Relational Database Design: Relational Model

- Tuple:
  - Each row in a table represents a record and is called a tuple .A table containing ‘n’ attributes in a record is called is called n-tuple.
- Attributes:
  - The name of each column in a table is used to interpret its meaning and is called an attribute. Each table is called a relation. In the above table, account_number, branch name, balance are the attributes.
- Domain:
  - A domain is a set of values that can be given to an attributes. So every attribute in a table has a specific domain. Values to these attributes can not be assigned outside their domains.

Relational Database Design: Relational Model

- Relation: A relation consist of
  - Relational schema
  - Relation instance

Relational Schema: A relational schema specifies the relation's name, its attributes and the domain of each attribute. If R is the name of a relation and A1, A2,…An is a list of attributes representing R then R(A1,A2,…,An) is called a Relational Schema. Each attribute in this relational schema takes a value from some specific domain called domain(Ai).

Example:

PERSON (PERSON_ID:INTEGER, NAME:STRING, AGE:INTEGER, ADDRESS:STRING)

Total number of attributes in a relation denotes the degree of a relation since the PERSON relation scheme contains four attributes, so this relation is of degree 4.

Relational Database Design: Relational Model

- Relation Instance: A relational instance denoted as r is a collection of tuples for a given relational schema at a specific point of time.
- A relation state r to the relations schema
  - R(A1, A2…, An) also denoted by r(R) is a set of n-tuples R{t1,t2,…tm}
  - Where each n-tuple is an ordered list of n values

    T= Where each vi belongs to domain (Ai) or contains null values.
- **Eg:**
  - **Relation schema for Student STUDENT(rollno:string, name:string, city:string, age:integer)**

Relation instance:
Student:

Rollno	Name	City	Age
101	Sujit	Bam	23
102	kunal	bbsr	22

Referential integrity

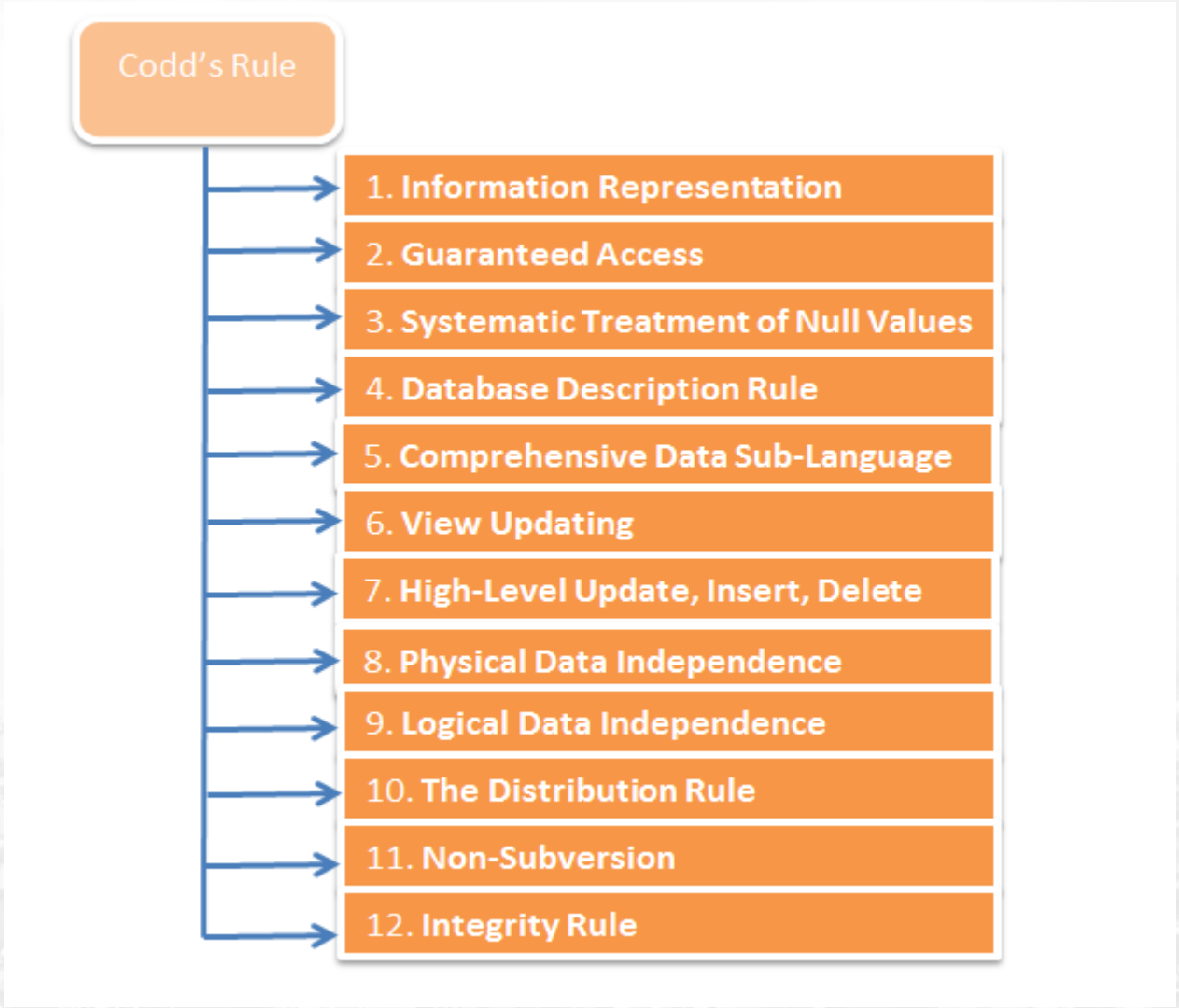
- *Referential integrity* requires that a foreign key must have a matching primary key or it must be null. This constraint is specified between two tables (parent and child); it maintains the correspondence between rows in these tables. It means the reference from a row in one table to another table must be valid.
- Examples of referential integrity constraint in the Customer/Order database of the Company:
  - Customer(**CustID**, CustName)
  - Order(**OrderID**, CustID, OrderDate)
- *The referential integrity constraint states that the customer ID (CustID) in the Order table must match a valid CustID in the Customer table.*

