Midterm Review

CS 377: Database Systems

Piazza Poll Results

A total of **45** vote(s) in **91** hours

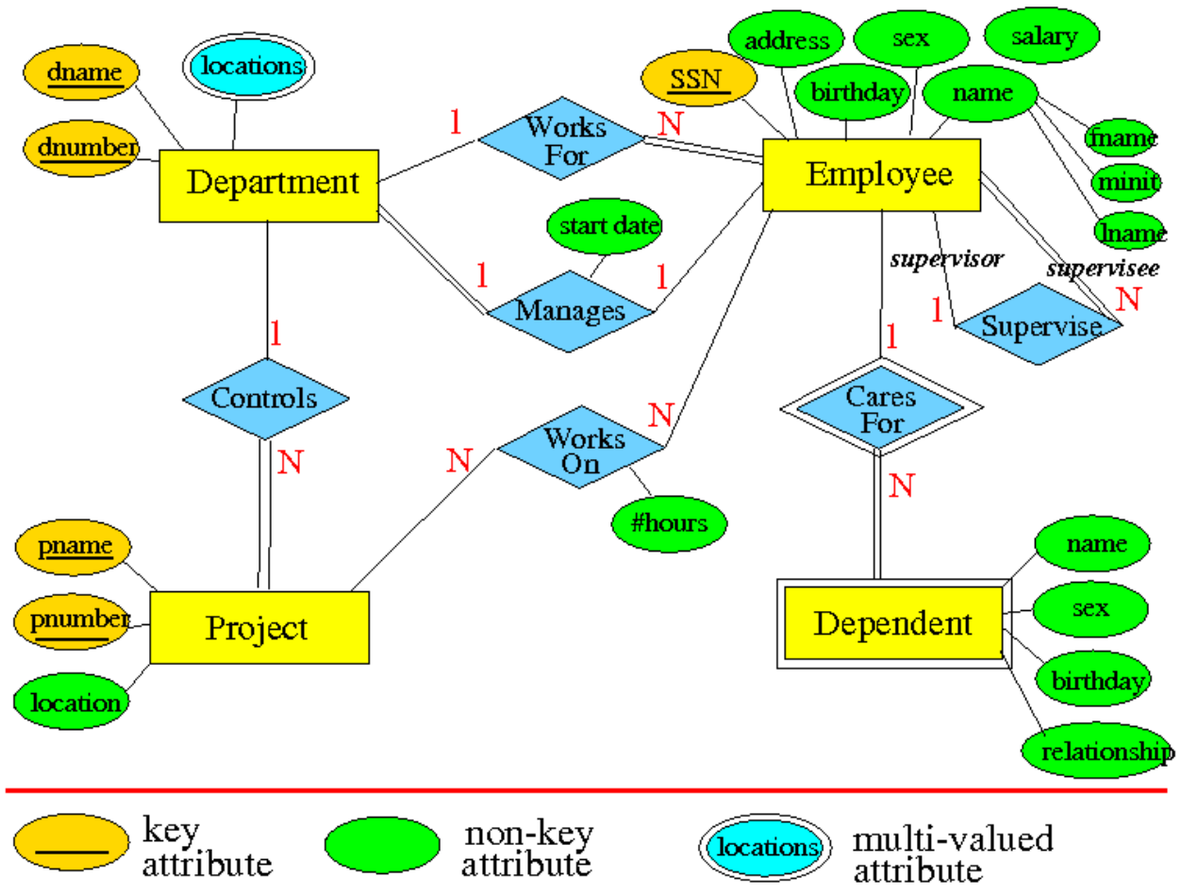
4 (9% of users)	Database Concepts	Show Voters
5 (11% of users)	ER Model	Show Voters
6 (13% of users)	Relational Model	Show Voters
23 (51% of users)	Relational Algebra	Show Voters
22 (49% of users)	Relational Calculus	Show Voters
6 (13% of users)	SQL Data Definition	Show Voters
9 (20% of users)	SQL Modification (Add, Update, Delete)	Show Voters
5 (11% of users)	SQL Basic Query (SELECT-FROM-WHERE)	Show Voters
23 (51% of users)	SQL Aggregation (Group by - Having)	Show Voters
30 (67% of users)	SQL Nested Queries	Show Voters
41 (91% of users)	SQL Advanced Queries (QFT)	Show Voters

