinct example is applied to the relation shown in Figure 3.12. The extinct example $\frac{1}{2}$ is the natural joint to write the natural join to write the customer $\frac{1}{2}$ and $\frac{1}{2}$ all customers having a series having $\frac{1}{2}$ and $\frac{1}{2}$ and $\frac{1}{2}$ and $\frac{1}{2}$ and $\frac{1}{2}$ and $\frac{1}{2}$ a customers having a loan at the Perryridge branch and their cities," can be written as

ntustamn-Mme,custmner~dty ϵ

 σ_{branch} , extense σ_{system} is, σ_{b} (borrow- \bowtie customer) \sim bri where \mathbf{w}

 $\frac{1}{2}$ We are now ready for a formal definition of the natural join. Consider $\frac{1}{2}$ two relation schemes R and S which are, of course, lists of attribute names. Let us consider the schemes to be sets rather than lists. This allows us to and the natural includes the performance of the natural intervalses. This allows us to
a contract performance the projection is contract to much of relations of relationships the natural database of relationships and the the relation shown in Figure 3.12. The early \mathbf{r}_i and \mathbf{r}_i all the earlier examples of its use \mathbf{r}_i all \mathbf talking, here, about union and intersection on sets of attributes, not relations.

denoted by $r \bowtie s$ is a relation on scheme $R \cup S$. It is the projection onto Consider two relations $r(R)$ and $s(S)$. The natural join of r and s.

|
Figure 3.12 Result "customer—name $R \cup S$ of a theta join where the predicate requires $r.A = s.A$ for each attribute A in $R \cap S$. Formally,

$$
r \bowtie s = \Pi_{R \cup S} (r \bowtie_{r, A_1} = s.A_1 \land \dots \land r.A_n = s.A_n s)
$$

where $R \cap S = \{A_1, ..., A_n\}.$

Section 3.2

Now that we have introduced the natural join, we adopt the following .
. prefix.

and practice, we give several examples of its use:

 \bullet Find the assets and name of all l $\frac{A}{A}$ is,

 $\Pi_{branch\text{-}name}$, assets

 $f(\sigma_{\text{customer-city}} = \text{``Pon Chester''}(customer \bowtie \text{deposit} \bowtie \text{branch}))$

1. Consider
attribute mames.
The results of a shall drop to a modern necessary to a Notice that we wrote customer *to* denoting

Figure and \mathbf{s} , and $\$

Perryridge branch.

 that we wrote customer ⁰⁶ $\prod_{\text{customer-range}} (\sigma_{branch_name} = \neg_{Pervoids})$

> Note that we could have intersection: ection:

> external branch (1990)

not specific the specific term intended because the specific theorem in the specific theorem in the specific theorem in the specific theorem intended because the specific theorem intended because the specific theorem in t

This example illustrates a general fact about the relational algebra: It is possible to write several equivalent relational algebra expressions that \mathcal{L} (borrow N \mathcal{L} and \mathcal{L} and \mathcal{L} and \mathcal{L}

 that information. are two

we ^give ^a formal definition

the last section. Find

to some of the queries for which we wrote relational algebra

the branch-name, loan-number, customer-name, and amount for loans

Ĩ

 now The $d\cdot$ suppose the control of the control o branches in 1970 located expression: can l find all customer-name, branch-mmoperation. TheFormally, let relationon scheme ^R

> with ^s ^g ^R

see that this is true, observe that "R

right side of the set difference operator,

the second condition of the definition of division. The expression on

 _ ⁵ (r) ^x ^s . This is ^a relation on scheme ^R which pairs every tuple in \sim S (r) with every tuple in 5.5 Thus (let

 \bullet Let $r(R)$ and $s(S)$ be relations without any attributes in common, that is, $R \cap S = \emptyset$. (\emptyset denotes the empty set.) Then $r \bowtie s = r \times s$.

We now introduce one final relational algebra operation, called division $(+)$. The division operation is suited to queries that include the phrase "for all." Suppose we wish to find all customers who have an account at all branches located in Brooklyn. We can obtain all branches in Brooklyn by the expression:

 $r_1 = \prod_{branch-name} (\sigma_{branch-city} = \text{``Brooklyn''}(branch))$

We can find all customer-name, branch-name pairs for which the customer has an account at the branch by writing $\frac{1}{2}$ \overline{r} . \overline{r} of tuples from

interoduce on final relationships on the final relationship of the final relationship of the called division of the called divisio

Now we need to find customers who appear in r_2 with every branc to the first of this customers who appear in r_2 with every branch han
in r_1 . The operation that provides exactly those customers is the divide operation. The query can be answered by writing

 $\Pi_{\text{customer-range, branch-name}}$ (deposit)
 $\div \Pi_{\text{branch-range}}$ ($\sigma_{\text{branch-city}} = \text{``Brooklyn''}}$ (branch))

pairs for the customer the customer sections, and let $S \subseteq R$. The relation $r \div s$ s there is a tuple t_n in r satisfying both of the following: y—e-KH'W
Persian
Any designation is a relation on scheme $R - S$. A tuple t is in $r + s$ if for every tuple t, in 3.5e, 3.53

 $t_r[S] = t_r[S]$ $|I_n(R - S)| = |I(R - S)|$

It may be surprising to discover that the division operation can, in fact, be be surprising to discover that the division operation can, in fact, be
in terms of the five fundamental operations. Let $r(R)$ and $s(S)$ be
sith $S \subset R$ given, with $S \subseteq R$. answer to ^adescription

 $r - s = 11_R - s$ (r) - $11_R - s$ (line)

To see that this is true, observe that Π_{R-S} (*r*) gives us all tuples *t* that \vdots represent to the set of the set o $\frac{dS}{dS}$ satisfy the second condition of the definition of division. The expression on e right side
.

 $\Pi_{R} = \varsigma \left(\left(\Pi_{R} = \varsigma \left(r \right) \times s \right) = r \right)$

serves to eliminate those tuples that fail to satisfy the first condition of the definition of division. Let us see how it does this. Consider $\overline{\Pi}_{R-S}(\vec{r})$ with every tuple in s. Thus $(\Pi_{R-S}(\vec{r}) \times s) - r$ gives us those ery tuple is

pairs of tuples from Π_{R-1} \in (r) and s which do not appear in r. If a tuple t is in

 $\Pi_{R} = g((\Pi_{R} = g(r) \times s) - r)$

then there is some tuple t_c in s that does not combine with tuple t to form a tuple in r. Thus t holds a value for attributes $R - S$ which does not appear in $r + s$. It is these values that we eliminate from $\Pi_{R} = g(r)$.

At times it is necessary to express the cartesian product of a relation \overline{a} relation, we must rename one of the operands of the cartesian product. For this purpose, we define a rename operator which allows us to refer to a relation by more than one name. Define rename($R1$, $R2$) to be a function which returns the relation specified by R1, but under the name R2. See exercises 3.5 e, 3.5 g and 3

3.2.2 The Relational Calculus

Section 3.2

 α e. For a value of the relational algebra is a procedural language. s^2 is a procedular and s^2 relational algebra expression, we provide it is necessarygenerates the answer to our query. The relational calculus, on the other hand, is a nonprocedural language. In the relational calculus, we give a $\frac{m}{s}$ is a nonprocedural language. In the re obtain that information. alle muomiauon uesiicu
1.

There are two forms of the relational calculus, one in which the variables represent tuples, and one in which the variables represent values of domains. These variants are called the tuple relational calculus and the domain relational calculus. The two forms are very similar. As a result, we shall emphasize the tuple relational calculus.

 $\frac{1}{2}$ A query in the tuple relational calculus is expression, when $\frac{1}{2}$

 $\{t \mid P(t)\}$

 $\frac{1}{\sqrt{2}}$ that is, the set of all tuples t such that predice $\frac{1}{4}$ our earlier notation, we use $f[A]$ to denote the A, and we use $t \in r$ to denote that tuple t is in relation r .
Before we give a formal definition of the tuple relational calculus, we

before the give a tornal demandon of the
turn to some of the queries for which v
toressions in the last section. expressions in the last section.

Find the branch-name, loan-number, customer-name, and amount for loans of over \$1200: $\frac{1}{2}$ or over $\frac{1}{2}$. Expressed as $\frac{1}{2}$. Calculus is expressed as $\frac{1}{2}$

Suppose we want only the customer-name attribute, rather than all attributes in a set of the *borrow* relation. To write this query in the value of t

U \ ϵ and the first condition of the first condition of the definition of the of division. Let us see how

need

to

"편법

 \cdot

exists ^a tuple ^s in

^s (s ^e borrow

branch, we used

 \mathcal{L}_{c} , μ is a function

 $\overline{}$

 by "or" (v). t. $\overline{3}$

the tuple relational mlculus, we shall need two "there exists" clauses, connected

 $\overline{\mathcal{C}}$. The same of $\overline{\mathcal{C}}$ is the set of $\overline{\mathcal{C}}$ is the set of $\overline{\mathcal{C}}$

 \mathbb{R}^d

branch

 \mathbf{F} i I \mathbf{I}

A

at the

Using

there is a second

Perryridge

borrower

Perryridge

customer.

Consider

 complexer i

have two start in the start in t
have two start in the start in t

have ω i

calculus in

predicate

above expression

appears

^s (s ^e borrow

Section 3.2

the Perryridge'brunch.

The above expression gives us the set of all customer-name tuples such that at least one of the following holds:

- · The customer-name appears in some tuple of the borrow relation as a borrower from the Perryridge branch.
- · The customer-name appears in some tuple of the deposit relation as a depositor of the Perryridge branch.

If some customer has both a loan and an account at the Perryridge branch, $\begin{array}{c}\n\hline\n\end{array}\n\text{definition of a set does not allow duplicate mer-
\n
$$
\begin{array}{c}\n\text{definition of a set does not allow duplicate mer-
\nif we now want only those customers that\n\end{array}
$$$

loan at the Perryridge branch, all we need to do is change the "or" (y) to $\langle \wedge \rangle$ in the above expression.

 $\{t \mid \exists s \ (s \in borrow \land t[customer-name] = s[customer-name] \}$ Perryridge branch. \wedge s[branch-name] = "Perryridge") $\wedge \exists u$ (u e deposit \wedge t[customer-name] = u[customer-name]

Now consider the query, "Find all customers who have an account at \sim some consider the query, \sim the Perryridge branch but do not The tuple relational calculus expression for this query is similar to those we ame] and the relations calculus expression for this can
ame]

> {*t* \exists *u* (*u* \in *deposit* \wedge *i*[*customer-name*] = *u*[*customer-name*]
 \wedge *u*[*branch-name*] = "Perryridge") $\wedge \neg \exists s$ (s \in borror "customer-name] ⁼ u[customer-name] \mathbb{Z}

The above tuple relational calculus expression uses the $\exists u$ (...) clause to require that the customer have an account at the Perryridge branch, and it but do not have a loan from the explored the $\neg \exists s$ (...) clause to eliminate those tuple relations and the use of the borrow relation as having

> the division operation, "Find all customerbranches located in Brooklyn

We need those tuples on (customer-name) such that there is a tuple in borrow pertaining to that customer-name with the amount attribute > 1200. In order to express this, we need the construct "there exists" from the predicate calculus in mathematical logic. The notation

 $\exists t (Q(t))$

means "there exists a tuple t such that predicate $Q(t)$ is true."

Using this notation, we may write the query "Find all customers who have a loan for an amount greater than \$1200" as: 3.2 ' Formal Query languages 655 ' Formal Query languages 655 ' \sim

 $\{t \mid \exists s \ (s \in borrow \land t[customer-name] = s[customer-name] \land s[amount] > 1200)\}$ $\frac{1}{2}$ and $\frac{1}{2}$ and $\frac{1}{2}$ are with the amount $\frac{1}{2}$ and $\frac{1}{2}$

EXPLEMENT
EXPLEMENT In English, we read the above expression as "the set of all tuples t such that there exists a tuple s in relation borrow for which the values of t and s ie customer-name attribute are equal, and the value of s for the amount $\frac{1}{2}$ and" $\frac{1}{2}$ \mathbf{I} the following \mathbf{I} from

exists a tuple t such a tuple t such a tuple t such a tuple to the such a tuple to Consider the query "Find all customers having a loan. $\frac{1}{2}$. Perrynage branch and the cities in which they live." This query is slightly more complex than we have seen so far since it involves two relations, namely, customer and borrow. But as we shall see, all this requires is that we have two "there exists" clauses in our tuple relational calculus $\frac{1}{2}$
expression. We write the query as follows: xpression. We write .
...
|} of the

 $\{t \mid \exists s \ (s \in borrow \land t[customer-name] = s[customer-name] \land s[branch-name] = "Pervridge" \}$ relation borrow \wedge s[branch-name] = " \mathbb{R} customer at the set of \mathbb{R} $\Lambda \exists u (u \in \textit{customer} \land u | \textit{customer} \textit{-num} \land u | \textit{customer} \textit{-num} \land u | \textit{customer} \textit{-num} \land u | \textit{course} \}$ 'u
C
C

In English, this is "the set of all *(customer-name, customer-city*) turiles for and the city in the city of the city of the city in the city is slightly which relations that the city is slightly than we have seen so that the city is slightly than α . the city of customer-name." Tuple variable s ensures that the customer is a borrower at the Perryridge branch. Tuple variable u is restricted to pertain $\frac{1}{25}$ contomer at the refryinge branch. Tuple vanable u is restricted to per
Now to the same customer as s, and u ensures that the customer-city is the cit .
".w.c.c. sc.h $\mathbf{1}$

To find all customers having a loan, an account, or both at the Perryridge branch, we used the union operation in the relational $\frac{1}{2}$ under the customer $\frac{1}{2}$ the tuple relational calculus, we shall need two "there exists" connected by "or" (\vee) .

 $\{t \in \exists s \ (s \in borrow \land t[customer-name] = s[customer-name] \land s[branch-name] = "Peryridge")$ \overline{z} and customer is a borror $\sqrt{3} u (u \in \text{depsit})$ the of customer-name." Tuple $\sqrt{3} u (u \in \text{depsit})$ \land u[branch-name] = "Perryridge"}}

the same customer as s, and it ensures that the customer customers that the customer-city is the city of the city find all customers having ^a loan, an account, or both at the $\text{calculus, we introduce the ``for all'' }\text{con}$ above tuple relational alculus expression uses the ³ ^u (...) clause to that the customer have an 3 s(\sim) clause to eliminate those customers who appear in the customers who appear in the customers who appear in

> thedivision

tuple of the burrow

consider the constant of the c

operation, "Find all customers who have an account at all

means of follows:"Q

tuples

The branch-name

A

V". Thus variablein: is said

of the following

^s ^e r, where ^s is ^a tuple variable and ^r is ^a relation.

 \mathbf{G} where s and u are tuple variables, \mathbf{x}

^y have domains whose members can be compare^d by 8.

5 is defined to define the contract of the unit of an attribute on \mathcal{L}

c, where s is a tuple variable, x is an attribute on which 5 is an attribute on which 5 is an attribute on which 5 is an

attribute6 isx. ^a comparison operator, and ^c is ^a constant in the domain

 $\overline{}$, $\overline{}$

Weare now

- Ž

which

defined,of the state

comparison

following

are built up

formulae.

P2 is equivalent to

means "Q is true for all tuples t ." We write the expression for our query as follows:

> $\{t: \forall u (u \in branch \vee u[branch-city] \neq "Brooklyn"$ \vee 3 s (s ϵ deposit \wedge t[customer-name] = s[customer-name] \land u[branch-name] = s[branch-name])}

In English, we interpret the above expression as "the set of all (customername) tuples t such that for all (branch-name, branch-city) tuples at least one $\frac{3}{2}$ is the contract of $\frac{3}{2}$ is $\frac{3}{2}$ if $\frac{3}{2}$ is $\frac{3}{2}$ if $\frac{3}{2}$ of the following is true:

- is true for all tuples t." We write the expression for our query as Formulaeot pertain \mathbf{I} to a branch in Brooklyn).
	- The value of u on attribute branch-city is not Brooklyn.
	- 3 s (s e de posit A en ministère de positionnel de positionnel de positionnel de la court at the $\frac{1}{2}$ formula. then0

 $\begin{CD} \begin{bmatrix} \mathbf{w} \ \mathbf{w} \end{bmatrix} \end{CD}$. We are now reduce to above \mathbf{w} the set $\frac{551}{20}$ is the calculus A tuple relational calculus at formal definition of $\frac{1}{20}$ calculus A tuple relational calculus aureoration is calculus. A tuple relational calculus expression is of the form:

 $u(x)$
where P is a formula. Several tuple variables may appear in a formula. A ^a branch in Brooklyn). The variable is said to be a *free variable* unless it is quantified by a "3" or \mathbb{R} customeration has anotacount H. at the branch whose name appears in the branch whose name by a "3" or interventional in appearance. In the tuple relationship of the tuple relationship of the tuple tupl
There are infinitely relationship of the tuple tup calculus, these

 $t \in \text{borrow} \land \exists s \ (t[customer-name] = s[customer-name])$

 \overline{z} \overline{z} to a free variable. Tuple variable s is said to be a bound variable. tuple variable variable is send to be a bound variable.
A tuple relational calculus formula is built up out of atoms. An atom $:$ or the $\frac{1}{2}$ is one final issue we must address in $\{t\}$

is external tuple variable and r is a re $\mathbf{S} \in \mathbf{r}$, where s is a tuple variable and r is a re

- \bullet s[x] Θ u[y], where s and u are tuple vari and \bf{y} have domains whose members can be compared by $\bf{\Theta}$. which s is defined, y is an attribute on which u is defined, and Θ is a are is a ji
^{i hutos v} butes *x*
- $\frac{1}{\sqrt{2}}$ α is a tuple value of α be a comparison open $\sum_{i=1}^{\infty}$ formula is built up of atom is built up of atom is of atom is of atom is of atom is $\sum_{i=1}^{\infty}$ of atom is $\sum_{i=1}^{\infty}$ d_{max}

Formulae are built up from atoms using the following rules:

· An atom is a formula.

• If P_1 is a formula, then so are $\neg P_1$ and (P_1) .

• If P_1 and P_2 are formulae, then so are $P_1 \vee P_2$ and $P_1 \wedge P_2$.

• If $P_1(s)$ is a formula containing a free tuple variable s, then:

$$
\exists s (P_1(s)) \text{ and } \forall s (P_1(s))
$$

are also formulae.

and α are attributes and c is a constant. It is possible that the only tuples and tuples only tuples and $S_{\rm 3}$:39 c are tuples whose values do not appear in the database. such ^a tuple requires ^a search among the potentially infinitenumber of tuples that doTo assist us in defining ^a restriction of the tuple relational alculus, we introduce

As was the case for the relational algebra, it is possible to write equivalent expressions that are not identical in appearance. In the tuple
relational calculus, these equivalences include two rules:

Pars in the formula end of $P_1 \wedge P_2$ is equivalent to $\neg \neg P_1$

2. \forall $f(P_1(t))$ is equivalent to $\ni \exists$ $f(\neg P_1(t))$.

There is one final calculus. A tuple relational
calculus A tuple relational
Suppose we wrote the expression:
 Suppose we wrote the expression:

$f(t)$ is the relation of the relationships $f(t)$ is possible to $f(t)$ $\{t \mid t \notin borrow\}$

upies contain v
lo not wish to do not wish to allow such expressions. Another type of expression we wish to disallow is:

 $\{t \mid \exists s (s[x] \neq c \land t[y] = s[y])\}$

tuple where x and y are attributes and c is a constant. tabase. Findin
inita mumburu

infinite number of tuples that do not appear in the database.
To assist us in defining a restriction of the tuple relational calculus, we introduce the concept of the *domain* of a tuple relational calculus formula. values that definite that do not even appear in the domain of a tuple

1. Let P be a formula. Intuitively, the domain of P

1. Which c is a set of all values referenced in P. These include values are domain of P is the sum appear in one or more of To assist us in defining a restriction of the tuple relational calculus, we

 m" . FST-IBM000855

These

dom(P).

dam(Pl)such

 Forevery

 Forevery

> true \mathbb{Z} if and

The

a Katila I

true, we need

 $\frac{d}{dt}$

 $\frac{1}{2}$ know

of this section \mathbf{r}

calculus

 \mathbf{H}

is ^a second

 calculus. Inthat take

formation of the second district of the second district of the second district of the second district of the s
of the second district of the second district of the second district of the second district of the second distr

to the tuple relational calculus.

the

The

every

equivalentl I

expression

expression.

possible

appear in

 expressions. Wehold:

These considerations motivate the concept of safe tuple relational calculus expressions. We say an expression $\{t | P(t)\}$ is safe if all of the following hold:

- 1. All values that appear in tuples of the expression are values from $dom(P)$.
- 2. For every "there exists" subformula of the form $\exists s (P_1(s))$, the subformula is true if and only if there is a tuple s with values from $dom(P_1)$ such that $P_1(s)$ is true.
- true if and only if $P_1(s)$ is true for all tuples s with values from $dom(P_1)$.

values from \vert values from
for all" and $\frac{1}{200}$ with a relation are values of the following from the following formulae without having to test infinitely many possibilities.

 t consider the second rule in the definition of safety. For $\exists s(P_1(s))$ to be true, we need to find only one s for which $R(s)$ is true. be true, we need to find only one s for which $P_1(s)$ is true. In general there would be infinitely many tuples to test. However, if the expression was been that the line from $dom(P_1)$. This reduces the number of tuples we must consider to a finite number. The situation for subformulae similar. To assert that $\nabla s(P_1(s))$ is true, we must, in general, test all possible tuples. This requires us to examine infinitely many tuples. As $p(\mathbf{x})$ subset if we know the expression is safe, it is subset in \mathbf{f} for those tuning subject and the subset in \mathbf{f} \mathbf{t}

All the tuple relational calculus expressions we have written in the examples of this section are acts. $\frac{1}{2}$ examples of this section are safe. is ^a formula.

be infinitely many tuples to this section are sale.
1. The tuple relational calculus, restricted to safe expressions, is
1. An atomic equivalent in expressive power to the relational abodius TP is Equivalent in expressive power to the relational algebra. This means that
for every relational algebra expression, there is an equivalent cafe $\frac{d\theta}{d\theta}$ every relational algebra expression, there is an equivalent constant consider to a must consider the number of the number o number: The situation Top School and Top School an
The situation Top School and Top Sc pression in the tuple relational calculus, and for every safe tuple
elational calculus expression there is an equivalent relational algebra $\overline{}$ for: \blacksquare expression. We will not prove this fact here, but the bibliographic notes contain references to the proof. Some parts of the equivalent in expressive power to the relational algebra. This means that

exercises.

