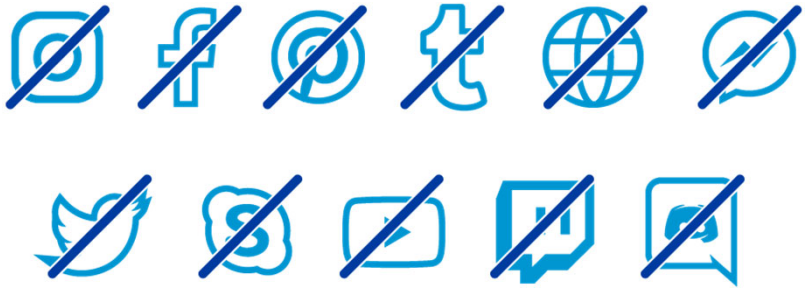
DATABASES

Source: https://en.itpedia.nl/2017/11/26/wat-is-een-database/

Prof. Dr. Ulrike Herster
Hamburg University of Applied Sciences

HAW HAMBURG

COPYRIGHT



The publication and sharing of
slides, images and sound recordings of this
course is not permitted

ORGANIZATION
OUR JOURNEY IN THIS SEMESTER

Repetition

- Integrity, Trigger & Security
- Database Applications
- Transactions
- Subqueries & Views
- **More SQL**
- Notations & Guidelines
- Constraints
- Relationships
- Simple Entities and Attributes
- Basics

Source: Foto von Justin Kauffman auf Unsplash

MORE SQL
RELATIONAL ALGEBRA


Repetition

- SQL → What!
  Relational Algebra → How!

- In mathematics an algebra is a values range combined with defined operations

- *Relational Algebra*: The values range is the content of the database;
  operations are functions to calculate the query results
  → a set of operations for the relational model

- *Relational Calculus*: Descriptive approach that is based on mathematical logic
  → higher-level declarative language for specifying relational queries,
  e.g., no order of operations, only what information the result should contain

Source: Elmasri, Fundamentals of Database Systems, Page 145ff

MORE SQL
RELATIONAL ALGEBRA: OPERATIONS


Repetition

Operation	Purpose	Notation
Selection	Selects all tuples that satisfy the selection condition from a relation R	$\sigma_{<selection\ condition>}(R)$
Projection	Produces a new relation with only some of the attributes of R, and removes duplicate tuples	$\pi_{<attribute\ list>}(R)$
Renaming	Column in the result relation gets new name	$\rho_{new\ name \leftarrow attribute\ name}(R)$
Join	Produces all combinations of tuples from R_1 and R_2 that satisfy the join condition	$R_1 \bowtie_{<join\ condition>} R_2$
Equijoin	Produces all the combinations of tuples from R_1 and R_2 that satisfy a join condition with only equality comparisons	$R_1 *_{<join\ condition>} R_2$

Source: Elmasri, Fundamentals of Database Systems, Page 145ff

MORE SQL
RELATIONAL ALGEBRA: OPERATIONS

Repetition

Operation	Purpose	Notation
Union	Produces a relation that includes all the tuples in R_1 or R_2 or both R_1 and R_2; R_1 and R_2 must be union compatible	$R_1 \cup R_2$
Intersection	Produces a relation that includes all the tuples in both R_1 and R_2; R_1 and R_2 must be union compatible	$R_1 \cap R_2$
Set Difference	Produces a relation that includes all the tuples in R_1 that are not in R_2; R_1 and R_2 must be union compatible	$R_1 - R_2$
Cartesian Product	Produces a relation that has the attributes of R_1 and R_2 and includes as tuples all possible combinations of tuples from R_1 and R_2	$R_1 \times R_2$

Source: Elmasri, Fundamentals of Database Systems, Page 145ff

ORGANIZATION
OUR JOURNEY IN THIS SEMESTER

Repetition

- Integrity, Trigger & Security
- Database Applications
- Transactions
- **Subqueries & Views**
- More SQL
- Notations & Guidelines
- Constraints
- Relationships
- Simple Entities and Attributes
- Basics

Source: Foto von Justin Kauffman auf Unsplash

SUBQUERIES AND VIEWS
SUBQUERIES


Repetition

- **SELECT** returns relation: a (multi-)set

- Result of **SELECT** can be included in query
  - **WHERE** clause
    → also, for **UPDATE**, **DELETE**
  - **HAVING** clause
  - **FROM** clause
  - **SELECT** clause (in column list)

- So, we have two (or more) **SELECT**s:
  - Outer **SELECT**
  - Nested (or inner) **SELECT**: *subquery*

SUBQUERIES AND VIEWS
SUBQUERIES – OPERATORS IN WHERE

- **IN**

```
SELECT E.Fname, E.Lname
FROM   EMPLOYEE AS E
WHERE  E.Ssn IN (   SELECT Essn
                    FROM   DEPENDENT AS D
                    WHERE  E.Sex = D.Sex );
```

- **=**

```
SELECT *
FROM y
WHERE x = ( SELECT MAX(x) FROM y ) ;
```

- **ANY**

```
SELECT name
FROM   Person
WHERE  PNr = ANY (SELECT PNr FROM book ) ;
```

- **ALL**

```
SELECT Lname, Fname
FROM   EMPLOYEE
WHERE  Salary > ALL
       (SELECT       Salary
       FROM          EMPLOYEE
       WHERE         Dno=5) ;
```

- **EXISTS**

```
SELECT E.Fname, E.Lname
FROM   EMPLOYEE AS E
WHERE  EXISTS
             (SELECT *
             FROM   DEPENDENT AS D
             WHERE  E.Ssn = D.Essn
                    AND E.Sex = D.Sex);
```

