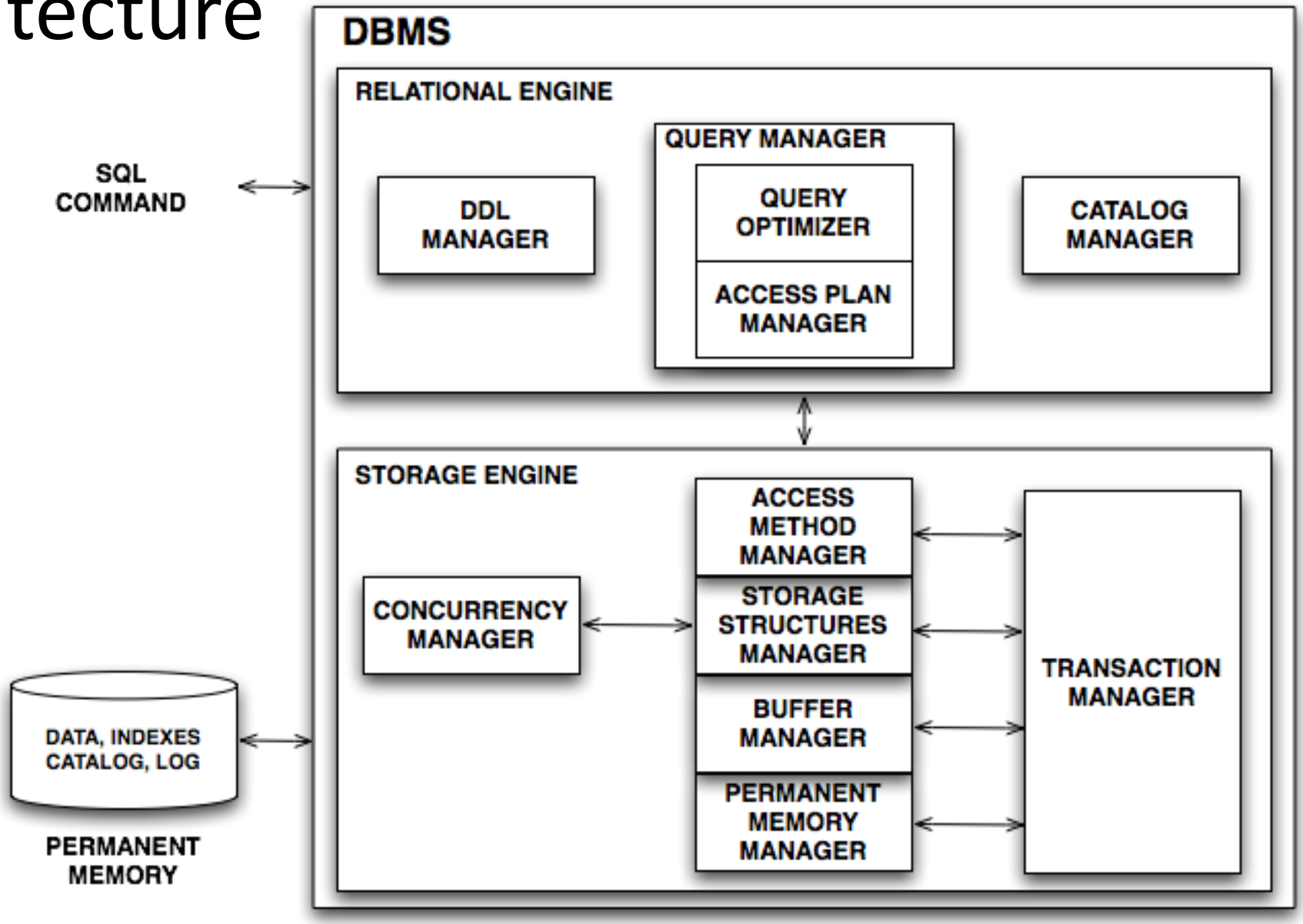
Architecture



Access Methods

- The Storage Structure Manager implements primary organizations and indexes
- The Access Methods Manager provides the operators to access such structures
- The operators provided are used to implement the physical operators of query plans generated by the query optimizer
- The Access Methods interface is the Storage Engine interface