There is a second form of the relational calculus called the d

relational calculus In this form of the relational calculus called the d tuple relational calculus called the *domain*
tuple relational calculus called the *domain*
stational variables that take on unluse formulational calculus, we use *domain* variables that take on values from an attribute's domain, rather than values for an entire tuple. The domain relational calculus, however, is
related to the tuple relational calculus, however, is related to the tuple relational calculus.

of the relatjonal calculus called the domain

relational algebra expression, there is an equivalent safe

in the tuple relational calculus. and for every safe tuple relational

will not prove this fact here, but the bibliographic notes contains \mathcal{L} exercises.references to the proof. Some parts of the proo^f are included in the

entire tuple. The domain relational calculus. however, is closely related

An expression in the domain relational calculus is of the form $\{x_1, x_2, ..., x_n\}$ $\{P(x_1, x_2, ..., x_n)\}$ where the $x_i, 1 \le i \le n$, represent domain variables. P represents a formula. As was the case for the tuple relational calculus, a formula is composed of atoms. An atom in the domain relational calculus is of the following forms:

- $\frac{3}{2}$ from $\frac{3}{2}$ $\frac{2}{3}$ $\frac{2}{$
- 3. For every "for all" subformula of the form $\forall s (P_1(s))$, the subformula is operator $\langle \langle s, s \rangle = 1$, $\neq 0$, $\langle s \rangle$ and $\langle s \rangle = 0$ is thus for all tumber such turbus $\langle s \rangle = 0$. Consists that can be compared by θ .
	- $\bullet x \Theta c$, where x is a domain variable. Θ is a represent values from $\begin{array}{c}$ is a constant in the domain of the attribution all $\begin{array}{c}$ is a constant in the domain of the attribution variable.

Formulae are built up from atoms using the following rules:

 \bullet An atom is a formula.

Section 3.2

- in the variables and \bullet if P_1 is a formula, then so are $\neg P_1$ and (P_1) .
sider to a
	- If P_1 and P_2 are formulae, then so are $P_1 \vee P_2$ and $P_1 \wedge P_2$.
- x_1, y_2, y_3 is a formula in x, where x is a domain is a comparison operator, and $\mathbf{F}_1(x)$ is a formula in x, where x is a domain

 $\exists x (P_1(x))$ and $\forall x (P_1(x))$

are also formulae.

As a notational shorthand, we write

$$
\exists a,b,c (P(a,b,c))
$$

$\exists a\; (\exists b\; (\exists c\; (P(a,b,c))))$

The notion of safety applies to the domain calculus as well. The domain relational calculus, restricted to safe expressions, is equivalent to
the tuple relational calculus, restricted to safe expressions. Since we noted earlier that the tuple relational calculus, restricted to safe expressions, is

Chapter 3

equivalent to the relational algebra, all three of the following are equivalent:

• The relational algebra.

equivalent:equivalent to

š

Ř

Find

Find

branch:Find

Final Section

ā

representing"user-friendly"

The relational algebra.

we now the similar $\frac{1}{2}$

Find of \mathbb{R}

in which they live they live

 Commercial Query Languages fomial languages we have just seen

languages: SQL, Quel, and

t and

· The tuple relational calculus restricted to safe expressions.

• The domain relational calculus restricted to safe expressions.

We now give domain relational calculus queries we considered earlier.
Note the similarity of these expressions with the corresponding tuple relational calculus expressions:

Find the branch name, loan number, customer name, and amount for loans of over \$1200:

 ${**b,l,c,a>1 **b,l,c,a> \epsilon** borrow \wedge a > 1200}**$

 $\frac{1}{200}$ Find all customers who have a loan for an amount greater than \$1200. view.

 $\{ \langle \langle \rangle \rangle \mid \exists b, l, a \langle \langle \rangle b, l, c, a \rangle \in \text{borrow} \land a \rangle \leq 1200 \}$

 $\lim_{x \to 0}$ do $\lim_{x \to 0}$ calculus $\lim_{x \to 0}$. $\begin{bmatrix} \mathbf{c} \\ \mathbf{c} \end{bmatrix}$ and all customers having a loan from the corresponding tuple relations which they live. city in which they live:

$$
\{1 \exists b,l,a (d,l,c,a > \epsilon \text{ borrow } \wedge \text{ }b = \text{``Peryridge''}
$$

$$
\wedge \exists y (\epsilon \text{ customer})\}
$$

 $Find$ all customare having ϵ α branch:

 $\{Cc > 1 \exists b, l, a \ (cb, l, c, d) \}$

 \mathbb{R} all customers who have an account Brooklyn:

$$
\{ < c > 1 \ \forall \ x, y, z \ (\lt x, y, z > \notin branch \ \lor \ z \neq \text{ "Brooklyn" } \ \lor \ (\exists \ a, n \ (\lt x, a, c, n > \epsilon \ \text{ deposit})))\}
$$

all customers having a loan, and account, or both at the Perryn

 $I_n = 1$ be a set of the set of $\sum_{i=1}^n$ behavior that I_n and I_n (the formal languages we have just a
Representing queries. However, datab representing queries. However, database system products require a more "user-friendly" query language. In this section, we study three of these product languages: SQL, Quel, and QBE. We have chosen these languages

man and the second company of the second second

because they represent a variety of styles. QBE is based on the domain
relational calculus; Quel is based on the tunk relational calculation relational calculus; Quel is based on the domain
relational calculus; Quel is based on the domain
uses a combination of relational algebra and relational calculus; and SQL uses a combination of relational dictional calculus; and SQL
All three of these languages have been influential calculus constructs. All three of these languages have been influential not only in research
database systems but also in commercially marketed customs. database systems but also in commercially marketed systems.
Although we neer to these 1.

Although we refer to these languages as "query languages," it is a
hally incorrect. SQL, Quel, and OBE on their manus of languages," this is actually incorrect. SQL, Quel, and QBE contain many other capabilities
besides querying a database. These include features from the capabilities besides querying a database. These include features for defining the structure of the data, features for modifying data in the database, and features for specifying security constraints. We shall defer discussion of these ecifying security

Our goal is not to provide a complete users' guide for these languages.

Nur goal is not to provide a complete users' guide for these languages. Rather, we present the fundamental constructs and concepts of these
languages. Individual implementations of these languages of these
diagonalism and concepts of these these det

3.3.1 SQL

 minimizetypical SQL query has the form:

select

 \sim

SQL was introduced as the query language for System R. SQL is an acconym for Structured Query Language. It is still referred to finally in the set acronym for Structured Query Language for System R. SQL is an
acronym for Structured Query Language. It is still referred to frequently by
its former name, Sequel.

-
- The basic structure of an SQL expression consists of three clauses:
et, from, and where. select, from, and where.
- The select clause corresponds to the projection operation of the relational algebra. It is used to list the attribution $\frac{d}{dx}$ is also the set of the relational algebra. It is used to list the projection operation of the
relational algebra. It is used to list the attributes desired in the result of
a query.
- The from clause is a list of relations to be scanned in the execution of
the expression.
- The where clause corresponds to the selection predicate of the relational algebra. It consists of a predicate involving studies rive vince clause corresponds to the selection predicate of the
relational algebra. It consists of a predicate involving attributes of the
relations that appear in the from clause. relations that appear in the from clause.

The different meaning of the term "select" in SQL and the relational
algebra is an unfortunate historical fact we smark in the relational algebra is an unfortunate historical fact. We emphasize the different
interpretations here to minimize potential confusion interpretations here to minimize potential confusion.
A typical SOI query has the fact of the solential confusion. A typical SQL query has the form:

$$
\begin{array}{l}\n\text{select } A_1, A_2, \dots, A_n \\
\text{from } r_1, r_2, \dots, r_m \\
\text{where } P\n\end{array}
$$

The A_{β} represent attributes, the r_{β} represent relations, and P is a predicate. This query is equivalent to the relational sky. predicate. This query is equivalent to the relational algebra expression:

the

attributes

predicate,

practice,

effidency

branches in 1980 ii.

The

relations. InTherefore,

as

implementationsSQL

> ussee

operateon

operations

Perryridge

duplicates infor every

dause,

 $\overline{\mathbf{S}}$

 \mathbb{Z} all

 \mathcal{A} and \mathcal{A}

performance and the energy of the energy

 \mathbf{I}

to Chapter 9.

result of an

 \mathbf{I}

insert the keyword the key

U, n, and

branch: branch select

SQL
Sql

of all relations appearing

Chapter 3

(selectunion alla suomalainen kunnan ole tunnet ole tunnet ole tunnet ole tunnet tunnet tunnet tunnet tunnet tunnet tu

(select

(select deposit where the con-

(selectfromwhere

 \mathbf{I}

where

 \mathbf{I}

 $\mathbf{L} = \mathbf{L}$

from the company of the company

Mandi-name ⁼

 \mathbf{P}

"Perryridge")

branch, we write

loan there, we write

from

 ζ

Next, let us find all customers having a loan from the Perryridge branch:

select customer-name from borrow where $branch$ - $name = "Perrying"$

To find all customers having a loan, an account, or both at the Perryridge branch we write

(select customer-name from deposit all customers having ^a loan fromwhere $branch____\$ "Perryridge") union (select customer-name from borrow where $branch$ -name = "Perryridge")

> Similarly, to find all customers who have both a loan and an account at the Perryridge branch, we write

> > (select customer-name from deposit where $branch\text{-}name = "Peryridge")$
intersect (select customer-name from borrow where $branch\text{-}name = "Perryingedge")$

To find all customers of the Perryridge branch who have an account there

(select customer-name from *deposit*
where *branch-name* = "Perryridge") minus (select customer-name from borrow
where branch-name = "Perryridge")

SQL does not have a direct representation of the natural-join operation. However, since the natural join is defined in terms of a matter to wi

 $\Pi_{A_1, A_2, ..., A_n}(\sigma_P(r_1 \times r_2 \times \cdots \times r_m))$

If the where clause is omitted, the predicate P is true. The list $A_1, A_2, ..., A_n$ of attributes may be replaced with a star (*) to select all attributes of all relations appearing in the from clause.

SQL forms the cartesian product of the relations named in the from clause, performs a relational algebra selection using the where clause predicate, and projects the result onto the attributes of the select clause. In S_{c} form that can

The result of an SQL query is, of course, a relation. Let us consider a $\frac{1}{2}$ branches in the *deposit* relation":

be replaced with ^a star (') to select all

from deposit

In the formal query languages, the mathe being a set was used. Thus, duplicate tupl relations. In practice, duplicate elimination is relatively time-consuming.
Therefore, SQL (and most other commercial query languages) allow duplicates in relations. The above query will, thus, list each branch-name once for every tuple in which it appears in the *deposit* relation.
In those cases where we want to force the elimination of duplicates,

 the deposit relation": ve insert the ke query as deposit

select distinct branch-name from deposit

a set was used. Thus, duplicate tuples did not ever appear in \mathbb{R}^n if we want duplicates removed. We note for historical accuracy that early implementations of SOL used the keyword unique in place of distinct.

EQL includes the operations union, in operate on relations, and correspond directly to the relational algebra operations $U_1 \cap A_2$ and $-$.

Let us see how the example queries that we considered earlier are written in SQL. First, we find all customers having an account at the rerrynage branch:

select customer-name from deposit $\frac{d}{dx}$ where $branch-name = "Per$

stration in the company of the state of

includes the operations union, intersect, and minus, which which we have \mathcal{N}

in SQL. First, we find all customers having an account at the

relations, and correspon^d directly to the relational algebra

FST-IBM000858

Recall

義王

 thequery and the set

Notice

relational

in the from

require

letus

mathematical $\overline{\mathbf{S}}$

aocount

Perryrldge

borrowers

in

that we

SQL, we write

that \mathbf{S}

customats and the customats

respective

select

where

for membership

Earlier, we

branch. We can

it leads

absence of set membership.

us to

holders in 1916.
Heimilia e Santo Sa

holders in der ste

 \mathbf{r}_{\perp}

but

in

 \mathbf{E}

take the alternative approac^h of finding all account

the Perry.
The Perry ridge branch and the Perry ridge branch. Clearly, this is an equivalent approach, the person of the

where you are not the in connective of SQL . We also the interest of SQL.

 connective tests for set membership, where the set is ^a collectionof values produced by ^a select dause. The not in connective tests

algebra, to

 \mathbf{t} at tribute customer-city custo

consider ^a

resulting

 \mathcal{L} , \mathcal{L}

from

borrow Indian State Communication

deposit) \

selectfrom

 \mathbf{I}

Recall that we wrote the relational algebra expression:

 $\Pi_{\text{customer-name}, \text{ customer-city}}$ (borrow \bowtie customer)

for the query "Find all customers having a loan at some branch and their city." In SQL, we write

select customer.customer-name, customer-city
from borrow, customer

Notice that SQL uses the notation relation-name.attribute-name, as did the naGlefl-Mme.in the scheme of more than one relation. We could have written
customer.customer-city instead of customer-city in the select clause Wommer customer.customer-city instead of customer-city in the select clause. However, since the attribute customer-city appears in only one of the relations named in the from clause, there is no ambiguity when we write customer-c

Let us consider a somewhat more complicated query in which we
require that the customers have a loan from the Perryridge branch: "Find $f(x) = f(x) - f(x)$ and $f(x) = f(x) - f(x)$ barrow.cuslomer-name a manner and the renamental media to state
their respective city." In order to state this query, we shall need to state
two constraints in the where clause, connected by "and." two constraints in the where clause, connected by "and."

uses the notation relationships relationships the notation relationships in cases where an attribute and the n

and the notation of the second states where an attribute appears in cases where a second states where an attri from borrow, customer where $borrow.customer-name = customer.customer-name and$ $branch$ -name = "Perryridge"

somewhat more complicated when we want we write complicate connectives "and," "or," t^2 and t^2 mathematical symbols γ , γ and γ .

names of all customers have soft a loan at the Perryridge branch and the Perryridge branch a tuples for membership in a relation. To illustrate this, reconsider the query "Find all customers who have both a loan and an account at the Perryndge branch." Earlier, we took the approach of intersecting two sets: the set of account holders at the Perryridge branch and the set of borrowers from the $\begin{bmatrix} \frac{1}{2} \\ \frac{1}{2} \end{bmatrix}$ Ferryridge branch. account holders at the Perryridge bra borrowers from the Perryridge branch. Clearly, this is an equ $\frac{1}{2}$ SQL.

The in connective tests for set membership, where the set is an equivalent

connective of \cdot to \cdot

collection of values produced by a select clause. The not in correct that the absence of set membership the collection of values produced by a select clause.

for the absence of set membership.

Let us use in to write the query "Find all customers who have both a loan and account at the the query ring all customers who have both a
account holders, and write the cubourement." We begin by finding all account holders, and write the subquery:

> (select customer-name from deposit where $branch$ -name = "Perryridge")

 $\sum_{n=1}^{\infty}$ We then need to find those customers who where borrow.customer-name = customer.customer-name
Notice that SQL uses the notation relation-name.attribute-name, as did the in the above subquery. We do this by embedding the above subquery in an
relational algebra, to and account at the resulting query is
appears of the branch. The resulting query is

select customer-name extends
where $\frac{1}{2}$ customer-name in (select customer-name fram deposit where branch-name = "Perryridge")

 $\frac{1}{2}$ These last two examples show that it is p $\frac{1}{2}$ and $\frac{1}{2}$ above subsquery. Solvernly we do the $\frac{1}{2}$ of $\frac{1}{2}$ and $\frac{1}{2}$ are different that $\frac{1}{2}$ and $\frac{1}{2}$ and $\frac{1}{2}$ and $\frac{1}{2}$ query several ways in SQL. This is beneficial
about the query in the way that appears me about the query in the way that appears most natural. We shall see that
there is a substantial amount of redundancy in SOI. Customer-name

In the above example, we tested membership in a one-attribute
tion. It is possible to test for membership in a one-attribute from-"borrow $\begin{array}{ccc} \text{uses the notation} < v_1, v_2 \\ \text{values } v_1, v_2, \dots, v_n \text{ lies} \end{array}$ rom deposits to the first set \mathbf{r}_1 hird way:

> select customer-name from borrow where branch-name = "Perryridge" and <branch-name, customer-name> in (select branch-name, customer-name
from deposit)

 to denote ^a tuple of arity ⁿ containing values ..., on. Using this notation, we can write the query "Find all custome ^h ^h ' customers who have an account at the Perryridge branch but do not have a loan at the Perryridge branch, we can write select

from deposit where

nga
Sal

A

EU D

of tuple variables by

were able to

 \mathbf{E}

testing

Brooklyn."

FST-IBM000860

where the con-

 $\alpha \sim 1$ customer-name \mathcal{R}_{G} and \mathcal{R}_{G}

 for equality between two branch names. Consider the query "Findall branches that have greater assets than some branch located in

calculus.com

relation.

Chapter 3

select T.branch-name from branch T , branch S where T *assets* $>$ *S assets* and $S.branch-city = "Brooklyn"$

Since the comparison is a "greater than" comparison, we cannot write this expression using the in construct.

SQL does, however, offer an alternative style for writing the above query. The phrase "greater than some" is represented in SQL by > any. see greater than some" is represented in SQ
allows us to rewrite the query in a form the
nulation of the query in English weekly our formulation of the query in English.

select branch-name
from branch where $assets$ > any
(select $assels$ Since the comparison and comparison a greater than comparison, where $bran ch - \frac{1}{2}$ where $bran ch - \frac{1}{2}$ where $bran ch - \frac{1}{2}$

 $\begin{bmatrix} 1 & 1 \end{bmatrix}$ the subquery subquery is a sense of \mathcal{S}

IS

select f where $\overline{}$ assets

Brooklyn.

for branches in

 rewrite the query in ^a form(select assets from branch

comparison in the where clause of the outer select is true if the assets value where S.customer-name = "Jones" and
S.branch-name = T.branch-name
of the tuple is greater than at least one member of the set of where $\frac{1}{2}$ ya branch - . - _ Note

In the same relation of the same relation. Suppose which reference to deposit is intended.
We note that an alternative way to express this query is that the same of the same of the some parater assets than all branches in $\frac{3}{2}$ have gn using the "> all" construct:
select branch-name

(select assets
from branch

The constructs in, $>$ any, $>$ all, etc. allow us to test a single value against members of an entire set. Since a select generates a set of tuples, we may, at times, want to compare sets to determine if one set contains all

select customer-name from deposit where $branch\text{-}name = "Pervridge"$ and customer-name not in (select customer-name from borrow where $branch\text{-}name = "Perrying"$

SQL borrows the notion of tuple variables from the tuple relational R are α and α and α is α and α is α summer a particular method in the system classe. We illustrate use of tuple variables by rewriting the query "Find all customers having a loan at some bank and their city":

> select *LC*
(select *LC*
cubero 5.4 branch-name = "Personner" + manner

 \mathbb{R} the name of the relation it is associated with.
Tuple unrighten are most unright under tuple -r
Iu

Tuple variables are most useful when we need to compare two the associated with-
Superior state and particular the angle of the state of the sta $\frac{d^2}{dx^2}$ in the same relation. Suppose we want to find all customers who have an any $\frac{d}{dx}$ and $\frac{d}{dx}$ any. The subquery. $\frac{f(x)}{f(x)}$ account at some bank at which Jones has an account. We write this query as follows: as follows: 7mm...tr
...tr
...tr $\frac{1}{2}$ for $\frac{1}{2}$ f

select brunch—name of the selection select T.customer-name

from deposit 5, deposit T

where S.customer-name = "lones" and

where S.customer-name = "lones" and

comparison in the where clause of the cubes actual attaction in the where clause of the cubes act T.custamer-city of the customer-city of the customer-city of the customer-
The customer-city of the customer-city of the customer-city of the customers of the customers of the customers select T.customer-name
from deposit S, deposit T
where S.customer-name = "Jones" and $S. branch-name = T. branch-name$

 $\frac{1}{\sqrt{2}}$ Observe that we consider the from Clause that we could not use the notation *deposit branch-name* since it

Figure 1 when the dear which reference to *deposit* is intended.

Sall also allows "< any," " = any," " = any," and " + would not be clear which reference to deposit is intended.