- **NOT EXISTS**

```
SELECT Fname, Lname
FROM   EMPLOYEE
WHERE  NOT EXISTS ( SELECT *
                    FROM   DEPENDENT
                    WHERE  Ssn=Essn );
```

SUBQUERIES AND VIEWS
SUBQUERIES – IN FROM


Repetition

- **SELECT** returns a new relation
- ... so, we can select values from it
- Necessary: give a name to the relation

- Example: Alias name newtab_b

```
SELECT tab_a.x , newtab_b.y
FROM   tab_a , (SELECT v1, v2 FROM tab_b) AS newtab_b ;
```

SUBQUERIES AND VIEWS
SUBQUERIES: ASSIGNMENT

- Suppose that we want the department name number of employees in each department whose departments have an average income of more than 30,000 $. How can we specify this query in SQL?
- Retrieve the names of all employees with the smallest salary.
- Retrieve the names of all employees whose supervisor's supervisor has ssn '888665555'.
- Retrieve the names of employees who make at least $10,000 more than the employee who is paid the least in the company.

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

WORKS_ON

Essn	Pno	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	NULL

PROJECT

Pname	Pnumber	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship
333445555	Alice	F	1986-04-05	Daughter
333445555	Theodore	M	1983-10-25	Son
333445555	Joy	F	1958-05-03	Spouse
987654321	Abner	M	1942-02-28	Spouse
123456789	Michael	M	1988-01-04	Son
123456789	Alice	F	1988-12-30	Daughter
123456789	Elizabeth	F	1967-05-05	Spouse

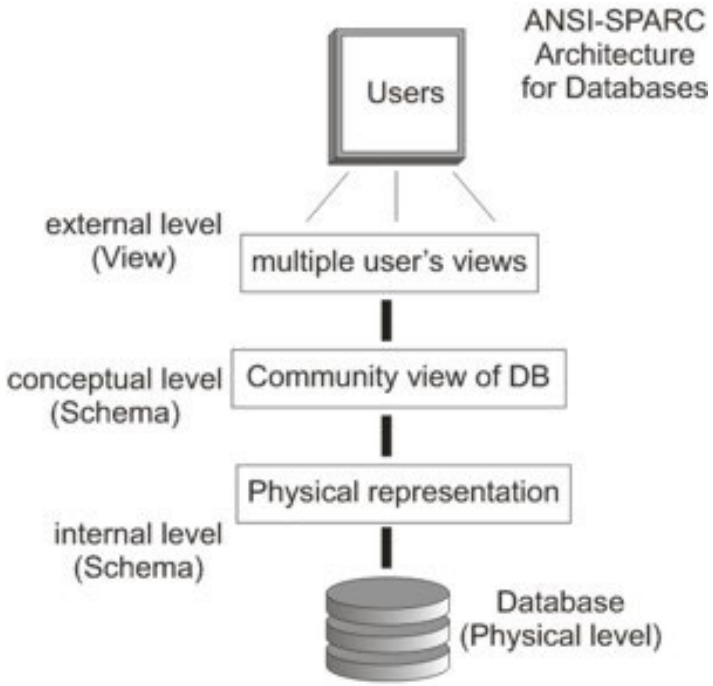
SUBQUERIES AND VIEWS
VIEWS – RECAP: THE ANSI-SPARC ARCHITECTURE

Repetition

External Schema

Conceptual Schema

Internal Schema



Source: www.wikipedia.org

SUBQUERIES AND VIEWS
VIEWS – BASICS


Repetition

- User or application specific views on data
- Only relevant portions of the data

- A *view* in SQL terminology is a single table that is derived from other tables
  → Other tables can be *base tables* or previously defined views

- A view is considered to be a *virtual table*
  → In contrast to base tables
  → Limits the possible update operations
  → No limitations on querying a view

Source: Elmasri, Fundamentals of Database Systems, Page 115ff

SUBQUERIES AND VIEWS
VIEWS – CREATE

- Example:

```
CREATE VIEW vPerson AS
   SELECT Name , Id , BirthDate FROM person ;
```

vPerson	Name	Id	BirthDate

Can rename columns in view:

```
CREATE VIEW vPerson ( lname , pnr , bd ) AS
   SELECT Name , Id , BirthDate FROM person
```

vPerson	lname	pnr	bd

SUBQUERIES AND VIEWS
VIEWS – UPDATING VIEWS

- Views are Relations
  → ... just like tables

- Should make no difference to users

- Question:
  Can we modify the view's data?
  → Depends on type of view!

SUBQUERIES AND VIEWS
VIEWS – UPDATING VIEWS

- Example:


```
UPDATE v_WORKS_ON1
SET    Pname = "Project2"
WHERE  Lname= "Borg"
       AND Fname= "James"
       AND Pname= "Project2";
```

Source: Elmasri, Fundamentals of Database Systems, Page 115ff

VIEWS – UPDATING VIEWS

Example:

- Possible update 1 (*Query Modification)*:

```
UPDATE WORKS_ON
SET    Pno=   SELECT Pnumber
              FROM   PROJECT
              WHERE  Pname= "Project2")
WHERE  Essn IN (SELECT Ssn
              FROM   EMPLOYEE
              WHERE  Lname= "Borg" AND Fname= "James" )
       AND
       Pno= ( SELECT Pnumber
              FROM   PROJECT
              WHERE  Pname= "Project1" );
```

Source: Elmasri, Fundamentals of Database Systems, Page 115ff

SUBQUERIES AND VIEWS
VIEWS – UPDATING VIEWS

Example:

- Possible update 2 (*View Materialization)*:

```
UPDATE PROJECT
SET    Pname = "Project2"
WHERE  Pname = "Project1";
```

