Prof. Joseph G. Vella

JOSEPH.G.VELLA@UM.EDU.MT ©

# Relational Data Model

1

## Introduce Relational Database Model & Logical Database Design (as a destination of an ERM)

2

Ed Codd is listed in IBM's 100 icons of progress.

"[Codd's] relational model was at first very controversial; people thought that the model was too simplistic and that it could never give good performance."

Jim Gray, 2004





## Relational Data Model

• The ***relational data model*** was introduced by Ed Codd [1969]:

**it uses "tables" and some structural rules to build databases (i.e. formally relations mathematical structures); &**

**has the following objectives:**

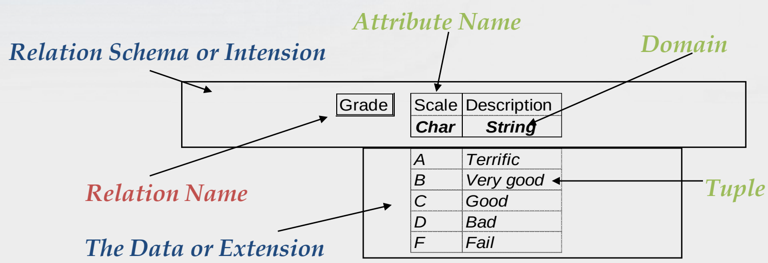
– introduce a high degree of ***logical data independence*** in its designs;

– basis for ***formal (mathematical) treatment*** of database semantics, database operations (e.g. programming languages), consistency and redundancy problems;

  - ❖ **Codd extensively used *first order predicate logic* over relations.**

– enable the realisation of ***set-oriented data manipulation languages***.

## A Relation (by example)








## Relational Databases

- A relation has a:
  - Structural definition
    - Relation name
    - A set of attributes names (and their domains)
  - Data part
    - Each row (i.e., tuple) represent a data item
      - This data item is usually a fact (or part of)
      - It composes a fact's attributes and relationships
- A relational database is a collection of relations (e.g., tables).
- A very popular language that has the relational model as its basis is SQL (Structured Query Model).

## Quick digression ... Datatypes

- **Each attribute is associate with at most one value from a *domain of values*:**
  **(the domains are typically called datatypes in a Relational DBMS).**

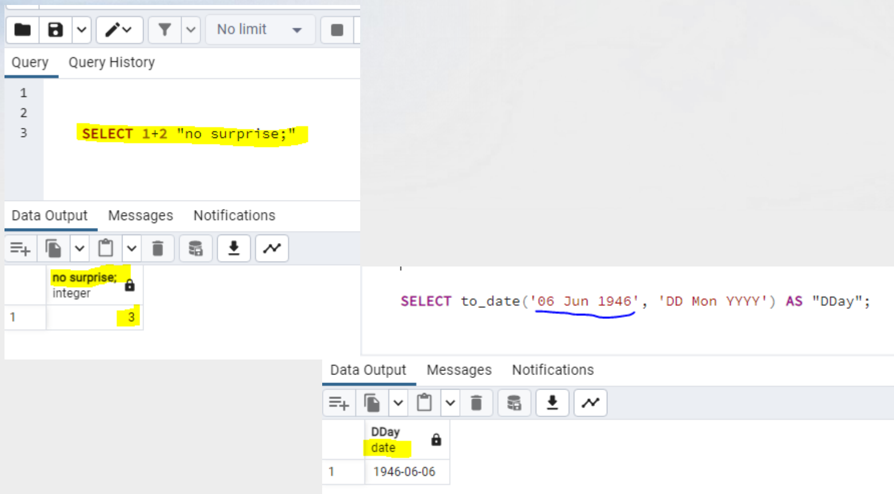
- **Each datatype:**
  - take *space*;
  - an associated *set of functions*;
  - SELECT 1+2 AS "no_surprises";
    SELECT abs(-2) AS "absolute";
    SELECT mod(9,2) AS "modular_arith";
    SELECT power(2,6) AS "pure_bin_power";
  - (sometimes) a domain implicitly ***orders*** it values (e.g. integers);
  - SELECT 1>0 AS "is_it?";