 $\frac{1}{2}$ follows: $\frac{1}{2}$ select customer-name $\frac{1}{2}$ from deposit $\frac{1}{2}$ from deposit $\frac{1}{2}$ from $\frac{1$ where the control of trom *deposit*

where *branch-name* in the select *branch-name* in the select *branch-name* select *branch-name* where σ is the contract of $\frac{1}{2}$ is the contract of $\frac{1}{2}$ is the contract of σ f (select branch-name f) and f is the set of f is (select branch-name
from deposit $\frac{1}{2}$ and $\frac{1}{2}$ where assets $>$ all where assets $>$ all $\sum_{i=1}^{n}$ from deposit
The outer select IS true if the outer select IS true is the assets value of the assets value is the assets value of the assets value of the assets value in the assets value of the assets value of - where customer-name $=$ "Jones") $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

,values

Brooklyn." We can write the SQL expression: all" construct: ble to use the in construct in the above query becau
clearabity between two branch names. Consider the **allows the sum of the sum of the showe query because we**
where the tranch-city = "Brooklyn")
"Find all branches that have greater assets than some branch located in." The constructs in, > any, > all, etc. allow us to tes "Find all branches that have greater assets than some branch located in greater assets than the second term in the second term in the second term in the second term in the second ter

minimum: min [~]

the members of some other set. Such comparisons are made in SQL using the contains and not contains constructs.

Consider the query "Find all customers who have an account at all branches located in Brooklyn." For each customer, we need to see if the set of all branches at which that customer has an account contains the set of all branches in Brooklyn.

> select customer-name
from deposit S where (select branch-name from deposit T
where S.customer-name = T.customer-name) contains
(select branch-name from *branch*
where *branch-city* = "Brooklyn")

The subquery

:10.
1980: The order in the order in
1980: The order in the order

 \mathcal{A} and tuples the order by dause causes the tuples in the tup

a loan at the Perryridge branch, we can write be calculated by the Perryridge branch, we can write be can write

 $\mathcal{T}_{\mathcal{B}}^{(n)}$ in all habetic $\mathcal{B}^{(n)}$ in all habetic $\mathcal{B}^{(n)}$ er a customers order. The customers of $\mathcal{B}^{(n)}$

members of some other set. Such as \mathbb{R}^n

of all branches locatedatinwhich Brooklyn."

 contains andConsider

branches

branches **branches and the series**

subquery

el
19

 \mathbf{z}

Manges?

relation

 \sim

at which

having

branches in

 4'»-... (select branch-name from branch where $branch\text{-}city = "Brooklyn")$

 $finds$ all tl

(select branch-name from deposit T where \dot{S} -customer-name = T.customer-name)

finds all the branch-Thus, the outer select takes each customer and tests whether the set of all branches at which that cust \cdot tuples.

SQL offers the user some control over the order in which tup of a query to appear in sorted order. To list in alphabetic order all casioniers naving a ioan at the Perryridge branch, we can write $\frac{1}{2}$ is in a second to $\frac{1}{2}$ use the having

> select customer-name from borrow where $branch$ -name = "Perryridge" order by customer-name

Section 3.3

Commercial Query Languages - 79

In order to fulfill an order by request, SQL must perform a sort. Since the view of the same of the by request, SQL must perform a sort. Since
sorting a large number of tuples may be costly, it is desirable to sort only
when necessary. when necessary.

SQL offers the ability to compute functions of groups of tuples using
group by clause. The attribute given in the groups of tuples using be groups the directions of groups of tuples using
the group by clause. The attribute given in the group by clause is used to
form groups. Tuples with the same unlus ex this can be a form groups. Tuples with the same value on this attribute are placed in
one groups. Tuples with the same value on this attribute are placed in one group. SQL includes functions to compute:

 $\frac{3}{3}$ $\frac{3}{2}$ $\frac{4}{2}$ $\frac{4}{2}$ $\frac{5}{2}$ $\frac{1}{2}$ $\frac{2}{3}$ $\frac{1}{2}$ $\frac{3}{2}$ $\frac{1}{2}$ $\frac{1}{2}$

o maximum: max
v total: sum must perform \blacksquare solidi: sum
 \blacksquare sorting some sorting some sorting some solidi: sum

 \bullet total: sum

in the having clause. We express this query in SQL

 $\frac{d}{dx}$

 ω relation. The notation used for this in SQL is in SQL i is count ('). Thus, to come the count of the country of t

find the number of tuples in the customer relation, we write the customer relation, we write the customer relation, we write

aggregate operator count is used to be a set of the count is used to be a set of the count is used to be a set of the count is used to be a set of the count is used to be a set of the count is used to be a set of the count

tur
turis in die sterne in

To find the average account balance at all branches, we write
select branch-name, avg (balance)

from deposit group by branch-name

Operations like avg are called aggregate operations because they operate on
aggregates of tuples. aggregates of tuples.

At times it is useful to state a condition that applies to groups rather
n to tuples. For example, we mink be just in the proups rather than to tuples. For example, we might be interested only in branches
where the average account halance is more than the interested only in branches where the average account balance is more than \$1,200. To express such a query, we use the having clause of SQL. Predicates in the having clause are applied after the formation of groups, so aggregate operators may be select in the naving clause

select branch-name, avg (balance) from deposit group by branch-name

having avg (balance) > 1200.
The aggregate operator count is used frequently to count the number ine aggregate operator count is used frequencies where the average account is used frequencies
experience order all the full be function. The notation used for this
to find the number of tuples in the customer all collate the formation of tuples in the *customer* relation of the formation of the for

select count (*)
from customer.

 ϵ counting can be use Counting ca
wish to find all

person.

 for \mathbb{P}

Asan

 \mathcal{L}

通道

الگ

among

including

 language suchprogrammer's ability

 we may $\overline{}$

 wesaid

select

clause.Set

notable

aggregate

includesil I represented

empty

count used to the count of statement

> selectfromwhere

clause. Algebra

 $m_{\rm th}$ $\frac{1}{\sqrt{2}}$

 operations, orderingSQL queries to be submitted from

customer tuples

selectfromwhere

alternative, SQL

count is 0.000 \pm

Commercial Query Languages -81

3.3.2 Quel

Section 3.3

but for whom no address is on file. Our approach is to count the number of customer tuples pertaining to each depositor of the Perryridge branch. If the count is 0 for a depositor, we know we have no address for that

r. ^t to

of the 1 is r2 $\overline{1}$

 \mathbf{f}

the second control in the cont

 our bank example, queries Quel. first. we find all cu to - Perryridgebranch: 5 September 1988

range

oft is dependent. "customer-name)

Let

select customer-name from deposit $\text{and } 0 =$ from customer where deposit.customer-name $=$ customer.customer-name

the Perryridge branch. If the Perryridge branch. If the formal depositor of the formal depositor of the formal
 As an alternative, SQL includes a special con.

select customer-name from deposit where branch-name = "Perryridge" and not exists
select * dc_position = $\frac{1}{2}$ from cusiomer includes ^a special construct for the application of in the above example. The predicate exists takes ^a select

 \overline{SOL} is as nowerful in avorassivances as the selectional state of \overline{SOL} relational algebra (which,
as we said earlier, is equivalent in nower to the relational calculum COI branch-name ⁼clause. Set union and difference appear in both the relational algebra and customer includes the five basic relational algebra operators. Cartesian product is represented by the from clause of SQL. Projection is performed in the select clause. Algebra selection predicates are represented in SQL's where SQL. SQL allows intermediate results to be stored in temporary relations;
thus we may encode any relational algebra expression in SQL.

SQL offers features that do not appear in the relational algebra. Most uses $\frac{d}{dx}$ critical statutes that do not approximate $\frac{d}{dx}$ strictly more powerful than the algebra.

In this section, we have seen that SQL offers a rich collection of earliers, including capabilities not included in the formal query languages.¹ and the value of the value of tupl

someone of tuple variable to tuple variable to the variable to attribute to tuple variable to tuple variab aggregate operations, ordering of tuples, etc. Many SQL implementations allow SQL queries to be submitted from a program written in a general
Duroose language such as Pascal. PLG Earters C as Cabal TLL purpose language such as Pascal, PL/1, Fortran refere aniguage such as raskal, $1D$ *l*, roman, C , or C 000. This ex
the programmer's ability to manipulate the database even further show how PL/1 and SQL are combined in an actual system in Chapter 15. al | 1

offers features that do not appear in the relational algebra. Most

 $\sigma_{\rm eff}$ and $\sigma_{\rm eff}$ features, capabilities not include in the formal query languages in the formal $\sigma_{\rm eff}$

how P111 and SQL are combined in an actual system

 these features are the aggregate operators. Thus, SQLis strictly and the strictly strictly and the strictly strictly and the strictly strictly and

Quel was introduced as the query language for the Ingres database system. The basic structure of the language for the Ingres database system.

relational calculus. Most Ouel quories edosely parallels that of the tuple

relational calculus. Most Ouel quories are relational calculus. Most Quel queries are expressed using three types of
clauses: range of, retrieve, and where clauses: range of, retrieve, and where.

- Each tuple variable is declared in a range of clause. We say range of t is r to declare t to be a tuple variable position of t is r to declare is declared in a range of clause. We say range of t
is r to declare t to be a tuple variable restricted to take on values of
tuples in r.
- The retrieve clause is similar in function to the select clause of SQL.
- The where clause contains the selection predicate.
- A typical Quel query is of the form:
	- range of t_1 is r_1 range of t_2 is r_2 range of t_m is r_m
retrieve $(t_{i_1} \cdot A_{j_1}, t_{i_2} \cdot A_{j_2})$ where P

The t_i are the tuple variables. The r_i a r -name

 $t.A$

as $f[A]$ in the tuple relational calculus.

(uel does not
anion, or minus $\begin{bmatrix} \text{ucl} \\ \text{umin, or minus} \\ \text{furlike EOL} \end{bmatrix}$

trary relations;
 $\begin{array}{c} \text{Let} \\ \text{Let} \\ \text{where} \end{array}$ settern to our bank example, and $\begin{array}{c} \text{Here} \\ \text{where} \end{array}$ queries using Quel. First, we find all cust LA

range of t is deposit neve (t.customer-nar
Ann thea where $t.brank-name = "Perrying"$

mers havrng an account at the

relational control algebra operations in the section of the intersection of the intersect, union, minus allows
The furthermore, allows the query "Find all customers having a loop of nested subqueries (unlike SQL). That is, we cannot have ^a nested retri ' ' where clause. . evevwhere clause rnsrde than

As

Perryridgerange

range

In

Perryridge

kal

Unfortunately,

customers and the second second and responsi

problem in

٦١.

depositors.

from

guided

not include the second control include that \mathbf{r}

t, l 3

Let us "Find

using

retrievewai

 \sim

relational calculus. '

another example.
Bernard example.

of ^s is barrow

 \mathbf{r}

thunk-name ⁼

consider ^a

this operation.

^s (s ^e borrow

or

retrieve tijde verheerde verheerd.
P

Commercial Query Languages 83

retrieve (t.customer-name, s.customer-city) where $t.brank-name = "Pervridge"$ and t .customer-name = s.customer-name

> rangeretrieve

 \mathbf{I}

range

append

where

rangeretrieve

range $\overline{}$

is depositor de positor de positor de la provincia de la provincia de la provincia de la provincia de la provi
La provincia de la provincia d

temp (madam-name)

Note that Quel, like SQL, uses the logical connectives and, or, and not, rather than the mathematical symbols " \wedge ," \neg ", and " \neg " as used in the tuple relational calculus.
As another example of a query involving two relations, consider the

range of t is borrow

range of s is customer

query "Find all customers who have both a loan and an account at the
Perryridge branch." of ^s is arstomev

tange of s is borrow $r = r$ ange of t is deposit $(s.$ customer-name $)$ $\frac{dS}{d\Omega}$, like Ω and t.customer-name = s.customer-name

> In SQL, we had the option of writing a query such as the above one using the relational algebra operation intersect does not include this operation.
Let us consider a query for which we used the union operation in

SQL: "Find all customers who have an account, a loan, or both at the Perryridge branch." Since we do not have a union operation in Quel, and

 $\{t \mid \exists s \ (s \in borrow \land t[customer-nan])\}$ \land s[branch-name] = "Perryridge") \vee 3 u (u e deposit \wedge t[customer-name] = u[customer-name] \land u[branch-name] = "Perryridge")}

 $\frac{d\mathbf{u}}{dt}$ is the union operation operation operation operation is that in the tuple relational can all customers who have a local cust customers from *both* tuple variable *s* (whose
" variable u (whose range is *deposit*). In Quel, our retrieve clause must be either \mathbf{b} our tuple relationships expression for the this query:

retrieve s.customer-name
or **existenties** a customer-name $\qquad \qquad \text{or} \qquad \text{if}$

If we choose the former, we exclude those depositors who are not not depositors. does not lead us to a query α and α and α α α

is that in the tuple relation $\mathcal{L}_\mathcal{A}$ and we obtain

s.custamer-mre

mustang-namesing-namesing-namesing-namesing-namesing-namesing-namesing-namesing-namesing-namesing-namesing-nam
Pang

both tuple variable ⁵ (whose range is borrow) and tuple variable

(whose range is deposited in \mathcal{C} range \mathcal{C} range clause must be either either the either \mathcal{C}

choose the former, we exclude the former second that are not borrowers.

In order to write this query in Quel, we must create a new relation and
ert tuples into this new relation I at us call this set a new relation and the this contribute this query in Quel, we must create a new relation and
insert tuples into this new relation. Let us call this new relation *temp*. We
obtain all depositors of the Perryrides branch by until the state of obtain all depositors of the Perryridge branch by writing

> range of u is deposit retrieve into temp (u.customer-name) where u , branch-name = "Perryridge"

 $\frac{14}{18}$ Causes a new relation, $\frac{1}{18}$ append command.

range of s is borrow append to temp (s.customer-name) \mathbf{r}_{in} the temperature of \mathbf{r}_{in}

The append command oper
That the tuples as :

We now have a relation *temp* containing all customers who have an
ount, a loan, or both, at the Perneides have it. This who have an executive a relation temp containing all customers who have an
account, a loan, or both, at the Perryridge branch. This relation may, of
course, have the same customer approximation. course, have the same customer appearing more than once. Quel, like
SQL, eliminates duplicates only if specifically requested to do so. If we

range of t is temp
retrieve unique (t.customer-name)

Quel sorts temp and eliminates duplicates.

to temp (s.cu5toruer—nnme)

s Quel
tempo
n that
Perryri
?
?
? The strategy of using append allows us to perform unions in Quel To perform a set difference $r - s$ (minus in SQL), we create a temporary n t
Ol P
2 n text
ar
Pele also in s. To find all customers who have an account at the Perryridge
branch but do not have a loan from the Perryridge
branch but do not have a loan from the Perryridge branch but do not have a loan from the Perryridge branch, we write the following following

range of u is deposit retrieve into temp (u.customer-name)
where u.branch-name = "Perryridge"

of this point *temp* has all customers used. who are not a set tus point *temp* has all customers who have a temporary of the set of this customers who have a temporary relations of the customers who have a hean. \mathcal{I} in \mathcal{I} find all customers who have an a Pauli customers who have an a Pauli customers who have an a Pauli customers who have a ry ⁿ at are

Perryridge'branch

is zero."

is borrow

"much—name ⁼

.more complicated

 of ^t is deposit is branch of ^s is br'anch retrievetrimaran-name

countu

 \mathbf{S}

 where shrank-city ⁼ "Brooklyn" andshard—name - t.bmnch-rurme) ⁼

(u.lmmd1-mrn¢ where u.bmnch-city ⁼

The relation temp contains the desired list of customers. We write

range of t is temp
retrieve (t.customer-name)

to complete our query.

take

 \mathbf{R} expression computes the average based band \mathbf{R}

represented by the present of this society of this society is sort are presented by the sort are sort are sort

relation

however, we had a set of the set o

 \mathbb{R}

ran

where \mathbb{R}

a segundar

frequent,

 $\overline{}$

 \sim

 allows thesyntaxia (n. 19
|

r. and the contract of

 \mathbf{b} and write write

clause in

the form

com

where

acgcgrueris

Fortunately, there is a more natural way to express this query in First, however, we must introduce the Quel aggregate expressions, which
take the form take the form s.customa-name

tempo contains the desired list of customers. We write \sim \leq aggregate-operation> (i.A where P)

> , min, or range of the variable, A is an attribute, and P is a predicate similar to the t whose clause in a setting A is an attribute, and P is a predicate similar to the where clause in a retrieve. An aggregate expression may appear anywhere a constant may appear.

lette our thus, to find the average account bala

Perryridge branch, we write
 $\frac{1}{2\pi}$ m_{ν} is the σ $\begin{array}{ccc} \n\text{r} & \text{r} \\
\text{r} & \text{r} \\
\text{r} & \text{r}\n\end{array}$ retrieve

> range of *i* is deposit retrieve avg (balance where branch-name = "Perryridge")

 $\mathcal{L} = \mathcal{L} \mathcal{L}$ $\overline{\mathbf{a}}$ accounts whose balance is higher than the average balance at the
where the account is held. We write: where the account is held. We write: $\mathbf x$ retrieve. An aggregate expression may appear any $\mathbf x$ m anch n

range of u is deposit
range of t is deposit
retrieve t .account-number where $t.balance > avg(u.balance where$ $u.branch-name = t.branch-name$ e of the original contract \mathbf{u} . The set of the set

The above avg $(...)$ expression computes the average balance of all a
stable branch-name enterprise of the strategy of the stra \overline{m} at the branch represented by *t*. Because expressions of this sort are $\frac{1}{2}$ bilgher than the average balance at the average balance $\frac{1}{2}$ where $\frac{1}{2}$ is $\frac{1}{2}$ frequent, Quel allows the syntax:

> range of t is deposit retrieve t.account-number where t balance $>$ avg (t balance by t branch-name)

Section 3.3

The avg (...) expression performs the same computation as above. For a given *t*, the average balance is computed of the set of all tuples having the same value on the branch-name attribute as t.branch-name.

Let us return to the query "Find all customers who have an account at the Perryridge branch but do not have a loan from the Perryridge branch." We can write this query using the count aggregate operation if we think of the query as "Find all customers who have an account at the Perryridge branch and for whom the count of the number of loans from the Perryridge branch is zero."

range of t is deposit range of u is borrow
retrieve t.customer-name tetrieve *i.customer-name*

where *i.branch-nume* = "Perryridee" and

where *i.branch-nume* = "Perryridee" and count (u.loan-number where u.branch-name = "Perryridge" and u . *customer-name* = t . *custom* write this query using the count aggregate operation if we think our think of we think of we think of we think

 $\begin{array}{c} \n\text{min}_{\lambda} \text{ or any,} \\
\text{min}_{\lambda} \text{ is an odd,} \\
\text{min}_{\lambda}$

Quel offers another aggregate operation that is applicable to this example, called any. If we replace count in the above query with any, we obtain 1 if the count is greater than 0; otherwise we obtain 0. The advantage in using any is that processing can stop as soon as one tuple is found. This allows faster execution of the query.

As a more complicated example, consider the query "Find all customers who have an account at all branches located in Brooklyn." Our strategy for expressing this query in Quel
many branches there are in Brooklyn. The many branches there are in Brood the branch decount. The count aggregate operation we use
Therefore, we use the countu operation, which is

 $r = \frac{1}{\pi} \sum_{i=1}^{n} \frac{1}{i}$ is deposit using any is that processing can stop as a stop as soon as one tuple is the query. The query is the query. The query $\frac{1}{2}$ is the query. The query is found to the query. The query is found to the query. The query is th allows the contract of the con retrieve *t.customer-name*
"Find all alleges" where countu (s.branch-name where s.branc. and s.branch-name = $i.b$ ranch-name) = branches the in Brooklyn. Then counter α counter α counter α is an accounter the number of α customer α counter α

We have observed that Quel is related clouds.
Calculus. The range of clause corresponds to the calculus. The range of clause correspond insertion and deletion to state in Quel some of the queries that we could

"Brooklyn")

FST-IBM000864

between **between**

query

The

This

t, e rigil

part

ensure

some

need

tlrl-'AI-l]

 12. \mathbf{r}

'._.=..

