Engineering Databases

Lecture 10 – Summary

February 1, 2023

M. Saeed Mafipour & Mansour Mehranfar

Contents of lecture 1

- A **Database** is where all the data is stored
- A **Database Management System** is the administrative software of the DB
- A **schema** defines the structure of the data
- An **instance** is one set of data that complies to the schema
- A **data model** describes how we can model the schema in the database
- Our data model of choice is the **relational model**

- We use the **Entity Relationship Model** to design a schema
- **Entities** are well-defined physical objects or mental concepts 

- **Relationship** are given between entity types 

- **Attributes** characterize entities and relationships 

- The **primary key** is a attribute (set) that identifies one entity 

- **Cardinalities** represent consistency rules (Chen Notation, 1:1, 1:N, M:N)

Contents of lecture 1

Abstraction layers/levels

Views

- Partial sets of entire data set
- Tailored to the needs of specific users

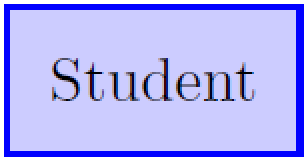
Logical Layer

- how data is logically organized → schema

Physical Layer

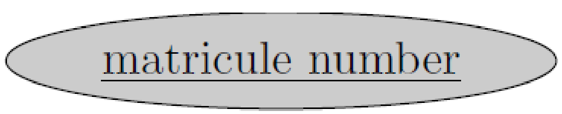
- how data is stored on disk

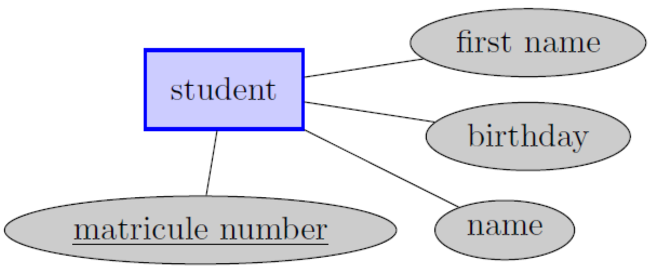
Contents of lecture 1

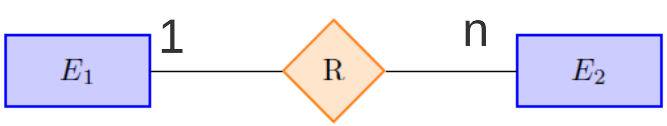
- ER-Diagrams comprise a well defined set of symbols
- Entity 