Enterprise Constraints

- Enterprise constraints – sometimes referred to as semantic constraints – are additional rules specified by users or database administrators and can be based on multiple tables.
- Here are some examples.
  - A class can have a maximum of 30 students.
  - A teacher can teach a maximum of four classes per semester.
  - An employee cannot take part in more than five projects.
  - The salary of an employee cannot exceed the salary of the employee's manager.

Codd's Rule

- A **relational database management system (RDBMS)** is a database management system (DBMS) that is based on the relational model as introduced by *E. F. Codd*.
- A short definition of an RDBMS may be a DBMS in which data is stored ***in the form of tables and the relationship among the data is also stored in the form of tables.***
- ***E.F. Codd*,** the famous mathematician has introduced 12 rules (0-12)for the relational model for databases commonly known as **Codd's rules**. The rules mainly define what is required for a DBMS for it to be considered *relational*, i.e., an RDBMS.



Rule-0

- This rule states that all subsequent rules are based on the notation that in order for a database to be considered relational, it must use it’s relational facilities exclusively to manage the database.

Rule 1: Information Rule

- All information in the database is to be represented in one and only one way.
- All information in an RDB is **represented as values in the tables.**
- This is achieved by values in column and rows of tables.
- All information including table names, column names and column data types should be available in same table within the database.
- The basic requirement of the relational model.

id	product_id	quantity	number	addDate
1	1	10	12986	2011-09-20 09:44:29
2	2	25	12986	2011-09-20 09:44:46
3	3	15	12986	2011-09-20 09:45:17
4	4	10	12986	2011-09-20 09:45:22
5	5	50	12986	2011-09-20 09:45:40
6	6	20	12986	2011-09-20 09:45:43
7	7	25	12986	2011-09-20 09:46:20
8	8	25	12986	2011-09-20 09:46:23

The data stored in a database, may it be user data or metadata, must be a value of some table cell. Everything in a database must be stored in a table format.

Rule 2: Guaranteed Access Rule

- **Each unique piece of data should be accessible by: table name + primary key(row) + attribute(column).**
- All data are uniquely identified and accessible via this identity.
- Most RDBMS do not make the definition of the primary key mandatory and are deficient to that extent .