## Quick digression ... Datatypes

- **We also need datatype conversion functions; for example:**
  - from a **number** to **string** (and vice versa):
    - SELECT cast(1+1 AS varchar) "that-s fine";
    - SELECT cast('1' AS int)+cast('1' AS int) "that-s ok too";
  - from a date to an integer:
    - SELECT CURRENT_DATE "Today is"; *-- returns a date*
    - SELECT cast(to_char(CURRENT_DATE, 'J') AS integer) "ThatJulianDate";
  - from a date/time to a string:
    - SELECT to_date('06 Jun 1944', 'DD Mon YYYY') AS "DDay";

- **Some latent problems exists when comparing or converting datatypes:**
  - **Comparing** with **==** and **<>** on FLOATs is lethal!?
  - **Truncation** (i.e. rounding toward dot zero - 3.78 is 3) and **Rounding**.






## Quick digression … Domains and Sets

- **Domains are sets of values specified by a name:**
  - e.g. the **integer** basic domain { -32k, …, -1, 0, 1, … , +32K }
  - e.g. the **Boolean** basic domain { t, f }
  - e.g. the **wine colour** basic domain { red, rose, white }
- **What is a Cartesian product of sets?**
- **The Cartesian product of** Boolean **and** wine colour **domains is the following set of records / tuples:**

  { [t,red], [t,rose], [t,white], [f,red], [f,rose], [f,white]}.

  **The following sets are *subsets* of the Cartesian product:**
  {[t,red], [t,rose], [t,white]};
  {[f,red]};
  {}.
  ***But* the set** {[t,red],[red,hood]} **is *not a subset of*.**

## Relations (by definition)

- **A relation is a subset of the *Cartesian product* of a list of domains characterised by a name (i.e. the relation name).**
  - A relation is analogous to a 2 dim table where each line corresponds to a relation instance / tuple and each column to a domain.
  - An attribute name can distinguish columns that have the same domain (therefore a unique name in a table).
- **A *relation schema* is represented as:**
  ***Rel_name(Attr_1:D_1, … , Attr_n:D_n)* and where**
  *Rel_name* **is the relation identifier;**
  *Attr_i* **is the ith attribute name; and**
  *D_i* **is the ith basic domain.**
  - The ***degree*** of this relation is ***n***.
    The instances of *Rel_name* are sometimes called *n-ary tuples*.
- **A *relation's state* is a set of n-tuples and each tuple, t, is:**
  - Ordered list of $n$ values $t = <v_1, v_2, \ldots, v_n$
  - Each value $v_i$, $1 \le i \le n$, is ***an*** element of dom(*Attr_i*) or is a special NULL value.





## A Relation Schema's semantic interpretation

- **The relation's schema is a data type declaration.**
- **The relation schema is also called the *intension* and is usually time-invariant.**

| COURSE | CourseName | CourseNumber | CreditHours | Department |
|---|---|---|---|---|
| | Intro to Computer Science | CS1310 | 4 | CS |
| | Data Structures | CS3320 | 4 | CS |
| | Discrete Mathematics | MATH2410 | 4 | MATH |
| | Database | CS3380 | 4 | CS |

## A Relation's semantic interpretation

- Each tuple can be interpreted as a *fact* of a certain relation schema.
- Of course the relation contents are dynamic.

| COURSE | CourseName | CourseNumber | CreditHours | Department |
|---|---|---|---|---|
| | Intro to Computer Science | CS1310 | 4 | CS |
| | Data Structures | CS3320 | 4 | CS |
| | Discrete Mathematics | MATH2410 | 4 | MATH |
| | Database | CS3380 | 4 | CS |



## Nulls!?

- ***Nulls are assigned in a table's attribute for the non-applicable or missing values.***
  - *Nulls are independent of a data domain.*
- **We need to actively design out nulls rather than introduce blanket not null integrity constraint in each attribute!?**

## Relational Data Model Characteristics

- **No such thing as implying a *tuples sequence* in a relation. Why?**
  - **Since a relation is a set of tuples the order of tuples is logically irrelevant.**
    - For example the following two sets *are identical*: **{[A:1], [A:2], [A:3]}** and **{[A:3], [A:1], [A:2]}.**
- **The *attribute's order* in the relation's schema definition is not a strict regime.**
    - For example the following two sets *are identical* too: **{[A:1,B:2], [A:3,B:4]}** and **{[B:2,A:1,], [B:4, A:3]} .**