- Relationships 

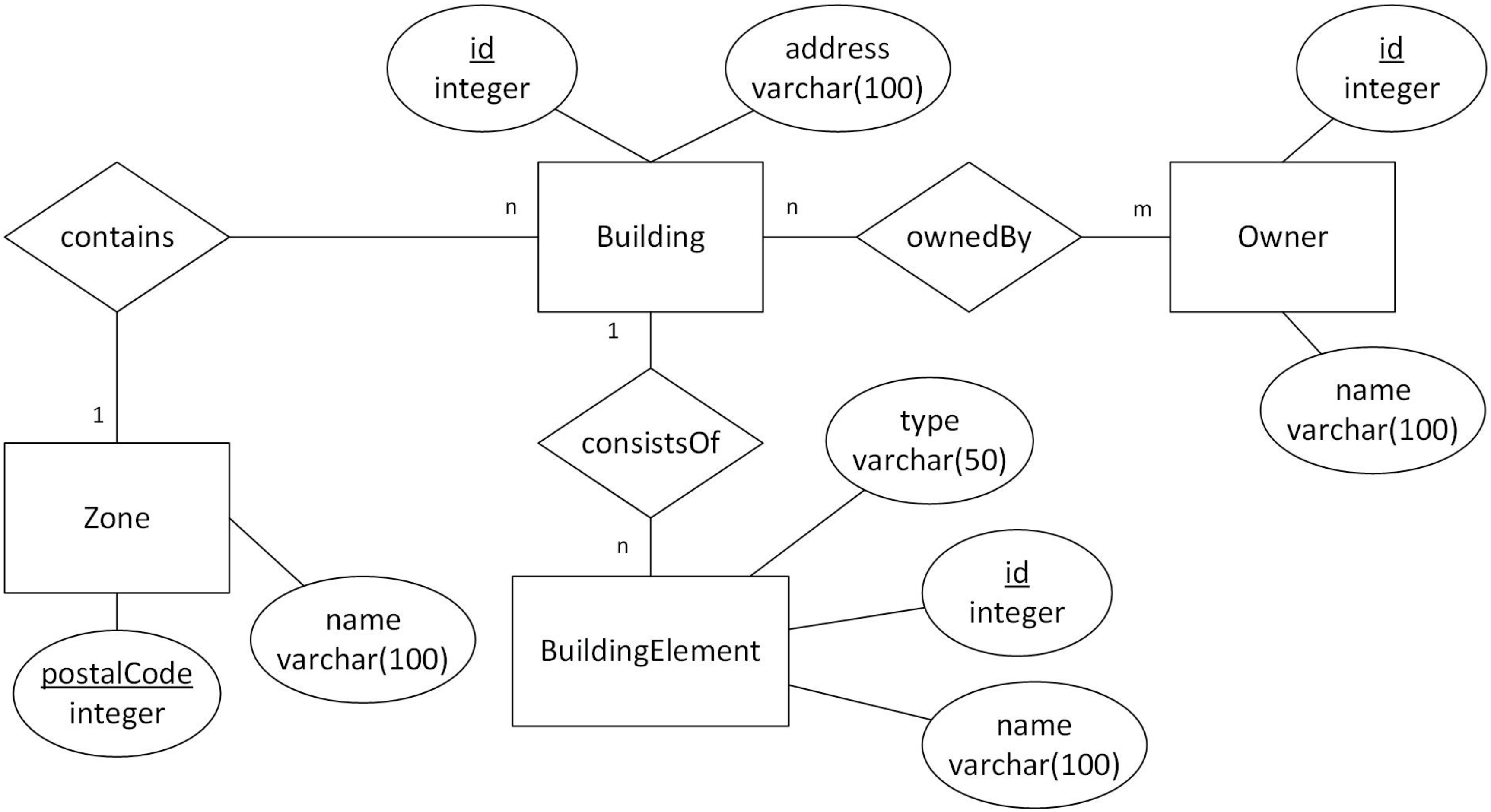
1:1, n:1, n:n
- Attributes 

- Primary key 





ER Diagram with data types



Contents of lecture 2

- The **Data Definition Language** (DDL) is used to define and modify schema
- **Data Manipulation Language** (DML) is used to add, remove, and query data
- SQL a **declarative language**, you define what you want and not how to compute it
- DDL:
  - create table <tableName> (<c1Name> <c1Type> <c1constrains> [, …]);
  - drop <tablerName1> [, …];
  - rename <tableName> to <newName>
  - alter table <tableName> <specifications: add, modify, change and drop>;
- DML (partial):
  - insert into <tableName> (c1Name [, …]) values (c1Value [, …])
  - select <columns or *> from <data source, e.g. tableName> where <Conditions>

SQL basic data types

- Each attribute (column) has a data type
- Numbers:
    - numeric/number: generic type
  - integer
  - float
  - date/time, datetime, e.g. date: 0001-01-01. . . 9999-12-31
  - time: 00:00:00.0000-23:59.59.9999
- Strings:
  - char(n)
  - varchar(n)
- Binary Data

http://dev.mysql.com/doc/refman/5.7/en/data-types.html

Integrity Constraints

- NOT NULL (data cannot be empty)
- UNIQUE (cannot be null, unique value in the table)

- PRIMARY KEY (identifier of the entity and unique)
- FOREIGN KEY (identifier of an entity of another table)

- DEFAULT (default value of a column)
- AUTO_INCREMENT (automatically set to ‘last value + 1’)