 \mathbb{E} is a set of \mathbb{E}

tlari bilinmeyen bilinmeyen bilinmeyen bilinmeyen bilinmeyen bilinmeyen bilinmeyen bilinmeyen bilinmeyen bili

 $\Box V$

 \mathcal{L} and \mathcal{L} and \mathcal{L}

languages,

 systemof QBE. Unlike

en en de segon de la provincia de la provincia
En la provincia de la provinci

that the kth

 $\frac{1}{\sqrt{2}}$

expression

there exists \mathbf{f}

t2 ^e 72A ^A 1m

as:

 \mathbf{u}

Section 3.3

 does amverston

is that . The . expressed "by example." in stead of giving. The stead of giving \mathcal{L} 3'the

 user ^gives anexample of . is

Commercial Query Languages

between Quel and the tuple relational calculus, consider the following Quel query

range of
$$
t_1
$$
 is r_1
\nrange of t_2 is r_2
\nrange of t_m is r_m
\nretrieve $(t_{i_1} \cdot t_{j_1}, t_{i_2} \cdot t_{j_2}, \ldots, t_{i_n} \cdot t_{j_n})$
\nwhere P

 $\begin{bmatrix} \mathbf{q} \\ \mathbf{q} \end{bmatrix}$

$$
\{i \in \mathbb{R} \mid t_1, t_2, \ldots, t_m \ (t_1 \in r_1 \land t_2 \in r_2 \land t_m \in r_m \land \text{if } t_{i_1} \land t_{j_1} \} = t_{i_1} [A_{j_1}] \land t [r_{i_2} \land t_{j_2}] = t_{i_2} [A_{j_2}] \land \ldots \land \text{if } t_{i_m} \land t_m = t_{i_m} [A_{j_m}] \land P \ (t_1, t_2, \ldots, t_m))\}
$$

This expression can be understood by looking at the formula within
the "there exists" formula in three parts: ..
a .'...,

- $t_1 \in r_1 \wedge t_2 \in r_2 \wedge \dots \wedge t_m \in r_m$. This part constrains each tuple in t_1, t_2, \dots, t_m to take on values of tuples in the relation it ranges over.
- \bullet $t_1r_{i_1}r_{j_1} = t_{i_1}A_{j_1} \wedge t_{i_2}A_{j_2}$ $\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$ $\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$ and $\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$ and $\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$ and $\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$ and $\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$ and $\begin{bmatrix} 1$ ensure that the k th attribute in tuple t corresponds to the k th entry in the retrieve clause. Conside some tuple of r_{i_1} (since range of t_{i_1} is r_{i_1}
need $f(A_1) = t_1(A_2)$. We used the n $t(r_{i_1}.A_{j_1})=t_{i_1}.A_{j_1}$ to be able to deal with the possibility that the same attribute name appears in more than one relation.
	- \bullet $P(t_1, t_2, t_3, \ldots, t_n)$. This part is the constraint on accentable values $t_1, t_2, ..., t_m$ imposed by the where clause is

\mathbb{E} $\begin{bmatrix} 1 \\ 2 \end{bmatrix}$ (3.3.3 Query-by-Example

P01. t2! tm). This par^t is the constraint on acceptable values for

^a query language and the database

 $\begin{array}{c} \text{F1} \\ \text{F2} \end{array}$ Overy-by-Example (QBE) is the name of both database system which includes this language. There are two distinctive tillities of QUE. State those quely tanguages and programmers languages, QBE has a two-dimensional syntax. A query in a one-dimension name appears in the second second
In the second secon 3.13

of the Comment of the Comme

Figure 3.13 QBE skeleton tables for the bank example.

language (for example, SQL or Quel) can be written in one (possibly very long) line. A two-dimensional language requires two dimensions for its expression. (There does exist a one-dimensional version of QBE. We shall
not consider this version in our discussion of QBE.) The second distinctive feature of QBE is that queries are expressed "by example." Instead of giving
a procedure for obtaining the desired angusta "by example." Instead of giving a procedure for obtaining the desired asswer, the user gives an example of
what is desired. The event and asswer, the user gives an example of what is desired. The system generalizes this example to compute the
answer to the curry Dependies these was example to compute the **EXECUTESPONDENCE between QBE and the bank example.**
Queries in OBE are expressed using

the can can be display with all skeletons, the user selects to two-dimensional language requires two dimenpsions for irt:

corn ute the

examplerow

element isreally

bring

constants,

other

any

above query and the set

of the variable ^x

Quel and

after the "P." command.

custan enumeration

be done to answer the domain relational calculus query:

executive the above grows above and the system

attribute as the borrow

"Perryridge"

communication in the second series of the second series of the second series in the second series of the serie

Unlike

automatically.

 \mathcal{T} \blacksquare

to

write:

흞

borrow

"Perryridge"system

customer-city

domain

languages

qualification.

Suppose

Section 3.3

Perryridge

\$1200":

Perryridge

 \overline{a}

Commercial Query Languages 89

$\{1 \; \exists s (\epsilon \text{ customer})\}$

A technique similar to the one above can be used to write the query "Find all customers who have both an account and a loan at the Perryridge branch":

Suppose our query involves a less than or greater than comparison, rather all customers who have buy be found and a loan at the Perryridge b
A loan and a loan a loan and a loan a loan at the Perryridge branch is below to be a loan of the Perryridge br balance of more than \$1200":

 branch-name PerryridgeUntil now, all the conditions we have imposed were connected by "and." To express an " α " in QBE, we give a separate example row for the two conditions being " α "-ed, using distinct domain variables. Consider the $\begin{bmatrix}\n\text{Redwood branch, or both"}:\n\end{bmatrix}$ query "Find all customers having an account at the Perryridge branch, the Redwood branch, or both":

conditions were very match with the pertyridge branch and the Redwood branch in QBE, we give a separate row for the Pertyridge branch and the Redwood branch in $\frac{1}{2}$

The critical distinction between these two queries is the use of the same domain variable (x) for both rows in the latter query, while, in the former query, we used distinct domain variables $(x$ and $y)$. To illustrate this, note that in the domain relational calculus, the former query would be written as $\frac{dS}{dt}$

given query. The user fills in these skeletons with "example rows." An example row consists of constants and "example elements." An example element is really a domain variable. To distinguish domain variables from constants, domain variables are preceded by an underscore character (" _ ") as in _ x. Constants appear without any qualification. This is in contrast to most other languages in which constants are quoted and variables appear without any qualification.

 $\frac{1}{2}$ bring up the skeleton for the *deposit* relation and fill it in as follows:

The above query causes the system to look for tuples in *deposit* that $\frac{1}{2}$ $\frac{1}{2}$. The strip of the state for the branch name attribute. For each such $\frac{1}{2}$. value of the variable x is "printed" (actually displayed) because the find a solution command "P." appears in the customer-name column next to the variation of the Perryridge branch, we have a solution of the Perryridge branch, we have a straight of the Perryridge branch, which is a straigh the value of the customer-name attribute is assigned to the variable x. The

Unlike Quel and SQL, QBE performs duplicate elimination automatically. To suppress duplicate elimination, the command "ALL." is inserted after the "P." command.

 $\frac{1}{20}$ The primary purpose of variables in QBE tuples to have the same value on certain attributes. Suppose we wish to find all customers having a loan from the Perryridge branch, and their cities. We write: p erforms duplicate elimination eliminat

 \mathbb{R} is the same value of certain \mathbb{R} to aver with the shows guess. Suppose \mathbb{R} $\frac{1}{2}$ customers having a loan from $\frac{1}{2}$ customers having "Perryridge" as the value for the branch-name attribute. For each such tuple, the system finds tuples in *customer* with the same value for the *customer*- $\frac{1}{n}$ customer-city attributes are displayed. Observe that this is similar to what $\begin{array}{ccc} \text{nat} & & \end{array}$ would be done to answer the

finds tuples in borrow with

finds tuples in customer with the same value for the customer-

Chapter 3

 $\{ 1 \ \exists b,a,n \ (**b,a,x,n> \epsilon deposit \land b = "Perryridge"\})**$ $\vee \exists b,a,n (\leq b,a,x,n > \epsilon \text{ deposit } \wedge b = \text{``Redwood''})$

while the latter query would be written as

 $\{ \langle x \rangle : \exists b,a,n \ (\langle b,a,x,n \rangle \in \text{deposit} \land b = \text{``Perryridge''}) \}$ $\wedge \exists b,a,n (\langle b,a,x,n \rangle \in deposit \wedge b = "Redwood")$

egation are expressed in QBE
top under the relation name sign (") in a table skeleton under the relation name and next to an

ignization the example fow in the corrow

ever, has a major effect on the processing of

Sesides

Sesides

Sesides

Sesides difference is the """ appearing next to the example row in the borrow
skeleton. This difference, however, has a major effect on the processing of
the query. QBE finds all x values for which
the query. QBE finds all x value the query. QBE finds all x values for which Compare the above query with our earlier query "Find all customers who
have both an account and a loan at the Perryridge branch." The only
difference is the "n" appearing next to the example row in the borrow
skeleton. Thi

- 1. There is a tuple in the deposit relation in which branch-name is "Perryridge" and customer-name is the domain variable x.
- 2. There is no tuple in the *borrow* relation in which *branch-name* is that QBE eliminates duplicates

"Perryridge" and *customer-name* is the same as in the domain variable x.

3.4 Modifying the D. **There** is no tuple in the *borrow* relation in which *branch-name* is that QBE eliminates column ensures that all balances are considered.
The includes and *customer-name* is the same as in the domain variable x.
The SQL

The """ can be read as "there does not exist."
The fact that we placed the """ under the relation name rather than $\frac{1}{2}$ under an attribute name is important. Use of a " γ " under an attribute in the difference information, remove information, or change information and information or change information. under an attribute name is in
name is a shorthand for " \neq ." under an attribute name is important. Use or a "" under an attribute is a shorthand for "#." To find all customers who have accounts at a shorthand for "#." To find all customers who have accounts at some insert and delete f he cases a shorthand for "#." To find all customers who have accounts at and some insert and delete operations in our Quel two different branches, we write
the database. Instead, we dealt with temporary

"-1" can be mad as "there does not exist." We have restricted our attention until now at the second our atte

The contract of the contract of

and is important. Use of a "-1" under an attribute '. Information, remove information, remove information, remove information, remove information, remove information. While we did do the second do '4 under we did do '4 und :
23: shorthand for «, t -- To find an customers who have accounts at f. some insert and delete operations in our
Quel examples, we never altered l name in our Quel examples, we never altered l name in our Quel examples, w different branches we write the database in database in a security temporary relations constructed for the second for

Section 3.3

Commercial Query Languages -91

In English, the above query reads "display all customer-name values that appear in at least two tuples, with the second tuple having a branch-name values that
different from the first."
different from the first." different from the first."

If the result of a query is spread over several tables, we need a
chanism to display this result in a circle table. The spread a mechanism to display this result in a single table. To accomplish this, we
mechanism to display this result in a single table. To accomplish this, we
can declare a temporary result table which is that is a second ish this, can declare a temporary result table which includes all the attributes of the
result relation. Printing of the desired result is done by including the Example \mathbf{c} command "P," only in the result table.

Quenes that involve negation are expressed in QBE by placing a not

sign (¬) in a table skeleton under the relation name and next to an

example row.

Let us now consider the query "Find all customers who have an

Let us n 'e n Example row.

Let us now consider the query "Find all customers who have an

account at the Perryridge branch but do not have a loan from that branch":

at two different branches. We want to include an "x \neq lones" con in the above query. We do that by bringing up the condition box and entering the constraint " $x \neq$ Jones":

The compare the above query with our earlier query "Find all customers who constraints on the domain 1 includes a not the domain 1 includes a compare to those of SQ.

To find the average balance at all branches, we may wri

analogous to SQL's "group by branch-name is analogous to SQL's "group by SQL".
is computed on a branch-by-branch haste. The average balance l

 $\frac{d}{dt}$

3.4 Modifying the Database
We have restricted our attention until now to the extraction of information from the database. We have not, however, shown how to add new iincluded included in the defence of the defect in the sole purpose of helping us to express the overy relations correctly the sole purpose of helping us to express the overy

The formal query languages (the relational algebra and the relational l

 \mathbf{t}_t to the extraction \mathbf{t}_t

FST-IBM000867

 $\sum_{i=1}^{n}$

the latter \mathbf{H}

 $\frac{d}{dx}$

al.

the above query

is ^a tuple in

f cot that we laced the that we laced the relationship \mathbf{x}

Compare 19

There

the contract of the contract o
The contract of the contract o

"Perryridge"

"Perryridge"

, where \mathbf{H} , we have the \mathbf{H}

examp \mathbf{C}

^A

Modifying the Database 07

3.4.1 Deletion

deletion in the control of the control of

only

In

of tuples from

delete tuples from

all tuples from \mathbf{u}

 ϵ ii

deletedeposite de la provincia de la contexta d
Desenvolvers de la contexta de

deletes

FST-IBM000868

empty where clause. The request:

deposit α

Delete all accounts at branches located in Needham

above delete reques^t first finds all branches in

^give some examples of SQL

Deletion

^Pwhich

We

communication and the state of the state

an

deletes

We

to

Deletion of tuples from a relation is simple. A delete request is expressed in much the same way as a query. However, instead of displaying tuples to the user, the selected tuples are removed from the database. We may delete only whole tuples; we cannot delete values on only particular attributes. In SQL, a deletion is expressed by

R^2 and R^2 and R^2 are R^2 and

 P represents a predicate and r represents a relation. Those tuples t in r for which $P(t)$ is true are deleted from r . $\begin{array}{c} \text{if } \text{we} \\ \text{if } \text{we} \end{array}$

a relation is the simple of the same way as way as way as well be delete command operates $\frac{1}{2}$ was a command for one behavior. The same second relations, the user, the user is the selected tuples are relation. The predicate in the selected from the database. A may complex as a select command's where clause. A complex as a select command's where clause. At the other extreme, we can
 $\frac{25}{24}$ have an empty where clause. The request: have an empty where clause. The request:

delete borrow

deletes all tuples from the borrow relation. (Well-designed systems will seek confirmation from the user before executing such a devastating request.)

note that a delete come examples of SQL delete requestion. If we give some examples of SQL delete requestion.

 \bullet Delete all of Smith's account uses on $f(x)$ believe all or smith's account records.

as a select community where clause \mathcal{M} where customer-name = "Smith"

• Delete all loans with loan numbers between 1300 and 1500.
delete *borrow*

 \vec{B} delete borrow devastating request.) $\frac{3.44}{2.44}$ requests: $\frac{3.44}{2.400}$ and $\frac{3.44}{2.44}$ requests $\frac{3.44}{2.44}$

· Delgte all accounts at branches located in Needham

delete deposit

where branch-na from branch

Delete all loans with loan numbers between ¹³⁰⁰ and 1500.

~'7'tniumt

number of the state of the 1300 and 1300 a where $branch\text{-}city = "Needham")$
The above delete request first finds all branches in Needham, and then .v.'

Note that although we may delete tuples from only one relation at a time, we may relieve the use the upper strom only one relation at a
time, we may reference any number of relations in a select-from-where
embedded in the where clause of a data. embedded in the where clause of a delete.

If the delete request contains an embedded select that references the
lifthe delete request contains an embedded select that references the that increase the existence is the same existence of the deleted select that references the
relation from which tuples dre to be deleted, we face potential anomalies.
Suppose we want to delete the records of all seems Suppose we want to delete the records of all accounts with balances below
the average. We mish to delete the records of all accounts with balances below the average. We might write

delete *deposit*
where balance < (select avg (balance) from deposit)

we may delete tuples from deposit, the reference reference as we delete tuples from deposit, the vertex reference and \mathbf{w}_i we reevaluate the select for each tuple in deposit, the final result will depend upon the order in which we process tuples of *deposit!*

Such ambiguities are avoided by the following simple rule: During the execution of a delete request, we only mark tuples to be deleted; we do not actually delete them. Once we have finished processing the request, that is, once we are done marking tuples, then we delete all marked
tuples. This rule guarantees a consistent interpretation of deletion. Thus, balance et request above de
expect. (Some implementati expect. (Some implementations of SQL simply disallow delete requests like
the above one.)

$3.4.2$ J

Section 3.4

delete tu

implementations of SQL

 reevaluate the select upon the order in

> inserti i

generally, we might want to

 $\frac{1}{2}$

3.4.2 Insertion
To insert data into a relation, we either specify write a query whose result is a set of tuples to be inserted. Obviously, the and the following simple rule: $\frac{1}{2}$ are a deleted by the execution of a deleted; we do be domain. Similarly, tuples inserted must be of the correct arity.
The simplest insert is a request to insert one tuple. Suppose we wish

we simplest insert is a request to insert on
to insert the fact that Smith has \$1200 in acconsistent interpretation. Thus, thus, thus, thus, thus, thus, th
3. $\frac{1}{2}$ branch $\frac{1}{3}$ wanch, we write

$\frac{1}{3}$ disallo the extending insert into *deposit* above one.) provides the set of \mathcal{A}

More generally, we might want to insert tuples based on the result of a query. Suppose that we want to provide all interests struct generally, we might want to insert tuples based on the result of a
query. Suppose that we want to provide all loan customers in the
Needham branch with a \$200 savines account that the start Needham branch with a \$200 savings account. Let the loan customers in the
as the account number for the new savings account. Let the loan number serve as the account number for the new savings account. Let the loan r

insert into deposit

provide all loan customers in the New York is a strong that the New York is a strong that the New York is a strong that the Ne $\mathcal{L} \left(\mathcal{L} \right)$

select branch-name, loan-number, customer-name, 200 from borrow where $branch$ -name = "Needham"

Instead of specifying a tuple as we did earlier, we use a select to specify a set of tuples. Each tuple has the branch-name (Needham), a loannumber (which serves as the account number for the new account), the name of the loan customer who is being given the new account, and the initial balance of the new account. \$200.

3.4.3 Updating

instead

 \mathbb{E}

of the loan

 Updating $-$ are situations in the situation of \mathbb{R}

balance

specify

number

changing

increased

 \mathbf{L} s: now it

percentstatements:

construction and

and

There are situations in which we wish to change a value in a tuple without to change. Instead, we use the update statement. As was the case for $\frac{1}{2}$ insert and delete, we may choose the tuples to be updated using a query. changing all values in the tuple. If we make these changes using delete There are situations in which we wish to change a value in a tuple without

changing all values in the tuple. If we make these changes using delete

and insert, we may not be able to retain those values that we do not wish

Suppose interest payments are being made, and all balances are
increased by 5 percent. We write \mathbf{c} increased by 5 percent. We write account, \$200. loan

update deposit set balance = balance * 1.05

The above statement is applied once to each tuple in deposit.

 all values in the tuple. If we make these changes using delete andAside from $\sum_{k=1}^{\infty}$ percent interest, while all others receive 5 per change. In the update statements:
Statements: example, might like

 $\frac{d}{dx}$ update *deposit*
set *balance* = *balance* $\frac{1}{2}$ 1.06 where $balance > 10000$

> update deposit set balance $=$ balance $*$ 1.05 where $balance \leq 10000$

the where clause of the undate statement may contain any. construct legal in the where clause of the select statement (including nested selects). Note that in the above example the lorder in which we has been selected between. The line in the above example the forder in which we wrote the two update statements is important. If we changed the order of balance in the update statements is important. If we changed the
e two statements, an account whose balance is just under \$10,00 receive 11.3 percent interest! use of views in SQL.

3.5 Views

selects). Note that in the above example the (order in which we

 \overline{a}

statements, and account whose balance is just under \mathcal{L}

our examples up to this point, we have operated at the conceptual model

are the actual relations stored in the database. The database in the database in the database. The database in the database. The database is defined in the database in the database. The database is defined in the database

balance ' 1.055 μ

interest payments are being

interest, while all others receive $\mathcal{L}_{\mathcal{A}}$

by 5 percent. We write

above statement is applied

setting the setting of the setting of the set

legal in

level. That is, we

percen^t interest!

 \mathbf{t} two 1

-41

Views

 θ model level. That is, we have assumed that the collection of relations we are given are the actual relations stored in the database. the where clause of the select statement (including nested

Section 3.5

"branch-name,

each in the control of the

visible to

 \mathbf{f} the create view community

to be called all-customer. We define this customer. We define this customer. We define this customer. We define this customer. We define the customer \mathbf{r}

all-customer at

 deleteview.

create

example, consider the view

view

 \mathbf{u} wishes

as follows: '

must ^give the view

expression>

Assume

<u> شاه به جایزه بازی میشود.</u>

that is, for

use the term

made

Views 95

It is not desirable for all users to see the entire conceptual model. Security considerations may require that we "hide" certain data from certain users. Consider, for example, a derk who needs to know a customer's loan number but has no need to see the loan amount. This derk should see a relation described, in the relational algebra, by

¹
Il tranch-name, loan-number, customer-name (borrow)

than is the conceptual model. An employee in the advertising department, f_{max} a query.
Security conceptual model model model model model model model model model. Security considerations may be that if f_{max} contract we would like $\frac{1}{2}$ and the navelet and account or a loan at that branch. to create for the employee is

II_{branch-name, customer-name} (deposit)

 \cup $\Pi_{branch-na}$

We use the term view to refer to any relation not part of the conceptual model that is made visible to a user as a "virtual relation." It is possible to support a large number of views on top of any given set of actual relations.