Source: Elmasri, Fundamentals of Database Systems, Page 115ff

SUBQUERIES AND VIEWS
VIEWS – UPDATING VIEWS

- Classify views based on the select:
  - *Projection View*
    - `SELECT a, b, c …`
  - *Selection View*
    - `… WHERE < condition > …`
  - *Join View*
    - `… FROM tab_a JOIN tab_b …`
  - *Aggregation View*
    - `SELECT MAX(x) …`
- Other types and combinations exist

SUBQUERIES AND VIEWS
VIEWS – UPDATING VIEWS

- *Projection View*
  - Example:
    … **AS SELECT** a , b , c **FROM** …
  - Manipulations can be transformed to base table quite easily

EMPLOYEE	Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

  - Problems:

VEMPLOYEE	Fname	Minit	Lname	Ssn	Salary	Dno

    - **INSERT**
      → **NOT NULL** columns in base table
    - **DELETE**
      → Problem if projection does not contain primary key
    - In general: Can violate integrity constraints of base table

SUBQUERIES AND VIEWS
VIEWS – UPDATING VIEWS

- *Selection View*
  - Example:
    ```
    CREATE VIEW v_top AS SELECT * FROM employee WHERE salary > 20000;
    ```
  - Problem:
    - Update can move tuples out of selection condition
    - So, the update looks like a delete!
  - Example:
    ```
    UPDATE v_top SET salary = 100;
    ```
  - This phenomenon is called "*tuple migration*"

VIEWS – UPDATING VIEWS

- *Join View*
  - Example:

```
CREATE VIEW v_depman AS
  SELECT *
  FROM   employee , department
  WHERE   employee.ssn = department.mgr_ssn ;
```

  - Data manipulation cannot be transformed to base tables in general case!

```
DELETE FROM v_depman WHERE id =11;
```

  - Transformation to base tables `employee` and `department`?
    → **`DELETE FROM`** `employee` ?
    → **`DELETE FROM`** `department` ?

VIEWS – UPDATING VIEWS

- *Aggregation View*
  - Example:

```
CREATE VIEW v_astats AS
  SELECT MAX(i) , MIN(i) , COUNT(*)
  FROM a ;
```

  - Update of the aggregated columns not possible!
  - Note: Aggregation may depend on other columns (**GROUP BY**)

SUBQUERIES AND VIEWS
VIEWS – UPDATING VIEWS

- A view with a single defining table is updatable if
  - the view attributes contain the primary key of the base relation,
  - as well as all attributes with the `NOT NULL` constraint that have a default value specified

- Views defined on multiple tables using joins are only updatable in special cases
  E.g., **`INSERT`** and **`UPDATE`** for Join Views, if join condition is based on PK-FK

- Views defined using grouping and aggregate functions are not updatable

Source: Elmasri, Fundamentals of Database Systems, Page 115ff

SUBQUERIES AND VIEWS
VIEWS – UPDATING VIEWS

- Oracle and standard SQL allow certain options at end of **`VIEW`** definition:
  - … **`WITH READ ONLY ;`**
    → Read only view, no data manipulation allowed
  - … **`WITH CHECK OPTION;`**
    → Updates leading to tuple migration are denied

SUBQUERIES AND VIEWS
GENERATED TABLES

- Syntax:
  CREATE TABLE <name> **AS SELECT** …
- Can create new table based on query
- New table is independent from old table
- Use cases:
  - Copy table
  - Copy parts of table
- Attention: New table does not have all constraints of the parent table!

SUBQUERIES AND VIEWS
GENERATED TABLES

Example from before:

```
INSERT INTO  Underpaid ( lname , fname )
       SELECT lname , fname
       FROM Employee
       WHERE salary < 1000 ;
```

→ **WHERE** clause belongs to **SELECT**

SUBQUERIES AND VIEWS
VIEWS: ASSIGNMENT

- Create a view that has the department name, manager name, and manager salary for every department.
- Create a view that has the project name, controlling department name, number of employees, and total hours worked per week on the project for each project.
- Create a view that has the project name, controlling department name, number of employees, and total hours worked per week on the project for each project with more than one employee working on it.

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

WORKS_ON

Essn	Pno	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	NULL

PROJECT

Pname	Pnumber	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship
333445555	Alice	F	1986-04-05	Daughter
333445555	Theodore	M	1983-10-25	Son
333445555	Joy	F	1958-05-03	Spouse
987654321	Abner	M	1942-02-28	Spouse
123456789	Michael	M	1988-01-04	Son
123456789	Alice	F	1988-12-30	Daughter
123456789	Elizabeth	F	1967-05-05	Spouse

SUBQUERIES AND VIEWS
VIEWS: ASSIGNMENT

Consider the following view v_DEPT_SUMMARY defined on the COMPANY database:

```
CREATE VIEW v_DEPT_SUMMARY
 (DNO, COUNT_EMPS, SUM_SALARY, AVG_SALARY)
 AS SELECT Dno, COUNT(*), SUM(Salary),
 AVG(Salary)
 FROM EMPLOYEE
 GROUP BY Dno;
```

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

WORKS_ON

Essn	Pno	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	NULL

PROJECT

Pname	Pnumber	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship
333445555	Alice	F	1986-04-05	Daughter
333445555	Theodore	M	1983-10-25	Son
333445555	Joy	F	1958-05-03	Spouse
987654321	Abner	M	1942-02-28	Spouse
123456789	Michael	M	1988-01-04	Son
123456789	Alice	F	1988-12-30	Daughter
123456789	Elizabeth	F	1967-05-05	Spouse

SUBQUERIES AND VIEWS
VIEWS: ASSIGNMENT

Describe the semantics oft he following SQL -statements. State which of the following queries and updates would be allowed on the view. If a query or update would be allowed, show what the corresponding query or update on the base relations would look like, and give its result when applied to the database.

