Федеральное государственное бюджетное образовательное учреждение высшего образования

«Чувашский государственный университет им. И.Н. Ульянова»

Кафедра вычислительной техники

Лабораторная работа №6

“Разработка отчетов ”

по дисциплине Базы данных

Вариант: Инвестор

Выполнил:

студент гр.ИВТ-42-23,

Назаров Я. С.

Проверил: старший

преподаватель Марков А.В.

Чебоксары, 2025

**Целью** настоящей лабораторной работы является изучение и практическое применение механизмов генерации печатных и аналитических отчетов в среде Odoo с использованием технологий QWeb, Pivot и Wizard.

**Odoo** представляет собой современную ERP-систему, построенную на основе объектно-реляционного отображения (ORM) и системы представлений (Views). Механизмы генерации отчетов в Odoo полностью покрывают и превосходят функциональные возможности традиционных СУБД, таких как Microsoft Access, обеспечивая при этом гибкость параметризации, сложную бизнес-логику и интеграцию с веб-интерфейсом.

В рамках данной работы рассматривается реализация отчетности в модуле «Инвестиционный модуль», предназначенном для учета инвестиционных операций, транзакций и анализа доходности портфелей инвесторов. Отчетность в модуле реализована с использованием следующих механизмов:

1. **QWeb-отчеты** — печатные многотабличные отчеты с параметризацией через Wizard;

2. **Pivot View** — аналитические сводные таблицы для группировки и агрегации данных;

3. **Graph View** — графическое представление данных в виде диаграмм различных типов;

4. **Wizard (TransientModel)** — мастера для задания параметров отчетов (модель данных, не хранящаяся в БД).

Все указанные механизмы реализованы на базе стандартных возможностей Odoo и демонстрируют соответствие классическим требованиям к отчетности в СУБД: однотабличные, многотабличные (составные), параметризованные и сводные отчеты.

**Раздел 1. Реализация печатного отчета (QWeb + Wizard)**

**1.1. Общая архитектура отчета «Отчет о доходах инвестора»**

Отчет «Отчет о доходах инвестора» реализован с использованием двух основных компонентов:

1. **TransientModel** investor.income\_report.wizard — мастер (Wizard) для ввода параметров отчета (инвестор, счет, период дат);

2. **Model** investor.income\_report — модель, содержащая вычисляемые поля для расчета итогов и налогов;

3. **QWeb-шаблон** investor.income\_report — XML-шаблон для визуализации отчета в формате HTML/PDF.

Такой подход соответствует требованиям к параметризованным отчетам в СУБД: пользователь задает критерии выборки через диалоговое окно (Wizard), после чего система генерирует отчет с вычисляемыми итогами и условной логикой.

**1.2. Реализация Wizard (Мастер параметров)**

**Wizard** реализован в модели investor.income\_report.wizard, наследующейся от TransientModel. TransientModel в Odoo представляет временную модель, записи которой автоматически удаляются из базы данных после выполнения действия, что соответствует концепции мастера в традиционных СУБД.

**Ключевые особенности Wizard:**

- Автоматическое заполнение поля investor\_id на основе текущего пользователя (метод default\_get);

- Динамический домен для поля account\_id в зависимости от выбранного инвестора (метод \_onchange\_investor\_id);

- Валидация диапазона дат (ограничение \_check\_dates);

- Проверка прав доступа при генерации отчета (метод action\_generate\_report).

Метод action\_generate\_report создает запись модели investor.income\_report и возвращает действие (ir.actions.report) для запуска QWeb-отчета.

**1.3. Модель отчета с вычисляемыми полями**

Модель investor.income\_report содержит следующие группы полей:

1. **Параметры отчета**: investor\_id, account\_id, date\_from, date\_to;

2. **Вычисляемое поле** transaction\_ids — Many2many-связь с транзакциями, рассчитываемая методом \_compute\_transactions на основе заданного домена;

3. **Итоговые вычисляемые поля**: total\_buy, total\_sell, total\_deposit, total\_withdrawal, total\_commission, financial\_result, taxable\_base, estimated\_tax, total\_income\_after\_tax.