Since the actual relations in the conceptual model may be modified by insert, update, or delete operations, it is not generally possible to store views. Instead, a view must be recomputed for each query that refers to it.
In Chapter 9 we shall consider techniques for reducing the overhead of views. Instead, a view must be recomputed for each query that refers to it. create for the employee is \parallel and use of views in SOL

A view is defined in SQL using the create view command. To de
view, we must give the view a name and state the query that comput A view is defined in SQL using the create view command. To define a view. The form of the create view command is

create view v as <query exposed

any
 $\frac{d}{d\log a}$ where <query expression> is any legal query ex must be recomputed by p.
As an example, consider the view consisting of branches and their

Chapter 3 we shall consider the overall consider the overhead of reducing the overhead of reducing the overhead of the definition the definition of th this view as follows:

create view all-customer as (select branch-name, customer-name
from deposit)

union
(select branch-name, customer-name from borrow) name is represented in the contract \mathbf{r} represented in the contract of \mathbf{r}

FST-IBM000869

Once

virtual relation

that we have

model of

clerk. We define this view

allows ^a view

However, to

 Reject the insertion and return an error message to the user. $\mathcal{I}^{(n)}$ and $\mathcal{I}^{(n)}$ and $\mathcal{I}^{(n)}$ into the borrowing int

am- The Communication of the Comm

symbol null represents ^a null~value, or ^place-holder value. It signifies that

insertion: for amount. There are two

 value is unknown systemsthe

create

clerk

difficulty and the set

conceptual

modification

the borrow

S

insertion <u>in</u> the series of the series

barrow

constructed. The construction of the construction of the construction of the construction of the construction o

that ^a relation

we have defined by the control of the cont

Section 3.5

to

branch-city as

Views 97

Once we have defined a view, the view name can be used to refer to the virtual relation the view generates. View names may appear in any place that a relation name may appear. Using the view all-customer, we can find all customers of the Perryridge branch by writing

> select customer-name from all-customer where $branch-name = "Peryridge"$

problems if updates, insertions, or deletions The difficulty is that a modification to the database expressed in terms of a $\frac{1}{2}$ view must be translated to a modification to conceptual model of the database. We illustrate the problem of database modification through views with a simple example. 1—".—

Consider the clerk we discussed earlier who needs to see all loan data Eunsi
in the *bor* the clerk. We define this view as $\frac{1}{2}$ is not possible. performance of the performance of the performance of the addition using the definition using the absence of the aggregate operators. As a result of the aggregate

View loan-info as

Select branch-name, loan-number, customer-name

from borrow

from borrow 3 Select branch-name, loan-number, customer-name
3 They Prefine to the from borrow
3 The special key prima be tised in a predicate to the prima problem in a null valid valid valid valid valid v ems in the enormous insertions. Or expresse using VIEWS. To find a sertion \mathcal{L} and \mathcal{L} in the borrow create views who and \mathcal{L} **relative** to the construction of the con select branch-name, loan-number, customer-name

is that a modification to the database expressed in the database expressed in terms of a model in terms of a m
The database expressed in the actual relation to a model to the actual relation to the actual relations in the
 allowed, the clerk may write of database through views with a simple example. The simple example example. \mathcal{C}

insert into loan-info values ("Perryridge", 3, "Ruth")

 $\frac{1}{100}$ is the actual relation from which the view loan-info is constructed. However, to insert a tuple into borrow, we must have some value for amount There are two reasonable approaches to dealing with this h this insertion:

 $\overline{}$ appear wherever a relation name is from borrow, customer allowed, the custome

 \bullet Insert a tuple ("Perryridge", 3, "Ruth", null) into the borrow relation.

The symbol null represents a null-value, or place-holder value. It signifies that the value is unknown or does not exist.

is a system take the latter approach and
"However the presence of pull values adds complexity However, the presence of null values adds complexity

relation. Mr."

ستشف ستنتضخ والمراد والأنزلية والمستنصف مفتقد

Assume we have inserted the above tuple, producing the relation shown in Figure 3.14. Consider the following query to total all loan balances:

select sum (amount) from barrow

It is not possible to perform addition using null. Similar problems arise where bunch-nume – retrynage
Recall that we wrote the same query in Section 3.3 without using views.
All comparisons invaling since the same duributes. except count ignore tuples with null values on the argument attributes.

All comparisons involving null are false by definition. However, a the above tuple to the above tuple to the supplex in the *borrow*:
tions in the **the balance**, we write

> select customer-name from borrow where balance is null

The illustrate another problem resulting from modification of the argument of the argument attributes. The argument attributes of the case of the case of the database through rious suit a summatic from modification of the

select branch-name, customer-city customer-name when \mathbf{w} is the set of \mathbf{w}

Figure 3.14 A borrow relation containing null values.

Section 3.5

Redwood

hton

customer-nameJones

Smith

 \blacksquare is a set of \blacksquare

a a

JohnsonGlenn

Brooks

nu

3.15

database through the second secon

A

a useful mechanical mechanical mechanical mechanical mechanical mechanical mechanical mechanical mechanical me
Expression mechanical mechanical mechanical mechanical mechanical mechanical mechanical mechanical mechanical

 Δ

consequences.

n.

 view lists the cities in which borrowers of each branch live. Consider following■ 劉 relations \mathbf{g}_k . "Woodside") relations shown that the contract of the contr not include the second sec that all comparisons of \mathbb{R}^n where \mathbf{r} customer. The **customer and customers** customer relations. ^asystemsimpose \blacksquare the following α de fined Under thisforbiddeneneral problem in the second seco of this subject. Another view~relatedmodel.al this model, the user is ^given ^a viewconsisting of \mathbf{c} one relationdatabase. Theconcernedwith remembering what attributes are in which relationships are in which relationships are in which relation. Thus, thus que establecer en la contec are i $\frac{1}{\sqrt{2}}$ in a standard relational database system. For example, a universal- relations a universal version of SQLThereuniversaldevelopeduniversal-relation

FST-IBM000871

This view lists the cities in which borrowers of each branch live. Consider the following insertion through this view:

insert into branch-city values ("Brighton", "Woodside")

The only possible method of inserting tuples into the borrow and customer relations is to insert ("Brighton", null, null, null) into borrow and (null)
 R_{max} and $R_{\$ the relations shown in Figure 3.15. This turns out to be unsatisfactory since

from branch-city

does not include the tuple (" possible method of into the view definition (*borrow.customer-nume* = customer and customer and customer and customer and customer and customer and customer relations of the tuples added to the *borrow* and customer relations. to be false. Thus, Perryridge \mathbf{a} , \mathbf{b}

and customer relations.
As a result of the anomaly we have just discussed, many database
customs in the following constraint on modifications allowed through views: $\ddot{\mathbf{S}}$.
j systems impose the following constraint on modifications allowed through Brighton

 \bullet A modification

Under this constraint, update, insert, and the view definition (borrow.cuslomer.cuslomer.cuslomer.cuslomer.cuslomer.cuslomer.cuslomer.cuslomer.cuslomer.cu
above. above.

The general problem of database modification through views is a $\prod_{i=1}^{n}$ subject of current research. The bibliographic on this subject.
Another view-related research area of interest is the *universal relation*

model. In this model, the user is given a view consisting of one relation.
This one relation is the natural join of all relations in the actual relational database. The major advantage of this model is that users need not be $\frac{d}{dx}$ most queries are easier to formulate in a unive than in a standard relational database system. For example, a universalrelation version of SQL would not need a from clause.

There remain unresolved questions regarding modifications to current research. The bibliographic notes in the bibliographic notes in the bibliographic notes in the bibliographic mention recent works are the bibliographic mention of the bibliographic mention of the bibliographic ment developed on the best definition of the meaning of certain complex types
of universal relation queries

Figure 3.15 Tuple

We can summarize our d are a useful mechanism for s of the database through views has potentially disadvantageous consequences. A strong case can be made for requiring all database modifications to refer to actual

3.6 Summary

 s strong case can be made for requiring all database modifications are α

refer to actual relations in the database.

 \mathbf{S}

for simplifying database queries, but modifimtion

 $\frac{1}{2}$ The relational data model is based o

formulate in ^a universal-relation database system

 $\overline{}$

^a new

Exercises 101

and update (modify) tuples. There are several languages for expressing these operations. The tuple relational calculus and the domain relational calculus are nonprocedural languages that represent the basic power required in a relational query language. The relational algebra is a procedural language that is equivalent in power to both forms of the relational calculus. The algebra defines the basic operations used within relational query languages.

IThe relational algebra and the relational calculi are terse, formal and the relational end of the relational of the relational calculi are terse, formal and the relational end of the database system.
In appropriate for cas Commercial database system have, therefore, used languages with more

"syntactic sugar." These languages include constructs for update, insertion,

and deletion of information as well for querying the database. We have

an es include constructs for update, insertion, and the set of the set and update (modification of indication as well for querying the database. We have $\frac{1}{2}$ are sextended the relational database of flows 3.4 Equal and the relational database of flows 3.17 Circums of Figure 3.17 Circums

THE Cuel, and QBE.

The Cuel and QBE consider the telational database of Figure 3.17. Give an example to show how such the state o

The tuple of the database. We used SQL as an example to show how such the relational datab $\frac{1}{2}$
 $\frac{1}{2}$
 views can be defined and used. en

algebra defines the basic operations \mathbf{F} relationship and the second secon

c. Finally, the contract of the

II

works

. In the contract of $\mathcal{C}(\mathcal{E})$. In the contract of the

f I manages in the second control of the secon

 \mathbf{q}_i .

"syntacticCommercialsus

exercred **N**

e de la construcción de la construcción
En la construcción de la const

ll Igure 1999
Lleibhean an Dùbhlachadh an Dùbhlachadh an Dùbhlachadh an Dùbhlachadh an Dùbhlachadh an Dùbhlachadh an Dùbhlac

and

 \mathbf{E}

 \mathbf{I}

that are

language

calculus. The

- 3.1 Design a relational database for a university registrarie of fice. The 3.1 Design a relational database for a university registrar's office. The $\frac{1}{2}$ \bullet SQL office maintains data about each class, including the instructor, the enrollment, and the time and place of the class meetings. deletion of information of information as well for the data about each class, including the instructor, the

encollment, and the time and place of the class meetings. For each
 Example 12. Denothe the time and place is r
- Suddent-Class pair, a grade is recorded.

3.2 Describe the differences between the terms relation and relation

Scheme. Illustrate your answer be referring to your solution to **individualization** 3.2 Describe the differences between the terms relation and relation $\frac{1}{2}$ for each of the queries below:
 Example 1999

Exercise 3.1. Exercise 3.1.
- 1. The tuple of all people who work for First (1998). The tuple relational database corresponding to the E-R diagram of the formula corporation.
The corporation.
The tuple who work for Find the name and city of all people

Figure 3.16 E-R diagram.

t van die 19de eeu n.C. Soos ander die 19de eeu n.C. Soos ander die 19de eeu n.C. Soos ander die 19de eeu n.C.
Verwys

none-m' -ult-n<~-

 $\mathcal{L}(\mathcal{$

l'antico della contratta della contratta della contratta della contratta della contratta della contratta della
L'antico della contratta della contratta della contratta della contratta della contratta della contratta della

 $-$ in the set of the s ' . 'l l'antication de la construction de la construction de la construction de la construction de la construc
'' le construction de la construct

I: The contract of the contr

 \sim

- 3.4 Construct the following SQL queries for the relational database of Exercise 3.3.
	- a. Find the total number of persons whose car was involved in an accident in 1092 . accident in 1983.
	- b. Find the number of accidents in which the cars belonging to
	-
	- d. Delete the car "Mazda" belonging to "John Smith."
- ϵ . Add a new accident record for the Toyota belonging
- the domain relational ' C58 ' ' "III ^I calculus
	-
- with the number of the tr $\frac{1}{2}$ $\frac{1}{2}$
- the relational calculus

the relational calculus
 the domain relational calculus
 the domain relational calculus

a control of the domain relational calculus \bullet The domain relational calculus
 \bullet SQL
	-
	-
	-

-
- Sough a relational database corresponding to the E-K diagram of Corporation.
Figure 3.16. The difference and city of all people who work for First Bank
Corporation. Corporation.
	-

lives (person-name, street, city) ' .. ^l works (person-name, company-name, salary)
located-in (company-name, city) α den α corden α manages (person-name, manager-name)

en 1988.

Figure 3.16 E-R diagram.

Figure 3.17 Relational database. Figure 3.17 Relational database. ¹ 1'.

Corporation and earn more than \mathcal{L} - . In the second corporation and earn more than \mathcal{L} - . In the second corporation and earn more than \mathcal{L} - . In the second corporation and earn more than \mathcal{L} - . In t

(person-name, street, city) (ll ^V

lives

 \sim \Box

lives in Newtown. Write relational algebra expressions equivalent to the following il

 ${\bf p}$ ${\bf p}$

Exercises 103

- d. Find all people who live in the same city as the company they work for.
- e. Find all people who live in the same city and on the same street as their manager.
- f. Find all people who do not work for First Bank Corporation.
- g. Find all people who earn more than every employee of Small **Bank Corporation.**
- Assume the companies may be located in several cities. Find \blacksquare Let relations $r(R)$ and $s(S)$ be given. Give an expression in the tuple all companies located in every city in which Small Bank relational calculus that i all companies located in every city in which Small Bank all people who live in the same component in the same city as the company theorem in the same company they show how show the same company of a .

to represent many-to-many- to-many- $\frac{1}{2}$. To represent many-to-one-to in: $\mathbf{b} \cdot \mathbf{\sigma}_{\mathbf{B}} = 17 \text{ (r)}$ \blacksquare solution \blacksquare

- \bullet SQL \bullet First \bullet
- \bullet Quel all people who earn more than every employee of α and α = α

d.

Find

Find

Find

h.Assume

m:

i QBE

b.Find

S SQL

a.Modify

b.lla i

 $\frac{1}{2}$. Find the set of $\frac{1}{2}$.

c. Given the contract of the c

first the company of the company

average de la propincia de la

 $\ddot{\mathbf{S}}$

 \vdash a. \mathbf{f}

salary

all managers ^a ¹⁰

. 3.11

d. Give all managers ^a ¹⁰ percen^t raise unless the salary becomes 5- {GAG

 $-$ e. Delete all tuples in the warts relation for employees of S and \mathcal{S} contains \mathcal{S}

3.6

for each of the queries below:

- $\mathbf v$ of $\mathbf v$. people working in their company.
- b. Find the company employing the most people.
- c. Find the company with the smallest payroll. $\begin{bmatrix} c & r_1 & \cdots & r_2 \\ r & r_2 & \cdots & r_n \end{bmatrix}$
- d. Find those companies that pay more, on average, than the $d \cdot r_1 \cap r_2$
average salary at First Bank Corporation.
- **E.** $r_1 r_2$
3.7 Consider the relational database of Figure 3.17. Give an expression in SQL for each query below: ' equivalent to:: $\prod_{AB}(r_1) \rtimes \prod_{BC}(r_2)$
	- a. Modify the database so that Jones now lives in Newtown.
- the company of the company employees of First Bank Corporation a 10 percent raise.

c. Give all managers a 10 percent raise.
	- c. Give all managers a 10 percent raise.
	- d. Give all managers a 10 percent raise unless the salary becomes greater than \$100,000. In such cases, give only a 3 percent
raise. $(c \cdot b > 1 \forall a (a, b > b \forall c)(a, c > e s)$ raise.
e. Delete all tuples in the works relation for employees of Small
	- E. Defecte all tuples in the works relation for employees of Small $\wedge \ll 1$, $\wedge \ll 1$, for each query below: _ '

.
Percent raise raise raise raise de la percent de la percent aux aux des percent de la percent de la percent de

la serie de la construcción de la
Entre el construcción de la constr

 $\frac{1}{\sqrt{2}}$ block and the contract of the co

Corporation ^a ¹⁰ percen^t raise. domain relational calculus expressions. ^l l: . ^I

b de la companyation de la companyation de la companyation de la companyation de décide de la companyation d

> [|] <a,b> ^e ^r ^A <a.c > ^e 5)) l' "l I:

\$100,000. In such cases, give only a 3 percent c. (11). Va (11). Va (11). Va (11). Va (11). Va (11). Va (11). V
2000. In such cases, give only a 3 percent c. (11). Va (11). Va (11). Va (11). Va (11). Va (11). Va (11). Va

- 3.8 In Chapter 2, we showed how to represent many-to-many, manyto-one, one-to-many, and one-to-one relationship sets. Explain how primary keys help us to represent such relationship sets in the relational model.
- 3.9 Let the following relation schemes be given:

$$
R = (A, B, C)
$$

$$
S = (D, E, F)
$$

- a. $\mathbf{H}_A(r)$
 $\mathbf{b} \cdot \mathbf{F}_B = \mathbf{b} \cdot (r)$ $\begin{array}{c} \text{c. } r \times s \\ \text{c. } r \times s \end{array}$. Records and the second s \bullet Quel \bullet Quel \bullet (ARC) \bullet
	- (D E n lull Bank Corporations)

	(D E n lull Bank Corporations) 3.10 Let $R = (A, B, C)$ and let r_1 and r_2 both be relations on $\begin{bmatrix}\n\text{for each of the queries below:} \\
	\text{for each of the queries below:} \\
	\text{if the original calculator is a function of the original function.}\n\end{bmatrix}$

- a. Modify the database so that Jones now lives in Newtown. 3.11 Let, $R = (A, B)$ and $S = (A, C)$, and let $r(R)$ and $s(S)$ be l
1 domain relational calculus expressions.
 $a. \{ \langle a \rangle | 3 b (\langle a,b \rangle \in r \land b = 17) \}$

b. $\{ \langle a,b,c \rangle | \langle a,b \rangle \in r \land \langle a,c \rangle \in s \}$
	-
	-

 $=$ (AB) and S $=$ (AC), and $5(5)$ be relations. Let $\mathcal{O}(5)$

-
- d. $\{ \leq a > 1 \}$ $\exists c \; (\leq a, c > \epsilon \; s \wedge \exists \; b_1, b_2 \; (\leq a, b_1 > \epsilon \; r$

universal relationships and the set of the s
The set of the set of t

discussed

entire database as one relation

the conceptual scheme. One such system

of the relation data model appears in $\mathcal{O}_\mathcal{A}$

3.12 Write expressions for the queries of Exercise 3.11 in

- a. QBE
- b. QUEL
- c. SQL

3.12

3.13

3.14

3.15

Research

constructing

in 1999.

 \mathbf{z} in \mathbf{r}

 \mathbf{F} and [1972b].

 \blacksquare

[query

 versions by IBMthe

systems. Query-by-Example is described

language SQL

auspices of the Amerian

is described in Todd $\frac{1}{2}$ 76]. The original definition of the original definition of the relation of the

relational algebra an be found in Codd [1972b] and Ullman [1982a].

was first defined by Chamberlin et al. [1976]. The chamberlin et al. [1976]

IBM

original

Center, andPRTV

Chapter

(1978b).

 standardfor

a. $\mathcal{D}(\mathcal{D})$ b.c.SQL

View

itself,

li.

e bibliographic control of the set of the se

 \mathbf{I}

insertion

employeessystem

- 3.13 Consider the relational database of Figure 3.17. Using SQL define a view consisting of manager-name and the average salary of employees working for that manager. Explain why the database W_{in} expressions for the structure $\frac{1}{2}$ in $\frac{1}{2}$ in
	- $\begin{array}{c} \mathbb{R}^n \\ \mathbb{R}^n \end{array}$
- \mathbf{C} insertion of the tuple (Brighton, Woodsie **consist manager in the tuple (Brighton, Woodside)** through *branch-city*.
Example and the angle of the angle of the angle of the a equal to \hat{I} itself, but if $i \neq j$, then $\perp_i \neq \perp_j$. One application of marked nulls is w branch-

Bibliographic Notes

List 1988
ENET START REASONS WAS DESCRIPTED INTO THE INTERNATIONAL START REASONS FOR THE UPPER PROPERTY OF THE UPPER PROPE
Company of the database of the data base of the database of the database of the database of the Some systems allowThe relational model was proposed by E. F. Codd of the IBM San Jose
Research Laboratory in the late 1960s [Codd 1970]. Following Codd's i at the original paper, several research projects were formed with the goal and marked nulls is to marked nulls in the goal of marked nulls is to marked in the stock of marked nulls in the marked nulls in the marked nulls constructing practical relational database systems, including System (Section 3.5). Show how marked nuclear the university of the California at Berkeley, Query-by-Example at the IBM T. J. Watson Research relational algebra is in Codd [1970] and that of
E. F. F. Codd [1972b]. A formal proof of the equivalence Codd [1972b]. A formal proof of the equivalence of the relational calculus and relational algebra can be found in Codd $[197]$ i
70
in
in l
n
a
-" (n r a - ,) Zig
t'Mwm
an the universal relation view Codd's $\frac{1}{2}$ by Ullmanll9823, $\frac{1}{2}$ language is similar to that of Quel. However, since the user sees Scientific \mathbf{I} $\frac{1}{2}$ is cussed in $\frac{1}{2}$ is a text devoted in $\frac{1}{2}$ Chapter 15. The bibliographic notes of that chapter provide references to those systems. Query-by-Example is described in Zloof [1977] and IBM

provide the process of SQL was first defined b
R attending System R at $\frac{2\pi}{\pi}$ Current versions of SQL available in commer
described by IBM [1982] and Oracle [1983]. Then
under the ausnices of the American National Str under the auspices of the American National State of the IBM Sciences.
At the IBM SCIENTIFIC SCIENTIFIC SCIENTIFICAL **Peterlee, Standard SQL language. The SQL language served in and Ingresor are discussed in an analyzie are discussed in an**
Peterlet in a standard interest in a more general relational database have proposal for a more general relational database language being developed by ANSI Committee X3H2 (Committee on Computer and Information Processing).

