CS2102 Lecture 12
Review

What you've learned in CS2102

- How to use relational database systems (RDBMS)
- Relational Data model
  - Database = data + constraints
    - Data represented as collections of tables
  - How to define a relational database schema
  - How to update a relational database
  - How to use SQL & transactions
  - How to reason with SQL using formal query languages
    - Relational algebra
    - Relational calculus
- Database Design
  - How to design a ER data model to capture application's data requirements
  - How to map ER data model to relational data model
  - How to refine relational schema using normal forms
- How to develop RDBMS applications

Schema Refinement

movie	rating	cinema	area	time
John Wick 2	M18	JCube	west	1400
Logan	M18	JCube	west	1700
Kong: Skull Island	PG13	Vivocity	south	1130
Logan	M18	Vivocity	south	1700
Silent Voice	PG	Lido	central	1400
Kong: Skull Island	PG13	Lido	central	2130

- Each movie has a rating: $\{\text{movie}\} \rightarrow \{\text{rating}\}$
- Each cinema is located in one area: $\{\text{cinema}\} \rightarrow \{\text{area}\}$
- Each cinema screens at most one movie at any time: $\{\text{cinema, time}\} \rightarrow \{\text{movie}\}$

Schema Refinement (cont.)

MovieList

movie	rating	cinema	area	time
John Wick 2	M18	JCube	west	1400
Logan	M18	JCube	west	1700
Kong: Skull Island	PG13	Vivocity	south	1130
Logan	M18	Vivocity	south	1700
Silent Voice	PG	Lido	central	1400
Kong: Skull Island	PG13	Lido	central	2130

- Problems with this schema design:
  - **Insertion anomaly**: We can't store information about a new movie if the screening cinema and time are not known
  - **Deletion anomaly**: If we delete all M18-rated movies, we lose information about the cinema JCube
  - **Update anomaly**: If the rating of a movie changes, we have to be careful of inconsistent updates

Schema Refinement (cont.)

MovieList

movie	rating	cinema	area	time
John Wick 2	M18	JCube	west	1400
Logan	M18	JCube	west	1700
Kong: Skull Island	PG13	Vivocity	south	1130
Logan	M18	Vivocity	south	1700
Silent Voice	PG	Lido	central	1400
Kong: Skull Island	PG13	Lido	central	2130

Movies

movie	rating
John Wick 2	M18
Logan	M18
Kong: Skull Island	PG13
Silent Voice	PG

Cinemas

cinema	area
JCube	west
Vivocity	south
Lido	central

Screenings

cinema	time	movie
JCube	1400	John Wick 2
JCube	1700	Logan
Vivocity	1130	Kong: Skull Island
Lido	1400	Silent Voice
Lido	2130	Kong: Skull Island

Trivial, Non-trivial, Completely Non-trivial FDs

- Consider $R = \{A, B, C\}$ with FDs $F = \{\{A\} \rightarrow \{B\}, \{B\} \rightarrow \{C\}\}$

Completely Non-trivial FDs in F^+

$\{A\} \rightarrow \{B\}$	$\{B\} \rightarrow \{C\}$
$\{A\} \rightarrow \{C\}$	$\{A, B\} \rightarrow \{C\}$
$\{A\} \rightarrow \{B, C\}$	$\{A, C\} \rightarrow \{B\}$

Remaining Non-trivial FDs in F^+

$\{A\} \rightarrow \{A, B\}$	$\{A, B\} \rightarrow \{B, C\}$
$\{A\} \rightarrow \{A, C\}$	$\{A, B\} \rightarrow \{A, B, C\}$
$\{A\} \rightarrow \{A, B, C\}$	$\{A, C\} \rightarrow \{A, B\}$
$\{B\} \rightarrow \{B, C\}$	$\{A, C\} \rightarrow \{B, C\}$
$\{A, B\} \rightarrow \{A, C\}$	$\{A, C\} \rightarrow \{A, B, C\}$