Метод \_compute\_totals выполняет агрегацию транзакций по типам операций и рассчитывает финансовые итоги, включая налоговую логику:

python

@api.depends('transaction\_ids', 'account\_id', 'account\_id.account\_type')

def \_compute\_totals(*self*):

    """Вычисляет итоговые суммы и налоги"""

*for* record *in* self:

*# Агрегация по типам операций*

        record.total\_buy = abs(sum(t.amount *for* t *in* record.transaction\_ids

*if* t.operation\_type == 'buy'))

        record.total\_sell = abs(sum(t.amount *for* t *in* record.transaction\_ids

*if* t.operation\_type == 'sell'))

        record.total\_commission = abs(sum(t.amount *for* t *in* record.transaction\_ids

*if* t.operation\_type == 'commission'))

*# Финансовый результат от сделок*

        record.financial\_result = record.total\_sell - record.total\_buy - record.total\_commission

*# Логика налогообложения*

        record.is\_iis = bool(record.account\_id *and* record.account\_id.account\_type == 'iis')

*if* record.is\_iis:

            record.tax\_rate = 0.0

            record.estimated\_tax = 0.0

*else*:

            record.tax\_rate = 13.0

            record.taxable\_base = record.financial\_result *if* record.financial\_result > 0 *else* 0

            record.estimated\_tax = record.taxable\_base \* (record.tax\_rate / 100.0)

*# Итоговый доход после налога*

        record.total\_income\_after\_tax = record.total\_income\_flow - record.estimated\_tax

Данный метод демонстрирует реализацию сложной бизнес-логики на уровне вычисляемых полей, что соответствует требованиям к отчетам с вычисляемыми итогами в СУБД.

**1.4. Анализ QWeb-шаблона отчета**

**QWeb-шаблон** investor.income\_report реализован в файле reports/income\_report.xml и демонстрирует следующие возможности:

**1.4.1. Многотабличный (составной) отчет**

Отчет объединяет данные из двух моделей:

- Основная модель: investor.income\_report (заголовок, параметры, итоги);

- Связанная модель: investor.transaction (детализация транзакций).

Объединение данных осуществляется через цикл t-foreach по полю transaction\_ids:

xml

<t t-foreach="doc.transaction\_ids" t-as="trans">

    <tr>

        <td class="text-center"><span t-esc="counter"/></td>

        <td class="text-center">

            <span t-esc="trans.transaction\_datetime.strftime('%Y-%m-%d %H:%M')"/>

        </td>

        <td>

            <span t-esc="trans.operation\_type == 'buy' and 'Покупка' or ..."/>

        </td>

        <td><span t-esc="trans.asset\_id.name if trans.asset\_id else '-'"/></td>

        <td class="text-center"><span t-esc="trans.quantity"/></td>

        <td class="text-end">

            <span t-esc="trans.amount" t-options='{"widget": "float", "precision": 2}'/>

        </td>

        <td class="text-center"><span t-esc="trans.currency"/></td>

        <td><span t-esc="trans.account\_id.name"/></td>

    </tr>

</t>

Данная конструкция обеспечивает реализацию требования к многотабличным отчетам: объединение данных из главной таблицы (отчет) и подчиненной таблицы (транзакции).

**1.4.2. Отображение вычисляемых полей**

**QWeb-шаблон** использует вычисляемые поля модели для вывода итоговых значений:

xml

<tr>

    <td>Финансовый результат (от сделок)</td>

    <td class="text-end fw-bold">

        <span t-esc="doc.financial\_result"

              t-options='{"widget": "float", "precision": 2}'/> RUB

    </td>

</tr>

<tr t-if="not doc.is\_iis">

    <td>Расчетный НДФЛ к уплате</td>

    <td class="text-end fw-bold">

        - <span t-esc="doc.estimated\_tax"

                t-options='{"widget": "float", "precision": 2}'/> RUB

    </td>

</tr>

<tr class="table-primary">

    <td class="fw-bold fs-5">Итоговый чистый доход</td>

    <td class="text-end fw-bold fs-5">

        <span t-esc="doc.total\_income\_after\_tax"

              t-options='{"widget": "float", "precision": 2}'/> RUB

    </td>

</tr>

Поля financial\_result, estimated\_tax и total\_income\_after\_tax рассчитываются в Python-коде модели через декоратор @api.depends, что обеспечивает автоматическое пересчитывание при изменении транзакций.