➢ **Re-emphasise that the attribute domain are the *basic ones only*.**





## Primary Keys

- **No two tuples can have the same combination of values for all their attributes.**
- **Superkey (SK subset of R's attributes)**
  - No two distinct tuples in any state *r* of *R* can have the same value for SK
- **Key (K subset of R's attributes)**
  - Superkey of R
  - Removing any attribute *A* from *K* leaves a set of attributes *K* that is not a superkey of *R* any more
    - **Neither can any attribute, part of K, takes a Null value.**
- **Key satisfies two properties:**
  - Two distinct tuples in any state of relation cannot have identical values for (all) attributes in key
  - Minimality of the superkey

## Primary Keys (& Candidate Keys)

- **Candidate key**
  - Relation schema may have more than one superkey
- **Primary key of the relation**
  - Designated among candidate keys
  - Underline attribute
- **Practical Note:**
  - **some candidate key sets are based on *arbitrary sequences* (or systems generated strings or numbers).**



17

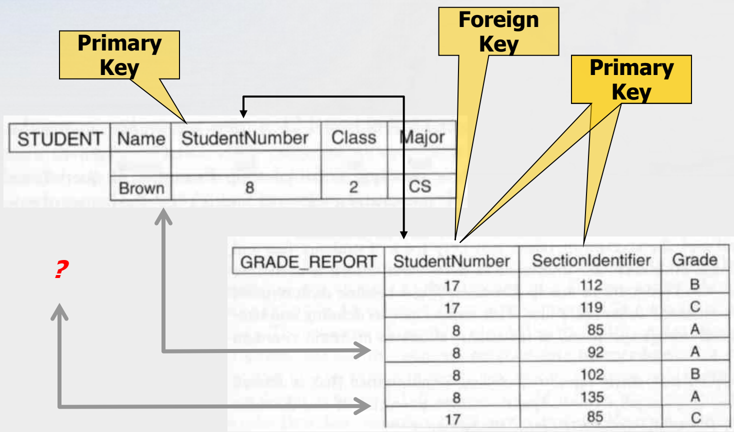
## Foreign Keys

- **Foreign keys are used to join tables or navigate between tables together.**
  - A relation schema may have an attribute, FK, that corresponds to the primary key of another relation. The attribute is called a foreign key.
  - Note: the result of a joining tables is not guaranteed against redundancy!?
- **Foreign key rules:**
  - The attributes in FK have the same domain(s) as the primary key attributes PK
  - Value of FK in a tuple $t_1$ of the current state $r_1(R_1)$ either occurs as a value of PK for some tuple $t_2$ in the current state $r_2(R_2)$ or is NULL



18

## Referential Constraint Depiction




| STUDENT | Name | StudentNumber | Class | Major |
|---|---|---|---|---|
| | Brown | 8 | 2 | CS |

| GRADE_REPORT | StudentNumber | SectionIdentifier | Grade |
|---|---|---|---|
| | 17 | 112 | B |
| | 17 | 119 | C |
| | 8 | 85 | A |
| | 8 | 92 | A |
| | 8 | 102 | B |
| | 8 | 135 | A |
| | 17 | 85 | C |



## Integrity Rules and Constraints (i)

- **What is an integrity constraint? It is an explicit design rule that any database state, at any point in time, has to uphold.**
  ***Through the relational model we can easily express*** **some** ***of these …***
  - *Clarification!*
  - *Remember our transaction are atomic therefore database consistency has to tally at the end of a transaction.*
- **Simple examples of integrity controls:**
  - **Cost price** is lower than **Selling price**.
  - **Date of Birth** must be an acceptable value (e.g. not more than 75 years of age).
- **More involved constraints:**
  - Referential constraints (next slide);
  - Each order must exceed 10 euros.

## Other Types of Constraints

- **Not all conceptual design rules are captured by a logical language!**
  - These require additional artifacts (data modelling extensions?) that fill in the gaps.
    - **These include:**
      - Triggers,
      - Stored procedures (functions),
      - Assertions,
      - Programming language coding.