Trivial FDs in F^+

$\emptyset \rightarrow \emptyset$	$\{B, C\} \rightarrow \emptyset$
$\{A\} \rightarrow \emptyset$	$\{B, C\} \rightarrow \{B\}$
$\{A\} \rightarrow \{A\}$	$\{B, C\} \rightarrow \{C\}$
$\{B\} \rightarrow \emptyset$	$\{B, C\} \rightarrow \{B, C\}$
$\{B\} \rightarrow \{B\}$	$\{A, B, C\} \rightarrow \emptyset$
$\{C\} \rightarrow \emptyset$	$\{A, B, C\} \rightarrow \{A\}$
$\{C\} \rightarrow \{C\}$	$\{A, B, C\} \rightarrow \{B\}$
$\{A, B\} \rightarrow \emptyset$	$\{A, B, C\} \rightarrow \{C\}$
$\{A, B\} \rightarrow \{A\}$	$\{A, B, C\} \rightarrow \{A, B\}$
$\{A, B\} \rightarrow \{B\}$	$\{A, B, C\} \rightarrow \{A, C\}$
$\{A, B\} \rightarrow \{A, B\}$	$\{A, B, C\} \rightarrow \{B, C\}$
$\{A, C\} \rightarrow \emptyset$	$\{A, B, C\} \rightarrow \{A, B, C\}$
$\{A, C\} \rightarrow \{A\}$	
$\{A, C\} \rightarrow \{C\}$	
$\{A, C\} \rightarrow \{A, C\}$	

Reasoning about FDs

- Let R be a relational schema & F be a set of FDs that hold on R
- Closure of F (denoted by F^+) = set of all FDs that must hold on R given F
  - A FD $X \rightarrow Y \in F^+$ iff F implies $X \rightarrow Y$
- Two sets of FDs F & G are equivalent (denoted by $F \equiv G$) if $F^+ = G^+$
- **Armstrong's Axioms**: Let X, Y, Z be subsets of attributes in relational scheme R

Reflexivity:	If $Y \subseteq X$, then $X \rightarrow Y$
Augmentation:	If $X \rightarrow Y$, then $X \cup Z \rightarrow Y \cup Z$
Transitivity:	If $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$

Reasoning about FDs (cont.)

- Let X be a subset of attributes of relational scheme R with FDs F
- Attribute closure of X (denoted by X^+) is the set of all attributes of R that are functionally determined by X (w.r.t. F)
  - $X^+ = \{A | A \text{ is an attribute of } R, X \rightarrow \{A\} \in F^+\}$
- $X \rightarrow Y \in F^+$ iff $Y \subseteq X^+$ (w.r.t. F)

Superkeys, Candidate Keys & Prime Attributes

- Let X be a subset of attributes of relational scheme R with FDs F
- X is a superkey of R if $X^+ = R$
- X is a candidate key of R if
  (1) X is a superkey of R, and
  (2) $\forall Y \subset X,\ Y^+ \neq R$
- An attribute $A \in R$ is a prime attribute if A is contained in some candidate key of R; otherwise, it is a nonprime attribute

Minimal Covers

- Let R be a relational schema with FDs F
- A minimal cover for F is a set G of FDs with the following properties
  1. $G \equiv F$
  2. For every FD $X \rightarrow Y$ in G, Y consists of a single attribute
  3. For every FD $X \rightarrow Y$ in G, X has no redundant attributes
     - An attribute $A \in X$ is redundant in the FD $X \rightarrow Y$ if $(G - \{X \rightarrow Y\}) \cup \{X - \{A\} \rightarrow Y\} \equiv G$
     - $A \in X$ is redundant in the FD $X \rightarrow Y$ if $Y \subseteq (X - \{A\})^+$ w.r.t. G
  4. There are no redundant FDs in G
     - A FD $X \rightarrow Y \in G$ is redundant if $G - \{X \rightarrow Y\} \equiv G$
     - $X \rightarrow \{B\} \in G$ is redundant if $B \in X^+$ w.r.t. $G - \{X \rightarrow \{B\}\}$
- A extended minimal cover is obtained from a minimal cover by combining FDs with the same left-hand side

Schema Decompositions

- The **decomposition of schema** R is a set of schemas $\{R_1, R_2, \cdots, R_n\}$ such that each $R_i \subseteq R$ and $R = R_1 \cup R_2 \cup \cdots \cup R_n$
- Each R_i is called a fragment

MovieList

movie	rating	cinema	area	time
John Wick 2	M18	JCube	west	1400
Logan	M18	JCube	west	1700
Kong: Skull Island	PG13	Vivocity	south	1130
Logan	M18	Vivocity	south	1700
Silent Voice	PG	Lido	central	1400
Kong: Skull Island	PG13	Lido	central	2130