Database Concepts

- Data model categories: high-level or conceptual data models, low-level or physical data models, and representational or implementation data models
- Physical data and logical data independence
  - How metadata fits into the picture
- Three schema architecture

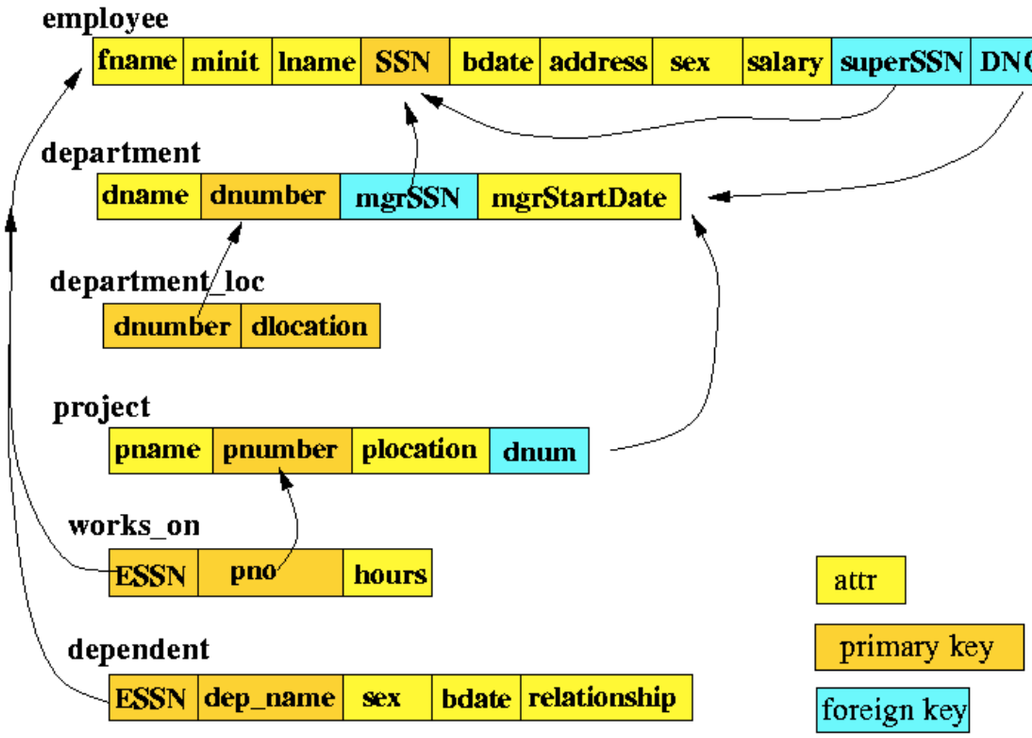
Entity Relationship (ER) Model

- Entity
  - Attributes
  - Weak Entity
- Relationship
  - Degree
  - Cardinality ratio constraint
  - Participation constraint



Relation Model

- Relation, attributes
- Schema vs instance
- Relational model constraints
  - Domain constraint
  - Key constraint
  - Referential integrity constraint