Primary Keys

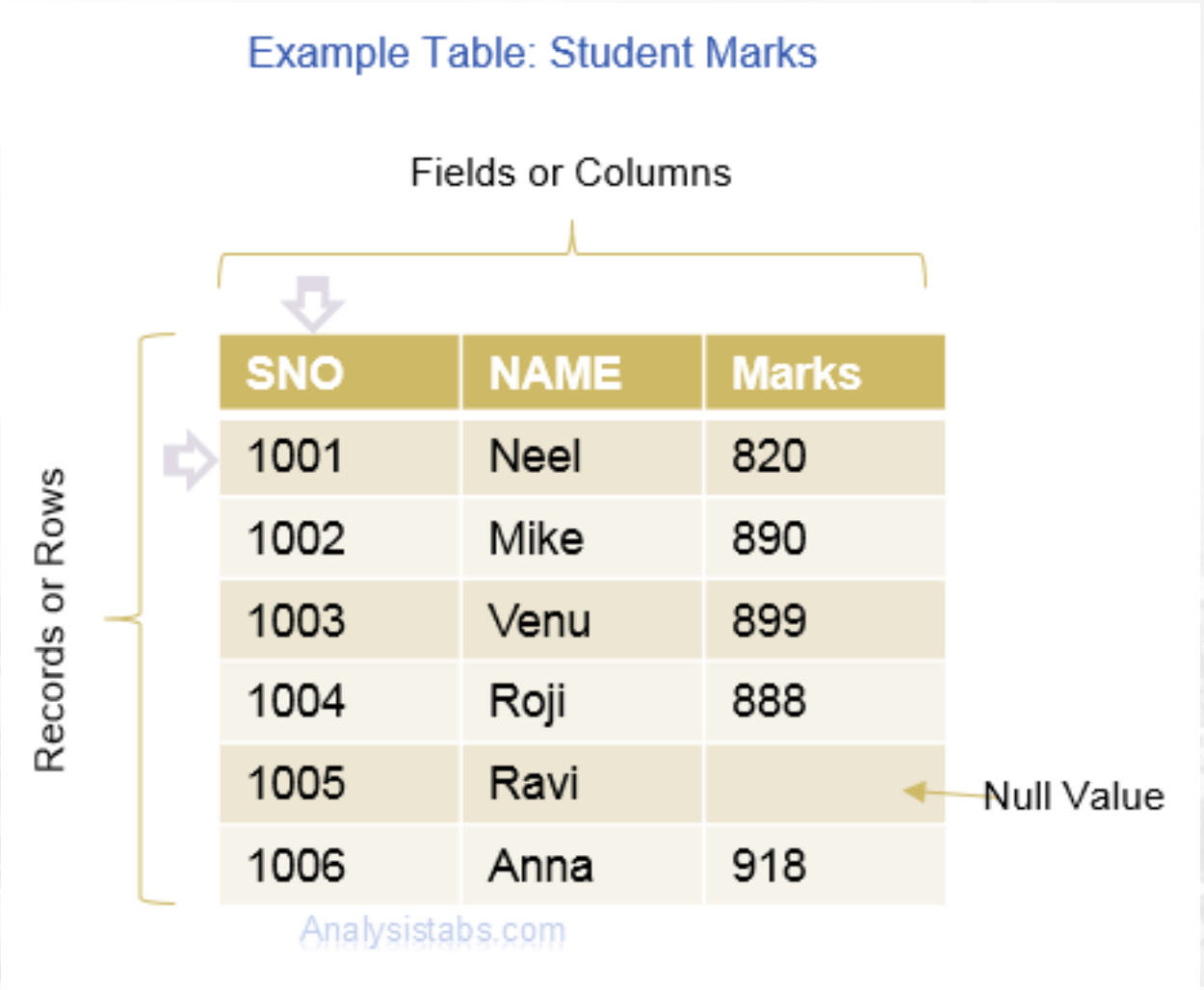
StudentId	firstName	lastName	courseId
L0002345	Jim	Black	C002
L0001254	James	Harradine	A004
L0002349	Amanda	Holland	C002
L0001198	Simon	McCloud	S042
L0023487	Peter	Murray	P301
L0018453	Anne	Norris	S042

Rule 3 : Systematic treatment of null values

The NULL values in a database must be given a systematic and uniform treatment. This is a very important rule because a NULL can be interpreted as one the following - data is missing, data is not known, or data is not applicable.

