MONASH University

MONASH
INFORMATION
TECHNOLOGY

Week 2 – Relational Data Model

FIT2094 Database

GROUP OF EIGHT AUSTRALIA

Overview

- Relational Model
- Relational Algebra

The Relational Model

- Introduced by CODD in 1970 - the fundamental basis for relational DBMS's
- Basic structure is the mathematical concept of a RELATION mapped to the 'concept' of a table (tabular representation of relation)
  - Relation - abstract object
  - Table - pictorial representation
  - Storage structure - "real thing" - eg. isam file
- Relational Model Terminology
  - DOMAIN - set of atomic (indivisible) values
    - specify
      - name
      - data type
      - data format
- Examples:
  - customer_number domain - 5 character string of the form xxxdd
  - name domain - 20 character string
  - address domain - 30 character string containing street, town & postcode
  - credit_limit domain - money in the range $1,000 to $99,999

A Relation

- A relation consists of two parts
  - heading
  - body
- Relation Heading
  - Also called Relational **Schema** consists of a fixed set of attributes
    - R (A1,A2.....An)
      - R = relation name, Ai = attribute i
  - Each attribute corresponds to one underlying domain:
    - Customer relation heading:
      - CUSTOMER (custno, custname, custadd, credlimit)
        - » dom(custno) = customer_number
        - » dom(custname) = name
        - » dom(custadd) = address
        - » dom(credlimit) = credit_limit

custno	custname	custadd	credlimit

Relation Body

- Relation Body
  - Also called Relation Instance (state)
    - r(R) = {t1, t2, t3, ..., tm}
    - consists of a time-varying set of n-tuples
      - Relation R consists of tuples t1, t2, t3 .. tm
      - m = number of tuples = **relation cardinality**
    - each n-tuple is an ordered list of n values
    - t = < v1, v2, ... vn>
      - n = number of values in tuple (no of attributes) = **relation degree**
  - In the tabular representation:
    - Relation heading ⇨ column headings
    - Relation body ⇨ set of data rows

custno	custname	custadd	credlimit
SMI13	SMITH	Wide Rd, Clayton, 3168	2000
JON44	JONES	Narrow St, Clayton, 3168	10000
BRO23	BROWN	Here Rd, Clayton, 3168	10000

Relation Properties

- No duplicate tuples
  - by definition sets do not contain duplicate elements
    - hence tuples are unique
- Tuples are unordered within a relation
  - by definition sets are not ordered
    - hence tuples can only be accessed by content
- No ordering of attributes within a tuple
  - by definition sets are not ordered

Relation Properties cont'd

- Tuple values are atomic - cannot be divided
    - EMPLOYEE(eid, ename, departno, dependants)
        - not allowed: dependants (depname, depage) multivalued
    - hence no multivalued (repeating) attributes allowed, called the first normal form rule
- COMPARE with tabular representation
    - normally nothing to prevent duplicate rows
    - rows are ordered
    - columns are ordered
    - tables and relations are not the same 'thing'

surname	firstname	degree	DOB
Black	Sam	BBIS	02-02-1996
Brown	Jane	BITS	01-01-1995
Chen	Chan	BITS	09-02-1996
Grey	Maria	BCS	15-12-1995
Indigo	Jose	BITS	28-10-1995
Black	Jet	BCS	13-05-1996
Chen	Maria	BITS	31-08-1995

- **Functional Dependency:**
  - A set of attributes X functionally determines an attribute Y if, and only if, for each X value there is exactly one Y value in the relation. It is denoted as X → Y.
- For example, given the data above:
  - firstname, surname → degree
    - *but*
  - firstname → degree does not hold
  - What about: degree → firstname, surname?