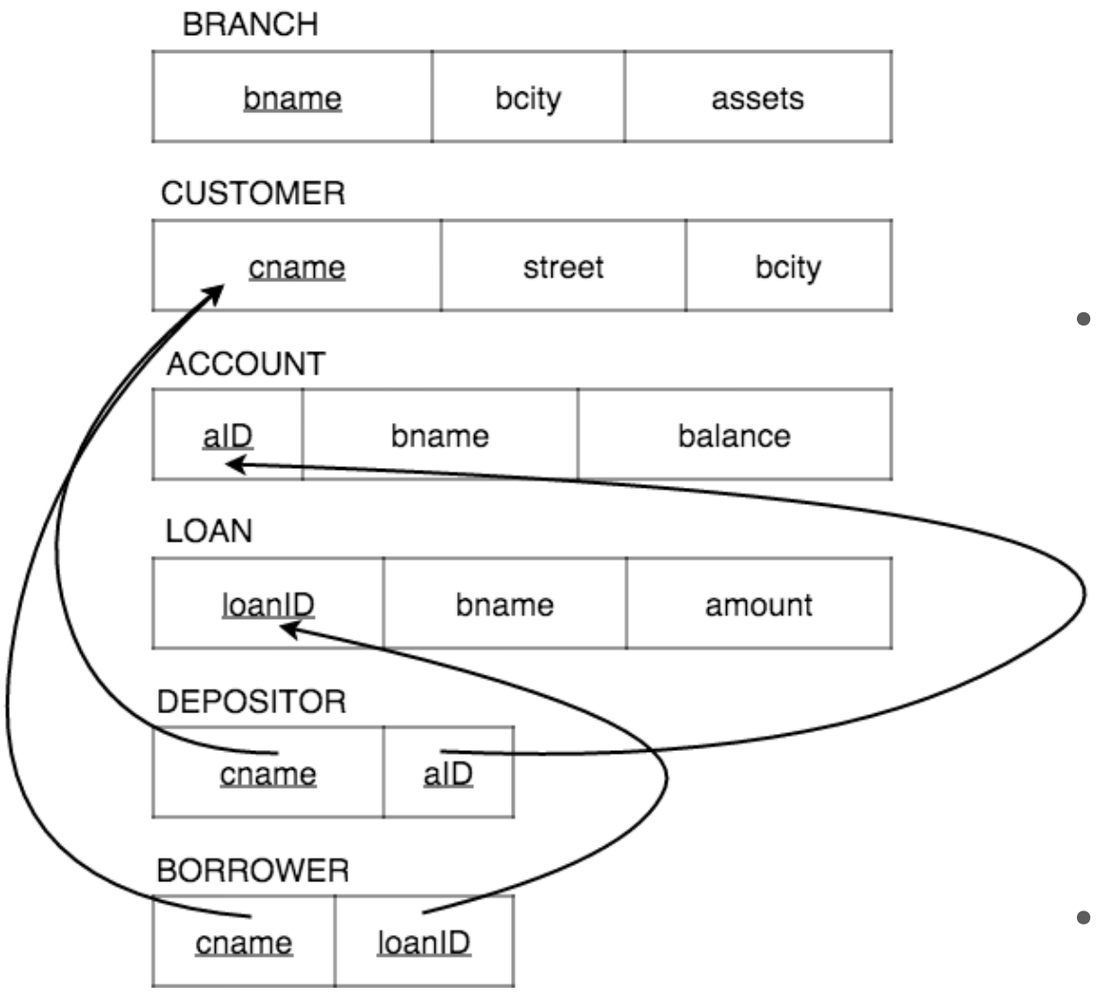
ER to Relational Model

ER Model	Relational model
Entity type	Entity relation
1:1 or 1:N relationship	Expand (or create R relation)
M:N relationship	Create R relation with two foreign keys
n-ary relationship type	Create R relation with n foreign keys
Simple attribute	Attribute
Composite attribute	Set of simple component attributes
Multivalued attribute	Relation and foreign key
Key attribute	Primary (or secondary) key

Relational Algebra

Operation	Notation	Purpose
SELECT	$\sigma_{<\text{selection condition}>}(R)$	Selects all tuples that satisfy the selection condition from a relation R
PROJECT	$\pi_{<\text{atttribute list}>}(R)$	New relation with subset of attributes of R and removes duplicate tuples
THETA_JOIN	$R_1 \bowtie_{<\text{join condition}>} R_2$	All combinations of tuples from R_1 and R_2 that satisfy the join condition
EQUIJOIN	$R_1 \bowtie_{<\text{join condition}>} R_2$	Theta join with only equality join comparisons
NATURAL JOIN	$R_1 *_{<\text{join condition}>} R_2$	Equijoin except join attributes of R_2 are not included in the resulting relation
UNION	$R_1 \cup R_2$	Relation that includes all tuples in R_1 or R_2
INTERSECTION	$R_1 \cap R_2$	Relation that includes all tuples in both R_1 and R_2
DIFFERENCE	$R_1 - R_2$	Relation that includes all tuples in R_1 that are not in R_2
CARTESIAN PRODUCT	$R_1 \times R_2$	Relation with attributes of R_1 and R_2 and includes tuples with all possible combinations of tuples of R_1 and R_2
DIVISION	$R_1(Z) \div R_2(Y)$	Relation that includes all tuples t[X] in $R_1(Z)$ that appear in R_1 in combination with every tuple from $R_2(Y)$ where $Z = X \cup Y$
GROUP BY AGGREGATE	$<\text{group attrs}>\mathcal{F}_{<\text{set funcs}>}$	Relation that includes the grouping attributes and the set function values

