

A Logical View of Data

Relational Model

- Enable us to view data logically rather than physically.
- Reminds us of simpler file concept of data storage.
- proposed by E.F.Codd in 1970 based on mathematical concept.
- logical structure between tables.
 - name
 - attribute
- objectives :-
 - to allow a high degree of data independence.
 - to provide substantial grounds for dealing with semantics, consistency and redundancy problems.
 - to enable the expansion of set-oriented data manipulation languages.

Copyright © Ami Yassin Abdul Rahman 2016

Infrastructure University

- prototype relational DBMS System R developed by IBM's San Jose Research Laboratory in US late 1970s.
- designed to prove the practicality of the relational model by providing an implementation of its data structures and operations.

Copyright © Ami Yassin Abdul Rahman 2016

Infrastructure University

Relational Data Structure

Relation - a table with columns and rows.

- only applies to logical structure of the database, not the physical structure.
- example : customer relation is a table that store data about customer.

Attribute - a named column of a relation.

- example : name, icNo, address, gender.

Domain - the set of allowable values for one or more attributes.

- every attribute is defined on a domain.
- domain may be distinct for each attribute, two or more attributes may be defined on the same domain.

Copyright © Ami Yassin Abdul Rahman 2016

Infrastructure University

- example: domain for attribute **city** – city in Malaysia.
- important because it allows the user to define in a central place the meaning and the source of values that attributes can hold.
- defined in data dictionary using DDL.

Tuple - a row of a relation.

- called as **extension/state** of a relation, which changes over time.

Copyright © Ami Yassin Abdul Rahman 2016

Infrastructure University

Degree - the number of attributes in a relation.

- in Branch relation has 7 attributes or degree 7.
- relation with 1 tuple and degree 1 = unary, 2 attributes = binary, 3 attributes = ternary, 4 attribute = quaternary.

Cardinality - the number of tuples in a relation

- changes as tuples are added or deleted.
- is a property of the **extension** of the relation and is determined from the particular instance of the relation at any given moment.

Copyright © Ami Yassin Abdul Rahman 2016

Infrastructure University

Relational Database - a collection of normalized relations with distinct relation names.

Alternative Terminology

Formal terms	Alternative 1	Alternative 2
Relation	Table	File
Tuple	Row	Record
Attribute	Column	Field

Instances of Branch and Staff Relations

Diagram illustrating the relationship between Branch and Staff relations. The Branch relation has attributes: branchNo, street, city, postcode. The Staff relation has attributes: staffNo, fName, lName, position, sex, DOB, salary, branchNo. The branchNo attribute in the Staff relation is a foreign key to the branchNo attribute in the Branch relation. The diagram also shows the cardinality and degree of the relations.

branchNo	street	city	postcode
B005	22 Deer Rd	London	SW1 4EH
B007	16 Argyll St	Aberdeen	AB2 3SU
B003	163 Main St	Glasgow	G11 9QX
B004	32 Manse Rd	Bristol	BS99 1NZ
B002	56 Clover Dr	London	NW10 6EU

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

Examples of Attribute Domains

Attribute	Domain Name	Meaning	Domain Definition
branchNo	BranchNumbers	The set of all possible branch numbers	character: size 4, range B001-B999
street	StreetNames	The set of all street names in Britain	character: size 25
city	CityNames	The set of all city names in Britain	character: size 15
postcode	Postcodes	The set of all postcodes in Britain	character: size 8
sex	Set	The sex of a person	character: size 1, value M or F
DOB	DatesOfBirth	Possible values of staff birth dates	date, range from 1-Jan-20, format dd-mm-yy
salary	Salaries	Possible values of staff salaries	monetary: 7 digits, range 6000.00-40000.00

Properties of Relations

- Relation name** is distinct from all other relation names in relational schema.
- Each **cell of relation** contains exactly one atomic (single) value.
- Each **attribute** has a distinct name.
- Values of an attribute** are all from the same domain.
- Each **tuple** is distinct; there are no duplicate tuples.
- Order of attributes** has no significance.
- Order of tuples** has no significance, theoretically.
- Key** - every table must have a unique key to identify each row.

Relational Keys

Superkey

- An attribute, or set of attributes, that uniquely identifies a tuple within a relation.
- example: staffNo, combination of staffNo and name.