- "Null values (distinct from the empty character string or a string of blank characters and distinct from zero or any other number) are supported in fully relational DBMS for **representing missing information and inapplicable information** in a systematic way, independent of data type."

- NULL may mean: Missing data, Not applicable
- Should be handled consistently - Not Zero or Blank
- Primary keys — Not NULL
- Expressions on NULL should give NULL.
- Separate handling of missing and/or non applicable data.
- This is distinct to zero or empty strings



Rule 4:Database Description Rule

- The data base description is represented at the logical level in the same way as-ordinary data, so that authorized users can apply the same relational language to its interrogation as they apply to the regular data.
- The authorized users can access the database structure by using common language i.e. SQL
- **The structure description of the entire database must be stored in an online catalog, known as data dictionary, which can be accessed by authorized users. Users can use the same query language to access the catalog which they use to access the database itself.**

Rule 5: Comprehensive Data Sublanguage

- A relational system may support several languages and various modes of terminal use .However, **there must be at least one language whose statements are expressible,** per some well-defined syntax, as character strings and that is comprehensive in supporting all the following items :
- Data Definition (create,insert,update)
- View Definition
- Data Manipulation (alter,delete,truncate)
- Integrity Constraints (primary key,foreign key,null values)
- Authorization (GRANT , REVOKE)
- Transaction boundaries (begin,commit,rollbacketc)

Every RDBMS should provide a language to allow the user to query the contents of the RDBMS and also manipulate the contents of the RDBMS.

Rule 6: View Updating Rule

- View = ”Virtual table”, temporarily derived from base tables.
- Example: If a view is formed as join of 3 tables, changes to view should be reflected in base tables.
- Not updatable: View does not have NOT-NULL attribute of base table.
- It allow the update of simple theoretically updatable views, but disallow attempts to update complex views.
- **A TABLE NAMED ‘STUDENT‘: RDBMS GIVES US THE FACILITY TO VIEW ONLY SOME PARTICULAR FIELDS ACCORDING TO OUR NEED WHICH ARE DIRECTLY ACCESSED FROM BASE TABLES WHEN REQUIRED.**

All the views of a database, which can theoretically be updated, must also be updatable by the system

name	class	Marks	PUPIN NUMBER
SONALI	BCA-2	95	17231
TAMANNA	BCA-2	90	17236
RAJWINDER	BCA-2	90	17267
SAKSHI	BCA-2	86	17893
SADHANA	BCA-2	82	17453

Rule 7 : High-level Insert , Update And Delete

- **This rule states that insert, update, and delete operations should be supported for any retrievable set rather than just for a single row in a single table.**
- **It also perform the operation on multiple row simultaneously .**
- There must be delete, updating and insertion at the each level of operation. Set operation like union, all union , insertion and minus should also supported.

EXAMPLE:

- Suppose if we need to change ID then it will reflect everywhere automatically.

Rule 8:Physical Data Independence

- The ability to change the physical schema without changing the logical schema is called physical data independence.

- This is saying that users shouldn't be concerned about how the data is stored or how it's accessed. In fact, users of the data need only be able to get the basic definition of the data they need.

- EXAMPLE:

A change to the internal schema, such as using different file organization or storage structures, storage devices, or indexing strategy, should be possible without having to change the conceptual or external schemas.

Rule 9 :Logical Data Independence Rule

- What is independence?

  The ability to modify schema definition in on level without affecting schema definition in the next higher level is called data independence

- The ability to change the logical (conceptual) schema without changing the External schema (User View) is called logical data independence.

-

- EXAMPLE:

  The addition or removal of new entities, attributes, or relationships to the conceptual schema should be possible without having to change existing external schemas or having to rewrite existing application programs.

Rule 10 : Integrity Independence Rule

- Data integrity refers to maintaining assuring the accuracy and consistency of data over its entire life cycle.