Banking Example



- Find the names of all customers who have a loan and a savings account at the bank
- Find the names of all customers who have a loan at the Decatur branch but do not have a savings account at any branch of the bank
- Find all customers who have a savings account at all branches located in Atlanta city

Relational Calculus (Tuple Relational Calculus)

Query of the form: {t | CONDITION(t) }

- Conditions are formulas and are recursively defined
- Atomic formula (Relation(t), R.a *op* S.b / constant)
- Special formula quantifiers
  - Universal quantifier $(\forall t)\ (\mathrm{Condition}(t))$
  - Existential quantifier $(\exists t)\ (\mathrm{Condition}(t))$

SQL Data Definition

- Create database
- Create table
  - Attribute datatypes and constraints
  - Key constraints (primary and foreign key)
  - Circular integrity constraints
- Alter tables
  - Add/remove attributes
  - Add/remove constraints
- Drop tables & databases

SQL Data Modification

- Data modifications does not return a result but changes the database

  - INSERT (add new tuples)
    INSERT INTO <relation> VALUES <attr values>;

  - DELETE (remove tuples)
    DELETE FROM <relation> WHERE <condition>;

  - UPDATE (change value(s) of existing tuples)
    UPDATE <relation>
    SET <list of attribute assignments>
    WHERE <condition>;

SQL Select-From-Where Query

- SQL Query:
  SELECT <attribute list>
  FROM <table list>
  WHERE <condition on the tables>

- RA Query:

$$\pi_{<\text{attribute list}>}\sigma_{<\text{condition}>}(R_1 \times R_2 \times \cdots \times R_n)$$

SQL Basic Query

```
SELECT      [DISTINCT] <attribute list>
FROM         <table list>
[WHERE       <condition on the tables>]
[ORDER BY  <attribute list>   ASC | DESC]
[LIMIT          <number of tuples>]
```

WHERE conditions: comparison operations, arithmetic operations, logical operations, **IN**, **LIKE**, **IS NULL**, **EXISTS**, **ANY**, **ALL**

SQL: Group By & Having

- GROUP BY: Apply aggregate functions to subgroups of tuples in a relation

- HAVING: Filters out groups that do not satisfy the group condition

- SQL Query:

```
SELECT     [DISTINCT] <attribute list>
FROM        <table list>
[WHERE      <condition on the tables>]
[GROUP BY <grouping attributes>]
[HAVING     <group condition>]
```

SQL: Nested Queries

- A subquery (parenthensized **SELECT-FROM-WHERE** statement) inside the **WHERE** clause

- **SELECT …**
  FROM …
  WHERE <some condition test> (SELECT …
  FROM …
  WHERE …)

Nested query

Set membership (IN, NOT IN)
Set comparison (ANY, ALL)
Empty relation test (EXIST)

SQL: Temporal Relations

- A subquery (parenthensized **SELECT-FROM-WHERE** statement) inside the **FROM** clause

- Must give it an alias

- **SELECT <attributes>**
  FROM R1, R2, (SELECT …) <alias>, …, RN
  WHERE <condition>;

temporal relation

SQL: Set Operations

- **UNION**: join two relations
  - Syntax: **(SELECT …) UNION (SELECT …)**
- **INTERSECT**: not in most SQL implementations
  - QFT: **WHERE x IN <set1> AND x IN <set2>**
- **DIFFERENCE**: not in most SQL implementations
  - QFT: **WHERE x IN <set1> AND x NOT IN <set2>**