1. `SELECT * FROM v_DEPT_SUMMARY;`
2. `SELECT DNO, COUNT_EMPS FROM v_DEPT_SUMMARY WHERE SUM_SALARY > 100000;`
3. `SELECT DNO, AVG_SALARY FROM v_DEPT_SUMMARY WHERE COUNT_EMPS > (SELECT COUNT_EMPS FROM v_DEPT_SUMMARY WHERE DNO=4);`
4. `UPDATE v_DEPT_SUMMARY SET DNO=3 WHERE DNO=4;`
5. `DELETE FROM v_DEPT_SUMMARY WHERE COUNT_EMPS > 4;`

SUBQUERIES AND VIEWS
VIEWS: ASSIGNMENT

```
CREATE VIEW v_DEPT_SUMMARY
 (DNO, COUNT_EMPS, SUM_SALARY, AVG_SALARY)
 AS SELECT Dno, COUNT(*), SUM(Salary),
 AVG(Salary)
 FROM EMPLOYEE
 GROUP BY Dno;
```

1. `SELECT * FROM v_DEPT_SUMMARY;`
2. `SELECT DNO, COUNT_EMPS FROM v_DEPT_SUMMARY WHERE SUM_SALARY > 100000;`
3. `SELECT DNO, AVG_SALARY FROM v_DEPT_SUMMARY WHERE COUNT_EMPS > (SELECT COUNT_EMPS FROM v_DEPT_SUMMARY WHERE DNO=4);`
4. `UPDATE v_DEPT_SUMMARY SET DNO=3 WHERE DNO=4;`
5. `DELETE FROM v_DEPT_SUMMARY WHERE COUNT_EMPS > 4;`

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

WORKS_ON

Essn	Pno	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	NULL

PROJECT

Pname	Pnumber	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship
333445555	Alice	F	1986-04-05	Daughter
333445555	Theodore	M	1983-10-25	Son
333445555	Joy	F	1958-05-03	Spouse
987654321	Abner	M	1942-02-28	Spouse
123456789	Michael	M	1988-01-04	Son
123456789	Alice	F	1988-12-30	Daughter
123456789	Elizabeth	F	1967-05-05	Spouse

ORGANIZATION
OUR JOURNEY IN THIS SEMESTER

- Integrity, Trigger & Security
- Database Applications
- **Transactions**
- Subqueries & Views
- More SQL
- Notations & Guidelines
- Constraints
- Relationships
- Simple Entities and Attributes
- Basics

Source: Foto von Justin Kauffman auf Unsplash [664]

TRANSACTIONS
BASICS

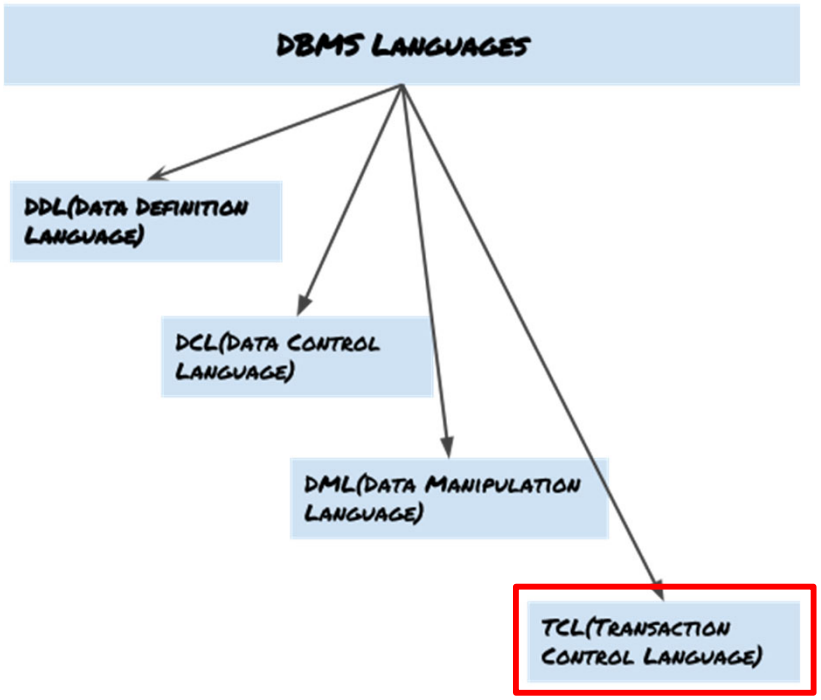
- A transaction bundles several operations into one logical unit
  - Unit of Work
  - Includes one or more database access operations
    E.g., **`INSERT`**, **`DELETE`**, **`UPDATE`**, **`SELECT`**
  - Operations must be executed all or none

- Example: Order a hotel room over the internet
  - Choose and reserve room
  - Payment
  - Final booking of the hotel room

TRANSACTIONS BASICS

- DBMS allow
  - many users &
  - concurrent access
- May lead to funny results if actions interfere
- Example:
  - Donald and Daisy withdraw money from their shared bank account

TRANSACTIONS
BASICS - DBMS LANGUAGES: SQL - TCL



TCL for performing or rollbacking of changes in the database that we made using DML commands

- **`SET TRANSACTION`**
- **`COMMIT`** → To persist the changes made by DML commands in database
- **`ROLLBACK`** → To rollback the changes made to the database
- **`SAVEPOINTS`**

Source: https://beginnersbook.com

TRANSACTIONS
ACID

- Key features of transactions
  - *Atomicity*: Transaction is executed in whole or not at all
  - *Consistency*: State of the DB is consistent before and after a transaction
  - *Isolation*: Transactions do not interfere with other concurrent transactions
  - *Durability*: Changes are stored permanently in the database and will not get lost

TRANSACTIONS
ACID - ATOMICITY

- A transaction can consist of many operations
  - **`SELECT`**
  - **`INSERT`**, **`UPDATE`**, **`DELETE`**
  - Note: statements for data definition (e.g., **`CREATE TABLE`**) usually outside transaction!