Movies

movie	rating
John Wick 2	M18
Logan	M18
Kong: Skull Island	PG13
Silent Voice	PG

Cinemas

cinema	area
JCube	west
Vivocity	south
Lido	central

Screenings

cinema	time	movie
JCube	1400	John Wick 2
JCube	1700	Logan
Vivocity	1130	Kong: Skull Island
Lido	1400	Silent Voice
Lido	2130	Kong: Skull Island

Properties of Schema Decompositions

- Two important properties:
  1. Decomposition must **preserve information**
     - Data in original relation = Data in decomposed relations
     - Crucial for correctness!
  2. Decomposition should **preserve FDs**
     - FDs in original schema $\equiv$ FDs in decomposed schemas
     - Facilitates checking of FD violations without performing joins

MovieList

movie	rating	cinema	area	time
John Wick 2	M18	JCube	west	1400
Logan	M18	JCube	west	1700
Kong: Skull Island	PG13	Vivocity	south	1130
Logan	M18	Vivocity	south	1700
Silent Voice	PG	Lido	central	1400
Kong: Skull Island	PG13	Lido	central	2130

Movies

movie	rating
John Wick 2	M18
Logan	M18
Kong: Skull Island	PG13
Silent Voice	PG

Cinemas

cinema	area
JCube	west
Vivocity	south
Lido	central

Screenings

cinema	time	movie
JCube	1400	John Wick 2
JCube	1700	Logan
Vivocity	1130	Kong: Skull Island
Lido	1400	Silent Voice
Lido	2130	Kong: Skull Island

Lossless Decompositions

- It is important that a decomposition *preserves information*; i.e., we can reconstruct the relation from its decomposed fragments

> **Property**: If $\{R_1, R_2, \cdots, R_n\}$ is a decomposition of R, then for any relation r of R, $r \subseteq \pi_{R_1}(r) \bowtie \pi_{R_2}(r) \bowtie \cdots \bowtie \pi_{R_n}(r)$

- A decomposition of R (with FDs F) into $\{R_1, R_2, \cdots, R_n\}$ is a lossless decomposition if for every relation r of R that satisfies F, $\pi_{R_1}(r) \bowtie \pi_{R_2}(r) \bowtie \cdots \bowtie \pi_{R_n}(r) = r$
- A decomposition that is not lossless is a lossy decomposition

Lossless Decompositions: Example

- Consider $R(A, B, C)$ with FDs $F = \{\{A\} \rightarrow \{B\}\}$
- **Example 1**: Decomposition of R into $\{R_1(A, B), R_2(B, C)\}$ is lossy

r

A	B	C
a_1	b_1	c_1
a_2	b_1	c_2

r_1

A	B
a_1	b_1
a_2	b_1

r_2

B	C
b_1	c_1
b_1	c_2

$r_1 \bowtie r_2$

A	B	C
a_1	b_1	c_1
a_1	b_1	c_2
a_2	b_1	c_1
a_2	b_1	c_2

- **Example 2**: Decomposition of R into $\{R_1(A, B), R_3(A, C)\}$ is lossless

r

A	B	C
a_1	b_1	c_1
a_2	b_1	c_2

r_1

A	B
a_1	b_1
a_2	b_1

r_3

A	C
a_1	c_1
a_2	c_2

$r_1 \bowtie r_3$

A	B	C
a_1	b_1	c_1
a_2	b_1	c_2

Lossless Decompositions

Property 1: The decomposition of R (with FDs F) into $\{R_1, R_2\}$ is a lossless decomposition (w.r.t. F) if and only if
$R_1 \cap R_2 \rightarrow R_1 \in F^+$ or $R_1 \cap R_2 \rightarrow R_2 \in F^+$

Property 2: If $X \rightarrow Y$ is a completely non-trivial FD that holds on R, then the decomposition of R into $\{R - Y, X \cup Y\}$ is a lossless decomposition

Property 3: If $\{R_1, R_2, \cdots, R_n\}$ is a lossless decomposition of R, and $\{R_a, R_b\}$ is a lossless decomposition of R_1, then $\{R_a, R_b, R_2, \cdots, R_n\}$ is a lossless decomposition of R