Candidate Key

- Superkey (K) such that no proper subset is a superkey within the relation.
- In each tuple of R, values of K uniquely identify that tuple (**uniqueness**).
- No proper subset of K has the uniqueness property (**irreducibility**).
- example: staffNo, icNo - each key is a candidate key.

Primary Key

- Candidate key selected to identify tuples uniquely within relation.
- selection based on the key semantics and environment of the relation.
- example: matricNo, staffNo etc.
- candidate keys not selected as primary key are called **alternate keys**.

Foreign Key

- Attribute or set of attributes, within one relation that matches candidate key of some (possibly same) relation.
- when an attribute appears in more than one relation, its appearance usually represents a relationship between tuples of the two relations.
- example: branchNo in Staff table.

Composite Key

candidate key consists of more than one attribute.

Table 5.24 Result table for Example 5.24.

clientNo	tName	iName	propertyNo	comment
CR56	Aline	Stewart	PG36	
CR56	Aline	Stewart	PA14	too small
CR56	Aline	Stewart	PG4	
CR62	Mary	Tregear	PA14	no dining room
CR76	John	Kay	PG4	too remote

Secondary key

attribute or set of attribute not candidate key used for retrieval purposes.

example : name

Representing Relational Database Schemas

a database can consists one or more number of relations.

keys for the relation : primary key or foreign key must be underlined.

example :

Customer (custNo, custName, regDate, agentCode)
Agent (agentCode, agentName, phoneNo, salary, area)

Relation

Integrity Rules

refer to reliability, validity and authority towards the data.

data model consists of 2 main parts :-

- manipulative part – defining the types of operation that are allowed on the data.
- set of integrity rules – ensure that the data is accurate.

every attribute has an associated domain, there are constraints (called domain constraints) that form restrictions on the set of values allowed for the attributes of relations.

two principles rules :-

- entity integrity
- referential integrity

Nulls

Represents value for an attribute that is currently unknown or not applicable for tuple.

Deals with incomplete or exceptional data.

Represents the absence of a value and is not the same as zero or spaces, which are values.

example :

Agent

agentCode	agentName	phoneNo	salary	area
C15	Ahmad	03-2934567	2,000.00	Bangi
C20	Cindy	03-5563498	4,000.00	Cheras
C22	Azmi		3,000.00	Kepong
C24	Suzy	03-4596780	2,500.00	Selayang

Entity Integrity

applies to the **primary keys** of base relation.

in a base relation, no attribute of a primary key can be null.

primary key – a minimal identifier that is used to identify tuples uniquely.

if primary key is null, it will lead to anomaly.

anomaly – **wasting of storage spaces that leads to error and inconsistencies of data.**

base relation – **named relation corresponding to an entity in conceptual schema, whose tuples are physically stored in database.**

example null allowed :- list of phone no. for the agents.

Referential Integrity

applies to the **foreign keys** exists in a relation.

if foreign key exists in a relation,

- either foreign key value must match a candidate key value of some tuple in its home relation or
- foreign key value must be wholly null.

example : agentCode is a primary key in Agent relation while in Customer relation as a foreign key.

2 strategies that can be implemented :-

- allow updating of record in a relation that has a foreign key if the primary key in master relation is updated.
- allow the deleting and updating done to the master relation but set all the keys to null.

General Constraints

- Additional rules specified by users or database administrators that define or constraint some aspect of the enterprise.
- known as business rules.
- example : the custNo in Customer table given a range from 10000 - 50000.

Relational Database Operators

- Relational algebra is a theoretical language with operations that works on one or more relations to define another relation without changing the original relations.
- Both operands and results are relations, so output from one operation can become input to another operation.
- Allows expressions to be nested, just as in arithmetic. This property is called closure.
- Five basic operations in relational algebra: Selection, Projection, Cartesian product, Union, and Set Difference.

Selection (or Restriction)

- $\sigma_{\text{predicate}}(R)$
 - Works on a single relation R and defines a relation that contains only those tuples (rows) of R that satisfy the specified condition (*predicate*).
- List all staff with a salary greater than £10,000.

$\sigma_{\text{salary} > 10000}(\text{Staff})$

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003

Projection

- $\Pi_{\text{col1}, \dots, \text{coln}}(R)$
 - Works on a single relation R and defines a relation that contains a vertical subset of R, extracting the values of specified attributes and eliminating duplicates.
- Produce a list of salaries for all staff, showing only staffNo, fName, lName, and salary details.