- Single operations are always atomic
  - Not trivial when looking at the implementation!

- In a transaction, all operations or none are performed

TRANSACTIONS
ACID - ATOMICITY

- Begin of Transaction (*BoT*)
  - SQL99: **`START TRANSACTION`**
  - Oracle: transaction is started automatically

- Commit a transaction: **`COMMIT;`**
  - All operations are made persistent
  - All changes are visible to other users

- Rollback transaction: **`ROLLBACK;`**
  - DB is in state at *BoT* again

TRANSACTIONS
ACID - ATOMICITY

- Autocommit
  - On some systems:
  - Single operations are committed automatically
  - Called **`autocommit`** mode

- May be turned off
  - ... by disabling it
  - ... by explicitly starting a transaction
  - Note: Method depends on the system!

TRANSACTIONS
ACID - CONSISTENCY

- DB: in consistent state before transaction
  → Also, in consistent state after transaction

- Integrity constraints assure that

- Constraints can be defined as
  - **`IMMEDIATE`** (default in mySQL)
    → are checked immediately after operation
  - **`DEFERRED`**
    → Check at time of commit

TRANSACTIONS
ACID - CONSISTENCY

Example: `Employee` ⬌ `Department`
If the relationship "*employee works in department*" is mandatory,
so that an employee should only exist, if he/she is working in a department,
then how can that be assured?
→ We must introduce a deferred constraint!

In Oracle: … **`FOREIGN KEY`** `(A)` **`REFERENCES`** `TAB1(A)` **`DEFERRABLE INITIALLY DEFEREED`**

TRANSACTIONS
ACID - CONSISTENCY

Example: `Employee` ⬌ `Department`
If the relationship "*employee works in department*" is mandatory,
so that an employee should only exist, if he/she is working in a department,
then how can that be assured?
→ We must introduce a deferred constraint!

In mySQL: Alternative solution

Employee	eid
	1

Works_in	eid (FK)	did (FK)
	1	1

Department	did
	1

TRANSACTIONS
ACID - ISOLATION

- Transactions are isolated from other concurrent transactions
- Concurrent transactions shall behave well

TRANSACTIONS
ACID – ISOLATION: CONCURRENCY CONTROL

- Concurrent operations can lead to problems
  - Lost Update
  - Dirty Read
  - Unrepeatable read
  - Phantom tuples

TRANSACTIONS
ACID – ISOLATION: CONCURRENCY CONTROL – LOST UPDATE PROBLEM

- Several transactions change the same value
- Example:

Time	Transaction 1	Transaction 2
1	`SELECT price`	-
2	-	`SELECT price`
3	`UPDATE price = 5`	-
4	-	`UPDATE price = 6`
5	`COMMIT`	-
6	-	`COMMIT`

Article	Price
Pen	1
Eraser	2
Ruler	3

TRANSACTIONS
ACID – ISOLATION: CONCURRENCY CONTROL – DIRTY READ (UNCOMMITED DEPENDECY PROBLEM)

- Transaction reads temporary value
- Example:

Time	Transaction 1	Transaction 2
1	`SELECT price`	-
2	-	`SELECT price`
3	`UPDATE price = 7`	-
4		`SELECT price`
5	`ROLLBACK`	-

Article	Price
Pen	1
Eraser	2
Ruler	3

- A dirty read (also uncommitted dependency) occurs when a transaction is allowed to read data from a row that has been modified by another running transaction and not yet committed

TRANSACTIONS
ACID – ISOLATION: CONCURRENCY CONTROL – UNREPEATABLE READ

- Transaction receives inconsistent value due to interfering transaction
- Credit account and debit account have to match with their values!
- Example:

Time	Transaction 1	Transaction 2
1	`SELECT credit_account`	-
2	-	`UPDATE credit_account`
3	-	`UPDATE debit_account`
4	-	`COMMIT`
5	`SELECT debit_account`	-
6	`COMMIT`	-

TRANSACTIONS
ACID – ISOLATION: CONCURRENCY CONTROL – PHANTOM PROBLEMS

- When doing the same **SELECT** twice, new tuples may appear that are inserted by another transaction
- Basically, same problem as Unrepeatable Read
- Example:

Time	Transaction 1	Transaction 2
1	**SELECT** amount	-
2	-	**SELECT** amount
3	-	**INSERT** new_amount
4	-	**COMMIT**
5	**SELECT** amount	-
6	**COMMIT**	-

Article	Price
Pen	1
Eraser	2
Ruler	3

Article	Price
Pen	1
Eraser	2
Ruler	3
Ink	2

TRANSACTIONS
ACID – ISOLATION: CONCURRENCY CONTROL – ISOLATION LEVELS IN SQL

- Lost Update is prevented by SQL

- Transactions: may choose *Isolation Level*
  - **`SERIALIZABLE`**
    → no problems
  - **`REPEATABLE READ`** (default in mySQL)
    → Open for phantom tuples
  - **`READ COMMITTED`** (default in Oracle, SQL Server)
    → Open for phantom tuples and unrepeatable read
  - **`READ UNCOMMITTED`**
    → Open for all problems