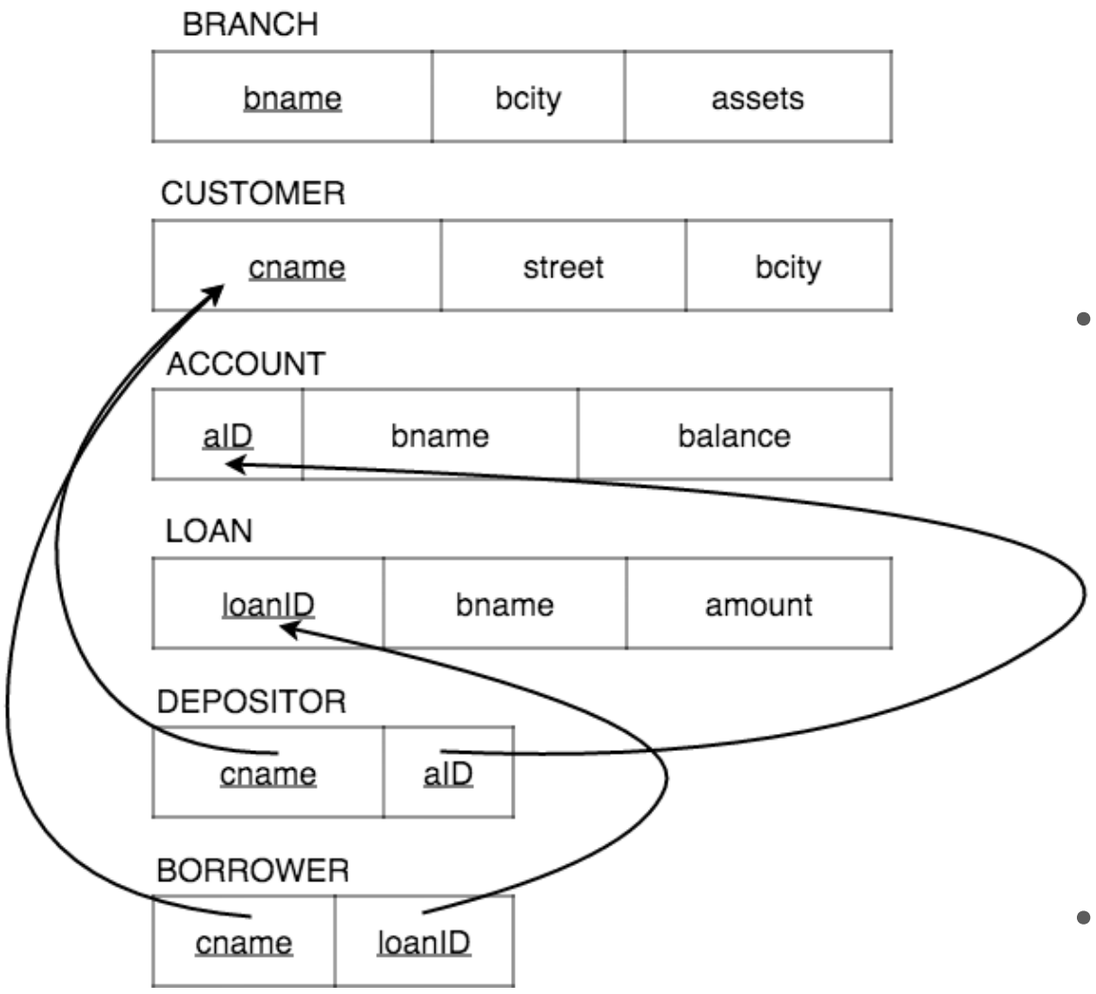
SQL: Set Operations

- **SUPERSET**: set1 superset set2
  - QFT: **SELECT …**
    WHERE NOT EXISTS (SELECT …
    WHERE x IN set2
    AND x NOT IN set1)
- **DIVISION**: A division B => all tuples where a is superset of B