$\Pi_{\text{staffNo}, \text{fName}, \text{lName}, \text{salary}}(\text{Staff})$

staffNo	fName	lName	salary
SL21	John	White	30000
SG37	Ann	Beech	12000
SG14	David	Ford	18000
SA9	Mary	Howe	9000
SG5	Susan	Brand	24000
SL41	Julie	Lee	9000

Union

- $R \cup S$
 - Union of two relations R and S defines a relation that contains all the tuples of R, or S, or both R and S, duplicate tuples being eliminated.
 - R and S must be union-compatible.
- If R and S have I and J tuples, respectively, union is obtained by concatenating them into one relation with a maximum of (I + J) tuples.
- List all cities where there is either a branch office or a property for rent.

$\Pi_{\text{city}}(\text{Branch}) \cup \Pi_{\text{city}}(\text{PropertyForRent})$

city
London
Aberdeen
Glasgow
Bristol

Set Difference

- $R - S$
 - Defines a relation consisting of the tuples that are in relation R, but not in S.
 - R and S must be union-compatible.
- List all cities where there is a branch office but no properties for rent.

$\Pi_{\text{city}}(\text{Branch}) - \Pi_{\text{city}}(\text{PropertyForRent})$

city
Bristol

Intersection

- $R \cap S$
 - Defines a relation consisting of the set of all tuples that are in both R and S.
 - R and S must be union-compatible.
- Expressed using basic operations:
$$R \cap S = R - (R - S)$$
- List all cities where there is both a branch office and at least one property for rent.

city
Aberdeen
London
Glasgow

Cartesian product

R X S

- Defines a relation that is the concatenation of every tuple of relation R with every tuple of relation S.
- List the names and comments of all clients who have viewed a property for rent.

$(\Pi_{\text{clientNo}}, \text{fName}, \text{iName}(\text{Client})) \times (\Pi_{\text{clientNo}}, \text{propertyNo}, \text{comment}(\text{Viewing}))$