Relational Keys

- A candidate key K of a relation R is an attribute or set of attributes which exhibits the following properties:
  - No two tuples of R have the same value for K (Uniqueness property)
  - No proper subset of K has the uniqueness property (Minimality or Irreducibility property)
- One candidate key is chosen to be the primary key of a relation. Remaining candidate keys are termed alternate keys.
- A superkey is an attribute or set of attributes which only exhibits the uniqueness property

Selection of a Primary key

- **A primary key must be chosen considering the data that *may be added to the table in the future***
  - Names, dates of birth etc are rarely unique and as such are not a good option
  - PK should be free of 'extra' semantic meaning, preferably single attribute, preferably numeric (see Table 5.3 Coronel & Morris)
  - Natural vs Surrogate

stu_no	surname	firstname	degree	DOB
1111	Black	Sam	BBIS	02-02-1996
1112	Brown	Jane	BITS	01-01-1995
1113	Chen	Chan	BITS	09-02-1996
1114	Grey	Maria	BCS	15-12-1995
1115	Indigo	Jose	BITS	28-10-1995
1116	Black	Jet	BCS	13-05-1996
1117	Chen	Maria	BBIS	31-08-1995

TABLE 5.3

DESIRABLE PRIMARY KEY CHARACTERISTICS

PK CHARACTERISTIC	RATIONALE
Unique values	The PK must uniquely identify each entity instance. A primary key must be able to guarantee unique values. It cannot contain nulls.
Nonintelligent	The PK should not have embedded semantic meaning other than to uniquely identify each entity instance. An attribute with embedded semantic meaning is probably better used as a descriptive characteristic of the entity than as an identifier. For example, a student ID of 650973 would be preferred over Smith, Martha L. as a primary key identifier.
No change over time	If an attribute has semantic meaning, it might be subject to updates, which is why names do not make good primary keys. If Vickie Smith is the primary key, what happens if she changes her name when she gets married? If a primary key is subject to change, the foreign key values must be updated, thus adding to the database work load. Furthermore, changing a primary key value means that you are basically changing the identity of an entity. In short, the PK should be permanent and unchangeable.
Preferably single-attribute	A primary key should have the minimum number of attributes possible (irreducible). Single-attribute primary keys are desirable but not required. Single-attribute primary keys simplify the implementation of foreign keys. Having multiple-attribute primary keys can cause primary keys of related entities to grow through the possible addition of many attributes, thus adding to the database workload and making (application) coding more cumbersome.
Preferably numeric	Unique values can be better managed when they are numeric, because the database can use internal routines to implement a counter-style attribute that automatically increments values with the addition of each new row. In fact, most database systems include the ability to use special constructs, such as Autonumber in Microsoft Access, sequence in Oracle, or uniqueidentifier in MS SQL Server to support self-incrementing primary key attributes.
Security-compliant	The selected primary key must not be composed of any attribute(s) that might be considered a security risk or violation. For example, using a Social Security number as a PK in an EMPLOYEE table is not a good idea.

Writing Relations

- Relations may be represented using the following notation:
  - relation_name(attribute1, attribute2, ...)
- The primary key is underlined.
- Example:
  - staff(staffid, surname, initials, address, phone)

Relational Database

- A relational database is a collection of normalised relations.
- Normalisation is part of the design phase of the database and will be discussed in a later lecture.

Example relational database:

order (order_id, orderdate,)

order-line (order_id, product_id, quantity)

product (product_id, description, unit_price)

Foreign Key (FK)

- An attribute/s in a table that exists in the same, or another table as a Primary Key.
- Referential Integrity
  - A Foreign Key value must either match the *primary key* in another table or be NULL.
- The pairing of PK and FK creates relationships (logical connections) between tables. Hence the abstraction away from the underlying storage model.

Data Integrity

- Entity integrity
  - Primary key values must be unique
  - Primary key value must not be NULL.
- Referential integrity
  - The values of FK must either match a value of the PK in the related relation or be NULL.
- Column/Domain integrity
  - All values in a given column must come from the same domain (the same data type and range).

MONASH University

Relational DMLs

- Relational Calculus
- Relational Algebra
- Transform-Oriented Languages (e.g. SQL)
- Graphical Languages
- Exhibit the "closure" property - queries on relations produce relations

Relational Calculus

- Based on mathematical logic.
- Non-procedural.
- Primarily of theoretical importance.
- May be used as a yardstick for measuring the power of other relational languages ("relational completeness").
- Operators may be applied to any number of relations.