- Combinations possible

```
CREATE TABLE Lectures (
CourseNo INTEGER PRIMARY KEY AUTO_INCREMENT);
```

http://dev.mysql.com/doc/refman/5.7/en/create-table.html

Contents of Lecture 3

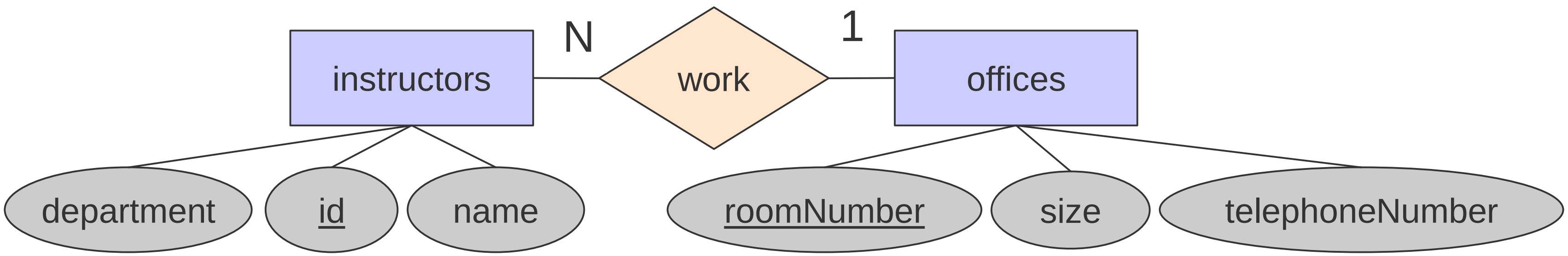
- DML (rest):
  - update <t1> set <c1>=<v1> [,<c2>=<c2>,…] [where <condition>]
  - delete from <t1> [where <condition>]
  - insert into for multiple data sets
- Advanced ER-Mapping Schema
  - 1:N, N:1, and 1:1 relation can be eliminated in the table design
- The Relational Model by Codd
  - The formal groundings of SQL based on set theory and first-order predicate logic
  - Domains, attributes, relations and relational schemas
  - Projections: $\Pi_{columns}$ (Relation), extract columns from a relation
  - Union: $R_i \cup R_j$, merge identical schemas
  - Selection: $\sigma_{statement}$(Relation), select sub-sets of a relation based on formulas

Contents of Lecture 4

- Foreign key integrity constraint
  - A essential principle in relational design
  - Ensures that a value in a table is contained in some other table
  - Control updates and changes of other tables
- Relational algebra and joins
  - Cartesian product $\times$
  - Renaming ρ
  - θ-Join $\bowtie_{\theta}$ and equi-Join $\bowtie_{\theta\ is\ =}$
  - Natural Join $\bowtie$
  - Right $\ltimes$ and left semi join $\rtimes$
  - Left ⟕, right ⟖, and full outer join ⟗

Foreign key integrity constraint

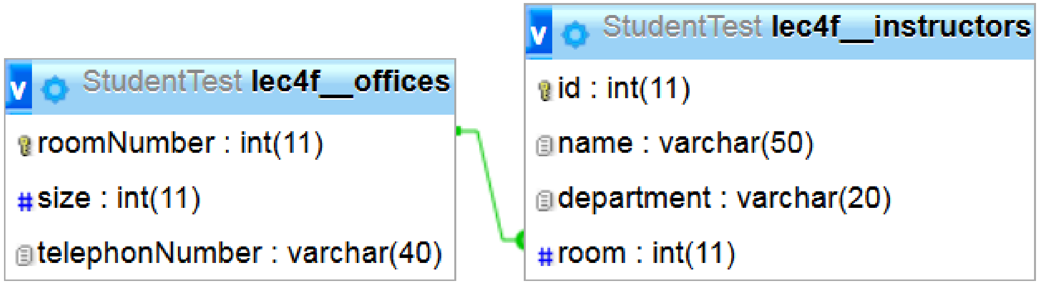
- Example:



instructors			
id	name	department	room
1	Bob	CMS	10
2	Tom	CIE	5
3	Alice	CMS	8

offices		
roomNumber	size	telephoneNumber
10	10	22678
8	25	35131

Foreign key integrity constraint



- Example:

```
CREATE TABLE offices (
    roomNumber INTEGER PRIMARY KEY,
    size INTEGER NOT NULL,
    telephonNumber VARCHAR(40) NOT NULL);

CREATE TABLE instructors (
    id INTEGER PRIMARY KEY,
    name VARCHAR(50) NOT NULL,
    department VARCHAR(20),
    room INTEGER,
    FOREIGN KEY (room) REFERENCES offices (roomNumber));
```

Content of Lecture 5

- Explicit renaming of output columns $\rho_{new\ name/old\ name}(relation)$

- Aggregate by a column GROUP BY
- Aggregate group columns by functions e.g. COUNT(column)

- Use nested queries in WHERE and SELECT clause
- Use quantor [NOT] EXISTS or IN to check if a SELECT has content
- Special language elements are helpful e.g. BETWEEN

Aggregate Functions and Grouping

- Aggregated group entries
- The SQL syntax for the renaming an attribute/column:
  SELECT <attributes> FROM <Relation>
  GROUP BY <attribute>
- Example: SELECT firstName FROM persons GROUP BY firstName

firstName	lastName	age	male
Max	Bügler	33	1
Max	Mustermann	20	1
Max	Müller	25	1
Parker	James	28	0
Parker	Miller	24	1
Jennifer	Milan	22	0
Jennifer	Turner	28	0
John	Turner	25	1
Jennifer	Turner	21	0

firstName
Jennifer
John
Max
Parker

firstName	lastName	age	male
Max	Bügler	33	1
Max	Mustermann	20	1
Max	Müller	25	1
Parker	James	28	0
Parker	Miller	24	1
Jennifer	Milan	22	0
Jennifer	Turner	28	0
John	Turner	25	1
Jennifer	Turner	21	0

Aggregate Functions and Grouping

```
SELECT
  firstName,
  GROUP_CONCAT(lastName) AS lastNames,
  GROUP_CONCAT(DISTINCT lastName) AS distinctLastNames,
  COUNT(lastName) AS count,
  COUNT(DISTINCT lastname) AS distinctCount,
  AVG(age) AS averageAge,
  STDDEV(age) AS ageStDev
FROM Persons
GROUP BY firstName;
```

firstName
Jennifer
John
Max
Parker

https://dev.mysql.com/doc/refman/5.7/en/group-by-functions.html

Content of lecture 6

- Pitfalls in SQL
  - Use ` and ‘ and “ and \
  - Be aware of attacks on Databases, e.g. SQL Injections
- Sort result of a query using ORDER BY
- Limit the number of rows of sorted query by LIMIT

- Triggers
  - Mechanism to react on updates, insert, and delete statements
  - Is connected to changes of the content of a table
  - Will run before or after these events
  - Runs a single or multiple statements
  - Will run for each row. This means for all rows that activate the trigger

Content of Lecture 7

- Views are virtual tables
- Views ‚save' SQL select statements

- Transaction ensure database consistency
- Transaction bundle multiple operations in a single unit
- ACID = Atomicity, Consistency, Isolation, and Durability

- Normalization improves relation database design in a formal way
- Normalizations avoid update, insert and delete anomalies
- Using the ER-Diagram mostly generates normalized tables

Transactions

- Properties of Transaction

- ACID = Atomicity, Consistency, Isolation, and Durability

- Atomicity: A transaction runs either entirely or not at all
- Consistency: A transaction takes the database from one consistent state to another
- Isolation: A transaction runs in isolation from other transactions
- Durability: Changes by a committed transaction must persist in the database

http://dev.mysql.com/doc/refman/5.7/en/sql-syntax-transactions.html

Normalization

- Change the track title ‘Run’ to ‘Runner’ for CD#1

CD #	Album Title	Musician	Date	Tracks
1	Magic	A	1999	Trick, Unity, Run, Runner
2	Dragon	B	1999	Fire, Smoke, Gold
3	Dance	A	1999	Neon, Light, Floor