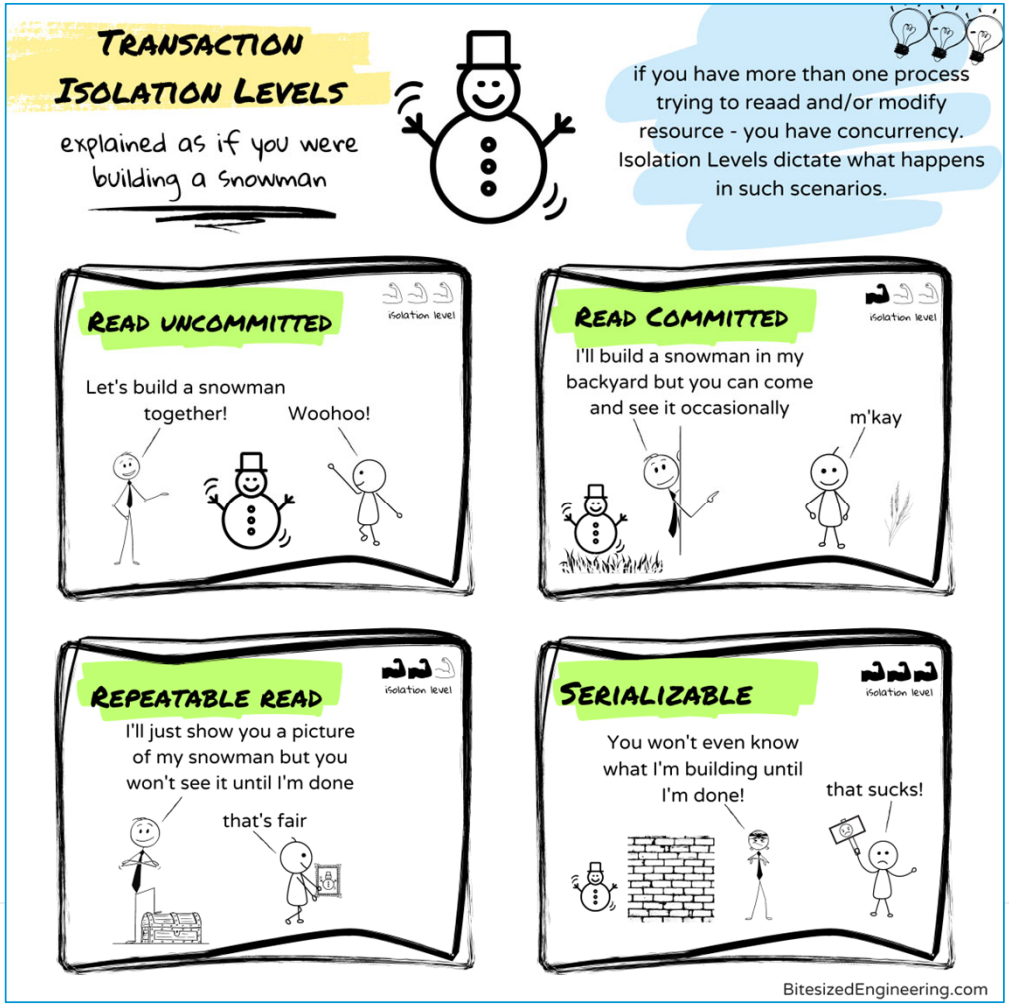
TRANSACTIONS
ACID – ISOLATION: CONCURRENCY CONTROL – ISOLATION LEVELS IN SQL

Isolation level \ Read phenomena	Lost updates	Dirty reads	Non-repeatable reads	Phantoms
Read Uncommitted	may occur	may occur	may occur	may occur
Read Committed	may occur	don't occur	may occur	may occur
Repeatable Read	don't occur	don't occur	don't occur	may occur
Serializable	don't occur	don't occur	don't occur	don't occur

Source: https://en.wikipedia.org/wiki/Isolation_(database_systems)

Source: https://www.youtube.com/watch?v=xR70UlE_xbo

TRANSACTIONS
ACID – ISOLATION: CONCURRENCY CONTROL – ISOLATION LEVELS IN SQL



Source: https://www.bitesizedengineering.com/p/database-isolation-levels-explained

TRANSACTIONS
ACID – ISOLATION: CONCURRENCY CONTROL – ISOLATION LEVELS IN SQL

- Isolation levels can be set
- Syntax:

```
SET TRANSACTION < transaction_mode > [, …]

<transaction_mode> ::= ISOLATION LEVEL {
                         SERIALIZABLE |
                         REPEATABLE READS |
                         READ COMMIT |
                         READ UNCOMMITED }
```

- Transactions can be *read only* if it contains only retrieval operations

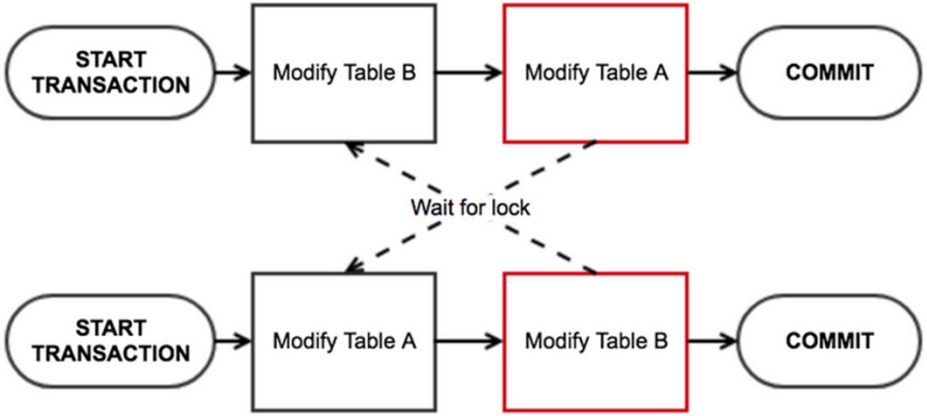
TRANSACTIONS
ACID – ISOLATION: CONCURRENCY CONTROL – IMPLEMENTATION

- Locks
  - Read Lock (share)
  - Write Lock (exclusive)