RELATIONAL ALGEBRA

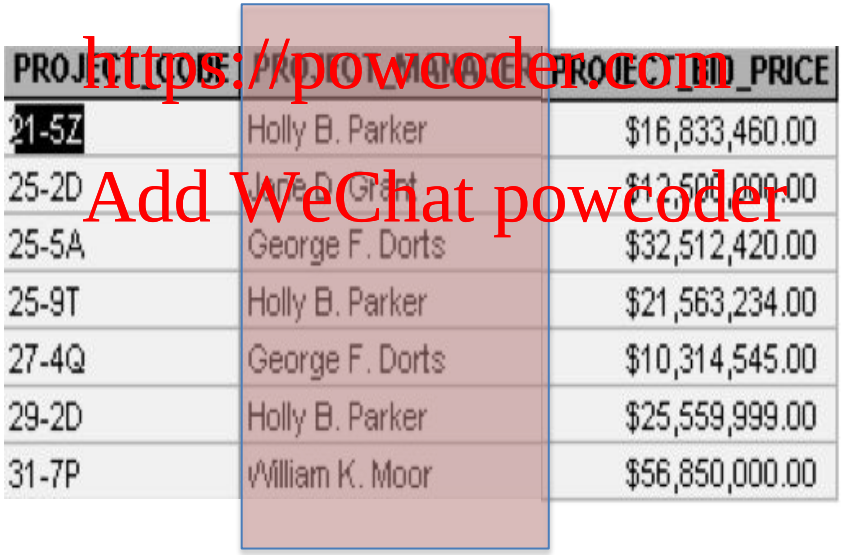
Manipulation of relational data

Relational Algebra

- Relationally complete.
- Procedural.
- Operators only apply to at most two relations at a time.
- 8 basic operations:
  - single relation: selection, projection
  - cartesian product, join
  - union
  - intersection
  - difference
  - division

Relational Operation PROJECT

π



PROJECT_ID	EMP_NAME	PROJECT_BID_PRICE
21-5Z	Holly B. Parker	$16,833,460.00
25-2D	Dave D. Grant	$12,500,000.00
25-5A	George F. Dorts	$32,512,420.00
25-9T	Holly B. Parker	$21,563,234.00
27-4Q	George F. Dorts	$10,314,545.00
29-2D	Holly B. Parker	$25,559,999.00
31-7P	William K. Moor	$56,850,000.00

Relational Operation SELECT

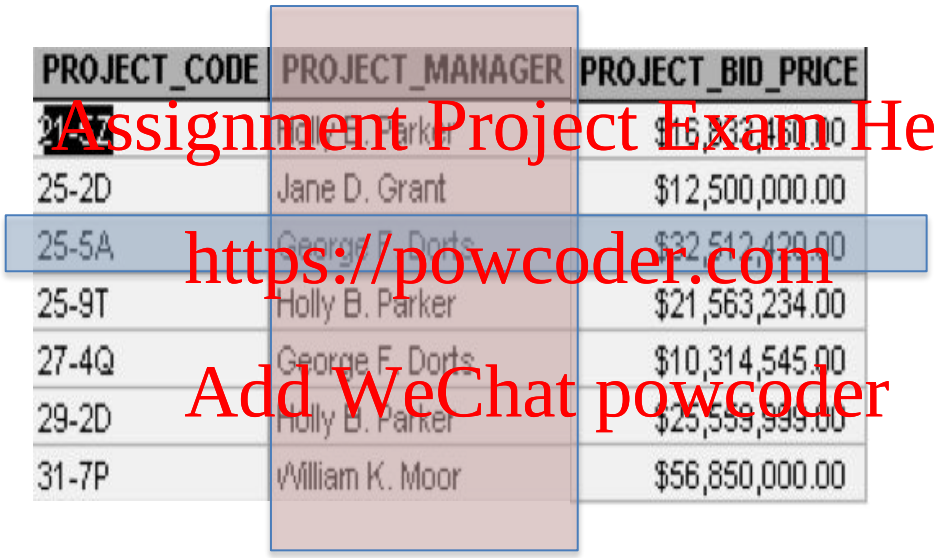
σ

PROJECT_CODE	PROJECT_MANAGER	PROJECT_BID_PRICE
21-5Z	Holly B. Parker	$16,833,460.00
25-2D	Jane D. Grant	$12,500,000.00
25-5A	George F. Dorts	$32,512,420.00
25-9A	Holly B. Parker	$[illegible]34.00
27-4Q	George F. Dorts	$10,314,545.00
29-2D	Holly B. Parker	$25,559,999.00
31-7P	William K. Moor	$56,850,000.00