**1.4.3. Условное форматирование и логика**

**QWeb-шаблон** использует условные конструкции t-if для реализации бизнес-логики отображения:

**Пример 1: Условное отображение информации о счете:**

xml

<div t-if="doc.account\_id" class="col-12 mt-2">

    <p class="mb-1">

        <strong>Счёт:</strong>

        <span t-esc="doc.account\_id.name"/>

    </p>

</div>

**Пример 2: Специальный налоговый режим для ИИС:**

xml

<t t-if="doc.is\_iis">

    <div class="alert alert-info" role="alert">

        <h5 class="alert-heading">Особый налоговый режим (ИИС)</h5>

        <p class="mb-0">

            Для счета ИИС (Индивидуальный Инвестиционный Счет) применяется

            особый режим. При использовании вычета типа "Б",

            доход от операций по счету освобождается от НДФЛ.

            Расчетный налог равен 0.

        </p>

    </div>

</t>

**Пример 3: Условное отображение полей налогообложения:**

**xml**

<tr t-if="not doc.is\_iis">

    <td>Налогооблагаемая база (по фин. результату)</td>

    <td class="text-end">

        <span t-esc="doc.taxable\_base"

              t-options='{"widget": "float", "precision": 2}'/> RUB

    </td>

</tr>

<tr t-if="not doc.is\_iis" class="table-danger">

    <td class="fw-bold">Расчетный НДФЛ к уплате</td>

    <td class="text-end fw-bold">

        - <span t-esc="doc.estimated\_tax"

                t-options='{"widget": "float", "precision": 2}'/> RUB

    </td>

</tr>

Условные конструкции **t-if** и t-if="not doc.is\_iis" обеспечивают динамическое изменение структуры отчета в зависимости от типа счета, что соответствует требованиям к условной логике в отчетах СУБД.

**1.4.4. Форматирование данных**

**QWeb-шаблон** использует виджеты форматирования через параметр t-options:

- Форматирование дат: **t-options**='{"widget": "date"}'

- Форматирование чисел с точностью**: t-options**='{"widget": "float", "precision": 2}'

- Форматирование даты и времени: trans.transaction\_datetime.strftime('%Y-%m-%d %H:%M')

Все указанные механизмы обеспечивают соответствие QWeb-отчета требованиям к печатным отчетам в СУБД: многотабличность, вычисляемые поля, условная логика и форматирование данных.

**Раздел 2. Реализация аналитических отчетов (Pivot и Graph)**

Аналитические отчеты в Odoo реализованы через встроенные представления **Pivot** View (сводные таблицы) и **Graph** View (диаграммы). Данные представления обеспечивают интерактивный анализ данных с возможностью группировки, агрегации и визуализации без необходимости написания дополнительного кода.

**2.1. Pivot View (Сводный отчет)**

**Pivot** View в **Odoo** является аналогом сводных таблиц (Pivot Tables) в Microsoft Excel и Access. Представление позволяет группировать данные по различным измерениям и агрегировать числовые значения по заданным функциям (сумма, среднее, количество и др.).

Реализация Pivot View для модели investor.transaction:

xml

<record id="view\_transaction\_pivot" model="ir.ui.view">

    <field name="name">investor.transaction.pivot</field>

    <field name="model">investor.transaction</field>

    <field name="arch" type="xml">

        <pivot string="Анализ Транзакций">

            <field name="operation\_type" type="row"/>

            <field name="amount" type="measure"/>

        </pivot>

    </field>

</record>

Анализ структуры:

- <pivot> — корневой элемент сводного представления;

- <field name="operation\_type" type="row"/> — поле группировки по строкам (тип операции: покупка, продажа, зачисление, списание, комиссия);

- <field name="amount" type="measure"/> — поле для агрегации (мера): по умолчанию выполняется суммирование значений amount для каждой группы.