SQL: Other QFTs

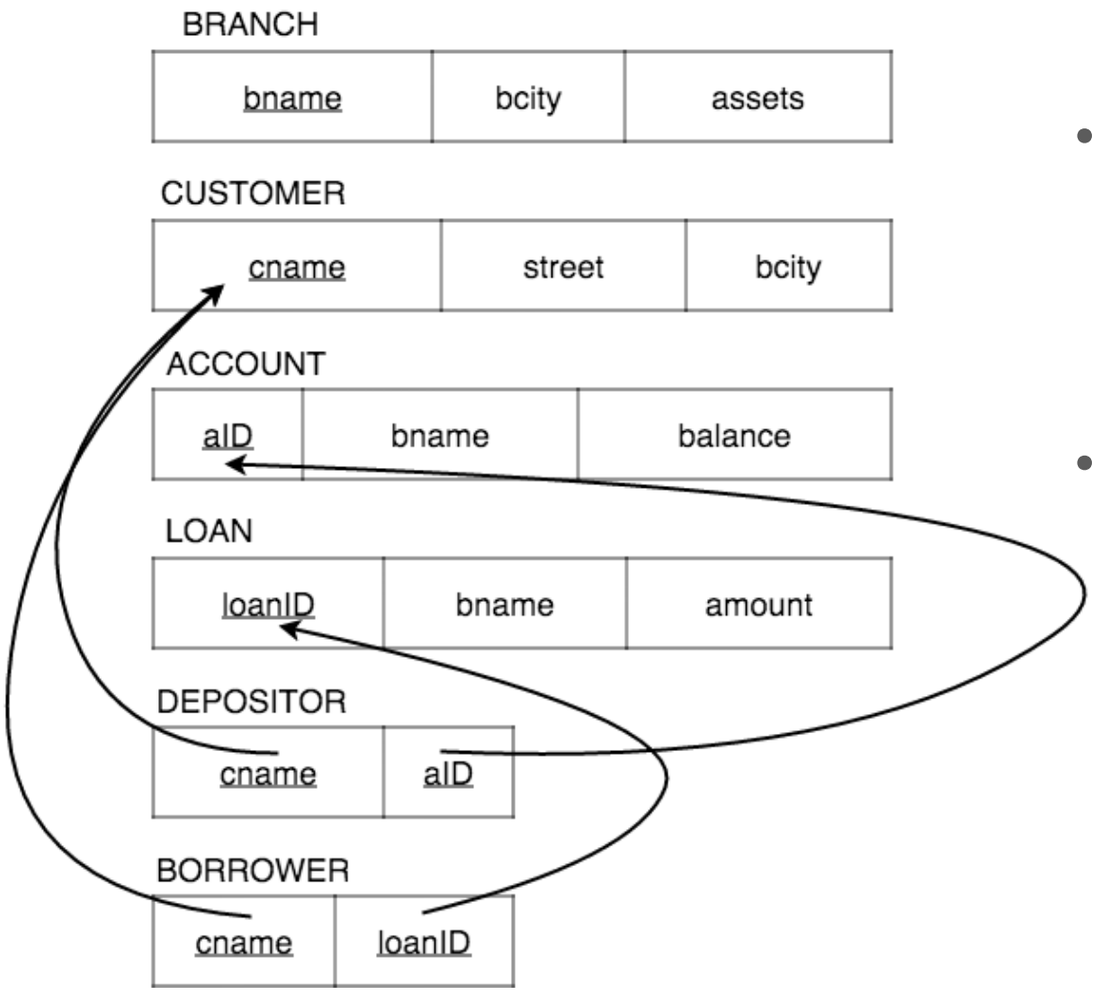
- Only: set1 subset set2
  - QFT: **SELECT …**
    WHERE NOT EXISTS (SELECT …
    WHERE x IN set1
    AND x NOT IN set2)
- Most number of some attribute y
  - QFT: **SELECT …**
    GROUP BY <group>
    HAVING setf(y) = (SELECT MAX(setf(y)) …
    GROUP BY <group>)

Banking Example



- Find the names of all customers who have a loan and a savings account at the bank
- Find the names of all customers who have a loan at the Decatur branch but do not have a savings account at any branch of the bank
- Find all customers who have a savings account at all branches located in Atlanta city