client_id/client	fname	lname	viewing_id/viewing	propertyNo	comment
C-0016	James	Scott	V-0016	PR-14	too small
C-0016	James	Scott	V-0017	PR-15	too small
C-0016	James	Scott	V-0018	PR-16	no driving room
C-0016	James	Scott	V-0019	PR-17	PR-14
C-0016	James	Scott	V-0020	PR-18	too small
C-0016	James	Scott	V-0021	PR-19	PR-15
C-0016	James	Scott	V-0022	PR-20	no driving room
C-0016	James	Scott	V-0023	PR-21	PR-16
C-0016	James	Scott	V-0024	PR-22	PR-17
C-0016	James	Scott	V-0025	PR-23	no view
C-0016	James	Scott	V-0026	PR-24	no view
C-0016	James	Scott	V-0027	PR-25	no view
C-0016	James	Scott	V-0028	PR-26	no view
C-0016	James	Scott	V-0029	PR-27	no view
C-0016	James	Scott	V-0030	PR-28	no view
C-0016	James	Scott	V-0031	PR-29	no view
C-0016	James	Scott	V-0032	PR-30	no view
C-0016	James	Scott	V-0033	PR-31	no driving room
C-0016	James	Scott	V-0034	PR-32	PR-14
C-0016	James	Scott	V-0035	PR-33	PR-15
C-0016	James	Scott	V-0036	PR-34	PR-16
C-0016	James	Scott	V-0037	PR-35	PR-17
C-0016	James	Scott	V-0038	PR-36	PR-18
C-0016	James	Scott	V-0039	PR-37	PR-19
C-0016	James	Scott	V-0040	PR-38	PR-20
C-0016	James	Scott	V-0041	PR-39	PR-21
C-0016	James	Scott	V-0042	PR-40	PR-22
C-0016	James	Scott	V-0043	PR-41	PR-23
C-0016	James	Scott	V-0044	PR-42	PR-24
C-0016	James	Scott	V-0045	PR-43	PR-25
C-0016	James	Scott	V-0046	PR-44	PR-26
C-0016	James	Scott	V-0047	PR-45	PR-27
C-0016	James	Scott	V-0048	PR-46	PR-28
C-0016	James	Scott	V-0049	PR-47	PR-29
C-0016	James	Scott	V-0050	PR-48	PR-30
C-0016	James	Scott	V-0051	PR-49	PR-31
C-0016	James	Scott	V-0052	PR-50	PR-32
C-0016	James	Scott	V-0053	PR-51	PR-33
C-0016	James	Scott	V-0054	PR-52	PR-34
C-0016	James	Scott	V-0055	PR-53	PR-35
C-0016	James	Scott	V-0056	PR-54	PR-36
C-0016	James	Scott	V-0057	PR-55	PR-37
C-0016	James	Scott	V-0058	PR-56	PR-38
C-0016	James	Scott	V-0059	PR-57	PR-39
C-0016	James	Scott	V-0060	PR-58	PR-40
C-0016	James	Scott	V-0061	PR-59	PR-41
C-0016	James	Scott	V-0062	PR-60	PR-42
C-0016	James	Scott	V-0063	PR-61	PR-43
C-0016	James	Scott	V-0064	PR-62	PR-44
C-0016	James	Scott	V-0065	PR-63	PR-45
C-0016	James	Scott	V-0066	PR-64	PR-46
C-0016	James	Scott	V-0067	PR-65	PR-47
C-0016	James	Scott	V-0068	PR-66	PR-48
C-0016	James	Scott	V-0069	PR-67	PR-49
C-0016	James	Scott	V-0070	PR-68	PR-50
C-0016	James	Scott	V-0071	PR-69	PR-51
C-0016	James	Scott	V-0072	PR-70	PR-52
C-0016	James	Scott	V-0073	PR-71	PR-53
C-0016	James	Scott	V-0074	PR-72	PR-54
C-0016	James	Scott	V-0075	PR-73	PR-55
C-0016	James	Scott	V-0076	PR-74	PR-56
C-0016	James	Scott	V-0077	PR-75	PR-57
C-0016	James	Scott	V-0078	PR-76	PR-58
C-0016	James	Scott	V-0079	PR-77	PR-59
C-0016	James	Scott	V-0080	PR-78	PR-60
C-0016	James	Scott	V-0081	PR-79	PR-61
C-0016	James	Scott	V-0082	PR-80	PR-62
C-0016	James	Scott			

Example - Cartesian product and Selection

- Use selection operation to extract those tuples where $\text{Client.clientNo} = \text{Viewing.clientNo}$.

$$\sigma_{\text{Client.clientNo} = \text{Viewing.clientNo}}((\text{O}_{\text{clientNo}, \text{fName}, \text{IName}}(\text{Client})) \times (\text{O}_{\text{clientNo}, \text{propertyNo}, \text{comment}}(\text{Viewing})))$$

client.clientNo	fName	IName	Viewing.clientNo	propertyNo	comment
CR76	John	Kay	CR76	PG4	too remote
CR56	Aline	Stewart	CR56	PA14	too small
CR56	Aline	Stewart	CR56	PG4	
CR56	Aline	Stewart	CR56	PG36	
CR62	Mary	Tregear	CR62	PA14	no dining room

- ◆ Cartesian product and Selection can be reduced to a single operation called a *Join*.

Infrastructure University

Join Operations

- Join is a derivative of Cartesian product.
- Equivalent to performing a Selection, using join predicate as selection formula, over Cartesian product of the two operand relations.
- One of the most difficult operations to implement efficiently in an RDBMS and one reason why RDBMSs have intrinsic performance problems.
- Various forms of join operation
 - Theta join
 - Equijoin (a particular type of Theta join)
 - Natural join
 - Outer join
 - Semijoin

Theta join (θ -join)

- $R \bowtie_{F} S$
- Defines a relation that contains tuples satisfying the predicate F from the Cartesian product of R and S .
- The predicate F is of the form $R.a_i \theta S.b_j$ where θ may be one of the comparison operators ($<$, \leq , $>$, \geq , $=$, \neq).

Theta join (θ -join)

- Can rewrite Theta join using basic Selection and Cartesian product operations.

$$R \bowtie_F S = \sigma_F(R \times S)$$

- Degree of a Theta join is sum of degrees of the operand relations R and S. If predicate F contains only equality ($=$), the term *Equijoin* is used.