1. First ensure that correct data type is used.
2. Check constraints: these allow column value to be checked against other column before insertion is allowed

Rule 11 : Distribution Independence Rule

- "THE RELATION DATA BASE MANAGEMENT HAS DISTRIBUTION INDEPENDENCE"

- Distribution independence implies that user should not have to be aware of whether a database is distributed at different sites or not.
- Application program and adhoc request are not affected by the change in distribution of physical data. Application program will work even if the programs and data are moved on different site

- The RDBMS may spread across the more one system or several networks

Rule 12 : Non-subversion Rule

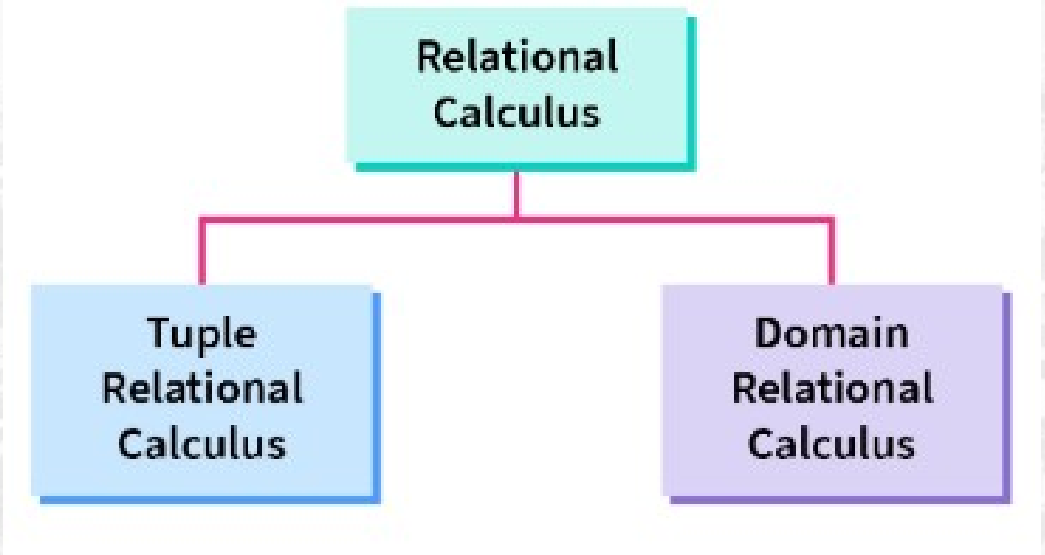
- There should be no way to modify to database structure other than through the multiple row data base language(SQL).
- If a system has an interface that provides access to low-level records, then the interface must not be able to subvert the system and bypass security and integrity constraints.

Example:

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

Relational Calculus

- Relational Calculus in database management system (DBMS) is all about ***"What you want ?"***.
- Relational calculus does not tell us how to get the results from the Database, but it just cares about what we want.
- The theory of Relational calculus was introduced by computer scientist and mathematician Edgar Codd.



Tuple Relational Calculus

- Tuple Relational Calculus is a **non-procedural query language** unlike relational algebra.
- Tuple Calculus provides only the description of the query but it does not provide the methods to solve it.
- Thus, it explains what to do but not how to do.
- In Tuple Calculus, a query is expressed as :

  {t| P(t)}

  where t = resulting tuples,
  P(t) = known as Predicate and these are the conditions that are used to fetch t

- Thus, it generates set of all tuples t, such that Predicate P(t) is true for t.

- P(t) may have various conditions logically combined with OR (∨), AND (∧), NOT(¬).
  It also uses quantifiers:
  ∃ t ∈ r (Q(t)) = ”there exists” a tuple in t in relation r such that predicate Q(t) is true.
  ∀ t ∈ r (Q(t)) = Q(t) is true “for all” tuples in relation r.

Example

- Let’s say we have a table called “Employees” with the following attributes:

- EmployeeID
- Name
- Salary
- DepartmentID

To retrieve the names of all employees who earn more than $50,000 per year, we can use the following TRC query:

$$\{ t \mid Employees(t) \land t.Salary > 50000 \}$$

In this query, the “Employees(t)” expression specifies that the tuple variable t represents a row in the “Employees” table. The “ ∧ ” symbol is the logical AND operator, which is used to combine the condition “t.Salary > 50000” with the table selection.

Example contd..