Storage engine operators: data and transactions

- JRS inspired (slightly different on the book)
- createDB: Path × DBName × TransId → DB
- createHF: DB × Path × HFName × TransId → HF
- createIdx: DB × Path × IdxName × HFName × Attr × Ord × Unique × TransId → Idx
- dropBD: DB × TransId → null
- dropIdx: Idx × TransId → null
- dropHF: HF × TransId → null
- Transactions: beginTransaction, commit, rollback

Storage structure: heap file

- HF_open: DB × HFName × TransId → HF
- HF_close: HF → null
- HF_getRecord: HF × RID → Record
- HF_deleteRecord: HF × RID → null
- HF_updateRecord: HF × RID
  × FieldNum × NewField → null
- HF_insertRecord: HF × Record → RID
- HF_getNPage: HF → integer
- HF_getNRec: HF → integer

Indexes

- I_open: BD × IndexName × TransId → Idx
- I_close: Idx → null
- I_deleteEntry: Idx × Entry → null (Entry = Value × RID)
- I_insertEntry: Idx × Entry → null
- I_getNKey: Idx → integer
- I_getNleaf: Idx → integer
- I_getMin: Idx → Value
- I_getMax: Idx → Value

Access method: heap file scan

- HFS_open: HF → HFS
- HFS_isDone: HFS → boolean
- HFS_getCurrent: HFS → RID
- HFS_next: HFS → null
- HFS_reset: HFS → null
- HFS_close: HFS → null
- while not hfs.HFS_isDone()

  do (rid := hfs.HFS_getCurrent(); ….; hfs HFS_next())

Access methods: index scan

- IS_open: Index × firstKey × lastKey → IS
- IS_isDone: IS→ boolean
- IS_getCurrent: IS → Entry
- IS_next: IS → null
- IS_reset: IS → null
- IS_close: IS → null