- The system finds the text: **Trick, Unity, Run, Runner**
- It has to scan for Run and exchanges Run with Runner
- The result is: **Trick, Unity, Runner, Runnerner**

- Normal Form 1: All attributes must be atomic
- Solutions: Never put distinct data items in an single attribute

Normalization

- Change the artist ‘A’ hometown to ‘Washington’

Genre	Artist	Hometown
Dance	A	New York
Rock	B	Berlin
Pop	A	New York

Artist	Hometown
A	New York
B	Berlin

Genre	Artist
Dance	A
Rock	B
Pop	A

- The system have to find all rows that correspond to artist A
- It has to scan the whole table and rename the Hometown multiple times for A.

- Normal Form 2: All non-key attributes are dependent on the complete primary key
- Here: Hometown (non-key) is dependent on Artist (key) but not on Genre (key)
- Solution: Split into separate tables in which this is true

Normalization

- The 3 normal form provides optimal balance between performance, redundancy and flexibility

Artist	Birth	Zipcode	Hometown
A	1975	503	New York
B	2003	313	Berlin
C	1993	313	Berlin

Artist	Birth	Zipcode
A	1975	503
B	2003	313
C	1993	313

Zipcode	Hometown
503	New York
313	Berlin

- Normal Form 3: No non-key attribute is transitively dependent on a primary key
- Here:

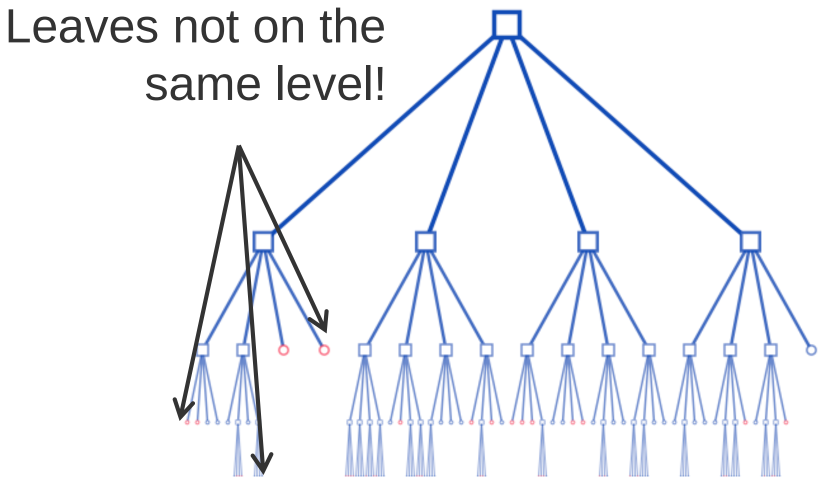
  Zipcode (non-key) is dependent on Artist (key). (Z depends on A)
  Birth (non-key) is dependent on Artist (key). (B depends on A)
  Hometown (non-key) is dependent on Artist (key). (H depends on A)
  However, Hometown is actually dependent on Zipcode (H depends on Z)
- Solution: Break (H depends A, via Z depends A) by splitting the table