21

## Relation Model as Implemented with RDBMS

22

## PostgreSQL Data types

- **Generic Types:**
  - Numeric
    - **INTEGER, DECIMAL (exact), REAL, SERIAL**
  - Money
    - **MONEY (2 decimal places)**
  - Character
    - **VARCHAR(n) – n is an upper limit, TEXT, CHAR(n) – n is fixed**
  - Date Time
    - **TIMESTAMP, TIME, DATE, INTERVAL**
  - Booleans
    - **BOOLEAN**
  - Many others
- **Interval, range, and array of … type available too; these are considered part of the Post Relational Data Model!**





## Database vs Schema – *a la* PostgreSQL

- **In PostgreSQL a database can have many schemas (other than the default Public).**
  - The command to create a schema is straight forward:
  - **CREATE SCHEMA** dbrules
    **AUTHORIZATION** postgres;
    **COMMENT ON SCHEMA** scott
    **IS** 'Databases rules ok!';
- **Caution:**
  1. Other DBMS schemas are more like users!? (e.g. Oracle & MS SQL Server);
  2. Naming of objects – table emp in schema dbrules becomes **dbrules.emp**;
  3. In PostgreSQL, if an instance (server process) has many databases a SQL statement scope cannot cross databases!

## Declaring Relations (Tables) in SQL

- The *definition of a relation* has the following declaration in a high level data definition language (*DDL*) as SQL (note the *Primary Key Constraint*):
  - CREATE TABLE student

```
(sname CHAR(30) NOT NULL,
 snumber NUMBER PRIMARY KEY,
 sclass NUMBER,
 smajor CHAR(15));
```

**To get rid of table definition:**

```
DROP TABLE person;
```



## Domains: DIY datatypes

- **A domain in the SQL standard's schema is an element that describes a data type (through those available in the DBMS).**
  - Saves the database designer to reduce the need of many look up tables!
- **For example the following domain describes our grade table:**
- CREATE DOMAIN uni_grade

```
CHAR(1) NOT NULL DEFAULT 'I'
CHECK (VALUE IN ('A','B','C','D','F','I'));
```

- **The SQL standard specifies commands to purge and alter these declarations (i.e. DROP DOMAIN and ALTER DOMAIN).**

## Moving tuples (rows) into tables

- What are the generic operations on a relation instance?
  - *Retrieving tuples that satisfy a user defined criteria;*
  - *Updating a relation's instance* by

inserting **tuples**

```
INSERT INTO student
VALUES ('Borg, Mario', 723345, 21, 'Eng. & Prod.');
INSERT INTO stud(sname, snumber,smajor,sclass)
VALUES ('Zarb, Pete', 723654, 'Pottery & Drama', 19);
```

amending **tuples attributes with new values**

```
UPDATE student
SET smajor = 'RDBD'
WHERE snumber = 723345;
```

deleting **tuples**

```
DELETE FROM student
WHERE CLASS = 21;
```



## Altering the structure of a table

- What are the generic operations on a relation schemas?
  - *Updating a relation's schema:*
    - ***introducing another column to the student's table:***

```
ALTER TABLE student
ADD (sfaculty CHAR(10));
```

- ***introduce a foreign key between tables student and faculty***

```
ALTER TABLE student
ADD CONSTRAINT stud_facullty_fk
FOREIGN KEY (sfaculty)
REFERNCES faculty(fid);
```