Example - Equijoin

- List the names and comments of all clients who have viewed a property for rent.

$(\Pi_{\text{clientNo}, \text{fName}, \text{lName}}(\text{Client})) \bowtie_{\text{Client.clientNo} = \text{Viewing.clientNo}} (\Pi_{\text{clientNo}, \text{propertyNo}, \text{comment}}(\text{Viewing}))$

client.clientNo	fName	lName	Viewing.clientNo	propertyNo	comment
CR76	John	Kay	CR76	PG4	too remote
CR56	Aline	Stewart	CR56	PA14	too small
CR56	Aline	Stewart	CR56	PG4	
CR56	Aline	Stewart	CR56	PG36	
CR62	Mary	Tregear	CR62	PA14	no dining room

Natural join

R \bowtie S

- An Equijoin of the two relations R and S over all common attributes x . One occurrence of each common attribute is eliminated from the result.
- List the names and comments of all clients who have viewed a property for rent.

$(\Pi_{\text{clientNo}, \text{fName}, \text{lName}}(\text{Client}))$

$(\Pi_{\text{clientNo}, \text{propertyNo}, \text{comment}}(\text{Viewing}))$

clientNo	fName	lName	propertyNo	comment
CR76	John	Kay	PG4	too remote
CR56	Aline	Stewart	PA14	too small
CR56	Aline	Stewart	PG4	
CR56	Aline	Stewart	PG36	
CR62	Mary	Tregear	PA14	no dining room

Aggregate Operations

- $\mathfrak{F}_{\text{AL}}(R)$
 - Applies aggregate function list, AL, to R to define a relation over the aggregate list.
 - AL contains one or more ($\langle \text{aggregate_function}, \langle \text{attribute} \rangle \rangle$) pairs.
- Main aggregate functions are: COUNT, SUM, AVG, MIN, and MAX.

Example – Aggregate Operations

- How many properties cost more than £350 per month to rent?

$\rho_R(\text{myCount}) \mathfrak{F}_{\text{COUNT}} \text{propertyNo} (\sigma_{\text{rent} > 350} (\text{PropertyForRent}))$

myCount
5

(a)

The Data Dictionary and System Catalog

Data dictionary

- Used to provide detailed accounting of all tables found within the user/designer-created database
- Contains (at least) all the attribute names and characteristics for each table in the system
- Contains metadata—data about data
- Sometimes described as “the database designer’s database” because it records the design decisions about tables and their structures

A Sample Data Dictionary

TABLE 3.6 A SAMPLE DATA DICTIONARY

TABLE NAME	ATTRIBUTE NAME	CONTENTS	TYPE	FORMAT	RANGE	REQUIRED	PK	FK
CUSTOMER	CUS_CODE	Customer acct. code	CHAR(5)	99999	0000-99999	Y	PK	
	CUS_LNAME	Customer last name	VARCHAR(20)	Xxxxxxx	100-999	Y		
	CUS_FNAME	Customer first name	VARCHAR(20)	Xxxxxxx		Y		
	CUS_INITIAL	Customer initial	CHAR(1)	X				
	CUS_RENEW_DATE	Customer insurance renewal date	DATE	dd-mm-yyyy				
AGENT	AGENT_CODE	Agent code	CHAR(3)	999		Y	PK	
	AGENT_AREACODE	Agent area code	CHAR(3)	999	000-9,999,999.99	Y		
	AGENT_PHONE	Agent telephone number	CHAR(10)	999-9999		Y		
	AGENT_LNAME	Agent last name	VARCHAR(20)	Xxxxxxx		Y		
	AGENT_YTD_SALES	Agent year-to-date sales	NUMBER(9,2)	9,999,999.99		Y		


FK = Foreign key

PK = Primary key

CHAR = Fixed character length data (1-255 characters)

VARCHAR = Variable character length data (1-2,000 characters)

NUMBER = Numeric data. NUMBER(9,2) is used to specify numbers with two decimal places and up to nine digits, including the decimal places. Some RDBMSs permit the use of a MONEY or CURRENCY data type.



■ System catalog

- Contains metadata
- Detailed system data dictionary that describes all objects within the database
- Terms “system catalog” and “data dictionary” are often used interchangeably
- Can be queried just like any user/designer-created table

Copyright © 2019, Pearson Education, Inc. All rights reserved.