Quel is defined by Stonebraker et al. [1976], Wong and Youssefi [1976], and Zook et al. [1977]. A commercial version of Quel is described in RTI $(1983]$

The problem of updating relational databases through views is addressed by Cosmadakis and Papadimitriou [1984], Dayal and Bernstein 980], Fagin et al. [1982] that a universal relation view is simpler to use. In such systems, the user views the entire database as one relation and the system translates operations on the universal relation view into operations on the set of relations forming the conceptual scheme. One such system is System/U, which was developed at Stanford University in 1980-1982. System/U is described by Ullman(1982a, 1982b) and Korth et al. [1984]. The System/U query language is similar to that of Quel. However, since the user sees view branch-

to allow the $\frac{1}{2}$ and $\frac{1}{2}$ control and Database is addressed by Cosman Cosman at Bernstein at a line and Bernstein at a line Bernstein at a line Bernstein at a line and Bernstein at a line and Bern

System, PLIS, is discussed by Maier et al. [1981]
Example: Seneral discussion of the relation data mode *b* School and Ullman et al. [1982a, 1982b]. Several discussion of the relation data mod

Several experimental data base systems have been built to the claim of the claims have been built to the claims have been built to t devoted exclusively to the relational data model

FST-IBM000874

9

In the preceding sections, we have considered how to structure the data in the database. These decisions are mode at the the database. These decisions are made at the time the data in
designed. Although it is possible to change at the time the database is
designed. Although it is possible to change this designed. Although it is possible to change this structure, it is relatively
costly to do so. Thus, when a query is presented to the system, it is necessary to find the best method of finding the answer using the existing
database_structure....There-are-a-large-number-of-possible strategies for processing a query, especially if the query is complex. Nevertheless, it is usually worthwhile for the system to spend a substantial amount of time
on the selection of a strategy. Typically, strategy selection can be done using information available in main memory, with little or no disk
accesses The actual oxecution of the neurony, with little or no disk the preceding sections, we have the transfer of data from disk is structure the data in t main memory and the central processor of the computer system, it is it is a divintage to spend a considerable amount
structure to do 50. Thus, when a considerable amount
 $\frac{1}{2}$

$\begin{array}{r} \text{9.1 Query Interpretation} \\ \text{Given a sum, the} \end{array}$

 $t_{\rm eff}$ into an equivalent query which can be computed more efficiently.

of the strategy for the strategy fo ^a query, is called query optimization. There is ^a close analogy betweencode optimization by a compiler and query optimization by a compiler and query optimization by a database optimiza

the issues in the issues in the interest of the
Interest of the interest of th

Although

to

example, we

 \blacksquare

system. We shall study

responsibility

Since

a civen a query, there are generally a variety of
answer. For example, we saw that in SOL a of Example, we saw that in SQL a q
several different ways. Each way of expressing
strategy for finding the angular memory or no disk strategy for finding the answer. However, we do not expect users to write $\frac{d\mathbf{r}}{d\mathbf{r}}$ the strangle many involvements in a way that suggests the most
becomes the responsibility of the system in the transfer of the transfer of the system in the transfer of the system in the system in the becomes the responsibility of the system to transform the query as entered memory and θ becomes the responsibility of the system to transmitted by the user into a considerable amount of processes. θ for processing a query, is called *query optimization*. There is a close analogy between code optimization by a compiler and query optimization by a database system. We shall study the issues involved in efficient query processing both in high-level languages and at the level of physical access

 sawQuery optimization is an important issue in any database system since different ways. Each structure was a structure way the difference in execution time between a good strategy and a bad one for the antierence in execution time between a go may be huge. In the network model and the system of the system of the most efficient strategy. Thus, it is an enter the system to the system to the system to the system to

 $\frac{1}{2}$

optimization

programming

hierarchical continues and continues are a series of the continues of

data manipulation

entire application

ت
ت

 \mathbf{q}

here and

indicate—the—

first action

to the contract of the contrac

relation

etc. If

the graduate state

to

expression

optimization

for processing the query.

Equivalence of Expressions .

the query will be executed.

all references to

in most compiler texts (see the bibliographic notes).

choice must be made as to exactly how

is based primarily on the number of disk accesses required.

seen that there are several ways to express ^a ^given query in the

relational algebra is ^a procedural language. Thus, each

Thus,the

query

^a relational query

Chapter 9

and

⁼

⁼

⁼

"branch-name.

we are concerned as a set of \mathbb{R}^n

not have customer-city ⁼

customer relationships and continued the continued position of the

 weneed

of the

relational algebra expression

intmediate moreresult. efficiently if there were ^a way to reduce the size of the

"Port of the Port of the Po

not consider those tuples of the customer relation that

"Port Chester." By reducing the number of tuples

represented by the relational algebra expression:

that we need to access, we reduce the size of

optimization is left, for the most part, to the application programmer. Since the data manipulation language statements are embedded in a host programming language, it is not easy to transform a network or hierarchical query to an equivalent one unless one has knowledge about the entire application program.

Since a relational query can be expressed entirely in a relational query language without the use of a host language, it is possible to optimize queries automatically. Since the most useful optimization techniques apply to the relational model, we shall emphasize the relational model in this chapter. The bibliographic notes reference techniques for optimization of network and hierarchical queries.

Before query processing can begin, the system must translate the query into a usable form. Languages such as SQL are suitable for human use, but ill-suited to be the system's internal represer useful internal representation of query is one based on the relational $\frac{1}{2}$ algebra. The only difference between the form of the relational algebra we define shall use here and that of Chapter 3 is that we shall add redundant parentheses to indicate the order of operation our relation

 automatically. Sincethe most useful optimization the system must take the most useful optimization the system must take $\frac{1}{2}$ the query into its internal form. This translation the query into its internal form. This translation process is similar to that bibliographic notes reference techniques for optimization of networkand by the public of a complicit. In the process of generating the internal form of the query, the parser checks the syntax of the user's query, verifies processing can be that the relation names appearing in the query are namely
database, etc. If the query was expressed in terms of database, etc. If the query was expressed in to be the system of a more with the relational algebra $\frac{1}{2}$ of a attribute the system of a compute a view name with the relational algebra $\frac{1}{2}$ attribute $\frac{1}{2}$ attribute $\frac{1}{2}$ attribute $\frac{1}{2}$ attribu internal representation of the second se expression to compute a view.

I The details of the parser are beyond the scope of this text. Parsing is a form of the relation of the relationship relationship covered in most compiler texts (see the bibliographic pates). ...~. w. Parsing is 美麗
for the assets and name of all banks who all banks who are depositors by
tional all banks who

the overed in most compiler texts (see the bibliographic notes).
- Chapter and reduction evaluation evaluation evaluation evaluation is a section.
- living in Portician evaluation evaluation evaluation is a section of the form, the optimization process begins. The first phase
done at the relational algebra latel the ottenut is into its internal form. This is internal form. This is similar to the translation process is similar to that the to that is similar to that is similar to the team of the team the expression that is equivalent to the given expression efficient to execute. The next phase involves the selection of a detailed strategy for processing the query. A choice must be made as to exactly
how the guery will be expected. A choice of querific is the set how the query will be executed. A choice of specific indices to use must be
made. The order in which turbs are presented indices \therefore made. The order in which tuples are processed must be determined.
final choice of a strategy is based primarily on the number of disk according to compute a view. The compute \mathfrak{m} required. t_{t} is the scope of the scope of the scope of this covered. e **an**ti final choice of a strategy is based primarily on the number of disk accesses

9.2 Equivalence of Expressions

process begins at the relational algebra is a procedural language. Thus, each relational is a subset of the relational and the relational is a procedural language. Thus, each relational is a probable of the relational is a algebra expression represents a particular sequence of operations. We have already seen that there are several ways to express a given quality

relational algebra. The first step in selecting a query processing strategy is to find a relational algebra expression that is equivalent to the given query
and is efficient to axecute and is efficient to execute.

We use our bank example to illustrate optimization techniques. In particular, we shall use the relations customer (Customer-scheme), deposit
(Denosit-scheme), and heavely (Partial of Customer-scheme), deposit (Deposit-scheme), and branch (Branch-scheme). As was the case earlier, we
define our relation scheme as (Blanch-scheme). As was the case earlier, we define our relation scheme as follows:

 $Customer\text{-}scheme = (customer\text{-}name, street, customer\text{-}city)$
Deposit-scheme = (branch-name, account-number, customer-name, balance) Branch-scheme = $(branch-name, assets, branch-city)$

9.2.1 Selection Operation

Let us consider the relational algebra expression we wrote in Chapter 3 for as an expression of the relations customers customers customers customers customers ($\frac{1}{\sqrt{2}}$ and $\frac{1}{\sqrt{2}}$ branches expectively. living in Port Chester":

$\Pi_{branch-name, assets}$ ($\sigma_{customer-city}$ = "Port Chester"
(customer \bowtie deposit \bowtie branch))

e internal account-name, accounter-not-number, customer-name, account-number, branch-name, branch-name, branch
balance, branch-city, branch-city, branch-city, branch-city, branch-city, branch-city, branch-city, branch-cit

on disk. This means that in add
the relations customer-denotity as $\frac{1}{2}$ are retainons $\frac{1}{2}$ disk to read and write intermediate results. Clearly, we could process the query more efficiently if there were a way to reduce the size of the intermediate result. intermediate result.
Since we are concerned only about tuples

termined. The to resident of the eight concerned only about tuple.

f disk accesses the eight at the eight at the eight attribute results in the eight at the eight at the eight that do not have customer-city = "Port Chester." By reducing the number of les of the *customer* re the intermediate result. Our query is now represented by the relational algebra expression:

each relational $\frac{1}{2}$
tions. We have $\frac{1}{2}$ w deposit w branch)

Chapter 9

.tis Ju-ti Ju-ti Ju- $\sim 10^{-10}$ relational algebra _000 million of the Contract of Perform $\frac{1}{2}$ and $\frac{1}{2}$ a with $($ (customer \bowtie deposit) \bowtie branch) . The contract of \mathcal{L} . Section . I is the contract of \mathcal{L} in \mathcal{L} is the contract of \mathcal{L} is the contract of \mathcal{L} is the contract of \mathcal{L} is the contrac $\begin{array}{ccc} \overline{a} & \overline{b} & \overline{c} \\ \overline{b} & \overline{c} & \overline{d} \\ \overline{c} & \overline{d} & \overline{d} \end{array}$ is then we cannot application.
The cannot apply customencies in the second state of the se \mathbb{R} to the customer relation, since the predicate involves at the predicate involves at the predicate involves at \mathbb{R} . 'l 4: ', _ customer-city or balance. If we decide to process the join as: ' ^E ^p ,-E .' ,/ «customer» (customer and customer and customer and customer and customer and customer and customer and customer
Experimental customer and custom $\mathcal{O}(\mathbb{R}^n)$ (see Figure). $\frac{1}{2}$ $\frac{1}{\sqrt{2}}$ $\frac{1}{2}$. ($\sigma_{\text{customer-city}}$ = "Pon Chester" (customer)) \approx (σ_{balance} > 1000 (deposit)) in the expression of computing the expression of consideration since it consider a large relation since it consider a large relation $\mathcal{L} = \frac{1}{2}$ - i. .: $\overline{}$ i. customer-city of the state of the . The contract of the contrac of the above expressions select tuples with customer $\frac{1}{2}$: andbalance **x** of the expression writing the expression of the expression with the expression of the expression writing the expression with the expression of the exp provides ^a new now, rewrite our query ("melanin-city Chasm," (customer)) In (chum > {000 (deposit)) _ ⁷ However, deposit ^N branch is hkely to be ^a large relation since it contains EE ^E in . ⁱ . E. -- 5- ⁱ ^t y in the second contract of the second contract of the second contract of the second contract of the second co the contract of l E \sim E experience in the contract of ^I ^l EEl E t en deux de la Communicación de la Communicación de la Communicación de la Communicación de la Communicac
En la Communicación de la Comm \4

The above example suggests the following rule for transforming relational algebra queries:

· Perform selection operations as early as possible.

In our example, we recognized that the selection operator pertained only to the customer relation, so we performed the selection on customer directly. Suppose that we modify our original query to restrict attention to customers with a balance over \$1000. The new relational algebra query is

 Γ $\alpha_{P_1}(\sigma_{P_2}(e))$ $($ using the second transformation \mathcal{L} is \mathcal{L} in the \mathcal{L} in the \mathcal{L} in the \mathcal{L} is a relational algebra expressional transformation \mathcal{L} and \mathcal{L} and \mathcal{L} and \mathcal{L} are predicates an α (customer \bowtie deposit \bowtie branch))

directly to the customer relation, since the predicate involves attributes of $\sigma_{P_1}(\sigma_{P_2}(e)) = \sigma_{P_1}(e) = \sigma_{P_1}(\sigma_{P_2}(e)) = \sigma_{P_1}(e) = \sigma_{P_1}(e)$. customer and deposit. However, the branch relation does not involve either customer-city or balance. If we decide to process the join as: $\frac{3}{4}$

> $((\text{customer} \Join \text{deposit}) \Join \text{branch})$ $\sum_{i=1}^{n}$ $\sum_{i=1}^{n}$ $\sum_{i=1}^{n}$ $\sum_{i=1}^{n}$ $\sum_{i=1}^{n}$ $\sum_{i=1}^{n}$ $\sum_{i=1}^{n}$ $\sum_{i=1}^{n}$ $\sum_{i=1}^{n}$

then we can rewrite our query as:

 $\Pi_{branch\-name, \; assets}$
(($\sigma_{customer\text{-}city = \text{``Port Chester''} \land \text{ balance} > 1000$) $\Pi_{branch\text{-}name, \; assets}$ ^e uiva ences amon--
"Port Chester City = "Port Chester" \land balance > 1000 $\%$
 \lor branch)

Let us examine the subquery:

1000. However, the latter form

Both of the above expressions select tuple
Chector" and belong ≥ 1000 . However, the Chester" and balance > 1000 . However, the latter form of the expression provides a new opportunity to apply the "perform selections early" rule.
We now rewrite our query as: $\frac{1}{3}$. The set of the system of customer-city = "Port Chester" (Customer) We now rewrite our query as:

We now add a second transformation rule:

• Replace expressions of the form:

Ъy

bection 5.2

 N'2)

"Port. ' - We consider the computation of the computation of the computation of the computation of the computation

' 1 N "W" N "W

 $\sigma_{P_1 \wedge P_2}$ (e)

We cannot apply the selection: $\frac{1}{2}$ An easy way to remember this transformation is by noting the formulation is the formulation of the form: if $\frac{1}{2}$ α stomer-city = "Port Chester" \land balance > 1000

By modifying queries so that selections are done early, we reduce the size of temporary results. Another way to reduce the size of temporary results An easy way to remember this transformation is by noting the following El E" ^t relati nal ' : '- ' ' Chapter 3 that natural join is associative. Thus, for all relations r_1 , r_2 , and r_3 . $\frac{3}{2}$ $\frac{1}{2}$, $\frac{7}{2}$, and $\frac{7}{3}$:
(r, \bowtie $r_n = r$, \bowtie (r, \bowtie r,)

\n The subquery:\n
$$
(r_1 \bowtie r_2) \bowtie r_3 = r_1 \bowtie (r_2 \bowtie r_3)
$$
\n

 $\sigma_{\text{customer-city}} = \text{``Port Chester''} \land \text{ balance} > 1000 \text{ (Customer & deposit)}$ although these expressions are equivalent, the costs of computing
 $\sigma_{\text{customer-city}} = \text{``Port Create into two. forming the expression:}$

We can split the selection predicate into two, forming the expression:
A strange assets (σ extomer city = "Pon Chester" (σ extomer \sim 4 σ extons i σ and σ and σ results σ by the size of temporary re

 $\frac{1}{100}$ $\frac{1$ Hes with customer-city = "Port \cdot . We could choose to compute *deposit* \bowtie branch first and then join to latter form of the expression

However, deposit \bowtie branch is likely to be a large relation since it contains

branch first and then join the result :E Eu

 $\label{eq:R1} P_{\text{max}}(P) = \frac{1}{2} \sum_{i=1}^N \frac{P_{\text{max}}(P_i)}{P_{\text{max}}(P_i)} \leq \frac{1}{2} \sum_{i=1}^N \frac{P_{\text{max}}(P_i)}{P_{\text{max}}(P_i)}$

306

 $t = t$

number of widely

obtain

1' and

is, natural join

could

 weto the $\frac{1}{2}$

customer has a series of the series of t

projection

first join

<u>in</u>

and

cartesian di Ba

9.2.3 Projection Operation '1

Thus, the

(customer-city

probably

compute deposit ^N

expression as a series of the series of

attributes and a

results. The

North Communication

relations

.' easy

Query Processing

Chapter 9

to the "perform

following

"branch-name.

modify

Other Operations

form

one tuple for every account. However,

$\sigma_{\text{customer-city}} = \text{``Port Chester''}$ (customer)

is probably a small relation. To see this, note that since the bank has a large number of widely distributed branches, it is likely that only a small fraction of the bank's customers live in Port Chester. If we compute:

$\sigma_{\text{customer-citiv}} = \text{``Port Chester''}(customer)) \bowtie \text{ deposit}$

 $\frac{d}{dx}$ are countries to the contract account them by a resident checker. Thus, the temporary relation we must store is smaller compute deposit \bowtie borrow first. advantage v

mpute *deposit* \bowtie *borrow* first.
There are other options to consider for evaluating our query. We do $\frac{4}{3}$ easy to change the order before displaying the result. Thus, for relations r_1 and r_2 : relations r_1 and r_2 : of the bank's customers live in Port Chester. [f we compute: property in \mathbb{R} is a set of \mathbb{R}

"Port Chester" (customer) to deposit to deposit

one that is, in turn for the continuative.
Using this fact, we can consider rewri expression as relation we must store is smaller than if we borrow first.
Borrow first states that the state of the sta
The state of the st

$\Pi_{branch\,-name, \, assets}$ ((($\sigma_{customer\,-city}$ = ")
 Θ_{branch} to consider Θ_{random} to change the order before displaying the result. Thus, for all o
h

 \mathbf{r}_1 attributes in common between *Branch*-scheme and Customer-coheme so the For a calify just a cartesian product. If there are c customers α chester and b branches, this cartesian product generates be tuples. $\mathbf{f}_{\text{ref}}^{\text{ref}}$ and $\mathbf{f}_{\text{ref}}^{\text{ref}}$ can consider relations that consider $\mathbf{f}_{\text{ref}}^{\text{ref}}$ and $\mathbf{f}_{\text{ref}}^{\text{ref}}$ and $\mathbf{f}_{\text{ref}}^{\text{ref}}$ and $\mathbf{f}_{\text{ref}}^{\text{ref}}$ are those that $\mathbf{f}_{\text{ref}}^{\text{ref}}$ and \mathbf{f}_{\text nbmmh—Mme, assetswe would reject this strategy. However, if the user had entered the above expression, we could use the associativity and commutativi
join to transform this expression to the more efficient express earlier. operation performed. Note, however, that there are no \mathbf{h} . \mathbf{u} . \mathbf{h} o
C".
C".
C". we obtain α appears that α the α the α to we would reject us suategy. However, it the user had entered the above the space of the Operations.

9.2.3 Projection Operation

 product will produce ^a large temporary relation. As ^a result, reject this strategy. However, if the user had entered the above

consider another technique for reducing the size of temporary

this expression to the more efficient expression to the more efficient α

 $\overrightarrow{=}$ is a cane in the move consider another technique for reducing $\overrightarrow{=}$ \mathbf{b} be two consider another technique for reduction product \mathbf{c} and \mathbf{c} results. The projection one ration like the selection possible projection operation, like the selection operation, reduces the $\frac{1}{2}$
 $\frac{1}{2}$

> Imrcvml-w.._

 $\mathcal{F}(\mathcal{F})$, where $\mathcal{F}(\mathcal{F})$

size of relations. Thus, whenever we need to generate a temporary relation, it is advantageous to apply any projections that are possible. This suggests a companion to the "perform selections early" rule we stated earlier:

• Perform projections early.

Consider the following form of our example query:

 $\frac{3}{2}$. Equivalence of Expressions 3000 $\frac{3}{2}$. The Chester' (Customer) l \bowtie deposit) \bowtie branch)

 $\frac{d}{dt}$ When we compute the subexpression:

selections early" rule we stated

(customer-name, customer-city, branch-name, account-number, balance)

 $\frac{3}{2}$ We can eliminate several attributes from we must retain are those that:

• Appear in the result of the query or

involves a sequence of natural joins. We chose the sequence of natural joins. We chose the sequence of natural

 \bullet Are needed to process subsequent operations.