Relational Operation Multiple Actions

2

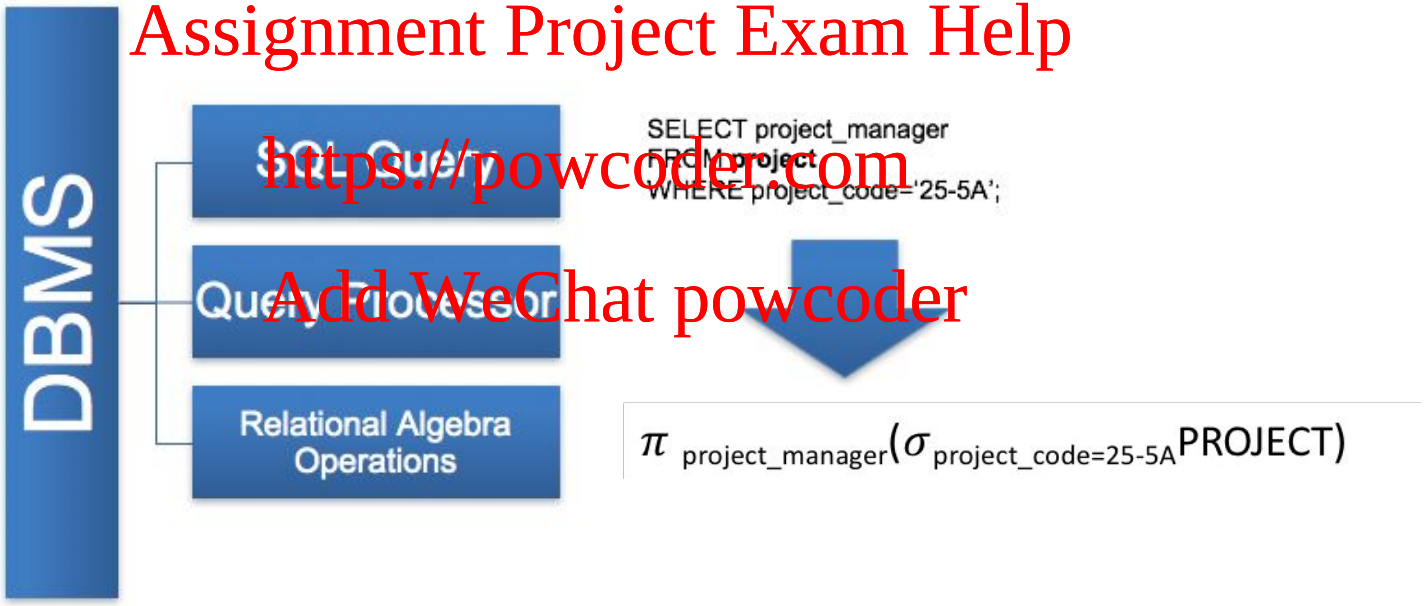
1



PROJECT_CODE	PROJECT_MANAGER	PROJECT_BID_PRICE
21-AA	[illegible]	[illegible]
25-2D	Jane D. Grant	$12,500,000.00
25-5A	George F. Dorts	$32,512,420.00
25-9T	Holly B. Parker	$21,563,234.00
27-4Q	George F. Dorts	$10,314,545.00
29-2D	Holly B. Parker	$25,559,999.00
31-7P	William K. Moor	$56,850,000.00

$$\text{Result} = \pi_{\text{project\_manager}}(\sigma_{\text{project\_code=25-5A}}\text{PROJECT})$$