Content of Lecture 8

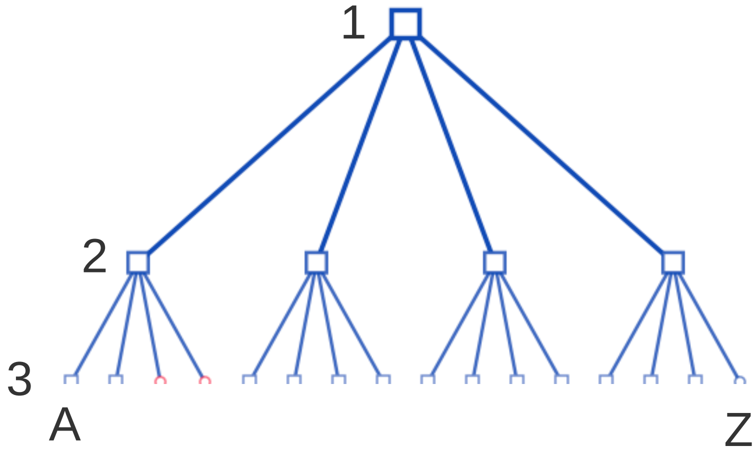
- Indexing improves the query performance of a database
- We measure performance based on the worst case complexity of an algorithm
- We describe the complexity via the Big-O Notation, e.g. O(1) = constant-time

- Indexing can be applied to attributes of a table
- Typically, the database creates an index for each primary key
- Index are not for free, e.g. higher effort for insert, delete, and update
- ISAM (Index Sequence Access Method) is an indexing method

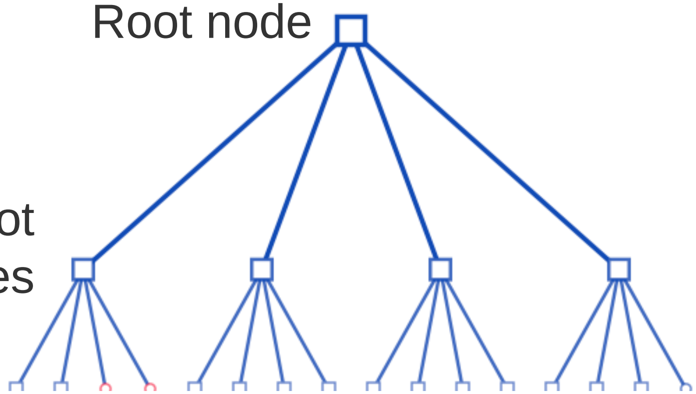
Content of Lecture 8
Indexing B-Tree



- B-Trees are balanced
  Leaves are on the same depth
  This makes the B-Tree efficient!
- Maximum number of disk accesses
  is limited by height of the tree
- Same number of operation for each search
  E.g. search for “Alfred” takes as
  long as for “Zeppelin”
- Search complexity is O(log n)



Content of Lecture 9
Indexing B-Tree



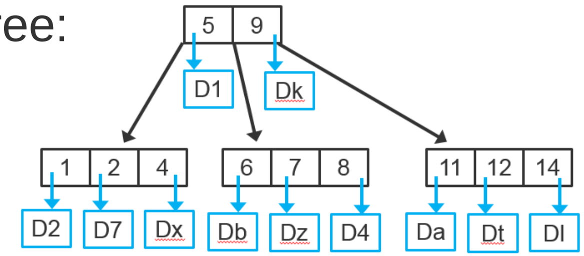
- Parameter:
  Order m ≥ 3

- All leaves are on the same depth (level).
- All keys in a node are in ascending order.
- A non-leaf node with **n-1** keys must have **n** child nodes.
- If the root node is a non-leaf node, it must have at least **2** child nodes.
- A non-root non-leaf node must have at least **m/2** child nodes.
- A nodes (except the root node) must have
  at least **m/2-1** keys and maximal **m-1** keys.