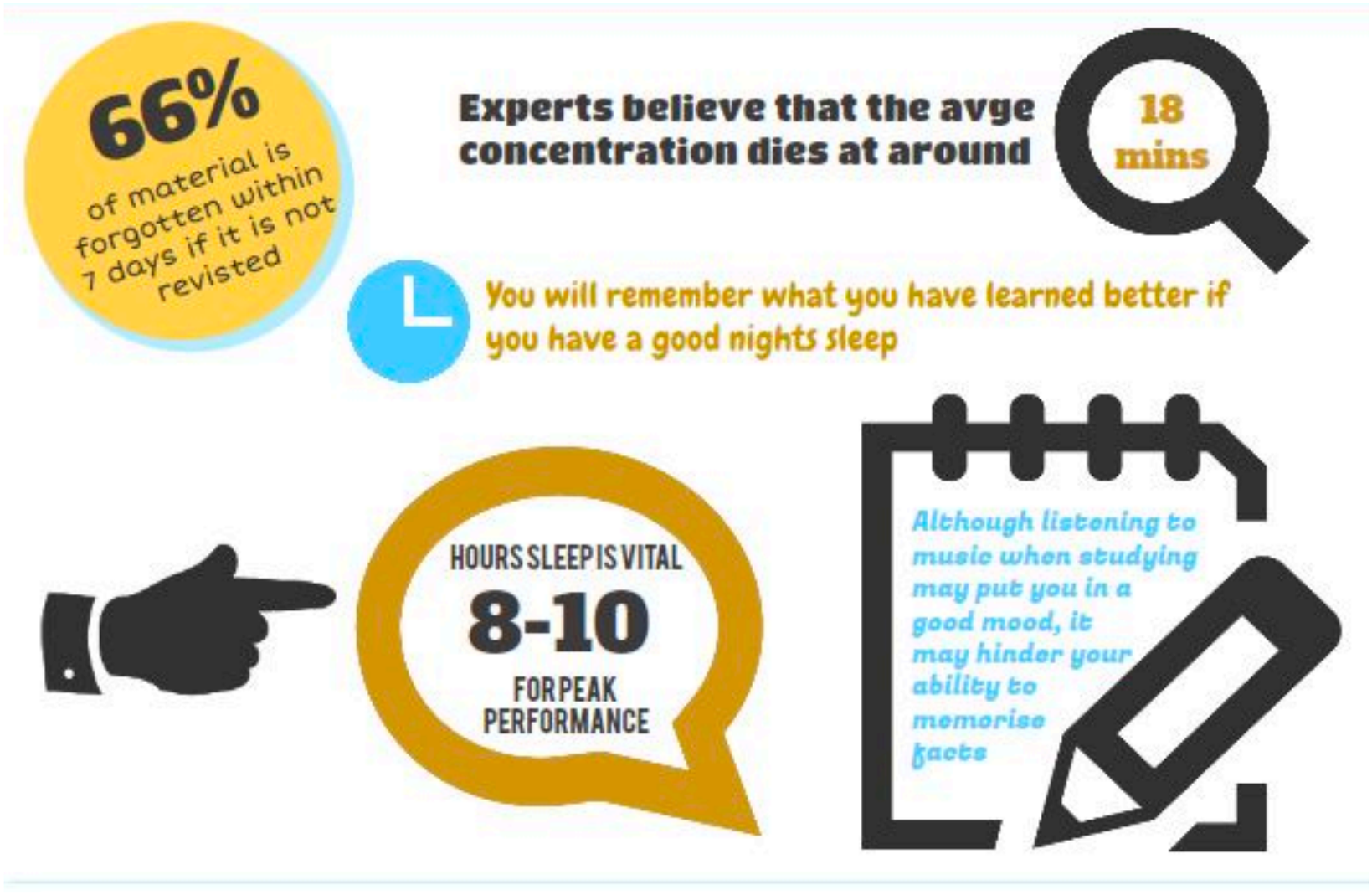
Banking Example



- Find the total sum of all loan amounts in the bank
- Find the names of all branches that have assets greater than those of at least one branch located in "Atlanta"

SQL Views

- View is a virtual table that does not exist in physical form
    - Allows ability to present information in different ways to different users
    - Can be used like an ordinary relation and simplifies complex queries
    - Limits data access to specific users (sensitive data can be hidden)
    - If conceptual schema changes, only the **SELECT** query needed to construct view needs to change



https://s-media-cache-ak0.pinimg.com/736x/68/7e/f6/687ef64b86cbd0a2291871e9e4939ba3.jpg