 $\mathcal{L}=\mathcal{L}^{\text{max}}$ $\frac{3}{4}$ By eliminating unneeded attributes, we reduce the number of columns of reduced. In our example, the only attribute we need is branch-name.
Therefore, we modify the expression to:

tuples, one for $\frac{1}{2}$. $\frac{1}{2}$ $\frac{$

ssion we used $\frac{1}{2}$ if $\frac{1}{2}$ The example we have used involves a sequence of natural joins. We charge $\frac{1}{2}$ intermediate results is intermediate result in this symple homoves natural is in a sequence of natura this example because natural joins arise frequently in practice and because the inatural joins are one of the more costly operations in query process
B. However, we note that equivalence will precedent in query process $\frac{3}{2}$ equivalences below: u set unterence operanc
" branch) i

have seen

algebra

choose from the second contract of the

complex

retain

strategy

'1', the

 $\mathbf{V}(\mathbf{A},\mathbf{A})$

first two

The

third **i**

in

with

major impact on

statistics for each

a strategy and the strategy of the strategy of

 \mathbf{F}

s occupies s, + 55 bytes. \mathbf{p} and for the form:

cartesian

order to perform

<u>tte andere namen av den stad om det som en der stad om det som en der stad om det som en der stad om det som e</u>

 such an estimation, we need to know how ⁱ ofteneach value appears in ^a column. if we assume that each value -

statistic is used to estimate how

⁼ <oalue>

size of ^a record

possible

simply

9.3

Section 9.3

branch-name

tuples inrl ^N nR2 ⁼

$$
\sigma_p(r_1 \cup r_2) = \sigma_p(r_1) \cup \sigma_p(r_2)
$$

\n
$$
\sigma_p(r_1 - r_2) = \sigma_p(r_1) - r_2 = \sigma_p(r_1) - \sigma_p(r_2)
$$

\n
$$
(r_1 \cup r_2) \cup r_3 = r_1 \cup (r_2 \cup r_3)
$$

\n
$$
r_1 \cup r_2 = r_2 \cup r_1
$$

We have seen several techniques for generating more efficient relational algebra expressions for a query. For queries whose structure is more complex than those of our example there may be a large number of possible strategies that appear to be efficient. Some query processors simply choose from such a set of strategies based on certain heuring the strategies of the control of the strategies of the strategi query optimization for each strategy. The final choice of strateg dout and an only after the d
estimate is made

9.3 Estimation of Query-Processing Cost

The strategy we choose for a query depends upon the size of each relation and the distribution of values within columns. In the example \mathbf{S} is that a used in this chapter, the fraction of customers who live in Port Chester ha a major impact on the usefulness of our techniques. In order to be choose a strategy based on reliable information, database systems may store statistics for each relation r . These statistics include: after the details of each strategy have been worked out and an

is matrices in the processing cost of the processing cost of the processing cost of the processing cost of $n_{\rm pl}$ the processing cost of the processing cost of the processing cost of the processing cost of the processin

- \overline{C} $\overline{$ R (for fixed-length) $\frac{n_{r_2}}{n_{r_2}}$ one tuple from
- 3. $V(A,r)$, the number of distinct values that appear in the relation r for $\frac{1}{2}$ tuples in r₁ in this chapter of continuous who live in \mathcal{A} . most difficult case to the case of the

attribute A.

The first two statistics allow us to estimate according

product. The cartesian product $r \times s$ contain product. The cartesian product $r \times s$ contains $n_r n_s$ tuples. Each tuple of $r \times s$ occurries $s + s$ bytes te of a ca<mark>r</mark> α appears with equal probability. Consider a tuple to α tuple that α tuple the set of α r2 with an value of HA]. So tuple t produces the so-

 number of tuples inThe third s

\leq attribute-name \geq = \leq value \geq

number of distinct values that appear in order to nectorm such an estimation 7 for \overline{a} statistics allowappears with equal probability, then $\sigma_{A} = q(r)$ is estimated to have that each value

⁵ contains "r": tuples. Each tuple of ..

 n_r /V(A,r) tuples. However, it may not always be realistic to assume that each value appears with equal probability. The branch-name attribute in the deposit relation is an example of such a case. There is one tuple in the deposit relation for each amount. It is reasonable to expect that the large branches have more accounts than smaller branches. Therefore certain branch-name values appear with greater probability than others.

Despite the fact that our uniform distribution assumption is not always true, it is a good approximation of reality in many cases. Therefore, many query processors make such an assumption when choosing a strategy. For simplicity, we shall assume a uniform distribution for the remainder of this chapter.

tuples. However, it may not a strongle in the size of a natural join is s than estimation of the size of a selection or a cartesian product. Let $r_1(R_1)$ and $r_2(R_2)$ be relations. If $R_1 \cap R_2 = \emptyset$, then $r_1 \bowtie r_2$ is the same as $\frac{1}{2}$ $\frac{1}{2}$ $\frac{1}{2}$ $\frac{1}{2}$ $\frac{1}{2}$ and we can use our estimation technique $\frac{1}{2}$ exactly one binds from ϵ . Therefore exactly one tuple from r_1 . Therefore, the number of tuples in $r_1 \bowtie r_2$ is no $\frac{d}{dx}$ greater than the number of tuples in r_2 .
The many cases that the number of tuples in r_2 .

The most difficult case to consider is when $R_1 \cap R_2$ is a key for neither ort Chester has $\begin{bmatrix} R_1 \text{ nor } R_2 \end{bmatrix}$. In this case, we use the third stati that each value appears with equal probability. Consider a tuple t of r_1 , and assume $R_1 \cap R_2 = \{A\}$. We estimate that there are $n_{r_1}/V(A,r_2)$ tuples $\frac{1}{2}$ in r_2 with an A value of it Al. So tunle t produce. $\frac{3}{2}$ be relations like the same as $\frac{1}{2}$ and $\frac{1}{2}$ is the same same as $\frac{1}{2}$ is the same same same.

fixed-length
$$
\frac{n_{r_2}}{V(A,r_2)}
$$

tuples in $r_1 \bowtie r_2$. Considering all of the tupl $\frac{1}{2}$ there are the third statistic and assume, as $\frac{1}{2}$ this statistic and as $\frac{1}{2}$

$$
\frac{n_{r_1}n_{r_2}}{V(A,r_2)}
$$

'2 estima These two estimates differ if $V(A, r_1) \neq V(A, r_2)$. If this situation occurs, there are likely to be some dangling tuples that do not participate in the join. Thus, the lower of the two estimates is probably the better one

 12.2 observe that if we reverse that if we reverse that if α in the '_ 92 in the '_ 92

Stu

since

 \sim

expressions.

 $\overline{}$

 course, thecontaining

indices

influence

value on

that corresponds

^physical clustering

and clustering indices allows a clustering indices allows a \sim

and the order in which operations are to be performed. Of

we estimate the cost of ^a strategy that involves the use of; indices.

us to take advantage . of

strategy for processing a query is alleged and access plan for the α ^plan includes not only the relational Operations to be performed the indices to be used and the order in which tuples are to be '

real-world

rI have few

dangling

 itis

 \mathbf{w}_i

is

information

branch-name ⁼

balance **x**

per node. This means that the

The above estimate of join size may be too high if the $V(A,r_1)$ A values in r_1 have few values in common with the $V(A,r_2)$ A values in r_2 . However, it is unlikely that our estimate will be very far off in practice since dangling tuples are likely to be only a small fraction of the tuples in a real-world relation. If dangling tuples appear frequently, then a correction factor could be applied to our estimates.

If we wish to maintain accurate statistics, then every time a relation is modified, it is necessary also to update the statistics. This is a subsident amount of overhead. Therefore, most systems do not update the sta a,.. above estimate of joint and provided and provided and provided and provided and provided and provided and prov
The formal of the provided and p juery processing strategy may not be accurate. However, if the interval and the Assur between the update of the statistics is not too long, the statistics will be outficiently acquired to provide a good estimation of the size of the spanit sufficiently accurate to provide a good estimation of the size of the results $\frac{1}{2}$ of expressions.

relations.

Statistical information about relations is particularly useful when a correction of the correction of the cor

a correction are provided to the correct of the correct of the correct of the correct of the correc be a several indices are available to assist in the processin shall see in Section 9.4. $\frac{1}{2}$ $\frac{1}{2}$ $\frac{1}{2}$ $\frac{1}{2}$ $\frac{1}{2}$ $\frac{1}{2}$ $\frac{1}{2}$ accouirt-nmnlxr

$\frac{3}{10}$ **9.4 Estimation of Costs of Access** every modification of the updated during periods and updated are updated are updated and updated are updated by λ . The updated during period of the updated during periods of the updated during periods of the updated dur ا
أو l
I

The cost estimates we have considered for relational algebra expressions $\frac{1}{2}$ did not consider the affects of indices and hash functions on the co evaluating an expression. The presence of these structures, however, has a significant influence on the choice of a query-processing strategy. V4,.-."

- Indices and hash functions allow fast access to records containing a specific value on the index key. .
.
. V(branclr-tmme. deposit) ⁼
- Estimate of a file in an order corresponding closely to priysical other. It all moex allows the reconsidered for relations of the affects of indices and hash functions of indices of indices and hash functions of indices and hash functions of α indices and α an expression. The physical entity index. Clustering indices allow us to take advantage
Structure of the physical clustering of records into blocks of the physical clustering of records into blocks. the records of a file to $\frac{1}{2}$. physical order. If an index allows the records of a file to be read in an

Find the the state and has hadden for the relational operations by being a the containing and the containing to record and the containing to the process to be purely a the containing to the indices to be used and the order accessed and the order in which operations are to be performed. n *access pla* query. A plan includes not only the relational operations to be performed and 5 leat nodes. With this number of leaf nodes, the entire tree has a but also the indices to be used and the order in which tuples are to be

 $\frac{1}{2}$ Of course, the use of indices imposes the overhead of access to those read to take these blocks containing the index. We need to take these blocks accesses
account when we estimate the cost of a strategy that involves the u account when we estimate the cost of a strategy that involves the use of $\frac{1}{2}$ indices. t the physical order of r records, we call that t records, we call that t and t that t are t that t is a call that t is a call

In this section, we consider queries involving only one relation. We use the selection predicate to guide us in the choice of the best index to
use in processine the query use in processing the query.

As an example of the estimation of the cost of a query using indices assume that we are processing the query:

select account-number

Section 9.4

from deposit

where branch-name $=$ "Perryridge" and customer-name $=$ "Williams"
and balance > 1000

predicate that we have the following statistical in the choice of the best index to the **relation**.

- $\begin{array}{l}\n\bullet 20 \text{ tuples of } \text{deposit} \text{ fit in one block.} \\
\bullet 20 \text{ tuples of } \text{deposit} \text{ fit in one block.}\n\end{array}$
	- \bullet V(branch-name, deposit) = 50.
	- \bullet V(customer-name, deposit) = 200.
	- \bullet V(balance, deposit) = 5000.

 to block reads are required to read' the deposit tuples. in addition. blocks must be read. Assume that the B+-tree index stores 2014.
Assume the B+-tree index stores 2014

must have between 3 and 3 ⁵ leaf nodes. With this number of leaf nodes, the entire tree has ^a of 2, so at most ² index blocks must be read. Thus the above

• The deposit relation has 10,000 tuples.

n the cost of $\frac{1}{2}$ is the following indices exist of vertex $\frac{1}{2}$ - $\frac{1}{2}$. Let us assume that the following indices exist of

-
- \bullet A monclustering, B⁺-tree index for customer-name.

As before, we shall make the simplifying assumption that values are distributed uniformly.

Since $V(branch-name, deposit) = 50$, we expect that $10000/50 = 200$ tuples of the *deposit* relation pertain to the Perryridge branch. If we use the index on *branch-name*, we will need to read these 200 tuples and check each one that the following indices exist of the where
 $200/20 = 10$ block reads are: $200/20 = 10$ block reads are required to read the *deposit* tuples. In addition, several index blocks must be read. Assume the B^+ -tree index stores 20 pointers per node. This means that the B^+ -tree index must have b $\frac{1}{\sqrt{2}}$ strategy requires 1

the accesses into $\frac{4}{3}$ accesses as follows. Since V (custome
ves the use of $\frac{5}{3}$ = 0000/200 = 50 tuples of the deposit $10000/200 = 50$ tuples of the *deposit* relation pertain to Williams. However, $\frac{3}{2}$ since the index for *customer-name* is noncluster
 $\frac{3}{2}$ If we use the index for customer-name, we estimate the number of block just

 $p = \frac{1}{2}$

the index \mathbb{Z} index

 $r = \frac{1}{\sqrt{2}}$

customer-name

"l"erryridge."

likely

Estimation

cost, in

 \Box

 Join Strategies estimated

expression. $\frac{1}{2}$ have seen

have

names, the

to

⁼

that if both

Without the

total of ²⁰²

toileustarting

Since

expression, it may be a set of \mathbb{R}^n

large number of strategies to

^a final choice of query-processing

Chapter 9

block read will be required for each tuple. Thus, 50 block reads are required, just to read the deposit tuples. Let us assume that 20 pointers fit into one node of the B⁺-tree index for customer-name. Since there are 200 customer names, the tree has between 11 and 20 leaf nodes. So, as was the case for the other B⁺-tree index, the index for customer-name has a depth of 2 and 2 block accesses are required to read the necessary index blocks. Therefore, this strategy requires a total of 52 block reads. We conclude that

The conduction of the conduction of the conduction of the conduct of the read without the dustering property, our first strategy would have required 200 block accesses to read the data plus 2 index block $\frac{1}{2}$ accesses for a total of 202 block reads. However, because of the clustering tree trees to the tranch-name index. it is actually less expense example to use the branch-name index.

and 2 block accessed to require the prediction of the necessary index blocks. The necessary indices and the necessary indices and the prediction of the matter of $\frac{1}{2}$ $balance > 1000$ as a starting point for a query processing strategy for two. reasons: $T_{\text{c}q}$ the cost of the sole purpose of $\frac{1}{2}$ the sole purpose of $\frac{1}{2}$ the sole purpose of processing $\frac{1}{2}$

\bullet There is no index for balance.

indices were nondustering, we would prefer to

 \bullet The selection prodict is an initiative involved. clustering in the clustering control of the control of th $\begin{array}{ll}\n\text{First, a specific set of matrices are not a number involves a greater.}\n\text{In general, equations are more selected, it is not a negative product of the product.}\n\end{array}$ predicates. Since we have an equality predicate available to the clustering such a predicate likely to select fewer tuples. use the branch-name index.
The branch-name index. .
r than'' compariso

Extimation of the cost of access using indices allows us to complete cost, in terms of block accesses, of a plan. For a g
algebra expression, it may be possible to formulate sever pain.t for the cost of access using indices allows us to estimate the

complete cost, in terms of block accesses, of a plan. For a given relational algebra expression, it may be possible to formulate several plans. The access plan selection phase of a query optimizer chooses the best plan for a given expression. VS
: OI
: SS estimate the \int

 $\frac{1}{\frac{1}{2}}$ Biven expression.
We have seen that different plans may have significant differences in cost. It is possible that a relational algebra expression for which a good plan exists may be preferable to an apparently more efficient algebra pair back any be preferant to an apparently more emittent algebra
expression for which only inferior plans exist. Thus, it is often worthwhile to select fewer tuples. Thus, it is once worldwide
for a large number of strategies to be evaluated down to the access plan
level before a final choice of query-processing strategy is made level before a final choice of query-processing strategy is made. tl inaccessibility of the second state of the s

9.5 Inin Strategies be possible to the search plans.

exists may be preferable to an apparently more efficient algebra expression \mathbb{R}^n for which only inferior plans exist. The sets \mathcal{L}

 the size of the result of ^a relational algebra expression . : involving^a natural join. In this section, we apply our techniques for , 'g

the cost of processing ^a query to the problem

 $\frac{1}{2}$ Earlier, we estimated the size of the result of a relational algebra expression $\frac{1}{2}$ end estimating the cost of processing a query to the problem of estimating the that ^a relational algebra expression \mathbf{v} which a good \mathbf{v} $\frac{20}{100}$ techniques for $\frac{10}{100}$

cost of processing a join. We shall see that several factors influence the selection of an optimal strategy:

• The physical order of tuples in a relation.

- . The presence of indices and the type of index (clustering or nonclustering).
- it is preferable to use the index for branch-name.

Observe that if both indices were nonclustering, we would prefer to

use the index for customer-name since we expect only 50 tuples with

a join. We shall see that see the see

and assume that we have no indices whatsoever. Let:

 \bullet $n_{density} = 10,000$.

Section 9.5

one query.

9.5.1 Simple Iteration

deptite the set of willing to the set of the
International contract the set of pair of tuples t_1 in *deposit* and t_2 in *customer*. Thus, we examine 10000 + 200 = 2000000 pairs of tuples.
If we execute this query cleverly, we can reduce the number of block

accesses significantly. Suppose that we use the procedure of Figure 9.1 for $\frac{d}{d}$ as many 20 tuples of deposit fit in one block, then reading deposit requires 10000/20 = 500 block accesses.

 \mathbf{t} to create an index, we must examine every possible every possible every possible every possible every possible. $\frac{1}{2}$ for each tuple d in deposit do

begin

Suppose

tuple ^c in customer do

tuple ^d in deposit do

begin

for each tuple c in customer do begin

 $\frac{1}{2}$ begin

test pair (d,c) to see if a tuple shoul as $10,000$ block accesses. However, if the tuples of deposit are $10,000$ block and tuples of deposits are $10,000$

Figure 9.1 Procedure for computing join.

 $\frac{1}{2}$

(0::th

d

read
L

that we were well as

read

 \equiv

for each block Bd of deposit do begin for each block Bc of customer do begin for each tuple b in Bd do begin for each tuple c in Bc do begin From the pair (b, c) to see if a tuple
should be added to the result and customer and continuous contin black Bond of damage of the do was arbitrary. If the relations of the outer sexual used customer as the outer loop effect of the outer loop effect end and the contract of the co help and black 8 f the inner loop, the inner slightly lower (5010 block access in which neither relation fire in the derivatio

the contract of gure 9.2 Procedure to compute *deposit* ∞ customer.

 tuple ^b ^m w
sugge customer tuples. As was the case for *deposit*, we can reduce the required
number of accesses significantly if we store the *customer* tuples together
physically. If we assume that 20 *customer* tuples fit in one block, th en and the state of the state of

end of the state o
 number of accesses significantly if we store the customer tuples together
physically. If we assume that 20 customer tuples fit in one block, then only
10 accesses are required to read the entire customer relation. Thus, on en and the corresponding tuples (if any) of the index are readers are in sorted order, tuples with the secretistic development of the entire customer only 100,000 block accesses are needed to process the query.
9.5.2 Block-Oriented Iteration

per-block basis rather than a per-tuple basis. Again, assuming that deposit and the entire customer relation fit in main memory. The algorithm of Figure 9.3 tuples are stored together physically and that customer tuples are stored
together physically, we can use the procedure of Figure 9.2 to compute
deposit ∞ customer. This procedure performs the join by considering an
en together physically, we can use the procedure of Figure 9.2 to compute number of accesses significant ac deposit in This procedure procedure or rigure 9.2 to compute
entire block of *deposit* tuples at once. We still must read the entire deposit
relation at a cost of 500 accesses. However, instead of reading the customer perf physically. The entire that $\frac{1}{2}$ is the set of $\frac{1}{2}$ we assume that $\frac{1}{2}$ is the set of $\frac{1}{2}$ is the set of relation once for each *tuple* of *deposit*, we read the customer relation once for
each *block* of *deposit*. Since there are 500 blocks of *deposit* tuples and 10
blocks of customer tuples, reading customer once for ever blocks of customer tuples, reading customer once for every block of deposit FREE ALLOTS IN ORDER TO JUNEAU A SUPPRESS US ONCE THE SUPPRESS US OF A CHAPTER CONSIDER THE VIDEO COMPUTER CO tuples requires $10 \times 500 = 5000$ block accesses. Thus, the total cost in terms of block accesses is 5500 accesses (5000 accesses to *customer* blocks plus 500 accesses to *deposit* blocks). Clearly, this is a significant improvement over the number of accesses that were necessary for our initia tuples requires $10 \times 500 = 5000$ block accesses. Thus, the total cost in

terms of block accesses is 5500 accesses (5000 accesses. Thus, the total cost in

plus 500 accesses to *deposit* blocks). Clearly, this is a signif

Our choice of *deposit* for the outer loop and customer for the inner loop was arbitrary. If we had used customer as the relation for the inner loop
and *deposit* for the inner loop the earts of relation for the outer loop and *deposit* for the inner loop, the cost of our final strategy would have
been slightly lower (5010 block accessor). See The cost of our final strategy would have been slightly lower (5010 block accesses). See Exercise 9.10 for a derivation
of these costs.

A major advantage to the use of the smaller relation (customer) in the Finally advantage to the use of the smaller relation (customer) in the
inner loop is that it may be possible to store the entire relation in main
memory temporarily. This speeds query presented in the small Hence the memory temporarily. This speeds query processing significantly since it is
necessary to read the inner loop relation only once. If *customer* is indeed
small enough to fit in main memory, out strategy requires on to read *deposit* plus 10 blocks to read customer for a total of only 510 block accesses.
9.5.3 Merge-Ioin

Section 9.5

In those case
possible to pr possible to process the join efficiently if both relations happen-to be stored rigure 9.2 Procedure to compute deposit ∞ customer.