Данное представление позволяет получить сводную таблицу вида:

Дополнительные возможности Pivot View:

В интерфейсе Odoo пользователь может интерактивно:

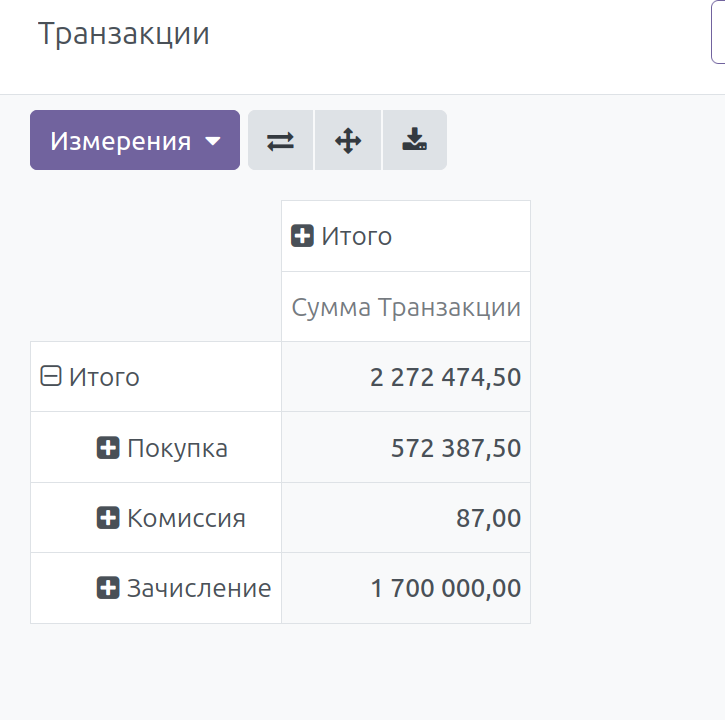
- Добавлять дополнительные измерения для группировки (например, по дате, счету, инвестору);

- Изменять функцию агрегации (сумма, среднее, количество, минимум, максимум);

- Фильтровать данные;

- Экспортировать сводную таблицу в Excel.

Таким образом, Pivot View полностью покрывает требования к сводным отчетам и отчетам с группировкой в СУБД Access, обеспечивая при этом интерактивность и гибкость анализа.



**2.2. Graph View (Отчет-диаграмма)**

Graph View в Odoo обеспечивает визуализацию данных в виде различных типов диаграмм: столбчатая (bar), линейная (line), круговая (pie), область (area).

Реализация Graph View для модели investor.transaction:

xml

<record id="view\_transaction\_graph" model="ir.ui.view">

    <field name="name">investor.transaction.graph</field>

    <field name="model">investor.transaction</field>

    <field name="arch" type="xml">

        <graph string="Анализ Транзакций" type="pie">

            <field name="operation\_type"/>

            <field name="amount" type="measure"/>

        </graph>

    </field>

</record>

Анализ структуры:

- <graph> — корневой элемент графика;

- type="pie" — тип диаграммы (круговая);

- <field name="operation\_type"/> — поле для группировки (сегменты круговой диаграммы);

- <field name="amount" type="measure"/> — поле для агрегации (размер сегментов пропорционален сумме).

Данное представление визуализирует распределение сумм транзакций по типам операций в виде круговой диаграммы, где каждый сегмент соответствует одному типу операции, а размер сегмента пропорционален общей сумме транзакций данного типа.

Альтернативные типы графиков:

В Odoo доступны следующие типы диаграмм:

- type="bar" — столбчатая диаграмма (гистограмма);

- type="line" — линейная диаграмма (для временных рядов);

- type="pie" — круговая диаграмма (для долей);

- type="area" — диаграмма с заливкой (для накопленных значений).

Для анализа транзакций также может быть использована столбчатая диаграмма:

xml

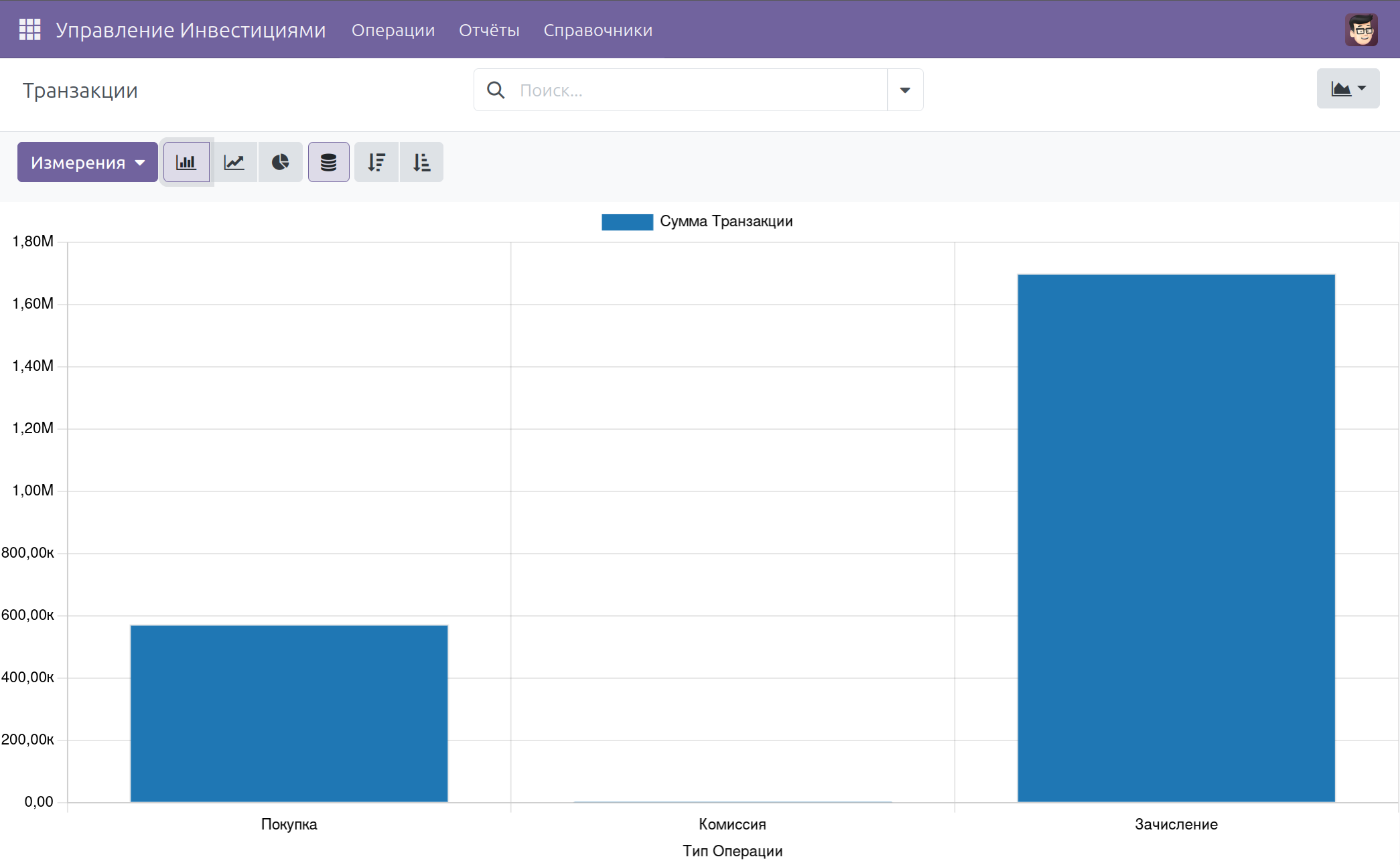
<graph string="Анализ Транзакций" type="bar">

    <field name="operation\_type"/>

    <field name="amount" type="measure"/>

</graph>

Данная диаграмма отобразит суммы транзакций по типам операций в виде столбцов различной высоты.



Graph View обеспечивает визуализацию данных, что соответствует требованиям к отчетам-диаграммам в СУБД Access, предоставляя при этом возможность интерактивного переключения между типами диаграмм и измерениями.

**Раздел 3. Обзор типов отчетов и экспорт**

В таблице представлено сопоставление типов отчетов, традиционно реализуемых в СУБД Microsoft Access, с соответствующими механизмами Odoo, реализованными в модуле «Инвестиционный модуль».