SQL vs Relational Algebra in the Database



JOIN

- Join operator used to combine data from two or more relations, based on a common attribute or attributes.
- Different types:
  - theta-join
  - equi-join
  - natural join
  - outer join

THETA JOIN (Generalised join)

$$(\text{Relation\_1}) \bowtie_F (\text{Relation\_2})$$

- F is a predicate (i.e. truth-valued function) which is of the form $\text{Relation\_1.a}_i\ \theta\ \text{Relation2.b}_i$
- θ is one of the standard arithmetic comparison operators, i.e. $<, \leq, =, \geq, >$
- Most commonly, θ is equals (=)

NATURAL JOIN

STUDENT

ID	Name
1	Alice
2	Bob

MARK

ID	Subj	Marks
1	1004	95
2	1045	55
1	1045	90

Step 1: STUDENT X MARK
Step 2: delete rows where IDs do not match (select =)

STUDENT.ID	Name	MARK.ID	Subj	Marks
1	Alice	1	1004	95
~~1~~	~~Alice~~	~~2~~	~~1045~~	~~55~~
1	Alice	1	1045	90
~~2~~	~~Bob~~	~~1~~	~~1004~~	~~95~~
2	Bob	2	1045	55
~~2~~	~~Bob~~	~~1~~	~~1045~~	~~90~~

NATURAL JOIN

STUDENT

ID	Name
1	Alice
2	Bob

MARK

ID	Subj	Marks
1	1004	95
2	1045	55
1	1045	90

Step 1: STUDENT X MARK
Step 2: delete rows where IDs do not match (select =)
Step 3: delete duplicate columns (project away)

STUDENT.ID	Name	MARK.ID	Subj	Marks
1	Alice	1	1004	95
1	Alice	1	1045	90
2	Bob	2	1045	55

NATURAL JOIN

STUDENT

ID	Name
1	Alice
2	Bob

⋈

MARK

ID	Subj	Marks
1	1004	95
2	1045	55
1	1045	90

Step 1: STUDENT X MARK
Step 2: delete rows where IDs do not match (select =)
Step 3: delete duplicate columns (project away)

ID	Name	Subj	Marks
1	Alice	1004	95
1	Alice	1045	90
2	Bob	1045	55

A natural join of STUDENT and MARK

OUTER JOIN

STUDENT

ID	Name
1	Alice
2	Bob
3	Chris

⟗

MARK

ID	Subj	Marks
1	1004	95
2	1045	55
1	1045	90
4	1004	100

No information for Chris and the student with ID 4

ID	Name	Subj	Marks
1	Alice	1004	95
1	Alice	1045	90
2	Bob	1045	55

A natural join of STUDENT and MARK

FULL OUTER JOIN

STUDENT

ID	Name
1	Alice
2	Bob
3	Chris

⟗

MARK

ID	Subj	Marks
1	1004	95
2	1045	55
1	1045	90
4	1004	100

Get (incomplete) information of both Chris and student with ID 4

ID	Name	Subj	Marks
1	Alice	1004	95
1	Alice	1045	90
2	Bob	1045	55
3	Chris	Null	Null
4	Null	1004	100

A full outer join of STUDENT and MARK

LEFT OUTER JOIN

STUDENT

ID	Name
1	Alice
2	Bob
3	Chris

⟕

MARK

ID	Subj	Marks
1	1004	95
2	1045	55
1	1045	90
4	1004	100

Get (incomplete) information of only Chris

ID	Name	Subj	Marks
1	Alice	1004	95
1	Alice	1045	90
2	Bob	1045	55
3	Chris	Null	Null

A left outer join of STUDENT and MARK

RIGHT OUTER JOIN

STUDENT

ID	Name
1	Alice
2	Bob
3	Chris

⟖

MARK

ID	Subj	Marks
1	1004	95
2	1045	55
1	1045	90
4	1004	100

Get (incomplete) information of the student with ID 4

ID	Name	Subj	Marks
1	Alice	1004	95
1	Alice	1045	90
2	Bob	1045	55
4	Null	1004	100

A right outer join of STUDENT and MARK