We read each tuple of customer once for each tuple of deposit. This

suggests that we read each tuple of customer and the smaller of the interest of the interest of We read each tuple of *customer* once for each tuple of *deposit*. This deposit are sorted by customer inner a suggests that we read each tuple of *customer* 10,000 times. Since only one could make as many as 2,000,000 acc We read each tuple of customer once for each tuple of deposit. This and the memory of the speeds that we read each tuple of customer 10,000 times. Since the speedies of the speeds of the first tuple of the respective of c relations. As the algorithm proceeds, the pointers move through the relation. A group of tuples of one relation with the same value on the join attributes is read. Then the corresponding tuples (if any) of the other relation are read. Since the relations are in sorted order, tuples with the it is accesses are required to read the entire customer relation. Thus, only 10
accesses per tuple of deposit rather than 200 are required. This implies that
only 100,000 block accesses are needed to process the query.
The to read, each tuple only once. In the case in which the tuples of the relations are stored together physically, this algorithm allows us to compute the join by reading each block exactly once. For our example of *deposit* relations are stored together physically, this algorithm allows us to \equiv

A major savings in block accesses results if we process the relations on a

per-block both customer and the interval of the earlier join method we pre-block accesses. This is as

the earlier join method we pre-block the earlier join method we presented for the special case in which the

suffices to keep all tuples with the same value for the join attributes in main memory. This is usually feasible even if both relations are large. A disadvantage of the merge-join method is the requirement that both relati

disadvantage Of the merge-[om

m

method '5 the "Winnie" at a cost of 500 accesses. However, instead of reading the customer ', reiations be worthwhile to sort the l. However, it may be word physically. However, it may be word to sort the l. However, it ma I for the complete that is a strong of the complete t

 $\begin{equation} \begin{aligned} \mathcal{A} \end{aligned}$

improvement over the number of accesses that were necessary for our and the second and index. The simple strategy of Figure 9.1 is more efficient if an individual memory. The algorithment of accesses that were necessary fo relations being joined. In such a case, we may consider a join strategy that $\frac{1}{3}$ index exists on customer for customer-name. Given a tuple d in dep

 $\frac{1}{2}$ customer on customer on the deposit \mathcal{C} . \mathcal{C} and \mathcal{C} . \mathcal{C} requires ¹⁰ ⁵⁰⁰ ⁼ ⁵⁰⁰⁰ blodt accesses. Thus, the total cost in ' Fr tl III ^I . . ⁵⁰⁰⁰ ^l equen y, the ^pin attributes forma search key for an index as one of the ' l I is tell th
The ' l I is tell the ' l I is tell th accesstes '2 550.? alclcesses (Cl access? of??? Pilimnt . .- ~- relations being joined. In such ^a case, we may consider ^a join strategy that :5! 'i ^I ii us accesses ho 370;; focks). 9;: 2" ^S ^l 3:; om' uses such an index. The simple strategy of Figure 9.1 is more effective if an "i n3" . t ; Wig??" over the s
"i n3" : Wig??" over the simple strategy of Figure 9.1 is more effective if an "i n3" . t wig??" over the simpl r o accesses a were necessary ' . 7 index exists on customer-name. Given a tuple d in deposit, it is l, ' ll 1
The customer for customer for customer for customer for customers and in deposit, it is l, ' ll 1: .5, .5, .5,
 . '1" 5 '98) - in2 it '1 state in 2 it '1
" in2 it '1 state in 2 i . The contract of the contract

of deposit tuples at once. We still must read the entire deposit " A still must read the entire deposit " A st
Must read the entire deposit " A still must read the entire deposit " A still must read the entire deposit " A

 $\frac{1}{2}$. The concerns of deposit, we read the customer relation on $\frac{1}{2}$

begin

while ($pc \neq null$) do

 $s_{c} := \{t_{c}\};$

 $pd :=$ address of first tuple of *deposit*;

 $pc :=$ address of first tuple of customer;

 I_c := tuple to which pc points;

316

Chapter 9

of an

might be

However, for

achieved

stored

than

that "deposit and

branch

block in the second second

Section 9.5

Join Strategies 317

Without use of an index, and without special assumptions about the physical storage of relations, it was shown that as many as 2 million accesses might be required. Using the index, but without making any
assumptions about phincipal states are index, but without making any assumptions about physical storage, the join can be computed with
assumptions about physical storage, the join can be computed with significantly fewer block accesses. We still need 10,000 accesses to read
deposit. However for each tuple of state in the 10,000 accesses to read deposit. However, for each tubes of deposit only an index lookup is
required. If we assume (as before) that the course of the required. required. If we assume (as before) that $n_{\text{cutoff}} = 200$, and that 20
pointers fit in one block, then this looking required. s to read the *custom*

index, and withouth a cost of 40,000 accesses appears

about that we achieved more efficient strategies only

tuples were stored physically beguing tuples were stored physically together. If this assumption does not hold for the index were stored physically together. If this as
the relations being joined, then the strategy with
mer: can be computed with significant with significant significant accesses.
The computed with significant accesses to read to rea
All needs to read to read to read to rea creation of the index: Even if we create the index for the sole purpose of Teation of the index Even if we treate the index \ddot{r} reprocessing this one query and erase the index \ddot{r} reprocessing that \ddot{r} fewer accesses than if we use the annual \ddot{r} fewer accesses than if we use the strategy of Figure 9.1.

9.5.5 Three-Way Join

(deposit ⁹⁰ customer)

blocks per tuple of $\frac{4}{3}$ and $\frac{1}{3}$ bet us now consider a join involving three relations.

 $\sum_{n=1}^{\infty}$ Assume that $n =$ and $n =$ $\sum_{n=1}^{\infty}$ **the relations being in the relations being in the strategy of the strategy of the strategy we just present pre**
(a) do is the strategy of the strategy we have a choice of strategy for isin or t^* a choice of which join to compute first. There at the consider. We shall analyze several of them if the exercises.

> **• Strategy 1.** Let us first compute the join *(deposit va customer)* using one of the strategies we presented above. Since *customer-name* is a key for *customer*, we know that the result of this join has at most 10,000 t the number of tup

$branch \bowtie (deposit \bowtie customer)$

by considering each tuple *t* of (deposit M customer) and looking up the tuple in branch with a branch-name value of *f*[branch-name]. Since branch-name is a key for branch, we know that we must examine only one to considering the considering $\frac{1}{3}$ by considering them to considering them to $\frac{1}{3}$ by considering them to $\frac{1}{3}$ tuple in *branch* tuple in *branch* with a *branch-name* value of *t*[branch-name]. Since branch-name is a key for branch, we know that we must examine only one branch tuple for each of the 10,000 tuples in (deposit \bowtie customer). The td 2 tuple to which Pd points; (the number of tuples in deposit). If we build an index on branch for 'index on bra

set pc to point to next tuple of customer: while $(\text{not } \text{done})$ do

begin;
 $t_n := \text{tuple to which } \text{no points:}$ r' if t_r [customer-name] = t_r [customer-name] \mathbf{u} $S_i := S_c \cup \{t_c\}$; tuple to which performance $\mathcal{O}_\mathcal{F}$.4.e.'ks..;__;vend deposit.point to next tuple of customer;
point tuple of customer;
the next tuple of customer;
the customer; $t_d :=$ tuple to which pd points; set pd to point to next tuple of deposit; while $(t_d$ [customer-name] $\lt t_c$ [customer-name]) do herein begin
 $t_d :=$ tuple to which pd points; a cost of $40,000$ and $\frac{3}{4}$ and $\frac{1}{4}$ is the must remember of $\frac{1}{4}$ is that $\frac{1}{$ that we ' tuplesset pa to point to nex

end

while $(t_d$ [customer-name] = l
Tarihi for each t in S_t do t $t_{\rm max}$ to which points; $\frac{1}{2}$ compute $t \bowtie t_d$ and add this to result; \mathbf{t} to point tuple of deposition of \mathbf{t} 9.5.5 Three-Way Join end
set pd to point to next tuple of *deposit*: tu
leta which pomts; $t_{\rm end}$

$\frac{1}{2}$ tranch ∞ (deposit ∞ cus Figure 9.3 Merge-join.

no longer necessary to read the entire customer relation. Instead, the index $\frac{1}{2}$ is the set of $\frac{1}{2}$ to longer necessary to read the entire customer relation. Instead, the index $\frac{1}{2}$ branch tuple for each of the 10,000 tuples is used to look up tuples in customer for which the customer-name value is exact number of strategies we present above. Since the strategies we presented by the strategies we presented by the result of to point the point to point the control of the control of deposit; customer, we know the control of th

' end. branches and branches are the control of th

9.3

name in the second control of the second contr

branchrome, we can compute it computed by the computer of the computer of the computer of the computer of the c
The computes of the computer o

Merge-join. By constant the constant with the topic tuple in branch With ^a brand-name value of "brunch-mime] Since brandi-

'11:2.tuples intervalue intervalue is exact number of block accesses required by this strategy depends on the c
This strategy depends on the customer-name value is exact number of block accesses required by this strategy d

production and the contract of the contract of

set political politi

done a state and the contract of the

while (pc aé null) dont a é null) dont a

5'. ;=

and the state of the

begin di sebagai kecamatan di sebagai kecamatan di sebagai kecamatan di sebagai kecamatan di sebagai kecamatan
Kecamatan di sebagai kecamatan di sebagai kecamatan di sebagai kecamatan di sebagai kecamatan di sebagai kecam

ende de la provincia de la pro
En 1930, en 1930, en

set particularly be a set of the set of the
Set of the set of the
Se

to the contract of the contrac d

 $\frac{f(x)}{x} = \frac{f(x)}{x}$ and $\frac{f(x)}{x} = \frac{f(x)}{x}$ and $\frac{f(x)}{x} = \frac{f(x)}{x}$

begin

 $\ddot{\bullet}$ \cdot

 $\sigma_{\rm{min}}$

while the

 :=*<u>Italian</u>* فاد

Therefore, most

projections early.

statistifi for each

FST-IBM000884

branch is stored physically. Several exercises examine the costs of various possibilities.

- Strategy 2. Compute the join without constructing any indices at all. This requires checking $50 \cdot 10000 \cdot 200$ possibilities, a total of 100,000,000.
- · Strategy 3. Instead of performing two joins, we perform the pair of

On branch for branch-name.

On customer for customer-name.

corresponding tuples in customer and the corresponding tuples in branch. Thus, we examine each tuple of deposit exactly once. α book up the α

Strategy 3. represents a form of strategy we have not considered.
 $\frac{1}{\sqrt{2}}$ consider the parameter is first to build the pair of the p $\ddot{\ddot{\bm{x}}}$: strategy 3, it is often possible to perform a join of three relations m the customer form of the settlem when the customer form of the customer costs are started.
The depend on the way in which the relations are stored, the distribution of an opportunity to compute these costs in several examples. Strategy 3 represents a form of strategy we have not considered before. It combines two operations into one special-purpose operation. Using relative costs fromercises
P

example of the Oue.

We have seen only some of the many query processing strategies used in
detables sustants. Most sustant incleaned as low that the set of the details \mathbf{R}^t , database systems. Most systems implement only a few strategies and, as a result, the number of strategies to be considered by the query optimizer is $\frac{3}{2}$ imited. Other systems consider a large number of strategies. For each $\frac{3}{2}$ to strategy a cost estimate is computed. strategy a cost estimate is computed. usuallybased on the relational algebra. In the process of generating the

 $\frac{1}{2}$ in order to simplify the strategy selection task, a query may be split columns, and the presence of indices. This not only simplifies that age selection but also and into several subqueries. This not only simplifies that the yelection but also allows the query optimizer to recognize cases whe \mathbb{R}^N allows the query optimizer to recognize cases where a particular sum Structure of the Structure of the design of the Query optimizing phase and in the Query optimizing phase and i only some of the many subexpressions in many optimizing \mathbf{S} compilers for programming languages. ing phase and in the \ch subqueries $\ddot{\ddot{\mathbf{r}}}$ references first step r relational algebra expression \mathcal{L}

 \mathbf{C} learly, examination of the query for common subquerie estimation of the cost of a large number of strategies impose a substanti $-\frac{1}{2}$ continuous extension of the cost of $\frac{1}{2}$ contribution of $\frac{1}{2}$ coverhead on query processing. However, $\frac{1}{2}$ optimization is usually more than offset by the savings at query execution

time. Therefore, most commercial systems include relatively sophisticated optimizers. The bibliographic notes give references to descriptions of query optimizers of actual database systems.

9.7 Summary

There are a large number of possible strategies for processing a query, especially if the query is complex. Strategy selection can be done using information available in main memory, with little or no disk accesses. The actual execution of the query will involve many accesses to disk. Since the transfer of data from disk is slow relative to the speed of main memory ita irom alsk is slow relative to
ral processor of the computer su bibliographic notes and the central processor of the computer systems of the computer systems of special systems.

the answer. It is the responsibility of the system to transform the query as entered by the user into an equivalent query which can be computed more instead₂-it $\frac{1}{2}$ strategy for processing a query is called *query* of

The first action the system must take on a query is to translate the query into its internal form which (for relational database systems) is usually based on the relational algebra. In the process of generating the internal form of the query, the parser checks the syntax of the user's query, verifies that the relation names appearing in the query are names of relation in the database, etc. If the query was expressed in terms of a view, the parser replaces all references to the view name with the relational $\frac{1}{2}$ algebra expression to compute the view assession

a relational algebra expression tha Each relational algebra expression represents a particular sequence of \mathbb{Z}_2^3 operations. The first step in selecting a query-port t and is efficient to execute. There are a num transforming relational algebra queries, including:

- \bullet Perform selection operations as early as possible.
- relationbut also are names of 1998.
ubquery was experienced in the database of 1998.

strategy we choose for a query depends upon the relation and the distribution of values within columns. In order to be able to choose a strategy based on reliable information, database systems may to choose a strategy based on reliable information, databased strategy based on reliable information, databased

is effect to execute the are a number of tuples in the relation rules is substantial $\frac{1}{2}$

 we choose for ^a query depends upon the size of each , . ' relation and the distribution of values within columns. In order to be able choose ^a strategy based on reliable information, database systems may store

relation r. These statistics include:

 \bullet The size of a record (tuple) of relation r in bytes (for fixed-length records).

write and the continues.

query. Justify

'N

 :2) u \mathbb{R}

keys ^A

.9.6 again, but now

nonclustering

to de

influence

- r2) ⁼

The first twoproduct.the company of the c
The company of the c
 Statisticalseveralindices de la composición de la compo
La composición de la query-processingdepending to the control of block-oriented merge-join sole purpose of allowingforConsider the following SQL

• The number of distinct values that appear in the relation r for a particular attribute.

The first two statistics allow us to estimate accurately the size of a cartesian product. The third statistic allows us to estimate how many tuples satisfy a simple selection predicate.

Statistical information about relations is particularly useful when several indices are available to assist in the processing of a query. The presence of these structures has a significant influence on the choice of a query-processing strategy.

 T depending on the availability of indices and the form of physical storage in the relations of a particular appear to consider the summer store of a particular summer T a block-oriented join strategy may be advantageous. If the relations are sorted, a merge-join may be desirable. It may be more efficient to sort a station prior to join computation (so as to a
strategy). It may also be advantageous to com the sole purpose of allowing a more efficient join strated

of these structures has ^a significant influence on

 \overline{a}

where

branch-city

"Brooklyn")

efficient relational algebra expression that is equivalent to ' ' this

involving

 $m₁$

 $\overline{}$ to

an

query. Justify

Example 1 on the form of the form of the form of the form of $\frac{1}{3}$. At what point during query processing does optimization occur?

- 9.2 Why is it not desirable to force users to make an explicit choice

outery processing strategy? Are there cases in which it is desired prior processing suaregy: The there cases in which it is desirable
for users to be aware of the costs of competing query processing
strategies? a strategies?
 9.3 Consider the following SQL query for our bank database: query processing strategy? Are there cases in which it is desirable $2\frac{1}{2}$
	-

assist in the processing of ^a query. The

select customer-name from deposit S where (select branch-name
from deposit T where S.customer-name $=$ T.customer-name) contains ey
estroke ')
th (select branch-name
from branch where branch-city = "Brooklyn")

ilent to $\frac{1}{2}$

9.4 Consider the following SQL query for our bank database:

select T.branch-name from branch T. branch S where T *assets* $>$ *S assets* andS.branch-city = "Brooklyn"

na: aigeora ex₁
aboice this query. Justify your choice.

9.5 Show that the following equivalences hold, and explain how they can be applied to improve the efficiency of certain queries:

a. $\sigma_p(r_1 \cup r_2) = \sigma_p(r_1) \cup \sigma_p(r_2)$ b. $\sigma_p(r_1 - r_2) =$

 $\mathcal{C}_1 \cup \mathcal{C}_2 \cup \mathcal{C}_3 = \emptyset$

efficient relational algebra expression that is equivalent to the isomorphism of \mathcal{E}

index. When must we create ^a nondustering index despite

processing strategy?

What are the advantages and disadvantages and disadvantages and disadvantages of hash functions of hash functions

the advantages of a clustering control of a clustering control of a clustering control of a clustering control of

B+'tree indices? How

the choice of ^a query

- 9.6 Consider the relations $r_1(A, B, C)$, $r_2(C, D, E)$, and $r_3(E, F)$, with primary keys A, C, and E respectively. Assume that r_1 has 1000 tuples, r_2 has 1500 tuples and r_3 has 750 tuples. Estimate the size of $r_1 \bowtie r_2 \bowtie r_3$, and give an efficient strategy for computing the join.
- 9.7 Consider the relations $r_1(A, B, C)$, $r_2(C, D, E)$, and $r_3(E, F)$ of Exercise rang 19.6 ap and $V(E,r_3)$ be 100. Assume that r_1 has 1000 tuples, r_2 has 1500 tuples and r_3 has 750 tuples. Estimate the size of $r_1 \boxtimes r_2 \bowtie r_3$, and give an efficient strategy for computing the join.
- $\frac{1}{2}$, and $\frac{1}{2}$ are the chatch size of tompo
1500 tuples and 13 has 1000 tuples. Estimate the size of th nonclustering index. When must we create a nonclustering index despite the advantages of a clustering index?
- 9.9 What are the advantages and disadvantages of hash functions relative to B^+ -tree indices? How might the type of index available influence the choice of a query processing strategy? $\frac{1}{2}$ $\frac{1}{2}$
- $\frac{3}{2}$ 3.10 Recompute the cost of the strategy of $\frac{3}{2}$ and the relation of the filter loop and custhe relation of the inner loop and customer as the relation of the example of Section 9.5.2). Clustering indices may allowfaster access to data than ^a

might the type of index available to the type of index available

FST-IBM000885

 rl has 20,000 tuples. $\frac{1}{2}$ $\frac{1}{2}$

Simple iteration.

Assume

^a progr

selection in System

 the optimal use of available main memory. These papers discuss of the strategies that we presented in this chapter. Wong and Y ossefi Y introduce a technique called decomposition, which is 'n technique called decomposition, which is 'n in the Ingres database system. The Ingres database system is a system of α ingressed in the Ingres decomposition strategy α the third strategy were considered for the three-way joins. [n Ingres.] we have the three-way joins. [n Ingres.] extension of the technique is used to choose a strategy for general \sim queries. The contract of the contract of the contract of the contract of the contract of

 ^R are discussed in more detail in Chapter 15. Ifentire group of queries is considered, it is possible to discover

subset of the evaluations that can be evaluated on the entire group. The enti Finkelstein[1982], and Hall [1976] consider optimization of ^a group of

Estimate

following

0 \longrightarrow

 $\frac{1}{2}$

Bibliographic Contract of the Second State of the Second State of the Second State of the Second State of the the ideas used

al. 1986 \rightarrow 1986 \rightarrow 1986 \rightarrow

 p and $\frac{1}{2}$.

and

lngres

common

access \pm

<u>of the contract of the contra</u>

25 tuples of 'I fit on one block.

 resultson

9.11 Explain the difference between a clustering index and a nonclustering index.

9.12 Let relations $r_1(A, B, C)$ and $r_2(C, D, E)$ have the following properties:

- \bullet r₁ has 20,000 tuples.
- r_2 has 45,000 tuples.
- \bullet 25 tuples of r_1 fit on one block.
- \bullet 30 tuples of r_2 fit on one block.

Estimate the number of block accesses required using each c

following join strategies for $r_1 \bowtie r_2$: following join strategies for $r_1 \bowtie r_2$: '1.m¢ni"tit'l.'t.,.

- · Simple iteration.
- · Block-oriented iteration.

 $\frac{3}{2}$ 9.13 Consi the number of block accesses required using the costs of the 3 strategies of Section $\overline{9.5.5}$ for $\overline{9.5.5}$ 40 tuples of r_3 internal form

Bibliographic Notes

standard programming languages. There are several texts that present optimization from a programming languages point of view, including [Aho $\frac{1}{2}$ et al. 1986], and [Tremblay and Sorenson 1 describes access path selection in System R. Kim [1981, 1982] describe join $f(x) = \frac{1}{2}$ rategies and and Yousseli [1976] introduce a technique called *decomposition*, which is used in the Ingres database system. The Ingres decomposition strategy an extension of this technique is used to d queries. Ingres and System R are discussed in more detail in Chapter 15.

 $\frac{1}{2}$ If an entire group of queries is considered, it is $\frac{1}{2}$ common subexpressions that can be evaluated once for the entire group.³ Emmen change 1982], and Hall [1976] consider opti

 $R_{\rm eff}$ is given μ

queries and the use of common subexpressions. When queries are generated through views, it is often the case that more relations are joined than is necessary to compute the query. A collection of techniques for join minimization have been grouped under the name tableau optimization. The notion of a tableau was introduced by Aho et al. [1979a, 1979c]. Ullman [1982a] and Maier [1983] provide a textbook coverage of tableaux.

Theoretical results on the complexity of the computation of relational algebra operations appear in [Gotlieb 1975], [Pecherer 1975], and [Blasgen ulleu 1270), li
Louent neon [Jarke and Koch 1984].

An actual query processor must translate statements in the query language into an internal form suitable for the analysis we have discussed in this chapter. Parsing query languages differs little from parsing of traditional programming languages. Most compiler texts (including [Aho necessary to compute the computer of techniques for interests in the name tableau optimization of the name tableau optimization of the name tableau optimization have been grouped under the name tableau optimization. The na techniques. A more theoretical presentation of parsing and language translation is given by Aho and Ullman [1972, 1973].

Query processing for distributed database systems use some concepts
from this chapter. Techniques specific to distributed systems appear in an swaren operati'tggsélappear . surve infrom this chapter. I echniques specific to div
Chapter 12 and the bibliographic notes to that
with a relation and the bibliographic notes to that the $\frac{1}{2}$ chapter 12 and the bibliographic notes to that the processor must be query changed.