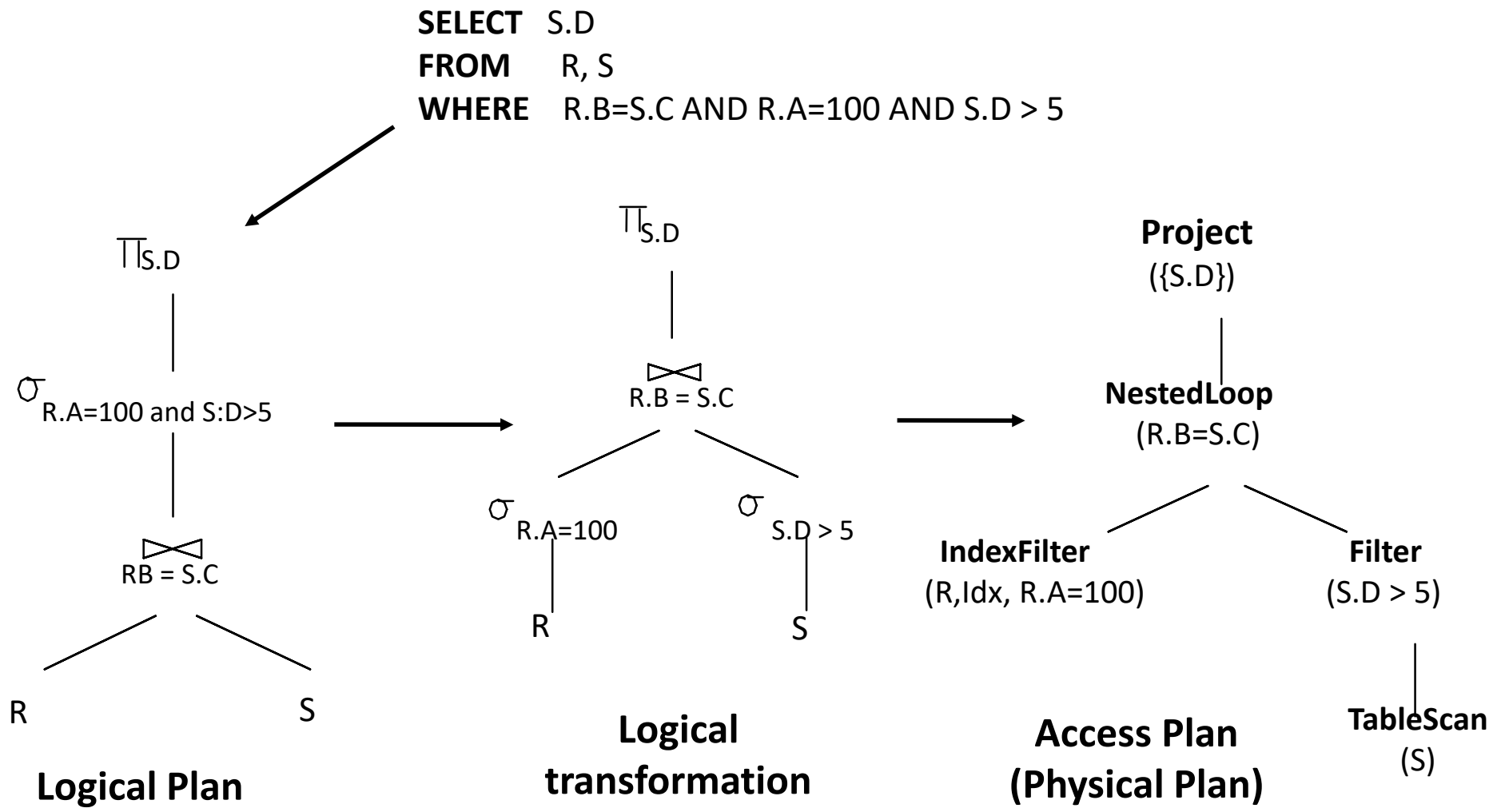
SELECT Name FROM Students WHERE Province = "PI"

```
HeapFile  students = HF_open ("path", "bd", "Students", transId);
ScanHeapFile cursorHF := HFS_open( students );
while ( !cursorHF.HFS_isDone() )
        {       RID rid = cursorHF.HFS_getCurrent();
                Record theRecord = students.HF_getRecord(rid);
                if (theRecord.getField(4).equals( "PI" ) )
                                System.out.println (theRecord.getField(1));
                cursorHF.HFS_next();
        };
students.HF_close ();
cursorHF.HFS_close();
```

Using an index

```
DB = DB_open ("path", "bd“);
HeapFile students = DB.HF_open ("Students”, transId);
Index IndexProvince  = DB.I_open("IdxProvince”, transId);
ScanIndex  IndexCursor := IndexProvince.IS_open ('PI','PI');
while ( ! IndexCursor.IS_isDone() )
{           RID rid = IndexCursor.IS_getCurrent().getRid();
            Record  theRecord = students.HF_getRecord(rid);
            System.out.println (theRecord.getField(1).);
            IndexCursor.IS_next();
}
students.HF_close (); IndexCursor.IS_close (); IndexProvince.I_close ()
```

Executing SQL



Physical plan execution

- Each operator is implemented as an iterator using a ‘pull’ interface: when an operator is ‘pulled’ for the next output tuples, it ‘pulls’ on its inputs and computes them
- An operator interface provides the methods open, next, isDone, and close, implemented using the Storage Engine interface

Query execution

```
// SQL COMMAND Q: Parsing and analysis
SQLCommand parseTree = Parser.parseStatement(Q);
Type type = parseTree.check();

// QUERY OPTIMIZATION
Value accessPlan = parseTree.Optimize();

// ACCESS PLAN EXECUTION
accessPlan.open();
while !accessPlan.isDone() do
{ Record rec = accessPlan.next();
  print(rec);
}
accessPlan.close();
```

Example: TableScan(R)

- open(openCondition)

  heapFileR = HF_open ("path", "bd", "R", transId);

  cursorHF-R = HFS_open(heapFileR);

  nextRecord = getNextRecord(cursorHF-R);

- isDone()

  return nextRecord.isNull();

- next()

  currentRecord = nextRecord;

  nextRecord = getNextRecord(cursorHF-R);

  return currentRecord;

- close()

  heapFileR.HF_close (); cursorHF-R.HFS_close();