## Some Integrity Constraint Definitions

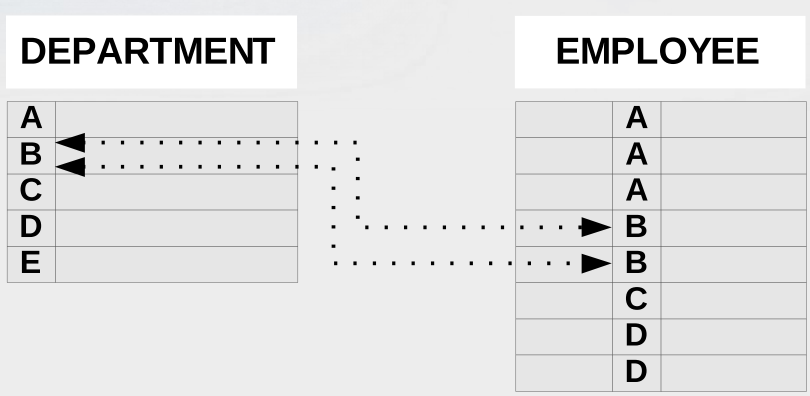
- NOT NULL
  - Specifies that a column's value cannot be assigned a null.
  - CREATE TABLE t1 ( … dob DATE NOT NULL , … ) ;
- UNIQUE
  - Specifies that an attribute set of a relation must have their values unique for all rows in the table (but allows nulls).
  - CREATE TABLE
    t1 ( … phone CHAR(15) CONSTRAINT l1 UNIQUE … ) ;
- CHECK
  - Specifies a condition that must always be true for a table or row.
  - CREATE TABLE
    t1 ( … sfactulty CHAR(6)
    CONSTRAINT ch_fac
    CHECK (sfactulty BETWEEN 1 AND 99) … ) ;





## Basic RDB Design: 1-N

**For example:**
***A number of employees work for a department.***



## Basic RDB Design in SQL: 1-N

- For example:
  *A number of employees work for a department.*
- This is a very common type of relationship.
- ```
  CREATE TABLE  department (
          dref              CHAR(6) PRIMARY KEY ,
          ...
  ) ;
  CREATE TABLE  employee (
          enum              CHAR(6) PRIMARY KEY ,
          ...
          edref             CHAR(6)
          CONSTRAINT   e_dept_dref_fk
          REFERENCES   department(dref)
  ) ;
  ```



31

## Basic RDB Design: 1-1

**For example:**
*A department is managed by a single employee and one employee can only manage, at most, one department.*