Projected Functional Dependencies

- Let $\{R_1, \cdots, R_n\}$ be a decomposition of R with FDs F
- Let F_i denote the set of FDs that hold on R_i

$$F_i = \{X \rightarrow Y \in F^+ \mid X, Y \subseteq R_i\}$$

- Each F_i is a set of projected functional dependencies for fragment R_i

Dependency Preserving Decompositions

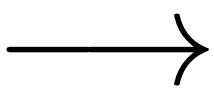
- A decomposition $\{R_1, \cdots, R_n\}$ of R is dependency preserving if $F_1 \cup \cdots \cup F_n \equiv F$
  - If there exists a FD $X \rightarrow Y \in F$ such that $X \rightarrow Y \notin (F_1 \cup \cdots \cup F_n)^+$, then $\{R_1, \cdots, R_n\}$ is not a dependency preserving of R; otherwise, $\{R_1, \cdots, R_n\}$ is a dependency preserving decomposition of R
- A dependency-preserving decomposition is desirable because it guarantees that for each update to a decomposed relation r_i, FD violations for r_i can be detected without computing joins

Example: $R(A, B, C)$ with FDs $F = \{\{B\} \rightarrow \{C\}, \{A, C\} \rightarrow \{B\}\}$

- The decomposition $\{R_1(A, B), R_2(B, C)\}$ is not dependency-preserving as $\{A, C\} \rightarrow \{B\}$ is not preserved
  - The completely non-trivial FDs in $F_1 = \emptyset$
  - The completely non-trivial FDs in $F_2 = \{\{B\} \rightarrow \{C\}\}$
  - $(F_1 \cup F_2)^+ \equiv \{\{B\} \rightarrow \{C\}\} \not\equiv F$
- Consider the following instance of R:

r

A	B	C
a_1	b_1	c_1
a_2	b_2	c_1



r_1

A	B
a_1	b_1
a_2	b_2

r_2

B	C
b_1	c_1
b_2	c_1

- Inserting the new tuple (a_1, b_2, c_1) into r would violate $\{A, C\} \rightarrow \{B\}$
- But inserting (a_1, b_2) into r_1 does not violate any FD in F_1 & inserting (b_2, c_1) into r_2 does not violate any FD in F_2
- Need to compute $r_1 \bowtie r_2$ to detect violations of $\{A, C\} \rightarrow \{B\}$!

Normal Forms

- A normal form restricts the set of data dependencies that are allowed to hold on a schema to avoid certain undesirable redundancy and update problems in the database
- Two normal forms based on FDs: **Boyce-Codd Normal Form (BCNF)** and **Third Normal Form (3NF)**
- Let R be a relational schema with FDs F
- R is in BCNF if for every non-trivial FD $X \rightarrow \{A\}$ in F^+, X is a superkey of R
- A non-trivial FD $X \rightarrow \{A\} \in F^+$ violates BCNF of R if X is not a superkey of R
- R is in 3NF if for every non-trivial FD $X \rightarrow \{A\}$ in F^+, either X is a superkey of R or A is a prime attribute of R
- A non-trivial FD $X \rightarrow \{A\} \in F^+$ violates 3NF of R if X is not a superkey of R and A is a nonprime attribute of R

BCNF Decomposition Algorithm

Input: A lossless decomposition $S = \{R_1, \cdots, R_n\}$ of R with FDs F
Output: A lossless BCNF decomposition of R

01. while (not every R_i in S is in BCNF)
02. let $R_i \in S$ be a schema that is not in BCNF w.r.t. F_i
03. let $X \rightarrow Y \in F_i$ be a completely non-trivial FD that violates BCNF of R_i
04. let $R' = X^+$ (w.r.t. F_i) and $R'' = R_i - R' \cup X$
05. $S = S - \{R_i\} \cup \{R', R''\}$
06. return S

3NF Synthesis Algorithm

Input: Schema R with FDs F
Output: A dependency-preserving, lossless 3NF decomposition of R

01. Initialize $S = \emptyset$
02. let F' be an extended minimal cover of F
03. for each FD $X \rightarrow Y$ in F'
04. if no relational schema in S contains $X \cup Y$ then
05. insert a relational schema $X \cup Y$ into S
06. if no relational schema in S contains a candidate key of S then
07. insert a relational schema K into S where K is any candidate key of R
08. return S