Table-1: Customer

Customer name	Street	City
Saurabh	A7	Patiala
Mehak	B6	Jalandhar
Sumiti	D9	Ludhiana
Ria	A5	Patiala

Table-2: Branch

Branch name	Branch city
ABC	Patiala
DEF	Ludhiana
GHI	Jalandhar

Table-3: Account

Account number	Branch name	Balance
1111	ABC	50000
1112	DEF	10000
1113	GHI	9000
1114	ABC	7000

Table-4: Loan

Loan number	Branch name	Amount
L33	ABC	10000
L35	DEF	15000
L49	GHI	9000
L98	DEF	65000

Table-5: Borrower

Customer name	Loan number
Saurabh	L33
Mehak	L49
Ria	L98

Table-6: Depositor

Customer name	Account number
Saurabh	1111
Mehak	1113
Sumiti	1114

Exercise

1: Find the loan number, branch, amount of loans of greater than or equal to 10000 amount.

2: Find the loan number for each loan of an amount greater or equal to 10000.

3: Find the names of all customers who have a loan and an account at the bank.

4: Find the names of all customers having a loan at the “ABC” branch.

Solution

- **1:** Find the loan number, branch, amount of loans of greater than or equal to 10000 amount.

$$\{t \mid t \in loan \ \wedge \ t[amount] >= 10000\}$$

- **2:** Find the loan number for each loan of an amount greater or equal to 10000.

$$\{t \mid \exists\, s \in loan(t[loan\ number] = s[loan\ number] \ \wedge \ s[amount] >= 10000)\}$$

- **3:** Find the names of all customers who have a loan and an account at the bank.

$$\{t \mid \exists\, s \in borrower(\ t[customer\text{-}name] = s[customer\text{-}name])$$
$$\wedge \ \exists\, u \in depositor(\ t[customer\text{-}name] = u[customer\text{-}name])\}$$

- **4:** Find the names of all customers having a loan at the “ABC” branch.

$$\{t \mid \exists\, s \in borrower(t[customer\text{-}name] = s[customer\text{-}name]$$
$$\wedge \ \exists\, u \in loan(u[branch\text{-}name] = \text{“ABC”} \ \wedge \ u[loan\text{-}number] = s[loan\text{-}number]))\}$$

Domain Relational Calculus

- **Domain Relational Calculus** is a non-procedural query language equivalent in power to Tuple Relational Calculus.
- Domain Relational Calculus provides only the description of the query but it does not provide the methods to solve it.
- Domain Relational Calculus uses domain Variables to get the column values required from the database based on the predicate expression or condition.

- In Domain Relational Calculus, a query is expressed as,

{ < x1, x2, x3, ..., xn > | P (x1, x2, x3, ..., xn) }

where, < x1, x2, x3, …, xn > represents resulting domains variables and P (x1, x2, x3, …, xn) represents the condition or formula equivalent to the Predicate calculus.

- Note: The domain variables those will be in resulting relation must appear before | within ≺ and ≻ and all the domain variables must appear in which order they are in original relation or table.

Example

Table-1: Customer

Customer name	Street	City
Debomit	Kadamtala	Alipurduar
Sayantan	Udaypur	Balurghat
Soumya	Nutanchati	Bankura
Ritu	Juhu	Mumbai

Table-2: Loan

Loan number	Branch name	Amount
L01	Main	200
L03	Main	150
L10	Sub	90
L08	Main	60

Table-3: Borrower

Customer name	Loan number
Ritu	L01
Debomit	L08
Soumya	L03

Queries

1: Find the loan number, branch, amount of loans of greater than or equal to 100 amount.

$$\{l \in b, a \mid \wedge \square l, b, a \square \ \square \ \text{loan} \ \square \ (a \geq 100)\}$$

2: Find the loan number for each loan of an amount greater or equal to 150.

$$\{\boxplus \notin \wedge b, a \ (\square l, b, a \square \ \square \ \text{loan} \ \square \ (a \geq 150)\}$$

3: Find the names of all customers having a loan at the “Main” branch and find the loan amount.

$$\{\langle c, a\rangle \mid \exists\, l\, (\langle c, l\rangle \in \text{borrower} \wedge \exists\, b\, (\langle l, b, a\rangle \in \text{loan} \wedge (b = \text{“Main”})))\}$$