**DEPARTMENT**

| | | |
|---|---|---|
| A | | 1 |
| B | | 3 |
| C | | 5 |
| D | | 7 |
| E | | 8 |

**EMPLOYEE**

| | |
|---|---|
| 1 | |
| 2 | |
| 3 | |
| 4 | |
| 5 | |
| 6 | |
| 7 | |
| 8 | |



32
  ```

## Basic RDB Design (SQL): 1-1

- This is not a *very common* type of relationship.
- For example:
  *A department is managed by a single employee and one employee can only manage, at most, one department.*

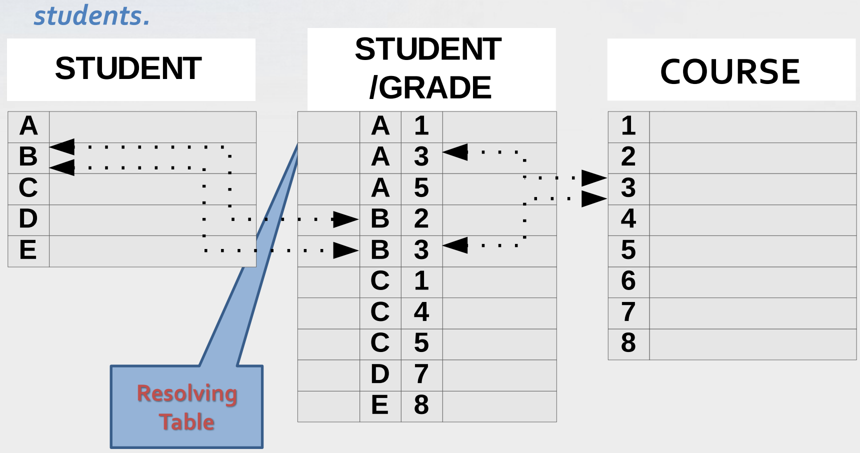
- 
```
CREATE TABLE employee (
    enum CHAR(6) PRIMARY KEY ,
    ... ) ;
CREATE TABLE department (
    dref CHAR(6) PRIMARY KEY ,
    ...
    denum CHAR(6) UNIQUE
    CONSTRAINT d_emp_mang_fk
    REFERENCES employee(enum)
) ;
```





## Basic RDB Design: N - M

**For example:**
*Many students attend a course, and many courses are offered to students.*



## Basic RDB Design (SQL): N - M

- For example:
  *Many students attend a course and many courses are offered to students.*
- 
```
CREATE  TABLE      course (
        cref            CHAR(6)  PRIMARY  KEY ,
        …  ) ;
CREATE  TABLE      student (
        enum               CHAR(6)  PRIMARY  KEY ,
        …) ;
CREATE  TABLE      sc_grade (
        sc_grade       CHAR(1) ,
        sc_cref        CHAR(6)
        CONSTRAINT  sc_c_cref_fk
        REFERENCES  course(cref) ,
        sc_enum        CHAR(6)
        CONSTRAINT  sc_e_enum_fk
        REFERENCES  student(enum) ,
        CONSTRAINT  sc_pk  PRIMARY  KEY  (sc_grade ,  sc_enum));
```

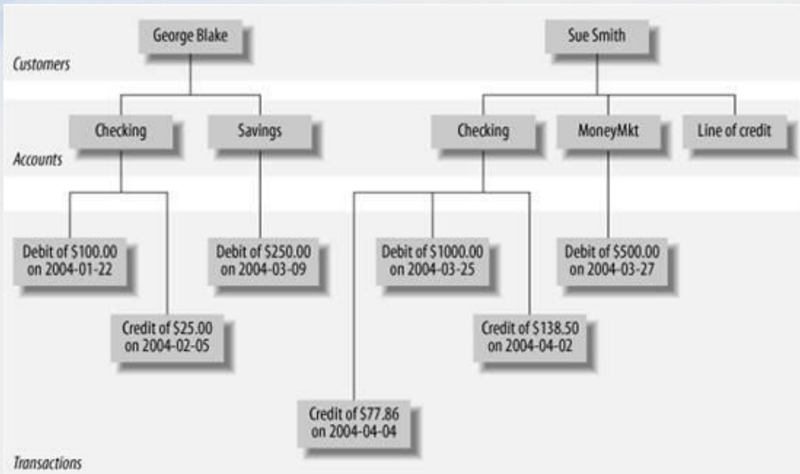
Resolving Table





## Data models and data their data representation

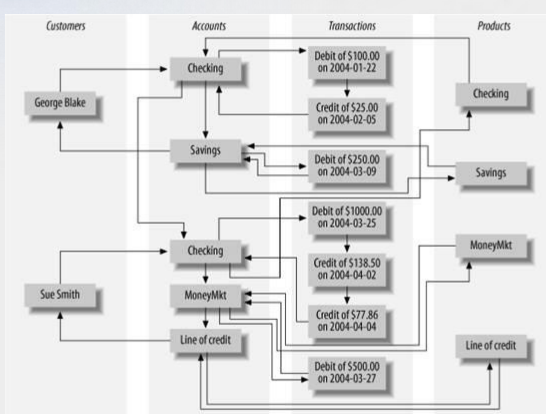
## Hierarchic data model




Joseph Vella - Introduction to Databases 37 Relational Model

37

## Network Data model




Joseph Vella - Introduction to Databases 38 Relational Model

38

## Relational data model

Customer

| cust_id | fname | lname |
|---|---|---|
| 1 | George | Blake |
| 2 | Sue | Smith |

Account