- Hint: if e.g. m = 3 and $\frac{3}{2} - 1 = 0.5$, we use the ceiling method. Thus, $\left\lceil \frac{3}{2} - 1 \right\rceil = 1$

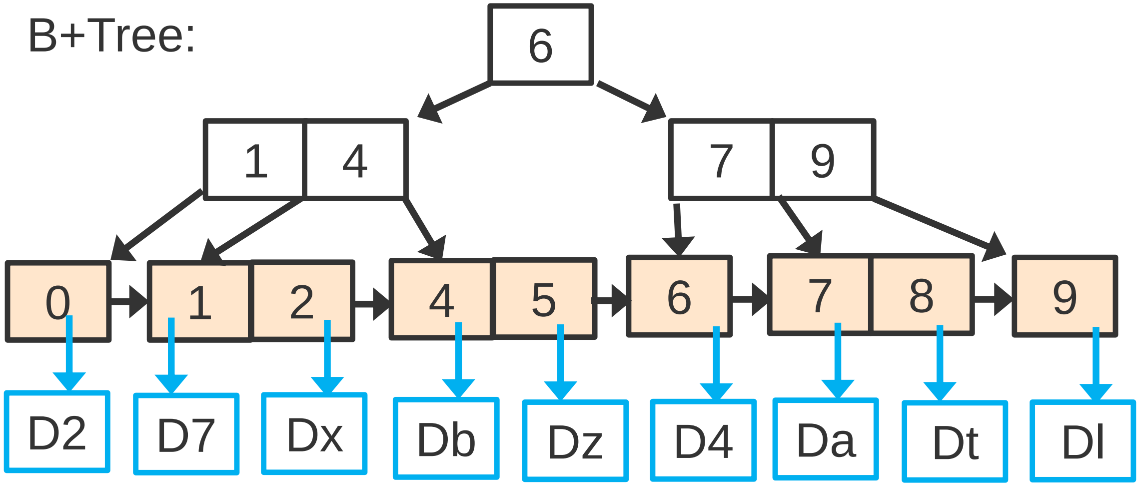
Content of Lecture 9
Indexing B+ Trees

B-Tree:



- Extension of the B-Tree

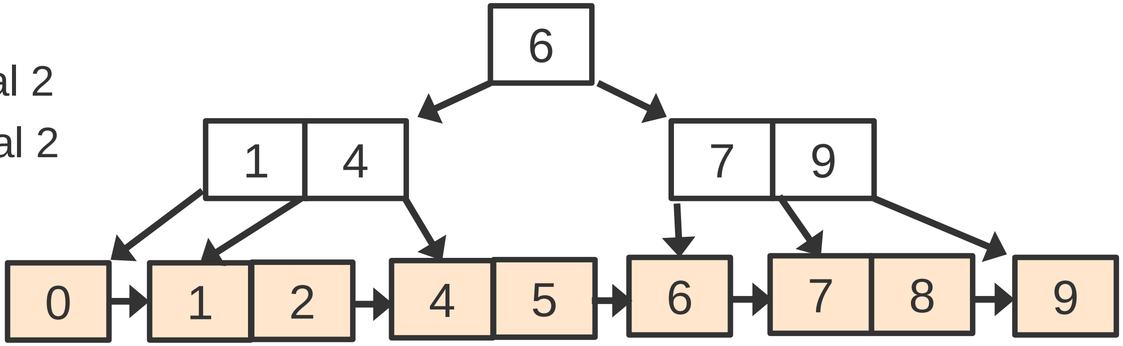
B+Tree:



- Nodes only store keys
- The pointers in the leaves point to data
- Leaves are connected by additional pointers: fast sequential access
- B+ Trees "grow from the root“

Content of Lecture 9
Indexing B+ Trees

- The rules for a B+Tree
  - Maximum number of keys per node: n
  - Root node: at least 2 children
  - Non-root nodes (inner nodes): at least (n+1)/2 children (rounding up, ceiling)
  - The values are sorted
  - All leaves are on the same depth
- Let n = 2
  Keys = maximal 2
  Root-node children = minimal 2
  Inner node children = minimal 2



Content of Lecture 9
NoSQL Databases

- Relational databases
  - are well standardized and established
  - successful but have problems regarding big data and distributed systems

- NoSQL technologies
  - address new requirements
  - are aggregate-oriented
  - give up the ACID principle

- NoSQL types
  - Map-Reduce model, Key-Value stores, Document databases, Column-family stores, Graph databases

End of Lecture

Thank you for your attention