- Locks may hold for
  - Row
  - Table
  - Also: Memory page, Disk block

TRANSACTIONS
ACID – ISOLATION: CONCURRENCY CONTROL – IMPLEMENTATION

- Deadlocks may occur!!!
  → Usually are resolved automatically by aborting one transaction



Source: https://blog.nodeswat.com/concurrency-mysql-and-node-js-a-journey-of-discovery-31281e53572e

TRANSACTIONS
ACID – ISOLATION: CONCURRENCY CONTROL – IMPLEMENTATION

- Multiversion Concurrency Control
  - Transactions see state of the database at *BoT* (begin of transaction)
  - Can reduce number of locks
  - DB has to store different versions of tuples

- Example:

Time	Object1	Object2	Object3
0	"Foo" by T1	"Bar" by T1	
1	"Hello" by T2		
2		(deleted) by T3	"Foo-Bar" by T3

Source: https://en.wikipedia.org/wiki/Multiversion_concurrency_control

- Conceptually similar to Subversion or Git

TRANSACTIONS
ACID - DURABILITY

- Once committed, changed data is safe

- Error types
  1. Computer failure
  2. Transaction or system error
     (constraint violation, $\frac{x}{0}$, blackout, system crash)
  3. Local Errors
  4. Concurrency control enforcement
  5. Disk error (harddisk broken)
  6. Physical problems and catastrophes
     (fire, earthquake, robbery, ...)

Source: Elmasri, Fundamentals of Database Systems, Page 750ff

TRANSACTIONS
ACID – DURABILITY: ERROR HANDLING

- Recovery from transaction failures usually means that the database is *restored* to the most recent consistent state just before the time of failure

- Minor damages due to error types 1-4 from slide "ACID – Durability"
  - DBMS provides handling
  - Recovery strategy is to identify any changes that may cause an inconsistency in the database
    - Changes are first written to redo logs (files on disk)
    - Written to database files after commit

Source: Elmasri, Fundamentals of Database Systems, Page 808ff

TRANSACTIONS
ACID – DURABILITY: ERROR HANDLING

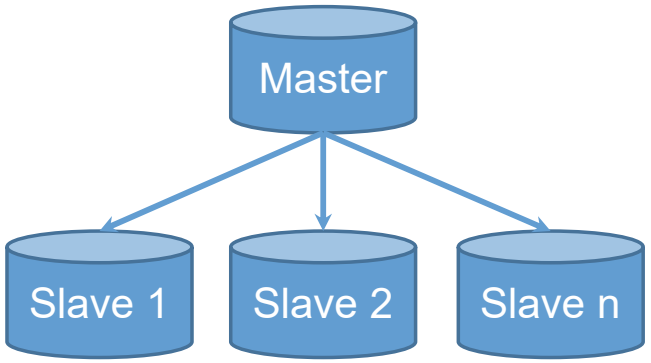
- Extensive damage due to error types 5-6 from slide "ACID – Durability"
  - recovery handling restores a past copy of the database from archival storage
  - reconstructs a more current state by redoing the operations
  - Last transactions are lost!

- Solution: Redundancy
  - RAID
    (**r**edundant **a**rray of **i**ndependent **d**isks)
  - Data Replication by DBMS

Source: Elmasri, Fundamentals of Database Systems, Page 808ff

TRANSACTIONS
ACID – DURABILITY: ERROR HANDLING – DATA REPLICATION

- Changes are performed on (replicated to) several database instances
  - Master/Slave
    - Updates only on one instance (master)
    - Slave: Read only vs. Standby
  - Multi-Master
    - Updates on different instances
    - Needs conflict resolution strategy



TRANSACTIONS

ACID – DURABILITY: ERROR HANDLING – DATA REPLICATION: SYNC VS. ASYNC

- *Synchronous*
  - Transaction valid only when committed on all DBs
  - Safest, but performance impact
  - May reduce availability of the system

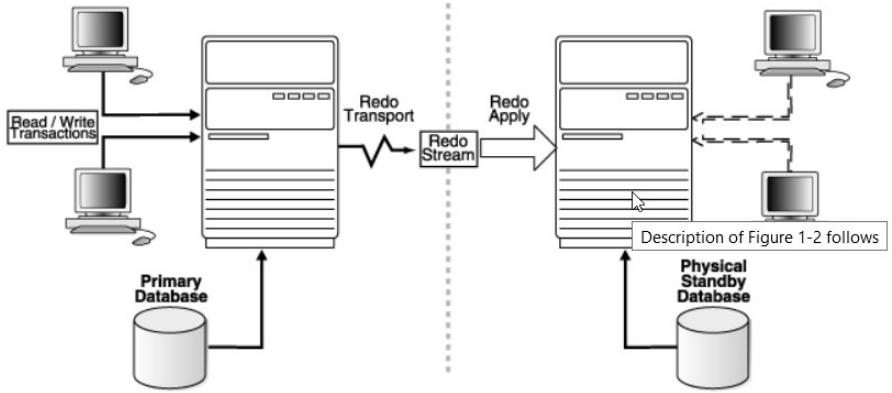
- *Asynchronous*
  - Transaction valid when committed locally

TRANSACTIONS
ACID – DURABILITY: ERROR HANDLING – DATA REPLICATION METHODS

- Low level (disk device)
- Trigger based
  → Update triggers the replication (SQL level)
- Logfile shipping
  - Changes are stored in redo logs (as usual)
  - redo logs are copied to standby DB

TRANSACTIONS
ACID – DURABILITY: ERROR HANDLING – DATA REPLICATION METHODS

- Oracle
  - Data Guard
    → Replication on second server, can be used to answer Read-Only queries