| account_id | product_cd | cust_id | balance |
|---|---|---|---|
| 103 | CHK | 1 | $75.00 |
| 104 | SAV | 1 | $250.00 |
| 105 | CHK | 2 | $783.64 |
| 106 | MM | 2 | $500.00 |
| 107 | LOC | 2 | 0 |

Product

| product_cd | name |
|---|---|
| CHK | Checking |
| SAV | Savings |
| MM | Money market |
| LOC | Line of credit |

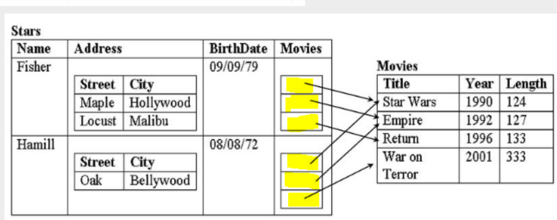
Transaction

| txn_id | txn_type_cd | account_id | amount | date |
|---|---|---|---|---|
| 978 | DBT | 103 | $100.00 | 2004-01-22 |
| 979 | CDT | 103 | $25.00 | 2004-02-05 |
| 980 | DBT | 104 | $250.00 | 2004-03-09 |
| 981 | DBT | 105 | $1000.00 | 2004-03-25 |
| 982 | CDT | 105 | $138.50 | 2004-04-02 |
| 983 | CDT | 105 | $77.86 | 2004-04-04 |
| 984 | DBT | 106 | $500.00 | 2004-03-27 |



39

## Post | Nested Relational Data model

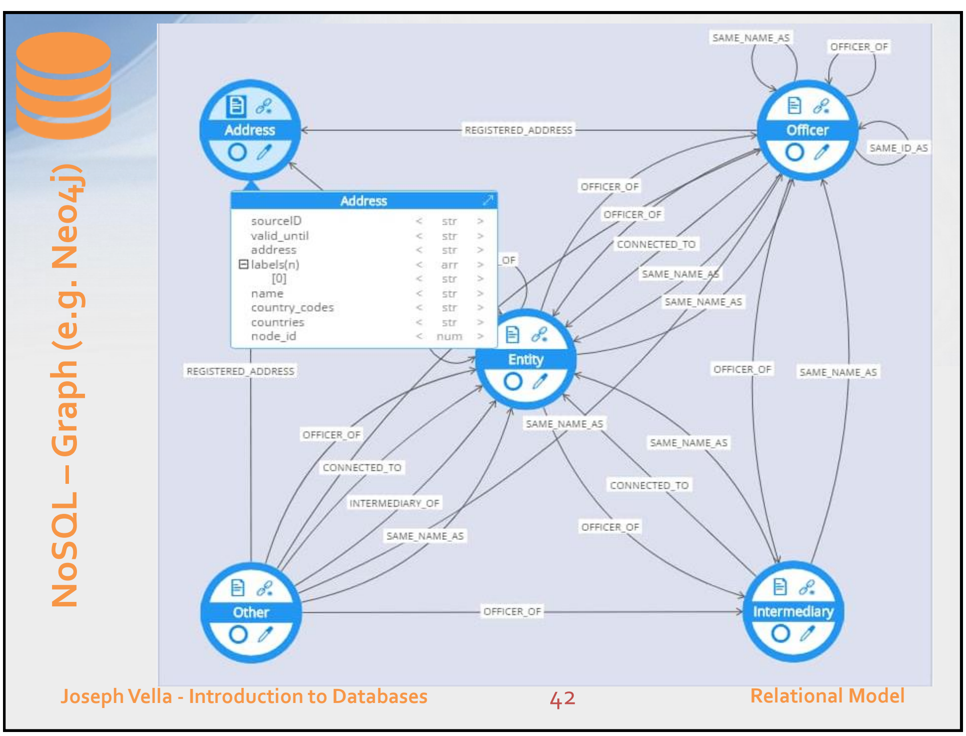






40


NoSQL (e.g. MongoDB)
> db.Order.find()
{ "_id" : ObjectId("54a25edd2fdd64e91fbc5d2d"),
"Data":{
"orderDate":"1999-10-20",
"shipTo":{
"country":"US",
"name":"Alice Smith",
"street":"123 Maple Street",
"city":"Cambridge",
"state":"MA",
"zip":12345
},
"billTo":{
"country":"US",
"name":"destin11",
"street":"8 Oak Avenue",
"city":"Cambridge",
"state":"MA",
"zip":12345
},
"items":{
"item":[
{
"partNum":"242-NO",
"productName":"Nosferatu - Special Edition (1929)",
"quantity":60,
"USPrice":19.99
},
{
"partNum":"242-MU",
"productName":"The Mummy (1959)",
"quantity":60,
"USPrice":19.98
},
]
}
},
}
Joseph Vella - Introduction to Databases
41
Relational Model

41


NoSQL – Graph (e.g. Neo4j)
Address
Officer
Entity
Other
Intermediary
REGISTERED_ADDRESS
SAME_NAME_AS
OFFICER_OF
SAME_ID_AS
CONNECTED_TO
INTERMEDIARY_OF
Address
sourceID < str >
valid_until < str >
address < str >
labels(n) < arr >
[0] < str >
name < str >
country_codes < str >
countries < str >
node_id < num >
Joseph Vella - Introduction to Databases
42
Relational Model

42

## NoSQL – Columnar (based on RDB) (e.g., Vertica)