Tuple Relational Calculus (TRC)

- An **atomic formula** has of one the following forms:
  1. $R \in Rel$
  2. $R.a\ op\ S.b$
  3. $R.a\ op\ constant$

  where R & S are tuple variables, Rel is some relation, and op is one of $<, >\leq, \geq, =, \neq$
- A **formula** has one of the following forms:
  1. an atomic formula
  2. $p \wedge q, \quad p \vee q, \quad \neg p, \quad p \Rightarrow q$
  3. $\exists\, R\,(p)$
  4. $\forall\, R\,(p)$

  where p and q are formulae; and R is a free variable in p
- A TRC query is of the form $\{T | p\}$ where T must be the only free variable in the formula p

Equivalence Rules

1. $\neg\neg p \equiv p, \quad \neg(p \wedge q) \equiv \neg p \vee \neg q, \quad \neg(p \vee q) \equiv \neg p \wedge \neg q$
2. $p \Rightarrow q \;\equiv\; \neg p \vee q$
3. $\neg(\exists R(p)) \equiv \forall R(\neg p)$
4. $\neg(\forall R(p)) \equiv \exists R(\neg p)$
5. $\forall R(p \wedge q) \equiv (\forall R(p)) \wedge (\forall R(q))$
6. $\exists R(p \vee q) \equiv (\exists R(p)) \vee (\exists R(q))$
7. If R is not a free variable in p, then
   - 7.1 $p \vee (\forall R(q)) \equiv \forall R(p \vee q)$
   - 7.2 $p \vee (\exists R(q)) \equiv \exists R(p \vee q)$
   - 7.3 $p \wedge (\forall R(q)) \equiv \forall R(p \wedge q)$
   - 7.4 $p \wedge (\exists R(q)) \equiv \exists R(p \wedge q)$
   - 7.5 $p \Rightarrow (\forall R(q)) \equiv \forall R(p \Rightarrow q)$
   - 7.6 $p \Rightarrow (\exists R(q)) \equiv \exists R(p \Rightarrow q)$

Order of Quantifiers

- $\exists\, X \exists\, Y(p) \equiv \exists\, Y \exists\, X(p)$
- $\forall\, X \forall\, Y(p) \equiv \forall\, Y \forall\, X(p)$
- $\exists\, X \forall\, Y(p) \not\equiv \forall\, Y \exists\, X(p)$

Example Database

`Customers` (cname, area)
`Restaurants` (rname, area)
`Contains` (pizza, ingredient)
`Sells` (rname, pizza, price)
`Likes` (cname, pizza)

$\exists X(p \wedge q)$ vs $\exists X(p \Rightarrow q)$

Question 1

Find restaurants located in the West area that sell some pizza that is liked by some customer.

Which of the following answers is correct?

Answer 1

$$\{T \mid \exists\, R(R \in \mathit{Restaurants} \wedge T.\mathit{rname} = R.\mathit{rname} \wedge R.\mathit{area} = \text{“}\mathit{West}\text{”} \wedge \exists\, S(S \in \mathit{Sells} \wedge S.\mathit{rname} = R.\mathit{rname} \wedge \exists\, L(L \in \mathit{Likes} \wedge L.\mathit{pizza} = S.\mathit{pizza})))\}$$

Answer 2

$$\{T \mid \exists\, R(R \in \mathit{Restaurants} \wedge T.\mathit{rname} = R.\mathit{rname} \wedge R.\mathit{area} = \text{“}\mathit{West}\text{”} \wedge \exists\, S(S \in \mathit{Sells} \wedge S.\mathit{rname} = R.\mathit{rname} \Rightarrow \exists\, L(L \in \mathit{Likes} \wedge L.\mathit{pizza} = S.\mathit{pizza})))\}$$

$\forall X(p \wedge q)$ vs $\forall X(p \Rightarrow q)$

Question 2

Find customers who like all the pizzas sold by Corleone.

Which of the following answers is correct?