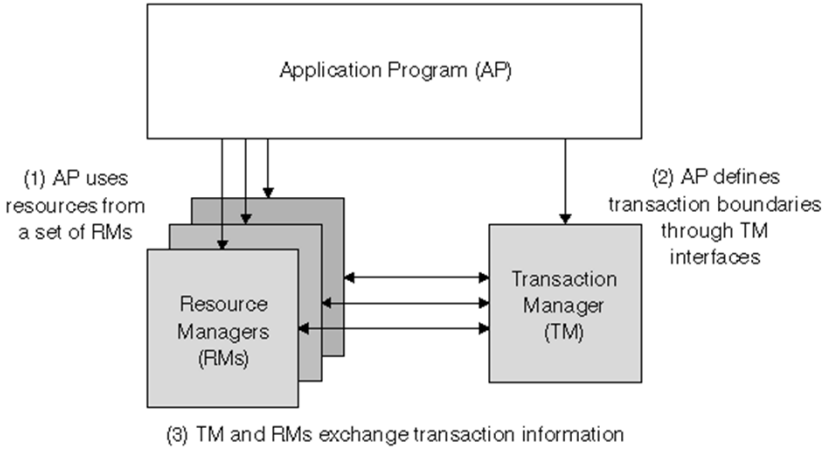


Source: https://docs.oracle.com/cd/B19306_01/server.102/b14239/concepts.htm#i1033808

  - Real Application Cluster (RAC)
    → Several servers share the same DB

TRANSACTIONS
DISTRIBUTED TRANSACTIONS

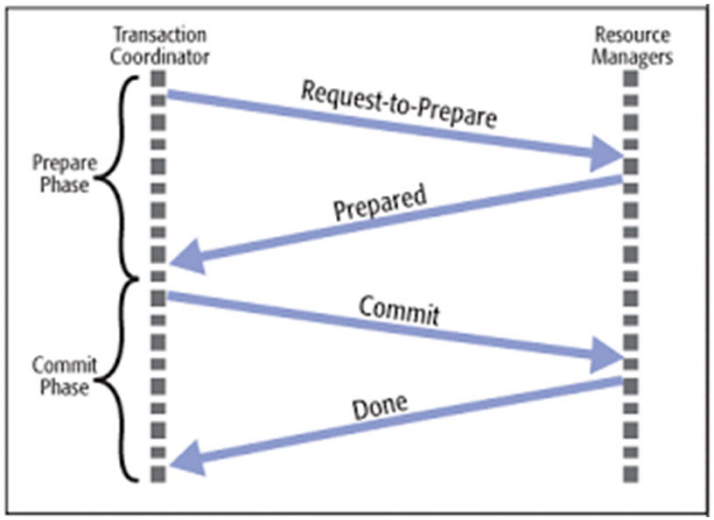
- Transactions not only in a single DBS
- Standardized by X/Open
  - Transaction Manager:
    A software component that guarantees transaction properties
  - Resource Manager:
    Every resource (e.g., DBS, GUI) that is able to work in a transactional mode without providing a transaction control structure itself



- The Transaction manager coordinates the Resource Manager that take part in the transaction.
  E.g., different DBS (distributed transactions) that appear as one DBS from outside (transparency!)

TRANSACTIONS
DISTRIBUTED TRANSACTIONS

- To ensure interoperability between the participating resource managers the *2-phase commit protocol* is realized



- It defines the final synchronization of different parts of a transaction of a global transaction
- In the first phase the transaction manager asks participating resource managers to announce the results of their local transaction part
- This leads to a global result (commit or rollback) that is then in the second phase announced to the participants

Source: https://medium.com/@balrajasubbiah/consensus-two-phase-and-three-phase-commits-4e35c1a435ac

TRANSACTIONS
SAVEPOINTS

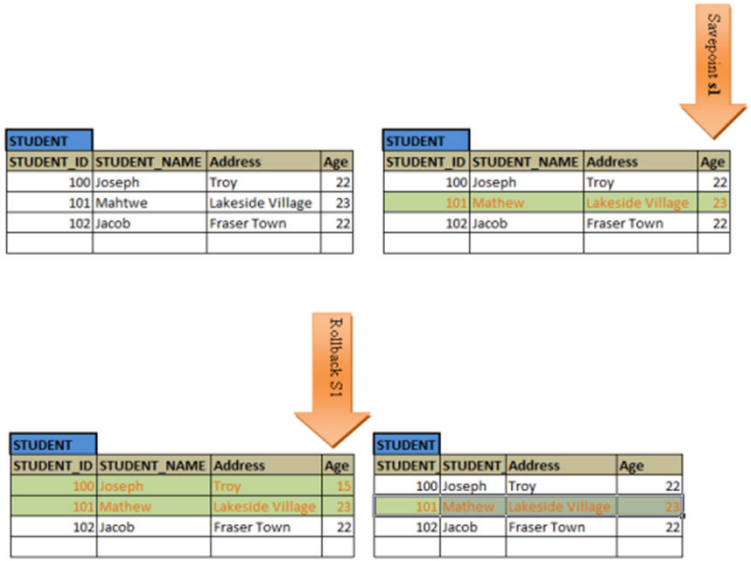
- There are operations that may be expensive to execute
  → time consuming
- If certain constraints fail within transaction execution, then maybe these constraints may not fail in a second attempt
  (e.g., time dependent)
- So "fall back" points can be defined, which are called *savepoints*
- It is possible to rollback up to a savepoint and restart transaction execution from this point on

TRANSACTIONS
SAVEPOINTS

Example:

Code

```
UPDATE STUDENT SET STUDENT_NAME = 'Mathew' WHERE STUDENT_NAME = 'Mahtwe';
SAVEPOINT S1;
UPDATE STUDENT SET AGE = 15 WHERE STUDENT_ID = 100;
ROLLBACK to S1;
```



Source: https://www.tutorialcup.com/dbms/transaction-control-language.htm