**Соответствие типов отчетов Access и механизмов Odoo**

| **№** | **Требование Access** | **Аналог в Odoo (Механизм)** | **Модель-источник** | **Доказательство реализации (Фрагмент кода/Функционал)** |
| --- | --- | --- | --- | --- |
| **1** | **Однотабличный отчет** | Tree View (Список) | investor.transaction, investor.account, investor.asset, investor.investor, investor.broker | Отображение всех полей таблицы с возможностью сортировки, фильтрации и группировки. Реализовано в файле views/transaction\_views.xml: <tree string="Транзакции"> с полями transaction\_datetime, name, account\_id, operation\_type, amount, currency. |
| **2** | **Составной/ Многотабличный отчет** | QWeb-шаблон (PDF/HTML) | investor.income\_report (главная) + investor.transaction (подчиненная) | QWeb-шаблон reports/income\_report.xml использует цикл <t t-foreach="doc.transaction\_ids" t-as="trans"> для объединения данных из двух моделей. Отчет содержит заголовок с информацией об инвесторе (из главной модели) и детализацию транзакций (подчиненная таблица). |
| **3** | **Отчет с вычисляемыми полями** | QWeb-шаблон + вычисляемые поля модели | investor.income\_report | Метод \_compute\_totals в модели income\_report.py рассчитывает итоговые значения: total\_buy, total\_sell, financial\_result, estimated\_tax, total\_income\_after\_tax. Поля отображаются в QWeb-шаблоне через конструкции вида <span t-esc="doc.financial\_result" />. |
| **4** | **Сводный отчет / Отчет с группировкой** | Pivot View | investor.transaction | Представление view\_transaction\_pivot в файле views/transaction\_views.xml определяет группировку по полю operation\_type (тип операции) и агрегацию по полю amount (сумма). Позволяет получить сводную таблицу сумм транзакций по типам операций. |
| **5** | **Отчет с параметром** | Wizard (TransientModel) + QWeb | investor.income\_report.wizard → investor.income\_report | Wizard (income\_report\_wizard.py) содержит параметры: investor\_id, account\_id, date\_from, date\_to. Метод action\_generate\_report создает запись с заданными параметрами и запускает QWeb-отчет. |
| **6** | **Отчет-диаграмма** | Graph View | investor.transaction | Представление view\_transaction\_graph в файле views/transaction\_views.xml определяет круговую диаграмму (type="pie"), группирующую данные по operation\_type и агрегирующую значения amount. |
| **7** | **Экспорт в Excel/XML** | Встроенный функционал Odoo | Любой Tree/Pivot/Graph View | В любом представлении списка (Tree View) доступна функция экспорта через меню «Действие → Экспорт». Поддерживаются форматы: Excel (XLSX), CSV, XML. |

**3.1. Процесс экспорта данных**

Экспорт данных в Odoo выполняется через стандартный интерфейс:

1. Пользователь открывает список записей (Tree View) или аналитическое представление (Pivot/Graph).

2. При необходимости применяет фильтры для ограничения набора данных.

3. Выбирает пункт меню «Действие → Экспорт».

4. В диалоговом окне выбирает поля для экспорта и формат файла (Excel, CSV, XML).

5. Система генерирует файл и предоставляет его для скачивания.

Данный механизм обеспечивает экспорт данных из любых представлений модуля, что соответствует требованиям к экспорту отчетов в СУБД Access.

**Вывод**

В ходе выполнения лабораторной работы были изучены и применены на практике механизмы генерации отчетов в среде Odoo с использованием технологий QWeb, Pivot и Wizard. Реализован полный комплекс отчетности для модуля «Инвестиционный модуль», включающий печатные отчеты, аналитические представления и мастеры параметризации.

Основные результаты работы:

1. Реализован параметризованный печатный отчет «Отчет о доходах инвестора» на базе QWeb-шаблона и Wizard (TransientModel). Отчет демонстрирует реализацию многотабличного отчета с вычисляемыми полями, условной логикой и форматированием данных.

2. Реализованы аналитические представления Pivot View и Graph View для модели транзакций, обеспечивающие интерактивный анализ данных с группировкой и агрегацией без необходимости написания дополнительного кода.

3. Проведено сопоставление механизмов Odoo с требованиями к отчетам в СУБД Access. Установлено, что все классические типы отчетов (однотабличные, многотабличные, параметризованные, сводные, диаграммы) имеют полные аналоги в Odoo, при этом функциональность Odoo превосходит базовые возможности Access за счет интеграции с веб-интерфейсом, ORM и возможностями интерактивного анализа.

Механизмы генерации отчетов в Odoo обеспечивают:

- Гибкость параметризации через Wizard (TransientModel);

- Сложную бизнес-логику через вычисляемые поля и методы моделей;

- Визуализацию данных через Pivot и Graph View;

- Интеграцию с веб-интерфейсом через QWeb-шаблоны;

- Экспорт данных в стандартные форматы (Excel, CSV, XML).

Разработанный модуль отчетности полностью соответствует требованиям лабораторной работы №7 и демонстрирует эффективное использование современных механизмов Odoo для решения задач аналитической отчетности в информационных системах.