Answer 1

$$\{T \mid \exists\, C(C \in \mathit{Customers} \wedge T.\mathit{cname} = C.\mathit{cname} \wedge$$
$$\forall\, S(S \in \mathit{Sells} \wedge S.\mathit{rname} = \text{“}\mathit{Corleone}\text{”} \wedge$$
$$\exists\, L(L \in \mathit{Likes} \wedge L.\mathit{cname} = C.\mathit{cname} \wedge L.\mathit{pizza} = S.\mathit{pizza})))\}$$

Answer 2

$$\{T \mid \exists\, C(C \in \mathit{Customers} \wedge T.\mathit{cname} = C.\mathit{cname} \wedge$$
$$\forall\, S(S \in \mathit{Sells} \wedge S.\mathit{rname} = \text{“}\mathit{Corleone}\text{”} \Rightarrow$$
$$\exists\, L(L \in \mathit{Likes} \wedge L.\mathit{cname} = C.\mathit{cname} \wedge L.\mathit{pizza} = S.\mathit{pizza})))\}$$

SQL

- Schema constraints (NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY, CHECK, DEFAULT)
- Set Operations (UNION [ALL], INTERSECT [ALL], EXCEPT [ALL])
- Join operations (natural, inner, left/right/full outer)
- Comparison predicates (IS NULL, IS DISTINCT FROM)
- Subquery expressions (EXISTS, IN, ANY/SOME, ALL, UNIQUE)
- Aggregate functions (COUNT, SUM, AVG, MIN, MAX)
- Table Expressions & Common Table Expressions
- Views

Conceptual Evaluation of SQL Queries

select	**distinct** select-list
from	from-list
where	where-condition
group by	groupby-list
having	having-condition
order by	orderby-list
limit	limit-specification

1. Compute the cross-product of the tables in **from-list**
2. Select the tuples in the cross-product that evaluate to *true* for the **where-condition**
3. Partition the selected tuples into groups using the **groupby-list**
4. Select the groups that evaluate to *true* for the **having-condition** condition
5. For each selected group, generate an output tuple by selecting/computing the attributes/expressions that appear in the **select-list**
6. Remove any duplicate output tuples
7. Sort the output tuples based on the **orderby-list**
8. Remove the appropriate output tuples based on the **limit-specification**

GROUP BY Clause: Properties

- Each output tuple corresponds to one group
- For each column *A* in relation *R* that appears in the `SELECT` clause, one of the following conditions must hold:
  1. *A* appears in the `GROUP BY` clause,
  2. *A* appears in an aggregated expression in the `SELECT` clause (e.g., **min**(A)), or
  3. the primary (or a candidate) key of *R* appears in the `GROUP BY` clause
- If an aggregate function appears in the `SELECT` clause and there is no `GROUP BY` clause, then the `SELECT` clause must not contain any column that is not in an aggregated expression

HAVING Clause: Properties

- For each column *A* in relation *R* that appears in the `HAVING` clause, one of the following conditions must hold:
  1. *A* appears in the `GROUP BY` clause,
  2. *A* appears in an aggregated expression in the `HAVING` clause, or
  3. the primary (or a candidate) key of *R* appears in the `GROUP BY` clause

Relational Algebra

Unary operators

- Selection σ
- Projection π
- Renaming ρ

Binary operators

- Union $\cup$
- Intersection $\cap$
- Difference $-$
- Cross-Product $\times$
- Natural Join $\bowtie$
- Inner Join $\bowtie_c$
- Left Outer Join $\rightarrow_c$
- Full Outer Join $\leftrightarrow_c$

Operators	SQL
$\sigma_c(R)$	**select** $*$ **from** R **where** c
$\pi_\ell(R)$	**select distinct** ℓ **from** R
$R \cup S$	**select** $*$ **from** R **union select** $*$ **from** S
$R \cap S$	**select** $*$ **from** R **intersect select** $*$ **from** S
$R - S$	**select** $*$ **from** R **except select** $*$ **from** S
$R \times S$	**select** $*$ **from** R **cross join** S
$R \bowtie S$	**select** $*$ **from** R **natural join** S
$R \bowtie_c S$	**select** $*$ **from** R **join** S **on** c

Transactions: Serializable Executions

- Consider a set of transactions $S = \{T_1, \cdots, T_n\}$
- An execution of S is serializable if it is equivalent to some serial execution of S
- Let E_1 & E_2 denote two executions of S
- E_1 and E_2 are equivalent executions of S if
  1. both executions produce the same final database state, &
  2. both executions retrieve the same values: for every value read by some T_i in E_1, the corresponding read by T_i in E_2 returns the same value
- **Serializable executions** guarantee correctness of transaction executions